Cache Operation
Version 31-Jul-2001

Wireless Application Protocol
WAP-175-CacheOp-20010731-a
Contents

1. SCOPE (INFORMATIVE)........................................................................................................... 4
2. REFERENCES .......................................................................................................................... 5
  2.1. NORMATIVE REFERENCES ............................................................................................. 5
  2.2. INFORMATIVE REFERENCES ......................................................................................... 5
3. TERMINOLOGY AND CONVENTIONS .................................................................................. 6
  3.1. CONVENTIONS ................................................................................................................ 6
  3.2. DEFINITIONS ................................................................................................................... 6
  3.3. ABBREVIATIONS .............................................................................................................. 7
4. INTRODUCTION (INFORMATIVE) .......................................................................................... 8
5. CACHE OPERATION CONTENT FORMAT (NORMATIVE) ..................................................... 9
  5.1. THE CO ELEMENT ......................................................................................................... 9
  5.2. THE INVALIDATE-OBJECT ELEMENT ........................................................................... 9
  5.3. THE INVALIDATE-SERVICE ELEMENT ......................................................................... 9
6. SEMANTICS (NORMATIVE) ..................................................................................................... 10
  6.1. BASIC OPERATIONS ...................................................................................................... 10
  6.2. RELATIVE URI RESOLUTION ......................................................................................... 10
  6.3. URI EQUIVALENCE ......................................................................................................... 10
  6.4. PREFIX MATCH .............................................................................................................. 10
7. SECURITY (NORMATIVE) ...................................................................................................... 11
8. CO REFERENCE INFORMATION (NORMATIVE) ................................................................... 12
  8.1. DOCUMENT IDENTIFIERS ............................................................................................ 12
    8.1.1. SGML Public Identifier ........................................................................................... 12
    8.1.2. CO Media Type ...................................................................................................... 12
  8.2. DOCUMENT TYPE DEFINITION (DTD) ........................................................................ 12
9. COMPACT BINARY REPRESENTATION OF CACHE OPERATION (NORMATIVE)................. 13
  9.1. EXTENSION TOKENS ....................................................................................................... 13
    9.1.1. Tag Tokens ................................................................................................................ 13
    9.1.2. Attribute Tokens ...................................................................................................... 13
  9.2. ENCODING SEMANTICS ................................................................................................. 13
    9.2.1. Document Validation ............................................................................................... 13
  9.3. NUMERIC CONSTANTS .................................................................................................. 13
    9.3.1. Tag Tokens ................................................................................................................ 13
    9.3.2. Attribute Start Tokens .............................................................................................. 13
    9.3.3. Attribute Value Tokens ............................................................................................ 14
10. EXAMPLE (INFORMATIVE) ................................................................................................ 15

APPENDIX A. STATIC CONFORMANCE REQUIREMENTS (NORMATIVE) .................................. 17
  A.1. CLIENT FEATURES ........................................................................................................... 17
    A.1.1 Content Format and Tokenisation ............................................................................. 17
    A.1.2 Semantics ................................................................................................................... 17
    A.1.3 Security ..................................................................................................................... 17
  A.2. PUSH PROXY GATEWAY FEATURES ......................................................................... 17
    A.2.1 General ..................................................................................................................... 17
    A.2.2 Validation .................................................................................................................. 17

APPENDIX B. CHANGE HISTORY (INFORMATIVE) ................................................................... 18

All rights reserved
1. Scope

Wireless Application Protocol (WAP) is a result of continuous work to define an industry wide specification for developing applications that operate over wireless communication networks. The scope for the WAP Forum is to define a set of specifications to be used by service applications. The wireless market is growing very quickly and reaching new customers and providing new services. To enable operators and manufacturers to meet the challenges in advanced services, differentiation, and fast/flexible service creation, WAP defines a set of protocols in transport, session and application layers. For additional information on the WAP architecture, refer to “Wireless Application Protocol Architecture Specification” [WAPARCH].

This specification defines a content type, the cache operation, which allows applications to invalidate the content cached in the user agent. It addresses the class of situations where the expiration time of the content to be cached cannot be predicted.
2. References

2.1. Normative References


2.2. Informative References


3. Terminology and Conventions

3.1. Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2. Definitions

**Application** - A value-added data service provided to a WAP Client. The application may utilise both push and pull data transfer to deliver content.

**Application-Level Addressing** - the ability to address push content between a particular user agent on a WAP client and push initiator on a server.

**Client** – in the context of push, a client is a device (or service) that expects to receive push content from a server. In the context of pull a client, it is a device initiates a request to a server for content or data. See also “device”.

