WISPr 2.0

08 April, 2010

by

Wireless Broadband Alliance

Doc Ref. No.: WBA/RM/WISPr
Version 01.00

TABLE OF CONTENTS

DISCLAIMER

The WBA (and all other organisations who may have contributed to this document) is providing this information on an 'as is' basis and makes no representations or warranties of any kind with respect to this information and disclaims all such representations and warranties. In addition, the WBA (and all other organisations who may have contributed to this document) makes no representations or warranties about the accuracy, completeness, or suitability for any purpose of the information. The information may contain technical inaccuracies or typographical errors. All liabilities of the WBA (and all other organisations who may have contributed to this document) howsoever arising for any such inaccuracies or errors are expressly excluded to the fullest extent permitted by law. None of the contributors make any representation or offer to license any of their intellectual property rights to the other, or to any third party. Nothing in this information or communication shall be relied on by any recipient. The WBA reserves the right to modify or amend this document without notice and in its sole discretion.

The WBA also disclaims any responsibility for identifying the existence of or for evaluating the applicability of any claimed copyrights, patents, patent applications, or other intellectual property rights, and will take no position on the validity or scope of any such rights. The WBA takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any effort to identify any such rights.

Neither the WBA nor any of the other organisations who may have contributed to this document will be liable for loss or damage arising out of or in connection with the use of this information. This is a comprehensive limitation of liability that applies to all damages of any kind, including (without limitation) compensatory, direct, indirect or consequential damages, loss of data, income or profit, loss of or damage to property and claims of third parties.

Notwithstanding the foregoing, none of the exclusions and limitations in the clause are intended to limit any rights you may have as a consumer under local law or other statutory rights which may not be excluded nor in any way to exclude or limit the WBA's or each contributing organisation's liability to you for death or personal injury resulting from its negligence or that of its members.

DOCUMENT CHANGE HISTORY

Version	Revision Date	Revised By	Description of Change
0.1	24 Aug. 2009	L. Freléchoux	Draft
0.2	08 Sept. 2009	L. Freléchoux	Integrated comments from WBA face-to-face meeting
0.3	13 October 2009	L. Freléchoux	Integrated comments from review of 0.2
0.4	28 October 2009	L. Freléchoux	New FSM: deprecation of AuthenticationPollReply, addition of PollNotification. Addition of informative FSM diagram. Addition of text + rules for EAP message sizes
0.5	5 November 2009	L. Freléchoux	Added new elements to Proxy message for multi VNP support
0.6	10 November 2009	L. Freléchoux	Added new FSM diagram, added the XSD part. Diverse corrections.
0.7	16 January 2010	L. Freléchoux	Added example of traces (Annex A). Rework of the AbortLogin. Update of FSM diagram. Diverse corrections.
0.8	28 January 2010	L. Freléchoux	XML encoding requirements. Clarification one EAP state machine. Update of FSM diagram. Diverse corrections.
0.9	08 February 2010	L. Freléchoux	Additional guidelines on handling of WISPr request retransmission. Method to handle concurrent login attempts. Clarification on search of WISPr response in HTTP response.
0.10	16 March 2010	L. Freléchoux	Added corrections.
01.00	08 April 2010	B Yerrapalli	Edits to release the v01.00 of WISPr 2.0

CONTRIBUTIONS

The following individuals from the WBA member companies have contributed to this specification:

Chair and Editor:

Laurent Freléchoux (Comfone AG)

Contributing members:

Andrew Gooday (IPASS Inc.)

Scot Hester (Boingo Inc.)

Marcelo Toledo (Vex Corp)

Blair Bullock, (IPASS Inc.)

Acknowledgements:

Chair would like to acknowledge the contributions made by the following:

- Andrew Gooday and Blair Bullock (IPass Inc.) for bringing in their expertise in WISPr 1.0 and GIS (proprietary specification of IPass) to the WISPr 2.0 specification
- Scot Hester (Boingo Inc.) for his comments and inputs to the specification
- Marcelo Toledo (Vex Corp) for his initial inputs to determine the content of the specification

1 Introduction

A public document named “Wireless ISP roaming (WISPr) 1.0” was released in February 2003. This document contained a description in Appendix D of a “Smart Client to Access Gateway Protocol”. This Universal Access Method (UAM) protocol has become widely used to access public IEEE 802.11 Wi-Fi networks with client software installed on the User Device.

The Wireless Broadband Alliance (WBA) has initiated a project for further enhancement and correction of the inter-working problems experienced with WISPr 1.0.

This document contains release 2.0 of the Annex D “Smart Client to Access Gateway Protocol” defined in WISPr 1.0. Other parts of the WISPr 1.0 document are not included in this specification and WISPr 2.0 scope is limited to the specification of client software to public WLAN network interface. WISP 2.0 is designed for “non” IEEE 802.1x networks as it requires IP communication with the AGW prior to the authentication of the user. WISPr offers authentication services based on layer 3 networking. WISPr 2.0 is designed as a front end to authentication protocols such as Radius, Diameter and the WBA WRIX specification [WRIX]. Figure 1 illustrates the overall architecture in a roaming situation and the relevant specifications.

This specification supersedes the “Client Web Login API” working document of the Wireless Broadband Alliance dated September 2006.

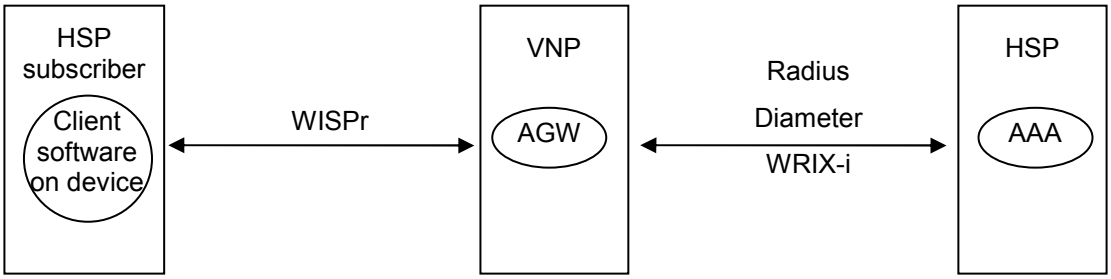


Figure 1: Relation between WISPr, AAA protocols and public WLAN roaming

2 Definitions

AAA	Authentication, Authorization and Accounting. Defines the server that stores the user account, including the login credentials.
AAA protocol	Means a protocol used for Authentication, Authorization and Accounting. RADIUS and DIAMETER from the IETF are AAA protocols.
Access Gateway	Access Gateway. The system in the public WLAN network that implements the WISPr server side of the protocol. The Access Gateway normally implements the WRIX-I or Radius client side of the protocol. The Access Gateway identifies the user/subscriber or User Device prior to accessing the WLAN service. The Access Gateway is also often called Network Access Server (NAS).

Client Software	Means an application that implements the client side of the WISPr. The Client Software can be a standalone application or integrated in the operating system.
EAP	Extensible Authentication Protocol
HSP	Home Service Provider.
URL	Uniform Resource Locator
User Device	Means a system that contains a WLAN interface and is used by a user to access the WLAN service.
VNP	Visited Network Provider
VPN	Virtual Private Network
WRIX	Wireless Roaming Intermediary eXchange

3 Compliance

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 RFC 2119.

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements in a NORMATIVE section of this document. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST level requirements but not all the SHOULD level requirements for its protocols is said to be "conditionally compliant."

4 Overview of the new features (Informative)

This release of WISPr introduces the following mandatory features at the access gateway:

- Recommended method for a client software to query the connection status on the Access Gateway
- Report of the maximum session time to the client software

This release of WISPr introduces the following optional features at the access gateway:

- Extension of the original scope of username/password authentication to support EAP authentication
- VNP discovery mechanism for WLAN infrastructures shared between several VNPs.

5 Backward compatibility and version negotiation (Informative)

WISPr 2.0 introduces a version negotiation during the initial WISPr redirect phase. The version negotiation indicates if both the client software and the access gateway support WISPr 2.0. If one of the two does not indicate its support for WISPr 2.0, both the client software and access gateway shall follow the rules of

WISPr 1.0. If both the client software and the access gateway support WISPr 2.0, they SHALL use the protocol specified in this release. Section 7.9 describes the details of the WISPr version negotiation.

WISPr 2.0 protocol is designed to re-use, whenever possible, the messages and codes defined in WISPr 1.0. WISPr 2.0 defines however new parameters for the feature extensions.

6 Overview of the WISPr protocol (Informative)

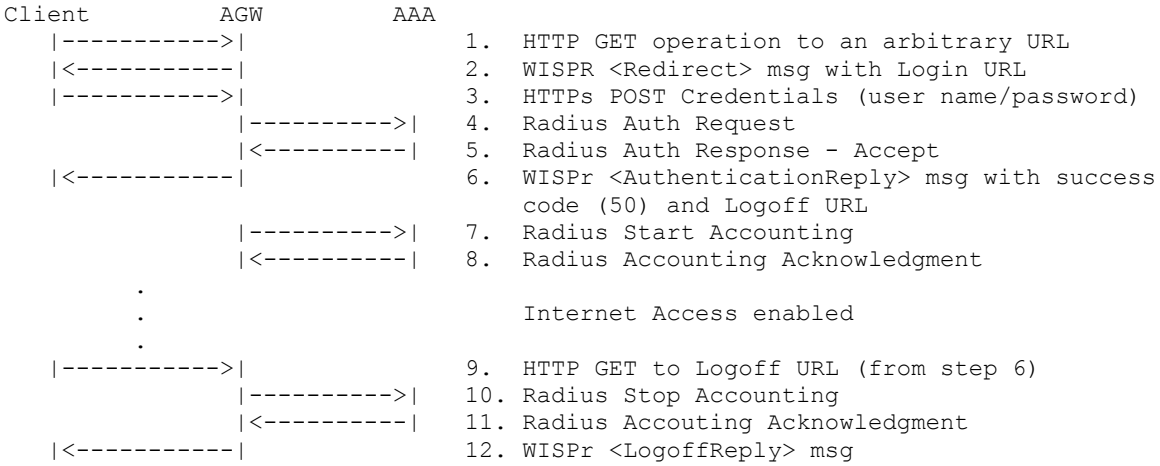
The WISPr protocol is implemented through the use of a client software initiated mix of unsecure and secure HTTP transactions.

WISPr transactions are always initiated by the client software. The client software passes parameters to the access gateway via the parameters of an HTTP Request. The access gateway responds to those requests by passing XML parameters in the HTTP response.

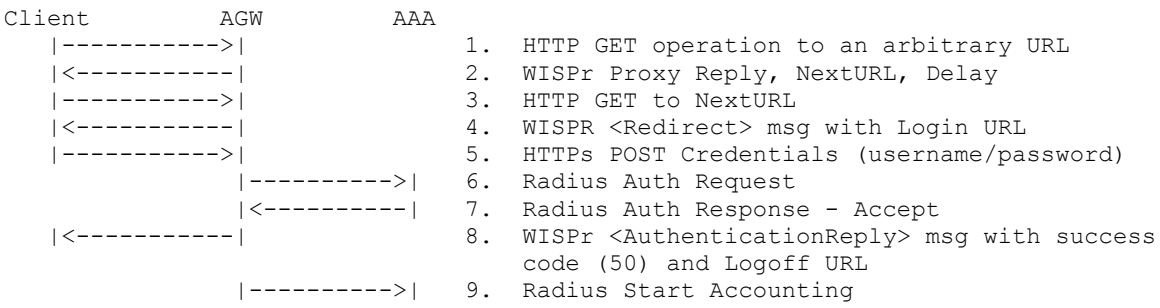
The following sections illustrate typical flows of the protocol for successful and rejected authentications.

