XEP-0252: BOSH Script Syntax

Ian Paterson
mailto:ian.paterson@clientside.co.uk
xmpp:ian@zoofy.com

2008-10-31
Version 0.1

Status	Type	Short Name
Deferred	Historical	NOT_YET_ASSIGNED

This specification provides historical documentation regarding the ”alternative script syntax” first defined in Version 1.6 of XEP-0124.

Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2018 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the ”Specification”), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

NOTE WELL: This Specification is provided on an ”AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).

Contents

1 Introduction

The cross domain security restrictions of some runtime environments permit clients to access pure XML text only if it was received from a specific server (e.g., the hostname a Web client was downloaded from). Surprisingly, the same environments typically permit clients to receive and execute scripts from any server. The Security Considerations section below describes the significant risks of deploying Script Syntax.

To enable domain-restricted clients to use BOSH with any connection manager, this section proposes an *optional* alternative to the standard "BOSH Pure Syntax" defined in BOSH (XEP-0124) [1]. The "BOSH Script Syntax" defined here essentially inserts each <body/> element sent by the client into an HTTP GET header instead of into the body of a POST request. Each <body/> element sent by the connection manager is wrapped inside an ECMAScript (JavaScript) [2] string and function call. No changes to the <body/> element or to any other aspects of the BOSH protocol are needed.

If, and only if, a client is *unable* to use the Pure Syntax defined in XEP-0124, then it MAY instead request the use of the Script Syntax defined herein.

2 Script Syntax

2.1 Requesting Use of Script Syntax

A BOSH client can send a session request to a BOSH connection manager for using Script Syntax instead of Pure Syntax.

If the connection manager supports Script Syntax then it MUST send its Session Creation Response using Script Syntax, and all subsequent client requests and connection manager responses within the session MUST be sent using Script Syntax. If the connection manager does not support the "BOSH Script" syntax then it SHOULD return either an 'item-not-found' terminal binding error (in Script Syntax) or an HTTP 404 (Not Found) error in response to the client's session request.

Note: The line break in the body of the HTTP response in the following example is included only to improve readability. In practice there MUST be no line breaks.

Listing 1: Script Syntax not supported binding error

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8
Cache-Control: no-store
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 212
```

[1] XEP-0124: Bidirectional-streams Over Synchronous HTTP <https://xmpp.org/extensions/xep-0124.html>.

[2] Standard ECMA-262: ECMAScript Language Specification 3rd edition <http://www.ecma-international.org/publications/standards/Ecma-262.htm>.

```
_BOSH_("<body type='terminate' condition='item-not-found'
        xmlns='http://jabber.org/protocol/httpbind'/>")
```

Listing 2: Script Syntax not supported HTTP error

```
HTTP/1.1 404 Not Found
Content-Length: 0
```

2.2 Changes to the Request Syntax

Clients MUST make the following changes to convert their requests to Script Syntax:

1. Certain octets within the UTF-8 encoded <body/> element SHOULD be replaced according to the rules for escaping octets within URIs defined by RFC 3986 [3]. Therefore all octets except those representing 7-bit alphanumeric characters or the characters -._~!$&'()*+,;=:@/? should be substituted with a character triplet, consisting of the percent character "%" followed by the two hexadecimal digits that represent the value of the octet.

2. A '?' character and the URI-encoded <body/> element MUST be appended to the URI at which the connection manager is operating within its server.

3. The resulting URI MUST be sent to the connection manager within an HTTP GET request.

4. Include extra HTTP headers to prevent request/response caching or storage by any intermediary.

Note: All whitespace between "GET " and " HTTP/1.1" in the HTTP GET header lines in the following two examples is included only to improve readability. In practice there MUST be no whitespace.

Listing 3: Requesting a BOSH session in Script Syntax

```
GET /webclient?%3Cbody%20content='text/xml;%20charset=utf-8'%20
    hold='1'%20rid='1573741820'%20to='jabber.org'%20
    route='xmpp:jabber.org:9999'%20secure='true'%20ver='1.6'%20
    wait='60'%20xml:lang='en'%20
    xmlns='http://jabber.org/protocol/httpbind'/%3E
    HTTP/1.1
Host: httpcm.jabber.org
```

[3] RFC 3986: Uniform Resource Identifiers (URI): Generic Syntax <http://tools.ietf.org/html/rfc3986>.

