EMV® 3-D Secure Specifications
Frequently Asked Questions (FAQ)

This document provides additional information about the EMV® 3-D Secure ("3DS") Specifications. The information is current as of July 2024.

New and updated responses are highlighted and can also be searched for using keywords (NEW) or (UPDATED).

Unless stated otherwise, the answers pertain to all active versions of the *EMV 3-D Secure—Protocol and Core Functions Specification*. For answers only applicable to specific versions, the version number is explicitly stated in the question or in the answer.

Other 3DS-related FAQs are available on the EMVCo website.

1. What documentation has EMVCo published in support of EMV 3DS? (UPDATED)

EMVCo has published the following final specifications and related educational materials to the industry in support of EMV 3DS:

- *EMV 3-D Secure—Protocol and Core Functions Specification* (hereinafter also referred to as the *Core Specification*)
- *EMV 3-D Secure—SDK Device Information*
- *EMV 3-D Secure—SDK Specification*
- *EMV 3-D Secure—SDK Technical Guide*
- *EMV 3-D Secure—Split-SDK Specification*
- *EMV 3-D Secure White Paper*
- *EMV 3-D Secure Approval—Administrative Process*
- *EMV 3-D Secure—JSON Message Samples*
- *EMV 3-D Secure—App-based Cryptographic Worked Samples*
- *EMV 3-D Secure Bridging Message Extension*
- *EMV 3-D Secure Device Acknowledgement Message Extension*
- *EMV 3-D Secure Payment Token Message Extension*
- *EMV 3-D Secure Travel Industry Message Extension*
- *EMV 3-D Secure Attribute Verification Message Extension*

Additionally, EMVCo has published the following 3DS-related bulletins:

- *3DSA 01 EMV 3-D Secure Approval Fee*

- *Specification Bulletin No. 204 EMV 3-D Secure Updates, Clarifications & Errata for version 2.1.0 (SB 204)*
- *Specification Bulletin No. 205 EMV 3-D Secure SDK and Device Information Updates, Clarifications & Errata for version 2.1.0 (SB 205)*
- *Specification Bulletin No. 207 EMV 3-D Secure Key Features for version 2.2.0 (SB 207)*
- *Specification Bulletin No. 211 EMV 3-D Secure SDK Features for version 2.2.0 (SB 211)*
- *Specification Bulletin No. 213 EMV 3-D Secure Device Information Key Features for version 2.2.0 (SB 213)*
- *Specification Bulletin No. 214 EMV 3-D Secure Updates, Clarifications & Errata for version 2.2.0 (SB 214)*
- *Specification Bulletin No. 222 EMV 3-D Secure SDK—Device Information Data Version 1.3 (SB 222)*
- *Specification Bulletin No. 225 EMV 3-D Secure SDK—Device Information Data Version 1.5 Updates, clarifications and errata (SB 225)*
- *Specification Bulletin No. 255 EMV 3-D Secure Specification Version Configuration (SB 255)*
- *Specification Bulletin No. 256 EMV 3-D Secure Protocol and Core Functions Specification for version 2.3.1.0 (SB 256)*
- *Specification Bulletin No. 271 EMV 3-D Secure Split-SDK Specification for version 2.3.1.0 (SB 271)*
- *Specification Bulletin No. 275 EMV 3-D Secure SDK Specification for version 2.3.1.0 (SB 275)*
- *Specification Bulletin No. 279 EMV 3-D Secure Protocol and Core Functions Specification for version 2.2.0–2.3.1.1 (SB 279)*
- *Specification Bulletin No. 280 EMV 3-D Secure SDK Specification for version 2.2.0–2.3.1.1 (SB 280)*
- *Specification Bulletin No. 285 EMV 3-D Secure SDK Device Information for version 1.6 (SB 285)*
- *Specification Bulletin No. 294 EMV® 3-D Secure Protocol and Core Functions Specification for version 2.3.1.1 (SB 294)*
- *Specification Bulletin No. 295 EMV 3-D Secure Bridging Message Extension v2.0 (SB 295)*
- *Specification Bulletin No. 296 EMV 3-D Secure SDK Specification for version 2.3.1.1 (SB 296)*

All 3DS documentation can be accessed, viewed, and downloaded from the EMVCo website.

2. **How will DS public keys be shared with 3DS vendors?**

   There are no processes defined in the *Core Specification* with regard to sharing public keys. Each 3DS vendor will need to work with their appropriate DSs and determine how those public keys will be shared prior to implementation.

3. **How should a 3DS Requestor use the Address Match Indicator?**

   The Address Match Indicator allows the 3DS Requestor to indicate to the ACS whether the Cardholder's billing and shipping address are the same.

   3DS Requestors can use the Address Match Indicator to identify that the Cardholder selected a checkbox indicating that the shipping address is the same as the billing address. This could be helpful in regions with privacy mandates that prohibit providing billing and shipping address details.

   3DS Requestors should still provide billing and shipping address information (assuming no privacy mandates exist in the region), even when the Address Match Indicator has been provided.

   Some 3DS Requestors always provide this indicator as part of their checkout process, even if it may not be meaningful – for example, in the case of digital goods for which the shipping address is irrelevant.

4. **Can a DS issue both Elliptic Curve Cryptography (ECC) keys and Rivest–Shamir–Adleman (RSA) keys to an SDK for Device Information encryption?**

   A DS can issue either an ECC or RSA key to an SDK. The type of key issued will be determined by the DS program rules. SDKs are required to support both key types per the specification requirements.

5. **To what extent does an EMV 3DS user interface (UI) need to follow the UI requirements shown in Chapter 4 of the *Core Specification*?**

   It is EMVCo's intention that the 3DS UI be globally consistent across all 3DS transactions. This consistent appearance is one of the benefits of the new protocol and contributes to an enhanced user experience. Therefore, the UI templates (the layout, the position of the logos and different text elements, etc.) must be implemented as shown in the *Core Specification* per the requirements.

   The UI templates contained in Chapter 4 are an output of EMVCo's usability studies conducted to understand what constitutes a 3DS UI.

   UI or requirements template questions can be addressed to EMVCo or to the Payment Systems.

6. **Can you provide additional information about supporting the fallback method mentioned in Step 10 of the Browser-based requirements?**

   This is meant to indicate that the implementation must support an alternative approach/solution for environments that do not support JavaScript. This fallback method could be some sort of HTTP POST that may require consumer participation, such as a click of a button. For example, a consumer may be provided a button to enable a submit if JavaScript is not supported.

7. **How do 3DS Servers verify authenticity as defined in Req 7 of the *Core Specification*?**

   The requirement is on the 3DS Server to ensure that the 3DS SDK is authentic. How that is done – and who does what – depends on the integration model chosen by the 3DS Integrator for the components in the 3DS Requestor Environment (as outlined in Section 2.1.1 of the *Core Specification*), so the actual method to verify the authenticity is outside the scope of the *Core Specification*.

   In one possible model, a 3DS Server could outsource the verification of the authenticity of the 3DS SDK to the 3DS Requestor, assuming that the 3DS Server has a way to trust the 3DS Requestor as well as the relationship between the 3DS SDK and the 3DS Requestor. Other models are equally possible.

8. **In Section 5.9 (Message Error Handling) of the *Core Specification*, the sentence "If a specific transaction can be identified, …" is used in numerous places. Which data elements are used to identify a transaction?**

   It depends on the specific error situation and the specific implementation. In some situations, one or more of the identification data elements (SDK Transaction ID, 3DS Server Transaction ID, DS Transaction ID, or ACS Transaction ID) may be recoverable and could be used to identify a specific transaction.

   Additionally, an implementation-specific "session ID" created when the connection between two 3DS components is established can be used to identify the transaction. This "session ID" could be used even in situations where the Transaction IDs from the 3DS messages are not recoverable.

   The *Core Specification* does not mandate a specific method of identifying a transaction – that will be up to each implementation.

9. **What is Base64url-encoding as used in the *Core Specification*?**

   RFC 7515 (JWS) is the reference for Base64url-encoding. This means no padding: all trailing '=' characters omitted, and no line breaks, whitespace or other additional characters included.

   RFC 7515 refers to RFC 4648, and also specifies options in this RFC for Base64url encoding in the note below (extract from RFC 7515, Chapter 2 Terminology).

> "Base64 encoding using the URL- and filename-safe character set defined in Section 5 of RFC 4648 [RFC4648], with all trailing '=' characters omitted (as permitted by Section 3.2) and without the inclusion of any line breaks, whitespace, or other additional characters. Note that the base64url encoding of the empty octet sequence is the empty string. (See [RFC 7515] Appendix C for notes on implementing base64url encoding without padding.)"

Certain merchants' implementations of the Base64url-encoding of the CReq message for the Browser-based transaction do not fulfil the *Core Specification* requirement (padding characters present '='). Therefore, it is strongly recommended that ACSs relax their Base64url check on the incoming CReq message during a Browser-based transaction and not reject an incorrectly formed Base64url CReq message in anticipation of a *Core Specification* update.

10. How do the EMV 3DS specifications define the term "Conditional"?

The EMV 3DS specifications use the term "Conditional" for three separate scenarios:

- Scenario 1: when the value present is based on conditions of other data elements within the transaction or conditions based on data elements across messages. For example, the ACS Challenge Mandated Indicator must be present in the ARes message if Transaction Status = C.
- Scenario 2: when the value present is based on the DS program rules. For example, the 3DS Server Operator ID is present in the AReq message due to a DS program rule stating that the 3DS Server Operator ID must be present for all transactions sent to that DS.
- Scenario 3: when the value is present because the local market allows such values to be sent between 3DS components. For example, the Cardholder Billing Address City must be present in the AReq message unless regional mandates restrict sending such information.

