International Civil Aviation Organization

The Third Meeting of System Wide Information Management Task Force (SWIM TF/3)

Bangkok, Thailand, 07 – 10 May 2019

Agenda Item 3: d) Task 1-4: SWIM governance

GUIDANCE FOR VERSIONING SWIM SERVICES

(Presented by USA, Federal Aviation Administration (FAA))

SUMMARY

An important but challenging problem in a multi-organizational technological framework of reusable and shared services such as SWIM is the coexistence of multiple releases of active services. Addressing these questions results in the need for a service versioning scheme, i.e., a set of rules for assigning version identifiers to services or service-related artifacts for the purpose of managing their changes.

This Working Paper provides requirements and guidelines for service versioning in the context of ICAO Asia and Pacific Office (APAC) System Wide Information Management (SWIM). The requirements defined in this document are based on non-proprietary industry-wide common practices and recommended guidelines.

1 INTRODUCTION

1.1 The intrinsically agile nature of SWIM as a multi-organizational service-oriented technological framework places special emphasis on the need for managing changes to services and service-related artifacts. These changes usually originate from new business requirements, constantly evolving technological solutions or modifications or upgrades to a common infrastructure [1].

1.2 To maintain interoperability among independently developed components, SWIM implementers must be cognizant of a) how the changes will impact existing (and potentially future) service consumers and b) how the changes should be communicated to the consumers to avoid potential interruptions of interactions with services [2].

1.3 Addressing these questions results in the need for versioning. 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 [3].

1.4 Creating and enforcing a cohesive and comprehensive versioning policy is as part of a governance effort of developing the APAC SWIM Governance framework, by further enhancing APAC SWIM's standards and policies catalogs as defined in the Policy-Centric model [2].

1.5 This Working Paper is in essence a *prestandard*[1] that specifies requirements for the creation and assignment of service version identifiers. These requirements will be applied to the services developed and managed in the context of APAC SWIM.

2 DISCUSSION

2.1 Background

2.1.1 After a service has been developed and made available for consumption, two things naturally occur: a service consumer begins to form dependencies with the service by developing a client or by deploying the service as a part of a service composite, and the service starts to evolve through a sequence of changes realized as new releases.

2.1.2 There are various types of changes that may result in a new release. For example:

2.1.2.1 Changes to a service interface or data being exchanged to improve service capabilities; e.g., an operation might be added, or a data element in a message might be added, removed, or revised.

2.1.2.2 Changes to a service as a whole; e.g., a service can be moved to a new implementation environment and, as a consequence, endpoints may have new addresses.

2.1.2.3 Changes to a service binding or policies; e.g., a security policy or a security mechanism might be added, removed, or revised.

2.1.3 There are two types of versioning typically applied to a software product such as a service: *internal* versioning, which is used by the service development team to keep track of the progress of the project, and *release* versioning, which identifies versions that are made available (released) to consumers. The former is usually organization-specific, invisible to consumers, and follows the provider's engineering practices, while the latter is made available to a wider audience and should therefore adhere to a commonly recognized approach to versioning. It should be noted that this paper focuses only on *release* versioning, i.e., the type of versioning that needs to be communicated to a requester of a service.

2.1.4 There are a number of versioning schemes used mostly – but not exclusively – for software development. Examples include:

2.1.4.1 *Sequence-based*: a version identifier that uses a sequence of numbers or letters in an alphabet to indicate a new version.

2.1.4.2 *Significance of changes*: a version identifier that provides an understanding of change impact in the context of existing releases.

2.1.4.3 *Date of release*: a version identifier that uses a string to represent a date when an artifact is released.

2.1.5 This Paper follows the approach to versioning described in the Semantic Versioning Specification [4]. Under this scheme, version numbers and the way they change convey meaning about what has been modified from one version to the next and whether consumers will be forced to make changes to existing software agents.

---

[1] International Organization for Standardization (ISO) defines a *prestandard* as a document that is adopted provisionally by a standardizing body and made available to the public in order that the necessary experience may be gained from its application on which to base a standard [5].

2.1.6 Although the Paper focuses on the versioning of SWIM-enabled services, the versioning scheme described herein can – and should – be used for identifying versions of other service-related artifacts, e.g., machine-processable service definitions, data models or schemas used by a service, etc.

2.2 Key Concepts

2.2.1 *Versioning* - The process of managing multiple releases of a service or its artifacts for the purpose of managing the service's evolution.

2.2.2 *Version identifier* - A unique name or number that denotes a particular version of a service or service-related artifact.

2.2.3 *Major changes* - 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 is said that these changes "break" a consumer agent.

2.2.4 *Minor changes* - 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).

2.2.5 *Patches* - Backward-compatible error corrections that do not affect in any way interaction between service and consumer agent (e.g., fixing a bug in a software, correcting a typographical error in an XML schema document).

