W3C Web Payments and EMV 3-D Secure

Goals for 3DS2 in W3C

EMV 3DS client functionality within W3C Payment API

- Standardized integration of EMV 3-D Secure (3DS) functions for merchants and processors that use Payment Request APIs
  - Integrate 3DS client functionality into the Payment Reuqest API flows as an option for merchants
- Simplify 3DS integrations for merchants and processors that leverage Payment Request API
- Avoid a disjointed user experience for merchant checkout when using Payment Request API in combination with 3DS
- Provide better payments experience for all stakeholders
  - Consumers, Merchants, PSPs, Acquirers, Issuers
  - Reduced fraud, better user experience, and higher approval rates

Some Known Caveats

- Merchants must enroll in 3DS through their acquirer or processor (or both)
  - This means that a merchants or processors must determine if they should use 3DS2or not, not the consumer themselves
- Merchants have specific data to share in the 3DS Authentication Request and therefore should integrate directly to their Merchant Integrator/3DS Server
  - Complete via PSP, Acquirer, MI provider, etc.
- The payment handler should provide the 3DS client functions as designed within the 3DS2 spec
  - First request to provide data for AReq (frictionless flow) and to execute 3DS Method
  - Second request to provide challenge UI (if needed for challenge flow)

Payment Apps vs. Embedded Payment

Embedded Payment

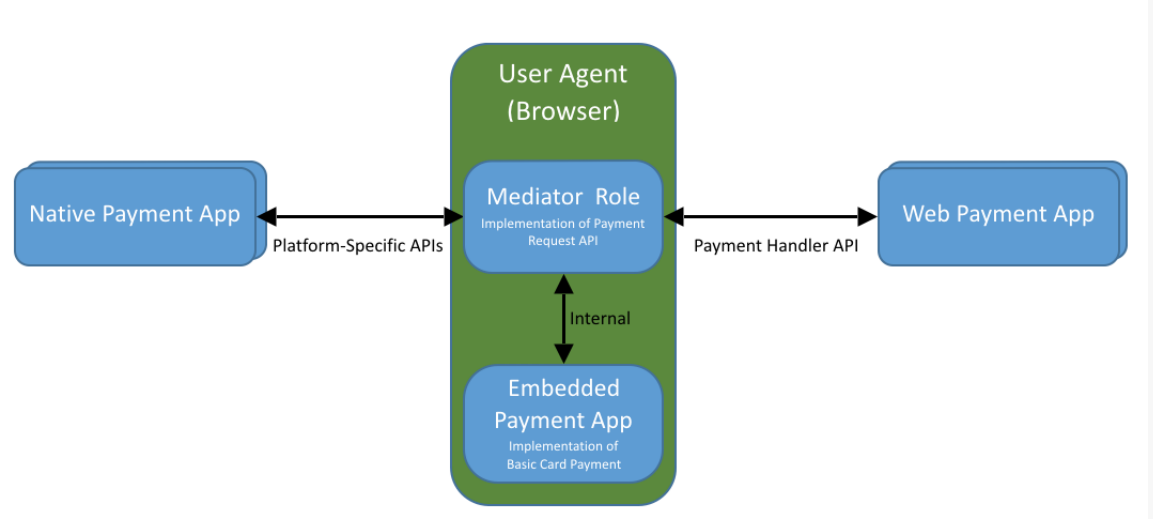
- Merchants should be able to request 3DS through Payment Request API
  - This would require a new optional 3DS2 data set to be supported

Payment Apps

Payment Apps may handle 3DS within their own flows, but require data to be passed

- Payment Request API would need to support new optional fields within payment app options from merchant

