INTERNET-DRAFT
<draft-khare-jepi-uppflow>
Expires: February 1, 1997

Joint Electronic Payments Initiative
World Wide Web Consortium
CommerceNet

August 16, 1996

SELECTING PAYMENT MECHANISMS OVER HTTP
Or, Seven Examples of UPP Over PEP (as used in JEPI)

Status of this memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Distribution of this document is unlimited. Please send comments to the JEPI working group care of Jim Miller, W3C (jmiller@w3.org), Rohit Khare (khare@w3.org), or Don Eastlake (dee@cybercash.com). This draft is also available formatted as HTML at
http://www.w3.org/pub/WWW/Payments/JEPI/draft-jepi-uppflow

NOTE: This memo does not reflect the work of any current IETF Working Groups. Discussion of this draft is intended to support the eventual release of an IETF specification of the Universal Payment Preamble (UPP) and the development of an HTTP Extension Protocol (PEP) in the HTTP WG.

1. Abstract

The Joint Electronics Payment Initiative aims to bring key industry players together to assure that multiple payment protocols can operate effectively in Web applications. The concrete goal is automatable payment selection over HTTP.

The first step towards this was Don Eastlake's development of the Universal Payment Preamble, which is also available as an internet-draft (draft-eastlake-universal-payment). The second is the development of an HTTP Extension Protocol to embed UPP in HTTP. The latter proposal is part of the chartered activities of the IETF HTTP WG (draft-ietf-http-pep-03).

This document describes how to use UPP over PEP to support payment selection between clients and merchants. It explains basic operations: requesting available payment choices, presenting multiple choices, demanding a selection, making a selection, and accepting and rejecting choices.

2. Introduction

The JEPI project is using PEP as a vehicle for negotiating over payment mechanism between a Web client and server. In order to accomplish this, JEPI has adopted the Universal Payment Preamble (UPP) proposed by Donald Eastlake as a particular protocol to be used over PEP. This document describes a set of seven fundamental operations that support payment mechanism negotiation, and shows how to use PEP and UPP to accomplish each. In addition, it contains comments intended for the implementation team for JEPI indicating the subset which are actually needed for the JEPI demonstration.

2.1 THE SEVEN FUNDAMENTAL OPERATIONS OF PAYMENT MECHANISM NEGOTIATION

1. Request payment choices. Either end (client or server) should be able to ask the other what forms of payment it supports. JEPI implementors: In the demo, only the server will generate these requests.

2. Present payment choices that it supports. Either end should be able to list the forms of payment it supports. Notice that this may not be a complete list, but rather a list of options that it "prefers" at the current moment. This list may be presented in response to a request (operation 1) or spontaneously. The latter behavior is analogous to the use of "logo stickers" on a store window or cash register. JEPI implementors: the demo will only require this operation in response to a specific request.

3. Demand payment choice. The merchant (server) may demand that the client choose a specific form of payment to be used to pay for items. JEPI implementors: In the demo, this happens when the "invoice page" is sent from the server to the client; the demand indicates what components of the page will require a response with payment choice (operation 4).

4. Make a payment choice. The cuustomer (client) can indicate the payment method to be used to make a payment. This normally indicates to the server that payment should actually begin, and the response will either be to accept (operation 5) or reject (operation 6) the chosen mechanism.

5. Accept a payment choice. The server, in response to a payment choice (operation 4), may accept the choice and initiate an actual payment operation. The payment operation itself is not part of the

JEPI project and may or may not use PEP to handle the payment.

6. Reject a payment choice. The server, in response to a payment choice (operation 4), may reject the choice and request that another choice be made. UPP specifies that a rejection can occur either because the user canceled the transaction prior to completion or because the transaction failed for other (payment-system specific) reasons. They are distinguished and can result in different client actions.

7. Do you accept payment by X? Either side can ask the other if it supports payment by a particular payment mechanism. JEPI implementors: This is not currently required for the demonstration, but it might be a useful addition for the client to ask the server this question prior to counter-offering with a payment mechanism not mentioned by the server in its list of supported mechanisms (operation 2).

3. Notation

Amongst other things, UPP provides a uniform vocabulary for naming options common to many payment systems, and a uniform syntax for each such option. It is not clear at the current time what mechanism should be used to allow independent payment system designers to name options so that they will not collide with the UPP namespace of shared options. We will use sub-bags to separate the name spaces. That is, we will assume that a bag of the form {upp {upp-parameter-name upp-parameter-value} ... } will be used to hold these parameters. A complete list of these common parameters and their syntax is available in the UPP specification.

