Service Loading

Version 31-Jul-2001

Wireless Application Protocol
WAP-168-ServiceLoad-20010731-a

A list of errata and updates to this document is available from the WAP Forum™ Web site, http://www.wapforum.org/, in the form of SIN documents, which are subject to revision or removal without notice.

This document is available online in PDF format at http://www.wapforum.org/.

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

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

Document History	
WAP-168-ServiceLoad-19991108-a	Approved
WAP-168_100-ServiceLoad-20010423-a	SIN
WAP-168_101-ServiceLoad-20010206-a	SIN
WAP-168_102-ServiceLoad-20010614-a (obsoletes WAP-168_100-ServiceLoad-20010423-a)	SIN
WAP-168-ServiceLoad-20010731-a	Current

Contents

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].

It is not always suitable to push content that is executed or rendered (the term "executed" is used throughout this document to refer either to "executed" or "rendered") directly upon reception to a mobile device, especially if the client is busy with other activities such as executing another service. This is due to the fact that memory and/or processing constraints found in many mobile devices are such that neither storing nor processing content in such a situation is feasible.

Another disadvantage of pushing content directly to a mobile client is that this may interfere with an executing service that is utilising the bearer used for push, which will deteriorate the end-user experience of that service.

In order to circumvent these problems, the Service Loading (SL) content type has been defined. The content type provides a means to convey a URI to a user agent in a mobile client. The client itself automatically loads the content indicated by that URI and executes it in the addressed user agent without user intervention when appropriate. Thus, the end-user will experience the service indicated by the URI as if it was pushed to the client and executed. By basically conveying only the URI of the service to the client the over-the-air message will be small. Hence, very modest requirements are placed on the bearer and on the client's ability to receive and store an SL if it is busy with other activities. The disadvantage of this approach is that the number of over-the-air messages and round-trips needed to deliver the service will increase.

Instead of executing the service, SL provides a means to instruct the client to pre-emptively cache the content indicated by the URI so it becomes readily available to the user agent(s) in the client. It is also possible to control whether the loading of the service may be carried out in a user-intrusive manner or not.

The SL content type is an application of the Extensible Markup Language (XML) 1.0 [XML]. WBXML [WBXML] tokens are defined to allow for efficient over-the-air transmission.

2. References

2.1. Normative References

[CREQ] "Specification of WAP Conformance Requirements". WAP Forum™, WAP-221-CREQ-20010425-a, http//www.wapforum.org/

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. November 1997, http://www.ietf.org/rfc/rfc2234.txt

[WAE] "Wireless Application Environment Overview", WAP Forum™, WAP-153-WAESpec-19991104-a, http//www.wapforum.org/

[WBXML] "WAP Binary XML Content Format", WAP Forum™, WAP-154-WBXML-19991104-a, http//www.wapforum.org/

[WML] "Wireless Markup Language", WAP Forum™, WAP-155-WML-19991104-a, http//www.wapforum.org/

[XML] "Extensible Markup Language (XML)", W3C Recommendation 10-February-1998, REC-xml-19980210", T. Bray, et al, February 10, 1998, http://www.w3.org/TR/REC-xml

2.2. Informative References

[PushOTA] "WAP Push OTA Specification", WAP Forum™, WAP-189-PushOTA-20010731-a http//www.wapforum.org/

[PushPAP] "WAP Push Access Protocol Specification", WAP Forum™, WAP-164-PAP-20010731-a, http//www.wapforum.org/

[RFC2396] "Uniform Resource identifiers (URI): Generic Syntax", T. Berners-Lee, et al., August 1998. http://www.ietf.org/rfc/rfc2396.txt

[WAP] Wireless Application Protocol Architecture Specification", WAP Forum, WAP-100-WAPArch-19980430-a. http//www.wapforum.org/

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.
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 markup 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*

3.3. Abbreviations

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 Coordinated
WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WSP	Wireless Session Protocol
WBXML	Wireless Binary XML
WINA	WAP Interim Naming Authority
WTLS	Wireless Transport Layer Security
XML	Extensible Mark-up Language

4. Introduction

The Service Loading (SL) content type provides the ability to cause a user agent on a mobile client to load and execute a service that, for example, can be in the form of a WML deck. The SL contains a URI indicating the service to be loaded by the user agent without user intervention when appropriate.