```
Accept-Encoding: gzip, deflate
Cache-Control: no-store
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 0
```

Listing 4: Transmitting stanzas in Script Syntax

```
GET /webclient?%3Cbody%20rid='1249243562'%20sid='SomeSID'%20
    xmlns='http://jabber.org/protocol/httpbind'%3E%3C
    message%20to='friend@example.com'%20xmlns='jabber:client'%3E%3C
    body%3EI%20said%20%22Hi!%22%3C/body%3E%3C/message%3E%3C/body%3E
    HTTP/1.1
Host: httpcm.jabber.org
Accept-Encoding: gzip, deflate
Cache-Control: no-store
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 0
```

Although RFC 2616 does not limit the length of HTTP URIs, the runtime environment of the client might restrict the length of the URI that it can include in each GET request. [4] In these cases the client MUST reduce the content of the <body/> element accordingly and send the remaining content in subsequent HTTP GET requests wrapped in new <body/> elements (with incremented 'rid' attributes). This is possible since, unlike Pure Syntax, with Script Syntax the connection manager MUST treat the string of characters between the opening and closing <body> tags of each request as part of an octet stream instead of as a set of complete XML stanzas. The content of any one <body/> element MUST NOT be parsed in isolation from the rest of the stream.

2.3 Changes to the Response Syntax

Connection managers MUST make the following changes to convert their responses to Script Syntax:

1. Certain characters within the <body/> element MUST be replaced according to the rules for escaping characters within strings defined by ECMAScript. The necessary substitutions are summarised in the table below.

Character	Unicode Code Point Value	Escape sequence
"	U+0022	\"
Line Feed (New Line)	U+000A	\n
Carriage Return	U+000D	\r

[4] Internet Explorer versions 4.0 thru 7.0 have a maximum *path* length of 2048 characters and a maximum URL length of 2083 characters. Other popular browsers appear to have no limit.

Character	Unicode Code Point Value	Escape sequence
Line Separator	U+2028	\u2028
Paragraph Separator	U+2029	\u2029
\	U+005C	\\

Each Unicode format-control character (i.e., the characters in category "Cf" in the Unicode Character Database, e.g., LEFT-TO-RIGHT MARK or RIGHT-TO-LEFT MARK) MUST also be substituted by its Unicode escape sequence (e.g. \u200e or \u200f).
2. The following eight characters MUST be prepended to the <body/> element:

```
_BOSH_("
```

3. The following two characters MUST be appended to the <body/> element:

```
")
```

4. If the client request does not possess a 'content' attribute, then the HTTP Content-Type header of responses MUST be either "text/javascript; charset=utf-8" or "application/x-javascript; charset=utf-8".
5. Include extra HTTP headers to prevent caching or storage by any intermediary.
Note: All line breaks in the bodies of the HTTP responses in the following two examples are included only to improve readability. In practice there MUST be no line breaks.

Listing 5: Session creation response in Script Syntax

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8
Cache-Control: no-store
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 233

_BOSH_("<body␣wait='60'␣inactivity='30'␣polling='5'␣requests='2'␣hold
    ='1'
␣␣␣␣␣␣␣␣ack='1573741820'␣accept='deflate,gzip'␣maxpause='120'␣sid='
    SomeSID'
␣␣␣␣␣␣␣␣charsets='ISO_8859-1␣ISO-2022-JP'␣ver='1.6'␣from='jabber.org'
␣␣␣␣␣␣␣␣secure='true'␣xmlns='http://jabber.org/protocol/httpbind'/>")
```

Listing 6: Receiving stanzas in Script Syntax

```
HTTP/1.1 200 OK
Content-Type: text/javascript; charset=utf-8
Cache-Control: no-store
```

```
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 212

_BOSH_("<body␣rid='1249243562'␣sid='SomeSID'
␣␣␣␣␣␣␣␣xmlns='http://jabber.org/protocol/httpbind'>\n
␣␣␣␣␣␣␣␣<message␣to='contact@example.com'␣xmlns='jabber:client'>\n
␣␣␣␣␣␣␣␣<body>I␣said␣\"Hi!\"</body>\n</message>\n</body>")
```

Note: As with Pure Syntax, each <body/> element sent to the client MUST encapsulate zero or more complete XML stanzas.

3 Error Handling

Any connection manager (not only legacy connection managers) can indicate that it does not support Script Syntax by sending an HTTP 404 error code (instead of sending a bad-request error using Scrypt Syntax).

Listing 7: Script Syntax not supported error

```
HTTP/1.1 404 Not Found
Content-Length: 0
```

4 Security Consideration

To avoid the storage of private communications by third parties, when using the alternative Script Syntax connection managers MUST (and clients SHOULD) include all the appropriate HTTP/1.0 and/or HTTP/1.1 headers necessary to ensure as far as possible that no request or response will ever be cached or stored by any intermediary.
The alternative Script Syntax returns code for the client to execute. This code is typically executed immediately without any validation and with the same rights as the code of the client itself. This vulnerability could be exploited to steal passwords and private keys, or to fabricate messages sent from and received by the client, or to forward or modify priviledged information on the servers to which the client has access, or to interfere with any aspect of the client's functionality -- limited only by the extent of the runtime environment ("sandbox"), by the extent that naive users can be tricked into doing things outside that environment, and by the attacker's fertile imagination. Therefore, although the client could use Script Syntax with any connection manager on the network, in practice it MUST take care to employ it only with connection managers that the client's user trusts (as much as the server from which the client was downloaded). To prevent a-man-in-the-middle from manipulating the code clients SHOULD only use Script Syntax over encrypted connections (see above). If the client was downloaded over an encrypted connection then it MUST NOT use Script Syntax over

connections that are not encrypted.