EMV®
3-D Secure

SDK Specification

Version 2.1.0
October 2017

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.

Revision Log—Version 2.1.0

The following changes have been made to the document since the publication of Version 2.0.0. The numbering of existing requirements has not changed.

Chapter 1:

- Updated the *EMV 3DS Protocol Specification* version number to 2.1.0.
- Updated the Normative References table.
- Removed the Definitions table and added a reference to the Definitions table present in *EMV 3DS Protocol Specification*.
- Added the Supporting Documentation section.

Chapter 3:

- Updated the description of the 3DS SDK Lifecycle Phases.
- Updated Req 7, Req 10 and Req 23.
- Modified the Challenge Flow diagram.
- Added Req 66 and Req 67.

Chapter 4:

- Updated the description of the initialize method in the ThreeDS2Service Interface.
- For the createTransaction method in the ThreeDS2Service Interface:
  - Updated the description.
  - Added an optional parameter called messageVersion.
  - Updated the description of InvalidInputException.
- For the doChallenge method in the Transaction Interface:
  - Updated the description.
  - Updated the description of the timeout parameter to include the minimum timeout interval.
  - Updated the description of InvalidInputException
- Updated the description of the close method in the Transaction Interface.
- Updated the description of setAcsSignedContent in the ChallengeParameters class.
- In class AuthenticationRequestParameters:
  - Removed reference to ACSRenderingType.
  - Added a new method called getMessageVersion.
  - Updated the description of getSDKEphemeralPublicKey
- Updated the description of ErrorMessage and ProtocolErrorEvent classes.

Chapter 5:

- Updated section 5.1 to include Message Version data element

Chapter 7:

- Updated Req 34, Req 35, Req 36 and Req 41.
- Added Req 64 and Req 65

Chapter 8:

- Updated the overall security-related information in the chapter.
- Deleted Req 43 to Req 57.
- Updated Req 59.
- Added Req 68.

Annex D

- Updated code samples to include messageVersion

Other editorial changes:
Miscellaneous wording to clarify original intent and verbiage consistency.

Contents

Figures

Tables

1 Introduction

The past few years have seen a dramatic rise in the use of mobile devices. A growing number of consumers now purchase products and log in to numerous online services through mobile apps. There is a need to improve authentication security in mobile-based apps.

The 3-D Secure protocol is aimed at securing authentication in both browser-based apps and mobile-based apps.

The mobile-device-side component of 3-D Secure is the 3DS Mobile SDK (later referred to as 3DS SDK in this document). 3-D Secure Requestors, such as Merchants, integrate this SDK with their mobile app and make the app available to end users.

1.1 Purpose

This document describes the specification for the 3DS SDK. Enhancements to the *EMV 3-D Secure Protocol and Core Functions Specification version 2.1.0* (later referred to as *EMV 3DS Protocol Specification* in this document) that have an impact on the SDK will be included in later versions of this document.

For purposes of this document, when the phrase 3-D Secure, and/or 3DS is utilised, the intent is EMV 3-D Secure.

1.2 Audience

This document is intended for use by implementers who want to develop a 3DS Mobile 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 7515	*The JavaScript Object Notation (JSON) Data Interchange Format*	https://tools.ietf.org/html/rfc7515
RFC 7515	*JSON Web Signatures (JWS)*	https://tools.ietf.org/html/rfc7515

Reference	Publication Name	Bookmark
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/rfc7517

1.4 Definitions

For the definition of the terms used in this specification, refer to Table 1.3: Definitions in the *EMV 3-D Secire Protocol and Core Functions Specification.*

1.5 Abbreviations

The abbreviations listed in Table 1.2 are used in this specification.

Table 1.2: Abbreviations

Abbreviation	Description
3DS	Three Domain Secure
3DS SDK	Three Domain Secure Software Development Kit
ACS	Access Control Server
AReq	Authentication Request
ARes	Authentication Response
CA	Certificate Authority
CA DS	Certificate Authority Directory Server
CReq	Challenge Request
CRes	Challenge Response
DS	Directory Server
JSON	JavaScript Object Notation

Abbreviation	Description
JWE	JSON Web Encryption
MAC	Message Authentication Code
OOB	Out-of-Band
OTP	One-time Passcode
RReq	Results Request Message
RRes	Results Response Message
SDK	Software Development Kit
URL	Uniform Resource Locator
UUID	Universally Unique Identifier

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 EMV 3-D Secure FAQs 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 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.

Ends 3-D Secure Processing

As outlined in Chapter 3 in the *EMV 3DS Protocol 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 *EMV 3DS Protocol Specification* for additional information.

Ends Processing

As outlined in Chapter 3 in the *EMV 3DS Protocol 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 *EMV 3DS Protocol Specification* for additional information.

