

Subj: Carl Kugler's suggested IIG text about using HTTP 100-Continue

Note: Issue 3.2 has been split into 3.2 and 3.9. 3.2 deals only with the client sending zero-length HTTP Post requests. So this suggested text is probably more appropriate as part of the resolution to Issue 3.9. - TH

Also see a preliminary mail message from Carl from 1/16/01 at the end of this mail. It describes how the client can force the Printer to send the nonce when using Digest Authentication. The client sends the HTTP "Expect: 100 Continue" header to force the Printer to send the nonce.

From: Carl Kugler [kugler@us.ibm.com]
Sent: Friday, February 09, 2001 17:08
To: ipp@pwg.org
Subject: IPP> IIG> B.O. Issue 3.2; Implementation Guidance for HTTP 100-Continue

In ftp://ftp.pwg.org/pub/pwg/ipp/minutes/ipp-minutes-010124.pdf , Lee Farrell wrote:

> ACTION: Pete Zehler and Tom Hastings will update the Implementer's Guide to reflect the possible concerns related to Issue 3.2 of the Bakeoff.

Tom and Pete, here is some proposed draft IIG text regarding implementing the HTTP 100-continue mechanism, for your consideration.

Use of the HTTP 100 (Continue) Status

The specific HTTP client and server requirements are laid out in section 8.2.3, "Use of the 100 (Continue) Status", in [RFC2616]. This section summarizes the HTTP requirements and provides IPP implementation guidance related to the 100-Continue mechanism.

In some cases, a request may be rejected on the basis of the HTTP header alone. (Here, the HTTP "header" includes the HTTP request-line, the HTTP header fields, and the terminating double CRLF.) This is likely to be the case when the requested resource is protected by Digest Authentication: the client needs the "nonce" value from the Printer's challenge in order to form a proper Authorization header field value. In these cases, a client may wish to avoid transmitting the HTTP request body containing the IPP request. For one thing, transmitting a large document for a request, only to have that request rejected on the basis of the HTTP header alone, would be a waste of time and network resources. For another, some clients, especially those transmitting dynamically generated content, may find it difficult, inefficient, or even impossible to tell the content generator to back up and regenerate the content from the beginning. The HTTP 100-continue mechanism provides a solution to this problem. The purpose of

the 100-continue status is to allow a client that is sending a message with a request body to determine if the Printer is willing to accept the request (based on the HTTP request header) before the client sends the request body.

Here is a summary of the rules for HTTP 100-continue:

- If a client will wait for a 100 (Continue) response before sending the request body, it MUST send an "Expect: 100-continue" header field.
- If an HTTP request contains an "Expect: 100-continue" header field, the Printer MUST either respond with 100 (Continue) status and continue to read from the input stream, or reject the request with a final HTTP status code.
- The Printer MUST NOT wait for the request body before sending the 100 (Continue) response.
- If the Printer responds with a final status code instead of 100 (Continue), it MAY close the connection (preferably, only the Printer's input side of the connection) or it MAY continue to read and discard the rest of the response. It MUST NOT perform the requested method.
- A Printer SHOULD NOT send a 100 (Continue) response if the request does not include "Expect: 100-continue".
- A Printer MUST NOT send a 100 (Continue) response to an HTTP/1.0 request.
- A Printer MAY omit a 100 (Continue) response if it has already received some of the request body for the corresponding request.
- A Printer that sends a 100 (Continue) response MUST ultimately send a final status code, once the request body is received and processed, unless it terminates the transport connection prematurely.

Some finer points:

- A client waiting for a 100 (Continue) response SHOULD NOT wait for an indefinite period before sending the request body
- A client SHOULD ignore any unexpected 100 (Continue) responses.

The basic algorithm is this:

1. The client sends an HTTP request header containing the "Expect: 100-continue" header field, but waits before transmitting the request body.
2. The Printer examines the HTTP header and decides whether or not to accept the HTTP request.
3. If the Printer accepts the HTTP request, it sends a 100 (Continue) response and continues to read from the input stream.
4. If the client receives a 100 (Continue) response, it now has a reasonable expectation that the HTTP request will succeed. The client now transmits the request body.
5. After the Printer receives and processes the request body, it sends a final HTTP status code in response.

If the Request-URI identifies a resource protected by digest authentication, the flow of events is more like this:

1. The client sends an HTTP request header containing the "Expect: 100-continue" header field, but waits before transmitting the request body.
2. The Printer examines the HTTP header and rejects the request with 401 (Unauthorized) status and a "WWW-Authenticate" header field containing at least one challenge.
3. The client sends a new HTTP request header containing an "Authorization" header field and an "Expect: 100-continue" header field.
4. If the Printer accepts the new HTTP request, it sends a 100 (Continue) response and continues to read from the input stream.
5. If the client receives a 100 (Continue) response, it now has a reasonable expectation that the HTTP request will succeed. The client now transmits the request body.
6. After the Printer receives and processes the request body, it sends a final HTTP status code in response.