6.1 Username/password authentication

Username/Password Login Request: Successful Case



Username/Password Login Request: Successful Case With Proxy Reply



```
                |<----------|   10. Radius Accounting Acknowledgment
          .
          .                        Internet Access Enabled
          .
     |----------->|              11. HTTP GET to Logoff URL
                |---------->|   12. Radius Stop Accounting
                |<----------|   13. Radius Accounting Acknowledgment
     |<-----------|              14. WISPr <LogoffReply> msg
```

Username/Password Login Request: Successful Case With Polling

```
Client          AGW          AAA
   |----------->|               1.  HTTP GET operation to an arbitrary URL
   |<-----------|               2.  WISPR <Redirect> msg with Login URL
   |----------->|               3.  HTTPs POST Credentials (username/password)
                |---------->|   4.  Radius Auth Request
   |<-----------|               5.  WISPr <PollNotication> msg with Polling URL
                                  and delay
   |----------->|               6.  HTTP GET to Polling URL
   |<-----------|               7.  WISPr <PollNotication> msg with Polling URL
                                  and delay
                |<----------|   8.  Radius Auth Reply - Accept
   |----------->|               9.  HTTP GET to Polling URL
   |<-----------|               10. WISPr <AuthenticationReply> msg with success
                                  code (50) and Logoff URL
                |---------->|   11. Radius Start Accounting
                |<----------|   12. Radius Accounting Acknowledgment
          .
          .                        Internet Access Enabled
          .
   |----------->|               13. HTTP GET Logoff URL
                |---------->|   14. Radius Stop Accounting
                |<----------|   15. Radius Accounting Acknowledgment
   |<-----------|               16. WISPr <LogoffReply> msg
```

Username/Password Login Request: Reject

```
Client          AGW          AAA
   |----------->|               1.  HTTP GET operation to an arbitrary URL
   |<-----------|               2.  WISPR <Redirect> msg with Login URL
   |----------->|               3.  HTTPs POST Credentials (username/password)
                |---------->|   4.  Radius Auth Request
                |<----------|   5.  Radius Auth Reply - Reject
   |<-----------|               6.  WISPr <AuthenticationReply> msg with reject
                                  code (100)
```

Username/Password Login Request: Reject With Polling

```
Client          AGW          AAA
   |----------->|               1.  HTTP GET operation to an arbitrary URL
```

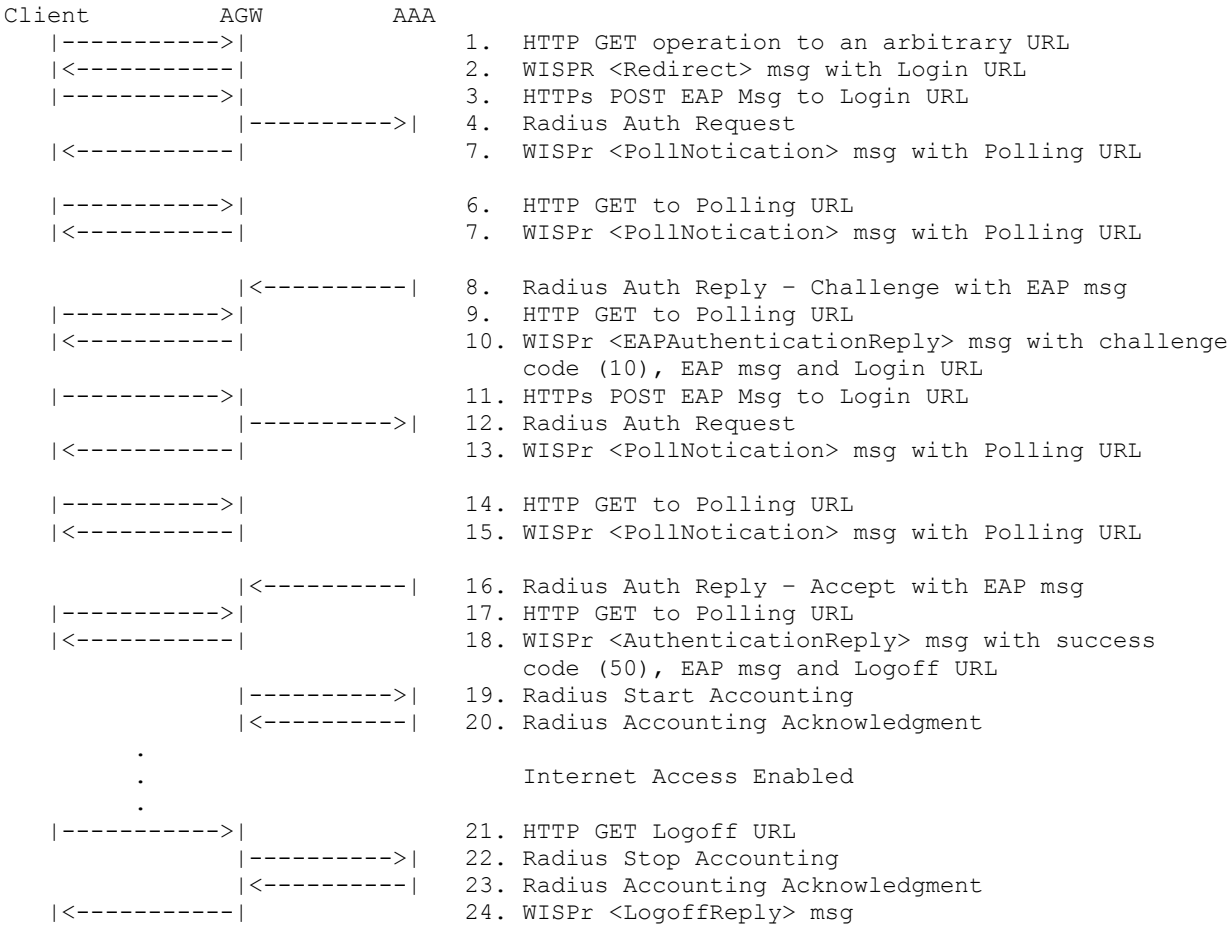
```
    |<-----------|               2.  WISPR <Redirect> msg with Login URL
    |----------->|               3.  HTTPs POST Credentials (username/password)
                 |---------->|   4.  Radius Auth Request
    |<-----------|               5.  WISPr <PollNotication> msg with Polling URL

    |----------->|               6.  HTTP GET to polling URL
    |<-----------|               5.  WISPr <PollNotication> msg with Polling URL

                 |<----------|   8.  Radius Auth Reply - Reject
    |----------->|               9.  HTTP GET to Polling URL
    |<-----------|               10. WISPr <AuthenticationReply> msg with reject
                                     code (100)
```

6.2 EAP authentication

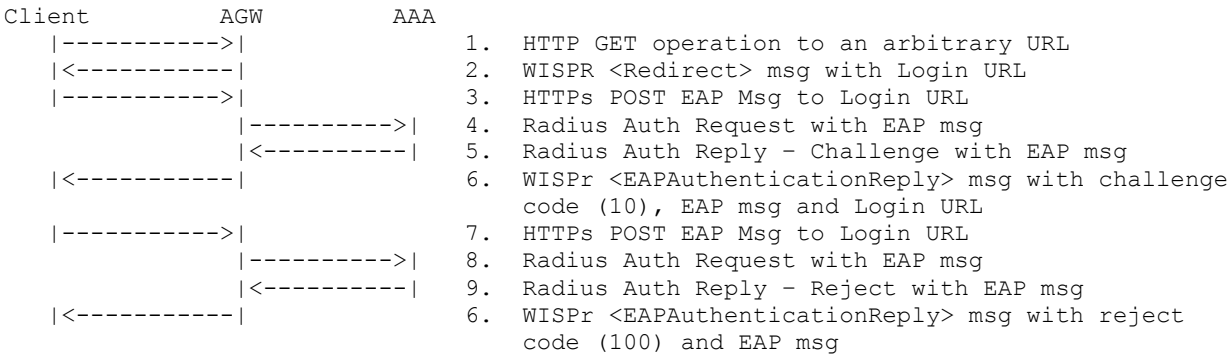
EAP Login Request: Successful Case

```
  Client         AGW         AAA
    |----------->|               1.  HTTP GET operation to an arbitrary URL
    |<-----------|               2.  WISPR <Redirect> msg with Login URL
    |----------->|               3.  HTTPs POST EAP Msg to Login URL
                 |---------->|   4.  Radius Auth Request with EAP msg
                 |<----------|   5.  Radius Auth Reply - Challenge with EAP msg
    |<-----------|               6.  WISPr <EAPAuthenticationReply> msg with challenge
                                     code (10), EAP msg and Login URL
    |----------->|               7.  HTTPs POST EAP Msg to Login URL
                 |---------->|   8.  Radius Auth Request with EAP msg
                 |<----------|   9.  Radius Auth Reply - Accept with EAP msg
    |<-----------|               6.  WISPr <EAPAuthenticationReply> msg with success
                                     code (50), EAP msg and Logoff URL
                 |---------->|   11. Radius Start Accounting
                 |<----------|   12. Radius Accounting Acknowledgment
          .
          .                          Internet Access Enabled
          .
    |----------->|               13. HTTP GET Logoff URL
                 |---------->|   14. Radius Stop Accounting
                 |<----------|   15. Radius Accounting Acknowledgment
    |<-----------|               16. WISPr <LogoffReply> msg
```

EAP Login Request: Successful Case With Polling



EAP Login Request: Reject



For PAP login, an access gateway must implement at the minimum the following WISPr messages: <Redirect>, <AuthenticationReply>, <StatusReply>, <LogoffReply>.

For EAP login, an access gateway must implement at the minimum the following WISPr messages: <Redirect>, <EAPAuthenticationReply>, <StatusReply>, <LogoffReply>.

7 Protocol specification (normative)

7.1 WISPr transaction between the client software and the access gateway

The WISPr protocol works over HTTP and HTTPs. The client software SHALL support HTTP 1.1, or at the minimum HTTP 1.0. The access gateway SHALL support and use HTTP 1.1 (RFC 2616). The access gateway SHALL not use HTTP Cookies (RFC 2965) during WISPr transactions.

Communication of the user credentials SHALL be protected by SSL using HTTPs. The access gateway SHALL use an SSL certificate that is valid and signed by a Certificate Authority. The certificate revocation list (CRL) SHALL be accessible prior to authentication. It is recommended to use certificate issued by a primary Public Root authority for better certificate recognition across operating systems. The client software SHOULD require the permission of the user before submitting user credentials through an unsecure channel or not proceed with the login procedure. By unsecure channel it is meant an HTTP request that is not SSL protected or if the SSL certificate is not valid or not signed by a recognized Certificated Authority.

The WISPrVersion used in WISPr request and WISPr responses is a ASCII string of format “major.minor” where major and minor are positive integers. The version of this specification is 2.0. The version of WISPr 1.0 Annex-D is 1.0.

7.2 WISPr request parameters

A WISPr request from the client software is an HTTP/HTTPs GET or POST with mandatory and optional request parameters. The format and rules defined in this document for the request parameters apply to both URL query parameters and parameters set in the HTTP BODY for the HTTP(s) POST operations) of the WISPr request.

The following request parameters are defined in this release of WISPr:

Parameter Name	Encoding of value	Max size[1] (bytes)	Parameter Meaning	Used in
WISPrVersion	WISPrVersion	8	Indicates which version of the WISPr protocol the client software will use for the WISPr transactions	WISPr Login Request
UserName	URL encoding	253	Contains the NAI of the user	WISPr Login Request
Password	URL encoding	128	Contains the password of the user account	WISPr Login Request
WISPrEAPMsg	URL encoding	87384 (Note 2)	Contains a Base64 encoded EAP response message	WISPr Login Request

Note 1: The maximum size denotes the maximum size of the data before the URL encoding process. The encoded message can theoretically be up to 3 times larger.

Note 2: For maximum inter-operability, and to avoid UDP MTU transport problems on a Radius AAA interface between the VNP and the HSP, it is recommended that the client software does not add more than 1265 bytes total length of EAP message per WISPr Request. The access gateway will then generate a maximum of 5 EAP-Messages Radius attributes out of the received EAP message.

Table 1: Parameters of a WISPr Request

The client software SHALL encode the values of those request parameters indicated with “URL Encoding” according to the character escaping scheme defined in RFC 3986 (see “query component” section) prior to the HTTP transmission. The access gateway SHALL decode the values of the request parameters on reception of a WISPr request.

The WISPrEAPMsg parameter is only used if the client software implements EAP over WISPr. The binary EAP message is encoded using the base 64 encoding algorithm defined in RFC 3548. The Base 64 encoded string is then character escaped according to RFC 3986. The access gateway SHALL decode the value of the WISPrEAPMsg parameter on reception of a WISPr request. The access gateway SHALL support the receipt and sending of a binary EAP message of size 1265 bytes at the minimum.

Other parameters may be used by the access gateway as explained further on. The request parameters in Table 1 are used by the WISPr protocol and reserved to this use. Access gateways that require additional parameters SHALL not use any of those parameters defined in Table 1. To ensure future compatibility, private parameters SHALL not start with the prefix “WISPr”.

The names of request parameters in Table 1 are case sensitive. The client software SHALL use the same casing as in Table 1. To avoid inter-operability problems, the gateway SHALL not use proprietary parameters with names defined in Table 1 but with a different casing.