**Content** - subject matter (data) stored or generated at an origin server. Content is typically displayed or interpreted by a user agent on a client. Content can both be returned in response to a user request, or being pushed directly to a client.

**Content Encoding** - when used as a verb, content encoding indicates the act of converting a data object from one format to another. Typically the resulting format requires less physical space than the original, is easier to process or store, and/or is encrypted. When used as a noun, content encoding specifies a particular format or encoding standard or process.

**Content Format** – actual representation of content.

**Context** – an execution space where variables, state and content are handled within a well-defined boundary.

**Device** – is a network entity that is capable of sending and/or receiving packets of information and has a unique device address. A device can act as either a client or a server within a given context or across multiple contexts. For example, a device can service a number of clients (as a server) while being a client to another server.

**End-user** - see “user”

**Extensible Markup Language** - is a World Wide Web Consortium (W3C) recommended standard for Internet markup languages, of which WML is one such language. XML is a restricted subset of SGML.

**Push Framework** - the entire WAP push system. The push framework encompasses the protocols, service interfaces, and software entities that provide the means to push data to user agents in the WAP client.

**Push Initiator** - the entity that originates push content and submits it to the push framework for delivery to a user agent on a client.

**Push Proxy Gateway** - a proxy gateway that provides push proxy services.

**Push Session** - A WSP session that is capable of conducting push operations.

**Server** - a device (or service) that passively waits for connection requests from one or more clients. A server may accept or reject a connection request from a client. A server may initiate a connection to a client as part of a service (push).

**User** - a user is a person who interacts with a user agent to view, hear, or otherwise use a rendered content. Also referred to as end-user.
User agent - a user agent (or content interpreter) is any software or device that interprets resources. This may include textual browsers, voice browsers, search engines, etc.

XML – see Extensible Markup Language

3.3. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>Cache Operation</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IANA</td>
<td>Internet Assigned Numbers Authority</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>OTA</td>
<td>Over The Air</td>
</tr>
<tr>
<td>PI</td>
<td>Push Initiator</td>
</tr>
<tr>
<td>PPG</td>
<td>Push Proxy Gateway</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RFC</td>
<td>Request For Comments</td>
</tr>
<tr>
<td>SGML</td>
<td>Standard Generalized Markup Language</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WAP</td>
<td>Wireless Application Protocol</td>
</tr>
<tr>
<td>WBXML</td>
<td>WAP Binary XML</td>
</tr>
<tr>
<td>WSP</td>
<td>Wireless Session Protocol</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Mark-up Language</td>
</tr>
</tbody>
</table>
4. Introduction

The Cache Operation provides a means to invalidate content objects in the user agent cache. The invalidated content objects must be reloaded from the origin server the next time they are needed. The Cache Operation is useful for a class of situations when an application cannot predict the expiration time of the content it creates.

A typical example is a mailbox application. The content of the mailbox can change unpredictably as a subscriber receives new messages. Whenever this occurs, the application can issue a cache invalidation operation to inform the user agent that the mailbox content has expired. The next time the user views the mailbox, the most recent version will be requested from the origin server. Use of the Cache Operation can be quite efficient when messages arrive more frequently than users check their mailboxes. Sending a cache operation causes much less network load than sending the content of the updated mailbox on every change.

The Cache Operation content type is specified as an XML document. It is used to indicate that cache objects with the given URIs or objects with the same URI prefix are no longer valid.

There are two operations: invalidate object and invalidate service,

- invalidate object - invalidate the object uniquely identified by the given URI
- invalidate service - invalidate all the objects that share the same URI prefix
5. Cache Operation Content Format (Normative)

This section defines the content format used to represent the Cache Operation (CO), which is an application of XML version 1.0 [XML]. The complete Cache Operation DTD is defined in section 8, which an implementation conforming to this specification MUST support.

5.1. The CO Element

\[
\text{<!ELEMENT co (invalidate-object | invalidate-service)+>}
\]

The cache operation contains one or more invalidate-object and/or invalidate-service elements.

5.2. The invalidate-object Element

\[
\text{<!ELEMENT invalidate-object EMPTY>}
\]

\[
\text{<!ATTLIST invalidate-object}
\text{ uri CDATA #REQUIRED >}
\]

The invalidate-object element specifies that a single cached object whose URI matches the specified uri must be invalidated.

**Attributes:**

\[
\text{uri=CDATA}
\]

The uri attribute specifies the URI of the cached object to invalidate. The uri attribute may be relative or absolute.

