SOFTWARE SPECIFICATION
Syntax and Processing of XML-Based Documents in the Context of SWIM-Enabled Services

Comments, suggestions, or questions on this document should be addressed to:

Federal Aviation Administration
SWIM Program Office, AJM-316
800 Independence Avenue, SW.
Washington, DC 20591
mark.kaplun@faa.gov

SIGNATURE PAGE

6-22-15

Jim Robb — Date

SWIM Program Manager

Version	Description of Change	Author	Date
1.0	Initial draft	Cam Dang, Leonid Felikson, Peadar Harmon,	05/31/2014
1.0	Security requirements	Kelly Mesveskas	07/24/2014
1.0	Versioning and documenting considerations	Mark Kaplun	05/26/2015

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 SCOPE

1.1 Scope

This specification defines requirements and guidelines for the preparation and publishing of XML-based documents intended for use in the context of FAA's System Wide Information Management (SWIM) program. This specification places conditions on a set of non-proprietary industry-wide specifications to improve interoperability, security, reusability, and maintainability of SWIM-enabled services.

This specification applies only to the XML documents that support services which are registered or intended to be registered in the NAS Service Registry/Repository (NSRR).

This specification has been prepared in accordance with FAA-STD-067 [STD-067].

1.2 Background

The Extensible Markup Language (XML), abbreviated XML, was originally published as a World Wide Web Consortium (W3C) recommendation in 1998, and was revised in a second edition in 2000. XML is a simple and very flexible text format designed to enable the definition, transmission, validation, and interpretation of data between applications created for any platform. As a platform-independent meta-language, XML is an integral part of nearly all Service-Oriented Architecture (SOA) efforts, including a major implementation of SOA in FAA's SWIM program.

SWIM utilizes XML in three primary ways:

1. As a meta-language for expressing the content of messages exchanged in the execution of SWIM SOA services
2. For defining the structure, content and constraints of XML-based messages
3. For describing types, messages, interfaces and their concrete protocols and data format bindings, and the network access points associated with Web services

The SWIM Governance Policies [SWIM GP] define a set of artifacts associated with a SOA service that should be cataloged in and made discoverable by the NSRR. The list of these artifacts usually includes a s*chema* that is used to validate XML messages produced by the service, an *XML-based service definition document* that describes the behavior of the service, and examples of *messages*. These artifacts are subsequently used for test and deployment in the NAS Enterprise Messaging Service (NEMS).

Every type of XML-based document stored in the NSRR is designed according to one or more non-proprietary (open) specifications. These specifications (also referred to as standards) are established to support technical interoperability among components exchanging XML-serialized data.

However, it should be noted that because XML was designed as a very adaptable and open-ended language (all XML specifications inherently follow this trend), there could be a variety of ways of accomplishing a particular goal. Electing the right mechanism when designing an XML document may

have significant ramifications regarding interoperability, maintainability, and susceptibility to security threats for components that participate in XML-based data exchange.

The requirements contained herein provide a set of instructions for designing XML documents that are interoperable and secure in the context of the SWIM environment.

1.3 Entity Type Description

This specification defines a data object called *XML Document* and describes the behavior of computer programs which support processing it in the SWIM environment.

In the context of this specification, *XML Document* is sub-classed into three entities with respect to areas of application:

- *Message* - an XML Document containing data that is passed between components.
- *Schema* - an XML Document that defines structure and constraints for the types of data to be validly used by other XML Documents.
- *Service Definition* - a formal description of types, messages, interfaces and their concrete protocols and data format bindings, and the network access points associated with Web services [WS-I]. An example of this is a WSDL (Web Service Description Language) document.

Figure 1 illustrates relationships between these entities.