In a complete implementation of UPP using PEP, it would be possible to specify these common parameters in the PEP-specified header fields Protocol:, Protocol-Request:, Protocol-Query:, and Protocol-Info: as well as in any payment-system specific headers. In the JEPI demonstration, however, we will not be using these parameters for the generic UPP protocol (they may be used in payment-specific protocols). In this document we will indicate where they are syntactically permitted by using the notation "upp-params." For the demonstration, these will always be omitted in the examples shown here.

For clarity, we omit all of the HTTP headers and message body with the exception of those parts directly related to the operation being demonstrated. The protocol-name URLs shown here are purely for example, and will be determined by the participants at a later date. The URLs for the various for lists will be determined by each merchant application. Because we do not expect proxy servers to participate in the payment negotiation shown during the JEPI demo, the scope parameters of all PEP headers have been omitted: they are defined to default to {scope origin} as required for the demonstration.

Similarly, the strength of PEP directives defaults to optional ({str opt}), so it is only shown otherwise.

4. Operation 1: Requesting Preferred Payment Choices

Either end (client or server) should be able to ask the other what forms of payment it supports.

CLIENT ASKS SERVER

In order for the client to ask the server what payment choices are available, the Protocol-Query: header is added to an HTTP request from the client to the server. JEPI implementors: In the demo, only the server will generate these requests.

```
GET URL
Protocol-Query: {http://www.w3.org/UPP upp-params}
```

This means "do you have UPP available at the URL specified in the HTTP request that contains this header." If any of the upp-params are specified then they further restrict the meaning of the query (i.e. if a {upp {amount {frf}}} were specified, the query would mean "do you have UPP available, for amounts denominated in French francs, at the URL specified in the HTTP request that contains this header").

In order to ask a more general question (such as "what payment choices are available for all URLs at your site") the for option must be used:

```
GET URL
Protocol-Query: {http://www.w3.org/UPP {for /*} upp-params}
```

Notice that in a for, the "URLs" ending in * are actually prefix strings. So the "/*" here means "any URL at your server that starts with '/'," which in turn means all URLs aat your server.

The response to this HTTP message will fall into one of two categories:

* No Protocol-Info: {http://www.w3.org/UPP A} header line. This indicates that the server does not support all of PEP. [It is also possible for a server to support PEP, but not UPP, in which case it would send Protocol-Info: {http://www.w3.org/UPP {str ref}}]
* A header line of the form Protocol-Info: {http://www.w3.org/UPP A} is included in the headers. This indicates that the server supports PEP, and the response is in the form described below under Operation 2. The header can also use a for list to hint where on the server payments will be discussed.

A proper implementation of PEP requires that the protocol module associated with the specified protocol will be invoked when a Protocol-Query: line is encountered specifying that protocol. A proper

implementation of the UPP protocol module will supply one of the responses indicated under Operation 2 (Present Payment Choices), indicating the payment options that the server wishes to advertise.

SERVER ASKS CLIENT

In order for the server to ask the client what payment choices are available, a similar mechanism is used. In this case, however, the server should use the {for } to indicate the parts of its URL space where payment might be discussed:

```
200 OK
Protocol-Query: {http://www.w3.org/UPP {for /PaymentPages/*}
                                       upp-params}
```

Technically, this is a way for the server to ask the client to reveal payments choices a user will consider for URLs that begin with /PaymentPages/. The client will reply (at least) whether the protocol can be used for the resource of the response, and (optionally) whether it might be used elsewhere (the range the server specified, anywhere on that server, etc).

A proper implementation of PEP requires that the protocol module associated with the specified protocol will be invoked when a Protocol-Query line is encountered specifying that protocol. A proper implementation of the UPP protocol module will supply one of the responses indicated under Operation 2 (Present Payment Choices), indicating the payment options that the client wishes to advertise.

Then, the next time the client accesses any resource in the for list from the query, it will include its answer(s) to the query.

5. Operation 2: Present Payment Choices

Either end should be able to list the forms of payment it supports. Notice that this may not be a complete list, but rather a list of options that it "prefers" at the current moment. This list may be presented in response to a request (operation 1) or spontaneously. The latter behavior is analogous to the use of "logo stickers" on a store window or cash register.

JEPI implementors: the demo will only require this operation in response to a specific request.