2.3 Key Words

2.3.1 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this Paper are to be interpreted as described in RFC 2119 [6].

2.3.2 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.

2.4 Requirements

2.4.1 A version identifier SHALL be formatted as three dot-separated positive integers without leading zeroes, e.g., "1.2.3".

2.4.1.1 The first digit SHALL represent a major change to the service or service artifact.

2.4.1.2 The second digit SHALL represent a minor change to the service or service artifact.

2.4.1.3 The third digit MAY represent a patch to the service or service artifact.

2.4.2 *Explanation: To illustrate usage of these requirements, figure 1 presents a scenario wherein a service consumer agent was designed to use version 1.0.0 of some service. 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 service. 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.

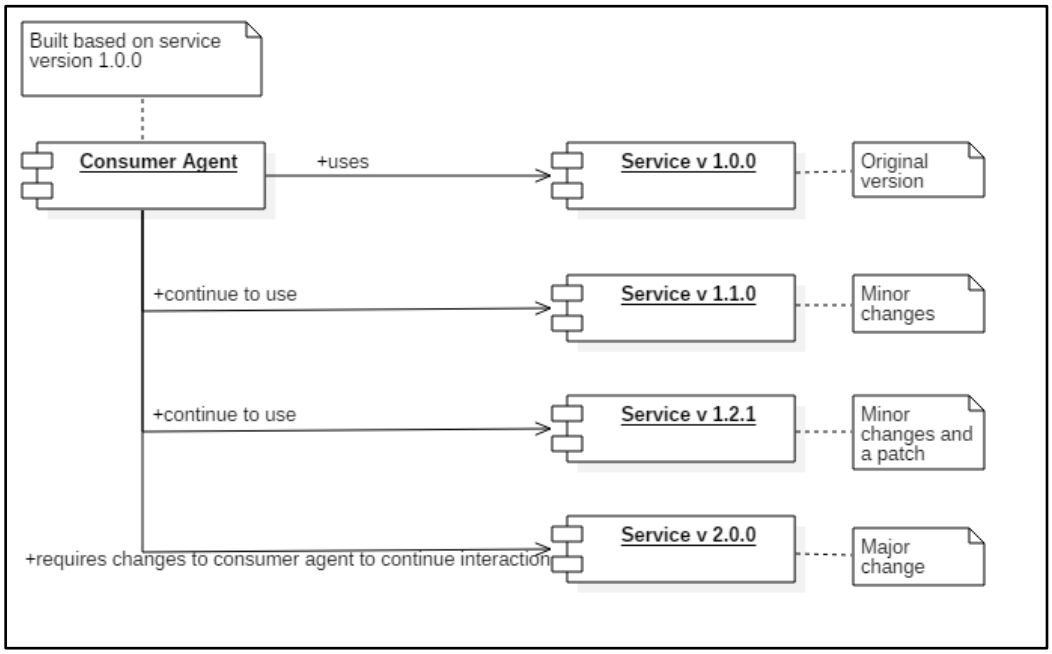

Figure 1. Example of applying described versioning approach

2.4.3 When major changes to an XML document are made, the document SHALL be published in a new namespace.

2.4.4 *Explanation: A number of XML documents are commonly associated with services (e.g., a service definition or WSDL document, an XML schema). To avoid naming conflicts that may arise from changing element names, a new version of a namespace must be created. Examples of XML document changes and how they affect service version levels are shown in Appendixes A and B.*

2.4.4.1 The version identifier SHALL be included in an XML document namespace URI as prescribed in APAC SWIM Guidance for Creating Service Identifiers [7].

2.4.4.2 *Explanation: For example, http://www.aixm.aero/schema/**5.1** or http://ansp.aero/fps/**1.2**).*

2.4.5 When changes to non-functional properties of a service (e.g., policies, including security policies, performance QoS) are made, the new service version identifier SHALL be constructed as described in section 2.4.1.

2.4.6 *Explanation: Changes to non-functional properties are sometimes harder to assess than changes to functional properties (e.g., changes to a service interface). However, the same approach for evaluating the level of potential impact on a consumer agent should be applied. For example, changing a security mechanism may require a consumer to make changes in the procedure by which the service is invoked and could be assessed as a major change, while increasing the data accuracy will not require any changes to a consumer agent and could be considered a minor change.*

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

2.4.8 *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 and should provide consistent versioning identification even when used outside of this tool.*

2.4.9 Note: This Paper makes no assertions about a correlation between versioning of a service and versioning of the documents that are used to describe or represent the service.

2.5 **Policies**

2.5.1 This section defines APAC SWIM Governance policies that prescribe usage of the requirements presented in section 2.4 of this Paper.

2.5.2 Each release of a SWIM service or service-related artifact will be versioned according to the guidance set forth in this Paper.