- 3DS2 Challenge UX handled as part of the Payment App flows



Embedded Payment – 3DS Browser Based Flows

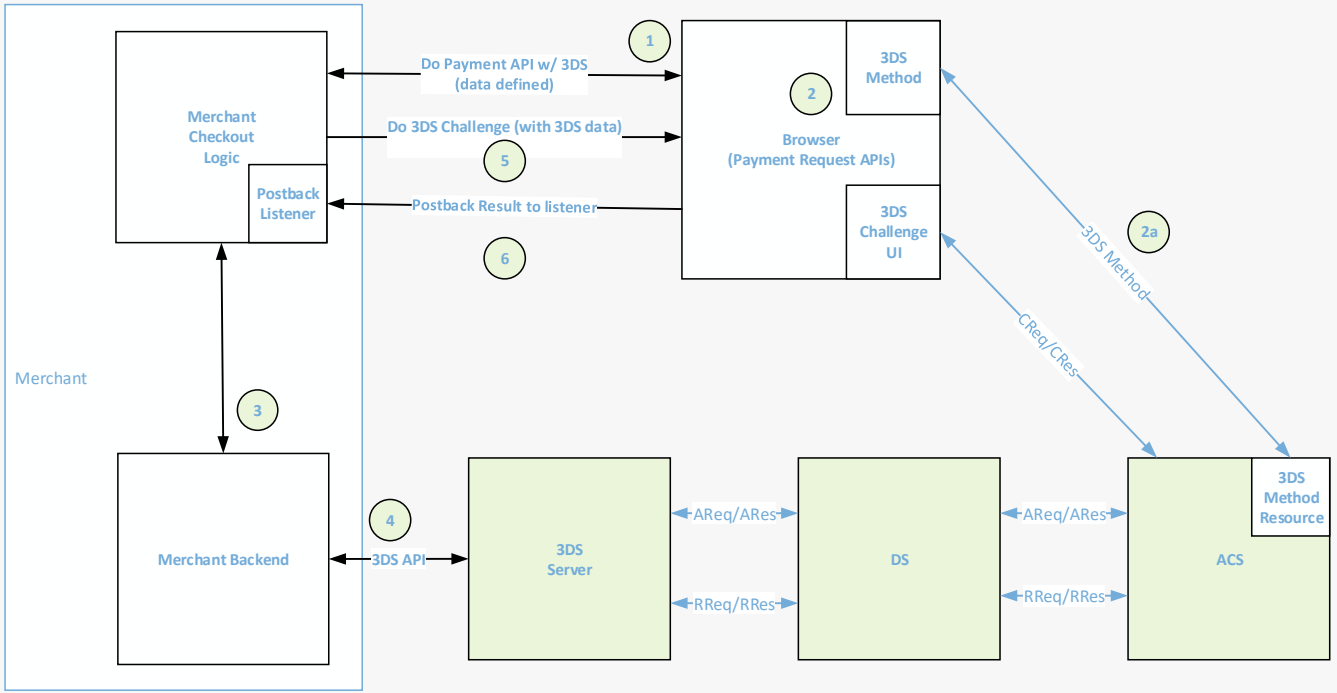
1) Merchant calls to Payment API, with 3DS2 flag set
   - Includes URL and TXN ID for 3DS Method

2) User selects stored card payment and address through payment handler
   2a) Callout to 3DS Method
3) Data returned to Merchant, and passed to back end systems

4) Make 3DS call to 3DS Server (merchant integrated with 3DS Server, PSP, etc.), 3DS Server returns data needed for **Auth *End If frictionless***

5) If challenge required, then call to Challenge API to trigger 3DS challenge UI and flows

6) Post results to postback URL and get results from backend



Demos

Embedded Payment - Frictionless Experience
https://vimeo.com/264986694/bf996bbc41

Embedded Payment with 3DS Issuer Challenge
https://vimeo.com/264983767/0aef50759d

Payment App – 3DS Browser Based Flows

1) Merchant calls to Payment API, requesting 3DS data with accepted payment methods

2) User selects a payment app

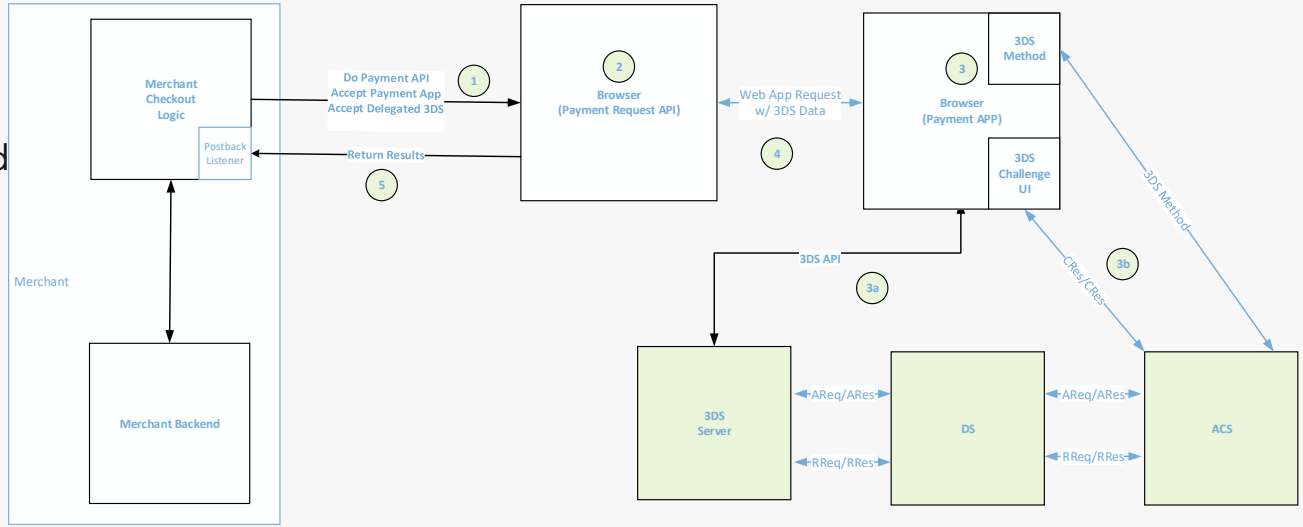
3) If delegated, 3DS passed and may be executed, returning data for authorization (AV, status, ECI, txn ID, etc.)

a.) Execute Frictionless messaging

b.) Execute Challenge (as required)

