SWIM-005
December, 2015
Version 1.0.0

SOFTWARE SPECIFICATION
Artifacts Versioning for 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

APPROVAL/SIGNATURE PAGE

[illegible] 12/17/15

Jeri Groce
SWIM Program Manager

[illegible] 12/17/2015

Mark Kaplun
SWIM Governance Lead

Date

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 SCOPE

1.1 Scope

This specification provides requirements and guidelines for versioning in the context of FAA's System Wide Information Management (SWIM) services. The rules defined in this specification are based on (but not limited to) non-proprietary industry-wide common practices and recommended guidelines.

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

1.2 Background

The inherently agile nature of SWIM as a multi-organizational service-oriented technological framework emphasizes the need for managing changes to SWIM-enabled service components. These changes usually originate from new business requirements, constantly evolving technological solutions, or modifications or upgrades to a common infrastructure. To maintain interoperability among independently developed components, software developers and governance bodies must be cognizant of a) whether the changes will negatively impact existing (and potentially future) service consumers and b) how changes that will and will not impact consumers should be implemented and communicated [Earl-2008].

Addressing these questions results in the need for versioning. Versioning supports the simultaneous existence of multiple (different) implementations of the same thing (artifact), with every implementation distinguishable and individually addressable [Lubinsky-2007]. In a service-oriented architecture (SOA), software versioning supports the creation and management of multiple releases of a service and provides a way to communicate changes to a service consumer agent. The versioning process is essential for reducing the impact of change on service consumers, reducing maintenance costs, and improving the overall manageability of services [Lubinsky-2007].

The goal of this document is to specify a process for assigning version identifiers to software artifacts for the purpose of managing their evolution.

1.3 Entity Type Description

This specification defines a data object called *version identifier* as a unique identifier that specifies a unique state of software artifact, and describes how the identifier is created and applied in the context of a SWIM-enabled service.

The first step toward establishing an effective versioning mechanism is to decide on a common way by which versions are identified. There are multiple schemes in industry for defining version identification patterns; this specification uses an approach in which the version identifier is constructed as a sequence of non-negative integers, each of which represents the measure or significance of the changes made. [SemVer2.0.0]

This specification classifies all changes that can be made into three categories: *major*, *minor* and *patch*.

1. The term *major* is reserved for changes or updates that are not backward-compatible; that is, they force a consumer agent to change in order to use the new version of the service. It says that these changes "break" a consumer agent.
2. The term *minor* is reserved for changes that allow a consumer agent to continue to use the existing version of the service (they do not "break" the consumer agent), although the consumer agent is unable to use or is unaware of the new features. These changes are considered backward-compatible (e.g., a new capability, new optional request parameters).
3. The term *patch* is reserved for backward-compatible bug fixes that do not affect in any way interaction between service and consumer agent (e.g., fixing a bug in software, correcting a typographical error in an XML schema document).

To illustrate these concepts, figure 1 presents a scenario wherein a service consumer agent was designed to use a version 1.0.0 of some artifact (e.g., an XML schema). After *minor* changes, which were communicated through the incremented second and third digits in the version identifier (1.1.0 and 1.2.1 respectively), the consumer agent could continue to use each subsequent version of this artifact. However, after *major* changes were made (and communicated via the incremented first digit), the consumer agent required modification in order to use version 2.0.0 of the artifact.