11. How should a 3DS Server route a co-badged card?

The *Core Specification* allows Issuers to enrol their co-badged account ranges onto the DS and for the 3DS Server to obtain those account ranges for Authentication routing. If a 3DS Server obtains multiple DS URLs for the same account ranges, the 3DS Server can choose how to route that transaction for Authentication.

The *Core Specification* does not specify how a 3DS Server routes co-badged cards. The choice of DS to which the transaction is routed will be implementation-specific, based on the local market conditions.

12. Does the Data Version Number need to align specifically with an EMV 3DS Protocol Version Number?

No. EMVCo has recently changed the version format of the SDK Device Information to indicate the most recent Data Version Number. The previous SDK Device Information version format indicated the EMV 3DS specification version number. This update reaffirms that the most recent Data Version Number is backwards-compatible with previous EMV 3DS specification version(s). Please see *SB 255* for reference.

ACS providers are urged to support all announced SDK Data Version Numbers to help avoid step-up Authentication.

13. How are the different UI-related data elements used during an App-based Authentication in version 2.1.0 and version 2.2.0 of the *Core Specification*?

During an App-based Authentication, the UI-related data elements are used at each step of the flow to:

- communicate the device and 3DS SDK capabilities
- communicate the ACS decision on how it will perform the Challenge
- process the Challenge between the ACS and the 3DS SDK; and
- report on the Challenge outcome.

The usage of the UI data elements in the flow is described below.

The 3DS Server collects all the data necessary for the AReq message. For UI-related data, it provides the Device Rendering Options Supported data element, which describes all the 3DS SDK-supported UIs and the options for rendering the Challenge screens.

The 3DS Requestor App and the 3DS Server do not have the option to change or select the values for the Device Rendering Options Supported data element. For 3DS version 2.1.0 and version 2.2.0, the 3DS SDK must support all the possible device rendering options.

- The 3DS Server conveys this data element in the AReq message.

  The Device Rendering Options Supported is a required data element in the AReq message and is detailed in Table A.13 in version 2.1.0 and version 2.2.0 of the *Core Specification*. For 3DS version 2.1.0 and version 2.2.0, the 3DS SDK shall support all the possible device rendering options.

- The ACS determines the type of Challenge that it will execute and the UI of the 3DS SDK that it will use. The ACS provides this information in the ARes message in two data elements:
  - Authentication Type – indicates the Authentication method that the ACS will use.
  - ACS Rendering Type – indicates the initial UI interface and template that the ACS will require the 3DS SDK to present to the consumer.

In the case of a Challenge, the Authentication Type (see Table A.1) and the ACS Rendering Type (see Table A.12 in version 2.1.0 and version 2.2.0 of the *Core Specification*) are required data elements in the ARes message.

- The 3DS SDK and the ACS establish a secure connection to perform the Challenge process.

  The ACS defines the information needed to display a Challenge on the Consumer Device for the selected Challenge method. Additionally, the ACS selects a UI template using the ACS UI Type and sends the Challenge information and UI to the 3DS SDK.

  For the first CReq/CRes message, the ACS should select the ACS UI Type in line with the information provided in the ACS Rendering Type.

  Therefore, if the ACS indicated in the ACS interface that it will use the Native UI, then it should select one of the four Native UI templates available on the 3DS SDK (Text, Single-Select, Multi-Select and Out-of-Band (OOB)). If the ACS indicated that it will use the HTML UI, then it should select the HTML template.

  The ACS UI Type is a mandatory data element in the CRes message and is detailed in Table A.1 of the *Core Specification*.

- When the Challenge is completed, the ACS reports the result of the Authentication in the RReq message, including the Authentication Type and ACS Rendering Type data elements.

  Authentication Type and ACS Rendering Type are mandatory data elements in the RReq message.

14. In the *Core Specification*, the sentence "For a message that cannot be recognised, …" is used in numerous places, for example, in Section 5.9.3 (DS ARes Message Error Handling) and Section 5.9.8 (DS RReq Message Error Handling). Which data elements are used to recognise a transaction?

It depends on the specific error situation and the specific implementation. In some situations, one or more of the identification data elements (e.g. Transaction IDs) may be recoverable and could be used to recognise a transaction if the Message Type is not readable. Additionally, the presence of elements specific to the Message Type could be used to recognise a message.

The *Core Specification* does not mandate a specific method of recognising a transaction – that will be up to each implementation.

If a 3DS component does not recognise a message, it may return an HTTP response status 400 (or equivalent) instead of an Erro message to the sending component.

15. What is the time standard for a 3DS Authentication?

The 3DS Server must provide all date- and time-related data converted from local time into Coordinated Universal Time (UTC) so that the DS and ACS can interpret them correctly during checks and risk analysis. This applies to "Purchase Date & Time" and all the date-related data elements in the Merchant Risk Indicator.

When validating the data elements of the AReq message, the DS or ACS may respond with an error (Error Code = 203) if the Purchase Date & Time is not converted into UTC.

16. Some devices return a Browser Screen Color Depth that is not on the list of accepted values in version 2.1.0 and version 2.2.0 of the *Core Specification*. How should a 3DS Server handle a Browser Screen Color Depth value that is not supported by the *Core Specification*?

Due to the continuous evolution of both devices and browsers, the recommendation is for the 3DS Server to provide the closest lower value listed in the *Core Specification* (for example, 24 instead of 30) to avoid an error in response to the AReq message.

If a non-accepted value is received by the DS, the DS may optionally change the Browser Screen Color Depth value to the closest lower value.

This has been addressed in version 2.3.1 of the *Core Specification* by allowing any value from 1 to 99.

17. How should the Cardholder Billing and Shipping Address Country and State data elements be encoded?

The billing and shipping address country values follow the ISO 3166-1 standard, which is encoded with 3 digits. For example, USA = 840 or China = 156.

The billing and shipping address state values follow the ISO 3166-2 standard, which is encoded in 1–3 alphanumeric characters. ISO 3166-2 includes the state and country. The first two characters are the alpha-2 country as provided by the ISO 3166-1, followed by a hyphen and the country subdivision name (or state).

For example, the state values for California, USA (US-CA) and Shanghai, China (CN-31) are ‘CA’ and ‘31’, respectively.

18. When should Transaction Status be present in the CRes message and what are the valid values?

The Transaction Status data element in the CRes message informs the 3DS SDK or Merchant whether the Challenge successfully completed. Therefore, the Transaction Status data element should be present only in the Final CRes message from the ACS and it should contain only the value N or Y, as defined in Table A.1 and Tables B.4 and B.5 of the *Core Specification*. If the Transaction Status is received in a non-Final CRes message, it should be ignored by the 3DS SDK as per Req 209.

19. The retrieved Browser Language has a length greater than 8 characters. How should the Browser Language data element be populated for version 2.1.0 and version 2.2.0 of the *Core Specification*?

The Browser Language as well as the principles of truncation are defined by BCP 47 (see Section 4.4.2 of the IETF BCP 47). The maximum length is 35 characters when it contains all the optional information. The 3DS Server should provide the main language tag, which has a maximum length of 8 characters.

20. As the ACS receives SDK Device Information version 1.4 or lower, should the value of the Advertising ID be consistent between transactions?

The Advertising ID is unique to the app and the device, and should not vary between transactions. If the 3DS SDK is not able to retrieve the Advertising ID, it must list this data element in the DPNA (Device Parameter Not Available). All device information that the 3DS SDK is unable to provide or retrieve must be listed in the DPNA. The Advertising ID is not present in Device Information version 1.5 or higher.

21. The existing definition of the Fully Qualified URL in version 2.1.0 and version 2.2.0 of the *Core Specification* does not provide technical details. What is the valid format and encoding for a Fully Qualified URL?

A Fully Qualified URL contains all the information necessary to locate a web resource and is defined as an 'absolute-URL string' with the scheme 'https', encoded in 'UTF-8' using URL code points from https://whatwg.org/.

The URL code points are ASCII alphanumeric, U+0021 (!), U+0024 ($), U+0026 (&), U+0027 ('), U+0028 LEFT PARENTHESIS, U+0029 RIGHT PARENTHESIS, U+002A (*), U+002B (+), U+002C (,), U+002D (-), U+002E (.), U+002F (/), U+003A (:), U+003B (;), U+003D (=), U+003F (?), U+0040 (@), U+005F (_), U+007E (~), and code points in the range U+00A0 to U+10FFFD, inclusive, excluding surrogates and noncharacters.

The 3DS components may Validate the URL format, so in order to avoid any error, it is important to ensure conformance to this definition when providing a URL.

Please refer to the following resources for additional information:

- https://url.spec.whatwg.org/#absolute-url-string
- https://url.spec.whatwg.org/#url-code-points
- https://infra.spec.whatwg.org/#surrogate
- https://infra.spec.whatwg.org/#noncharacter

Example: https://server.domainname.com/acs/auth%20(*ret

Note: A Fully Qualified URL does not contain credentials (See https://url.spec.whatwg.org/#include-credentials).

22. Is it possible to use special characters for Cardholder Name encoding?

Special characters are not supported in version 2.1.0 and version 2.2.0 of the *Core Specification*. The 3DS Server encodes the Cardholder Name as per the Table A.1 definition, strictly using the set of characters listed in EMV Book 4, "Annex B" (ASCII). Therefore, the Cardholder Name field in the AReq message must use ASCII characters without special characters. For example, special characters á, ë, ï, ö must be converted to the ASCII equivalent of a, e, i, o.

Version 2.3.1 of the *Core Specification* permits the use of special characters in the Cardholder Name field.

23. The *Core Specification* defines `sdkCounterStoA` and `ACSCounterAtoS` as a three-digit number in Table A.1, but as an octet in the CReq/CRes message encryption/decryption in Section 6.2. Can you clarify how these values should be processed?

