Requirements on Change Management by:
AmberPoint, Confluent Software, IBM, Merant, TalkingBlocks

Change Management Requirements

6/2003

Scott Hinkelman, IBM
Shamik Sharma, Confluent Software
Mark Potts, Talking Blocks
Brian Carroll, Merant
Fred Carter, AmberPoint

1 Preface

1.1.1 Purpose

The purpose of this document is to define the requirements for an XML and web services based mechanism for communicating information about changes in a web service to interested parties, such as consumers of that service. [Issue: Should this also address changes in a set of web services?]

1.2 Scope

The focus of this requirements document is to facilitate the communication of permanent or semi-permanent changes to a web service that could impact consumers or brokers for that service. The following are deemed outside the scope of this document:

- The communication of operational status or similar transient changes
- The definition of a versioning scheme
- The communication of changes to UDDI information, since that is covered by UDDI version 3.0

1.3 Terminology

This document will use the following terms:
Change Detail – A data structure describing an aspect of a web service that has changed or will change.
Change Package – A collection of related Change Details. The collection would typically represent a logical set of changes, but could also represent a grouping based on time of implementation or other common characteristic.

2 General Requirements

2.1 Consistent with other web services standards

Change management vocabularies must be consistent with and/or make use of the existing and emerging standards in the web services space. These include, but are not limited to, WSDL, SOAP, WSIL, UDDI, BPEL4WS, WS-Security, WS-Context, OASIS Mgmt Protocol, and Grid. The W3C has mandated the requirements management (in terms of lifecycle and evolution) and consideration for consistency is needed.

2.2 Specifications must be XML-based

Change management vocabularies must be specified using XML Schemas.

2.3 Designed for Extensibility

Change management vocabularies must define a set of elements that describe the changes to a web service. To promote interoperability, the syntax and meaning of those elements must be clearly defined in each release of the specification. However, recognizing that the specification cannot anticipate all requirements, each release must not preclude two types of extensions:

1. User-defined extensions to the elements, attributes, and values that will be used by specific communities of web service providers and consumers
2. Additions and extensions to the elements, attributes, and values as new releases of the specification are created.

2.4 Not to preclude Embedability

Change management rendering of the XML vocabularies must not preclude for its embedding in other XML-based languages. This allows for, say, a WSDL description to contain information about all of its versions. If change management vocabularies are embeddable, then separate language files/messages need not exist (not to say they cannot, but they need not). Thus, a WSDL file may always, in the abstract WSDL layer, describe the current system.

2.5 Changeability across protocol stack

Example: To get a client and a service to talk to each other, a common understanding is needed on the transport protocol (HTTP 1.1/ SMTP) messaging protocol (SOAP1.1/Doc-literal), discovery protocol (UDDI v2.0), security protocol, management protocol, as well as the application-conversation protocol (My-ERP interface). Each of these protocols may be further parameterized, and interaction requires agreement on the valid parameters (e.g. the compression-scheme). A change to any of these protocols, or to any of the parameters can disable communication.

WS-Change should define a framework that allows evolution at different layers of the protocol stack. Any solution that focuses solely on the obvious WSDL-versioning problem, should to be designed within this larger framework.

2.6 Change Detail Identification and Compatability

Each Change Detail should be uniquely identified, so that recipients of change notifications can easily detect duplicate notices. The unique identification could be accomplished either by assigning each Change Detail a unique id or by assigning each change aspect an id that is unique within the scope of a Change Package and each Change Detail is assigned a unique ID. An indicator of backwards compatibility needs to be accommodated. The compatibility relationship is at the service level and pertains to the instances of the service within its lineage.

2.7 Change Integrity

WS-Change vocabularies must not to preclude the use of the WS-Security.

2.8 Change Schedule

We don't want to work out a negotiation protocol. Change Scheduling needs to allow readiness for users. User should be able to signal readiness back to provided. (avoid negotiation). Consistency with any other web services work for negotiation.

2.9 Separation of Change Data from Notification Mechanisms

The service via policies should be able to describe how the notification of change will take place. Note: This is supportive of the principals of separation of how notification takes place and what the change is.

3 Requirements common to both Human and Machine Processing

This section contains requirements that are relevant to both a human reader of a change Detail and for machine processing of a change Detail. For all the data requirements described in this section, if a single format is not suitable for both a human reader and for machine processing, then alternative forms of the data should be represented (perhaps similar to the way the xml-lang attribute allows multiple language representations of the same content.)