The example below illustrates the procedure:

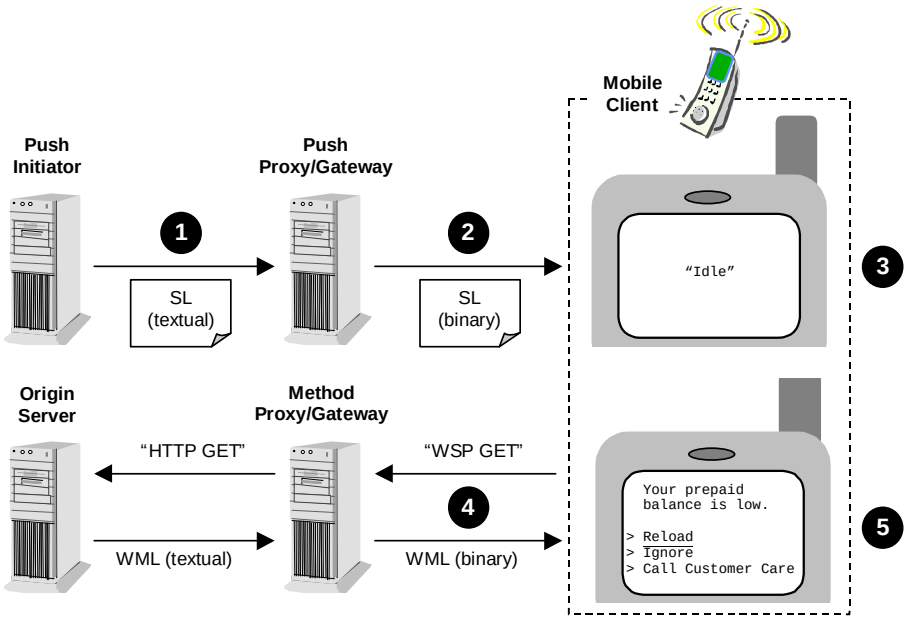

Figure 1 : Service Loading - the basic concept

The example illustrates how a mobile network operator chooses to force an end-user with a prepaid subscription to take action on his/her low balance by using SL to cause the user agent to load and execute the appropriate service (in the form of a WML deck). The following steps are involved:

1. The Push Initiator, in this case the mobile network operator, instructs the Push Proxy/Gateway to push an SL to the mobile client using the Push Access Protocol [PAP]. The Push Initiator provides the SL with the URI to the WML deck that should be executed in the client's user agent.
2. The Push Proxy/Gateway sends the SL to the mobile client using the Push OTA Protocol [OTA].
3. The mobile client receives the push containing the SL. The end-user is not made aware of this.
4. The service indicated by the SL's URI is retrieved ("pulled") from the origin server via the Method Proxy/ Gateway or optionally from the client's cache memory.
5. The service starts executing on the mobile client.

In addition to the basic functionality described above, the SL content type also provides the following mechanisms:

- **Control of the level of user-intrusiveness**
  It is possible to control when the indicated service should be loaded if the client is busy with other activities.
- **Pre-emptive content caching**
  By setting a certain attribute of the SL, the indicated content is downloaded and cached instead of executed. This can be used to improve the end-user's experience of services that otherwise would have required content to be retrieved from an origin server.

5. The Service Loading Content Format

This section defines the content format used to represent the Service Loading (SL), which is an application of XML version 1.0 [XML]. The complete Service Loading DTD, which an implementation conforming to this specification MUST support, is defined in chapter 7.

5.1. Service Loading Character Set

The SL content type MUST use the same character set rules as specified in [WML], with the rules for meta-information placed within the content, since such information is not supported in SL.

5.2. The SL Element

```
<!ELEMENT sl EMPTY>
<!ATTLIST sl
  href        %URI;                              #REQUIRED
  action     (execute-low|execute-high|cache)        "execute-low"
>
```

Attributes

`href=%URI`

This attribute specifies the URI that is used to access the service.

`action=(execute-low|execute-high|cache)`

This attribute may contain a text string specifying the action to be taken when the SL is received.

Attribute value	Description reference
`execute-low`	Section 6.1.
`execute -high`	Section 6.1.
`cache`	Section 6.1.

If the attribute is not specified, the value "`execute-low`" is used.

6. Semantics

6.1. Reception and Service Invocation

A client receiving an SL should start processing it as soon as possible. The processing involves determining the value of the `action` attribute and, based on that value, either loading and executing the service indicated by the `href` attribute or loading and caching it.