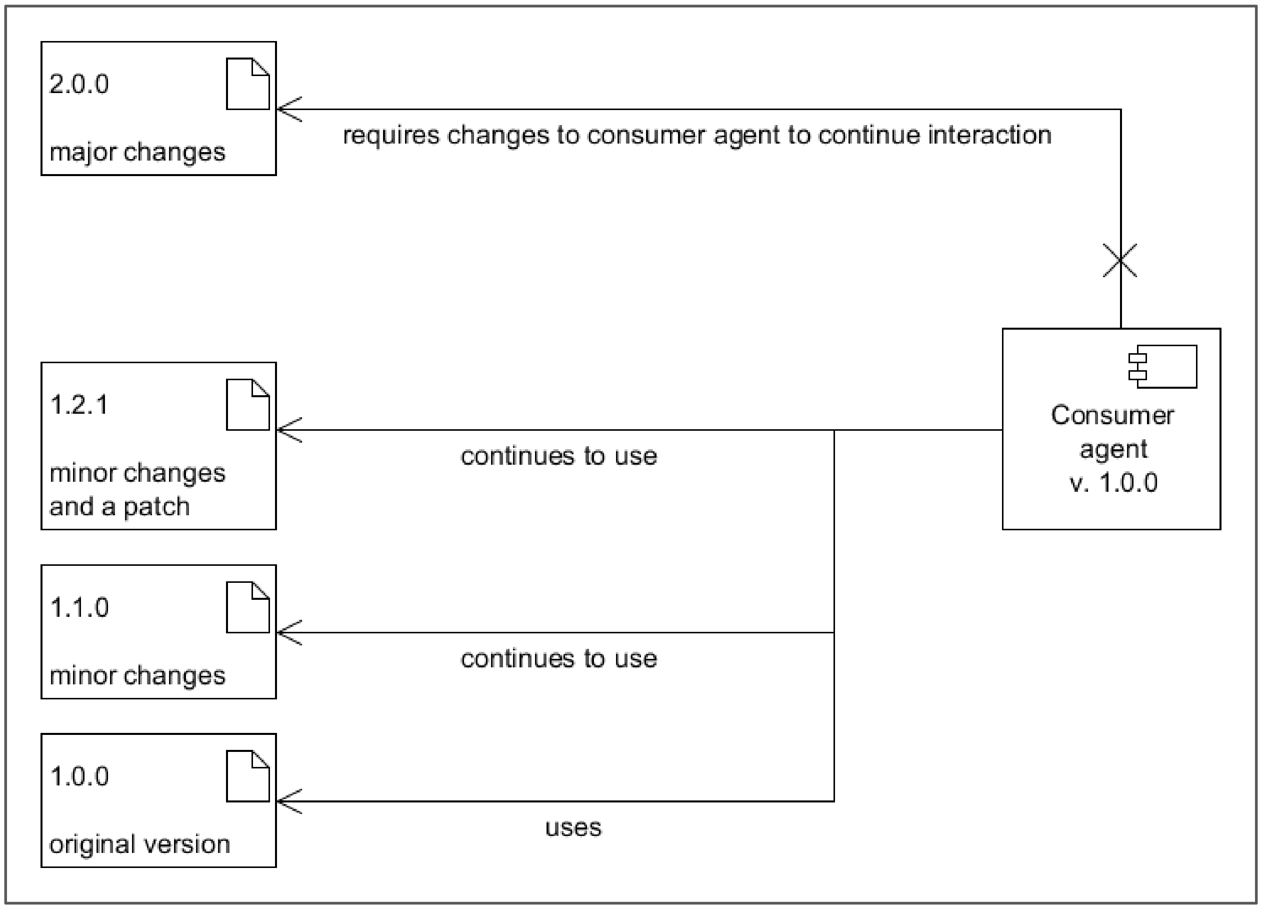


FIGURE 1 Version usage scenario

Detailed requirements for creating a version identifier are presented in section 3.2 of this specification.

1.4 Process Overview

According to the SWIM Governance Policies [SWIM GP], the service provider is responsible for developing and maintaining services and their artifacts, deciding when a new version of a service or artifact is warranted, maintaining version control, and assigning version identifiers. The provider is also responsible for uploading required service artifacts to the NAS Service Registry/Repository (NSRR), which maintains service descriptions and 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 NAS/SWIM stakeholders for future use. When the service is ready to be deployed or tested, copies of appropriate artifacts are uploaded to the NAS Enterprise Messaging Service (NEMS) to support data exchange.