3.1 Provision for a Change Type Indicator

All change management vocabularies must provide an encoding of the type of change.
Change Type indicators within change management vocabularies must be provided in a machine readable and human readable form, if a single form is not suitable for both. The change type indicator can be used by tools (or clients or services) to determine if they can make the change (e.g. *semantic* changes are often not manageable by a tool at present [Semantic Information], but *protocol* changes are quite doable.

The types of changes may include, but are not limited to, the following:

- General Types
  - Syntactic
  - Semantic
  - Location
  - Operational
  - Security
  - Licensing
  - Pricing
  - Additions Only
- Syntactic (Probably WSDL specific)
  - MessageFormat
  - MessageStyle
  - MessageEncoding
  - ProtocolVersion
  - Message Element values (e.g., a range changes from 0-100 to 1-99)
- Semantic
  - MessageSemantics ("size" changes from inches to centimeters, etc.)
  - OperationSemantics ("Commit" means you agree to pay, not you agree to call, etc.)
  - FaultChanges
  - Behavoral Changes
  - ImplementationChange
- Protocol
  - Endpoint
  - Port
    - Service has moved
  - Binding
- Operational [NOTE: Are operational changes out-of-scope?]
  - Availability -- Temporarily the service's availability will change (only available weekdays, business hours East Coast, down for backups/maintenance Saturday nights 10-1amGMT, etc.)

    - o
- Security Type of Change
    - o Service consumers should be notified if the authorization requirements to be able to access a web service have been made more stringent.
- Licensing -- Failure to obtain proper license changes may result in failed operation or an altered fee schedule
- Pricing -- No change in operation, but cost per unit will change.

- Additions Only (perhaps this is a qualifier on the above)
    - o New Interface Message element
    - o NewService
    - o NewOperation
    - o NewBinding

Note: We need to understand how to deal with human-intervention indicators [NOTE: An example is needed here.]

3.2 Provision of an Impact indicator

Change management vocabularies must provide for an encoding of the impact of the change. The impact indicator informs the reader or tool (or client or service) of the urgency of the change. If the change is a deprecation, then it can be dealt any time. If the impact is failure, then a tool must deal with (or cause to be dealt with) the change immediately (or, at least, before the date of the associated timeline (see below).
This impact probably includes (but is not limited to) the following.

- Failure -- Failure to make these changes will result in failure of the operation. This SHOULD include specific fault information.
- Deprecation -- The "old" version is deprecated but will continue to work. This is probably a transitional state.
- Warning -- Failure to adhere to these changes will result in a warning message, but otherwise proper operation.
- Altered or Incomplete Operation -- Failure to adhere to these changes will result in *something happening*, but not necessarily precisely the same thing (as before).
- New Response -- A new response type(s) may be returned. Callers should be prepared for the new response.
- New Faults -- A new fault type(s) may be returned. Callers should be prepared for this eventuality.
- Pricing -- No change in operation, but cost per unit will change.
- Availability -- Permanently the service's availability will change

3.3 Provision of a Time Line

Change management vocabularies must allow the definition of the time line at which the changes apply. Separate timing information may apply to portions of a change (*i.e.* a change may be optional this week, required next week).
Timeline Indicators within change management vocabularies must be provided in both machine readable and human form (if a single format will not suffice.) This timeline indicator is used by a tool (or client or service) to determine when a change must be implemented. For example, a tool that generated a service call might generate different syntax for the call depending upon the time it was run (in comparison to the timeline information). A more sophisticated tool might generate code whose behavior varied depending upon the time at which the code was invoked.

[Recommend that the time formats defined in XML Schema are sufficiently human readable that we recommend the use for both human-readable and tool-readable time lines.]

The time line should allow for duration. This may be of use for separate impact statements (*Deprecated* 01-Dec-2002:15-Dec-2002; *Failure* 16-Dec-2002), or for temporary availability changes.

4 Human-Oriented Requirements

This section contains the requirements placed on descriptions by human consumers of the information. These are distinct from the machine-based requirements described in the tool, client, and service requirements.

4.1 Provision for a human-readable Description

Change management vocabularies must provide for human readable (descriptive) text. This must be capable of being linked with any technical information about the change. The vocabulary should support the reason for change.

4.2 Facilitate Relationships

Change Detail, Types, Impacts are related. Vocabularies should facilitate these relationships.

5 Tool-Oriented Requirements

5.1 Design for Machine Processability

Change management vocabularies must encode information in a machine-readable format. That is, in addition to being XML, well-defined concepts must have a defined representation so that tools (or, by implication, clients and services) can make use of the information.

5.2 Provision for Protocol Change Information

Change management vocabularies must provide information about changes to access. This includes binding, endpoint, port type, URL, MEP (Message Exchange Patterns -- one-way, request/response, etc.), etc.