5.3. The invalidate-service Element

\[
\text{<!ELEMENT invalidate-service EMPTY>}
\]

\[
\text{<!ATTLIST invalidate-service}
\text{ uri CDATA #REQUIRED >}
\]

The invalidate-service element requests that all the cached objects with a URI prefix which matches the specified uri must be invalidated.

**Attributes:**

\[
\text{uri=CDATA}
\]

The uri attribute specifies the URI prefix of the cached objects to invalidate. The uri attribute may be relative or absolute.
6. Semantics

6.1. Basic Operations

A Cache Operation requests the user agent to invalidate one or multiple cache objects that may currently reside in the user agent cache when the CO is received. It can be either pushed (i.e., server initiated) or pulled (i.e., client initiated). The effect of the operation is the same regardless of how it is transferred into the user agent.

Invalidation prevents the cached objects from being re-used. The user agent MUST NOT use an invalidated object in any way (e.g., presenting it to the user).

If no object(s) can be identified by using the match rules as defined in section 6.3, the CO MUST be silently discarded.

A CO may be discarded under certain security considerations (see section 6.3).

If a Date header [PushMsg] is present, the client MAY use it to prevent a delayed CO from invalidating fresh objects. A CO is delayed when its Date header indicates an earlier time than the Date or Last-Modified headers of a cached object. If the CO involves multiple cached objects, this check is applied to each cached object individually. If a CO is delayed with respect to a cached object, it MUST NOT have any effect on the cached object.

6.2. Relative URI Resolution

If a URI in the CO is relative, the base URI is determined according to the rules in [HTTP] and [RFC2396] with the exception that X-Wap-Content-URI header [PushMsg] is substituted for the Request-URI [HTTP] in the context of push. If the base URI for a relative URI in an invalidate element cannot be determined, the element MUST be ignored.

Furthermore, [RFC2396] defines additional rules for resolving a relative URI to an absolute URI after the base URI has been determined. These MUST be applied to resolve the relative URI in the CO.

6.3. URI Equivalence

The URI Comparison rules in HTTP/1.1 [HTTP] MUST be used to decide if two URIs match. URI Equivalence rule MUST used by the invalidate object operation.

6.4. Prefix Match

The rules below are used for URI prefix match. Those rules MUST be used by the invalidate service operation. The applicable exception rules for URI Comparison in HTTP/1.1 [HTTP] must be applied when the following rules are utilised.

- The scheme [RFC2396] of the specified URI in the CO must match the scheme of the URI for the cached object.
- The authority [RFC2396] of the specified URI in the CO must match the authority of the URI for the cached object. If the authority of the specified URI in the CO is not present, there is a match only if the authority of the URI for the cached object is not present either.
- The path of the specified URI in the CO must match the partial or the complete path of the URI for the cached object, if the specified URI in the CO has the path. The match is performed at the segment level (e.g., /abc/ef matches /abc/ef/gh, but not /abc/efz/mnop) and starts at the leftmost segment in the path for abs_path and the entire path for opaque_part [RFC2396].
- The query component is ignored, if it is present in the specified URI in the CO.
7. Security

A user agent which supports the Cache Operation may be subject to denial of service attacks. Denial of service attacks neither risk the privacy and integrity of the cached objects nor change the persistent state of the user agent, but they may adversely affect the performance of the user agent because it may be forced to reload still valid objects that have been falsely invalidated.

To protect against denial of service attacks, the user agent SHOULD provide a means to filter out those COs that cannot be proved to be from a trusted or authenticated source. No matter what method the user agent uses for this purpose, the user agent SHOULD accept the CO under one of the following conditions for the pushed CO:

- The Trusted flag is present.
- The Authenticated flag is present and the authority [RFC2396] of the URI in the CO matches the authority of the URI in the value of the X-Wap-Initiator header [PushMsg].

Likewise, the user agent SHOULD accept the CO under the following condition for the pulled CO,

- The authority [RFC2396] of URI in the CO matches the authority of the URI in the value of the Request-URI [HTTP].

The origin server should understand that the CO may be discarded if the above conditions are not met or if implementation dependent means of protection are used in the user agent. As a consequence, a user agent may keep on using the cached objects which are actually invalid. The origin server can alleviate this situation by using an Expires header [PushMsg] even on objects that it expects to invalidate explicitly by using the CO. The value of the Expires headers should be based on the estimated maximum expiration time for the cached object.
8. CO Reference Information

Cache Operation (CO) is an application of XML version 1.0 [XML].

8.1. Document Identifiers

8.1.1. SGML Public Identifier

Note: This identifier has not yet been registered with the IANA or ISO 9070 registrar

`-//WAPFORUM//DTD CO 1.0//EN`

8.1.2. CO Media Type

Note: These types are not yet registered with the IANA, and are consequently experimental media types.

