SPECIFICATION OF WEB INGEST OF METEOROLOGICAL BULLETINS IN WIS

2013-07-10 TOYODA Eizi, JMA
2013-09-09 1st revision, TOYODA Eizi, JMA
2013-12-24 2nd revision, MIZUSHIMA Hirofumi, JMA

INTRODUCTION

This document describes a mechanism to ingest meteorological bulletins in WIS via HTTPS (HTTP over TLS)[1]. The objective is to back up the dissemination of GTS bulletins when the primary way is under interruptions.

The purpose and protocol look similar to the Web Data Ingest[2] of GTS. The difference is that the present document is intended for automated submission of files accumulating meteorological bulletins, while the Web Data Ingest is designed for human typing on text forms.

GENERAL DISCUSSIONS

TERMINOLOGY AND REASON FOR PUSH

This document discusses “push” type of protocol, in which sender of data initiates the TCP connection. A software sending data is called “client”, and the receiver called “server”. Push protocol can avoid delay due to polling, while the client has to take responsibility to detect error. The HTTP provides PUT and POST methods for push. Between the two methods, PUT is recommended for simplicity.

ONE REQUEST PER ONE FILE

One HTTP request is used to submit one file with a name. One HTTP response is returned from the server. The client must inspect the response to detect error. The format of the file may be agreed by both parties of communication, but one possibility is a so-called WMO FTP format[3].

---

[1] IETF RFC 2818: “HTTP Over TLS”. http://tools.ietf.org/html/rfc2818

[2] Section B, Attachment II-16 “Procedures for Transmitting and Collecting Meteorological Bulletins on the Internet”, the Manual on Global Telecommunication System, WMO No. 386.

[3] Section “FTP Procedures” and accompanying Figure 4.2 of the Attachment II-15, the Manual on GTS.

SECURITY

The web service must be protected by authentication using a username and a password. The basic access authentication[4] is recommended for wide support of application software.

That means that the sending WIS center has to be registered to the receiving WIS center before starting communication. The receiving WIS centre notifies:

- Username
- Password
- Endpoint URL[5]

HTTPS must be used to protect username and password. The digest access authentication is stronger than the basic authentication, but still not enough without HTTPS; the basic access authentication is considered to have acceptable strength when used with HTTPS.

The HTTPS server must maintain valid (non-expired) digital certificates. The WIS centre running the server must provide the way to verify the certificate. It is suggested using a certificate issued by widely-recognized certificate authority, to reduce the setup work for some implementation, but is not mandated because of cost implication.

REQUEST FROM CLIENT

In HTTP, the request from the client has three components, i.e., request-line, headers and request body. In Figure 1 shown is a very simple example of an HTTP request using PUT method to send an alphanumeric bulletin in a WMO FTP format (note the alphanumeric content is chosen simply for readability). Symbols for control characters[6] are SOH (┌) , ETX (┘), LF (≡), and CR (←). Note that the newline in HTTP is CR LF (←≡), while a newline in alphanumeric bulletin is CR CR LF (←←≡). It is suggested, however, the HTTP server try to understand a newline of LF only.

---

[4] IETF RFC 2617 "HTTP Authentication: Basic and Digest Access Authentication", Section 2. http://tools.ietf.org/html/rfc2617#section-2

[5] Following text uses `http://gisc.example.org/endpoint` as an example purpose only. This URL must be different in actual practice, and may not include "`endpoint`".

[6] Symbols are defined in ISO 2047:1975 Information processing – Graphical representations for the control characters of the 7-bit coded character set.

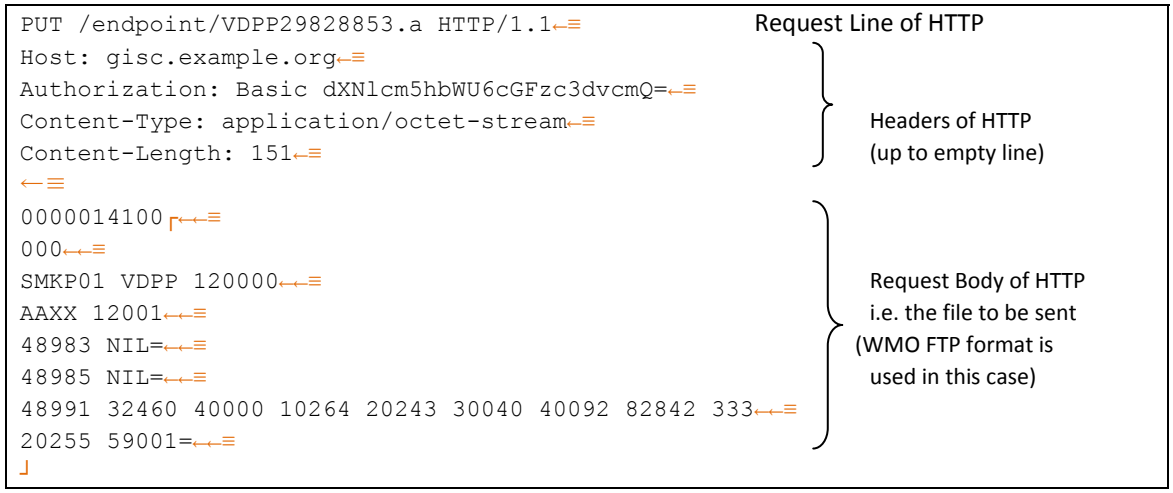

Figure 1: example of PUT request

PUT METHOD (RECOMMENDED)

REQUEST LINE

The version of HTTP must be 1.1[7] (or greater in the future). The PUT method was not defined in HTTP 1.0.

