EMV®
3-D Secure

Split-SDK Specification

Version 2.3.1.0
August 2022

Legal Notice

The EMV® Specifications are provided “AS IS” without warranties of any kind, and EMVCo neither assumes nor accepts any liability for any errors or omissions contained in these Specifications. EMVCO DISCLAIMS ALL REPRESENTATIONS AND WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AS TO THESE SPECIFICATIONS.

EMVCo makes no representations or warranties with respect to intellectual property rights of any third parties in or in relation to the Specifications. EMVCo undertakes no responsibility to determine whether any implementation of the EMV® Specifications may violate, infringe, or otherwise exercise the patent, copyright, trademark, trade secret, know-how, or other intellectual property rights of third parties, and thus any person who implements any part of the EMV® Specifications should consult an intellectual property attorney before any such implementation.

Without limiting the foregoing, the Specifications may provide for the use of public key encryption and other technology, which may be the subject matter of patents in several countries. Any party seeking to implement these Specifications is solely responsible for determining whether its activities require a license to any such technology, including for patents on public key encryption technology. EMVCo shall not be liable under any theory for any party’s infringement of any intellectual property rights in connection with the EMV® Specifications.

Contents

Figures

Tables

1 Introduction

For an introduction to EMV® 3-D Secure (EMV 3DS), refer to the *EMV® 3-D Secure Protocol and Core Functions Specification* (hereinafter referred to as the *Core Specification*). As part of EMV 3DS, the 3DS Client is the component that facilitates cardholder interaction. This document introduces the Split-SDK Client approach to that interaction.

The Split-SDK functions much like the 3DS Default-SDK described in the *EMV® 3-D Secure—SDK Specification* and follows the App-based flow. The distinction of the Split-SDK is that some client functionality does not run on the device, but on a server component, thus implementing a model that splits the functionality between a Split-SDK Client (client side) and a Split-SDK Server (server side). SDK Type indicates whether a Default-SDK or Split-SDK is involved.

Although other 3DS components (for example, the ACS) have the same interaction model as for a 3DS Default-SDK, the Split-SDK integration model with the 3DS Requestor (for example, with Merchants) differs in that the Split-SDK may be integrated within a server component rather than within a 3DS Requestor App.

This *EMV® 3-D Secure Split-SDK Specification* covers the basic functionality of a split architecture and identifies three implementation variants.

- Split-SDK/Native: the Client functionality is implemented using native platform code embedded within a 3DS Requestor App, as defined in Chapters 3 and 4.
- Split-SDK/Browser: the Client functionality is implemented using JavaScript running in a device Browser. The JavaScript is delivered from the Split-SDK Server at the time of the authentication. A Split-SDK/Browser-based Client utilises the Split-SDK Client/Server flow as defined in Figure 3-1, but with a JavaScript Client as defined in Chapter 5.
- Split-SDK/Shell: the Client functionality is implemented using JavaScript running in a secured WebView opened by the Split-SDK/Shell that is embedded in the 3DS Requestor App. The JavaScript is delivered from the Split-SDK Server at the time of the authentication. A Split-SDK/Shell-based Client utilises the Split-SDK Client/Server flow as defined in Figure 3-1, but with a JavaScript Client as defined in Chapter 6.

If the Client cannot securely encrypt the CReq message, then the Split-SDK is considered Limited, as defined in Section 3.3. For a Limited Split-SDK, the range of allowed Authentication Methods is limited to those that are dynamic (static Authentication Methods are not supported).

In the *Core Specification*, the Split-SDK Type data element indicates the implementation variants (Split-SDK Variant) and whether the Split-SDK is Limited (Limited Indicator).

1.1 Purpose

This document describes the EMV 3DS Split-SDK approach and model and outlines the architecture requirements. Specific implementations may vary depending on the 3DS Integrator’s choice. Therefore, many requirements do not describe actual messages or APIs, instead providing the high-level functionality implemented on the client side versus the server side.

Additionally, this document describes the Split-SDK variants and their specific requirements.

1.2 Audience

This document is intended for use by implementers developing a Split-SDK.

1.3 Normative References

The following standards contain provisions that are referenced in this specification. The latest version, including all published amendments, shall apply unless a publication date is explicitly stated.

Table 1.1: Normative References

Reference	Publication Name	Bookmark
RFC 4122	A Universally Unique IDentifier (UUID) URN Namespace	https://tools.ietf.org/html/rfc4122
RFC 7159	The JavaScript Object Notation (JSON) Data Interchange Format	https://tools.ietf.org/html/rfc7159
RFC 7515	JSON Web Signatures (JWS)	https://tools.ietf.org/html/rfc7515
RFC 7516	JSON Web Encryption (JWE)	https://tools.ietf.org/html/rfc7516
RFC 7517	JSON Web Key (JWK)	https://tools.ietf.org/html/rfc7517
RFC 7518	JSON Web Algorithms (JWA)	https://tools.ietf.org/html/rfc7518
ECMA-262	ECMAScript® 2020 Language Specification	https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

1.4 Definitions

In addition to the definitions listed in Table 1.3 in the *Core Specification*, this specification uses the definitions listed in Table 1.2 below.

Table 1.2: Definitions

Term	Description
3DS Default-SDK	Software embedded in a 3DS Requestor App that operates on the user device as defined in the *EMV® 3-D Secure—SDK Specification*.
Cross-Origin Resource Sharing (CORS)	HTTP header-based mechanism that allows a server to declare resources that can be shared with other origins (domains).
Content Security Policy (CSP)	HTTP response header that instructs browsers on the location and the resources that a page is allowed to load.
Split-SDK	Entity representing the combination of the Split-SDK Client and the Split-SDK Server. Specified SDK functions can be performed by the Split-SDK Client or the Split-SDK Server, or both in certain situations.
Split-SDK Client	System or software that operates on the user device and enables user interaction as part of the transaction flow in a Split-SDK implementation.
Split-SDK Server	System or software that operates on the provider-defined platform and enables interaction with other 3DS components and Split-SDK Client in a Split-SDK implementation.
Limited Split-SDK	A Split-SDK implementation where the CReq encryption is not performed by the Split-SDK Client but by the Split-SDK Server.
Split-SDK/Browser	Implementation choice for a Split-SDK Server, where the Client functionality is implemented using JavaScript running in a Browser (or similar web technology).
Split-SDK/Native	Implementation choice for a Split-SDK Server, where the Client functionality is implemented using native platform code embedded within a 3DS Requestor App.
Split-SDK/Shell	Implementation choice for a Split-SDK Server, where the Client functionality is implemented as a JavaScript running in a WebView secured by the Split-SDK Shell that is embedded in a 3DS Requestor App.

1.5 Abbreviations

In addition to the abbreviations listed in Table 1.4 of the *Core Specification*, this specification uses the abbreviations listed in Table 1.3 below.

Table 1.3: Abbreviations

Abbreviation	Description
CORS	Cross-Origin Resource Sharing

Abbreviation	Description
CSP	Content Security Policy
UI	User Interface

1.6 Supporting Documentation

The following documents are specific to the EMV® 3-D Secure protocol and should be used in conjunction with this specification. These documents, as well as the *EMV® 3-D Secure Frequently Asked Questions*, are located on the EMVCo website under the 3-D Secure heading.

- *EMV® 3-D Secure—Protocol and Core Functions Specification*
- *EMV® 3-D Secure—SDK Specification*
- *EMV® 3-D Secure SDK Technical Guide*
- *EMV® 3-D Secure SDK—Device Information*
- *EMV® 3-D Secure JSON Message Samples*

1.7 Terminology and Conventions

The following words are used often in this specification and have a specific meaning:

Shall

Defines a product or system capability which is mandatory.

May

Defines a product or system capability which is optional or a statement which is informative only and is out of scope for this specification.

Should

Defines a product or system capability which is recommended.

End(s) 3-D Secure Processing

As outlined in Chapter 3 of the *Core Specification*, defines a specific exception scenario in the 3-D Secure authentication flows where further processing is outside the scope of this specification. Refer to Table 1.3 in the *Core Specification* for additional information.

End(s) Processing

As outlined in Chapter 3 of the *Core Specification*, defines a specific exception scenario in the 3-D Secure authentication flows where a 3-D Secure component experiences an error and does not process the transaction normally. Therefore, subsequent components take action on the error instance. Refer to Table 1.3 in the *Core Specification* for additional information.

3DS SDK

When the *Core Specification* refers to the 3DS SDK in the context of a Split-SDK implementation, it refers to this specification (not the *EMV® 3-D Secure—SDK Specification*).

Activate(s) the 3DS SDK

When the *Core Specification* refers to “Activate the 3DS SDK” in the context of a Split-SDK implementation, the applicable action is the Initiate Authentication Request step, as defined in Step 1 in Section 3.2 of this specification.