4) Return data

5) Post results to URL



Demos

Payment App - Frictionless Experience
https://vimeo.com/264642045/00f1963fa3

Payment App - 3DS Issuer Challenge
https://vimeo.com/264654981/0e0484b390

New Requirements for W3C APIs

These would need to be added to specifications to accommodate optional 3DS2 data sets and callouts

- 3DS options needed in Payment Request API for merchant request
  - Must include data for 3DS Method
    - ACS 3DS Method URL, Transaction ID
- New method for calling ACS for Challenge and handling UI (frame) and domain control
  - 3DS2 message handler for call to ACS via CReq/CRes messaging
- Post back method for the browser to communicate to merchant (TBD)
- Consider allowing payment request to remain open to allow merchant interaction with UI before completing request

Alternate Approach (for technical consideration)

Native app based flows within browser environments

- Mimic Native App Based 3DS2 challenge flows within browser environment
  - Standardized data set generated by browser for risk based authentication
    - Could remove requirement for an additional callout to ACS
  - Key generation via JOSE standards
    - Used to protect payload within 3DS challenge messaging
    - May be able to provide new methods within browser
  - Challenge data exchange with ACS via JSON data, not HTML in IFRAME
    - UI rendered within browser, leveraging issuer ACS data over 3DS messaging
  - Could provide a more cohesive experience within Payment Request API UX

Note: This would require changes to 3DS specification within EMVCo

Consistent authentication message structure and data across apps and browsers

3DS Server Initiated

Non-Payment Authentication

App Client

Payment Authentication

App Client

3DS Server Initiated

Non-Payment Authentication

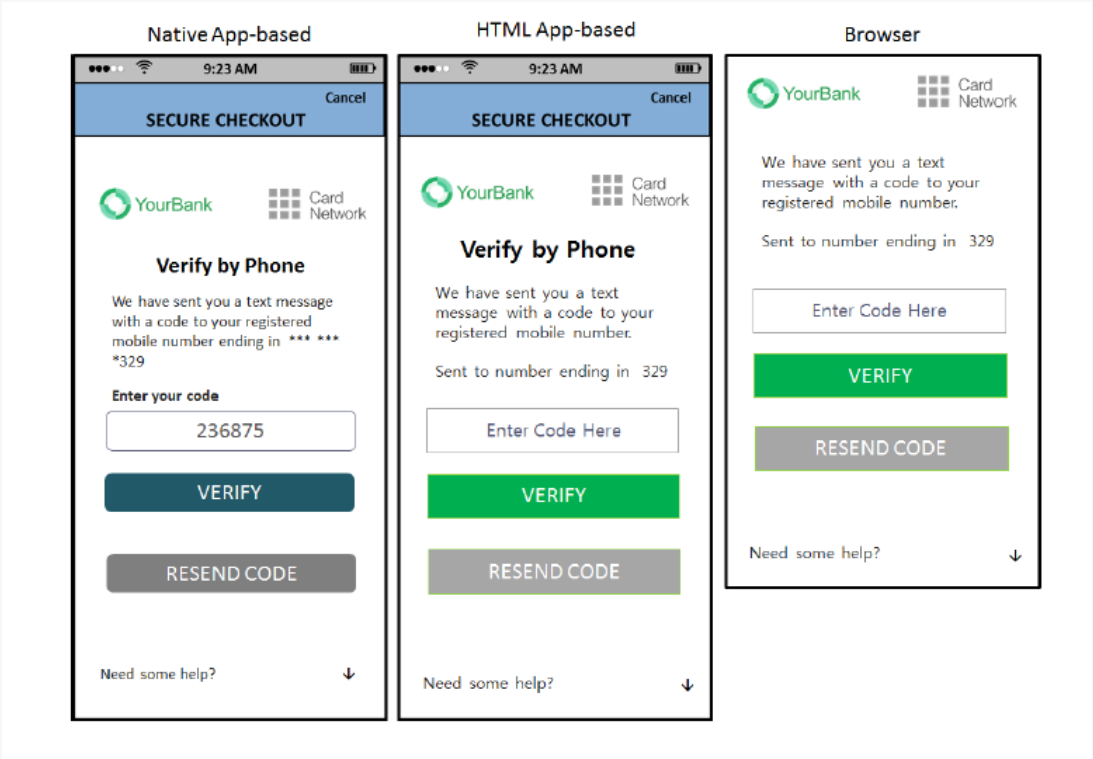
Browser Client

Payment Authentication

Browser Client

EMV 3DS Challenge User Interfaces

- Consistent look and feel across:
  - ✓ Device channels
  - ✓ Payment Systems
  - ✓ Authentication Methods
- Specified UI types allow consistency yet still maintain flexibility
- Provides a channel for cardholder to issuer communication within the merchant payment flow
- Allows an issuer to iterate between UIs to complete a cardholder authentication
  - For example, allow a user to select a passcode delivery method, then display the data entry UI to complete the step-up



Data Driven UI Elements

- The current 3DS2 app based flows leverage JSON data passed to a native app SDK in the mobile environment
- Browsers may be able to leverage the same approach to render the 3DS Challenge UI in a consistent manner