



Versioning Schemas and WSDLs

Best Practices

Version: 1.0

AN ACTIVE ENDPOINTS TECHNICAL NOTE

© 2010 Active Endpoints Inc. ActiveVOS is a trademark of Active Endpoints, Inc. All other company and product names are the property of their respective owners.



2010

Content

Introduction.....	3
Major/Minor Updates	3
XML Schema Major Updates.....	3
XML Schema Minor Updates	3
WSDL Major Updates.....	4
WSDL Minor Updates.....	4
Document Revisions	4
Adding, Removing or Updating a Local Element.....	5
Versioning.....	5
Schema Versions.....	5
WSDL Versions	5
Resource locations.....	6
Impact of Revisions.....	6
Schema Revisions.....	6
New Schema Version	6
Incremental (delta) Schema Updates	7
WSDL Revisions.....	7
Backwards-compatible Versions	7
Backwards-incompatible Versions.....	8
About Active Endpoints	8

Introduction

This document proposes best practices for versioning WSDL and XML Schema documents. It covers major and minor update concepts, recommendations for versioning these artifacts, and the impact of doing so.

Major/Minor Updates

The term **major update** is reserved for updates that break existing service consumers – i.e. clients, or invalidates previously shared (and consumed) documents. The term **minor update** is reserved for backward-compatible updates of an interface which does not break existing clients or existing documents such that they remain valid if continuing to use a minor update to an existing document.

XML Schema Major Updates

Updates to an XML Schema document are considered to result in a major update if:

- There are changes to the type of a local element
- There are changes to a local element or an element reference requiring that it become mandatory (i.e. changing the definition from **optional** to **required**).
- Adding or removing an enumeration value
- Renaming or removing a global type or element
- Changing the type of a global element
- Updates are made to the **appInfo** element's content
- Changes are made to the elementFormDefault attribute value

XML Schema Minor Updates

The following updates to an XML Schema document are considered to result in a minor update if:

- There are changes to a local element or element reference making it optional (i.e. changing from the definition from **required** to **optional**).
- Adding a global element or type
- Adding documentation or comments

WSDL Major Updates

The following updates to a WSDL document are considered to result in a major update if:

- Updates are made to existing binding, service or portType definitions
 - Adding a binding, a port (Type) or an operation however does not result in a major update.
- Updates are made to a message element with a reference to a schema type that is not derived as an extension of the original
- Any element declared in the WSDL (message, message part, portType, binding or service) is removed

Major updates **MAY** be backwards-incompatible for existing clients.

WSDL Minor Updates

The following updates to a WSDL document are considered to result in a minor update if:

- A new operation, possibly with message and message parts definitions is added
- A new portType, binding or a service to a WSDL is added
- A new, optional, behavior is added without changing the message signatures or types
- Updating a message element with a reference to a schema type that is derived as an extension of the original

If a major update is made to a schema referenced by a WSDL document, the WSDL will be isolated from the update if and only if a specific schema document version is referenced from the WSDL. Under such condition no changes are required to the WSDL major or minor version identifiers. If this is not the case, the major version identifier needs to be updated.

Minor updates **MUST** be backwards-compatible for existing clients.

Document Revisions

A **revision** is an update to a document which implies no semantic changes to its meaning, e.g. an update in a white space, formatting, non-normative documentation, comments etc. Incremental development before publishing of the document can be also seen as revisions of the same (e.g. 1.0) version. A revision to a previously published document **MUST NOT** impact the functionality of either clients or service endpoint's implementer.

Individual revisions of a document in a source control system represent revisions in this sense unless they are classified as major or minor updates (per above). The initial development of a service, before it is published for production use, or internal incremental development can also be construed as revisions.

Adding, Removing or Updating a Local Element

Adding an element implies an update to the content model of the element's parent. As the element's name is fixed it cannot be *substituted* and changed, the only viable alternative is to update its definition type and the content model.

Versioning

If a major update to a schema or its part occurs, the updated entities **MUST** be published in a new namespace to avoid conflicts with existing usages of the schema. Note that unchanged entities **MAY** remain in their old namespace; guidelines are provided below.

Likewise, if a *major* update in a WSDL occurs the update **MUST** be published in a different namespace, so that existing clients are not affected. A *major* update **MAY** be classified as **backwards-incompatible** if there is no intention to support old clients.

Schema Versions

XML Schema allows specifying an (optional) attribute of the **xs:schema** element's **version** attribute. The content model permits Dewey notation of major.minor version number. Consistent usage of the version number **is** encouraged. Major updates in the schema **SHOULD** be reflected by an update in the schema namespace **AND** the version attribute.