When used in the cryptographic functions in Section 6.2, the `acsCounterAtoS` and `sdkCounterStoA` are coded as one byte, and count from 0 to 255.

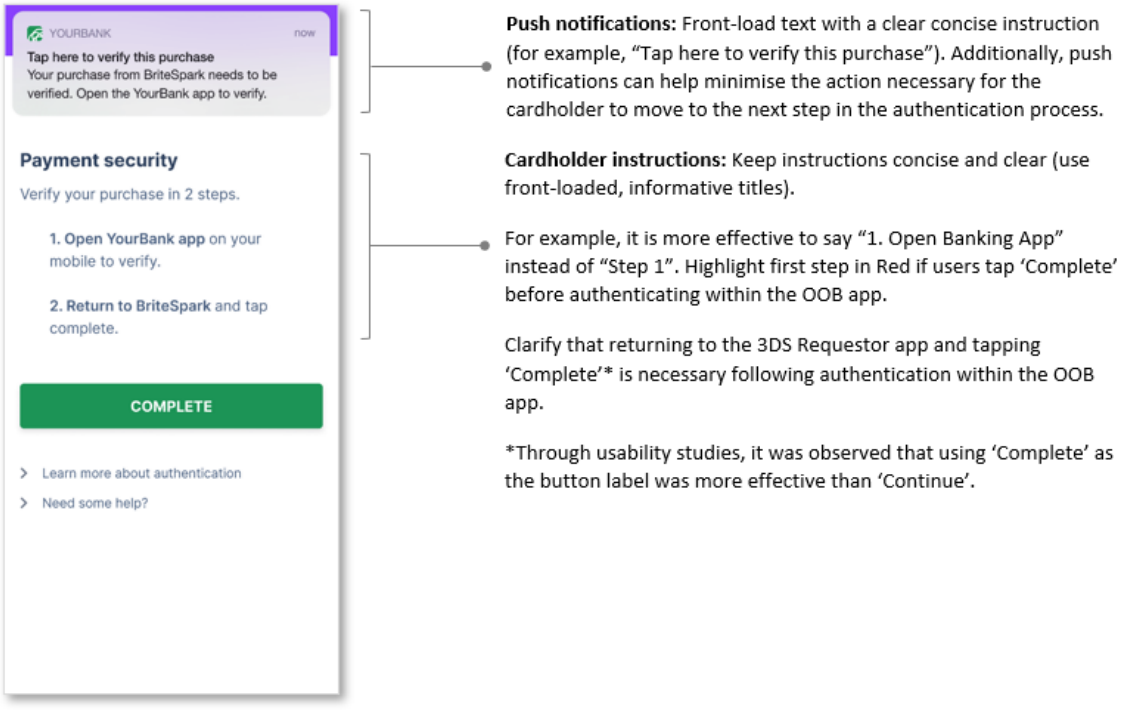
When transmitted in the CReq/CRes messages, the counters are represented as the decimal value equivalent of the byte, encoded as a numeric string.

For example: 0b00100011 (or 0x23) as a byte is equivalent to 35 in decimal and is encoded '035' as a string.

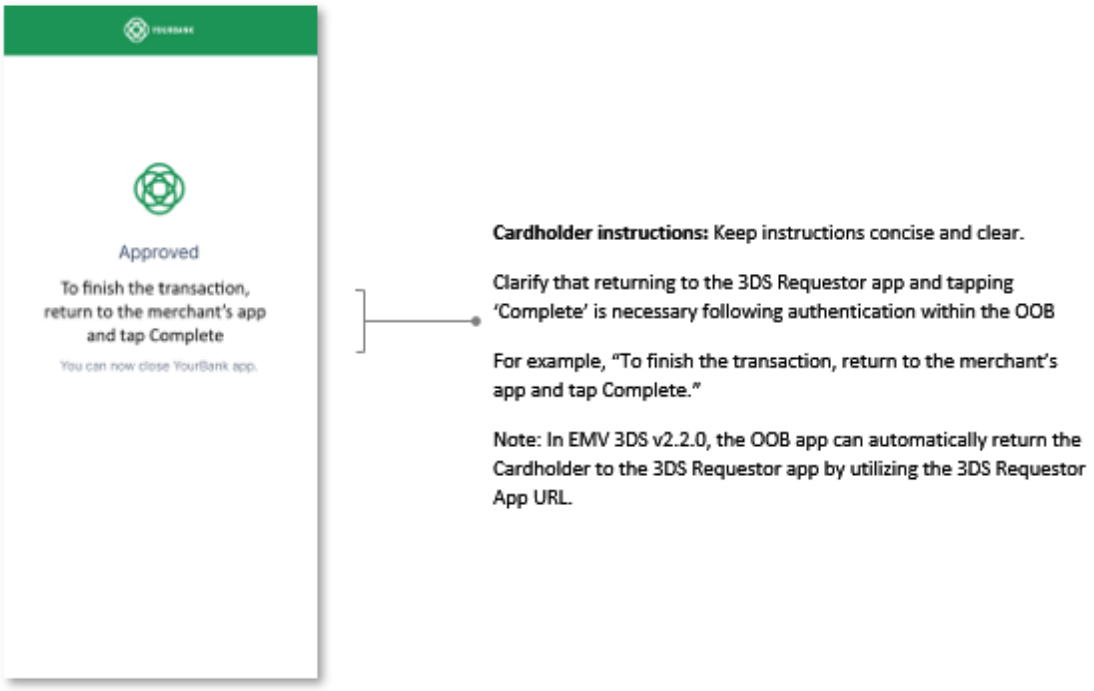
24. For EMV 3DS OOB Authentications, there have been cases where the Cardholder successfully authenticates in the OOB App, but fails to tap the Complete button after returning to the 3DS Requestor App. What are some recommendations to improve the user experience for the App-based OOB flow?

Issuers should educate Cardholders on the OOB Authentication process and flow so that Cardholders know their expected actions to complete an App-based OOB transaction.

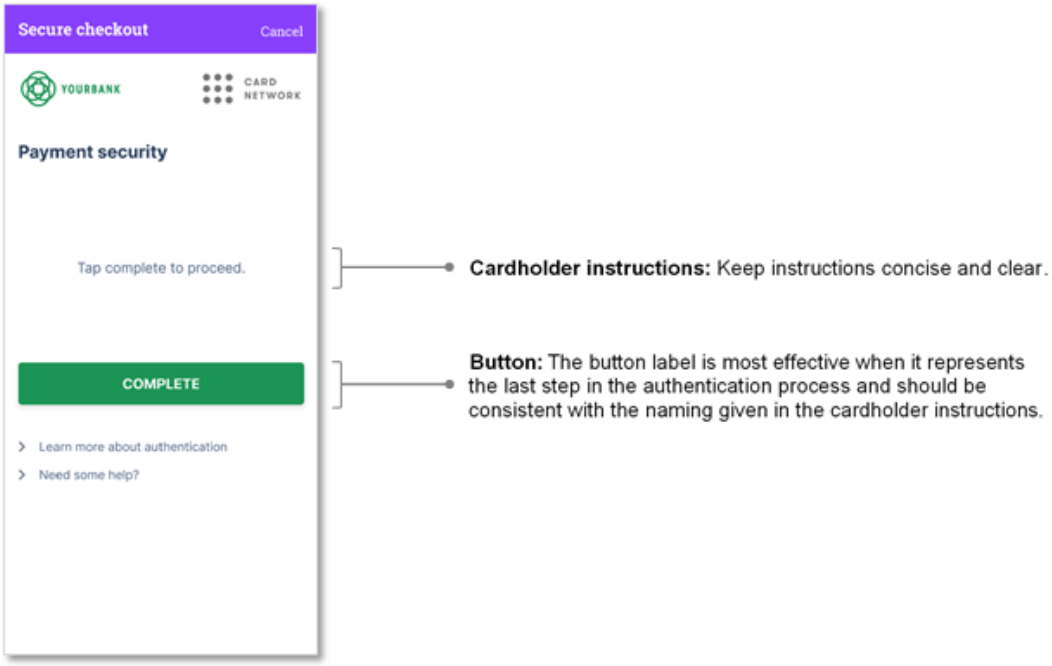
The following is an example OOB template used to provide instructions to the Cardholder before transitioning to the OOB App. The following recommendations are highlighted:



The following is an example screen within the OOB App used to provide instructions to the Cardholder after a successful Authentication. The following recommendations are highlighted:



The following is an example OOB template used to provide instructions to the Cardholder after the Authentication process has completed in the OOB App and returned to the 3DS Requestor App. The following recommendations are highlighted:



Note: In the EMV 3DS version 2.2.0 Native flow, there is an improved user experience as the Cardholder will not need to tap Complete because the automatic CReq message functionality will direct the Cardholder to the 3DS Requestor confirmation page as soon as the 3DS Requestor App is brought to the foreground.

25. The *Core Specification* states that all the messages shall include the UTF-8 charset in their HTTP headers. The examples provided in the *Core Specification* define the charset using upper-case (UTF-8). Is it acceptable to use lower-case (utf-8) to define the charset?

The *Core Specification* references RFC 7231 (https://tools.ietf.org/html/rfc7231) that defines the HTTP header.

The media type present in the HTTP header (Section 3.1.1.1), for example, text/html or charset=utf-8, is specified as case-insensitive. Therefore, both UTF-8 and utf-8 are valid, as is any combination of lower-case and upper-case characters.

26. Which 3DS Requestor Challenge Indicator values are valid for Transaction Status = I in version 2.2.0 of the *Core Specification*?

Transaction Status = I is valid if 3DS Requestor Challenge Indicator = 05, 06 or 07 within the AReq message. See Table A.15 in the *Core Specification* for defined Transaction Status Conditions.