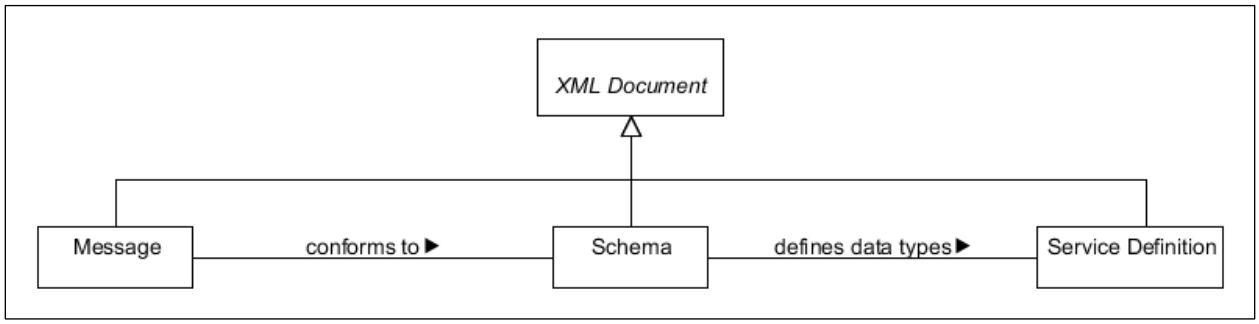

FIGURE 1 XML Document - conceptual model

The requirements for designing each of the entities are presented in sections 3.3.1, 3.3.2, and 3.3.3 respectively.

1.4 System Software Overview

According to the SWIM Governance Policies [SWIM GP], the service providers are responsible for developing and defining service description artifacts and uploading them to the NSRR. The NSRR maintains service description metadata and associated artifacts throughout the service's lifecycle. The NSRR processes the artifacts in accordance with the NSRR functionality and processing logic, and then it makes those artifacts, including WSDL documents and XML schemas, available to NSRR users for future

use. When the service is ready to be deployed and/or tested, copies of appropriate artifacts are uploaded to NEMS to support data exchange.

1.5 Intended Audience

This specification is intended for use by application architects and software developers. While the document contains much explanatory material, it assumes the user has a basic familiarity with XML syntax, XML schemas, and WSDL specification.

1.6 Typographical Conventions

Page headers, page numbers, figure and table captions, and other formatting are in accordance with FAA-STD-067 [STD-067].

```
Examples of code and relevant artifacts used in this document are
presented in constant width font and in shaded paragraphs.
```

2 APPLICABLE DOCUMENTS

2.1 Government Documents

2.1.1 Specifications, Standards, and Handbooks

The following specifications, standards, and handbooks form a part of this document to the extent specified herein. Unless otherwise specified, the issues of these documents are those cited in the solicitation or contract.

[STD-067] FAA Standard Practice, Preparation of Specifications, FAA-STD-067, December 4, 2009, https://sowgen.faa.gov/docs/FAA-STD-067.pdf

[SWIM GP] System-Wide Information Management (SWIM) Governance Policies, Version 2.0, March 2014, http://www.faa.gov/nextgen/programs/swim/governance/standards/media/Governance-Policies-v20.html

[SWIM CV] SWIM Controlled Vocabulary (CV), http://www.faa.gov/nextgen/programs/swim/vocabulary/

2.1.2 Other Government Documents, Drawings, and Publications

[NIST 800-95] Guide to Secure Web Services, NIST Special Publication 800-95, August 2007. http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf

2.2 Non-Government Publications

[WSDLv1.1] Web Services Description Language (WSDL) 1.1, 15 March 2001 http://www.w3.org/TR/wsdl

[XML-W3C] Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008, http://www.w3.org/TR/2008/REC-xml-20081126/

[XMLSCHEMA-W3C] XML Schema Part 1: Structures Second Edition; W3C Recommendation; 28 October 2004, http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Datatypes Second Edition; W3C Recommendation; 28 October 2004, http://www.w3.org/TR/xmlschema-2/

[XMLCan-W3C] W3C Canonical XML, Version 1.0, W3C Recommendation 15 March 2001, http://www.w3.org/TR/xml-c14n

[XML-NS-1.0] Namespaces in XML 1.0 (Third Edition),W3C Recommendation 8 December 2009 http://www.w3.org/TR/REC-xml-names/

[RFC-2119] RFC 2119, *Key words for Use in RFCs to Indicate Requirement Levels*, Network Working Group, March 1997. http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC-3936] RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, Network Working Group, January 2005 http://www.rfc-editor.org/rfc/rfc3986.txt

[WS-I] WS-I Basic Profile Version 2.0, Final Material, 2010-11-09 *http://ws-i.org/profiles/basicprofile-2.0-2010-11-09.html*

2.3 Order of Precedence

In the event of a conflict between the text of this document and the references cited herein, the text of this document takes precedence. Nothing in this document, however, supersedes applicable laws and regulations unless a specific exemption has been obtained.