The request URI must be the endpoint URI *followed by* slash ('/') and a filename. The filename must consist of digits, letters, hyphen, plus sign, underscore, and period (that covers both WMO FTP format and WMO Filename Conventions). The server may, nevertheless, try to accept harmless deviations from above regulation of the request line:

- HTTP 1.0 is no problem
- Trailing slash can be simply ignored
- Multiple slashes (directories) under the endpoint URI can be tolerated; the rightmost path segment should be used as filename in this case
- If the filename is missing, i.e. the URI in the request line is the endpoint URL itself `(/endpoint` for above example), the server may generate filename in lieu of the client.

HEADERS

See appendix for full detail.

- **Host** is mandatory in HTTP.
- **Authorization** is needed, since authentication is mandated.
- **Content-Length** must be used; that is the size of body (i.e. file) in octets.

[7] IETF RFC 2616 "Hypertext Transfer Protocol -- HTTP/1.1". http://tools.ietf.org/html/rfc2616

- **Content-Type** is also recommended; suggested value is "`application/octet-stream`"; it may be anything else subject to the agreement, but "`application/x-www-form`", "`multipart/*`" and "`message/*`" are not allowed, because the request body has different meaning.
- **Content-Encoding**: It can be useful to use gzip compression if the server supports; that is indicated by "`Content-Encoding: gzip`" header.
- It is not recommended to use chunked transfer, partial content, content negotiation, cache control, or conditional headers.

REQUEST BODY

The request body must be the file to be submitted. The octet sequence in the request body is identical to that of the file, i.e no conversion is needed, unless **Content-Encoding** or **Transfer-Encoding** headers are used.

When WMO FTP format is transmitted in this procedure, it is suggested to use fixed zero "`000`" as *nnn* (the transmission sequence number). The purpose of *nnn* is the request for repetition[8], which is usually not possible in this procedure.

POST METHOD (INFORMATIONAL)

REQUEST LINE

The POST method is supported in versions of HTTP 1.0 and 1.1 (or greater in the future).

HEADERS

See appendix for full detail.

- **Host** is mandatory in HTTP.
- **Authorization** is needed, since authentication is mandated.
- **Content-Length** must be used; that is the size of body in octets.
- **Content-Type** is also recommended; suggested value is "`multipart/form-data`".
- **Content-Encoding**: It can be useful to use gzip compression if the server supports; that is indicated by "`Content-Encoding: gzip`" header.
- It is not recommended to use chunked transfer, partial content, content negotiation, cache control, or conditional headers.

REQUEST BODY

[8] The request for repetition is defined in section 2.5.3, part II of the GTS Manual.

When WMO FTP format is transmitted in this procedure, it is suggested to use fixed zero "000" as *nnn* (the transmission sequence number) as the same as PUT method. File should be given as a part with name="uploaded". Only one part should be given.

RESPONSE FROM SERVER

In HTTP, the response from the server has a similar structure as in the request:

- Status-Line
- Headers
- Response body (may be empty)

The status line is a three digit response code followed by text explanation. The first digit indicates the class of response: 2 for success, 4 for client error, and 5 for server are relevant in this procedure.

The server is recommended to return response codes described in the following section. The client is recommended to prepare for other response codes not described here: see description of classes (2XX, 3XX, 4XX, and 5XX).

2XX (SUCCESS)

The response code 2XX represents success.

200 OK

The server is recommended to use **204** (No Content) instead of this response code. HTTP allows **200**, but it can also be returned from a broken server that performs GET method instead of PUT.

See **204** for client's recommended reaction.

201 CREATED

The HTTP states the server must return this response code when PUT and POST succeeds in creating new resource.

HTTP tells the server should return **Location** header pointing to the resource being created. Also HTTP tells that the response body (in any media type) should describe the same URL. Those are intended for enabling GET access afterwards. In the procedure of present document, the server may refuse to respond to GET method.

The client has nothing more to do for this response. It may continue sending new requests on the same connection, or disconnect and wait for new data.

```
HTTP/1.1 201 Created←≡
Location: https://gisc.example.org/endpoint/VDPP29828853.a←≡
Date: Fri, 12 Jul 2013 10:59:42 GMT←≡
Content-Type: text/plain←≡
Content-Length: 66←≡
←≡
https://gisc.example.org/endpoint/VDPP29828853.a created←≡
```

Figure 2: example of response 201 (Created).