The DS may have defined additional use cases, so footnote 12 should read: “This indicator (I) can be sent only if the 3DS Requestor Challenge Indicator = 05, 06, or 07 within the AReq message (or as specified by DS rules).”

27. In version 2.2.0 and version 2.3.1 of the *Core Specification*, the 3DS Requestor App URL definition states: "Each transaction would require a unique Transaction ID by using the SDK Transaction ID". Should the ACS expect the presence of the SDK Transaction ID in the 3DS Requestor App URL?

The ACS should not expect the presence of the SDK Transaction ID in the 3DS Requestor App URL. It is not mandated as the term "shall" is not used in the description (refer to Section 1.9 of the *Core Specification*).

The SDK Transaction ID is present in each CReq message as a separate data element.

28. How should the App IP Address and Browser IP Address data elements be encoded?

The App IP Address and Browser IP Address data elements are encoded as a string.

- For the representation of the IPv4 address, refer to RFC 791 (https://tools.ietf.org/html/rfc791).
- For the representation of the IPv6 address, refer to RFC 4291 (https://tools.ietf.org/html/rfc4291), in which Section 2.2 defines the different representation options.

29. What UI data elements shall the ACS provide in the CRes message for the different Native UI templates in version 2.1.0 and version 2.2.0 of the *Core Specification*?

Even though the UI data elements are specified as optional in Table A.1, the ACS must provide all the required UI data elements as specified in Table A.18 in the CRes message to the 3DS SDK – otherwise the 3DS SDK will abort the Challenge and return an Error Message indicating that a mandatory data element is missing.

Please note that the presence of the Challenge Information Label is mandatory but missing in the following figures:

- version 2.1.0 – SB 204v7: Figures 4.14 and 4.15
- version 2.2.0 – SB 214v3: Figures 4.15 and 4.16.

Therefore, the ACS provides either:

- the Challenge Information Label as a string with 1 space character (" "); OR
- a meaningful string for the Challenge Information Label.

30. When are the presence conditions of the subfields of a data element that is a JSON object (contains other data or objects) evaluated?

Some data objects have optional or conditional presence in the 3DS messages. The data elements inside these objects may also have presence conditions (required, optional or conditional).

The presence condition of the data elements in an object only applies when the object is present in the message.

For example, the Multi-Transaction object is optional in the AReq message. If it is present in the AReq message, the presence of the Merchant List data element is required, but the presence of the Merchant Amount data element is optional.

31. Is the RReq message always needed for a 3DS transaction?

If the ACS risk assessment concludes that no Challenge is needed (Frictionless transaction), the ACS provides its final response in the ARes message, and its processing of the Authentication ends at this point. Therefore, the ACS never sends an RReq message for a Frictionless transaction.

If the ACS risk assessment concludes that a Challenge is needed (Transaction Status = C or D), it will process the Challenge and provide its final response in the RReq message after the Challenge is completed.

32. Can Transaction Status = C be used for a 3RI transaction?

A 3RI Authentication is initiated by the 3DS Requestor (Merchant) without the Cardholder being present, so the ACS cannot reply with Transaction Status = C as the Challenge cannot be processed by the 3DS Requestor. If there is a need for a Challenge, the only option is to request a Decoupled Authentication (Transaction Status = D) if supported by the ACS. During a Decoupled Authentication, the ACS reaches the Cardholder using an alternative communication channel (not a 3DS flow) to verify the transaction with the Cardholder.

33. How should an HTTP error for a 3DS protocol message be processed?

The *Core Specification* covers the error cases on the 3DS protocol (message validation, conflict, timeout) and the use of the Erro message.

The *Core Specification* refers to the HTTP standard (RFC 2616) and has no additional requirement for the message exchanges (AReq/ARes, CReq/CRes, Erro, …) between the 3DS components. The expectation is that the components will comply with the requirements in RFC 2616 and use the 200 response status code for normal 3DS message exchanges.

Servers may use the standard HTTP status codes to communicate with the client to report any additional errors outside of 3DS message processing, i.e. 3DS components (acting as a server) will issue the relevant HTTP status code as defined in RFC 2616 and RFC 6585 in the response to the client's request made to the server.

HTTP status codes used to communicate errors may or may not have a message returned in the response body. There may be error cases that are treated by the lower layer of the message exchange like HTTP or TCP/IP.

Examples:

- The POST method is mandatory to use for 3DS messages. If the PUT method is used, the error response may come from the HTTP layer with a 405 Method Not Allowed response status code.

- A server cannot respond due to traffic volume; it may respond with a 429 Too Many Requests response status code.
- For invalid Content-Type, the error response may come from the HTTP layer with a 415 Unsupported Media Type response status code.
- To protect the network from attacks such as denial-of-service (DoS), cross-site scripting (XSS) or code (SQL, PHP etc.) injections, the server may return a 400 Bad Request or 403 Forbidden response status code.

The *Core Specification* does not define any message between the 3DS components and the external environment (3DS Requestor Environment or Issuer environment). Other requirements could apply, and an HTTP response status code other than 200 may be used when 3DS components interface with these environments.

Note: Implementers should refer to the relevant HTTP standard, as information RFC 2616 is obsolete but still referenced by the industry and is replaced by RFC 7230 to 7235. For additional HTTP status codes, please also refer to RFC 6585.

34. Why is the Final CRes message content different from the other CRes messages in an App-based flow?

During a Challenge, the ACS provides the UI-related data elements to the 3DS SDK for display to the Cardholder. Therefore, the ACS UI Type is required in all CRes messages for the App-based flow (refer to Table A.1 for the ACS UI type and Table B.4 for the CRes message in the *Core Specification*).

In the Final CRes message, the ACS signals to the 3DS SDK the end of the Challenge and the outcome of the transaction. As the Challenge is completed, there is no UI to display to the Cardholder. Therefore, the Final CRes message has a different message content without any UI-related data elements – in particular, without the ACS UI Type, as specified in Table B.5.

35. Some versions of the ISO standards referenced in the *Core Specification* are outdated. What version should be used?

The latest version of each of the ISO standards referenced in the *Core Specification*, including all published amendments, applies unless a publication date is specified. This is now explicitly stated in Table 1.2 in version 2.3.1 of the *Core Specification*.

36. How can the DS handle multiple keys for the SDK Encrypted Data?

In version 2.1.0 and version 2.2.0 of the *Core Specification*, there is no key identifier in the SDK Encrypted Data. The DS has the following options for decryption:

- Identify the different 3DS SDKs and their DS public keys using the SDK Reference Number.
- Try the different keys.

- Identify the keys by the algorithm (ECC versus RSA) if the DS is using different algorithms for the new key.
- For the RSA key, identify the key from the size of the encrypted data (assuming the keys have different lengths).

In version 2.3.1 and higher, the DS identifies the key used by the 3DS SDK to encrypt Device Information with the key identifier (kid) present in the SDK Encrypted Data.

37. What are the values for the ACS Reference Number and Transaction ID when the DS responds on behalf of the ACS in the ARes message?

In version 2.1.0 and version 2.2.0 of the *Core Specification*, if the DS creates the ARes message on the ACS's behalf (for example, the DS returns Transaction Status = A), the DS sets an ACS Reference Number equal to the DS Reference Number and an ACS Transaction ID equal to the DS Transaction ID.

In version 2.3.1 and higher of the *Core Specification*, please refer to Req 411, 412 and 421.

38. What are the impacts of the Cardholder using private browsing during a 3DS transaction?

Using private browsing has no impact on the 3DS Authentication flow from the 3DS protocol perspective. The 3DS Requestor and the ACS can complete a 3DS Frictionless Flow or a Challenge Flow.

39. How does version 2.2.0 and higher of the *Core Specification* define the 3DS Requestor Decoupled Max Time format?

The 3DS Requestor Decoupled Max Time is coded as a string with a length of 5 characters. It must be padded with leading zeros if the value is less than 5 characters in length.

For example, a numeric value 1 is encoded as 00001.

40. What are the Recurring Indicator use cases?

The Recurring Indicator enables the 3DS Server to describe the type of recurring transaction in terms of amount and frequency – variable or fixed.

- Fixed amount and fixed frequency – for example, newspaper subscription, cloud service
- Fixed amount and variable frequency – for example, prepaid rail travel card with automatic top-up
- Variable amount and fixed frequency – for example, utility bills
- Variable amount and variable frequency – for example, pay-per-use travel card.

In addition, it is possible to differentiate the transaction amount for the first transaction from the transaction amount for subsequent transactions, for example, in case of a promotion for the first amount. The amount of the first transaction is provided in the Transaction Amount data element, and the subsequent transaction amount is provided in the Recurring Amount data element.

41. How does an ACS control the interaction with the Cardholder during a Challenge in an App-based flow using the HTML UI?

During a Challenge on a mobile device, if the ACS uses the HTML UI template, it must include the specified URLs in its HTML code:

- HTTPS://EMV3DS/challenge URL related to the action when the Cardholder submits the data input.
- HTTPS://EMV3DS/openoobApp URL for the OOB HTML UI template when the Cardholder selects the button to switch to the OOB Authentication App.

The 3DS SDK monitors the URL changes to retrieve the Cardholder response as query parameters from the URL (HTTPS://EMV3DS/challenge) and return a parameter string (HTML Action = GET) containing the Cardholder data input.

If the Cardholder clicks the Submit button without entering a value, the 3DS SDK retrieves a form as follows: HTTPS://EMV3DS/challenge?submitbutton=&inputtext=, assuming that the button name is “submitbutton” and some input field name is “inputtext”), but there is no value in the query string. This is also the case when there is only a button but no entry box or radio button, for example, the Complete button used in the OOB example (see Figure 4.41 in version 2.3.1 of the *Core Specification*).