3 REQUIREMENTS

3.1 Key Words

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", “MAY", and "OPTIONAL" in this specification are to be interpreted as described in RFC 2119 [RFC-2119]. These key words are capitalized when used to unambiguously specify requirements. When these words are not capitalized, they are meant in their natural-language sense.

3.2 General Requirements

This section specifies requirements that are common to all types of XML documents described in this specification.

a. All XML documents SHALL be developed in compliance with XML version 1.0 [XML-W3C].

b. All XML documents SHALL use UTF-8 encoding.

c. All elements in an XML document SHALL have qualified names.

Explanation: This specification prohibits usage of default namespace. See sections 3.3.1.1 and 3.3.2.1 for examples that illustrate this requirement.

d. To signal a new version of an XML document, new namespaces containing version information SHOULD be declared.

Explanation: XML specifications allow usage of the optional `version` attribute; however, an XML validation tool is not required to validate an instance of an XML document using this attribute, i.e., `version` attribute is provided purely for documentation purposes and is not enforceable by a validator. Conversely, using a new namespace identifier for a new version of an XML document does not require additional custom processing by a validator.

e. XML document namespace declarations pursuant to the “Namespaces in XML” specification [XML-NS-1.0] MAY be defined as a URL using the HTTP scheme.

f. The values of the namespace declarations in XML documents SHOULD be dereferenceable.

g. Each namespace defined as a URL SHOULD resolve to a human or machine processable document that directly or indirectly defines the intended semantics of the document.

Explanation: The requirements ‘e’ through ‘g’ are *forward compatible*, i.e., they are expected to support practices used in most of today’s industry XML documents (see for example AIXM and W3C schemas) and semantic technologies that SWIM plans to embrace in the future.

3.3 Detailed Requirements

3.3.1 XML Message Document Requirements

This section addresses XML syntax issues relevant to the design of XML message documents. Although the main purpose of an XML message document is to convey data between SOA components, in the SWIM/NSRR environment an instantiation of an XML message document (called an "XML instance document") can also be uploaded to the NSRR to provide a sample that should help a service consumer to develop and/or test a consumer agent.

3.3.1.1 Syntax Requirements

a. Every XML message document SHALL conform to a particular schema.

b. The start tag of a root element in an XML instance document SHOULD contain the standard "XML Schema Instance Namespace" declaration `http://www.w3.org/2001/XMLSchema-instance.`

c. The start tag of a root element in an XML message document SHALL contain a declaration of the target namespace for the schema that defines the instance document.

d. The start tag of a root element in an XML instance document SHALL contain a non-empty `xsi:schemaLocation` attribute indicating the location and namespace of the schema that the document conforms to.

e. When the defining schema is submitted to the NSRR together with the XML message document, the part of the `xsi:schemaLocation` attribute that provides the physical location of the schema MAY be a relative URI.

The following example illustrates two applications of XML instance syntax requirements. It should be noted that both of the cases will be recognized by most validators as valid and no errors will be produced.