Textual form:

`text/vnd.wap.co`

Tokenised form:

`application/vnd.wap.coc`

8.2. Document Type Definition (DTD)

```xml
<!--
Cache Operation (CO) Document Type Definition.
CO is an XML language. Typical usage:
  <?xml version="1.0"?>
  <!DOCTYPE co PUBLIC "-//WAPFORUM//DTD CO 1.0//EN" "http://www.wapforum.org/DTD/co_1.0.dtd">
  <co>
    ...
  </co>
-->

<!------------------------ The co Element ------------------------>
<!ELEMENT co (invalidate-object | invalidate-service)>`

<!------------------------ The invalidate-object Element ------------------------>
<!ELEMENT invalidate-object EMPTY>
<!ATTLIST invalidate-object
  uri CDATA #REQUIRED`

```

<!------------------------ The invalidate-service Element ------------------------>
<!ELEMENT invalidate-service EMPTY>
<!ATTLIST invalidate-service
  uri CDATA #REQUIRED>`
9. Compact Binary Representation of Cache Operation (Normative)

The CO content format MAY be encoded using a compact binary representation. This content format is based upon the WAP Binary XML Content Format [WBXML].

9.1. Extension Tokens

9.1.1. Tag Tokens

CO defines a set of single-byte tokens corresponding to the tags defined in the DTD. All of these tokens are defined within code page zero.

9.1.2. Attribute Tokens

CO defines a set of single-byte tokens corresponding to the attribute names and values defined in the DTD. All of these tokens are defined within code page zero.

9.2. Encoding Semantics

9.2.1. Document Validation

XML document validation (see [XML]) SHOULD occur during the process of tokenising a CO and, if done, it MUST be based on the DOCTYPE declared in the CO. When validating the source text, the tokenisation process MUST accept any DOCTYPE or public identifier, if the document is identified as a CO media type (section 8.1.2).

The tokenisation process MUST check that the source CO is XML well-formed, and it SHOULD notify the end user (in case of pull) or the push initiator (in case of push) of any well-formedness or validity errors detected in the source CO.

9.3. Numeric Constants

9.3.1. Tag Tokens

The following token codes represent tags in code page zero (0). All numbers are in hexadecimal.

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>co</td>
<td>5</td>
</tr>
<tr>
<td>invalidate-object</td>
<td>6</td>
</tr>
<tr>
<td>invalidate-service</td>
<td>7</td>
</tr>
</tbody>
</table>

9.3.2. Attribute Start Tokens

The following token codes represent the start of an attribute in code page zero (0). All numbers are in hexadecimal.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attribute Value Prefix</th>
<th>Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>uri</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>uri</td>
<td>http://</td>
<td>6</td>
</tr>
<tr>
<td>uri</td>
<td><a href="http://www">http://www</a>.</td>
<td>7</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Attribute Value Prefix</td>
<td>Token</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>uri</td>
<td>https://</td>
<td>8</td>
</tr>
<tr>
<td>uri</td>
<td><a href="https://www">https://www</a>.</td>
<td>9</td>
</tr>
</tbody>
</table>

9.3.3. Attribute Value Tokens

The following token codes represent attribute values in code page zero (0). All numbers are in hexadecimal.

<table>
<thead>
<tr>
<th>Attribute Value</th>
<th>Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>.com/</td>
<td>85</td>
</tr>
<tr>
<td>.edu/</td>
<td>86</td>
</tr>
<tr>
<td>.net/</td>
<td>87</td>
</tr>
<tr>
<td>.org/</td>
<td>88</td>
</tr>
</tbody>
</table>
10. Example

The example below illustrates how a CO can be tokenised.

```xml
<?xml version="1.0"?>
<!DOCTYPE co PUBLIC "-//WAPFORUM//DTD CO 1.0//EN"
"http://www.wapforum.org/DTD/co_1.0.dtd">
<co>
  <invalidate-object uri="foo.wml"></invalidate-object>
  <invalidate-service uri="/bar"></invalidate-service>
</co>
```

For this example, let’s assume that the value in the Content-Location header is `/abc/` and the value in the X-Wap-Content-URI header is `http://www.xyz.com/`. The base URI is resolved as `http://www.xyz.com/abc/`. The outcome of this CO may be to invalidate a single cached object as identified by `http://www.xyz.com/abc/foo.wml` and multiple cached objects which have the URI prefix `http://www.xyz.com/bar`.