A versioning process with respect to service development must answer the following questions:

- What artifacts should be versioned?
- How should these artifacts be versioned?

Changes made to a SOA service are declared and effectuated through the artifacts that express service functionalities or characteristics. There are a variety of such artifacts in industry and FAA, but this specification focuses only on the following as the most germane to the SWIM's change management processes. These artifacts are:

1. **XML schema**: an XML document that defines structure and constraints for the types of data to be transmitted by service components. [SWIM XML-DOC]
2. **WSDL document**: a formal description of types, messages, interfaces and their concrete protocols and data format bindings, and the network access points associated with Web services [SWIM XML-DOC].
3. **JMS Message header**: the part of a JMS message that allows the message producer, service broker (NEMS) and message recipient to identify or route the message. It is important to note that this specification does not suggest that the message header itself is being versioned, but that the message header includes versioning information relevant to the content of the message.
4. **NSRR meta-card**: a set of metadata attributes and artifacts associated with a single service in the NSRR. Every service meta-card in the NSRR includes a required "Service Version" attribute that indicates the current version of the service.

The requirements describing how each of these artifacts should be versioned are presented in sections 3.3.1, 3.3.2, 3.3.3 and 3.3.4 respectively.

1.5 Intended Audience

This specification is intended for use by application architects and software developers. While the document contains some explanatory material, it assumes the user has a basic familiarity with XML schemas, WSDL specifications, and implementation of JMS in the context of NEMS.

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 used in this document are presented in `constant width font`.

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 XML-DOC]	Syntax and Processing of XML-Based Documents in the Context of SWIM-Enabled Services; Software Specification; Version 1.0; June 16, 2015 http://www.faa.gov/nextgen/programs/swim/governance/standards/media/Syntax%20and%20Processing%20of%20XML-Based%20Documents%2006162015.pdf
[SWIM CV]	SWIM Controlled Vocabulary (CV), http://www.faa.gov/nextgen/programs/swim/vocabulary/

2.2 Non-Government Publications

[SEI-2012]	Best Practices for Artifact Versioning in Service-Oriented Systems; TECHNICAL NOTE; Software Engineering Institute Marc Novakouski, Grace Lewis, William Anderson, Jeff Davenport; January 2012; http://www.sei.cmu.edu
[Lubinsky-2007]	Lublinsky, B. “Versioning in SOA.” *Microsoft Architecture Journal 11* (April 2007). http://msdn.microsoft.com/en-us/library/bb491124.aspx
[SemVer2.0.0]	Semantic Versioning 2.0.0; Tom Preston-Werner; Creative Commons; 2013; http://semver.org/spec/v2.0.0.html

[Earl-2008]	Web Service Contract Design and Versioning for SOA; Thomas Erl et al;Prentice Hall; 2008
[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

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 service artifacts described in this specification, as well as to the service itself. This specification follows the approach described in Semantic Versioning 2.0.0 [SemVer2.0.0].

a. Every version of a service or service artifact SHALL have a version identifier formatted as three dot-separated positive integers without leading zeroes, e.g., “1.2.3”.

1. The first digit SHALL represent *major* changes to the service or service artifact.
2. The second digit SHALL represent *minor* changes to the service or service artifact.
3. The third digit MAY represent a *patch* to the service or service artifact.

b. The default value for every version identifier SHALL be 1.0.0.

Explanation: When a service or service artifact is provided to SWIM Governance without a version identifier, it will be assumed that the current version of the service or artifact is 1.0.0, and that future versions will be identified as described above.

c. A version identifier SHALL be presented as a part of the versioned artifact without reliance on any additional tooling.

Explanation: Versioning data associated with an artifact should be *autonomous;* that is, it should not rely on a tool or repository that may store or manage the artifact (e.g., NSRR) and should provide consistent versioning identification even when used outside of this tool.