INCORRECT

```
…
<?xml version="1.0" encoding="UTF-8"?>
<FlightPlan
   flightRule="I"
   numberOfAircraft="1"
   filingTime="2001-12-17T09:30:47Z" >
      <FlightPlanId nil="true" />
      <Originator airmanId="215336745">
            <Name>John Doe</Name>
      </Originator>
…
```

CORRECT

```
…
<?xml version="1.0" encoding="UTF-8"?>
<fps:FlightPlan
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:fps="http://faa.gov/fps"
      xsi:schemaLocation="http://faa.gov/fps FlightPlan.xsd"
      flightRule="I" numberOfAircraft="1"
      filingTime="2001-12-17T09:30:47Z">
        <fps:FlightPlanId xsi:nil="true"/>
         <fps:Originator airmanId="215336745">
            <fps:Name>John Doe</fps:Name>
        </fps:Originator>
…
```

3.3.1.2 Security Requirements

a. The XML message document SHALL NOT define inline XML schemas.

Explanation: Inline XML schemas are XML schemas included inside XML documents. An XML message document containing inline XML schemas could allow the message author to subvert the validation of the XML message document by the recipient.

b. The XML message document SHALL NOT include `xlink` references.

Explanation: The use of `xlink` introduces multiple security vulnerabilities. A malicious consumer could insert into an otherwise valid message a modified `xlink` value that contains an exploit.

The following is an example of incorrect usage of `xlink` in an XML document:

INCORRECT

```
…
<aw:observedProperty xlink:href://http://www.eurocontrol.int/
/1.1/xw.owl#AirspaceWx"/>
…
```

c. XML escape characters SHOULD NOT be used in a `string` data type and in data types derived from the `string` data type.

Explanation: This requirement asserts that if XML escape characters are not absolutely necessary in the data, the schema should prohibit their use in a `string` data type and data types derived from

the `string` data type. A valid XML schema could introduce hidden security vulnerability. Even when an XML schema establishes the validity of an XML document, there is a common exploit attempt that uses a technique called "hidden markup" that may go undetected. Escape characters are sometimes used in an XML document to represent characters that cannot be included as the content of an element.

The prime examples are the angle brackets "<" and ">". An XML document cannot have a string element such as "`<reason>Because Range < 1000 yards</reason>`" since the second "<" would be misinterpreted by the XML parser as the beginning of another metadata tag. Instead one would use "<" (an escape character sequence) in place of "<" like this: **"**`<reason>Because Range < 1000 yards</reason>`**"** and the XML parser would convert "<" to "<" before it is presented to the receiving application. Unfortunately the escape character sequence could be used to introduce malicious code that when sent on to the application (a browser for example) could be executed. The malicious code may go undetected by a human looking at the XML document or by the schema itself unless the schema does not allow escape character sequences. Therefore it is recommended to generate only schemas that do not allow escape character sequences. Using CDATA to define a section of unparsed character data is another way to prevent characters from undergoing schema validation.

3.3.2 XML Schema Requirements

The section specifies constraints which should be applied during development of application-specific XML schemas. These constraints and requirements are not applicable to the XML schema models developed by external (non-FAA) organizations, which from the perspective of this specification are immutable.

3.3.2.1 Syntax Requirements

a. All XML schemas SHALL conform to the structure and constraints specified in the XML Schema 1.0 Recommendation [XMLSCHEMA-W3C].

b. The start tag of the schema root element `<xsd:schema>` SHALL contain a declaration of the namespace `http://www.w3.org/2001/XMLSchema`.

c. For all schema elements defined in the XML Schema namespace, the qualifier "`xsd`" SHALL be used.

d. Every XML schema SHALL specify a non-empty `targetNamespace` attribute on the `<xsd:schema>` element.

e. The value for the attribute `elementFormDefault` of the `<xsd:schema>` element SHALL be "`qualified`".

Explanation: This requirement reinforces the general requirement 3.2.c.

f. The value for the attribute `attributeFormDefault` of the `<xsd:schema>` element SHALL be "`unqualified`".

Note: because "`unqualified`" is the default value for the attribute `attributeFormDefault`, the attribute `attributeFormDefault` can be omitted with the same result.

