

WAP Cache Operation

Version 06-Dec-1999

Wireless Application Protocol Cache Operation Specification

Notice:

© Wireless Application Protocol Forum, Ltd. 1999.

Terms and conditions of use are available from the Wireless Application Protocol Forum Ltd. Web site
(<http://www.wapforum.org/what/copyright.htm>).

Disclaimer:

This document is subject to change without notice.

Contents

1. SCOPE	4
2. DOCUMENT STATUS	5
2.1 COPYRIGHT NOTICE	5
2.2 ERRATA	5
2.3 COMMENTS	5
3. REFERENCES.....	6
3.1 NORMATIVE REFERENCES.....	6
3.2 INFORMATIVE REFERENCES	6
4. DEFINITIONS AND ABBREVIATIONS.....	7
4.1 DEFINITIONS	7
4.2 ABBREVIATIONS	9
5. INTRODUCTION	10
6. CACHE OPERATION CONTENT FORMAT	11
6.1 THE CO ELEMENT	11
6.2 THE INVALIDATE-OBJECT ELEMENT	11
6.3 THE INVALIDATE-SERVICE ELEMENT	11
7. SEMANTICS.....	12
7.1 BASIC OPERATIONS	12
7.2 RELATIVE URI RESOLUTION.....	12
7.3 URI EQUIVALENCE	12
7.4 PREFIX MATCH.....	12
8. SECURITY	13
9. CO REFERENCE INFORMATION.....	14
9.1 DOCUMENT IDENTIFIERS	14
9.1.1 SGML Public Identifier.....	14
9.1.2 CO Media Type.....	14
9.2 DOCUMENT TYPE DEFINITION (DTD)	14
10. COMPACT BINARY REPRESENTATION OF CACHE OPERATION.....	15
10.1 EXTENSION TOKENS	15
10.1.1 Tag Tokens.....	15
10.1.2 Attribute Tokens.....	15
10.2 ENCODING SEMANTICS.....	15
10.2.1 Document Validation.....	15
10.3 NUMERIC CONSTANTS	15
10.3.1 Tag Tokens.....	15
10.3.2 Attribute Start Tokens.....	15
10.3.3 Attribute Value Tokens	16
11. EXAMPLE.....	17
12. STATIC CONFORMANCE REQUIREMENTS.....	19
12.1 CLIENT FEATURES	19
12.1.1 Content Format and Tokenisation.....	19

12.1.2 Semantics 19

12.1.3 Security 19

12.2 PUSH PROXY GATEWAY FEATURES..... 20

12.2.1 General 20

12.2.2 Validation..... 20

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*” [WAP].

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. Document Status

This document is available online in the following formats:

- PDF format at <http://www.wapforum.org/>.

2.1 Copyright Notice

© Copyright Wireless Application Forum Ltd, 1999.

Terms and conditions of use are available from the Wireless Application Protocol Forum Ltd. web site at <http://www.wapforum.org/docs/copyright.htm>.

2.2 Errata

Known problems associated with this document are published at <http://www.wapforum.org/>.

2.3 Comments

Comments regarding this document can be submitted to the WAP Forum in the manner published at <http://www.wapforum.org/>.

3. References

3.1 Normative references

- [RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL: <http://www.ietf.org/rfc/rfc2119.txt>
- [WBXML] "WAP Binary XML Content Format", WAP Forum, 04-Nov-1999, URL: <http://www.wapforum.org/>
- [XML] "Extensible Markup Language (XML)", W3C Recommendation 10-February-1998, REC-xml-19980210", T. Bray, et al, February 10, 1998, URL: <http://www.w3.org/TR/REC-xml>
- [RFC2396] "Uniform Resource identifiers (URI): Generic Syntax", T. Berners-Lee, et al., August 1998, URL: <http://www.ietf.org/rfc/rfc2396.txt>
- [HTTP] "Hypertext Transfer Protocol – HTTP/1.1", R. Fielding, et al. June 1999, URL: <http://www.ietf.org/rfc/rfc2616.txt>
- [PushOTA] "WAP Push OTA Specification", WAP Forum, 08-Nov-1999, URL:<http://www.wapforum.org/>
- [PushMsg] "WAP Push Message Specification", WAP Forum, 16-August-1999, URL: <http://www.wapforum.org/>

3.2 Informative references

- [PushArch] "WAP Push Architecture Overview", WAP Forum, 08-Nov-1999,URL:<http://www.wapforum.org/>
- [WAP] "Wireless Application Protocol Architecture Specification", WAP Forum, 30-April-1998, URL: <http://www.wapforum.org/>
- [WAE] "Wireless Application Environment Overview", WAP Forum, 04-Nov-1999, URL: <http://www.wapforum.org/>

4. Definitions and Abbreviations

4.1 Definitions

The following are terms and conventions used throughout this specification.

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 by [RFC2119].

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.

Bearer Network - a network used to carry the messages of a transport-layer protocol between physical devices. Multiple bearer networks may be used over the life of a single push session.

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”.

Contact Point – address information that describes how to reach a push proxy gateway, including transport protocol address and port of the push proxy gateway.

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 mark-up languages, of which WML is one such language. XML is a restricted subset of SGML.

Multicast Message - a push message containing a single OTA client address which implicitly specifies more than OTA client address.

Push Access Protocol - a protocol used for conveying content that should be pushed to a client, and push related control information, between a Push Initiator and a Push Proxy/Gateway.