Perform(s) the Challenge

When the *Core Specification* refers to “Perform the Challenge” in the context of a Split-SDK implementation, the applicable action is the Pass Data from ARes step, as defined in Step 12 in Section 3.2 of this specification.

2 Getting Started with the Split-SDK

This chapter provides an overview of the Split-SDK components, lifecycle, and flows.

Further information about other components in the 3-D Secure architecture and how a 3DS SDK is integrated into the 3-D Secure authentication flow can be found in the *Core Specification*.

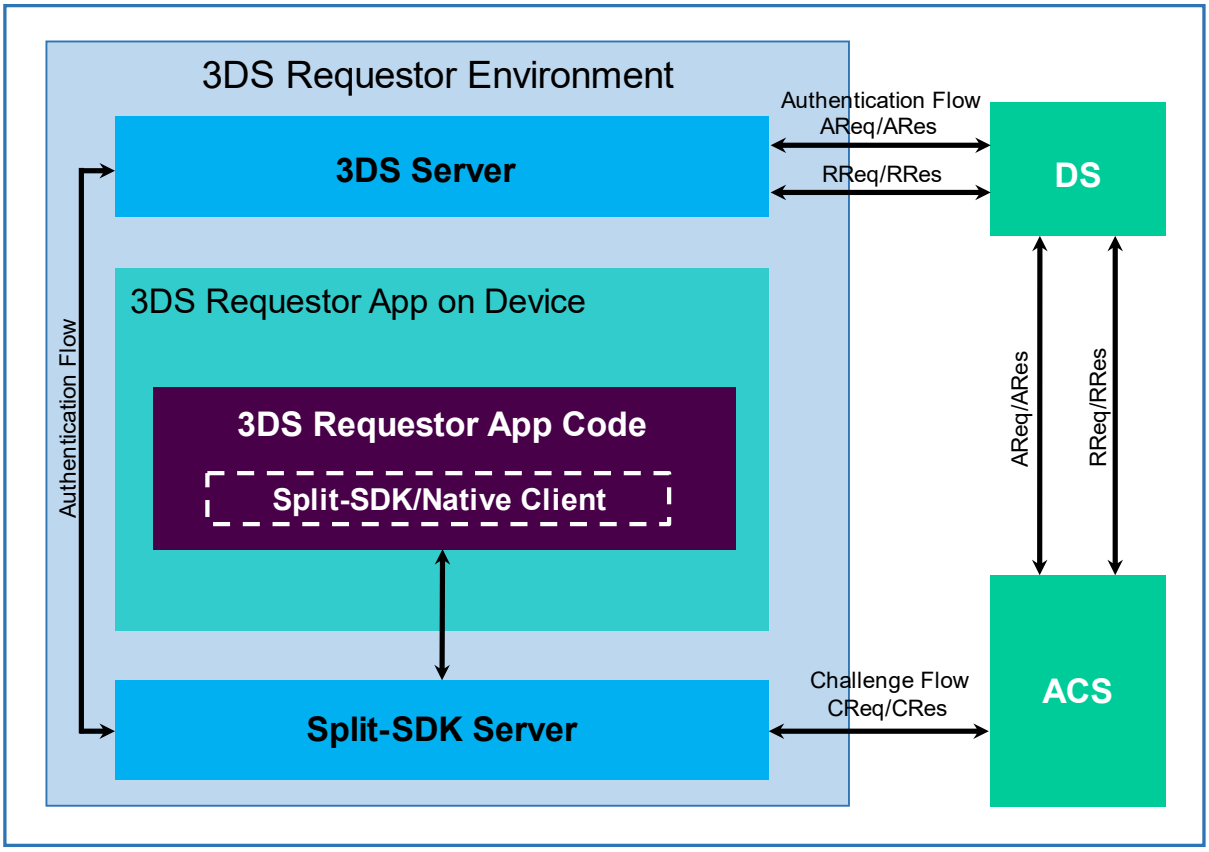
The Split-SDK is a Cardholder interaction side component of the 3-D Secure ecosystem. When a Cardholder initiates an App-based transaction, the Split-SDK integrated with the 3DS Requestor or, as an implementation option, the 3DS Server performs operations related to the client side of a 3-D Secure authentication.

2.1 Component Architecture

Figure 2-1 depicts a typical Split-SDK/Native component architecture. The 3DS Requestor has embedded a Split-SDK Client in its 3DS Requestor App (typically, an SDK embedded in a Merchant mobile application).

Note that different architectures may exist and comply with the requirements in this specification.

Figure 2-1: Split-SDK Component Architecture



The Split-SDK flow has the following steps:

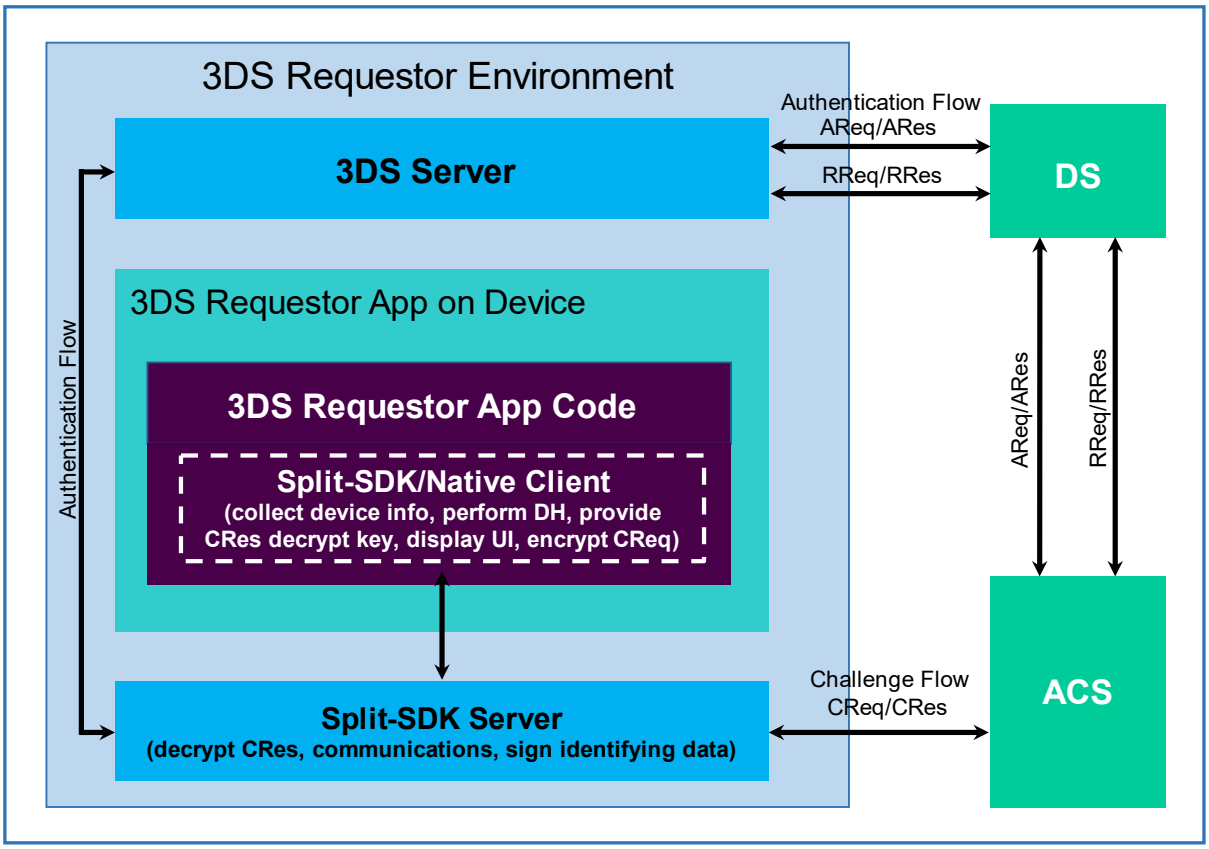
1. The 3DS Requestor utilises the Split-SDK to collect the required authentication parameters and initiate the authentication flow.
   a. If the authentication flow indicates that no challenge is required, then the Frictionless Flow is applied.
   b. If the authentication flow indicates that a challenge is required, then the 3DS Requestor invokes the Split-SDK to apply the Challenge Flow.
2. The Split-SDK performs the following steps:
   a. Communicates with the ACS to initiate the Challenge Flow
   b. Displays the challenge user interface (UI) to the Cardholder
   c. Collects the Cardholder's challenge response
   d. Completes the Challenge Flow
   e. Returns the challenge response to the 3DS Requestor

2.2 Split-SDK Client vs. Split-SDK Server

The Split-SDK divides functionality between the Split-SDK Client and the Split-SDK Server.

Figure 2-2 provides a typical overview of 3DS component functions. However, there may exist implementation variations for some functions that can be performed by both the Split-SDK Client and the Split-SDK Server. See Chapter 3 for information about the functions allowed by each component.