INCORRECT

```
…
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                 xmlns="http://faa.gov/fps"
                 targetNamespace="http://faa.gov/fps"
                 elementFormDefault="unqualified" >
  <xsd:element name="FlightPlan" type="FlightPlanType"/>
  <xsd:element name="Aircraft" type="AircraftType"/>
  <xsd:element name="Route" type="RouteType"/>
…
```

CORRECT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                 xmlns:fps="http://faa.gov/fps"
                 targetNamespace="http://faa.gov/fps"
                 elementFormDefault="qualified"
                 attributeFormDefault="unqualified"
                 version="1.0">
  <xsd:element name="FlightPlan" type="fps:FlightPlanType"/>
  <xsd:element name="FlightPlanId" type="fps:FlightPlanIdType"/>
  <xsd:element name="Originator" type="fps:OriginatorType"/>
…
```

3.3.2.2 Security Requirements

"Because XML Schemas can rigidly define the types of data and format of XML elements, they can be used to prevent the Web service from processing invalid requests."[NIST 800-95] "An improperly configured XML Parser is susceptible to several attacks: External references to other XML documents or XML schemas can be used to bypass XML validators."[NIST 800-95] The XML schema is very important to the proper configuration of a validating XML parser and development of an XML schema with security in mind can significantly increase the chances of catching an invalid or malicious XML message before it gets to a consumer application.

a. XML schemas SHALL constrain a `QName` data type via the `pattern` attribute and be based on the requirements of the data.

Explanation: The `xsd:QName` (Qualified Name) data type is derived from the `xsd:string` data type and should also be constrained based on the condition for the `xsd:string` data type. However, the `xsd:maxLength` attribute is ignored for the `QName` data type (see http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/#rf-maxLength). Instead, the `xsd:pattern` attribute must be used to constrain the characters and length of a `QName` value.

CORRECT

```
…
<xsd:element name="CountryCode">
  <xsd:simpleType>
    <xsd:restriction base="xsd:Qname">
      <xsd:pattern value="[a-zA-Z:]{1,10}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
…
```

b. All numeric data types SHALL have minimum and maximum values specified in the XML schema according to the legal ranges of values for the data.

Explanation: Unexpected but allowable data ranges introduce a potential security exploit. The primitive data types `xsd:float` and `xsd:double` allow values of NaN (not a number), INF (infinity), and –INF (negative infinity). It is possible that there are not any provisions for these three values, which would cause an exception in the application. If not handled, these unexpected values would likely cause application termination. The solution is to provide minimum values and maximum values for `xsd:float` and `xsd:double` data types via usage of the following attributes: `xsd:minExclusive, xsd:maxExclusive, xsd:minInclusive` and `xsd:maxInclusive`.

c. NaN, INF, and –INF values SHOULD NOT be used for numeric data types.

CORRECT

```
…
<xsd:element name="distance">
  <xsd:simpleType>
    <xsd:restriction base="xsd:float">
      <xsd:minExclusive value="0.0"/>
      <xsd:maxExclusive value="1000000.0"/>
      <xsd:pattern value="[^0].*"/>
```

```
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
…
```

d. Each element in an XML schema SHALL have a finite upper limit for the number of occurrences.

Explanation: Allowing usage of the attribute `maxOccurs="Unbounded"` introduces a potential security exploit. In a computer, nothing is unbounded. The solution is to specify a legitimate number for the maximum occurrences of an element, even a large number if applicable, such as `maxOccurs="10000"`. This finite upper limit allows a consumer to expect and allocate enough space to hold up to 10000 entries of the data element.

INCORRECT

```
…
<xsd:element name="dataPoint" type="xsd:float" maxOccurs="Unbounded"/>
…
```

CORRECT

```
…
<xsd:element name="dataPoint" type="xsd:float" maxOccurs="10000"/>
…
```

e. Attributes in an XML schema SHALL contain only alphanumeric characters or be implemented as child elements.

Explanation: The use of non-alphanumeric characters (i.e., not [A-Z0-9a-z]) in attributes in XML documents introduces security vulnerability in `Xpath`. An attacker may be able to exploit dynamic `Xpath` statements implemented by the receiver of an XML document. This is true if certain special characters are allowed in attributes by the schema such as ("*^';&><</), which are all characters that can be used for various injection attacks. If an attribute is not restricted to prohibit all special characters that are part of the `Xpath` expression syntax, then the attacker could provide a value for an attribute that would result in the creation of a highly complex expression leading to a Denial of Service (DoS) attack. This is similar to a Structured Query Language (SQL) injection exploit. If a field's content could legitimately contain non-alphanumeric characters, that field should be implemented as a child element instead of an attribute.

INCORRECT