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 OTA Protocol - a protocol used for conveying content between a Push Proxy/Gateway and a certain 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*

4.2 Abbreviations

For the purposes of this specification, the following abbreviations apply.

CO	Cache Operation
CPI	Capability and Preference Information
DNS	Domain Name Server
DTD	Document Type Definition
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
OTA	Over The Air
PAP	Push Access Protocol
PI	Push Initiator
PPG	Push Proxy Gateway
QOS	Quality of Service
RDF	Resource Description Framework
RFC	Request For Comments
SGML	Standard Generalized Markup Language
SI	Service Indication
SIA	Session Initiation Application
SIR	Session Initiation Request
SL	Service Loading
SSL	Secure Socket Layer
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Time Co-ordinated
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WBXML	WAP Binary XML
WINA	WAP Interim Naming Authority
WTLS	Wireless Transport Layer Security
XML	Extensible Mark-up Language

5. 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

6. Cache Operation Content Format

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 9, which an implementation conforming to this specification MUST support.

6.1 The CO Element

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

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

6.2 The invalidate-object Element

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

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

Attributes:

`uri=CDATA`

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

6.3 The invalidate-service Element

```
<!ELEMENT invalidate-service EMPTY>
<!ATTLIST invalidate-service
  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:

`uri=CDATA`

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

7. Semantics

7.1 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 7.3, the CO **MUST** be silently discarded.

A CO may be discarded under certain security considerations (see section 8).

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.

7.2 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.

7.3 URI Equivalence

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

7.4 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/mn*) 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.

8. 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 can not 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 [PushOTA] is present.
- The *Authenticated* flag [PushOTA] is present and the *authority* [RFC2396] of the URI in the CO matches the *authority* of the URI in the value of *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.

9. CO Reference Information

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

9.1 Document Identifiers

9.1.1 SGML Public Identifier

Editor's note: This identifier has not yet been registered with the IANA or ISO 9070 registrar

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

9.1.2 CO Media Type

Editor's 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

9.2 Document Type Definition (DTD)

```
<!--
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
>
```

10. Compact Binary Representation of Cache Operation

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].

10.1 Extension Tokens

10.1.1 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.

10.1.2 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.

10.2 Encoding Semantics

10.2.1 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 9.1.2).

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.

10.3 Numeric Constants

10.3.1 Tag Tokens

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

<u>Tag Name</u>	<u>Token</u>
co	5
invalidate-object	6
invalidate-service	7

10.3.2 Attribute Start Tokens

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

<u>Attribute Name</u>	<u>Attribute Value Prefix</u>	<u>Token</u>
uri		5
uri	http://	6
uri	http://www.	7
uri	https://	8
uri	https://www.	9

10.3.3 Attribute Value Tokens

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

<u><i>Attribute Value</i></u>	<u><i>Token</i></u>
.com/	85
.edu/	86
.net/	87
.org/	88

11. Example

The example below illustrates how a CO can be tokenised.

```
<?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 10, is found below. The textual CO consists of about 150 octets, while the tokenised form consists of 27 octets.

```
00 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:

<u>Token Stream</u>	<u>Description</u>
00	Version number - WBXML version 1.0
07	CO 1.0 Public Identifier
03	Charset="US-ASCII"
00	String table length
45	co with content
86	invalidate-object with attributes
05	uri =
03	Inline string follows
'f', 'o', 'o', '.', 'w', 'm', 'l', '\0'	String
01	END (of invalidate-object)
87	invalidate-service with attributes
05	uri =
03	Inline string follows
('/', 'b', 'a', 'r', '\0'	String
01	END (of invalidate-service element)

<u>Token Stream</u>	<u>Description</u>
01	END (of CO element)

12. Static Conformance Requirements

This static conformance clause defines a minimum set of features that should be implemented to support WAP CO content type. A feature can be optional (O), mandatory (M) or conditional (C (<condition>)). If optional/conditional features have labels (O.<n> or C.<n>), support of at least one in the group of options labelled by the same number is required.

12.1 Client Features

12.1.1 Content Format and Tokenisation

Item	Functionality	Reference	Status
COC_C001	Support for CO in textual form (text/vnd.wap.co)	6	O
COC_C002	Support for CO in tokenised form (application/vnd.wap.coc)	10	M
COP_C003	Support for the CO token table	10.3	M

12.1.2 Semantics

Item	Functionality	Reference	Status
COC_S010	Invalidate object	7	M
COC_S020	Invalidate service	7	M
COC_S030	Use URI Equivalence rule	7.3	M
COC_S031	Use Prefix Match rule	7.4	M
COC_S040	Handling relative URI	7.2	M

12.1.3 Security

Item	Functionality	Reference	Status
COC_SE001	Protection for the denial of service attacks	8	O
COC_SE002	Use Acceptance Conditions for CO	8	O

12.2 Push Proxy Gateway Features

12.2.1 General

Item	Functionality	Reference	Status
COP_G001	Send a CO to client in textual form (text/vnd.wap.co)	6	O
COP_G002	Send a CO to client in tokenised form (application/vnd.wap.coc)	10	M
COP_G003	Support for the CO token table.	10.3	M

12.2.2 Validation

Item	Functionality	Reference	Status
COP_V001	XML well-formed	10.2.1	M
COP_V002	XML validation	10.2.1	M