If the query string is completely blank, it is assumed that the acsHTML is malformed (no button related to the action). Therefore, the 3DS SDK sends an Erro message to the ACS.

The ACS must not include any CSS with external references or any JavaScript in the acsHTML. In such cases, the 3DS SDK will block their execution as a security measure, the HTML page will be unusable, and the Cardholder will not be able to complete the Challenge.

42. How can the ACS check the provenance of the 3DS Requestor App and of the 3DS SDK?

For a transaction initiated from a mobile device (App-based flow), the ACS receives the app store from which the 3DS Requestor App was loaded in the Device Information:

- for iOS: data elements I014/I015 in the NSBundle indicate whether the app was obtained and installed from the App Store;
- for Android: data element A126 getInstallerPackageName identifies the Google Play Store from which the package was loaded.

The ACS should verify if the app store is a trusted source for the 3DS Requestor App in its market and utilise this information in its risk assessment.

Additionally, the ACS can verify that the SDK App ID and SDK Reference Number provided in the AReq message are the same as those provided in the SDK Device Information.

43. When should the 3DS Server use 3DS Requestor Challenge Indicator = 13 (Challenge requested by the Issuer) in a 3DS version 2.3.1 Authentication?

Depending on the market and DS rules, the Merchant may directly send an e-commerce transaction for authorisation. In response, the Issuer can indicate that the Merchant should first initiate a 3DS Authentication for the Issuer to perform a Challenge (known as a soft decline). In this case, when the Merchant initiates the 3DS Authentication, the 3DS Server will set the 3DS Requestor Challenge Indicator to 13 (Challenge requested by the Issuer).

44. What is the purpose of *Specification Bulletin No. 255*?

Specification Bulletin No. 255 (SB 255) documents the 3DS version management approach, which aims to provide more independence between the specifications and bulletins, and to eliminate the need to update all the EMV 3DS specifications or to release new specification bulletins in case of a change in one of the documents.

SB 255 provides the status of the *Core Specification*, the *EMV3-D Secure SDK—Device Information* versions, and the list of *EMV 3-D Secure Message Extensions*.

For example, the *EMV 3-D Secure SDK—Device Information* update cycle is linked to OS updates (Android, iOS) and not the EMV 3DS specification release cycle. Other examples are that EMVCo may release a new or updated *3-D Secure Message Extension* or sunset the testing support for a particular protocol version.

45. What is the difference between the Transaction Status in the Final CRes message and the Transaction Status in the RReq message?

The Transaction Status data element is present in the Final CRes message and in the RReq message for different purposes.

The 3DS Requestor uses the Transaction Status provided in the Final CRes message to proceed to the next step in the transaction with the Cardholder by providing the relevant UI (e.g. completion message, closing the Challenge iframe for a Browser-based transaction, closing the 3DS SDK for an App-based transaction, restarting the checkout in case of failure). The Transaction Status values are limited to Y or N to simplify the 3DS Requestor implementation.

The 3DS Server uses the Transaction Status from the RReq message and the related data (Authentication Value) to complete the transaction, usually proceeding with the authorisation for an e-commerce transaction. The ACS has more options for the Transaction Status values in order to provide more detailed information to the 3DS Server.

46. What URL format should be used for the 3DS Requestor App URL in a 3DS version 2.2.0 and higher Authentication?

Version 2.2.0 of the *Core Specification* defines the 3DS Requestor App URL as a Fully Qualified URL ("A Fully Qualified URL contains all the information necessary to locate a resource using the following format: scheme://server/path/resource").

The new OS version (Android and iOS) has made changes to these URLs and their security. Using the Universal/App Link, as defined in version 2.3.1 of the *Core Specification* (refer to the Table 1.3 definition), is strongly recommended.

Universal App Links: standard HTTPS links for opening a specific mobile app, installed on a device. The implementation is platform-specific:

- Android App Links: https://developer.android.com/training/app-links
- iOS Universal Links: https://developer.apple.com/ios/universal-links

47. How does the ACS handle multiple CReq messages during a Challenge?

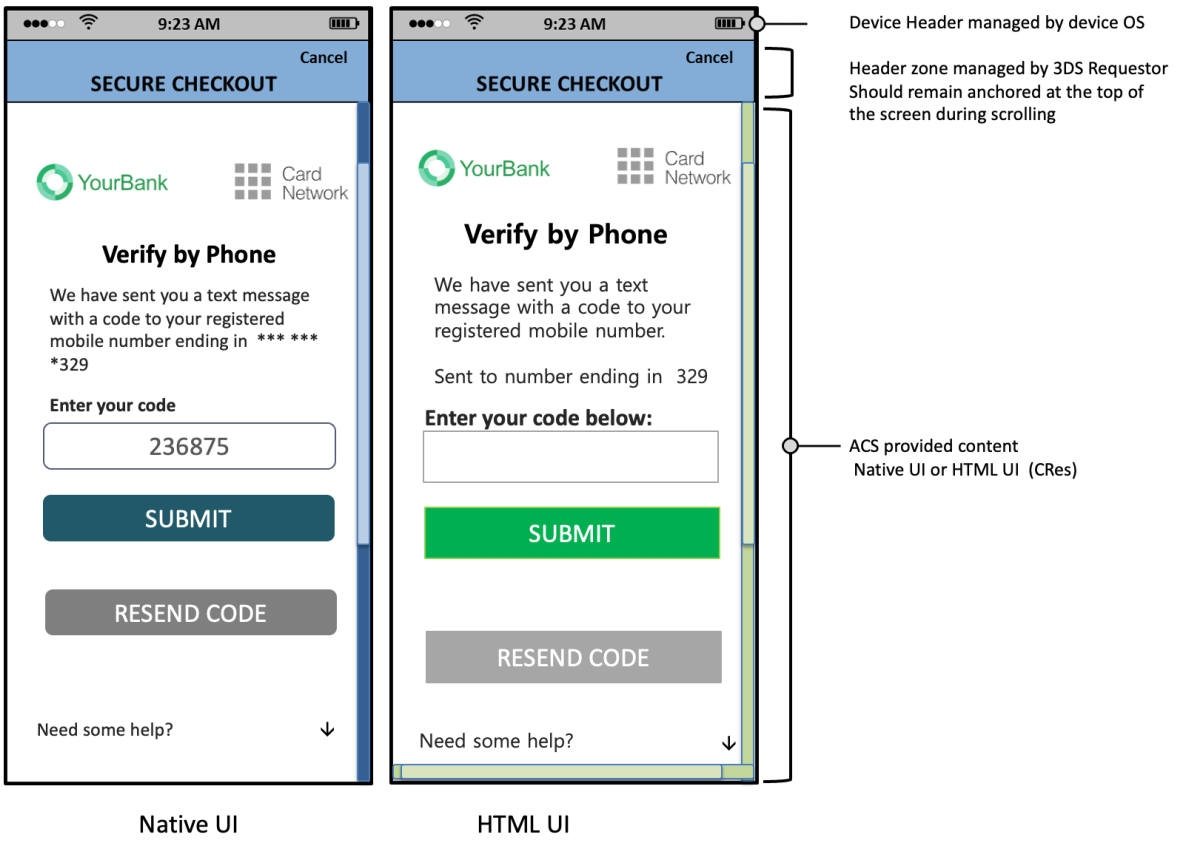
For version 2.1.0 and version 2.2.0 of the *Core Specification*, during a Challenge in a Browser-based flow, the 3DS Server and 3DS Requestor may send multiple CReq messages to recover from an error situation with the Cardholder (for example, the Cardholder requested a page refresh on the Browser after a connection was lost). The ACS can reject the CReq message as a duplicate message, but also has the option to accept the new CReq message, as long as the subsequent CReq message is identical to the original one. The ACS can decide to either restart the Challenge with the Cardholder or continue the Challenge from the last exchange.

This handling of multiple CReq messages is described in Sections 3.3 and 5.9.6 of version 2.3.1 of the *Core Specification*.

48. What is the meaning of full-screen scrolling for the 3DS SDK?

In accordance with Req 418 from version 2.3.1 of the *Core Specification*, the 3DS SDK supports screen scrolling if the UI display is larger than the device screen. The scrolling requirement applies to both the Native and HTML UI templates. The recommendation is to keep the Header zone anchored at the top of the screen (i.e. not allow the header to scroll), so the Cardholder can always see the context of the transaction and access the Cancel button.

The 3DS SDK supports vertical and horizontal scrolling for HTML UI and, at minimum, vertical scrolling for the Native UI as shown below.



49. In the Browser-based flow, can the 3DS Requestor build the CReq message?

In the Browser-based flow as specified in Req 117, the 3DS Server creates and Base64url-encodes the CReq message, then posts the CReq message through the Cardholder Browser to the ACS URL received in the AReq message.

From an implementation point of view, the 3DS Requestor may perform these requirements, but the 3DS Server operator shall ensure that they are executed according to Req 117, in particular, with the correct Base64url-encoding without padding according to Req 192.

50. Why can't the OOB App URL and 3DS Requestor App URL be used in the *EMV® 3-D Secure Bridging Message Extension* for version 2.1.0 messages?

The 3DS Requestor App URL was introduced in version 2.2.0 of the *Core Specification* and the OOB App URL was introduced in version 2.3.1. These enhancements were possible as a result of OS providers improving app-to-app transition through the use of Universal App Links (refer to App Links for Android and to Universal Links for iOS) and security through the use of an HTTPS connection.

Support of the OOB App URL contained in the *EMV® 3-D Secure Bridging Message Extension* requires major changes in the 3DS SDK code. This was accomplished in version 2.2.0 3DS SDKs for the 3DS Requestor App URL through the support of Universal App Links. However, as no such changes were made for version 2.1.0 3DS SDKs, allowing them to implement the OOB App URL and 3DS Requestor URL without EMVCo testing available would present a significant risk of error. Therefore, the decision was made to support OOB App URL only for version 2.2.0.