```
…
<xsd:element name="CountryName">
  <xsd:attribute name="Value" type="xsd:string"/>
</xsd:element>
…
```

CORRECT

```
…
<xsd:element name="CountryName">
  <xsd:attribute name="Value">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="[a-zA-Z:]{1,30}"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
<xsd:element>
…
```

f. The `processContents` attribute SHALL only be defined as "strict" in an XML schema.

Explanation: The `xsd:any` and `xsd:anyAttribute` elements enable the extension of an XML document with elements and attributes not specified by the schema. These elements provide flexibility in extending a schema, but they can be exploited if not defined with proper limitation. The `processContents` attribute of these elements is what exposes security vulnerability that can weaken the XML schema validation process. The `processContents` attribute can take on any one of three values: "strict" meaning the XML processor must obtain the schema and validate the element; "lax" meaning the XML processor must try to obtain the schema but if the schema cannot be obtained then no error will occur; and "skip" meaning the XML processor does not attempt to validate any elements. Without enforcement of schema validation, an attacker could extend an XML document with additional elements that are not legal. This is a major security issue that permits the potential of many types of cyber-attacks.

INCORRECT

```
…
<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="FirstName" type="xsd:string"/>
    <xsd:element name="LastName" type="xsd:string"/>
```

```
    <xsd:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="customerID" type="xsd:integer"/>
  <xsd:anyAttribute namespace="##targetNamespace"
processContents="lax"/>
</xsd:complexType>
…
```

CORRECT

```
…
<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="FirstName" type="xsd:string">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd: maxLength value="50"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="LastName" type="xsd:string">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd: maxLength value="50"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:element>
    <xsd:any namespace="##targetNamespace" processContents="strict"
minOccurs="0" maxOccurs="1000"/>
  </xsd:sequence>
  <xsd:attribute name="customerID" type="xsd:integer"/>
  <xsd:anyAttribute namespace="##targetNamespace"
processContents="strict"/>
</xsd:complexType>
…
```

g. The use of default values for attributes SHALL NOT be permitted in XML schemas.

Explanation: Relying on default attribute values introduces security vulnerability. For attributes that are truly optional, this is not a problem. However, default attributes are frequently used for attributes that are not optional. So, if the XML document is not validated against a schema and the

attribute is not truly optional, the receiver may assume that the attribute's default value exists even when it does not. Thus, the receiver may not be able to continue processing.

INCORRECT

```
…
<xsd:attribute name="Language" type="xsd:string" default="EN"/>
…
```

CORRECT

```
…
<xsd:attribute name="Language"/>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-zA-Z:]{1,10}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
…
```

3.3.2.3 Documentation Requirements

XML Schema-compliant documents use appropriate annotation mechanisms to convey additional information about elements.

a. XML schema documents SHOULD use the <xsd:annotation> element to convey additional information about elements.

b. The schema's root element <xsd:schema> SHALL be followed by the <xsd:annotation> element with a single <xsd:documentation> child element containing information about the schema's purpose and the organizational entity that owns the schema.

c. All of the schema's global elements SHALL be annotated using the <xsd:annotation> element with a single <xsd:documentation> child element.

The following is an example of documenting an XML service definition document:

INCORRECT

```
<!--
  This schema declares XML elements for defining a Flight Plan
  transmitted by FlightPlanService.
 -->
```

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

CORRECT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema” ...>
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
       This schema declares XML elements for defining a Flight Plan
      transmitted by FlightPlanService.
     </xsd:documentation>
  </xsd:annotation>
…
```

3.3.3 XML Service Definition Requirements

This section defines requirements for XML service definition documents. Because XML-based service definitions are almost exclusively developed in accordance with the WSDL specification, the document is often referred to as a “WSDL file” or simply a “WSDL”.