5.3 Provision for Syntactic Change information

Change management vocabularies must provide for the definition of changes to message syntax. This may be done at a high-level, specifying only that the "old" version used schema xyz and the "new" version uses schema xyz1. However, change management vocabularies should provide the ability to annotate this to a lower level, allowing for definitions down to the element or attribute level. This will allow tools, clients, and servers to make use of this low-level information in impact analysis, change planning, and/or automatic code or transformation generation.

5.4 Provision for Old/New Linkage

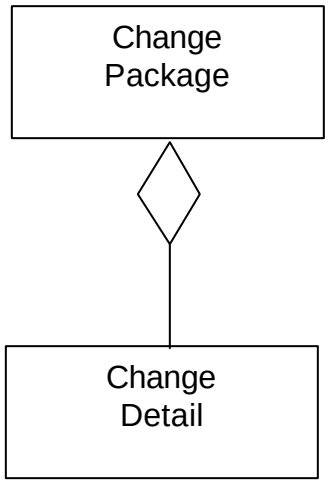
Change management vocabularies must provide syntactic information in such a way that old and new syntactic components can be linked to one another. For example, if an element or attribute is renamed, a linkage between these with a "rename" indication is far preferable to a "removed old element, added new element" type of description. The former provides information regarding translation semantics; the latter provides little information regarding the transition between these changes. To be of value, change management vocabularies should allow for the largest possible information transfer.

5.5 Provision for Security change Information

Change management vocabularies must provide information regarding security changes to its target. This includes, but is not limited to authentication methods, authorization required, non-repudiation requirements (digital signatures), privacy requirements (encoding, etc.).

5.6 Provision for grouping of changes

Change management vocabularies should provide for grouping of a number of changes into a Change Package. The change Package will have an impact indicator, time line, type of change, etc. Such a feature is of value for the consumer in 1) analyzing the impact and implementation requirement, and 2) scheduling the work.



The association of a group of Change Detail with a Change Package can be done by reference or embedding. That is, a Change Detail could reference a URL for a Change Package document, or it could embed the Change Package element in its entirety.
A Change Package could contain the following elements:

- Name – A human readable name
- A global unique ID (machine assigned). Any Change Detail associated with this Change Package will reference the same ID.
- Description – a textual description of the change (Human readable)
- Motivation – Why the change was needed, perhaps a reference to an issue and the requestor of the change. (Human readable)
- Implementer of the change – What organization has implemented the change
- Expected effective date – Serves as a default, though can be overridden by dates in the Change Detail (expressed in a universal time format)
- Prerequisites – Earlier Change Details or software levels that must be in place to accommodate the change, if any.
- XML-Signatures of the organizations involved in the change, including the role, for example: implementer of the change, the one who communicates the change. These are to provide some confidence that a change notification is legitimate.

5.7 Provision for detailed Impact indicators

Change management vocabularies should allow individual changes to have separate impact statements independent of any type of package or grouping. The Change Package as a whole may have an impact and the separate changes may have their own impact. There will need to be a clear model to define this.

6 Client Requirements

6.1 Provision for Change Notification

Change manage vocabularies should allow for the provision of change information as part of normal interaction with the Web service via message packaging, etc.
Change management vocabularies must provide mappings to web service protocols (must provide SOAP, should provide others as appropriate) to define for notification of impending changes. That is, there will be a mechanism (educated SOAP guess: a well-defined SOAP header entry) by which a client can recognize a service it uses is advertising an upcoming change. Upon such notification, a client may choose to obtain change information or inform its support staff. Optionally, the alert may include the change description itself, or simply provide information on obtaining it.

6.2 Runtime Support for Version Changes

Change management version indication (deprecation schedules, available new versions, etc) needs supported in the runtime to indicate to the caller that an aspect of service (service-level, operation-level, etc) versioning has changed. This runtime support needs to accommodate all behaviors of success/fail.

7 Service Requirements

7.1 Provision for cooperate with Service Registry and Description Standards

Change management should consider defining usage guidelines with UDDI, WSIL, and WSDL. [Educated guess:] These may take the form of extensions to WSIL and WSDL, and some use of UDDI to indicate versioning information. This will allow consumers of the service registry and description standards to make use of change information through their normal course of interaction.

7.2 Provision for Versioning

- Change management vocabularies must provided versioning structures and semantics.

7.3 Provision for Pre-requisites / Co-requisites

Prerequisites – Earlier Change Packages or software levels that must be in place to accommodate the change, if any.. Each change may have prerequisite/corequisite changes that must be applied for this change to work.