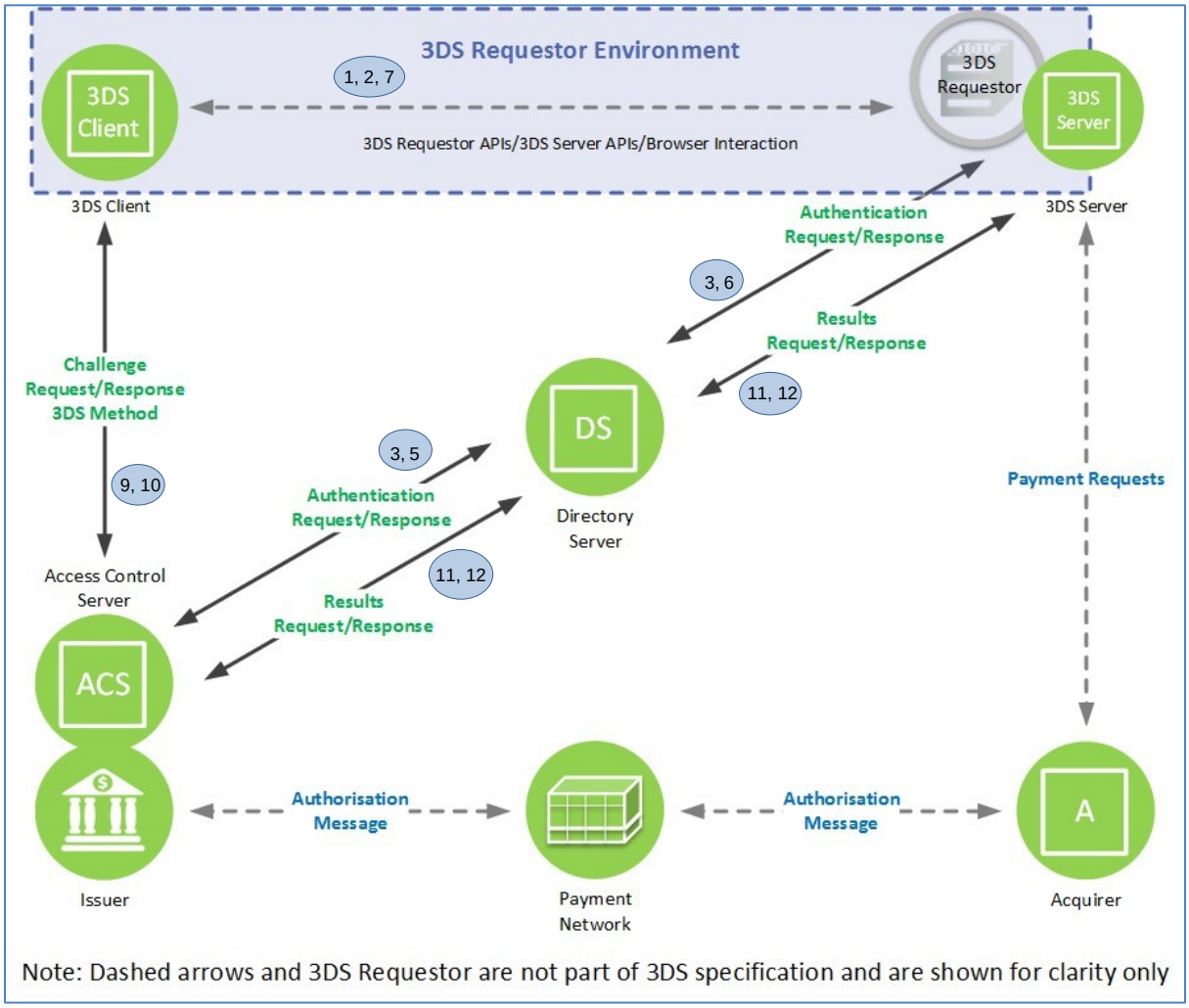
2 Overview of EMV 3-D Secure Architecture and Flows

This chapter provides an overview of the components in the 3-D Secure architecture and an introduction to the 3-D Secure authentication flows. For detailed information about these components and authentication flows, see Chapter 2, "EMV 3-D Secure Overview" in the *EMV 3DS Protocol Specification*.

2.1 Participating Components

Figure 2-1 illustrates the interaction of the components in the EMV 3-D Secure ecosystem.

Figure 2-1: EMV 3-D Secure Component Interaction



2.2 Authentication Flows—Overview

This section provides an overview of the following two 3-D Secure authentication flows:
- Frictionless Flow
- Challenge Flow

2.2.1 Frictionless Flow

The Frictionless Flow initiates a 3-D Secure authentication flow and consists of an AReq message and an ARes message.

The Frictionless Flow does not require further Cardholder interaction to achieve a successful authentication and complete the 3-D Secure authentication process.

The following steps provide a high-level view of the Frictionless Flow:

Start: The Cardholder initiates a transaction using a 3DS Requestor App on a Consumer Device.
1. The 3DS SDK collects the device information and provides it to the 3DS Requestor App.
2. The 3DS Requestor App initiates communication with the 3DS Server and provides the information that is required to create an AReq.
3. The 3DS Server creates and sends an AReq message to the DS. The DS then forwards the message to the appropriate ACS.
4. The ACS evaluates the payment, Cardholder, and device authentication data provided in the message.
5. If the ACS determines that the transaction does not require additional authentication, then the ARes message that it returns to the DS indicates that the Frictionless Flow must be applied.
6. The DS forwards the message to the 3DS Server.
7. The 3DS Server communicates the result of the ARes to the 3DS Requestor App.
   Note: 3-D Secure processing ends here. For Payment Authorisation, the subsequent steps apply:
8. The Merchant proceeds with authorisation exchange with its Acquirer. If appropriate, the Merchant, Acquirer, or Payment Processor can submit a standard authorisation request.
9. The Acquirer can process an authorisation with the Issuer through the Payment System and return the authorisation results to the Merchant.