3.3.3.1 Syntax Requirements

a. All XML service definition documents SHALL conform to the structure and constraints specified in the WSDL 1.1 Specification [WSDLv1.1].

Explanation: Although W3C has released version 2.0 of the WSDL Specification (which is expected to supersede version 1.1), currently WSDL version 2.0 is not supported by NEMS, therefore version 1.1 is mandated by this specification.

b. The start tag of the WSDL root element `<wsdl:definition>` SHALL explicitly contain the namespace declaration `http://schemas.xmlsoap.org/wsdl/`.

c. The elements defined in the WSDL namespace SHALL use the qualifier ‘wsdl’ throughout the document.

INCORRECT

```
…
<definitions name="Flight Plan service definition"
   targetNamespace="http://faa.gov/fps"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
…
<message name="FileFlightPlanRequest">
  <part name="fp" element="FlightPlan"/>
</message>
```

…

CORRECT

```
…
<definitions name="Flight Plan service definition"
   targetNamespace="http://faa.gov/fps"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
…
<wsdl:message name="FileFlightPlanRequest">
  <wsdl:part name="fp" element="FlightPlan"/>
</wsdl:message>
…
```

d. Every service definition document SHALL specify a non-empty `targetNamespace` attribute of the `<wsdl:definition>` element.

e. The `targetNamespace` attribute of the `<wsdl:definitions>` element of a WSDL file SHALL match the target namespace in the namespace declaration.

3.3.3.2 Documentation Requirements

WSDL-compliant service definition documents use appropriate annotation mechanisms to convey additional information about elements.

a. Every service definition document SHOULD use the `<wsdl:documentation>` element to convey human-readable information about elements defined in the `'wsdl'` namespace.

Explanation: Although usage of the xsd:annotation/xsd:documentation construct will be valid from an XML parser perspective, the NSRR parser will not be able to read the content. XML type comments also cannot be processed by the NSRR parser.

b. In a service definition document, the `<wsdl:documentation>` element SHOULD be present as the first child element of the `'wsdl:'` document being described.

Explanation: Although the NSRR parser can process the `<wsdl:documentation>` element correctly regardless of the order of child elements, having it as the first child element makes it more readable by a human.

c. The `<wsdl:documentation>` element SHALL be a simple element, that is, an element that contains only text and has no child elements or attributes.

Explanation: the NSRR parser may not be able to read a parent-child tree within a complex `<wsdl:documentation>` element, even though the element will be valid from an XML parser perspective.

The following is an example of documenting an XML service definition document:

INCORRECT

```
…
<!-- Used by a service consumer to submit (file) a flight plan. -->
<wsdl:message name="FileFlightPlanRequest">
  <wsdl:part name="fp" element="fps:FlightPlan"/>
</wsdl:message>
…
```

INCORRECT

```
…
</wsdl:message>
 <xsd:annotation>
  <xsd:documentation>
  Used by a service consumer to submit (file) a flight plan.
  </xsd:documentation>
 </xsd:annotation>
</wsdl:message>
…
```

CORRECT

```
…
<wsdl:message name="FileFlightPlanRequest">
<wsdl:documentation>
  Used by a service consumer to submit (file) a flight plan.
</wsdl:documentation>
<wsdl:part name="fp" element="fps:FlightPlan"/>
…
```

4 QUALITY ASSURANCE PROVISIONS

All XML-based documents that support services which are registered or intended to be registered in the NSRR SHOULD support all requirements as described in the requirements section of this specification.

XML-based documents presented for uploading to the NSRR SHALL be subject to verification that they conform to the requirements prescribed by this specification.

Verification SHALL be performed by inspection and test according to the following table.