Note that a Printer can reject a request at either the HTTP level or the IPP level. E.g., you could get an HTTP (401 Unauthorized) or you could get HTTP 200 (OK) with an IPP client-error-not-authenticated (0x0402). Receiving 100 (Continue) status tells a client that the Printer is willing to accept the HTTP request, but says nothing about whether or not an IPP request (in the body of the HTTP request) will be accepted. A client should use the Validate-Job IPP operation to determine whether or not an IPP Print-Job request will be accepted. Printers MUST always apply the same authorization requirements to Validate-Job as to Print-Job. I.e., if a given Print-Job request would result in a challenge, then so must the corresponding Validate-Job request.

Some Printers may authorize access by object, identified by the HTTP Request-URI, while others may authorize access by operation, identified by the IPP "operation-id" request attribute. If a client receives the HTTP 200 (OK)/IPP client-error-not-authenticated (0x0402) combination, it means that the client should look at the Printer's "uri-authentication-supported" and "uri-supported" attributes and look for a more authenticated URI.

According to the Digest Authentication standard [RFC2617], the "nonce" value in the Printer's challenge may be good for one use only (for those really paranoid about replay attacks). Therefore, a Printer may issue a challenge for each new request. A client may include an Authorization header preemptively; doing so improves server efficiency and avoids extra round trips for authentication challenges. The Printer may choose to accept the old Authorization header information, even though the nonce value included might not be fresh. Alternatively, the Printer may return a 401 HTTP response with a new nonce value, causing the client to retry the request; by specifying stale=TRUE with this response, the server tells the client to retry with the new nonce, but without prompting for a new username and password.

Some clients cannot produce the document data for a Print-Job more than one

time, making complete retries impossible. Such clients should use this algorithm to print jobs reliably:

1. The client sends an HTTP POST request header containing the "Expect: 100-continue" header field.
2. The client waits for a response before transmitting the request body.
 - a) If the client receives a 100 (Continue) response the client transmits an HTTP request body containing a Validate-Job IPP request.
 - b) If the client receives a 401 (Unauthorized) response, it sends a new HTTP POST request header containing an "Authorization" header field with a response the Printer's "WWW-Authenticate" challenge, and goes back to step 2.
4. If the client receives an HTTP 200 (OK) response containing an IPP response with one of the success status codes, the client sends an HTTP POST request header containing the "Expect: 100-continue" header field and an "Authorization" header field containing any cached credentials.
5. The client waits for a response before transmitting the request body.
 - a) If the client receives a 100 (Continue) response the client transmits an HTTP request body containing a Print-Job IPP request.
 - b) If the client receives a 401 (Unauthorized) response, it sends a new HTTP POST request header containing an "Authorization" header field with a response the Printer's "WWW-Authenticate" challenge, and goes back to step 5.

It is possible to achieve the same results without using 100-continue, but it takes more round trips.:

1. Send a Validate-Job request to provoke a challenge from the Printer.
2. If the Printer responds with HTTP 401 (Unauthorized), send another Validate-Job request containing an "Authorization" HTTP header field with a response the Printer's "WWW-Authenticate" challenge, to see if the Print-Job request will be accepted.
3. If the Printer accepts the Validate-Job, send another Validate-Job without an "Authorization" header field, to get a fresh nonce.
4. Finally, send the Print-Job request containing an "Authorization" HTTP header field with a response the Printer's "WWW-Authenticate" challenge.

Note that for this to work, the response to a Printer's "WWW-Authenticate" challenge for Validate-Job must also be valid for Print-Job.

From: Carl Kugler [kugler@us.ibm.com]
Sent: Tuesday, January 16, 2001 14:32

To: ipp@pwg.org
Subject: IPP> Bakeoff issues 3.1 and 3.2

<<<<<<<<<<<

Issue 3.1: AGREED

IPP Client failed when an unexpected HTTP "100 continue" was received. Some printers sent a "100 continue" even before the Client sent a request.

Issue 3.2: OPEN

Some IPP Clients issues a zero length HTTP Post. The Client assumed that this would force a challenge if security is enabled on the Printer. The Client would have a problem if a subsequent print operation were challenged.

>>>>>>>>>>>>

It occurs to me that these two issues are related, and that issue 3.1 contains the solution to issue 3.2.

The crux of the 3.2 problem is this: for Digest Authentication, the client wants to provoke a challenge so it can get the "nonce" it needs in order to form the authentication-info for a request. It wants to get this challenge BEFORE it sends the document data to the printer; otherwise, the request will be rejected (Unauthorized) and will have to be resent with authentication-info.

This is exactly the type of problem that the "100-Continue" mechanism is designed to solve! If a request includes an "Expect: 100-Continue" header, the Printer MUST either respond with 100 (Continue) status and continue to read from the input stream, or reject the request with a final status code. The Printer MUST NOT wait for the request body before sending the 100 (Continue) response.

Problem 3.2 is solved if a client sends an HTTP request containing the "Expect: 100-Continue" header and waits for a 100 (Continue) response before sending the request body. When a request includes the 100-Continue expectation, and security is enabled on a Printer, the Printer will respond with 401 (Unauthorized) and include a WWW-Authenticate header containing the challenge, instead of sending 100 (Continue). This response MUST be sent after the Printer processes the HTTP headers, without waiting for the request body. At this point, the client can form the appropriate WWW-Authenticate request-header, and retry the request. This time it should receive 100 (Continue), indicating it should proceed to send the request body.

So the client has successfully provoked a challenge BEFORE sending its

Print-Job request, using only standard mechanisms that already are required.

-Carl