The parameters “Button”, “OriginatingServer” and “FNAME” indicated in WISPr 1.0 are deprecated and are not used anymore in WISPr 2.0.

The access gateway SHALL ignore any request parameter that it does not expect or understand. This applies to both URL query parameters and request parameters in the HTTP body. By ignoring it is meant that the request is processed according to the rules defined in this specification.

The access gateway SHALL return a WISPr response code of *Protocol Error* (254) if mandatory parameters are not submitted by the client software, or if the combination of parameters is not allowed by the protocol.

The client software SHALL set the Accept header of the HTTP request to accept at the minimum “text/html”. The client software MAY indicate that it accepts other formats. The access gateway SHALL serve HTTP requests with Accept header that specifies text/html.

The client software SHALL set the HTTP Accept-Encoding header to accept no encoding. The Access Gateway SHALL never return a WISPr response in a HTTP Response that contains an encoded (i.e. compressed) HTTP body.

The access gateway, in the event that it receives the same parameter name in both the URL query and the HTTP body, shall use the value in the HTTP body.

Please refer to section 7.5 for the requirements on the HTTP User-Agent header.

7.3 WISPr response parameters

7.3.1 WISPr XML data

A WISPr response from access gateway to the client software is an XML data that follows the normative scheme of Appendix A. The WISPr XML data SHALL be contained anywhere within the <HTML> … </HTML> tags of the HTML page returned to the client software. It is recommended that the WISPr XML block be located as early as possible in the transmission of the HTTP response so that the client software can start processing the WISPr response before having received the whole HTTP response.

The WISPr XML data SHALL follow [XML 1.0]. The WISPr XML start with an XML header that specifies the use of XML version 1.0 and the use of UTF-8 character encoding. The access gateway SHALL use the UTF-8 character encoding for the XML data, even if the HTML header field or the HTTP Content-Type header indicates that another character encoding applies for the HTTP content returned.

All XML values (i.e. data between XML tags) SHALL comply with the XML syntax. Restricted XML characters (&, >, <, “, ‘) SHALL be escaped or the value SHALL be embedded in a CDATA[[…]] container by the access gateway (see [XML 1.0]). The client software SHALL support both the CDATA[[…]] container and the un-escaping of XML special characters. Upon receipt of a value contained in a CDATA[[…]] container, the client software SHALL used the contained string as the value without any un-escaping operation. Upon receipt of a value that is not contained in CDATA[[…]] clause, the client software shall process the value through the XML special characters un-escaping to recover the original string.

The WISPr response message SHALL be contained between <WISPAccessGatewayParam> … </WISPAccessGatewayParam> tags.

XML Tag	Message Type	Message Type Meaning
<Redirect>	100	Initial redirect message
<Proxy>	110	Proxy notification
<AuthenticationReply>	120	PAP Authentication notification
<EAPAuthenticationReply>	121	EAP Authentication notification
<LogoffReply>	130	Logoff notification
~~<AuthenticationPollReply>~~	~~140~~	Deprecated (see note 1)
<AbortLoginReply>	150	Response to Abort Login
<StatusReply>	160	Response to Status query
<PollNotification>	170	Authentication Poll notification

Note 1: The <AuthenticationPollReply> message defined in WISPr 1.0 is deprecated and SHALL not be used in WISPr 2.0.

Table 2: Message Type of WISPr Response

The following WISPr response elements are defined in this release of WISPr:

Element Name	Element type	Description
<MessageType>	Positive Integer	Indicate the type of the message
<ResponseCode>	Positive Integer	Indicate the result of the requested execution
~~<AccessProcedure>~~	String	Deprecated. (see note 1)
<VersionLow>	WISPrVersion	Indicates the lowest version of the WISPr protocol supported by the access gateway
<VersionHigh>	WISPrVersion	Indicates the highest version of the WISPr protocol supported by the access gateway
<LocationName>	String	Indicates a textual description of the hotspot or group of hotspots to which the user is connected to
<AccessLocation>	String	Indicates a key that uniquely identify a hotspot or group of hotspots to which the user is connected to
<MaxSessionTime>	Positive Long	Indicates in seconds the time left before the access gateway terminates the session.
<EAPMsg>	Base64 Encoded Binary	Contains a Base 64 encoded EAP request message
<Delay>	Positive Integer	Indicates in seconds the time that the client software must wait before sending the next WISPr request.
<ReplyMessage>	String	Describes in a human readable format the cause of an error. Reply messages can be displayed to the user by the client software.

Element Name	Element type	Description
<AbortLoginURL>	URL	Indicates the URL that can be requested to abort a login request when the access gateway is using the polling mechanism.
<LoginURL>	URL	Indicates the URL that must be requested to open a session (authenticate the user)
<LogoffURL>	URL	Indicates the URL that must be requested to terminate the session
<StatusURL>	URL	Indicates the URL that can be requested to retrieve the status of the session
<RedirectionURL>	URL	Indicates the URL in the client software must open in a browser upon successful login
<LoginResultsURL>	URL	Indicates the URL that must be requested for polling to fetch the result of a pending authentication
<NextURL>	URL	Indicates the URL that must be requested following a proxy operation
<OperatorURL ID=xxx>	URL	Indicates the URL that must be requested to reach the infrastructure of an hosted VNP
<Status>	Integer	Indicates the status of a session. Value 1 indicates that the session is active. Value 0 indicates that the session is terminated.

Note 1: The <AccessProcedure> element defined in WISPr 1.0 is deprecated and SHALL only be used if the access gateway also provides WISPr 1.0 access service.

Table 3: Type of elements in WISPr Response Messages

The <EAPMsg> element is only used if the access gateways implements EAP over WISPr support. The binary EAP message is encoded using the base 64 encoding algorithm defined in RFC 3548.

A value of type “URL” SHALL strictly comply with RFC 3986. Implementations of access gateways must take care that the values and variable names of query parameters returned in the URL are character escaped as per RFC 3986 prior to any XML encoding (be character escaping or CDATA[[..]] container).

The <LoginUrl> shall contain a reference to a secured URL (https). Specification of the port is optional, the standard port value 443 shall be considered as the default. It is recommended that the access gateways use the standard port value to avoid unwanted firewall problems at the User Device.

The client software SHALL ignore any response element that it does not recognize and process the response according to the rules defined in this specification.

Response Code	Response Code Meaning
0	No error
10	Login challenged (Access CHALLENGE)
50	Login succeeded (Access ACCEPT)

Response Code	Response Code Meaning
100	Login failed (Access REJECT)
102	Authentication server error/timeout
105	Network Administrator Error: No authentication server enabled
150	Logoff succeeded
151	Login aborted
152	Invalid session
200	Proxy detection/repeat operation
~~201~~	*Deprecated (Note 1)*
252	Invalid state for WISPr request
253	MTU of AAA message is too big
254	Protocol error
255	Access gateway internal error

Note 1: The Response Code 201 defined in WISPr 1.0 is deprecated and SHALL not be used in WISPr 2.0. This response code is replaced by the <PollNotification> message.

Table 4: Response Codes of WISPr Response Messages

7.3.2 Use of response codes that indicate errors

An access gateway that receives a WISPr login request with a UserName that contains a realm or routing prefix to which the access gateway is not provisioned to route to, SHALL use the response code 105 to indicate the failure of the WISPr login request.

An access gateway that times out when waiting on the response to a AAA request, SHALL use the response code 102 to indicate the failure of the WISPr login request.

An access gateway that fails to transmit a AAA message because the size of the Radius message is exceeding the MTU of the transport path, SHALL use response code 253 to indicate to the client the cause of the error. Such error can occur with EAP method that requires the transmission of large data (e.g. EAP-TLS, EAP-TTLS).

An access gateway that receives a WISPr request that misses some expected parameters, be it mandatory WISPr parameters or private parameters, SHALL use the response code 254 to indicate the error of the WISPr request.

An access gateway that receives a WISPr request in a state where it does not expect such request, SHALL use the response code 252 to indicate to the client the cause of the error.

An access gateway that must indicate to the client software an error that does not fall under the description of the other error codes SHALL use response code 255.

7.3.3 HTTP transmission of a WISPr response

The access gateway SHALL always return an HTML page when it wants to communicate a WISPr response to the client software. The XML data SHALL be embedded in a HTML comment to prevent interpretation by web browsers.

The access gateway SHALL only return a WISPr response in an HTTP 200 or HTTP 302 response. A WISPr response to an HTTP POST SHALL only be returned in an HTTP 200 response. The client software SHALL search for WISPr responses in all HTTP 200 (OK) and HTTP 302 (Redirect) responses. It SHALL search for WISPr responses in HTTP 200 and HTTP 302 responses even if the HTTP response does not contain a Content-Length attribute.

An access gateway SHOULD not use the HTML <META HTTP-EQUIV="Refresh" Content="…"> but rather use respond with an HTTP 302 response code.

The client software shall search for HTML <META HTTP-EQUIV="Refresh" Content="…"> tags in an HTTP 200 response. The content parameter shall comply with the following format: Content=#seconds URL=#redirect_URL. Example: <META http-equiv="refresh" content="11; URL=http://www.yoursite.com/">. It should be noted the W3C strongly discourages the use of the <META HTTP-EQUIV="Refresh". The client software shall wait the indicated number of seconds prior to sending a new HTTP request to the redirect_URL.

This specification uses the term HTTP REDIRECT to indicate the receipt by the client software of either an HTTP 302 response or an HTTP 200 with an HTML <META HTTP-EQUIV="Refresh" Content="…"> tag.

The access gateway SHALL set the Content-Type header of the HTTP response to `Content-Type: text/html`. The access gateway MAY add a media type as per RFC 2616.

All WISPr responses from the access gateway to the client software SHALL contain one <MessageType> element and one <ResponseCode> element. Additional parameters may be returned based on the message type and the response code (see section 7.8 and followers).

The client software that receives a WISPr response (i.e. WISPr XML block) in a HTTP REDIRECT SHALL not follow the redirection indication but SHALL process the WISPr response and act based on the indications in the WISPr response message.

7.3.4 Example of a WISPr response

Example of a WISPr XML Response embedded in an HTML comment:

```
<HTML> <!--
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
    <Redirect>
       <MessageType>100</MessageType>
       <ResponseCode>0</ResponseCode>
       <AccessProcedure>1.0</AccessProcedure>
       <VersionLow>1.0</VersionLow>
       <VersionHigh>2.0</VersionHigh>
       <AccessLocation>TMOBILE:US, 10403 SFO San Francisco Intl
Airport</AccessLocation>
       <LocationName> SFO San Francisco Intl Airport </LocationName>
       <LoginURL>https://www.acmewisp.com/login/?sid=AA34Bbf</LoginURL>
       <EAPMsg>AQAABQE=</EAPMsg>
    </Redirect>
</WISPAccessGatewayParam>
--> </HTML>
```

7.4 Client software retry policy

The client software MAY resend a WISPr request if it did not receive a WISPr response within an implementation specific time out. The access gateway SHALL provide a mechanism to detect that a WISPr request is a retry from client software.

7.5 URLs returned in WISPr XML elements

The access gateway has to indicate URLs to the client software as part of the WISPr transaction process. The following rules apply to all WISPr elements of type URL indicated in Table 2:

- The URL SHALL conform with RFC 3986 and RFC 2616 and SHALL contain the protocol identifier (“http:” or “https:”)
- The URL may contain access gateway implementation specific query parameters. The value and variable names of those query parameters SHALL be character escaped as per section 3.4 of RFC 3986 (URL encoding)
- The client software SHALL support URLs of a length of 1000 characters at the minimum
- The access gateway SHALL never return a URL type element with no URL in the element. The URL SHALL be RFC 3986 compliant.

The client software SHALL use the URLs as follows:

- To the exception of the <LoginURL> where the client software has to use the HTTP POST method, all other URLs SHALL be requested with the HTTP GET method
- All query parameters of the received URL element SHALL be submitted as received, without undergoing any URL encoding (values are already URL encoded)
- In the case of an HTTP POST, all query parameters part of the <LoginURL> SHALL remain in the URL. Parameters of Table 1 SHALL be exclusively present in the body of the HTTP POST request.

7.6 HTTP User-Agent values and filtering

It is recommended that the access gateway does not use the HTTP User-Agent of HTTP requests to decide whether or not to trigger the WISPr protocol (i.e. to return a WISPr Response with <Redirect> or <Proxy> message).