Figure 2-2: Split-SDK Client vs. Split-SDK Server



As depicted in Figure 2-2, the Split-SDK Client:

- Collects data from the device for building the Device Information data element.
- Performs the DH calculation with the parameters from the ACS and generates a pair of session keys for use with the CReq/CRes messages (one for each direction).

  Note: only the A128GCM algorithm is used so that the SDK to ACS (CReq) and ACS to SDK (CRes) keys are different.
- Provides to the SDK Server the session key for CRes message decryption.

  Note: this key cannot be used to decrypt the CReq message, so the Split-SDK Server will not have access to the Cardholder's challenge response credentials.
- Displays challenge content from the Issuer.
- Collects Cardholder credentials during the Challenge Flow
  - Includes the credentials within the CReq message; and
  - Encrypts the CReq message.

The Split-SDK Server:

- Generates the SDK Server Signed Content for inclusion in the AReq message.
- Forwards the CReq message to the ACS.

- Decrypts the CRes message and validates against content security policies.
- Forwards (from the CRes) the necessary information to the Split-SDK Client.

2.3 Limited Split-SDK

A Split-SDK is defined as Limited when one or more functions (outlined in Section 2.2) destined for the Split-SDK Client is/are performed by the Split-SDK Server.

It is up to implementers of a Limited Split-SDK to define which functions are moved to the Split-SDK Server. If any of the required Split-SDK Client functions (as outlined in Section 3.3) is not performed by the Split-SDK Client, the Split-SDK Type is inherently defined as Limited (Limited Indicator = Y) and can therefore support only a limited set of SDK Authentication Types – specifically the types that do not include static information.

2.4 Protocol Version Support

As defined in Requirement 311 in the *Core Specification*, the Split-SDK supports all lower active protocol versions (Protocol Version Status set to Active in *EMV® Specification Bulletin 255*).

The Split-SDK flow as described in this document continues to apply for protocol versions 2.1.0 and 2.2.0, and the Split-SDK implements the requirements of the respective *Core Specification* versions, in particular, for the UI and the OOB authentication.

Note: 3DS Server operators should consider the use of the Device Acknowledgement Message Extension to provide better information on the Split-SDK to the ACS.

3 Message Processing Requirements

This chapter provides the Split-SDK processing flow for both Frictionless and Challenge Flows for a Split-SDK/Native. Further information about other components in the 3-D Secure architecture and how a 3DS SDK is integrated into the 3-D Secure authentication flow can be found in the *Core Specification*.

3.1 Split-SDK Prerequisites

3.1.1 Split-SDK Predefined Data

The Split-SDK flow is depicted in Figure 3-1. Before the message flow is initiated, it is assumed that the Split-SDK has knowledge of the following:

- SDK Reference Number
- SDK App ID
- Split-SDK Server ID
- DS public key used to encrypt device information identified by the Directory Server ID (see Section 6.2.2.1 of the *Core Specification*)
- DS public key used to verify the root of the ACS Signed Content (see Section 6.2.3.3 of the *Core Specification*)
- Split-SDK Server certificate Cert (Pb_{SDK}) (see Section 6.2.2.3 of the *Core Specification*)
- Device data identifying or originating from the device on which the 3DS authentication occurs
- Information about which Message Version(s) is/are supported
- Information about which Device Data Version is supported
- Payment System logos representing each DS
- 3DS Requestor App URL, if this OOB automatic switching feature is supported.

See the *Core Specification* and the *EMV® 3-D Secure SDK—Device Information* for details.

Whether the predefined data is held by the Split-SDK Client or the Split-SDK Server is implementation-specific.

3.1.2 Split-SDK Version Number

To differentiate Split-SDK versions, implementers need to maintain a versioning system. For optimal readability, it is recommended that vendors follow their own nomenclature to determine the version number adhering to the platform-specific versioning standard. For example, the version number may be a string value with the format <major>.<minor>.<build>.<revision>.

Split-SDK implementers shall:

Seq 3.1 ***[Req 1]*** Apply a versioning system for each version of a Split-SDK.

Seq 3.2 ***[Req 2]*** Implement a mechanism and frequency of a version check of the Split-SDK to determine whether the Split-SDK requires an update.

3.1.3 Region-Specific Configuration

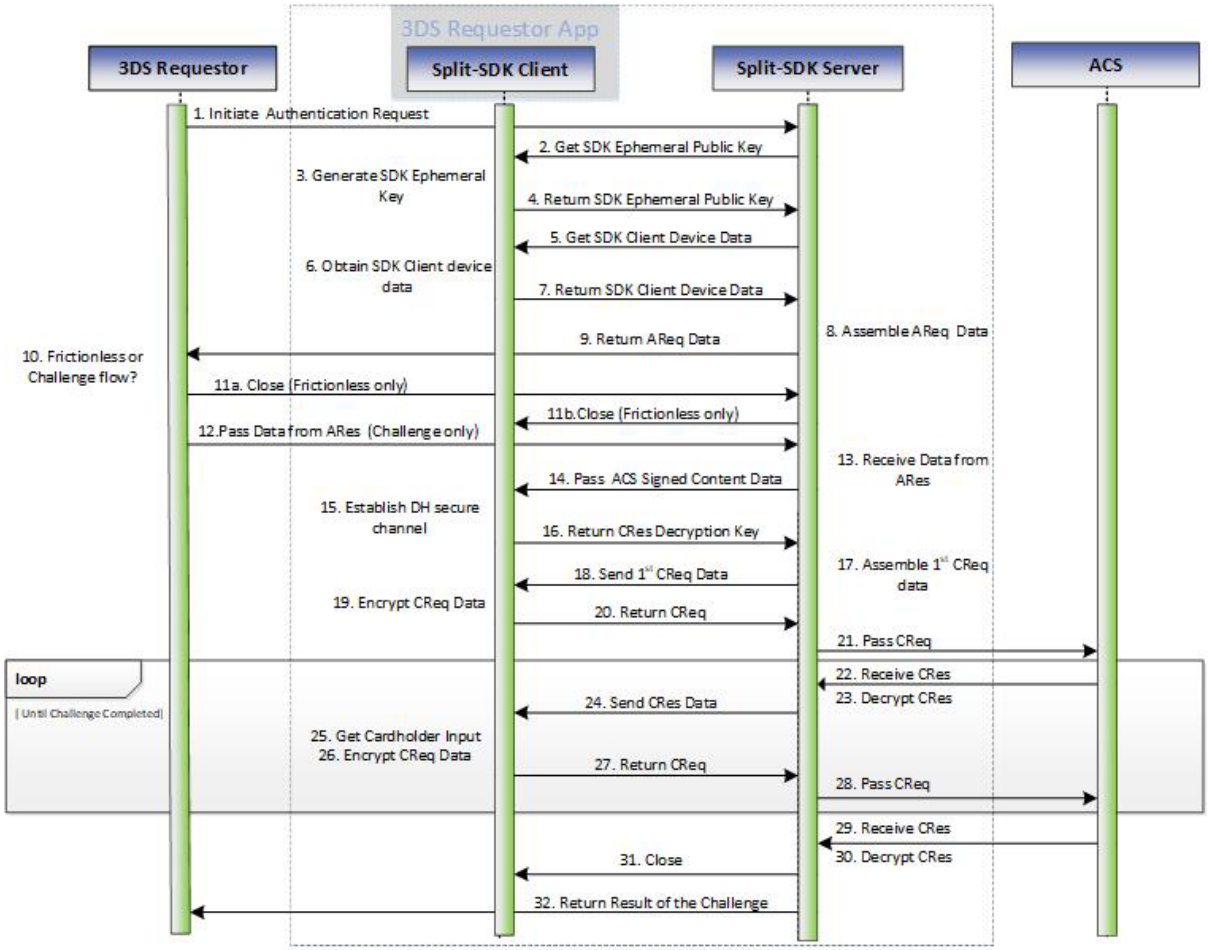
A Split-SDK intended for a global market may need to be configured for regional rules and regulations. Therefore, the Split-SDK may need to support mechanisms to identify the country/region in which the Split-SDK is operating and understand the local/regional rules of operating in that country/region. For example, the type of data that can be collected in a given country/region may determine the type of device data that can be collected during a transaction.

3.2 Split-SDK Message Flow Requirements

The overall flow for a Split-SDK is as defined in Figure 3-1 and for each step in the flow, the function of the step is outlined following the figure.

The flow is outlined for a specific implementation approach. However, implementation variations are allowed provided that the result as seen from the 3DS Requestor/3DS Server and the ACS is indistinguishable (and the security requirements in Section 4 are followed). This particularly relates to whether it is the Split-SDK Client or the Split-SDK Server that performs functions that are not specifically constrained to one component or the other.

Figure 3-1: Split-SDK Message Flow



The 3-D Secure processing flow for App-based implementations contains the following Steps:

Step 1 Initiate Authentication Request

The 3DS Requestor calls the Split-SDK to initiate the authentication request. It is expected that the request ends at the Split-SDK Server.