3.3 Detailed Requirements

3.3.1 XML Schema Versioning Requirements

This section specifies the changes to an XML schema document that require changes to the schema's version identifier and how the version identifier should be presented in the context of the document. Since an XML schema in the SWIM environment is always uploaded to or referenced by the NSRR, the requirements specified in this section are aligned with the requirements defined in the specification SWIM-002 "Syntax and Processing of XML-Based Documents in the Context of SWIM-Enabled Services" [SWIM XML-DOC].

a. When *major* changes to an XML schema document are made (see section 3.3.1.1), the schema SHALL be published in a new namespace.

Explanation: Because the namespace for an XML schema document is presented as a value of the attribute `targetNamespace` in the XML schema root element, the new value for this attribute should be defined.

b. An XML schema's namespace SHALL conform to the structure and constraints specified in the "Syntax and Processing of XML-Based Documents in the Context of SWIM-Enabled Services" specification [SWIM XML-DOC].

c. The version identifier SHALL be embedded in the XML schema namespace URI (e.g., `http://www.aixm.aero/schema/5.1` or `http://www.opengis.net/gml/3.2`).

d. To support backward compatibility with current versioning and version identifier engineering practices, for the first schema version (1.0.0) the versioning part MAY be omitted from the namespace URI.

3.3.1.1 Major changes

a. The following modifications to an XML schema SHALL be considered *major*:

1. Renaming or removing a global type or element.

2. Changing the type of a global element.
3. Changing the type of a local element.
4. Changing the definition from optional to required, for both global and local elements.
5. Adding or removing an enumeration value.
6. Changing the value of an `elementFormDefault` attribute.

3.3.1.2 Minor changes

a. The following modifications to an XML schema SHALL be considered *minor*:

1. Adding a global element or a type.
2. Changing the definition of a local element from required to optional.

3.3.1.3 Patches

a. The following modifications to an XML schema MAY be considered a *patch*:

1. Adding or changing descriptions or annotations.

3.3.2 WSDL Versioning Requirements

This section specifies the changes to a WSDL document that require changes to the WSDL's version identifier and how the version identifier should be presented in the context of the document. Since a WSDL document in the SWIM environment is always uploaded to the NSRR, the requirements specified in this section are aligned with the requirements defined in the specification SWIM-002 "Syntax and Processing of XML-Based Documents in the Context of SWIM-Enabled Services" [SWIM XML-DOC].

a. When *major* changes to a WSDL document are made (see section 3.3.2.1), the document SHALL be published in a new namespace.

Explanation: Because the namespace for the WSDL document is presented as a value of the attribute `targetNamespace` in a `<wsdl:definition>` element, the new value for this attribute should be defined.

b. The version identifier SHALL be embedded in the WSDL document `targetNamespace` URI (e.g., `http://faa.gov/fps/2.1.0`).

c. To support backward compatibility with current versioning and version identifier engineering practices, for the first schema version (1.0.0) the versioning part MAY be omitted from the namespace URI.

3.3.2.1 Major Changes

a. The following modifications to a WSDL document SHALL be considered *major*:

1. Removing any element declared in the WSDL (`message, message part, portType, binding` or `service`).
2. Renaming any element declared in the WSDL (`message`, `message part`, `portType`, `binding` or `service`).
3. Changing `operation` or `message` signature.
4. Changing the message exchange pattern (MEP) of any existing `operation` definition.

3.3.2.2 Minor Changes

a. The following modifications to a WSDL document SHALL be considered *minor*:

1. Adding a new `operation.`
2. Adding a new `portType`, `binding` or a `service`.
3. Updating a `message` element with a reference to a schema type that is derived as an extension of the original.
4. A new, optional, behavior is added without changing the message signatures or types.