The practice however indicates that action based on the HTTP User-Agent is used when the WLAN infrastructure is shared between different hotspots network providers (e.g. in airports). The provider of the WLAN infrastructure uses this information to direct HTTP requests to one hotspot network provider or the other. The use of HTTP User-Agent as a filtering mechanism alters the discovery and dynamics of the WISPr protocol in the sense that each HSP must have the User-Agent value of its software registered by infrastructure owner in order to be served with the WISPr parameters of the appropriate VNP. The provisioning issues become even larger when 3rd party roaming providers, such as aggregators, are involved.

This release introduces a structure to the HTTP User-Agent field so that provisioning of the filtering rules is simplified.

The client software shall use the following HTTP User Agent structure value for all WISPr requests:

```
User Agent: WISPR!client_software_identifier
```

The character '!' is a special character and shall not be present in any of the values in the User Agent structure for WISPr to allow future extension.

This structure simplifies the HTTP User Agent filtering rules for providers of WLAN infrastructure that require HTTP User Agent filtering. Instead of having to provision each individual User Agent values, any HTTP request that starts with WISPR! denotes a WISPr request (or trigger of the WISPr protocol).

7.7 Trigger of the WISPr protocol: HTTP GET on an arbitrary URL

The client software shall perform an HTTP GET to a valid web site to initiate the WISPr protocol. The URL for the initial HTTP GET is referred thereafter to as the "Arbitrary URL".

In situations where the User Device is already authorized or no authorization is needed, the access gateway shall route the request following the standard IP routing. The targeted web site will then return the requested content.

When the User Device is not already authorized for access, the access gateway shall return one of the following in reply to the initial HTTP GET operation:

- An HTTP REDIRECT and no WISPr Response.
- A WISPr Response with a WISPr <Redirect> message (see section 7.9)
- A WISPr Response with a WISPr <Proxy> message (see section 7.8)

If the client software receives an HTTP Redirect that does not contain a WISPr response, it shall follow the redirect indication according the HTTP/HTML specifications.

The access gateway SHALL make no other assumption on the Arbitrary URL than that the URL complies with RFC 3986 and RFC 2616. The access gateway must therefore consider that the Arbitrary URL may contain a hostname or IP address, a port, and query parameters. The client software SHALL however fulfil the requirements described in section 7.2 for the HTTP request. The Arbitrary URL shall therefore be chosen to fit the requirement of section 7.2.

The client software SHALL use unsecure HTTP to perform the initial HTTP GET. The client software SHALL NOT use any other HTTP Method than GET to trigger the WISPr protocol.

7.8 WISPr Proxy

The access gateway SHOULD not use a WISPr Proxy message as a plain HTTP Redirect 302.It SHOULD use an HTTP Redirect 302 instead. The Proxy message can be used in situation where a WLAN infrastructure is shared between one or more VNPs and the authentication service is provided separately by each VNP. The Proxy message lists the VNPs served by a given WLAN infrastructure and provides a VNP specific URL that the client software can use to trigger the WISPr process on a the chosen VNP.

The access gateway, as the result of the trigger on an arbitrary URL, MAY return a WISPr response with a <Proxy> message to indicate to the client software where to fetch the <Redirect> message.

The **<Proxy> message** contains the following elements:

Information name	Field format/value	Required/ Optional
Message Type	<MessageType> 110 </MessageType>	Required
Response Code	<ResponseCode> {Response Code} </ResponseCode>	Required
Next URL	<NextURL> http[s]://{<site specific URL>} </NextURL>	Optional
Delay in seconds	<Delay> {Number of seconds data} </Delay>	Optional
VNP specific next URL	<OperatorURL ID=”xxx”> http://{<VNP specific URL>} </OperatorURL>	Required

Table 5: Proxy Message Elements

If the <Proxy> message includes a <Delay> element, the client software SHALL suspend execution for the number of seconds specified in the element before it sends the next WISPr request.

The access gateway SHOULD specify a <Delay> interval large enough to reflect actual processing delay to prevent large numbers of WISPr requests. Gateways SHOULD specify a non-zero <Delay> value whenever there is an actual processing delay on the access gateway.

If the <Proxy> message includes a <NextURL> element, the client software SHALL use the URL value contained in the <NextURL> element for the next WISPr request. This implements a URL “redirect” at the WISPr application layer. The <Proxy> message can be used to redirect the client software to a different gateway page set without changing the normal, browser-oriented gateway page sequence.

The next URL MAY be different in each <Proxy> message. If no <NextURL> element is present in the <Proxy> message, the client software SHALL use the last-used URL value for the next WISPr request.

The <Proxy> message MAY contain one or several OperatorURL element. Each OperatorURL element SHALL contain one, and only one ID variable that equals a string, delimited with double quote characters (“…“). This string identifies the VNP being hosted on the WLAN infrastructure. The variable name SHALL be in capital characters (i.e.”ID” not “id” or “Id”). For WBA members, the format SHALL follow the WBA member identification format (e.g. ID=”ORANGE:FR”).

Each OperatorURL element listed in the <Proxy> message shall contain a valid URL that can be used by the client software to trigger the WISPr process on the corresponding VNP.

A client software MAY, when it receives a <Proxy> message, search the OperatorURL elements to identify a VNP that provides roaming services to the subscriber. If a suitable VNP is found, the client software shall then make a WISPr request on the URL indicated in the selected OperatorURL element.

The <Proxy> message may be returned by the access gateway multiple times (but only once in response to each WISPr request). Some reasonable limit to the number of recurrences SHOULD be implemented by the client software to avoid an undesirable user experience. A 6-cycle limit is recommended.

The access gateway SHALL indicate one of the following response codes in the <Proxy> message:

Response Code	Response Message
200	Proxy detection/repeat operation
254	Protocol error
255	Access gateway internal error

Table 6: Proxy Response Codes

7.9 WISPr Redirect

The access gateway, as the result of the trigger on an arbitrary URL or a request following a <Proxy> message SHALL return a WISPr response with <Redirect> message to indicate to the client software where to submit the user credentials.

The **<Redirect> message** contains the following elements:

Field format/value	Required/ Optional
<MessageType> 100 </MessageType>	Required
<ResponseCode> {Response Code} </ResponseCode>	Required
<VersionLow> {Protocol version} </VersionLow>	Required
<VersionHigh> {Protocol version} </VersionHigh>	Required
<AccessLocation> {Location ID} </AccessLocation>	Required
<LocationName> {User readable location name} </LocationName>	Required
<LoginURL> https://{login URL} </LoginURL>	Required

Field format/value	Required/ Optional
<AbortLoginURL> http[s]://{abort login URL} </AbortLoginURL>	Optional
<EAPMsg> Encoded Identity Request EAP Message </EAPMsg>	Optional
<AccessProcedure> </AccessProcedure>	Deprecated (note 1)

Note 1: The <AccessProcedure> element defined in WISPr 1.0 is deprecated in the <Redirect> message and SHALL only be used if the access gateway also provides WISPr 1.0 access service. WISPr 2.0 client software SHALL ignore this parameter on receipt.

Table 7: Redirect Message Elements

The <AccessProcedure> element is deprecated and SHALL only be present, with value “1.0” if the access gateway supports also WISPr 1.0. This ensures inter-operability with client software that does not support WISPr 2.0

The access gateway SHALL set the <VersionLow> to the lowest version of the protocol supported by the access gateway. Currently defined values are “1.0” and “2.0”.

The access gateway SHALL set the <VersionHigh> to the highest version of the protocol supported by the access gateway. It MAY be set to “2.0” if and only if the access gateway complies with this specification.

The valid combinations of VersionLow and VersionHigh are as follows:

Version High = 1	Version Low = 1
Version High = 2	Version Low = 1
Version High = 2	Version Low = 2

The <AccessLocation> element SHALL contain a String that matches the location information sent in the AAA requests. For WRIX, it SHALL match the value and format of the “Location Name” VSA (Vendor ID: 14122) defined in WRIX-i (see Appendix C).

The <LocationName> also identifies the hotspot or group of hotspots to which the User Device is associated. It SHALL contain a human friendly string that can be displayed by the client software to indicate the point of connection to the user. It is recommended that the value of the <LocationName> element be set to value of the “English_Location_Name” field in WRIX-L.

Note: Due to historical reasons, and to ensure backward compatibility, the “LocationName” VSA MUST be contained in the <AccessLocation> element and NOT the <LocationName> element.

The access gateway MAY add an <AbortLoginURL> element if it implements an abort login procedure. Implementation of the abort login process is optional. Section 7.13 describes the abort login process.

The <EAPMsg> element MAY only be present if the access gateway implements EAP over WISPr. Its presence indicates to the client software that access gateway implements EAP over WISPr and accepts EAP

authentication requests. The <EAPMsg> element SHALL contain an encoded EAP Identity Request. See section 7.3.1 for the encoding rules.

The access gateway SHALL indicate one of the following response codes in the <Redirect> message:

Response Code	Response Message
0	No error
105	Network Administrator Error: No authentication server enabled
254	Protocol error
255	Access gateway internal error

Table 8: Redirect Response Codes

7.10 WISPr Login

The client software receives in the WISPr <Redirect> message the login URL that it MAY use to request access to the service (i.e. authentication).

The client software SHALL perform a WISPr request over an SSL protected HTTP POST transaction to the <LoginURL> element received in the <Redirect> message. Please refer to section 7.3.1 for the default port value.

The body of the HTTP POST SHALL contain the list of login parameters (see below). The character '&' SHALL be used delimit pairs of parameter/value. The character '=' SHALL be used to delimit the parameter from the value.

Example: param1=value1&param2=value2&param3=value3

For a username/password (PAP) authentication the client software SHALL include the following elements in the WISPr request:

Element name	Required/ Optional
WISPrVersion	Required
UserName	Required
Password	Required

Table 9: Username/Password Login Elements

For an EAP authentication the client software SHALL include the following elements in the WISPr request:

Element name	Required/ Optional
WISPrVersion	Required

Element name	Required/ Optional
UserName	Required
WISPrEAPMsg	Required

Table 10: EAP Login Elements

All request parameters SHALL conform to section 7.2.

The client software SHALL set the WISPrVersion parameter to the version that it wishes to use for login/logoff procedure. It SHALL be set to “2.0” if the client follows the rules of this specification[1].

The WISPr request SHALL NOT contain both a <Password> and a <WISPrEAPMsg> parameter. The access gateway SHALL return a WISPr Response with an <AuthenticationReply> message that indicates a *Protocol Error (254)* response code if the WISPr request contains both parameters.

A client software MAY only request an EAP authentication if it received an EAP identity request in the <EAPMsg> element of the <Redirect> message. The EAP response message and subsequent EAP handshake shall comply with RFC 3748.

The access gateway SHALL verify if the WISPrVersion parameter is present. If it is not present, and provided that it supports WISPr 1.0, it SHALL proceed according to the WISPr 1.0 document.

If the WISPrVersion parameter is present and set to “1.0” it SHALL proceed according to the WISPr 1.0 document.

If the WISPrVersion parameter is present and set to “2.0” it SHALL proceed according to this specification.

If the WISPrVersion parameter is present but the value does not match any of the above, the access gateway MAY either:

- return a WISPr response with an <AuthenticationReply> message that indicates a *Protocol error (254)* response code, or
- proceed with the highest version supported by the infrastructure.

If the access gateway requires the client software to poll for the result of the authentication, it SHALL return a WISPr response with a <PollNotification> message in response to the WISPr request. Please refer to section 7.11 for the <PollNotification> message format and procedures.

When the access gateway has received the AAA response, or when an error occurs, it SHALL return a WISPr response with a < AuthenticationReply > or < EAPAuthenticationReply > message depending on the authentication mode used in the request.

If the access gateway receives wrongly a WISPr login request for EAP when the access gateway did not indicate in the <Redirect> message that it supports EAP, it shall return return a WISPr response with an <AuthenticationReply> message that indicates a *Protocol error (254)* response code.

[1] The value "2.0" complies with the URL encoding scheme of RFC 3986. It can therefore be processed by the access gateway as a URL encoded value.

If the access gateway receives a WISPr login request from a User Device, while a session is already active, it SHALL return a WISPr response with an < AuthenticationReply > or < EAPAuthenticationReply > message that indicates a *Invalid State for Request (252)* response code. Upon receipt of a WISPr response with a < AuthenticationReply > or < EAPAuthenticationReply > message that indicates a *Invalid State for Request (252)* response code, the client software SHALL restart the WISPr discovery procedure using an "Arbitrary URL" as described in section 7.7.

The access gateway SHALL always return a WISPr response in response to the WISPr request sent by the client software.

***Note**: the parameters "button=", FNAME=" and "OriginatingServer=" specified in the login request of WISPr 1.0 are not used by this specification.*

7.10.1 PAP Authentication Reply message

If the client software has initiated username/password (PAP) authentication, the access gateway SHALL return a WISPr response with an <AuthenticationReply> message in response to the WISPr request.