Step 2 Get SDK Ephemeral Public Key

The Split-SDK Server requests the SDK Ephemeral Public Key from the Split-SDK Client.

Step 3 Generate SDK Ephemeral Key

The Split-SDK Client generates the SDK Ephemeral key pair, retains the private key and returns (in Step 4) the public key to the Split-SDK Server.

The Split-SDK Client shall:

Seq 3.3 ***[Req 3]*** Generate the SDK Ephemeral Key and retain the private key.

Step 4 Return SDK Ephemeral Public Key

The Split-SDK Client shall:

Seq 3.4 ***[Req 4]*** Provide the SDK Ephemeral Public Key to the Split-SDK Server.

Step 5 Get SDK Client Device Data

The Split-SDK Server requests data necessary to identify the device with the Split-SDK Client and possible attributed data from the device.

Step 6 Obtain SDK Client Device Data

The Split-SDK Client obtains:

- the Device Information, OR
- the SDK Encrypted Data, OR
- the Split-SDK Client data from the device that can be used by the Split-SDK Server to uniquely identify the device and form the Device Information.

Step 7 Return SDK Client Device Data

The Split-SDK Client returns the requested device data to the Split-SDK Server.

Note: Depending on implementation, the function of identifying the Split-SDK Client and its Device Information could occur before the execution of this step – for example, when the first connection between the device and the 3DS Requestor was established, in which case Steps 5–7 are not necessary.

Step 8 Assemble AReq Data

The Split-SDK Server assembles the necessary data for the authentication request. The Device Information/SDK Encrypted Data for the AReq message can be generated/created by either the Split-SDK Client or the Split-SDK Server.

The Split-SDK shall:

Seq 3.5 ***[Req 813]*** Provide the Platform Provider-specific Parameters in the SDK Encrypted Data (refer to Section 2.8 of the *EMV® 3-D Secure SDK—Device Information*).

The Split-SDK Client or Split-SDK Server shall:

Seq 3.6 ***[Req 5]*** Encrypt the Device Information using the DS public key identified by the Directory Server ID as defined in Section 6.2.2.1 of the *Core Specification*.

Note: The SDK Encrypted Data could have been created by the Split-SDK Client, in which case this requirement is executed as part of Step 6.

The Split-SDK Server shall:

Seq 3.7 ***[Req 6]*** Generate the SDK Transaction ID as defined in Table A.1 of the *Core Specification*.

Seq 3.8 ***[Req 7]*** Generate the Split-SDK Server Signature JWS object, as defined in Section 6.2.2.3 of the *Core Specification*.

The Split-SDK Server assembles the following data:

- SDK Encrypted Data (encrypted in ***[Req 5]***)
- SDK Transaction ID (generated in ***[Req 6]***)
- Split-SDK Server Signature (generated in ***[Req 7]***)
- SDK App ID
- SDK Reference Number

- Split-SDK Server ID
- Protocol Version (Version used by the Split-SDK for this transaction)
- Device Rendering Options Supported
- SDK Type (Split-SDK = 02)
- Split-SDK Type

Step 9 Return AReq Data

The Split-SDK Server returns the authentication request data assembled in Step 8 to the 3DS Requestor for inclusion by the 3DS Server into the AReq message.

Note: Steps 3–8 can be performed via one or more interactions between the Split-SDK Client and the Split-SDK Server.

Note: With the exception of Steps 3 and 6 (only in the Split-SDK Client) and Step 8, *[Req 7]* (only in the Split-SDK Server), Steps 2–9 can be performed by the Split-SDK Client or by the Split-SDK Server. Depending on implementation choice, some of the interactions described in Figure 3-1 may not be necessary.

Note: The order of execution of Steps 2–7 is implementation-specific.

Step 10 Frictionless or Challenge Flow

Based on the response data from the ARes message returned via the 3DS Server, the 3DS Requestor determines whether a challenge is necessary to complete the authentication request.

If not, the 3DS Requestor closes the transaction with the Split-SDK in Step 11a and Step 11b and **ends 3-D Secure Processing.**

If a challenge is requested (and the 3DS Requestor chooses to continue the 3DS authentication process for the transaction), the 3DS Requestor continues with Step 12.

Step 11a Close (Frictionless only)

The 3DS Requestor closes the transaction with the Split-SDK.

Step 11b Close (Frictionless only)

The Split-SDK Client shall:

Seq 3.9 ***[Req 8]*** Clean up the SDK Ephemeral Key.

Note: Subsequent steps (Steps 12–32) are performed only for a Challenge Flow.

Step 12 Pass Data from ARes

The 3DS Requestor calls the Split-SDK for a challenge process passing the data elements obtained from the 3DS Server via the ARes message.

Step 13 Receive Data from ARes

The Split-SDK Server receives the request for a challenge.

The request is validated by either the Split-SDK Client or Split-SDK Server.

The Split-SDK Client or Split-SDK Server shall:

Seq 3.10 ***[Req 9]*** Report an error to the 3DS Requestor if any of the following data elements are not correctly formatted or are not present in the request:

- ACS Transaction ID

- ACS Signed Content
- 3DS Server Transaction ID
- Authentication Method

Seq 3.11 ***[Req 10]*** Start a timer to measure the overall time taken by the challenge process.

Note: If the Split-SDK Client performs the validation, *[Req 9]* and *[Req 10]* are performed as part of Step 15.

Step 14 Pass ACS Signed Content

The Split-SDK Server passes ACS Signed Content from the ARes message to the Split-SDK Client:

The Split-SDK Server shall:

Seq 3.12 ***[Req 11]*** Send the ACS Signed Content JWS object to the Split-SDK Client.

Step 15 Establish DH Secure Channel

The Split-SDK Client verifies the ACS Signed Content, completes the DH key exchange process, derives the two encryption keys, retains the key used for CReq message encryption and returns (in Step 16) the key used for CRes decryption to the Split-SDK Server. This is described in Step 16 and may be performed for each CReq/CRes (Step 20 and Step 27) if the Split-SDK Server has no temporary key storage (or as an implementation preference).

The Split-SDK Client shall:

Seq 3.13 ***[Req 12]*** Check if the CA public key (root) of the Directory Server CA (DS CA) is present. If not, **end processing**.

Seq 3.14 ***[Req 13]*** Use the CA public key of the DS CA to validate the ACS Signed Content JWS object. If validation fails, **end processing**.

Seq 3.15 ***[Req 14]*** Complete the DH key exchange and key derivation process as defined in Section 6.2.3.3 of the *Core Specification*.

Note: The output of this process is a pair of CEKs. For A128GCM, one half (128 bits) will be used for CReq encryption and the other half will be used for CRes decryption.

Seq 3.16 ***[Req 15]*** Retain the CEK used for CReq encryption.

Step 16 Return CRes Decryption Key

The Split-SDK Client returns to the Split-SDK Server the CEK used for the CRes message decryption to the Split-SDK Server or in Step 20 for each CRes in Step 27.

The Split-SDK Client shall:

Seq 3.17 ***[Req 16]*** Send the CEK used for CRes message decryption to the Split-SDK Server.

Seq 3.18 ***[Req 47]*** NOT send the CEK used for CReq message encryption to the Split-SDK Server.

Note: The requirements in Step 16 can instead be performed as part of each Return CReq (Step 20 and Step 27).

Step 17 Assemble First CReq Data

The Split-SDK Server prepares the data for the first CReq message.

Step 18 Send First CReq Data

The Split-SDK Server sends the first CReq message data for encryption.

The Split-SDK Server shall:

Seq 3.19 ***[Req 17]*** Use the same Message Version for the CReq message as was used in Step 8 for the AReq message data.

Seq 3.20 ***[Req 18]*** Send the data for the first CReq message to the Split-SDK Client.

Note: Steps 17 and 18 are depicted in Figure 3-1 as being performed by the Split-SDK Server, but these steps could alternatively be performed by the Split-SDK Client or by a combination of both components. Therefore, depending on implementation choice, some calls depicted in Figure 3-1 may be unnecessary.

Step 19 Encrypt CReq data

The Split-SDK Client encrypts the CReq message data using the A128GCM algorithm and the CEK used for CReq message encryption.

The Split-SDK Client shall:

Seq 3.21 ***[Req 19]*** Encrypt the CReq message using A128GCM and the CReq message encryption key, as defined in Section 6.2.4.1 of the *Core Specification*.

Step 20 Return CReq

The Split-SDK Client sends the (encrypted) CReq message to the Split-SDK Server. The Split-SDK Client may include the (encrypted) CEK used for the CRes message decryption as defined in ***[Req 16]***.

The Split-SDK Client shall:

Seq 3.22 ***[Req 20]*** Send the CReq message to the Split-SDK Server.