51. When does the 3DS transaction complete for an ACS?

For a Frictionless transaction, the 3DS transaction for an ACS is completed after the ACS sends the ARes message to the DS. This corresponds to Step 7 of the App-based flow or Step 8 of the Browser-based flow.

For a Challenge transaction (Transaction Status = C or D), the 3DS transaction for an ACS is completed after the ACS sends the Final CRes message in Step 23 of the App-based flow or Step 21 of the Browser-based flow.

52. What are the Card Range Data Download options for the DS and 3DS Server?

Version 2.3.1 of the *Core Specification* introduces a new method of retrieving Card Range Data using file download.

The 3DS Server indicates in the Card Range Data Download Indicator if it supports the file download method.

The DS has the option to return the Card Range Data in the PRes message or, if supported, to provide the Card Range Data File URL for the file download. Both the 3DS Server and the DS need to support the file download in order for this feature to be used.

When the 3DS Server retrieves the Card Range Data from the file download, it performs the same validation and processing as if the Card Range Data were received in the PRes message.

The Card Range Data Download method is also available for the version 2.1.0 and version 2.2.0 protocol using the *EMV® 3-D Secure Bridging Message Extension*. When the Card Range Data File URL is present, the file contains the entire Card Range Data, and the 3DS Server ignores any Card Range Data and Serial Number present in the PRes message.

53. How to proceed if Session Data is not provided in the CRes message for the Browser-based flow?

If an ACS does not return the Session Data in the CRes message, or the Session Data is corrupted or incorrect, the 3DS Requestor should use the information from the CRes message (3DS Server Transaction ID, ACS Transaction ID, Transaction Status) to identify and successfully complete the transaction.

54. What are the guidelines on the maximum length of the HTTP header?

RFC 2616: Hypertext Transfer Protocol – HTTP/1.1 (and updates) does not define a maximum length for the HTTP header. HTTP header values are usually restricted by server implementations. If the header size exceeds the server limit, the server returns an HTTP 413 Payload Too Large response status code.

The *Core Specification* only defines the Content-Type Header and, in version 2.3.1, the presence of the X-Response-ID and X-Request-ID for the Transaction ID for the HTTP header.

Clients or servers in the 3DS protocol should only include relevant information in the HTTP header (Content-Type Header, X-Response-ID and X-Request-ID).

Clients or servers in the 3DS protocol should follow RFC 2616 and ignore any fields that are not relevant for the processing of 3DS messages.

55. Can you provide more information on the presence of the Browser User ID and Browser User Device ID data elements in the Browser-based flow?

In a transaction initiated from a 3DS Requestor App, the 3DS SDK provides the Device Information to the 3DS Server for inclusion in the AReq message, in particular, the User ID and the Device ID. Equivalent data elements (Browser User ID and/or Browser User Device ID) were introduced in version 2.3.1 of the *Core Specification* for use in the Browser-based flow.

In the Browser-based flow, the 3DS Requestor may be able to provide the Browser User ID and/or Browser User Device ID, for example, when cookies are in use. The ACS uses this additional information for its risk analysis, and to better identify the device and the Cardholder.

56. What Browser information is provided by the 3DS Server when JavaScript is not enabled on the Cardholder Browser?

The Browser information consists of the following data elements:

- Browser Java Enabled (`browserJavaEnabled`)
- Browser Screen Color Depth (`browserColorDepth`)
- Browser Screen Height (`browserScreenHeight`)
- Browser Screen Width (`browserScreenWidth`)
- Browser Time Zone (`browserTZ`)

The 3DS Server can obtain this data only if JavaScript is enabled.

If JavaScript is not enabled, the Browser information needs special handling from the 3DS Server and the ACS. The handling depends on the EMV 3DS protocol version and the 3DS component, and is defined in the *EMV 3-D Secure AReq Browser Data Elements Recommendations* document available on the EMVCo website.

57. What may a 3DS component do when it receives an Error Message in error?

The *Core Specification* does not define how a 3DS component should behave upon receiving an Error Message in error.

However, if a specific transaction can be identified from the Error Message, the 3DS component may format and send a new Error Message (as defined in Annex B of the *Core Specification*) to the relevant 3DS components to complete the transaction.

For example, if a DS receives an Error Message from an ACS containing an error (e.g. invalid Message Version Number), the DS may send a correctly formatted Error Message to the 3DS Server to close the transaction.

Note: A receiving component never responds to the sending component of an Error Message, also when it is in error.

58. How do the 3DS Server and the ACS handle linked authentications?

During a 3DS Authentication, in some cases it is useful for the merchant to refer to a previous 3DS transaction. This is possible for the 3DS Server by indicating the previous transaction's ACS Transaction ID in the 3DS Requestor Prior Transaction Reference. This capability is used during SPC and Decoupled Authentication Fallback flows to refer to the initial transaction. Other possible use cases are:

- when renewing a recurring transaction, the Merchant can refer to the initial transaction; and
- when a customer performs an additional transaction that complements an initial transaction.

The ACS should be able to retrieve the previous authentications to conduct a risk analysis that takes into account the relevant transactions and their contexts.

59. How should the 3DS Server populate the 3RI Indicator during a Decoupled Authentication Fallback?

During a Decoupled Authentication Fallback, the 3DS Server shall refer to the original Authentication using the 3DS Requestor Prior Transaction Reference and populate the 3RI indicator, similarly to the 3DS Requestor Authentication Indicator. There is not always a direct match between the values of the 3DS Requestor Authentication Indicator and the 3RI Indicator – for example, for 01 = Payment transaction.

The table below provides recommendations on how to map the 3DS Requestor Authentication Indicator and 3RI Indicator values.

3DS Requestor Authentication Indicator	3RI Indicator
01 = Payment transaction	11 = Other payment
02 = Recurring transaction	01 = Recurring transaction
03 = Instalment transaction	02 = Instalment transaction
04 = Add card	03 = Add card
05 = Maintain card	04 = Maintain card information
06 = Cardholder verification as part of EMV token ID&V	No recommendation
07 = Billing Agreement	12 = Billing Agreement
08 = Split shipment	06 = Split shipment
09 = Delayed shipment	15 = Delayed shipment *
10 = Split payment	16 = Split payment *

*only in v2.3.1.1

60. What is the relation between the challenge exemption requested by the 3DS Server in the 3DS Requestor Challenge Indicator (AReq) and the Transaction Challenge Exemption provided by the ACS (ARes)?

In the AReq message, the 3DS Server indicates a challenge exemption that may be applicable for the transaction using the 3DS Requestor Challenge Indicator (value 5, 8, 10 and 11). If the ACS risk decision is to apply a challenge exemption for the transaction, it provides the information in the Transaction Challenge Exemption (value 5, 8, 10, 11 and 79) in the ARes message. The challenge exemption applied by the ACS may be different from the one indicated by the 3DS Server. The ACS may also apply a challenge exemption even if the 3DS Server did not request one.

3DS Requestor Challenge Indicator

05 = No challenge requested (transactional risk analysis is already performed)

08 = No challenge requested (use Trust List exemption if no challenge required)

10 = No challenge requested (use low value exemption)

11 = No challenge requested (Secure corporate payment exemption)

Transaction Challenge Exemption

05 = Transaction Risk Analysis exemption

08 = Trust List exemption

10 = Low Value exemption

11 = Secure Corporate Payments exemption

79 = No exemption applied

61. What are the criteria for the minimum platform versions that shall be supported by the 3DS SDK in the SDK—Device Information?

The main criteria are platform security and assurance that OS vendors provide security fixes for this version.

62. What are the requirements applicable to x5c certificates in the ACS Signed Content?

The *Core Specification* refers to RFC 7515 (JWS) Section 4.1.6, which stipulates requirements regarding the order of the certificates and their validations. Any error in the certificates should result in a 3DS error during JWS validation.

RFC 7515 refers to RFC 5280 (Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile), which defines the x5c certificate requirements and validation criteria, in particular:

- Validation of the signature on the certificate;
- The certificate period is valid.
- The certificate is not revoked.

The DSs and their CAs must ensure that the certificates provided to the ACS are in the correct order (per RFC 7515) and comply with RFC 5280.

63. What are the requirements and recommendations for the 3DS Method?

The ACS must ensure that there is no user impact or interaction needed during the 3DS Method execution (refer to Req 264 in the *Core Specification*).

The 3DS Requestor redirects the iframe to the ACS URL using an HTTP Post. Therefore, the ACS should use the correct Content-Type (Content-Type: text/html) for the first interaction with the iframe.

The recommendation is for the 3DS Requestor and ACS to perform extensive testing of the 3DS Method to ensure that its execution remains hidden and transparent to the Cardholder.

64. What is the Limited option for the Split-SDK?

The Limited option for the Split-SDK applies to devices that are not capable of supporting cryptographic functions such as key generation and encryption of CReq messages.

The Split-SDK Limited option is only applicable to the Native Client. It is not available for the Browser or Shell Client.

For the Browser and Shell Client Split-SDK variants, the Client is coded as a JavaScript that executes in a Browser iframe or a WebView. These environments can support the 3DS cryptography functions of the 3DS SDK. Therefore, the Limited option is not applicable.

A Limited SDK or Limited Split-SDK only supports dynamic challenges (no static data, e.g., password).

65. What happens if the ACS provides the OOB App URL but not the OOB App Label to the 3DS SDK?

If the 3DS SDK only receives the OOB App URL, the 3DS SDK must not return an Error Message to the ACS.

The lack of OOB App Label does not end the 3DS flow. The 3DS SDK does not display the OOB App URL button in the user interface, as the OOB App Label is not present. The Cardholder would need to manually switch to the OOB App.