There are three possible values for the `action` attribute, each implying:

- `action`= "`execute-low`"

  The service identified by the URI provided by the SL's `href` attribute is loaded in the same way as the user agent otherwise performs method requests initiated by the end-user [WAE]. This implies that service content is fetched either from an origin server or from the client's cache, if available. Once the method request is successfully completed, the user agent loads the service into a clean user agent context and executes it.

  This MUST be carried out in an non-user-intrusive manner

- `action`= "`execute-high`"

  The service is loaded and executed in the same way as for "`execute-low`", but MAY result in a user-intrusive bahavior.

- `action`= "`cache`"

  The service is loaded in the same way as for "`execute-low`". However, instead of executing the service (as described above) it is placed in the cache of the client. If no cache exists, the SL MUST be silently discarded.

6.2. Reception of Multiple Service Loadings

A client receiving multiple SLs that for some reason are not processed as described in section 6.1 upon reception (e.g. if the client is busy with other activities) MUST treat them as described below when they are about to be processed:

1. Remove duplicate SLs based on the `href` attribute value so that only one SL indicating a certain service is kept. If duplicates exist and they have different `action` attribute values, the following precedence order MUST be considered when deciding upon which one to keep:
   1. `execute-high`
   2. `execute-low`
   3. `cache`
2. Process the remaining SLs as described in section 6.1, sorted by:
   1. The `action` attribute value, starting with "`execute-high`", then "`execute-low`", and finally "`cache`".
   2. The order in which they were received.

A client MUST be able to maintain an implementation dependent number of SLs that are not processed upon reception. The number MUST be greater than or equal to one, but a value below three is NOT RECOMMENDED. A RECOMMENDED minimum storage space for each of these SIs is 500 octets.

7. Security Considerations

This section is informative.

A user agent which supports the SL is subject to certain attacks. The implementation should provide a means to protect the user agent against those security risks. An SL or the resource referred by SL may be discarded if a chosen security policy is not satisfied. The security policy is implementation specific.

Some possible security measures are:

- The user agent providing a means to disable acceptance of SL content type
- The user agent discarding any SL which is not authenticated or authorized
- The user agent discarding any resource referred by SL which is not allowed by the security policy
- A PPG providing a means to control which Push Initiators are allowed to push SLs

8. SL Reference Information

Service Loading (SL) is an application of [XML] version 1.0.

8.1. Document Identifiers

8.1.1. SGML Public Identifier

```
-//WAPFORUM//DTD SL 1.0//EN
```

8.1.2. SL Media Type

Textual form:

```
text/vnd.wap.sl
```

Tokenised form:

```
application/vnd.wap.slc
```

8.2. Document Type Definition (DTD)

```
<!--
Service Loading (SL) Document Type Definition.
SL is an XML language.  Typical usage:
   <?xml version="1.0"?>
   <!DOCTYPE sl PUBLIC "-//WAPFORUM//DTD SL 1.0//EN"
          "http://www.wapforum.org/DTD/sl.dtd">
   <sl>
   ...
   </sl>
-->

<!ENTITY % URI      "CDATA">         <!-- URI designating a
                                          hypertext node    -->

<!--==================== The SL Element ====================-->
<!ELEMENT sl EMPTY>
<!ATTLIST sl
  href        %URI;                              #REQUIRED
  action     (execute-low|execute-high|cache)        "execute-low"
>
```

9. A Compact Binary Representation of Service Loading

The SL 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

SL 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

SL 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 an SL and, if done, it MUST be based on the DOCTYPE declared in the SL. When validating the source text, the tokenisation process MUST accept any DOCTYPE or public identifier, if the document is identified as an SL media type (see section 8.1.2).

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

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.

Tag Name	***Token***
sl	5

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.

Attribute Name	***Attribute Value Prefix***	***Token***
action	execute-low	5
action	execute-high	6
action	cache	7
href		8
href	http://	9
href	http://www.	A
href	https://	B
href	https://www.	C

9.3.3. Attribute Value Tokens

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

Attribute Value	***Token***
.com/	85
.edu/	86
.net/	87
.org/	88

10. Example

This section is informative.

The example below illustrates how the SL used in chapter 4 can be designed and tokenised.

```
<?xml version="1.0"?>
     <!DOCTYPE sl PUBLIC "-//WAPFORUM//DTD SL 1.0//EN"
                    "http://www.wapforum.org/DTD/sl.dtd">
<sl href="http://www.xyz.com/ppaid/123/abc.wml"></sl>
```

The tokenised form of the example above (numbers in hexadecimal), using the WBXML encoding defined in chapter 9, is found below. This example assumes an UTF-8 character encoding and NULL terminated strings.