This operation is performed by adding one or more Protocol-Info: headers to the HTTP packet. If the list is being presented in response to a request (operation 1), PEP requires that it include a header in the following form:

```
200 OK -or- GET ...
```

```
Protocol-Info: {http://www.w3.org/UPP [for] [{str strength}]
                                      upp-params}
```

where the for should be the same as the for clause in the request (or omitted if it wasn't in the request); and the strength (if present) must be ref, req, or opt. The strength can be opt (or omitted) in any case; it may be ref only if payment won't be permitted at any of the URLs specified by the for clause; it may be req only if payment is required at all of the URLs specified by the for clause.

In addition, there should be Protocol-Info: headers for each of the payment systems that are to be presented to the other end.. These will have the form:

```
200 OK -or- GET ...
Protocol-Info: {http://...payment-system... [for] [{str strength}]
                                            payment-params}
```

where payment-protocol is the URL for the specific payment protocol, the for and strength are as discussed above, and the payment-params are additional parameters (including the UPP parameters) that are specific to the payment system.

For example, if a client receives the request:

```
200 OK
Protocol-Query: {http://www.w3.org/UPP {for /PaymentPages/*}
                                       upp-params}
```

and wishes to indicate that it can pay using VISA over SET and via CyberCash coins it might reply as follows (details of the payment-specific lines are not finalized yet):

```
HEAD ...
Protocol-Info: {http://www.w3.org/UPP {for /PaymentPages/*}}
   {http://www.SET.org/PEPSpec
      {params {upp {instrument-brand VISA}}}
      {for /PaymentPages/*}}
   {http://www.CyberCash.com/PEPSpec
      {params {upp {instrument-type ECASH}}}
      {for /PaymentPages/*}}
```

6. Operation 3: Demand Payment Choice

The merchant (server) may demand that the client choose a specific form of payment to be used to pay for items.

JEPI implementors: In the demo, this happens when the "invoice page" is sent from the server to the client; the demand indicates what components of the page will require a response with payment choice (operation 4). In the demonstration, this same invoice page will

carry both the operation 2 and operation 3 headers together: the server will announce some of its payment options at the time it issues the invoice and requires that payment be accompanied by a particular payment choice.

As part of a standard server (successful) reply, it may deliver a page that includes references that will require payment (i.e. a "Pay Button" or "Pay URL"). These should be ideentified in the header of the response packet by asking the client to respond by initiating a UPP payment protocol sequence:

```
200 OK
Protocol-request: {http://www.w3.org/UPP {str req} {for /PayButton}}
```

Technically, this means that the server asks the client to use the UPP protocol (operation 4) whenever it asks for retrieval of the exact URL /PayButton from this same server. The {str req} is a hint to the client that if it doesn't use the protocol, the request for that URL will be refused. Thus, the client is not absolutely required to remember that it should use UPP with the specified URL - but a network roundtrip will be avoided if it does so.

7. Operation 4: Make a Payment Choice

The customer (client) can indicate the payment method to be used to make a payment. This normally indicates to the server that payment should actually begin, and the response will either be to accept (operation 5) or reject (operation 6) the chosen mechanism.

In practice, this will only happen when a client replies to an operation 3 request for payment method. It must then respond with two headers: one indicating that it is responding to a request to use the UPP protocol by choosing a compatible payment protocol, and the compatible protocol header itself. For example, if the payment choice is to use VISA over SET, then we might expect a response as follows:

```
GET ...
Protocol: {http://www.w3.org/UPP {via http://www.SET.org/PEPSpec}}
  {http://www.SET.org/PEPSpec
     {str req}
     {params {upp {instrument-type CREDIT} {instrument-brand VISA}}
             other-SET-params}}
```

The expected response is either an operation 5 (server accepts the choice of SET and VISA) or operation 6 (server refuses the choice).

It is expected that somewhere between receiving the operation 3 and issuing the operation 4 the client application will have to decide on the payment mechanism. Neither PEP nor UPP specifies how this happens. For the JEPI demonstration, it is assumed that the browser will intercept the request to access any specified payment URLs (from the

for list of the required challenge) and will engage in a dialog with the user if necessary to produce the desired choice. This implies that what merchants might typically describe as the "Pay" button becomes the "Choose a Payment Mechanism and Pay" button.

8. Operation 5: Accept a Payment Choice

> The server, in response to a payment choice (operation 4), may accept the choice and initiate an actual payment operation. The payment operation itself is not part of the JEPI project and may or may not use PEP to handle the payment.