The **< AuthenticationReply > message** contains the following elements:

Field format/value	Required/ Optional
<MessageType> 120 </MessageType>	Required
<ResponseCode> {Response Code} </ResponseCode>	Required
<ReplyMessage> {Reply Message Text} </ReplyMessage>	Conditional Required
<LogoffURL> http[s]://{site specific logoff URL} </LogoffURL>	Conditional Required
<RedirectionURL> http[s]://{redirection URL} </RedirectionURL>	Optional
<StatusURL> http[s]://{status URL} </StatusURL>	Conditional Required
<MaxSessionTime> {Maximum Session Time} </MaxSessionTime>	Conditional Required

Table 11: AuthenticationReply Message Elements

The access gateway SHALL indicate one of the following response codes in the < AuthenticationReply > message:

Response Code	Response Meaning
50	Login succeeded (Access ACCEPT)
100	Login failed (Access REJECT)
102	Authentication server error/timeout
105	Network Administrator Error: No authentication server enabled
151	Login Aborted
252	Invalid state for WISPr request
253	MTU of AAA message is too big
254	Protocol error
255	Access gateway internal error

Table 12: AuthenticationReply Response Codes

7.10.2 EAP Authentication Reply message

If the client software has initiated EAP authentication, the access gateway SHALL return a WISPr response with an <EAPAuthenticationReply> message in response to the WISPr request.

The **< EAPAuthenticationReply > message** contains the following elements:

Field format/value	Required/ Optional
<MessageType> 121 </MessageType>	Required
<ResponseCode> {Response Code} </ResponseCode>	Required
<EAPMsg> Encoded EAP Request Message </EAPMsg>	Conditional Required
<LoginURL> https://{login URL} </LoginURL>	Conditional Required
<ReplyMessage> {Reply Message Text} </ReplyMessage>	Conditional Required

Field format/value	Required/ Optional
<LogoffURL> http[s]://{site specific logoff URL} </LogoffURL>	Conditional Required
<RedirectionURL> http[s]://{redirection URL} </RedirectionURL>	Optional
<StatusURL> http[s]://{status URL} </StatusURL>	Conditional Required
<MaxSessionTime> {Maximum Session Time} </MaxSessionTime>	Conditional Required

Table 13: EAPAuthenticationReply Message Elements

The access gateway SHALL indicate one of the following response codes in the < EAPAuthenticationReply > message:

Response Code	Response Meaning
10	Login challenged (Access CHALLENGE)
50	Login succeeded (Access ACCEPT)
100	Login failed (Access REJECT)
102	Authentication server error/timeout
105	Network Administrator Error: No authentication server enabled
151	Login Aborted
252	Invalid state for WISPr request
253	MTU of AAA message is too big
254	Protocol error
255	Access gateway internal error

Table 14: EAPAuthenticationReply Response Codes

For EAP authentication when Radius is used as the AAA protocol, the access gateway SHALL conform with RFC 3579.

If the access gateway receives a RADIUS/DIAMETER Access-Challenge, the access gateway SHALL return a WISPr Response with a <EAPAuthenticationReply> Message that indicates the *Login challenged (10)* response code. The <EAPAuthenticationReply> Message SHALL contain an <EAPMsg> element set to the encoded value of the EAP-Message Attribute received in the RADIUS/DIAMETER Access-Challenge (see RFC 3579). The <EAPAuthenticationReply> Message SHALL also contain one <LoginURL> element that indicates the secure URL to which the client software must POST the next WISPr request with the EAP response to the challenge. The process to send the EAP response is the same as described in 7.10. The <LoginURL> element SHALL not be present with another response code than *Login challenged (10)*.

If the client software requested an EAP authentication, and the access gateway receives a RADIUS/DIAMETER Access-Accept or Access-Reject, The <EAPAuthenticationReply> Message SHALL contain an <EAPMsg> element set to the encoded value of the EAP-Message Attribute received in the RADIUS/DIAMETER Access response.

If the client software requested an EAP authentication, the access gateway SHALL verify the Message-Authenticator Attribute of the RADIUS/DIAMETER Access reply (see RFC 3579). Should the Message-Authenticator be incorrect, the access gateway SHALL return a WISPr Response with a <EAPAuthenticationReply> Message that indicates *the Authentication server error/timeout (102)* response code.

***Note**: Client software must not consider that the EAP identifier field of the initial EAP Identity Request contained in the <Redirect> message will be 0. The access gateway MAY start with any value.*

7.10.3 Processing rules for the authentication reply

The following rules apply to both the < AuthenticationReply > and < EAPAuthenticationReply > messages.

If the access gateway has successfully abort the login process following the request from the client software (see section 7.13), the access gateway SHALL return a *Login Aborted (151)* response code.

The <ReplyMessage> element must be present in the <AuthenticationReply> or <EAPAuthenticationReply> messages when the access gateway receives a RADIUS Access-Response with one or several RADIUS/DIAMETER attribute 18, Reply-Message. When multiple RADIUS/DIAMETER Reply-Message attribute instances are present in the Access-Reply message, each instance SHALL be returned in a separate <Reply Message> element. The client software MUST expect one or more <ReplyMessage> elements. The order of the individual <Reply Message> elements SHALL reflect the order of the RADIUS Reply-Message attribute instances in the Access-Reply message. The <ReplyMessage> elements allow the AAA server or intermediate RADIUS proxies to provide human readable reasons for rejecting an authentication request.

If the access gateway receives a RADIUS/DIAMETER Access-Accept it SHALL return a WISPr response with *Login succeeded (50)* response code. This < AuthenticationReply > or < EAPAuthenticationReply > message:

- SHALL contain one <LogoffURL> element that contains the URL that the client software must use to request the termination of the session (see section 7.14).
- SHALL contain one <StatusURL> element that contains the URL that the client software may use to request the status of the session (see section 7.12).
- SHALL contain one <MaxSessionTime> element that contains the maximum time that the session may elapse before the access gateway terminates the session. This value SHALL be the minimum between any Session-Timeout Attribute indicated by the HSP AAA in the RADIUS/DIAMETER Access-Accept and any VNP configured value in the access gateway. If neither the HSP nor the VNP have a limitation on the duration of the session, the <AuthenticationReply> or <EAPAuthenticationReply> message SHALL not contain a <MaxSessionTime> element.

- MAY contain one <RedirectionURL> element to indicate to the client software a page to be displayed to the user following the completion of the authentication. If the access gateway received an RADIUS Access-Reply with a Vendor Attribute of Vendor ID: 14122 that contains a Redirection URL (see WRIX-i), it MAY set the <RedirectionURL> element to the URL contained in the Radius attribute. The client software MAY ignore the <RedirectionURL> element.

7.11 WISPr Login Results Polling

As indicated in section 7.10, the access gateway MAY return a WISPr Response with an <PollNotification> message with a *No Error (0)* response code to indicate the use of the authentication polling. If the client software receives a WISPr response with a <PollNotification> message that indicates the *No Error (0)* response code, the client software SHALL begin the authentication results polling procedure. The polling procedure SHALL consist of one or more WISPr requests to the <LoginResultsURL> element of the last received <PollNotification> message until the access gateway returns a <AuthenticationReply> message (see section 7.10.1) or an <EAPAuthenticationReply> message (see section 7.10.2).

A <PollNotification> message with *No Error(0)* response code SHALL include a <LoginResultsURL> element that indicates to which URL the client software SHALL submit the next WISPr request. It MAY include a <Delay> element that indicates in seconds the time that client software MUST wait before sending the next WISPr Request. If a <Delay> element is present, the client software SHALL wait the number of seconds indicated before sending the next WISPr request to query the status of the authentication. If the <PollNotification> Message does not contain a <Delay> element, it shall wait at the minimum 1 second before the next query.

The access gateway SHALL return a WISPr response with a <PollNotification> message if the result from the authentication is not yet know by the Access Gateway. If the access gateway has successfully aborted the login process following the request from the client software (see section 7.13), the access gateway SHALL return a *Login Aborted (151)* response code.

The **< PollNotification > message** contains the following elements:

Field format/value	Required/ Optional
<MessageType> 170 </MessageType>	Required
<ResponseCode> {Response Code} </ResponseCode>	Required
<LoginResultsURL> http[s]://{site specific login URL} </LoginResultsURL>	Required
<Delay> {Number of seconds data} </Delay>	Optional

Table 15: PollNotification Message Elements

The access gateway SHALL indicate one of the following response codes in the < PollNotification> message:

Response Code	Response Meaning
0	No Error
151	Login Aborted
254	Protocol error
255	Access gateway internal error

Table 16: PollNotification Response Codes

7.12 WISPr Status

Once a response code *Login succeeded (50)* has been received by the client software, and until the client software issues a logoff request, the client software MAY request the access gateway to return the status of the session.

To initiate a status query the client software SHALL perform a WISPr request to the <StatusURL> returned in either the <AuthenticationReply> or < EAPAuthenticationReply > message.

The access gateway shall return a WISPr response with a <StatusReply> message to the status query request.

The **<StatusReply> message** contains the following elements:

Field format/value	Required/ Optional
<MessageType> 160 </MessageType>	Required
<ResponseCode> {Response Code} </ResponseCode>	Required
<Status> {Status code} </Status>	Required

Table 17: StatusReply Message Elements

The access gateway SHALL indicate one of the following status codes in the <StatusReply> message:

Status Code	Response Message
0	Session inactive
1	Session active

Table 18: StatusReply Status Code

The access gateway SHALL indicate one of the following response codes in the <StatusReply> message:

Response Code	Response Message
0	No error
254	Protocol error
255	Access gateway internal error

Table 19: StatusReply Response Codes

The client software SHALL only send a WISPr status request if it suspects that the session was terminated. It SHALL not send it in a repeated fashion at regular intervals as it might overload the access gateway with processing.

7.13 WISPr Abort Login

Access gateway implementation of the abort login process is optional.

When the gateway indicated a <Redirect> message with an <AbortLoginURL> element and the client software wants to abort the undergoing login request, whether in a direct or polling situation, the client MAY perform a WISPr request to the <AbortLoginURL> that was contained in the <Redirect> message. The Abort Login request can be made concurrently to an outstanding Login or Login Results Polling request. The Abort Login request can also be requested when the client software is waiting to trigger the next Login Results Polling request.

When an Abort Login request is received by the access gateway, every attempt should be made to abort the session cleanly without signalling accounting messages and/or generating an accounting record.

An access gateway SHALL always return an <AbortLoginReply> message as the result of an Abort Login request from the client software. The <AbortLoginReply> SHALL contain:

- A response code of value *Login Aborted (151)* if the access gateway could cleanly abort the session and no AAA accounting messages have been generated
- A response code of value *Login Succeeded (50)* if the access gateway could not abort the login request and the session is active.

- A response code of value *Protocol Error (254)* if the client software attempts to abort a session while it hasn't requested a login request and does not have an active session open.

A client software SHALL not send an Abort Login request unless it is in the login procedure and the result is pending.

In the login polling situation, when the client software receives an <AbortLoginReply> message with *Login Succeeded (50)* response code and the client software has no Login Results Polling request active, the client software SHALL send a Login Results Polling request to receive the <AuthenticationReply> or <EAPAuthenticationReply> message. In all other cases when the client software receives an <AbortLoginReply> message with *Login Succeeded (50)* response code, the client software will receive the <AuthenticationReply> or <EAPAuthenticationReply> message via the outstanding Login request or Login Results Polling request.

The <AbortLoginReply> message SHALL contain the elements described in the table below.

The **< AbortLoginReply > message** contains the following elements:

Field format/value	Required/ Optional
<MessageType> 150 </MessageType>	Required
<ResponseCode> {Response Code} </ResponseCode>	Required
<LogoffURL> </LogoffURL>	Deprecated (Note 1)

Note 1: The <LogoffURL> element defined in WISPr 1.0 is deprecated and SHALL not be used in WISPr 2.0. The client software must ignore this value upon receipt in a WISPr 2.0 <AbortLoginReply> message.

Table 20: AbortLoginReply Message Elements

The access gateway SHALL indicate one of the following response codes in the <AbortLoginReply> message:

Response Code	Response Meaning
50	Login succeeded (Access ACCEPT)
151	Login aborted
254	Protocol error
255	Access gateway internal error

Table 21: AbortLoginReply Response Codes

7.14 WISPr Logoff

To initiate a logoff, the client software SHALL perform a WISPr request to the <LogoffURL> returned in either the <AuthenticationReply> or < EAPAuthenticationReply > message.