Step 21 Pass CReq

The Split-SDK Server establishes a secure link to the ACS and passes the first CReq message to the ACS.

The Split-SDK Server shall:

Seq 3.23 ***[Req 21]*** Establish a secure link to the ACS as defined in Section 6.1.4.1 of the *Core Specification*. The link is established using the ACS URL received from the 3DS Requestor and verified as part of ***[Req 13]***. If the secure channel cannot be completed, the Split-SDK Server reports the error to the 3DS Requestor and **ends 3-D Secure processing**.

Seq 3.24 ***[Req 22]*** Send the CReq message to the ACS using the secure link established in ***[Req 21]***.

Step 22 Receive CRes

The Split-SDK Server receives the CRes message from the ACS.

Step 23 Decrypt CRes

The Split-SDK Server decrypts the CRes message and can validate against content security policies.

The Split-SDK Server shall:

Seq 3.25 ***[Req 23]*** Decrypt the CRes message using the CEK used for CRes decryption as defined in Section 6.2.4.2 of the *Core Specification* (using the A128GCM algorithm as per ***[Req 19]***).

Seq 3.26 ***[Req 24]*** The Split-SDK Server may validate challenge content from the ACS against content security policies (e.g., check of HTML and images).

Seq 3.27 ***[Req 25]*** If the Split-SDK Server validates against content security policies and the validation fails, the Split-SDK Server shall send an Error Message, as defined in Section A.5.5 of the *Core Specification*, to the ACS with Error Component = C and Error Code = 309.

Step 24 Send CRes Data

The Split-SDK Server sends the CRes data containing the challenge information from the ACS to the Split-SDK Client or a formed UI integrating the challenge questions from the CRes message.

Step 25 Get Cardholder Input

The Split-SDK Client obtains the information requested from the Cardholder using the CRes message data containing challenge questions and UI information.

Depending on the implementation, the UI can be built or formed by either the Split-SDK Client or the Split-SDK Server, but the resulting UI is conveyed by the Split-SDK Client.

The Split-SDK Client shall:

Seq 3.28 ***[Req 26]*** Convey the UI to the Cardholder and obtain the challenge information from the Cardholder as defined by the UI requirements and guidelines in Section 3.4.2 of this specification and Sections 4.1 and 4.2 of the *Core Specification*.

Step 26 Encrypt CReq Data

Using the Cardholder input obtained in Step 25, the Split-SDK Client forms a CReq message and encrypts it using the CEK used for the CReq encryption.

The SDK Client shall:

Seq 3.29 ***[Req 27]*** Use the Cardholder information obtained in ***[Req 26]*** to form a CReq message that is encrypted using A128GCM and the CEK used for CReq encryption, as defined in Section 6.2.4.1 of the *Core Specification*.

Step 27 Return CReq

The Split-SDK Client sends the (encrypted) CReq message to the Split-SDK Server. The Split-SDK Client may include the (encrypted) CEK used for the CRes message decryption, as defined in ***[Req 16]***.

The Split-SDK Client shall:

Seq 3.30 ***[Req 28]*** Send the CReq message to the Split-SDK Server.

Step 28 Pass CReq

The Split-SDK Server passes the CReq message to the ACS.

The Split-SDK Server shall:

Seq 3.31 ***[Req 29]*** Send the CReq message to the ACS using the secure link established in ***[Req 21]***.

Note: Steps 22 through 28 are repeated until a successful challenge verification or until the ACS has exhausted the number of response attempts.

Step 29 Receive CRes

The Split-SDK Server receives the CRes message from the ACS.

Step 30 Decrypt CRes

The Split-SDK Server decrypts the CRes message.

The Split-SDK Server shall:

Seq 3.32 ***[Req 30]*** Decrypt the CRes message using the CEK used for CRes decryption as defined in Section 6.2.4.2 of the *Core Specification* (using the A128GCM algorithm as per ***[Req 26]***).

Step 31 Close

The Split-SDK Server closes the challenge process with the Split-SDK Client.

The Split-SDK Client shall:

Seq 3.33 ***[Req 31]*** Clean up the SDK Ephemeral Key, the CEK used for CReq encryption and the CEK used for CRes decryption.

The Split-SDK Server shall:

Seq 3.34 ***[Req 32]*** Clean up the CEK used for CRes decryption.

Step 32 Return Result of the Challenge

The Split-SDK Server conveys the result of the challenge process to the 3DS Requestor.

3.3 Limited Split-SDK Requirements

While the requirements in Chapters 2, 3 and 4 of this document apply to a Split-SDK, this section specifically defines a Limited Split-SDK and its additional requirements.

Seq 3.35 ***[Req 33]*** If the Split-SDK Client cannot perform one or more of the functions defined in ***[Req 3]***, ***[Req 4]***, ***[Req 8]***, ***[Req 12]***, ***[Req 13]***, ***[Req 14]***, ***[Req 15]***, ***[Req 16]***, ***[Req 19]***, ***[Req 20]***, ***[Req 27]***, ***[Req 28]***, or ***[Req 31]***, then the Split-SDK is categorised as a Limited Split-SDK.
This means that if the SDK Client is not capable of performing any one of the following:

- Verification of the ACS Signed Content; OR
- Completion of the DH exchange and generation of the session keys used for the encryption/decryption of the CReq and CRes messages; OR
- Encryption of the CReq message,

then it is categorised as a Limited Split-SDK.

A Limited Split-SDK shall:

Seq 3.36 ***[Req 34]*** Set SDK Type = 02 (Split-SDK), Split-SDK Variant, and Limited Indicator =Y

Seq 3.37 ***[Req 35]*** Not support SDK Authentication Type = 01 (Static Password) or 06 (KBA).

Seq 3.38 ***[Req 36]*** If Authentication Method containing values of 01 (Static Password) or 06 (KBA) is returned from the ARes message, **end processing**.

3.4 Other Requirements

3.4.1 Timeout Requirements

The Split-SDK Server uses the timer from ***[Req 10]*** to control timeout situations.

The Split-SDK Server shall:

Seq 3.39 ***[Req 37]*** Perform the timeout requirements as defined in Section 5.5 of the *Core Specification*.

3.4.2 UI Requirements

For a Native UI, the Split-SDK can customise the UI elements of the challenge screens.

The Split-SDK shall:

Seq 3.40 ***[Req 38]*** Allow customisation of the UI elements listed below on the challenge screens to be used for a Native UI. The information required for UI customisation is passed to the Split-SDK during initialisation.

- Text font (for Label Text, Button Text, Textbox Text)
- Text size (for Label Text, Button Text, Textbox Text)
- Text colour (for Label Text, Button Text, Textbox Text)
- Button Style
- Textbox Style
- Toolbar (defined as the Header zone set forth in Section 4.1 of the *Core Specification*)

3.4.3 Cancel Situations

The Split-SDK Client or the Split-SDK Server recognises and acts upon a situation where the cardholder cancels the transaction.

The Split-SDK Client or the Split-SDK Server shall:

Seq 3.41 ***[Req 39]*** Follow the Cancel situation requirements as defined in Chapter 3 and Section 4.2 of the *Core Specification*.

3.4.4 Error Situations

Validation of messages and data elements is performed by the Split-SDK Client and the Split-SDK Server, as defined in Chapter 5 of the *Core Specification*.

The Split-SDK Client or the Split-SDK Server shall:

Seq 3.42 ***[Req 40]*** Perform validation of messages and data elements as defined in Sections 5.1, 5.4 and 5.9.7 of the *Core Specification*.

3.4.5 Split-SDK Change Requirements

When a Split-SDK changes, a change process is established using the version number as defined in ***[Req 1]***.

Seq 3.43 ***[Req 41]*** The following types of changes shall require an update to the Split-SDK:

- Changes to the Split-SDK
  - The Split-SDK may change if any functional changes, bug fixes, security fixes, performance fixes, etc. are implemented.
- Changes to the predefined data in the Split-SDK
  - Refer to Section 3.1 for the list of predefined data items.

4 SDK Security

This chapter describes the basic security requirements that are to be implemented by a Split-SDK.

4.1 Split-SDK Security Goals

The 3-D Secure ecosystem processes sensitive information that comes from cardholders and payment processing systems. Therefore, security is a fundamental aspect of each 3-D Secure component, including the Split-SDK.

The four primary security goals for the Split-SDK are as follows:

- Protect sensitive cardholder information (for example, a Cardholder's response to an authentication challenge) while being transferred, processed or at rest.
- Protect sensitive 3-D Secure system information while being transferred, processed or at rest. In the SDK context, this information is used to connect the SDK to the ACS.
- Control access to the process and information that is used during interactions between the 3DS Requestor App and Split-SDK.
- Ensure that the interaction between the Split-SDK Client and Split-SDK Server is performed with the three previous security goals in mind.