At this point, operation 4 has provided enough information to the server that it is willing to kick off the actual payment system. JEPI, PEP, and UPP provide no information on precisely how to do this, but there is one additional PEP/UPP header which can be optionally sent back to the client. If a normal MIME-based helper application is available to do the payment on the client side, then there is no need for the following header. On the other hand, a better user interface can often be produced if a helper application can be run while the client (browser) waits for the application to complete. To support this, UPP adds one final message from the server to the client. It provides the URLs that should be shown in each of three cases:

* the payment is successfully completed
* the payment is canceled because of user intervention
* the payment is unable to complete because the computers are unable to finish the transaction (network outage, over credit limit, etc.)

The header is as follows:

200 OK

```
Protocol: {http://www.w3.org/UPP
   {params {upp {abort abort-URL}
                {cancel cancel-URL}
                {success success-URL}}}
```

9. Operation 6: Reject a Payment Choice

> The server, in response to a payment choice (operation 4), may reject the choice and request that another choice be made. UPP specifies that a rejection can occur either because the user canceled the transaction prior to completion or because the transaction failed for other (payment-system specific) reasons. They are distinguished and can result in different client actions.

If a client proposes a payment system that is not acceptable to the server, the server responds with a 400- or 500-class PEP error message. The body of the message should explain what went wrong as well as possible, including any explanation that the requested payment

system may be able to supply. It should probably include a button to go back to the invoice page, if possible, but the browser's BACK button will work, too. The server should include one additional header on this message to reduce the chance that the same payment system will be tried a second time:

420 Client PEP Error -or- 520 Server PEP Error
Protocol-Info: {http://...payment-system...
{str ref} payment-params}
Followed by an operation 3 header

where payment-system is the payment protocol that is being rejected, payment-params are the parameters of the payment system which caused the problem, and pay-URL is the URL of the item just requested (i.e. the one that initiates the payment protocol on the server side).

The error code is distinguished mainly by whether the server has the protocol and doesn't accept it and the client should know better (422 Protocol Extension Refused) or if the server does not have it (521 Protocol Extension Not Implemented) or cannot get it to work (520 Protocol Extension Error or 522 Protocol Extension Parameters Not Acceptable). Other PEP error codes may be more specifically applicable for particular payment systems.

10. Operation 7: Do you accept payment by X?

Either side can ask the other if it supports payment by a particular payment mechanism.

JEPI implementors: This is not currently required for the demonstration, but it might be a useful addition for the client to ask the server this question prior to counter-offering with a payment mechanism not mentioned by the server in its list of supported mechanisms (operation 2).

The PEP header Protocol-Query can be used by either party at any time to ask this question. As with operation 1, there is a technical meaning for the query that requires the other end to respond with a Protocol-Info response that is specific to the particular URL being queried, and the {for A} construct can be used to generalize the query.

Also as with operation 1 and 2, a proper implementation of a payment system module for use with UPP should provide additional information about where and with which parameters the payment system will operate when it is possible to do so. That is, a request for "do you support SET for VISA at URL /MerchantHomePage" must be answered "no" (unless payment happens on the home page), but a more thorough response will volunteer the information that such a payment is permitted elsewhere at the site.

200 OK -or- GET ...
Protocol-Query: {payment-system [payment-system-params]}

where payment-system is the URL of the payment system protocol, and payment-system-params are parameters specific to that protocol (including common UPP parameters).

11. Security Considerations

None of these message headers have security protection. They should be trusted only if received through a trusted medium (private channel, etc). In addition, UPP makes no security claims about the contents of the headers; ALL payment-related data should be recapitulated within the particular (presumably cryptographically secure) payment protocol.

In short, this protocol only addresses payment selection in the clear. Security of the overall payments process lies in other components.

12. Authors' Addresses

Donald E. Eastlake 3rd
CyberCash, Inc.
318 Acton Street
Carlisle, MA 01741 USA
Tel: +1 (508) 287 4877 (+1 703 620 4200 main office, Reston, Virginia, USA)
Fax: +1 (508) 371 7148
Email: dee@cybercash.com

Rohit Khare
Technical Staff, W3 Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, U.S.A.
Tel: +1 (617) 253 5884
Fax: +1 (617) 258 5999
Email: khare@w3.org

Jim Miller
Technology & Society Area Leader, W3 Consortium
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139, U.S.A.
Tel: +1 (617) 253 3194
Fax: +1 (617) 258 5999
Email: jmiller@w3.org