The access gateway shall return a WISPr response with a <LogoffReply> message to the logoff request.

The <LogoffReply> message contains the following elements:

Field format/value	Required/ Optional
<MessageType> 130 </MessageType>	Required
<ResponseCode> {Response Code} </ResponseCode>	Required

Table 22: LogoffReply Message Elements

The access gateway SHALL indicate one of the following response codes in the <LogoffReply> message:

Response Code	Response Meaning
150	Logoff succeeded
152	Invalid session
254	Protocol error
255	Access gateway internal error

Table 23: LogoffReply Response Codes

An access gateway that receives a logoff request for a session for which it cannot find a context shall respond with an *Invalid Session* (152) response code.

8 Implementation considerations (Informative)

8.1 Interaction with Virtual Private Networks

It is common practice to install the access gateway is a private IP subnet. The access gateway is therefore reachable from client software only on a private IP address.

A user may launch VPN software once connected to the Internet as the result of a WISPr session. If the VPN is a full tunnel VPN, the WISPr client software will fail in attempting any request on the <LogoffURL> or <StatusURL> if the access gateway is reachable through a private IP address.

To allow a proper logoff, the user or client software must first stop the VPN and then proceed with a WISPr logoff.

8.2 Interaction with other networking interfaces

A User Device might have several IP interfaces active, one being an IEEE 802.11 associated with a public WLAN hotspot.

The WISPr protocol only works when the arbitrary URL is received by the access gateway. Depending on the routing preference in the IP routing table of the User Device, it might therefore not be possible to open a WISPr session when another IP interface is active. This problem can however be solved by setting specific routing rules in the IP routing table of the User Device.

This dependency on the IP routing is different from an IEEE 802.1x authentication that allows the authentication of a given interface regardless of the routing status of the IP layer.

Implications of the use of IPv6 are for further study.

8.3 IP considerations

It is common that public Wi-Fi hotspots are in a private IP subnet. The User Device gets an IP address for its 802.11 interface via DHCP. The DHCP server located in the hotspot operator will also provide the IP addresses of the DNS servers and the default IP gateways.

User Devices that have static IP addresses for their IEEE 802.11 interfaces will, in most cases, not get IP connectivity unless they enable DHCP. User Devices that point to fixed DNS server might also encounter problems when trying to resolve the WISPr login URL. It is recommended to use the local DNS servers indicated by DHCP.

8.4 Arbitrary URL

Networks do commonly block HTTP requests to sites that are serving client applications (e.g. updates for virus database, update of the operating systems, instant messaging) to avoid unnecessary trigger of the WISPr protocol. The client software shall therefore not use such URLs as arbitrary URLs.

8.5 Authentication polling

It is recommended that access gateways wait for the execution of the authentication request and return the result directly in the <AuthenticationReply> or <EAPAuthenticationReply> messages. With the authentication polling mechanism described in this specification, the access gateway may choose, instead, to immediately return an <PollNotificiation> message causing the client software to poll for the result of its authentication request. Authentication polling is not recommended because it introduces more processing, more traffic and additional delays. The delay affects the overall time required to login and therefore the user experience.

8.6 Use of the Proxy message for network discovery

WISPr 2.0 introduces an extension of the <Proxy> message to enable the support of WISPr in WLAN infrastructures that are shared between several hotspot operators (e.g. airports). The provider of the WLAN infrastructure commonly does not provide authentication services and each hotspot operator has its own

access gateways and WISPr servers. The standard trigger URL mechanism is not sufficient in situations where the client software must choose between several WISPr servers. This situation is very often solved by misusing the HTTP User-Agent to route the trigger request to the correct WISPr server. This solution is unsatisfactory for many reasons.

With WISPr 2.0 the WLAN infrastructure provider can with the <Proxy> message indicate to the client software the hosted VNPs and what URL must be requested by the client software to access the WISPr service from a specific VNP.

Special care must however be taken to ensure backward compatibility with WISPr 1.0 client software if the WLAN infrastructure is not providing its own WISPr service. Such WLAN infrastructure may now add a <Proxy> message that contains the list of VNPs served. The NextURL element of the <Proxy> message shall point to another page, hosted by the WLAN infrastructure, where a <Redirect> message with Response Code 105 is returned. With this method, WISPr 1.0 client software will parse the <Proxy> message and ignore the <OperatorURL> tags. It will follow the NextURL element and indicate to the client software that the infrastructure is not configured for roaming.

8.7 EAP state machines

[RFC 4137] describes peer and authenticator EAP state machines that can be used for the EAP over WISPr authentication method described in this document. Unlike EAP over LAN that allow unsolicited communication in both directions between the peer and the authenticator, WISPr is based on HTTP, a client (peer) – server (authenticator) communication channel. [RFC 4137] can be adapted as follows to take into account the client – server architecture of the transport channel between the peer and the authenticator:

- There is no unsolicited trigger from the authenticator to the peer to start the EAP authentication procedure. The trigger of a WISPr Response with a <Redirect> message shall be equivalent to a “portEnabled” signal and the <Redirect> message shall contain the EAP-Request identity sent by the authenticator to the peer.
- Due to the client/server architecture of WISPr, there is no possible unsolicited retransmission of a sent EAP request by the authenticator. This is equivalent to considering that the value of MaxRetrans in [RFC 4137] is equal to zero. A peer may however retransmit the last EAP response and therefore trigger the resending of the next EAP-Request if it didn’t receive the next EAP-Request following its last EAP-Response.

  It should be noted that HTTP uses TCP, a communication channel that have its own retransmission mechanism.
- The receipts by the EAP authenticator/WISPr server of an “AbortLogin” WISPr request, while the authenticator is not in “SUCCESS” or “ FAILURE” state (see sections 5 and 7 of [RFC 4137]), shall be treated as an EapRestart set to TRUE (see sections 5 and 7 of [RFC 4137]).

8.8 Identification of the User Device and Client Software

The Access Gateway must maintain a context for each WISPr client. It is recommended that the Access Gateway uses the MAC address of User Device, combined with the IP address assigned to the User Device and the HTTP User Agent of the client software as the key to this context.

Access Gateway should take into account that more than one client software implementing WISPr might reside on the User Device. Those application might have auto-login features that will attempt concurrently to login to a WLAN service. Each application might be associated with a different user account. The access

gateway should use the HTTP User Agent to distinguish between a WISPr request retransmission from a single application from a concurrent login attempt from different applications.

The access gateway can use the error code 252 to indicate to an application that its request to login is outdated and that the application should re-trigger the WISPr discovery process from start to get the current state of connection.

8.9 Retransmission of WISPr requests

Access Gateway should take into account that client software may re-transmit an outstanding WISPr request. An access gateway may send the same WISPr response to all outstanding re-transmission of the same WISPr request.

An access gateway should not respond to the re-transmission of a WISPr request by closing the socket as this might be interpreted by the client software as a loss of connectivity to the access gateway. The access gateway should also not return an HTTP response without a WISPr XML block

8.10 Security considerations

It is recommended that access gateways do not only rely solely on the IP address of User Device bur rather on the tuple “IP address”, “MAC address” to identify an end-system and make more difficult address spoofing attacks. For EAP authentication, it is recommended that the access gateway verifies that the NAIs provided in each WISPr login request during the EAP challenge/response process are the same.

Implementations must take special care that the Abort Login procedure cannot be used as a denial of service attack to prevent users from creating sessions by systematically aborting them.

8.11 Client software migration to WISPr 2.0

WISPr 2.0 introduces both a version negotiation and a new format of the HTTP User-Agent. VNPs do process the HTTP User-Agent prior to the handling the version negotiation. It is therefore important to plan the upgrade of the HTTP User-Agent first and then the WISPr 2.0 logic. It should be noted that the new HTTP User-Agent structure defined in WISPr 2.0 is compatible with WISPr 1.0.

HSP that add WISPr 2.0 in their client software MUST ensure first that the new HTTP User-Agent value is supported by all roaming VNP that implements HTTP User-Agent filtering. From a VNP’s stand point, the use of the new HTTP User-Agent structure is independent from the decision to upgrade its access gateways to WISPr 2.0 as the new structure is fully compatible with WISPr 1.0.

Once all VNPs using HTTP User-Agent filtering are able to handle the new HTTP User-Agent string, the HSP can release its WISPr 2.0 client software and use the new HTTP User-Agent string:

- If the client software receives a <VersionHigh> element that indicates “2.0”, the client software can proceed using the WISPr 2.0 specification
- If the client software does not receive a <VersionHigh> element that indicates “2.0”, the client software must proceed with the WISPr 1.0 recommendation.

8.12 Hotspot network migration to WISPr 2.0

If the network uses HTTP User-Agent filtering, the operator shall provision that every HTTP request with an HTTP User-Agent value that starts by WISPR! is a request to the WISPr protocol.

The hotspot network can then add support in its access gateways for WISPr 2.0. The version negotiation ensures that if the access gateways support both WISPr 1.0 and WISPr 2.0, client software that does not support WISPr 2.0 will operate using WISPr 1.0.

9 References

[GIS 1.5] "Generic Interface Specification", IPass Inc, October 2005

[RFC 2616] "Hypertext Transfer Protocol -- HTTP/1.1", IETF, June 1999

[RFC 3986] "Uniform Resource Identifiers (URI): Generic Syntax", IETF, January 2005

[RFC 2965] "HTTP State Management Mechanism", IETF, October 2000

[RFC 3548] "The Base16, Base32, and Base64 Data Encodings", IETF, July 2003

[RFC 3579] "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", IETF, September 2003

[RFC 3748] "Extensible Authentication Protocol (EAP)", IETF, June 2004

[RFC 4137] "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", IETF, August 2005

[WISPr 1.0] "Wireless ISP roaming 1.0", February 2003

[WRIX] "Wireless Roaming Intermediary eXchange (WRIX) 1.03", Wireless Broadband Alliance, July 2007

[WRIX-L] "Location Feed Format & File Exchange Standard 1.01", Wireless Broadband Alliance, June 2009

[XML 1.0] "Extensible Markup Language (XML) 1.0 (Fifth Edition)", W3C, November 2008

Appendix A Example of WISPr dialog

The messages documented in this section are associated with their [*originator*], either the client software or the access gateway.

A.1 Authentication procedure for username/password (no polling)

In this example the access gateway does support both WISPr 1.0 and WISPr 2.0 but does not support EAP authentication.

Authentication Procedure Initiation [request from client]

```
GET / HTTP/1.0
User-Agent: WISPR!My Client Software 4.0
{Other HTTP headers}
```

Authentication Redirect [response from gateway]

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
    <Redirect>
       <MessageType>100</MessageType>
       <ResponseCode>0</ResponseCode>
       <AccessProcedure>1.0</AccessProcedure>
       <VersionLow>1.0</VersionLow>
       <VersionHigh>2.0</VersionHigh>
       <AccessLocation>TMOBILE:US, 10403 SFO San Francisco Intl
          Airport</AccessLocation>
       <LocationName> SFO San Francisco Intl Airport </LocationName>
       <LoginURL>https://www.acmewisp.com/login?sid=4a45b7&amptry=1</LoginURL>
    </Redirect>
</WISPAccessGatewayParam>
--> </HTML>
```

SSL protected Authentication Request [request from client]

```
POST /login?sid=4a45b7&try=1 HTTP/1.0
User-Agent: WISPR!My Client Software 4.0
{Other HTTP headers}

WISPrVersion=2.0&UserName=joseph%40company.com&Password=wi%23sp%25r
```

a) Authentication Reply [response from gateway] (Login Successful)

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
```

```
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <AuthenticationReply>
        <MessageType>120</MessageType>
        <ResponseCode>50</ResponseCode>
        <MaxSessionTime>3600</MaxSessionTime>
        <StatusURL>http://www.acmewisp.com/status?sid=4a45b7</StatusURL>
        <LogoffURL>http://www.acmewisp.com/logoff?sid=4a45b7</LogoffURL>
   </AuthenticationReply>
</WISPAccessGatewayParam>
--> </HTML>
```

b) Authentication Reply [response from gateway] (credentials rejected)

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!--
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <AuthenticationReply>
        <MessageType>120</MessageType>
        <ResponseCode>100</ResponseCode>
        <ReplyMessage>Invalid Username or Password</ReplyMessage>
   </AuthenticationReply>
</WISPAccessGatewayParam>
--> </HTML>
```

c) Authentication Reply [response from gateway] (invalid realm)

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!--
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <AuthenticationReply>
        <MessageType>120</MessageType>
        <ResponseCode>105</ResponseCode>
        <ReplyMessage>No route for realm</ReplyMessage>
   </AuthenticationReply>
</WISPAccessGatewayParam>
--> </HTML>
```