204 NO CONTENT

The HTTP states the server should return this response code (or **200**) when PUT and POST succeeds in updating existing resource. The server must give an empty response body in this case.

The client should treat it as failure, even though it is success in terms of HTTP. This indicates that the client has PUT the same filename more than once in a short period of time, before the server deletes the older file. Possible reasons include situations that require immediate maintenance:

- Client fails to generate unique filename
- More than one clients send the same filename
- Server is accepting HTTP but the backend processing of the file is in trouble

```
HTTP/1.1 204 No Content←≡
Date: Fri, 12 Jul 2013 10:59:42 GMT←≡
Content-Length: 0←≡
←≡
```

Figure 3: example of response 204 (No Content).

3XX REDIRECTION

Response codes starting with "3" are used to inform that the client should use the URI indicated by the response header **Location**.

4XX CLIENT ERROR

Response codes starting with "4" are used to indicate that there is some error in the request. The client may retry if it can fix the problem.

401 NOT AUTHORIZED

The server is recommended to use this response code if it detects authentication is failed. See HTTP for details of the basic authentication.

403 FORBIDDEN

404 NOT FOUND

The server is recommended to use these response codes if it detects privilege-related error.

409 CONFLICT

The server is recommended to use this response code if it has capability of computing MD5, and actually detects mismatch of MD5. In this web ingest this is not so useful because TLS can detect error in communication line.

411 LENGTH REQUIRED

The server sends this status code when it refuses to accept the request because it lacks a defined Content-Length. Clients must add a valid Content-Length header field to their requests.

412 PRECONDITION FAILED

The server sends this status code when one or more of the request-header fields evaluated were false.

415 UNSUPPORTED MEDIA TYPE

HTTP says the server should use this response code if it detects unsupported Content-Type or Content-Encoding.

5XX SERVER ERROR

Response codes starting with "5" are used to indicate that there is some error in the server. The client should not retry automatically, since human intervention is desirable.

500 INTERNAL SERVER ERROR

The server may use this response code as a "catch-all" code for unidentified serious error.

501 NOT IMPLEMENTED

The server should use this response code if the request method is not acceptable. "PUT" is recommended in this case, but the same endpoint URI may accept others.

APPENDIX: DETAILED GUIDANCE ON HTTP HEADERS IN PUT AND POST REQUEST

The **Allow** header may be ignored by the server.

The **Authorization** header must be used in authorization (see above); the server should return **401** (Unauthorized) response if the header is missing or invalid.

The **Cache-control** general-header may be ignored by the server.

HTTP/1.1 states the "**Connection**: close" header must be used for client that does not support persistent connection.

The **Content-Disposition** header is not a part of HTTP. The client must not use it to specify filename; use URL path as described above. The server should ignore this header field.

The **Content-Encoding** header must be understood by the server; HTTP/1.1 states the response code should be **415** (Unsupported Media Type) if the server does not support the content coding such as "gzip".

The **Content-Language** header may be simply ignored by the server. The GTS cannot deliver language tag with bulletin.

The **Content-Length** header must be used.

The **Content-Location** header is discouraged; the meaning is undefined in HTTP.

The **Content-MD5** header may be understood by the server to detect error in communication channel. HTTP does not define the response if MD5 test fails: this document recommends **409** (Conflict). In that case the client can simply retry the request.

The **Content-Transfer-Encoding** header is not allowed in HTTP. The server doesn't have to decode Base64 or Quoted-Printable. The client must not simply copy email message into HTTP request.

The **Content-Type** header may be ignored by the server; therefore the client must not use composite media type for PUT method (i.e. "`multipart/*`" and "`message/*`") in which a file is encapsulated in the request body. Right now there is no official or unofficial media type defined for WMO FTP format, so "`application/octet-stream`" is suggested for PUT method. For POST method, suggested value is "multipart/form-data".

The **Date** header may be ignored by the server. The AHL including YYGGgg are supposed to be a part of request body, and the GTS cannot deliver another datestamp.

The **Expect** header is discouraged.

The **Expires** header is discouraged.

The **From** header may be ignored by the server.

The **Host** header is mandatory in HTTP/1.1.

Conditional headers i.e. **If-Match**, **If-Modified-Since**, **If-None-Match**, and **If-Range** are all discouraged.

The **Last-Modified** header may be ignored by the server.

The **Pragma** header is discouraged.

The **Proxy-Authorization** header may be used if necessary.

The **Range** header is discouraged.

The **Referer** header may be ignored by the server.

The **Transfer-Encoding** header is discouraged since there should be no need to use chunked encoding. The server is however free to implement this feature.

The **Upgrade** header must not be used. HTTPS is mandated as above, so there is no need to upgrade HTTP.

The **User-Agent** header may be ignored by the server, but it is still a good practice to include it.

(END)