4.2 SDK Initialisation Security Checks

The Split-SDK conducts security checks to inform the ACS as to the state of the device. These checks originate from the device but can be collected by either the Split-SDK Client or the Split-SDK Server.

The Split-SDK Client or the Split-SDK Server shall:

Seq 4.1 ***[Req 42]*** During initialisation, conduct the security checks as outlined in Table 4.1 and make the result available as a list of warnings in the Device Information JSON data with key as "SW".

For additional information, refer to the *EMV® 3-D Secure SDK—Device Information*.

Table 4.1: Split-SDK Initialisation Security Checks

Security Warning ID	Description	Severity Level
SW01	The device is jailbroken	HIGH
SW03	An emulator is being used to run the 3DS Requestor App	HIGH
SW04	A debugger is attached to the 3DS Requestor App	MEDIUM

Security Warning ID	Description	Severity Level
SW05	The OS or the OS version is not supported	HIGH

4.3 Split-SDK Prerequisite Data

The prerequisite data utilised by the Split-SDK needs to be protected against accidental or deliberate modification.

Seq 4.2 ***[Req 43]*** The prerequisite data (as outlined in Section 3.1 of this document) shall be securely stored in the Split-SDK code to prevent modification.

4.4 Split-SDK Security Requirements

An integral aspect of the security of a Split-SDK is how the Split-SDK Client and the Split-SDK Server interact securely to avoid leakage between or substitution of these components. Therefore, the Split-SDK Server, which is expected to be the controlling entity, needs to have robust mechanisms in place to identify and authenticate the Split-SDK Client.

This specification does not define a specific architecture, protocol, or method for doing so, but it is expected that a Split-SDK implementer is able to identify each Split-SDK Client uniquely and uses mechanisms to protect against leakage, modification and substitution of these components on a level similar to what is defined in Annex D to the *Core Specification.*

The Split-SDK Server shall:

Seq 4.3 ***[Req 44]*** Be able to uniquely identify a device with the Split-SDK Client.

The Split-SDK Client and the Split-SDK Server shall:

Seq 4.4 ***[Req 45]*** Implement a mutual authentication protocol and an encryption protocol that protects the data exchanged between the Split-SDK Client and the Split-SDK Server.

The Split-SDK Server shall:

Seq 4.5 ***[Req 55]*** Stop the transaction if it is not able to authenticate the Split-SDK Client or detects that the Split-SDK Client is compromised.

Section 6.2.4 of the *Core Specification* defines two algorithms for usage of the CReq encryption/CRes decryption keys. In this specification, as set forth in ***[Req 19]*** and ***[Req 27]***, in order to use two keys, only the A128GCM algorithm can be used.

The Split-SDK Server is not allowed to change or store any of the content received in a CRes message.

The Split-SDK Server shall:

Seq 4.6 ***[Req 46]*** Not store the content of a received CRes message beyond the current transaction.

Seq 4.7 ***[Req 814]*** Not alter the content of a received CRes message.

5 Split-SDK/Browser Requirements

The Split-SDK/Browser variant implements the same functions as the Split-SDK Client, as defined in Section 3.2.

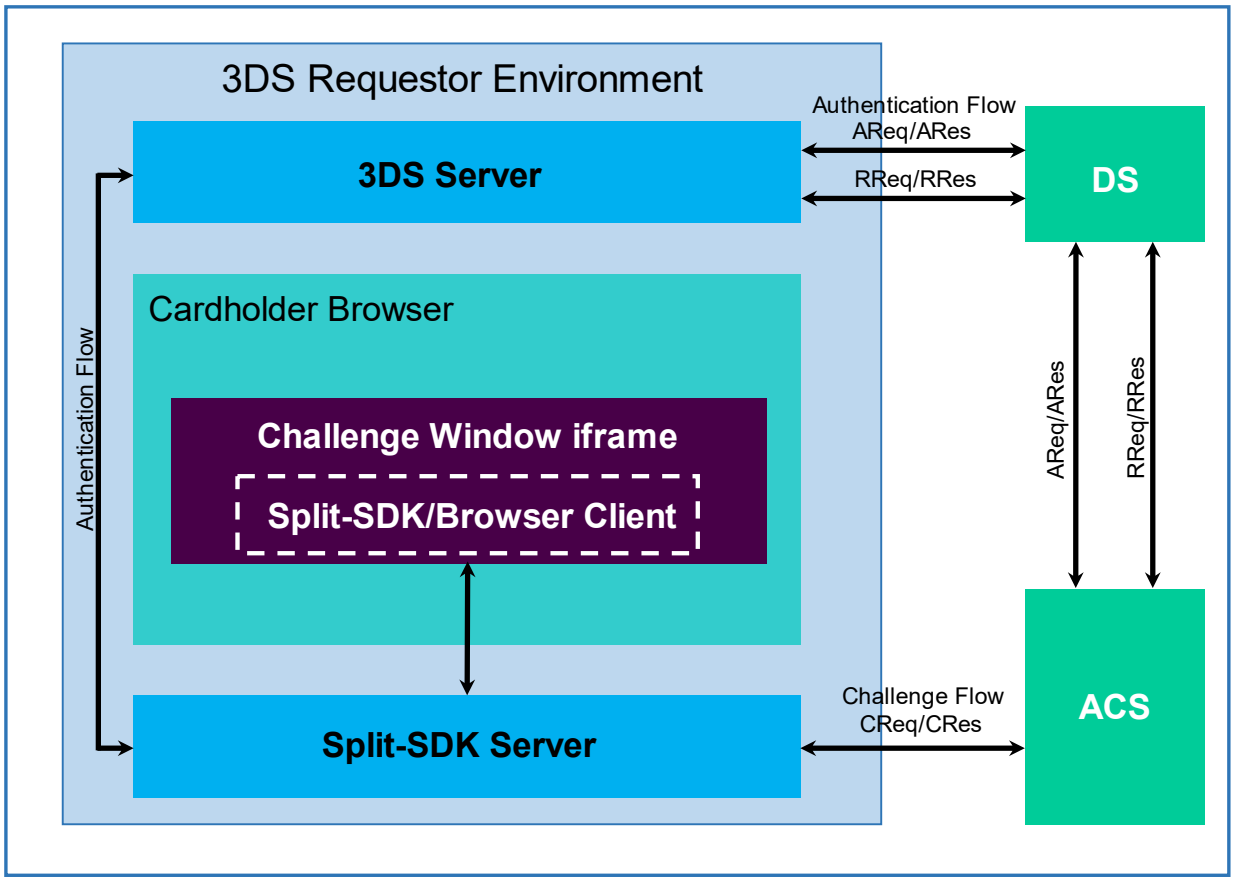
The following sections describe only the differences between the Split-SDK/Browser and the Split-SDK/Native in terms of transaction flow and security.

5.1 Split-SDK/Browser Architecture

Figure 5-1 depicts a typical Split-SDK/Browser component architecture. The architecture is identical to the Split-SDK/Native architecture (Figure 2-1), except that the Client is executed as JavaScript in a Browser.

Note: For the relevant Browser definition, refer to Section 1.5 of the *Core Specification*.

Figure 5-1: Split-SDK/Browser Component Architecture



5.2 Changes to the 3-D Secure Core Protocol Message Flow for a Split-SDK/Browser Flow

The authentication flow for a Split-SDK/Browser follows the same steps as the App-based flow defined in the *Core Specification* v2.3.1.0 and higher, with the exception of the following steps:

Step 1: The Cardholder

The Cardholder interacts with the 3DS Requestor using a Browser on a Consumer Device and confirms the applicable business logic. For example, the Cardholder makes an e-commerce purchase on a Merchant website using a Consumer Device.

Step 2: The 3DS Requestor/3DS Server

Depending on the 3DS Requestor Environment, additional information may be obtained – for example, payment and shopping cart information for Payment Authentication.

The 3DS Server provides the following to the Split-SDK Server through the 3DS Requestor Environment:

- the Directory Server ID value used to identify the key for the Device Information encryption
- the Message Version Number used in the CReq/CRes messages
- the Split-SDK Server URL used by the 3DS Requestor to establish the connection between the Browser and the Split-SDK Server

The 3DS Server shall:

Seq 5.1 ***[Req 800]*** Determine the Directory Server ID and the Message Version Number as defined in ***[Req 11]*** and ***[Req 419]*** of the *Core Specification*.

The 3DS Requestor shall:

Seq 5.2 ***[Req 801]*** Process the 3DS transaction according to the Browser flow as defined in Section 3.3 of the *Core Specification* if JavaScript is not enabled on the Browser.

The 3DS Requestor opens an iframe and redirects the iframe to the Split-SDK Server URL. The Split-SDK Server loads the Split-SDK/Browser Client in the iframe within the Browser, this iframe is used to display the Processing screen and, in case of challenge, the Split-SDK/Browser renders the UI selected by the ACS.