Client-initiated Connection Termination (logoff) of Authenticated User [request from client]

```
GET /logoff?sid=4a45b7 HTTP/1.1
User-Agent: WISPR!My Client Software 4.0
{Other HTTP headers}
```

Logoff Reply [response from gateway]

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!--
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <LogoffReply>
       <MessageType>130</MessageType>
       <ResponseCode>150</ResponseCode>
   </LogoffReply>
</WISPAccessGatewayParam>
--> </HTML>
```

A.2 Authentication procedure for username/password (with polling)

In this example the access gateway only supports WISPr 2.0 but and does support EAP authentication. The client software however uses username/password for authentication.

Authentication Procedure Initiation [request from client]

```
GET / HTTP/1.1
User-Agent: WISPR!My Client Software 5.0
{Other HTTP headers}
```

Authentication Redirect [response from gateway]

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
    <Redirect>
       <MessageType>100</MessageType>
       <ResponseCode>0</ResponseCode>
       <VersionLow>2.0</VersionLow>
       <VersionHigh>2.0</VersionHigh>
       <EAPMsg>AQAABQE=</EAPMsg>
       <AccessLocation>TMOBILE:US, 10403 SFO San Francisco Intl
          Airport</AccessLocation>
       <LocationName> SFO San Francisco Intl Airport </LocationName>
       <LoginURL>https://www.acmewisp.com/login?token=af4szut79684235</LoginURL>
    </Redirect>
</WISPAccessGatewayParam>
--> </HTML>
```

SSL protected Authentication Request [request from client]

```
POST /login?sid=4a45b7&try=1 HTTP/1.1
User-Agent: WISPR!My Client Software 5.0
{Other HTTP headers}

WISPrVersion=2.0&UserName=weroam%2Fjoseph%40company.com&Password=xxxxx
```

(a) Authentication Poll Notification [response from gateway]

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <PollNotification>
```

```
      <MessageType>170</MessageType>
      <ResponseCode>0</ResponseCode>
      <LoginResultsURL>http://srv2.acmewisp.com/loginresults?
token=bd53gtq79684235</LoginResultsURL>
   </PollNotification>
</WISPAccessGatewayParam>
--> </HTML>
```

Client-initiated Authentication Result Poll [client]

```
GET /loginresults?token=bd53gtq79684235 HTTP/1.1
User-Agent: WISPR!My Client Software 5.0
{Other HTTP headers}
```

(a) Authentication Poll Notification [response from gateway] (Result Still Pending-repeat polling operation again in 5 seconds)

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <PollNotification>
      <MessageType>170</MessageType>
      <ResponseCode>0</ResponseCode>
      <LoginResultsURL>http://srv2.acmewisp.com/loginresults?
token=tc43sui79684235</LoginResultsURL>
      <Delay>5</Delay>
   </PollNotification>
</WISPAccessGatewayParam>
--> </HTML>
```

Client-initiated Authentication Result Poll [client]

```
GET /loginresults?token=tc43sui79684235 HTTP/1.1
User-Agent: WISPR!My Client Software 5.0
{Other HTTP headers}
```

b) Authentication Reply [gateway] (Login Successful)

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <AuthenticationReply>
      <MessageType>120</MessageType>
```

```
        <ResponseCode>50</ResponseCode>
        <MaxSessionTime>3600</MaxSessionTime>
        <StatusURL>http://www.acmewisp.com/status?token=aa76bv179684235 </StatusURL>
        <LogoffURL>http://www.acmewisp.com/logoff?token=r2wf6cd79684235 </LogoffURL>
   </AuthenticationReply>
</WISPAccessGatewayParam>
--> </HTML>
```

Verification of the session status [request from client]

```
GET /status?token=aa76bv179684235 HTTP/1.1
User-Agent: WISPR!My Client Software 5.0
{Other HTTP headers}
```

Verification of the session status [response from gateway]

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <StatusReply>
        <MessageType>160</MessageType>
        <ResponseCode>0</ResponseCode>
        <StatusReply>1</Status>
   </StatusReply>
</WISPAccessGatewayParam>
--> </HTML>
```

The login rejects and logoff examples are the same as in example A.1

A.3 Proxy procedure with shared Wi-Fi infrastructure

Authentication Procedure Initiation [request from client]

```
GET / HTTP/1.1
User-Agent: WISPR!My Client Software 6.0
{Other HTTP headers}
```

Proxy Reply with selection of WISP [response from gateway]

```
<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <Proxy>
        <MessageType>110</MessageType>
        <ResponseCode>200</ResponseCode>
        <OperatorURL ID=”ORANGE:FR”>  http://wispr.orange.fr/wispr.html</OperatorURL>
```

```
      <OperatorURL
   ID=”SFR:FR”>http://www.meteor.fr/redirect?source=airport1</OperatorURL>

       <NextURL>http://www.airport1.com/nowispr.html</NextURL>
   </Proxy>
</WISPAccessGatewayParam>
--> </HTML>
```

Client software selects to authenticate through Orange France.

Authentication Procedure Initiation [request from client]

Request sent to wispr.orange.fr:

```
GET /wispr.html HTTP/1.1
User-Agent: WISPR!My Client Software 7.0
{Other HTTP headers}
```

```
<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
    <Redirect>
       <MessageType>100</MessageType>
       <ResponseCode>0</ResponseCode>
       <VersionLow>1.0</VersionLow>
       <VersionHigh>2.0</VersionHigh>
       <AccessLocation>CDATA[[ORANGE:FR, Airport Charles De Gaulles/Terminal 1
          ]]</AccessLocation>
       <LocationName> CDATA[[Airport Charles De Gaulles/Terminal 1]]</LocationName>
       <LoginURL>CDATA[[https://wisprsvr1.orange.fr/login?session=1739538&time=79458
3723242]]</LoginURL>
    </Redirect>
</WISPAccessGatewayParam>
--> </HTML>
```

A.4 Authentication procedure for EAP-SIM (no polling)

In this example the access gateway does support both WISPr 1.0 and WISPr 2.0 including EAP authentication.

Authentication Procedure Initiation [request from client]

```
GET / HTTP/1.0
User-Agent: WISPR!My Client Software 6.0
{Other HTTP headers}
```

Authentication Redirect [response from gateway]

```
HTTP 200 OK
{Other HTTP headers}
```

```
<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
    <Redirect>
       <MessageType>100</MessageType>
       <ResponseCode>0</ResponseCode>
       <AccessProcedure>1.0</AccessProcedure>
       <VersionLow>1.0</VersionLow>
       <VersionHigh>2.0</VersionHigh>
       <EAPMsg>AQAABQE=</EAPMsg>
       <AccessLocation>SWISSCOM:CH, 05647</AccessLocation>
       <LocationName> Hotel Bellevue, Bern </LocationName>
       <LoginURL>https://www.acmewisp.com/login?sid=4a45b7</LoginURL>
    </Redirect>
</WISPAccessGatewayParam>
--> </HTML>
```

SSL protected Authentication Request [request from client] (EAP-Identity)

```
POST /login?sid=4a45b7 HTTP/1.0
User-Agent: WISPR!My Client Software 6.0
{Other HTTP headers}

WISPrVersion=2.0&UserName=weroam%2F1295023820005362%40mnc002.mcc295.3gppnetwork.org
&WISPrEAPMsg=AgAAMgEyOTUwMjM4MjAwMDUzNjJAbW5jMDAyLm1jYzI5NS4zZ3BwbmV0d29yay5vcmc%3D
```

Authentication Reply [response from gateway]

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!--
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <EAPAuthenticationReply>
       <MessageType>121</MessageType>
       <ResponseCode>10</ResponseCode>
       <EAPMsg>AQEAFBIKAAANAQAADwIAAgABAAA=</EAPMsg>
       <LoginURL>https://www.acmewisp.com/login?sid=4a45b7</LoginURL>
   </EAPAuthenticationReply>
</WISPAccessGatewayParam>
--> </HTML>
```

SSL protected Authentication Request [request from client] (EAP SIM/Start)

```
POST /login?sid=4a45b7 HTTP/1.0
User-Agent: WISPR!My Client Software 6.0
{Other HTTP headers}
```

```
WISPrVersion=2.0&UserName=weroam%2F1295023820005362%40mnc002.mcc295.3gppnetwork.org
&WISPrEAPMsg=AgEAVBIKAAAHBQAA6LUBW2YTEG5qfPlRF2JSYhABAAEODQAtMjk1MDIzODIwMDA1MzYyQG
1uYzAwMi5tY2MyOTUuM2dwcG5ldHdvcmsub3JnAAAA
```

Authentication Reply [response from gateway]

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!--
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <EAPAuthenticationReply>
       <MessageType>121</MessageType>
       <ResponseCode>10</ResponseCode>
       <EAPMsg>AQIAUBILAAABDQAA7L/lxpFOnaNIH/xhGxglFG5nELC7HgUCESJgFXqs0OgkAGM9/+2O6
   O/VTCgf0WeHCwUAAMHG3DlEi2yOPx3B/D6/qQw=</EAPMsg>
       <LoginURL>https://www.acmewisp.com/login?sid=4a45b7</LoginURL>
   </EAPAuthenticationReply>
</WISPAccessGatewayParam>
--> </HTML>
```

SSL protected Authentication Request [request from client] (EAP SIM/Challenge)

```
POST /login?sid=4a45b7 HTTP/1.0
User-Agent: WISPR!My Client Software 6.0
{Other HTTP headers}

WISPrVersion=2.0&UserName=weroam%2F1295023820005362%40mnc002.mcc295.3gppnetwork.org
&WISPrEAPMsg=AgIAHBILAAALBQAAQgWJoEnY7UZrI60RM2RwcA%3D%3D
```

Authentication Reply [response from gateway] (EAP Success – Login successful)

```
HTTP 200 OK
{Other HTTP headers}

<HTML> <!—
<?xml version="1.0" encoding="UTF-8"?>
<WISPAccessGatewayParam
  xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”
  xsi:noNamespaceSchemaLocation="http://www.wballiance.net/wispr/wispr_2_0.xsd">
   <AuthenticationReply>
       <MessageType>120</MessageType>
       <ResponseCode>50</ResponseCode>
       <MaxSessionTime>600</MaxSessionTime>
       <EAPMsg>AwAABA==</EAPMsg>
       <StatusURL>http://www.acmewisp.com/status?sid=4a45b7</StatusURL>
       <LogoffURL>http://www.acmewisp.com/logoff?sid=4a45b7</LogoffURL>
   </AuthenticationReply>
</WISPAccessGatewayParam>
--> </HTML>
```

The logoff example is the same as in example A.1

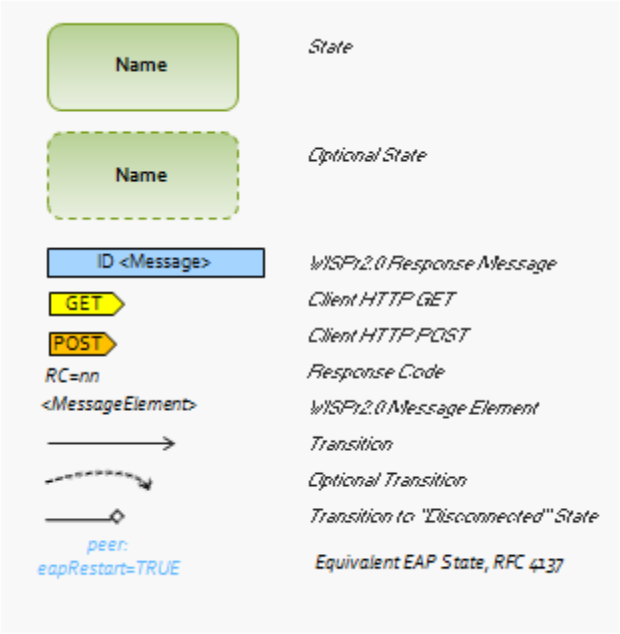
Appendix B XML schema (Normative)