66. What value should the ACS provide in the Device Information Recognised Version?

The ACS must return the highest Data Version Number of the Device Information that it supports. Refer to *Specification Bulletin No. 255* for active Data Version Numbers.

ACSs are expected to upgrade their systems as quickly as possible after EMVCo's release of a new version of the Device Information specification and *Specification Bulletin No. 255*.

67. What are the key length constraints for the Challenge Selection Information and Challenge Data Entry data elements in EMV 3DS version 2.1.0 and version 2.2.0?

For a multi-select challenge, the ACS provides an array of key/value pair in the Challenge Selection Information, for example:

```
"challengeSelectInfo":[  {"phone":"Mobile **** **** 321" },
{"email":"Email a*******g**@g***.com"} ]
```

Versions 2.1.0 and 2.2.0 of the *Core Specification* state that the length of key/value pair shall be maximum 45 characters, interpreted as 45 characters for the key + 45 characters for the value. The Challenge Data Entry data element also has a maximum length set to 45 characters.

Therefore, the ACS must ensure that the total length of the keys for a multi-select challenge remains under 45 characters. In the example below, the length of the Challenge Data Entry is 11 characters:

```
"challengeDataEntry":"phone,email"
```

If the total length of the keys is greater than 45 characters, and the Cardholder selects multiple options, their response does not fit in the Challenge Data Entry, and the 3DS SDK is not able to send the response back to the ACS, for example:

```
"challengeDataEntry":"phonephonephonephonephonephonephonephon
ephone,email"
```

In version 2.3.1.1 of the *Core Specification*, the Challenge Selection Information data element is limited to 8 options, with the maximum key length set to 4 characters and the maximum value length set to 45 characters. The Cardholder selection always fits in the Challenge Data Entry, including when 8 options are selected, and the ACS does not need to take care of the total maximum length of the keys.

68. What precautions should be taken when using the Autofill function on iOS devices

Version 2.3.1.1 of the *Core Specification* supports the Autofill function commonly used in mobile applications. In the context of a 3DS Challenge, the ACS indicates that Autofill is supported for data entry, and specifies the expected data entry type (password or SMS OTP) for the 3DS SDK.

On iOS-based devices, the Autofill function uses heuristics to determine when the operating system can automatically input the best possible password entry or when the Cardholder has to enter the data manually. For example, in the case of a two-challenge data entry on the same screen, iOS Autofill heuristics assume that the username and password inputs are on the same page, thus filling both data entries regardless of the Challenge Data Entry Autofill Type. Additionally, iOS may suggest an existing login username and password that may not be relevant to the context of the challenge. It is strongly recommended that ACSs test the Autofill function on iOS-based devices before offering it to Cardholders.

Please note that other operating systems might have similar Autofill heuristics.

Apple references:
- Password AutoFill
- Enabling Password AutoFill on a text input view

69. What is the length and format of the Token Status Indicator in the *EMV® 3-D Secure Payment Token Message Extension*?

The Token Status Indicator is defined as a 40-character string in the *EMV® 3-D Secure Payment Token Message Extension*. The length should read as variable and a maximum of 40 characters, in line with the Token Status Indicator as set forth in version 2.3.1.1 of the *Core Specification*.

70. When should the DS return Error Codes 312 and 313?

Error Codes 312 and 313 were introduced in version 2.3.1.1 of the *Core Specification* in replacement of Error Code 301, to differentiate between two errors in the 3DS flow:

- Error Code 312 – when the DS or the 3DS Server receives multiple RReq messages during a transaction; and
- Error Code 313 – when the DS or the 3DS Server receives an RReq message during a transaction and Transaction Status did not = C or D or S in the corresponding ARes message.

For DSs and 3DS Servers still supporting version 2.1.0 or version 2.2.0, it is acceptable to return Error Code 301 instead of Error Codes 312 and 313, as in their existing implementation.

71. What are the required settings for the Challenge iframe?

In September 2021, EMVCo released the *EMV® 3-D Secure Browser Flow Best Practices* to explain the benefits of using iframes and provide recommendations on iframe settings for secure and smooth processing of the 3DS Challenge and 3DS Method.

These recommendations are now requirements in the different versions of the *Core Specification* (versions 2.1.0, 2.2.0 and 2.3.1.1). Merchants (3DS Requestors) must implement these requirements for their security and to enable the correct processing of the 3DS flow, especially in view of the introduction of SPC and WebAuthn (FIDO authentication) in version 2.3.1.1 of the *Core Specification*.

Not implementing these requirements puts 3DS transactions at risk of failure.

72. What is the format of the Delivery Email Address data element?

The Delivery Email Address data element follows the same format, length and value as the other email data elements in the *Core Specification* (Cardholder Email Address and Seller Email Address), and therefore meets the requirements of Section 3.4 of IETF RFC 5322.

73. Does the DS validate the Base64url-encoding of the SDK Encrypted Data?

If the Base64url-encoding of the SDK Encrypted Data is not valid but the DS is able to recover and decrypt the SDK Encrypted Data, the DS may proceed with 3-D Secure processing and not return an error for incorrect Base64url-encoding.

74. Does the DS validate the Base64url-encoding of the ACS Signed Content?

The DS receives the ACS Signed Content in an ARes message. As it does not consume this data and only passes it to the 3DS Server, the DS is not expected to validate the field content, in particular, correct Base64url-encoding.

75. Does the 3DS SDK validate the Base64url-encoding of the ACS Signed Content?

If the Base64url-encoding of the ACS Signed Content is not valid but the 3DS SDK is able to recover and decrypt the ACS Signed Content, the 3DS SDK may proceed with 3DS processing and not return an error for incorrect Base64url-encoding.

76. What are the accepted Error Codes for missing required data in an optional JSON object that is empty?

If a conditionally optional or optional field is sent as empty, the receiving component returns an Error Message with Error Code 203 (Format or value of one or more Data Elements is Invalid according to the Specification) as per Req 309.

If the empty field is a JSON object and contains mandatory data or data object, the receiving component may return an Error Message with Error Code 201 (Required Data Element Missing).

77. Why was 3DS Requestor Authentication Method Indicator (from version 2.3.1.0) replaced with DS Authentication Information Verification Indicator (in version 2.3.1.1)?

In the 3DS Requestor Authentication Information object, the 3DS Requestor Authentication Method Indicator (`threeDSReqAuthMethodInd`) data element from version 2.3.1.0 of the *Core Specification* was changed to DS Authentication Information Verification Indicator (`dsAuthInfVerifInd`) in version 2.3.1.1 because the DS, rather than the 3DS Requestor, is the source of this data element.

78. How should Base64-encoding (or Base64url-encoding) be applied to binary data elements such as cryptographic information like SHA-256 in 3DS messages?

Base64-encoding directly converts binary data to an ASCII string format, with each 6-bit chunk of binary data converted to a character representation. In the 3DS context, Base64 is used for binary data, such as cryptographic information or images transmitted across channels that only reliably transport text content (ASCII). The sending component directly Base64-encodes the raw binary data and there is no need to convert the binary data to a hexadecimal representation. The receiving component performs the reverse operation (Base64-decoding) to retrieve the original binary data.

For example, for the Cardholder Account Requestor ID, the 3DS Server applies the SHA-256 function to the Cardholder Account Requestor ID. The resulting 256-bit output (raw hashed data) is then encoded using Base64url. This process transforms the original ID into an ASCII format, and the end-result is provided as a string.

79. Should the DS check the consistency of different SDK Transaction IDs present in the AReq message?

The SDK Transaction ID data element may be present up to three times in the AReq message in an App-based flow:

- as the SDK Transaction ID generated by the 3DS SDK and provided by the 3DS Server;
- in the SDK Encrypted Data generated by the 3DS SDK and decrypted by the DS as the Device Information for the ACS; and
- in the SDK Server Signed Content generated by the Split-SDK and validated by the DS (only present if a Split-SDK is in use).

The SDK Transaction ID is present in the SDK Encrypted Data and SDK Server Signed Content to prevent fraud (replay attack).

It is strongly recommended that the DS verify that the value is the same for all three instances of the SDK Transaction ID.

80. What are the conditions for the 3DS SDK to access sensitive user data?

In line with regional regulations and/or Operating System (OS) provider policy, the 3DS Requestor is required to provide prominent disclosure of the use of data in the application at the time of submission to the OS provider. This includes specifying which data elements may be collected and used as part of a 3DS authentication.

At the time of installation, the 3DS Requestor App must prominently disclose to the user that:

- the 3DS Requestor App will access (and use) the sensitive user data (e.g., phone number); and/or
- the 3DS Requestor App will provide access to the sensitive user data to the 3DS SDK (and ACS) for transaction risk assessment.

When use of sensitive user data is invoked for a 3DS authentication, the 3DS SDK must verify user consent and permission prior to requesting any sensitive user data from the 3DS Requestor App. This shall be implemented in accordance with the Permission designation provided in the *EMV® 3-D Secure SDK—Device Information* specification.

Note: The 3DS SDK shall never prompt for user consent or permission to any data within a 3DS authentication.

81. Does the Bridging Message Extension contain all the data elements for an instalment transaction?

The Bridging Message Extension contains many useful data elements for instalment transactions in the Recurring Data object (such as Recurring Amount or Recurring Frequency) that the ACS can use for its risk assessment. However, it is also important to take into account Instalment Payment Data, which is only present in the AReq message.

82. What IP address should the 3DS Server provide in the AReq message for a Browser- or App-based authentication?

The 3DS Server should provide the App IP Address for App-based authentications and the Browser IP Address for Browser-based authentications in the AReq message.

The Browser IP Address is the public IP address of the Browser connecting to the 3DS Requestor. The App IP Address is the external IP address (i.e., the device public IP address) used by the 3DS Requestor App when it connects to the 3DS Requestor Environment.