2.5.3 No retroactive application of these guidelines is envisioned for currently published services' versions. Existing version identifiers will be exempted.

2.6 **Conclusions**

2.6.1 In a service-centric environment such as SWIM, service versioning is a critical factor in managing changes to services or service-related artifacts, while mitigating impacts of these changes on service consumers.

2.6.2 The service versioning scheme described in this Paper provides a syntax (structure of parts constituting the service identifier) along with semantics (meaning of each part) for a service identifier.

2.6.3 The rules described in this Paper for creating and incrementing a service identifier are expected to be followed by APAC SWIM service developers for engineering multiple versions of the same service or service-related artifact.

<table>
<tr><td colspan="3">Daft Conclusion/Draft Decision/Decision XX/XX - GUIDANCE FOR VERSIONING SWIM SERVICES</td></tr>
<tr><td colspan="2">What: Adopt the requirements defined in this Paper as the APAC SWIM Standard</td><td>Expected impact:
☐ Political / Global
☐ Inter-regional
☐ Economic
☐ Environmental
☒ Ops/Technical</td></tr>
<tr><td>Why: The versioning process is essential for reducing the impact of change on service consumers, reducing maintenance costs, and improving the overall manageability of services</td><td colspan="2">Follow-up: ☐Required from States</td></tr>
</table>

When:	7-May-19	Status:	Draft to be adopted by PIRG
Who:	☒Sub groups ☒APAC States ☐ICAO APAC RO ☐ICAO HQ ☐Other: XXXX		

3 ACTION BY THE MEETING

3.1 The meeting is invited to:

a) Review the contents of this Working Paper;

b) Request the Secretariat to establish a repository for APAC SWIM standards and policies;

c) Adopt the requirements defined in this Paper as the APAC SWIM Standard for versioning services and service-related artifacts;

d) Include the policies proposed in section 2.5 of this Paper into a future APAC SWIM Governance policies catalog.

4 REFERENCES

[1] SWIM-005, Artifacts Versioning for SWIM-enabled Services, Software Specification, Version 1.0.0; FAA SWIM; December 18, 2015.
https://www.faa.gov/air_traffic/technology/swim/governance/standards/media/SWIM Service Versioning Spec.pdf

[2] Policy-Centric Governance Model for APAC SWIM; Second Meeting of System Wide Information Management Task Force (SWIM TF/2); Bangkok, Thailand; April 2018.

[3] Lublinsky, B. "Versioning in SOA." Microsoft Architecture Journal 11; April 2007.
http://msdn.microsoft.com/en-us/library/bb491124.aspx

[4] Semantic Versioning 2.0.0; Tom Preston-Werner; Creative Commons; 2013.
http://semver.org/spec/v2.0.0.html

[5] ISO/IEC CD 20944-002 Information technology - Metadata Registry - Interoperability & bindings Part 2: Common vocabulary; SC 32 Secretariat; 2004-07-27.
http://jtc1sc32.org/doc/N1101-1150/32N1105T-CD20944-002.pdf

[6] 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

[7] APAC SWIM Guidance for Creating SWIM Service Identifiers, DRAFT; ICAO SWIM T/F; October 2018.

[8] W3C XML Schema Definition Language (XSD) 1.1 Part 1; W3C Recommendation; 5 April 2012.
https://www.w3.org/TR/xmlschema11-1/

[9] Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language; W3C Recommendation; 26 June 2007.
https://www.w3.org/TR/2007/REC-wsdl20-20070626/

5 APPENDIXES

5.1 Appendix A. Example of changes to an XML schema with consequent level of changes to the service version.

The following examples are non-normative. The examples use a terminology defined in W3C XML Schema Definition Language (XSD), Version 1.1 [8]

Level of changes	***Changes***
Major	Renaming or removing a global type or element. Changing the type of a global element. Changing the type of a local element. Changing the definition from optional to required, for both global and local elements. Adding or removing an enumeration value.
Minor	Adding a global element or a type. Changing the definition of a local element from required to optional.
Patch	Adding or changing descriptions or annotations.

5.2 Appendix B. Example of changes to a WSDL document with consequent level of changes to the service version.

The following examples are non-normative.
Note, the examples follow the component model and syntax defined in WSDL specification version 2.0 [9].

Level of changes	***Changes***
Major	Removing any element declared in the WSDL (message, message part, interface, binding or service). Renaming any element declared in the WSDL (message, message part, interface, binding or service). Changing operation or message signature. Changing the message exchange pattern (MEP) of any existing operation definition.
Minor	Adding a new operation. Adding a new interface, binding or a service. Updating a message element with a reference to a schema type that is derived as an extension of the original. A new, optional, behavior is added without changing the message signatures or types.
Patch	Adding or changing descriptions or annotations.