The 3DS Requestor shall:

Seq 5.3 ***[Req 802]*** Select the size of the HTML iframe to be generated by the 3DS Requestor from one of the window sizes specified in the Challenge Window Size data element.

Seq 5.4 ***[Req 803]*** Utilise a server-authenticated TLS session as defined in Section 6.1.4.2 of the *Core Specification*.

Seq 5.5 ***[Req 804]*** Create a 3-D Secure challenge window by generating a message and creating an HTML iframe in the Cardholder Browser with the following settings:

- iframe attributes as defined in Table 5.2

- sandbox attributes as defined in Table 5.3

and generate an HTTPS POST through the iframe to the Split-SDK Server URL.

The Split-SDK Server shall:

Seq 5.6 ***[Req 805]*** Post the HTML code containing the Processing screen and Split-SDK/Browser Client code to the iframe.

The Split-SDK/Browser Client executes and establishes a secure connection with the Split-SDK Server.

The Split-SDK/Browser Client shall:

Seq 5.7 ***[Req 806]*** Verify that it is running within an iframe. If not, then the Split-SDK/Browser Client reports an error to the Split-SDK Server and **ends processing**.

The Split-SDK Server provides to the Split-SDK/Browser Client:

- the Directory Server ID value
- the Message version (obtained from the 3DS Requestor Environment)

Note: After Step 2, the flow is similar to the App-based flow.

Step 9: The 3DS Server

In a Split-SDK/Browser implementation, the 3DS Server supports the same requirements as those defined in Section 3.1, Step 9 of the *Core Specification*, except that the next step is executed within a Browser instead of an App.

Therefore, all the Requirements in Section 3.1, Step 9 of the *Core Specification* are applicable and all references to ***[Req 79]*** are replaced with references to ***[Req 810]*** of this *EMV® 3DS Split-SDK Specification*.

The note at the end of Step 9

"**The next step for a:**

- **Decoupled Authentication transaction is Step 18**"

is replaced with the following wording:

"For a Decoupled Authentication transaction:

Seq 3.55 ***[Req 807]*** The Split-SDK Server and the Split-SDK/Browser Client shall proceed with Step 31 of the Split-SDK Message Flow.

Seq 3.56 ***[Req 808]*** The 3DS Requestor shall close the challenge window.

The next step is Step 18."

Step 24: The 3DS Requestor Environment

The Split-SDK Server shall:

Seq 5.8 ***[Req 809]*** Receive the final CRes message or Error Message from the ACS.

Seq 5.9 ***[Req 810]*** Proceed with Step 31 and Step 32 of the Split-SDK Message Flow Requirements.

The Split-SDK Server closes the challenge process with the Split-SDK Client.

The Split-SDK/Browser Client shall:

Seq 5.10 ***[Req 815]*** Clean up the SDK Ephemeral Key, the CEK used for CReq encryption and the CEK used for CRes decryption.

The Split-SDK Server shall:

Seq 5.11 ***[Req 816]*** Clean up the CEK used for CRes decryption.

The Split-SDK Server conveys the result of the challenge process to the 3DS Requestor.

The 3DS Requestor shall:

Seq 5.12 ***[Req 811]*** Close the challenge window upon receiving the result of the challenge process.

Step 25: The 3DS Requestor

The 3DS Requestor displays the appropriate result to the Cardholder via the Browser.

5.3 Changes to the Split-SDK Message Flow Requirements for a Split-SDK/Browser

Step 8: Assemble AReq Message Data

In Step 8 of this specification, SDK Type = 02 (Split-SDK) and Split-SDK Variant = 02 (Browser) for a transaction as defined in this chapter.

5.4 Split-SDK/Browser Security

This section describes the differences between the Split-SDK/Browser and the Split-SDK for the basic security requirements that are to be implemented by the Split-SDK/Browser.

5.4.1 SDK Initialisation Security Checks

Instead of ***[Req 42]*** and Table 4.1 in Section 4.2 of this specification, the following requirement applies:

The Split-SDK/Browser of the Split-SDK Server shall:

Seq 5.13 ***[Req 812]*** During initialisation, conduct the security checks outlined in Table 5.1 and make the result available as a list of warnings in the Device Information JSON data with key as “SW”.

For additional information, refer to the *EMV® 3-D Secure SDK—Device Information*.

Table 5.1: Split-SDK/Browser Initialisation Security Checks

Security Warning ID	Description	Severity Level
SW06	The development tools are enabled on the browser	MEDIUM

5.4.2 SDK Server CSP and CORS Guidelines

The protection of Split-SDK content (Split-SDK/Browser) from a Merchant is ensured by the iframe used to insulate Split-SDK content from Merchant content.

Secure CSP Guidelines

- Move inline JavaScript to standalone files.
- Create a reporting endpoint able to receive and process reports of resources that are blocked by the CSP.
- Set a CSP that trustlists trusted sources of JavaScript and disables inline JavaScript.
  - In developing the policy, a tool such as Google's CSP Evaluator can be used to check the configuration for security issues.
  - During the testing or transition period, the Content-Security-Policy-Report-Only header can be used instead of Content-Security-Policy. With this header set, browsers will not block resources, but will still send notifications to the specified report URI when a resource would have been blocked by the policy.

Secure CORS Guidelines

- There is no need for cross-domain access between Merchant and Split-SDK domains – they are isolated by the iframe.
- If cross-domain communication is not needed by the Split-SDK content, CORS Access-Control headers should be removed entirely. In many cases, due to the same origin policy, CORS may not be applicable to the environment since the framed content will be sourced from the Split-SDK domains.
- Otherwise, Access-Control headers should be limited to the minimal set of permissions necessary to perform cross-domain functionality:
  - Access-Control headers should only be returned for specific, necessary resources or routes, rather than applied globally.
  - For each route, the relevant Access-Control headers should be explicitly set to allow only the specific methods, headers, or credentials that are necessary.
- In the case of the Access-Control-Allow-Origin header, domains should be specified using a Trust List. Only domains that are necessary for cross-domain communication should be included. This list will need to be generated programmatically – at the time of drafting this document, there is no way to use wildcard notation in this header.

5.4.3 Iframe and Sandbox Attributes

Table 5.2 specifies the iframe attributes used by the 3DS Requestor when creating the Split-SDK/Browser iframe.

Table 5.2: iframe Attributes

Attribute[1]	Value
`allowfullscreen`	false
`allowpaymentrequest`	false
`height`	as per Challenge Window Size
`sandbox`	refer to Table 5.3
`srcdoc`	may be used to initialise redirection content
`width`	as per Challenge Window Size

Table 5.3 specifies the sandbox attributes used by the 3DS Requestor when creating the Split-SDK/Browser iframe.

Table 5.3: Sandbox Attributes

Attribute	Description	Inclusion
`allow-forms`	Allows the processing of forms.	R
`allow-scripts`	Allows the processing of scripts. Note the `allow-scripts` permission does not give the iframe the ability to create pop-ups or modal windows, which can help prevent clickjacking attacks from occurring.	R
`allow-same-origin`	Gives the iframe permission to only use the data from the same ACS domain.	R
`allow-pointer-lock`	Gives access to the mouse position and events.	R
`allow-downloads-without-user-activation`	Prevents downloads to be initiated for content in the iframe without user action.	Not Allowed
`allow-downloads`	Prevents downloads to be initiated for content in the iframe.	Not Allowed
`allow-modals`	Prevents to open modal window from the iframe.	Not Allowed
`allow-orientation-lock`	Prevents to lock the screen orientation.	Not Allowed
`allow-popups`	Prevents popup windows.	Not Allowed

[1] Attributes not listed in Table 5.2 should not be present.

Attribute	Description	Inclusion
`allow-popups-to-escape-sandbox`	Prevents popups to open new windows without inheriting the sandboxing.	Not Allowed
`allow-presentation`	Prevents to initiate a presentation session.	Not Allowed
`allow-storage-access-by-user-activation`	Prevents access to the parent's storage capabilities.	Not Allowed
`allow-top-navigation`	Prevents access to the top-level browsing context.	Not Allowed
`allow-top-navigation-by-user-activation`	Prevents access to the top-level browsing context also with user interaction.	Not Allowed

5.5 Split-SDK/Browser User Interface

The Split-SDK/Browser implements the 3-D Secure User Interface Templates as defined in Section 4 of the *Core Specification* v2.3.1.0.

The Split-SDK/Browser displays the different UI Template zones (refer to Figure 4.1 in the *Core Specification*) – in particular, the Header zone with the Cancel action button and the Header text.

6 Split-SDK/Shell Requirements

The Split-SDK/Shell variant implements the same functions as the Split-SDK Client, as defined in Section 3.2.

The following sections describe only the differences between the Split-SDK/Shell and the Split-SDK/Native in terms of transaction flow and security.