3.3.2.3 Patches

a. The following modifications to a WSDL document MAY be considered a *patch*:

1. Adding or changing descriptions or annotations.

3.3.3 JMS Message Header Versioning Requirements

A JMS message header is constructed as a set of header fields, which are name-value pairs that are used by message producers, brokers and recipients to identify and route messages. This section specifies how pertinent versioning information should be presented in the context of a JMS message header.

a. All JMS headers SHALL include the service version information formatted as follows: `[service name]:version = [service version identifier]`; for example `fps:version=”2.3.0”.`

b. When a JMS message contains XML-based content, the JMS header SHALL contain the target namespace for the schema that defines the message’s content formatted as follows: `[service name]:content:namespace = [xml content target namespaces]`; for example `fps:content:namespace=”http://faa.gov/fps/1.2.3”.`

Explanation: When multiple versions of a service run concurrently on the NEMS, a service producer should communicate to the NEMS which version of the service has produced the message. Subsequently this information is passed to a consumer agent to assure that the message is interpreted (parsed) appropriately.

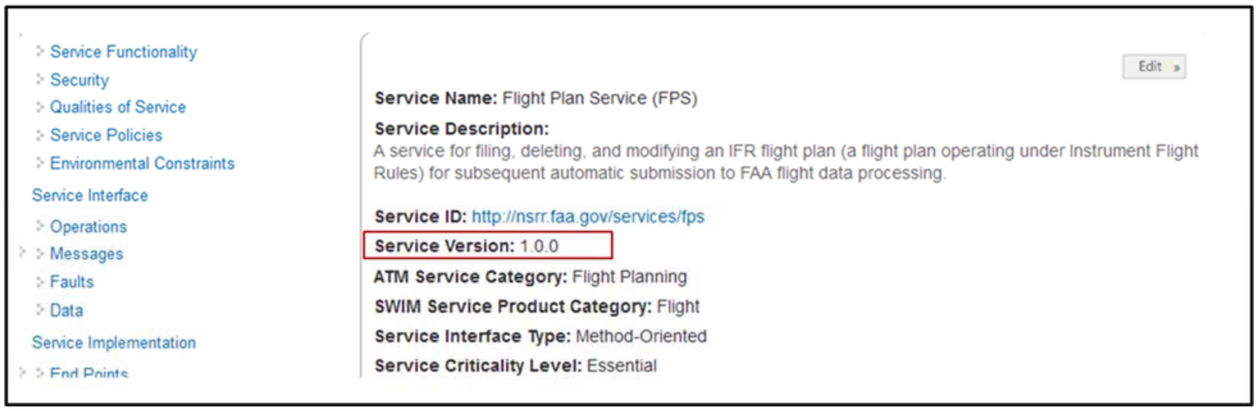
3.3.4 Service Versioning Requirements

The main goal of the NSRR is to provide insight into all known SWIM-enabled services, both currently available and under development. It does this by maintaining a set of metadata attributes, called a meta-card, that describes a particular registered service. The NSRR meta-card allows entry of the version identifier of a service as well as version identifiers for all documents associated with the service (e.g., XML schema, WSDL document, etc.). Defining what constitutes a new version of the service requires analyzing the possible changes, their potential impact on the consumer agent's execution, and identifying the ones that will "break" it [Lubinsky-2007].

This section specifies the changes to a service's meta-card that require incrementing the service's version identifier. **Note:** an exhaustive list of changes to an NSRR service meta-card that may result in changes to the service version identifier is not presented in this specification; the service provider is expected to work with SWIM Governance and SWIM Engineering personnel to analyze the possibility of changes to the service version identifier.

a. All NSRR meta-cards SHALL include the service version identifier formatted as prescribed in section 3.2 of this document.

The following is an example of a version identifier entry in the NSRR.



Service Name: Flight Plan Service (FPS)
Service Description:
A service for filing, deleting, and modifying an IFR flight plan (a flight plan operating under Instrument Flight Rules) for subsequent automatic submission to FAA flight data processing.