TABLE I Requirements Verification Matrix

Requirement	Verification Method
3.2.a	Inspection
3.2.b	Inspection
3.2.c	Inspection
3.2.d	Inspection
3.2.e	Inspection
3.2.f	Inspection
3.2.g	Inspection
3.3.1.1.a	Inspection
3.3.1.1.b	Inspection
3.3.1.1.c	Inspection
3.3.1.1.d	Inspection
3.3.1.1.e	Inspection
3.3.1.1.f	Inspection
3.3.1.2.a	Inspection
3.3.1.2.b	Inspection
3.3.1.2.c	Inspection
3.3.2.1.a	Inspection
3.3.2.1.b	Inspection

Requirement	Verification Method
3.3.2.1.c	Inspection
3.3.2.1.d	Inspection
3.3.2.1.e	Inspection
3.3.2.1.f	Inspection
3.3.2.2.a	Inspection
3.3.2.2.b	Inspection
3.3.2.2.c	Inspection
3.3.2.2.d	Inspection
3.3.2.2.e	Inspection
3.3.2.2.f	Inspection
3.3.2.2.g	Inspection
3.3.2.3.a	Inspection
3.3.2.3.b	Inspection
3.3.2.3.c	Inspection
3.3.3.1.a	Validation by the NSRR
3.3.3.1.b	Validation by the NSRR
3.3.3.1.c	Validation by the NSRR
3.3.3.1.d	Validation by the NSRR
3.3.3.1.e	Validation by the NSRR
3.3.3.2.a	Validation by the NSRR

Requirement	Verification Method
3.3.3.2.b	Inspection
3.3.3.2.c	Inspection

5 PREPARATION FOR DELIVERY

All WSDL documents are published on the NAS Services Registry/ Repository (NSRR) in .zip format in accordance with the SWIM Governance Policies document [SWIM GP].

This specification enforces the concept of a "self-contained" package that includes the WSDL document and all associated XML schemas to be uploaded to the NSRR at the time of submitting the WSDL to the NSRR. "Self-contained" package means that WSDL file(s) must be packaged together with all XML schemas that the WSDL file(s) refer(s) to, without use of any reference to external XML schema file(s).

a. A root .wsdl file and the referenced .wsdl and .xsd files SHALL be bundled into a single ZIP archive.

b. The ZIP archive SHALL contain only one root .wsdl file.

c. The non-root .wsdl and .xsd files SHALL be in the same or child directories of the root .wsdl file.

d. The ZIP archive that is used to upload the service description artifacts, i.e., WSDL file(s) and XML schema(s), SHALL contain all XML schemas that are referred to by the WSDL document(s) for the service description.

e. All XML documents to be uploaded to the NSRR SHALL be canonicalized according to [XMLCan-W3C].

Explanation: This specification enforces the use of canonicalization [XMLCan-W3C] for XML documents in order to provide equivalent versions for logically identical XML documents.

6 NOTES

6.1 Definitions

Note: this section contains only the terms that are not included in the SWIM Controlled Vocabulary [SWIM CV].

Controlled Resource	A version, access and availability controlled physical or virtual component within a computer system.
Default Namespace	A namespace that is declared using a reserved attribute '`xmlns`'. This means that for all elements in this namespace no prefix is required, that is, each element that has no namespace prefix is considered to be a part of this namespace.
Uniform Resource Identifier (URI)	A compact string of characters for identifying an abstract or physical resource. [RFC-3936]
Validator	A software program that accepts as input an XML document and determines whether it is well-formed and valid.
XML Document	One of several XML-based artifacts associated with a SOA service that is or will be cataloged in and made discoverable by the NAS Service Registry/Repository (NSRR). These artifacts include a *schema* that is used to validate XML messages produced by the service, an XML-based *service definition document* that describes the behavior of the service, and examples of *messages*.

6.2 Acronyms

AIXM	Aeronautical Information Exchange Model
HTTP	Hypertext Transfer Protocol
NAS	National Airspace System
NEMS	NAS Enterprise Messaging Service
NSRR	NAS Service Registry/Repository
SOA	Service-Oriented Architecture
SWIM	System Wide Information Management
URI	Uniform Resource Identifier

URL	Uniform Resource Locator
UTF-8	Universal Character Set Transformation Format - 8-bit
WSDL	Web Service Description Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language