The first published version of a document **MAY** contain version information in its namespace; namespaces for subsequently published modifications **MUST** contain the version information.

WSDL Versions

There is no element or attribute to carry version information defined by W3C for WSDLs. The only means of versioning a Web Service is to publish its WSDLs at different URIs, giving each a different namespace.

Resource locations

As new versions of a document (XSD, WSDL) are published, the original or prior version should remain available and accessible for older clients which are bound to it. The version or published date information **SHOULD** be a part of the resource URI under which the resource is published.

The first published version of the resource **MAY** contain version information in its URI; URIs for subsequently published modifications **MUST** contain the version information.

Impact of Revisions

During development all updates to resources are permitted without any constraint. After the document is published, a version and change management policy **SHOULD** be applied to it.

Schema Revisions

Major updates to a schema **MUST** be published in a separate namespace. **Minor** updates **MAY** be incorporated to the original namespace. The new version of the XML Schema document **MUST** be published as a separate resource using a versioned URI.

The Schema publisher **MAY** choose, considering the impact on clients, to handle a *minor* update as a major update - especially when independent service implementers are expected.

Updates that require a new namespace **SHOULD** be published in the following manner:

New Schema Version

In the case where a new schema is created, all definitions in the XML schema will be published using a *new versioned namespace* with the goal of isolating older clients from any updates made to the definitions. Old clients will not be able to use a Web Service that references and uses exclusively the new schema version¹. This approach is to be used for *major* updates to the XML Schema.

¹ This approach is idealistic and not a necessarily the way to manage change in an SOA environment. Understanding dependencies between consumers and producers of service and their related document artifacts is required.

Incremental (delta) Schema Updates

If a local element's type is updated, the update **SHOULD** be published as a new schema document that imports the original one (reusing the definitions that do not change). The incremental schema will then contain only the updated definition and will use a *new versioned namespace*. The delta schema **MAY** leverage XML Schema inheritance and imports from the original version, or previous deltas to reduce the number of definitions. Clients of incrementally updated schema documents will be able to utilize the extended or updated semantics without impacting clients of the original schema who will not be affected by such an update.

Major updates **MAY** be published as delta schemas. When a new version (i.e. complete document) of the schema documents is published, it **MUST** incorporate all deltas from the previously published versions of the schema document.

WSDL Revisions

Major updates **MUST** be published in a separate namespace. **Minor** updates may be incorporated into the original namespace. The new version of a WSDL document **MUST** be published as a separate resource, using a versioned URI.

Major updates **MAY** be classified as backward incompatible. **Minor** updates **MUST** be published in a backward compatible manner.

Backwards-compatible Versions

Updates and definition additions **MUST** be published using a new port type element (i.e. `wsdl:portType`) whose local name is the same as specified as the original `wsdl:portType`'s and uses a versioned URI as its namespace URI. The WSDL document **SHOULD** import the previous document version using the **import** directive for the previous document's namespace in order to reuse old definitions where appropriate.

The service **MUST** contain an additional port for the new portType, declared in the *new* versioned namespace that uses the *same* local name as the original port. The service **MUST** support all previous versions' ports in order to preserve backward compatibility for callers.

The ports that correspond to different *versions* of the same business service **MUST** be located at the same endpoint. The implementer should

be able to discern the specific service behavior requested by the client using the **SOAPAction**'s qualified name.

Backwards-incompatible Versions

A new version of the WSDL **MUST** be published using a versioned namespace URI. All definitions (portTypes, bindings, operations, messages) should be copied to the new WSDL version rather than imported. Service definitions **SHOULD NOT** contain ports with a namespace set to the previous versions of the WSDL resource.

The backwards-incompatible method of creating a revision obviously breaks existing clients, as they need to adapt to new qualified names. If several backwards-compatible versions of WSDL are created, the resulting definition will use several namespaces, whose content follows the course of a service's development. The increased "clutter" of the service definition into distinct namespaces may become sufficient to motivate a major, backwards-incompatible version.

About Active Endpoints

Active Endpoints' (www.activevos.com) ActiveVOS is the business process management system ([BPMS](#)) that development teams will love. ActiveVOS empowers project teams to create business process management ([BPM](#)) applications using services, making their businesses more agile and effective. ActiveVOS promotes mass adoption of SOA-enabled BPM applications by focusing on accelerating project delivery time with a complete, affordable and easy-to-use system. Active Endpoints is headquartered in Waltham, MA with development facilities in Shelton, CT.

To find out how Active Endpoints can help your business, visit <http://www.activevos.com>, call +1 781 547 2900 and press 1 for Sales, or email us at info@activevos.com.