Service ID: http://nsrr.faa.gov/services/fps
Service Version: 1.0.0
ATM Service Category: Flight Planning
SWIM Service Product Category: Flight
Service Interface Type: Method-Oriented
Service Criticality Level: Essential

FIGURE 2 Example of version identifier in the NSRR 2.0

3.3.4.1 Major Changes

a. The following modifications to a service's meta-card SHOULD be considered *major*:

1. Changes to the XML schema or schemas that are identified as *major* in section 3.3.1.1 of this specification.
2. Changes to the WSDL document that are described as *major* changes in section 3.3.2.1 of this specification.

3. Changes or additions to qualities of service (QoS) that will negatively impact existing service consumers or consumer agents.

Explanation: Changes to a service's implementation – even if its interface remains intact – may result, for example, in changes to the service response time (a QoS). If a service consumer is invoking this service synchronously, an increase in service response time can negatively impact service consumer execution. Such changes in the service meta-card should be considered as major changes that require a new service version.

4. Changes or additions to security mechanisms that may prevent existing consumers from using the service.

Explanation: Adding a new security mechanism, such as message encryption, may result in rendering an existing consumer agent unusable.

3.3.4.2 Minor Changes

a. The following modifications to a service's meta-card SHALL be considered *minor*:

1. Changes to the XML schema or schemas that are identified as *minor* in section 3.3.1.2 of this specification.
2. Changes to the WSDL document that are identified as *minor* in section 3.3.2.2 of this specification.
3. Changes or additions to QoS that may improve service performance and will not negatively impact existing service consumers or consumer agents.

Explanation: Decrease in a message's latency, for instance, will not require changes to an existing consumer agent, although it may improve the agent's performance. This kind of change reflected in the service meta-card should be communicated to consumers by incrementing the service version identifier.

4. Changes or additions to security mechanism(s) that will not negatively impact existing service consumers.

Explanation: Adding integrity measures to service security will not require a consumer to make changes to an existing agent, but the change should be communicated to consumers by incrementing the service version identifier.

3.3.4.3 Patches

a. The following modifications to a service's meta-card MAY be considered a *patch*:

1. Changes or additions to descriptions of a service attribute in the context of the service meta-card.

4 QUALITY ASSURANCE PROVISIONS

All service artifacts presented for uploading to the NSRR will be subject to verification that they conform to the requirements prescribed by this specification.

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

TABLE I Requirements Verification Matrix

Requirement	Verification Method
3.2.a.1	Inspection
3.2.a.2	Inspection
3.2.a.3	Inspection
3.2.b	Validation by the NSRR
3.2.c	Validation by the NSRR
3.3.1.a	Inspection
3.3.1.b	Inspection
3.3.1.c	Inspection
3.3.1.d	Inspection
3.3.1.1.a	Inspection
3.3.1.2.a	Inspection
3.3.1.3.a	Inspection
3.3.2.a	Inspection
3.3.2.b	Inspection

Requirement	**Verification Method**
3.3.2.c	Inspection
3.3.2.1.a	Inspection
3.3.2.2.a	Inspection
3.3.2.3.a	Inspection
3.3.3.a	Inspection
3.3.3.b	Inspection
3.3.4.a	Validation by the NSRR
3.3.4.1.a	Inspection
3.3.4.2.a	Inspection
3.3.4.3.a	Inspection

5 PREPARATION FOR DELIVERY

Not applicable.

6 NOTES

6.1 Definitions

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

Service Meta-card	A set of metadata attributes associated with a single service registered in the NSRR.

Uniform Resource Identifier (URI)	A compact string of characters for identifying an abstract or physical resource. [RFC-3936]
Version Identifier	A unique name or number that denotes a particular version of a service or service-related artifact.
Versioning	The process of managing multiple releases of a service or its artifacts for the purpose of managing the service's evolution.

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
QoS	Quality(ies) of Service
SOA	Service-Oriented Architecture
SWIM	System Wide Information Management
URI	Uniform Resource Identifier
UTF-8	Universal Character Set Transformation Format - 8-bit
WSDL	Web Service Description Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language