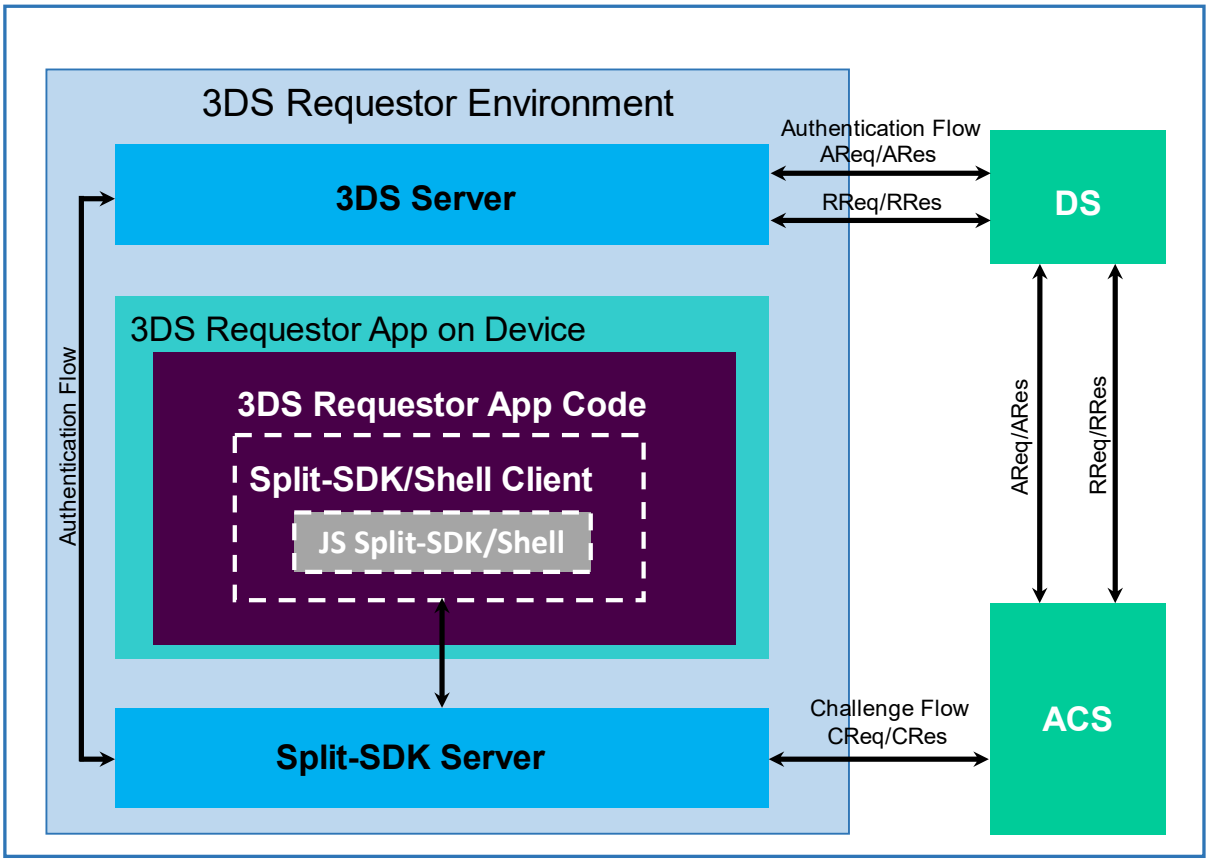
6.1 Split-SDK/Shell Architecture

Figure 6-1 depicts a typical Split-SDK/Shell component architecture.

The Split-SDK/Shell component architecture is identical to the Split-SDK architecture (Figure 2-1), except that the SDK Client utilises a wrapped WebView component for both App-based Native and App-based HTML Challenge Flows.

Unlike the Default-SDK, which renders the Native UI using platform-specific display elements (i.e., button, textbox, text label, etc.), the Split-SDK/Shell renders the Native UI challenges using a JavaScript (JS Split-SDK/Shell) running in a WebView.

Figure 6-1: Split-SDK/Shell Component Architecture



6.2 Changes to the 3-D Secure Core Protocol User Interface Requirements and Guidelines for a Split-SDK/Shell Flow

The following items replace the requirements for the App-based User Interface in the *Core Specification* v2.3.1.0 and higher.

6.2.1 Processing Screen Requirements

The 3DS Requestor App or the 3DS Split-SDK/Shell shall, for the AReq/ARes message exchange:

Seq 6.1 ***[Req 822]*** Display the Processing screen supplied by the 3DS SDK during the entire AReq/ARes message cycle.

Seq 6.2 ***[Req 823]*** Display the Processing screen for a minimum of two seconds.

6.3 Changes to the Split-SDK Message Flow Requirements for a Split-SDK/Shell

The Split-SDK/Shell Server and Client mutually authenticate before the Client loads the JS Split-SDK/Shell. When loaded and running, the JS Split-SDK/Shell mutually authenticates with the Split-SDK Server.

A Split-SDK/Shell authentication flow follows the same steps as the Split-SDK/Native flow defined in this document, with the exception of the following steps:

Step 2 Get SDK Ephemeral Public Key

Seq 6.3 ***[Req 48]*** The Split-SDK/Shell shall:

- Implement a mutual authentication protocol and an encryption protocol that protects the data exchanged between the Split-SDK/Shell Client and the Split-SDK Server.
- Retrieve the information related to the transaction, the HTML for the Processing screen, and the JavaScript of the JS Split-SDK/Shell from the Split-SDK Server.

Seq 6.4 ***[Req 49]*** The Split-SDK/Shell shall:

- Initialise for a new 3DS transaction
- Create and protect a WebView
- Post the HTML and JavaScript in WebView

Seq 6.5 ***[Req 50]*** The JS Split-SDK/Shell shall:

- Implement a mutual authentication protocol and an encryption protocol that protects the data exchanged between the JS Split-SDK/Shell and the Split-SDK Server.

- Send the SDK Ephemeral Public Key to the SDK Server (which could occur through Split-SDK/Shell or directly through a secured connection to the Split-SDK Server).

Step 8: Assemble AReq Message Data

In Step 8 of this specification, SDK Type = 02 (Split-SDK) and Split-SDK Variant = 03 (Shell) for a transaction as defined in this chapter.

Step 11b Close (Frictionless only)

The JS Split-SDK/Shell shall:

Seq 6.6 ***[Req 818]*** Clean up the SDK Ephemeral Key.

The Split-SDK/Shell shall:

Seq 6.7 ***[Req 51]*** Close the WebView.

The Split-SDK Server shall:

Seq 6.8 ***[Req 819]*** Clean up the CEK used for CRes decryption.

Step 31 Close

The Split-SDK Server closes the challenge process with the Split-SDK/Shell.

The JS Split-SDK/Shell shall:

Seq 6.9 ***[Req 820]*** Clean up the SDK Ephemeral Key, the CEK used for CReq encryption and the CEK used for CRes decryption.

The Split-SDK/Shell shall:

Seq 6.10 ***[Req 52]*** Close the WebView.

The Split-SDK Server shall:

Seq 6.11 ***[Req 821]*** Clean up the CEK used for CRes decryption.

6.4 SDK Security

This section describes the differences between the Split-SDK/Shell and the Split-SDK for the basic security requirements that are to be implemented by the Split-SDK/Shell.

6.4.1 SDK Initialisation Security Checks

The Split-SDK/Shell of the Split-SDK Server shall conduct the security checks as defined in ***[Req 42]*** and Table 4.1 in Section 4.2 of this specification.

6.4.2 Split-SDK/Shell Security

The protection of Split-SDK content (Split-SDK/Shell) from a Merchant is ensured by wrapping the WebView component to prevent direct access by the Merchant.

Additionally, the Split-SDK/Shell Client should only allow the WebView component to run scripts and load other external resources (e.g., CSS files) that have been loaded from the Split-SDK Server or from within the Split-SDK/Shell package itself.

Seq 6.12 ***[Req 53]*** The Split-SDK/Shell shall set a WebView that:

- allows the execution of the JS Split-SDK/Shell
- prevents the 3DS Requestor App or other external application from accessing the content of the WebView.

Seq 6.13 ***[Req 54]*** The Split-SDK/Shell shall:

- load and run the JS Split-SDK/Shell provided by the Split-SDK Server.
- not accept other external sources for the JS Split-SDK/Shell package, or any other codes (HTML, CSS).

Navigation attempts from within the Split-SDK/Shell WebView are captured by the JS Split-SDK/Shell and processed internally, rather than being passed to the operating system and network stack. In addition to navigation attempts, the 3DS SDK also captures external resource requests (image loads, external JS scripts, CSS, etc.).

For the App-based Native flow, the Split-SDK/Shell should allow loading of image URLs (Issuer Image, Payment System Image) coming from an external source.

Seq 6.14 ***[Req 817]*** If the Split-SDK/Shell uses an iframe in the WebView, it shall use the setting defined in Table 5.2 to open the iframe.

Requirements

*** END OF DOCUMENT ***