The tokenised form of this example (numbers in hexadecimal), using the WBXML encoding defined in section 9, is found below. The textual CO consists of about 150 octets, while the tokenised form consists of 27 octets.

```
02 07 6A 00 45 86 05 03 'f' 'o' 'o' '.' 'w' 'm' 'l'  \0 01 87 05 03 '/ ' 'b' 'a' 'r' ' \0' 01 01
```

In an expanded and annotated form:

<table>
<thead>
<tr>
<th><strong>Token Stream</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>Version number - WBXML version 1.2</td>
</tr>
<tr>
<td>07</td>
<td>CO 1.0 Public Identifier</td>
</tr>
<tr>
<td>03</td>
<td>Charset=&quot;US-ASCII&quot;</td>
</tr>
<tr>
<td>00</td>
<td>String table length</td>
</tr>
<tr>
<td>45</td>
<td>co with content</td>
</tr>
<tr>
<td>86</td>
<td>invalidate-object with attributes</td>
</tr>
<tr>
<td>05</td>
<td>uri =</td>
</tr>
<tr>
<td>03</td>
<td>Inline string follows</td>
</tr>
<tr>
<td>'/ ', 'o', 'o', '.', 'w', 'm', 'l', '\0'</td>
<td>String</td>
</tr>
<tr>
<td>01</td>
<td>END (of invalidate-object attribute list)</td>
</tr>
<tr>
<td>87</td>
<td>invalidate-service with attributes</td>
</tr>
<tr>
<td>05</td>
<td>uri =</td>
</tr>
<tr>
<td>03</td>
<td>Inline string follows</td>
</tr>
<tr>
<td>'/ ', 'b', 'a', 'r', '\0'</td>
<td>String</td>
</tr>
<tr>
<td>01</td>
<td>END (of invalidate-service attribute list)</td>
</tr>
<tr>
<td><strong>Token Stream</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>01</td>
<td>END (of co element)</td>
</tr>
</tbody>
</table>
Appendix A. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [CREQ].

A.1. Client Features

A.1.1 Content Format and Tokenisation

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Status</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO-CF-C-001</td>
<td>Support for CO in textual form (text/vnd.wap.co)</td>
<td>5</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>CO-CF-C-002</td>
<td>Support for CO in tokenised form (application/vnd.wap.coc)</td>
<td>9</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CO-CF-C-003</td>
<td>Support for the CO token table</td>
<td>9.3</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>

A.1.2 Semantics

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Status</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO-SEM-C-001</td>
<td>Invalidate object</td>
<td>6</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CO-SEM-C-002</td>
<td>Invalidate service</td>
<td>6</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CO-SEM-C-003</td>
<td>Use URI Equivalence rule</td>
<td>6.3</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CO-SEM-C-004</td>
<td>Use Prefix Match rule</td>
<td>6.4</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CO-SEM-C-005</td>
<td>Handling relative URI</td>
<td>6.2</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>

A.1.3 Security

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Status</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO-SEC-C-001</td>
<td>Protection for the denial of service attacks</td>
<td>7</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>CO-SEC-C-002</td>
<td>Use Acceptance Conditions for CO</td>
<td>7</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>

A.2. Push Proxy Gateway Features

A.2.1 General

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Status</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO-PPG-S-001</td>
<td>Send a CO to client in textual form (text/vnd.wap.co)</td>
<td>5</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CO-PPG-S-002</td>
<td>Send a CO to client in tokenised form (application/vnd.wap.coc)</td>
<td>9</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CO-PPG-S-003</td>
<td>Support for the CO token table.</td>
<td>9.3</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>

A.2.2 Validation

<table>
<thead>
<tr>
<th>Item</th>
<th>Function</th>
<th>Reference</th>
<th>Status</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO-VAL-S-001</td>
<td>XML well-formed</td>
<td>9.2.1</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CO-VAL-S-002</td>
<td>XML validation</td>
<td>9.2.1</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>
## Appendix B. Change History (Informative)

<table>
<thead>
<tr>
<th>Type of Change</th>
<th>Date</th>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAP-175-CacheOp-19991206-a</td>
<td>06-Dec-1999</td>
<td>All</td>
<td>The initial version of this document.</td>
</tr>
<tr>
<td>WAP-175_100-CacheOp-20010420-a</td>
<td>20-Apr-2001</td>
<td>Appendix A</td>
<td>SCR SIN</td>
</tr>
<tr>
<td>WAP-175_101-CacheOp-20010206-a</td>
<td>06-Feb-2001</td>
<td>10</td>
<td>Corrects errors in Example</td>
</tr>
<tr>
<td>Class 3</td>
<td>31-Jul-2001</td>
<td>All</td>
<td>WAP 2.0 roll-up and template</td>
</tr>
</tbody>
</table>