In this example the textual SL consists of 159 octets, while the encoded form consists of 32 octets.

```
02  06  6A  00  85  0A  03  'x' 'y' 'z' 00  85  03  'p' 'p' 'a'
'i' 'd' '/' '1' '2' '3' '/' 'a' 'b' 'c' '.' 'w' 'm' 'l' 00  01
```

In an expanded and annotated form:

Token Stream	***Description***
02	Version number - WBXML version 1.2
06	SL 1.0 Public Identifier
6A	Charset=UTF-8 (MIBEnum 106)
00	String table length
85	`sl`, with attributes
0A	Token for "`href="http://www.`"
03	Inline string follows
‘x’, ‘y’, ‘z’, 00	String
85	Token for "`.com/`"
03	Inline string follows
‘p’, ‘p’, ‘a’, ‘i’, ‘d’, ‘/’, ‘1’, ‘2’, ‘3’, ‘/’, ‘a’, ‘b’, ‘c’, ‘.’, ‘w’, ‘m’, ‘l’, 00	String
01	`END` (of `sl` attribute list)

Appendix A. Static Conformance Requirements (Normative)

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

A.1 Client Features

A.1.1 Predicates

Item	Function	Reference	Status	Requirement
SL-PRE-C-001	The client has a cache.	-	O	SL-SEM-C-002

A.1.2 Character Set and Encoding

Item	Function	Reference	Status	Requirement
SL-CSE-C-001	UTF-8 Encoding.	5.1	O	WML-01
SL-CSE-C-002	UTF-16 Encoding.	5.1	O	WML-02
SL-CSE-C-003	UCS-4 Encoding.	5.1	O	WML-03
SL-CSE-C-004	Other character encoding.	5.1	O	WML-04
SL-CSE-C-005	Reference processing (no meta-information).	5.1	M	
SL-CSE-C-006	Character entities.	5.1	M	
SL-CSE-C-007	Characters must be encoded	5.1	M	SL-CSE-C-001 OR SL-CSE-C-002 OR SL-CSE-C-003 OR SL-CSE-C-004

A.1.2 Content Format and Tokenisation

Item	Function	Reference	Status	Requirement
SL-CF-C-001	Support for the SL DTD.	8	M	
SL-CF-C-002	Support for SL in textual form (`text/vnd.wap.sl`).	8	O	
SL-CF-C-003	Support for SL in tokenised form (`application/vnd.wap.slc`).	9	M	

A.1.3 Semantics

Item	Function	Reference	Status	Requirement
SL-SEM-C-001	Processing of SLs with `action="execute-low"` and `action="execute-high"` attribute values.	6.1	M	
SL-SEM-C-002	Processing of SLs with `action="cache"` attribute value.	6.1	O	
SL-SEM-C-003	Handling of at least one SL that is not processed upon reception.	6.2	M	

A.2 Push Proxy Gateway Features

A.2.1 General

Item	Function	Reference	Status	Requirement
SL-PPG-S-001	Support for SL in textual form (`text/vnd.wap.sl`).	8	M	
SL-PPG-S-002	Support for encoding an SL into tokenised form (`application/vnd.wap.slc`).	9	M	
SL-PPG-S-003	Support for the SL token table.	9.3	M	

A.2.2 Validation

Item	Function	Reference	Status	Requirement
SL-VAL-S-001	XML well-formed.	9.2.1	M	
SL-VAL-S-002	XML validation.	9.2.1	O	

Appendix B. Change History (Informative)

Type of Change	Date	Section	Description
WAP-168-ServiceLoad-19991108-a	30-Jun-2000	All	The initial version of this document.
Class 3	31-Jul-2001	All	Applied current document template
WAP-168_100-ServiceLoad-20010423-a	31-Jul-2001	Appendix A	Reformat Static Conformance Requirements as specified
WAP-168_101-ServiceLoad-20010206-a	31-Jul-2001	5.2, 8.2, 10	Replaced quotation marks: “smart quotes” replaced with "basic quotes" as specified in changes 1 and 2. Corrected example as specified in change 3
WAP-168_102-ServiceLoad-20010614-a	31-Jul-2001	Appendix A	Reformat Static Conformance Requirements as specified (obsoletes WAP-168_100-ServiceLoad-20010423-a)