As this is key information used by the ACS for risk analysis, it is important to provide accurate and meaningful App IP Address and Browser IP Address values.

83. Is it possible to operate a v2.3.1.1-certified Split-SDK in a 3DS v2.2 environment?

A v2.3.1.1-certified Split-SDK may be used in a v2.2 authentication. In such cases, the Split-SDK behaves like a v2.2 Default-SDK from an ACS point of view. Therefore, all the v2.2 UI requirements and CReq/CRes messages as defined in the v2.2 Core and Default-SDK Specifications should be supported.

In addition, the 3DS Server should use the Device Acknowledgement Message Extension, indicating the use of the Split-SDK to the ACS.

84. What device information should be returned by 3DS SDKs for devices not based on an Android or iOS Operating System?

In version 1.6 of the *EMV® 3-D Secure SDK—Device Information* specification, the Platform Provider-specific parameters are specified for use with a Split-SDK implementation. In prior versions of the *Device Information* specification, the Platform Provider-specific fields were acknowledged to be sent when the Common parameters and Device platform-specific parameters were not applicable. The intent of the Platform Provider-specific fields is to enable device data sharing outside of iOS and Android devices, which are typically attributed to Split-SDK usage.

Use of the Platform Provider-specific fields is encouraged when a Default-SDK identifies an Operating System that is not aligned with the iOS- or Android-specific fields. This will be clarified in the next version of the *Device Information* specification.

85. In the PRes message, what values are valid for the ACS Information Indicator data element?

ACS Information Indicator values are only valid for the protocol version for which they are defined. For example, when a 3DS Server sends a PReq message to the DS, the valid ACS Information Indicator values in the EMVCo-defined range are:

- 01 and 02 in a PRes message for v2.1.0
- 01, 02, 03, and 04 in a PRes message for v2.2.0
- 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, and 11 in a PRes message for v2.3.1.

For values reserved for DS use, 3DS Servers should refer to the various 3DS programmes to understand which ACS Information Indicator values have been defined and are considered valid.

If a PRes message were to contain an undefined ACS Information Indicator value according to the protocol version, 3DS Servers must respond with an Error Message.

86. In the v2.2.0 PRes message, what values are valid for ACS End Protocol Version and DS End Protocol Version data elements?

While the description in Table A.6 in version 2.2.0 of the *Core Specification* references Table 1.5 for active protocol version numbers, the only requirements for ACS End Protocol Version and DS End Protocol Version data elements are that they are a String data type and 5 to 8 characters in length. Since there are no specific values defined for these elements, it is acceptable for DSs to set higher protocol versions (e.g., 2.3.1 or any other future versions) as values, as it meets the length/format/values requirements. 3DS Servers must not respond in error in these cases.

Further, with the introduction of v2.3.1, references to Table 1.5 were replaced with *EMV 3-D Secure Specification Bulletin No. 255* (SB 255) to allow for more flexibility as future versions are released. 3DS Servers on version 2.2.0 can reference SB 255 for information on the active protocol versions.

87. How should out-of-band authentication for mobile Browser-based transactions be handled? (NEW)

For detailed recommendations on handling out-of-band authentication for mobile Browser-based transactions, please refer to the Appendix.

Appendix

Out-of-Band Authentication for Mobile Browser-Based Transactions

Background

Some 3DS Requestors and Access Control Servers have reported a lower success rate for OOB authentications initiated from a mobile browser.

These failures manifest in the ACS sending declined authentication responses to the 3DS Server, such as failed challenge interactions (e.g., mainly through timeouts with Challenge Cancelation Indicator 04 or 05), or in the 3DS Requestor being unaware that the Cardholder has successfully completed the challenge.

The issue has been identified as relating to application and memory management by operating systems (Android and iOS).

Error Scenario

This section describes the likely error scenario during an OOB authentication when a Cardholder is interacting with the 3DS Requestor on a mobile Browser, and the OOB authentication is performed on an app on the same device.

1. During the challenge, the ACS instructs the Cardholder to authenticate the transaction using the OOB Authentication App.
2. The Cardholder switches from the mobile Browser (merchant web page) to the OOB Authentication App (usually a mobile banking app).
3. The device operating system (OS) manages the switch of applications, puts the Browser in the background, and starts the OOB Authentication App, bringing the user interface to the foreground. If the memory available on the device is limited, the OS instructs the apps in the background (e.g., the Browser) to free up memory, or, in a worst-case scenario, kill certain apps. In turn, the Browser deactivates tabs to comply with the OS request.
4. When the Cardholder returns to the Browser, the Browser reactivates the tab the Cardholder was on. However, the reactivation process essentially behaves as a refresh of the tab.
5. If the 3DS Requestor did not receive the authentication result from the 3DS Server in the meantime (RReq message from the ACS/Directory Server), the 3DS Requestor Environment reloads the checkout page, opens the challenge iframe, and posts a CReq message to the ACS to restore the challenge context of the transaction.
6. Depending on how the ACS handles the challenge and the receipt of multiple CReq messages, the ACS may return an RReq message with a failed authentication result or an Error Message to the 3DS Server, or the transaction may time out.

This scenario is possible in theory for a transaction initiated from a 3DS Requestor application

(Merchant app), but it is unlikely due to its smaller memory footprint compared to a Browser.

The memory and app management by the OS depends on multiple factors, including memory size, the number of apps open simultaneously, or the OS's capabilities to save and restore the app's context. Therefore, it is not possible to predict the OOB authentication error. Feedback from major OS providers indicates that lower-end devices with limited RAM and processing power contribute to this issue, but higher-end devices are also not immune to this behaviour.

Recommendations

The following recommendations and solutions are proposed to develop best practices for handling OOB challenges and harmonise the Merchant, 3DS Server, and ACS implementations.

1. 3DS Servers should share with the 3DS Requestor, the authentication status (Transaction Status) from the RReq message as soon as it is received. To prevent any delay, the 3DS Server should push the information to the Merchant (instead of the Merchant itself having to retrieve the information).
2. In case of a Browser restart, the 3DS Requestor shall always restore the challenge iframe so that the ACS can send the Final CRes message. The 3DS Requestor should refresh the payment page, reopen an iframe in the checkout page and post the same CReq message as the initial one through the iframe to the ACS.
3. The ACS should verify the completion of the OOB authentication directly from the authentication server and immediately send the RReq message when the OOB authentication is completed. The ACS should not wait for the Cardholder to confirm the completion through Browser interaction (e.g., the Continue button), as the challenge window is lost when the Browser is put to sleep or closed by the device OS, which results in a timeout as the Cardholder interaction is no longer possible.
4. ACSs shall accept multiple CReq messages for a mobile Browser-based transaction, as a possible option under Req 442 (in *Core Specification* version 2.3.1 and Specification Bulletin 214 for version 2.2). The ACS should verify the consistency of the IP address, 3DS Requestor Session Data, and CReq message content before continuing the challenge with the Cardholder. The ACS should check the OOB authentication status before taking any action.
   - If the OOB authentication was completed, the ACS should confirm the challenge completion with the Cardholder and send the Final CRes message.
   - If the OOB authentication was not completed or performed, the ACS should restart the OOB authentication with the Cardholder.

     Note: If the ACS receives the CReq message more than once after attempting an OOB authentication, it must ensure that it is not a loop: OOB authentication not performed – receipt of a new CReq message, and exit it if necessary.

If a CReq message is received after the RReq/RRes message, the ACS should accept the CReq message (not return an Error message), and confirm challenge completion with the Cardholder, and send the Final CRes.

If the ACS has already sent the RReq message and lost the connection to the challenge iframe, the ACS may not be able to send the Final CRes message if the 3DS Requestor does not re-open the challenge iframe.

5. Issuers should be mindful of the size of the OOB Authentication App and keep it minimal to avoid device memory issues. Heavy applications can increase the likelihood of the OS requiring the Browser to free up resources.

Addressing these issues through standardised practices and leveraging new technologies is expected to improve the challenge success rate, leading to better user experiences and more reliable transactions.

Sequence Diagram

The following sequence diagram illustrates the 3DS Browser-based flow, taking into account the above recommendations to prevent the failure of the OOB authentication when performed on a mobile device.

1. The Cardholder makes a purchase and proceeds to checkout.
2. The 3DS Requestor initiates a 3DS authentication.
3. The 3DS Server sends an AReq message.
4. The ACS responds with an ARes message requesting a challenge.
5. The 3DS Requestor proceeds with the challenge, opens an iframe in their checkout page and posts the CReq message through the iframe to the ACS.
6. The ACS provides the UI in the iframe and instructs the Cardholder to proceed with an OOB authentication.
7. The Cardholder switches to the mobile OOB App that is on the same device. The device OS sets the Browser to an inactive state or closes the Browser to free up device resources.
8. The ACS actively retrieves the result of the authentication from the OOB authentication server.
9. The ACS sends the result of the authentication in an RReq message to the 3DS Server.
10. Immediately upon receiving the RReq message, the 3DS Server shares the authentication result with the 3DS Requestor.
11. The 3DS Server sends an RRes message to the ACS through the DS.
12. In the meantime, the Cardholder switches from the OOB App back to the Browser.
13. The device OS brings the Browser back to the foreground.

14. The 3DS Requestor refreshes the payment page, reopens an iframe in the checkout page and posts the same CReq message as the first one through the iframe to the ACS.
15. The ACS provides the UI in the iframe and instructs the Cardholder to confirm the challenge completion.
16. After receiving the Cardholder's confirmation, the ACS sends the Final CRes message through the iframe to the 3DS Requestor to indicate the end of the challenge and the outcome of the authentication.

The 3DS Requestor closes the iframe and updates the UI according to the outcome of the authentication and/or authorisation.