Intent to Pay

Robin Berjon | http://berjon.com/ | @robinberjon

Abstract

Payment is a broad and innovative domain and in integrating it more closely with the Web we need to ensure that we do not impede its creative and potentially disruptive nature by enshrining specific payment solutions as they exist today. This paper outlines an architectural approach that enables the integration of powerful payment systems in today's Web without preventing radical innovation in future.

The position outlined in this paper is solely my own and does not necessarily reflect that of the W3C as an organisation.

Payment as an Intentional Architecture

It would be very easy to design an API for payment that, while in appearance open, would in fact constrain its domain of application to highly specific architectures. For instance, an API could be designed that would constrain itself to interacting with a credit card chip. While the API itself would be open, it would be nevertheless be tied to a specific, established means of payment and would largely restrict innovation to the limited circle of companies that can produce such systems, thereby foregoing the invention of novel means of payment.

An intent-based architecture is one in which the services client expresses its needs using nothing more than a verb, a subject for the verb, and a minimal amount of ancillary information. It allows for maximal expressivity while decoupling invocation from implementation.

HTTP is a classic example of an intent-based architecture. For example, the client will indicate that it wishes to "GET" a specific path, and will provide a small amount of additional information encoded in headers. What happens at the other end is immaterial so long as a response is provided. The path could match a document stored on disk, or perhaps in a database, could be synthesised on the fly, or could be read in real-time off a measuring instrument.

The details do not matter, and thanks to this very generic — yet trivially interoperable — approach the Web is extremely rich in the variety of its resources despite exposing a radically simple API to all of them.

While relatively common in protocol design, intent-based architectures have only recently become more mainstream in programming APIs (most notably through their extensive use on the Android platform). Web Payments provide an excellent example of an area in which an intent approach would shine.

Describing payment operations through intents leads to a natural lifecycle:

1. The end user activates a control, typically clicks on a "Pay" button in a site's user interface.
2. The site's code calls the user agent's payment intent API using a verb ("`PAY`"), an identifier of what is being paid for, and ancillary data such as the amount and the user's credentials.
3. The user agent presents the end user with a prompt offering to choose the payment method. The site has no knowledge of this, the decision takes place between the user and her agent. If the user agent has been configured to systematically use a given method, this step can be automatically skipped.
4. The end user is then presented with the payment system that they use. They interact with it in whichever way is required to complete the operation. This could be through exposing a traditional credit card interface, through the platform's ability to carry out payment on its own, or through less conventional methods such as carrying out a Mechanical Turk operation or filling out a survey.
5. Upon completion, the service is notified of the user's successful payment and returns.

In addition to opening the door to increased innovation, this also addresses the "Nascar" problem whereby upon payment users are presented with an overly long list of payment options, only a small subset of which they can use, and which may not include their preferred method because it is new relative to the site or considered in the minority (e.g. Dogecoin).

Note that a complete payment system would require more than just the one "`PAY`" verb (e.g. "`SUBSCRIBE`"), but capturing the full list represents a manageable task.

Intent Registration

Payments present a specific difficulty when compared with other intent-based approaches that have been explored in Web technology previously: trust in the payment intent service.

For example, when using intents to edit a picture on the Web, the calling site does not care in the least that the service site that is carrying out the actual editing is really trustworthy. So long as it returns a picture all is good. On the other hand, the site performing the sale would need more than just an acknowledgement that the sale has taken place: it needs to know that the acknowledgement matches genuine funds being (eventually) transferred.

This points to the requirement for a way of establishing trust. There are multiple potential approaches to this issue, the choice of which will have to be debated in light of a far more in-depth technical analysis. An example of an intent-based API that requires the establishment of a measure of trust and operates on the Web today is Mozilla's Persona (https://login.persona.org/) system which provides a partially decentralised identity system. It would be desirable for a payment system to be even more decentralised so as to avoid privacy breaches through personal payment tracking, but this is meant solely as an example.

Conclusion

There exists a simple path to enabling payments on the Web by binding them to a specific technology. This may be appealing but it would lead to locking the Web into an excessively small list of existing players.

A more open system, aligned with the Web's own architecture of openness and competitive level playing field, is possible and can be deployed, in everyone's interest. It requires addressing a number of technical issues, but they are within reach.