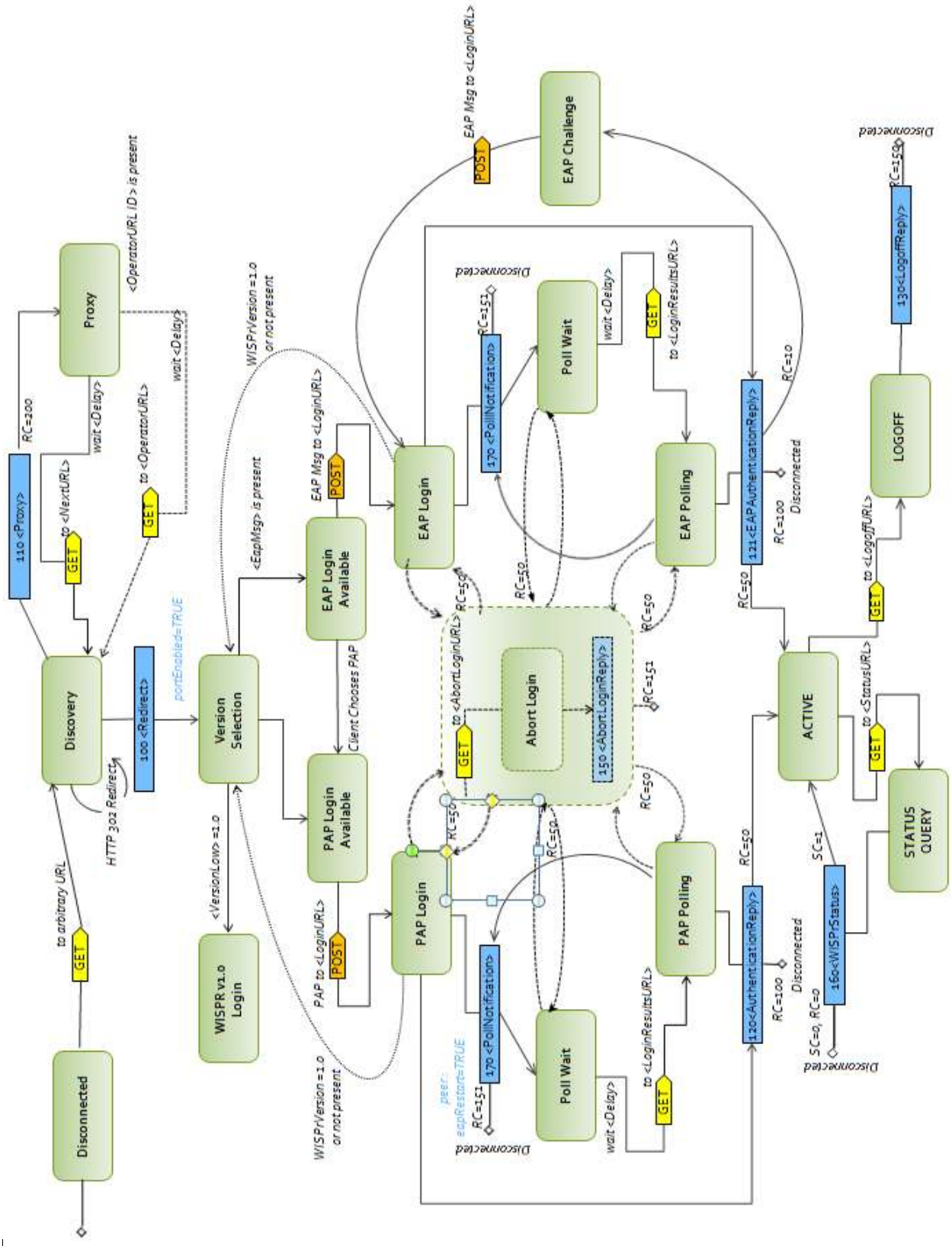
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<!-- definition of simple elements -->
    <xs:simpleType name="AbortLoginURLType">
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:simpleType name="NextURLType">
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:simpleType name="AccessLocationType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="LocationNameType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="LoginURLType">
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:simpleType name="RedirectionURLType">
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:simpleType name="MessageTypeType">
        <xs:restriction base="xs:nonNegativeInteger"/>
    </xs:simpleType>
    <xs:simpleType name="ResponseCodeType">
        <xs:restriction base="xs:nonNegativeInteger"/>
    </xs:simpleType>
    <xs:simpleType name="ReplyMessageType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="LoginResultsURLType">
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:simpleType name="LogoffURLType">
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:simpleType name="DelayType">
        <xs:restriction base="xs:nonNegativeInteger"/>
    </xs:simpleType>
    <xs:simpleType name="StatusURLType">
        <xs:restriction base="xs:anyURI"/>
    </xs:simpleType>
    <xs:simpleType name="StatusType">
        <xs:restriction base="xs:integer">
            <xs:pattern value="[0-1]/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="WISPrVersionType">
        <xs:restriction base="xs:string">
            <xs:pattern value="([0-9])+\.[0-9])+"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="MaxSessionTimeType">
        <xs:restriction base="xs:unsignedLong”/>
```

```
    </xs:simpleType>
    <xs:simpleType name="EAPMsgType">
        <xs:restriction base="xs: base64Binary"/>
    </xs:simpleType>
    <xs:complexType name="OperatorURLType">
        <xs:simpleContent>
            <xs:extension base="xs:anyURL"/>
            <xs:attribute name="ID" type="string" use="required"/>
        </xs:simpleContent>
    </xs:complexType>

    <!-- definition of root WISPrAccessGatewayParam tag -->
    <xs:element name="WISPAccessGatewayParam">
        <xs:complexType>
        <xs:choice>
            <xs:element name="Redirect" type="RedirectType"/>
            <xs:element name="Proxy" type="ProxyType"/>
            <xs:element name="AuthenticationReply" type="AuthenticationReplyType"/>
            <xs:element name="EAPAuthenticationReply" type="EAPAuthenticationReplyType"/>
            <xs:element name="LogoffReply" type="LogoffReplyType"/>
            <xs:element name="AbortLoginReply" type="AbortLoginReplyType"/>
            <xs:element name="StatusReply" type="StatusReplyType"/>
            <xs:element name="PollNotification" type="PollNotificationType"/>
        </xs:choice>
        </xs:complexType>
    </xs:element>

    <!-- definition of WISPr messages -->

    <xs:complexType name="ProxyType">
        <xs:all>
            <xs:element name="MessageType" type="MessageTypeType"/>
            <xs:element name="ResponseCode" type="ResponseCodeType"/>
            <xs:element name="NextURL" type="NextURLType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="Delay" type="DelayType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="OperatorURL" type="OperatorURLType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:all>
    </xs:complexType>
    <xs:complexType name="RedirectType">
        <xs:all>
            <xs:element name="MessageType" type="MessageTypeType"/>
            <xs:element name="ResponseCode" type="ResponseCodeType"/>
            <xs:element name="VersionLow" type="WISPrVersionType"/>
            <xs:element name="VersionHigh" type="WISPrVersionType"/>
            <xs:element name="AccessLocation" type="AccessLocationType"/>
            <xs:element name="LocationName" type="LocationNameType"/>
            <xs:element name="LoginURL" type="LoginURLType"/>
            <xs:element name="AbortLoginURL" type="AbortLoginURLType" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="EAPMsg" type="EAPMsgType" minOccurs="0" maxOccurs="1"/>
        </xs:all>
    </xs:complexType>
    <xs:complexType name="AuthenticationReplyType">
        <xs:all>
            <xs:element name="MessageType" type="MessageTypeType"/>
            <xs:element name="ResponseCode" type="ResponseCodeType"/>
```

```
            <xs:element name="ReplyMessage" type="ReplyMessageType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="LogoffURL" type="LogoffURLType" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="RedirectionURL" type="RedirectionURLType" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="StatusURL" type="StatusURLType" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="MaxSessionTime" type="MaxSessionTimeType" minOccurs="0"
maxOccurs="1"/>
        </xs:all>
    </xs:complexType>
    <xs:complexType name="EAPAuthenticationReplyType">
        <xs:all>
            <xs:element name="MessageType" type="MessageTypeType"/>
            <xs:element name="ResponseCode" type="ResponseCodeType"/>
            <xs:element name="EAPMsg" type="EAPMsgType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="LoginURL" type="LoginURLType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="ReplyMessage" type="ReplyMessageType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="LogoffURL" type="LogoffURLType" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="RedirectionURL" type="RedirectionURLType" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="StatusURL" type="StatusURLType" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="MaxSessionTime" type="MaxSessionTimeType" minOccurs="0"
maxOccurs="1"/>
        </xs:all>
    </xs:complexType>
    <xs:complexType name="PollNotificationType">
        <xs:all>
            <xs:element name="MessageType" type="MessageTypeType"/>
            <xs:element name="ResponseCode" type="ResponseCodeType"/>
            <xs:element name="LoginResultsURL" type="LoginResultsURLType" minOccurs="0"
maxOccurs="1"/>
            <xs:element name="Delay" type="DelayType" minOccurs="0" maxOccurs="1"/>
        </xs:all>
    </xs:complexType>
    <xs:complexType name="LogoffReplyType">
        <xs:sequence>
            <xs:element name="MessageType" type="MessageTypeType"/>
            <xs:element name="ResponseCode" type="ResponseCodeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="StatusReplyType">
        <xs:sequence>
            <xs:element name="MessageType" type="MessageTypeType"/>
            <xs:element name="ResponseCode" type="ResponseCodeType"/>
            <xs:element name="Status" type="StatusType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="AbortLoginReplyType">
        <xs:sequence>
            <xs:element name="MessageType" type="MessageTypeType"/>
            <xs:element name="ResponseCode" type="ResponseCodeType"/>
        </xs:sequence>
    </xs:complexType>
```

Appendix C WISPr Finite State Machine (Informative)




Proxy
<OperatorURL ID > is present
wait <Delay>
RC=200
wait <Delay>
110 <Proxy>
GET
to <NextURL>
GET
to <OperatorURL>
Discovery
100 <Redirect>
portEnabled=TRUE
HTTP 302 Redirect
to arbitrary URL
GET
Disconnected
Version Selection
<VersionLow> =1.0
WISPR v1.0 Login
WISPrVersion =1.0 or not present
<EapMsg> is present
EAP Login Available
EAP Msg to <LoginURL>
POST
Client Chooses PAP
PAP Login Available
PAP to <LoginURL>
POST
EAP Login
EAP Msg to <LoginURL>
POST
EAP Challenge
170 <PollNotification>
RC=151
Disconnected
Poll Wait
wait <Delay>
GET
to <LoginResultsURL>
EAP Polling
RC=10
121<EAPAuthenticationReply>
RC=100
Disconnected
RC=50
to <AbortLoginURL>
GET
Abort Login
150 <AbortLoginReply>
RC=151
RC=50
PAP Login
170 <PollNotification>
RC=151
Disconnected
peer: eapRestart=TRUE
Poll Wait
wait <Delay>
GET
to <LoginResultsURL>
PAP Polling
RC=50
120<AuthenticationReply>
RC=100
Disconnected
SC=0, RC=0
Disconnected
160<WISPrStatus>
SC=1
ACTIVE
GET
to <StatusURL>
STATUS QUERY
GET
to <LogoffURL>
LOGOFF
130<LogoffReply>
RC=150
Disconnected

Appendix D WBA format of Location Name (Informative) defined in WRIX 1.03

A text string of no more than 253 bytes that is used to identify the VNP and the location at which service is provided. The format of the string must be

```
Location-Name       =
      ( operator-name Colon country SEP location-id ) |
      ( operator-name Colon country SEP location-id SP text-description )

operator-name       = let-dig-string
country             = IANA ccTLD two character country code
location-id         = let-dig-string
let-dig-string      = let-dig | ( let-dig-string let-dig )
let-dig             = Alpha | Digit
text-description    = %x00-FF | ( text-description %x00-FF )
Alpha               = %x41-5A | %x61-7A       ; A-Z, a-z
Digit               = %x30-39                 ; 0-9
SEP                 = Comma | ( Comma SP )
SP                  = %x20                    ; ASCII/ISO 646 space character
Comma               = %x2C                    ; ASCII/ISO 646 ","
Colon               = %x3A                    ; ASCII/ISO 646 ":"
```

Note that operator-name uses only ISO 646 compliant 7 bit encodings.

Operator name may be no more than 13 characters in length.

country is the IANA ccTLD two-letter country-code Top-Level-Domain of the VNP. Note that, unlike DNS, the country code is case-sensitive and use of upper case is specified.

location-id is the value used to uniquely identify the location of service within the VNP as defined in the VNP's implementation of the WBA location feed.

text-description should contain location-specific data of the originating hotspot. If a VNP is unable to provide location-specific data, a static or null value is acceptable.

For example:

- "StarHub:SG, 1A2340"

- “Telstra:AU, 1234567890”
- “TMOBILE:US, 10403 SFO San Francisco Intl Airport”

This will allow a recipient to identify the original source that generated the RADIUS message as well as the location of service.