2.2.2 Challenge Flow

In addition to the AReq and ARes messages that comprise the Frictionless Flow, the Challenge Flow consists of CReq, CRes, RReq, and RRes messages.

If the ACS determines that further Cardholder interaction is required to complete the authentication, the Frictionless Flow transitions into the Challenge Flow. For example, a challenge may be necessary because the transaction is deemed high-risk, is above certain thresholds, or requires a higher level of authentication due to country mandates (or regulations).

3DS Requestors decide whether to proceed with the challenge, or to terminate the 3-D Secure authentication process.

The following steps provide a high-level view of the Challenge Flow:

Start: The Cardholder initiates a transaction using a 3DS Requestor App on a Consumer Device.

1. The 3DS SDK collects the device information and provides it to the 3DS Requestor App.
2. The 3DS Requestor App initiates communication with the 3DS Server and provides the information that is required to create an AReq.
3. The 3DS Server creates and sends an AReq message to the DS. The DS then forwards the message to the ACS.
4. The ACS evaluates the payment, Cardholder, and device authentication data provided in the message.
5. If the ACS determines that the transaction requires additional authentication, then the ARes message that it returns to the DS indicates that the Challenge Flow must be applied.
6. The DS forwards the message to the 3DS Server.
7. The 3DS Server returns the authentication status to the 3DS Requestor App.
8. The 3DS Requestor App invokes the 3DS SDK to perform Cardholder authentication.
9. The 3DS SDK sends a CReq message directly to the ACS.
10. The ACS receives the CReq message and returns a CRes message to the 3DS SDK.
    Note: Based on the CRes obtained from the ACS, the 3DS SDK displays the challenge-specific screens for the Cardholder to enter their authentication credentials. Steps 9 and 10 are repeated until the ACS has determined the outcome of the authentication.
11. The ACS sends an RReq message to the DS, which then forwards the message to the 3DS Server.
12. The 3DS Server receives an RReq message and returns an RRes message to the DS, which then forwards the message to the ACS.
13. The ACS sends the final CRes message to the 3DS SDK with the outcome of the authentication.
    Note: 3-D Secure processing ends here. For Payment Authorisation, the subsequent steps apply:
14. The Merchant proceeds with authorisation exchange with its Acquirer. If appropriate, the Merchant, Acquirer, or Payment Processor can submit a standard authorisation request.
15. The Acquirer can process an authorisation with the Issuer through the Payment System and return the authorisation results to the Merchant.

2.3 UI Types for Challenge Flow

The UI for the Challenge Flow can be rendered in one of the following formats:

- Native UI
- HTML UI

2.3.1 Challenge Flow Implemented Using Native UI

The Native UI integrates into the 3DS Requestor App UI to facilitate a consistent user experience. The Native UI has a similar look and feel as the 3DS Requestor's App with the authentication content provided by the Issuer.

This format also allows for Issuer and Payment System branding. Both the 3DS Requestor App and the 3DS SDK control the rendering of the UI such that the authentication pages inherit the 3DS Requestor's UI design elements. For more information about the Native UI, refer to Section 4.2.2, "Native UI Templates" in the *EMV 3DS Protocol Specification.*

2.3.2 Challenge Flow Implemented Using HTML UI

The HTML UI provides Cardholders with an Issuer-consistent App-based experience across Consumer Devices that are able to render HTML. The HTML UI templates provides Issuers the ability to include Issuer-specific design elements (for example, branding, colours, and/or fonts).

The HTML UI implementation establishes a client-server relationship between the ACS-provided HTML document loaded in a 3DS Requestor's web view and the SDK process itself. This is accomplished by intercepting remote URL requests issued by the web view, and handling them within the SDK, rather than allowing them to pass through to the Consumer Device operating system and hence on to the Internet. This has two effects:

- Prevents maliciously formed HTML within the web view flow from requesting external resources or redirecting to an external malicious site (for example, a phishing page).
- Changes the web view form into an extension of the SDK's UI, one that's defined by the remote ACS using HTML, rather than by the SDK or 3DS Requestor's App.

For more information about the HTML UI, refer to Section 4.2.4, "HTML UI Templates" in the *EMV 3DS Protocol Specification.*

2.3.3 Challenge Flow Implemented Using Out-of-Band (OOB) UI

The Out-of-Band (OOB) user interface allows Issuers to utilise authentication methods other than dynamic and static data such as an Issuer's mobile app. When an OOB challenge is necessary, the Issuer/ACS provides instructions to the Cardholder to explain the authentication process.

3 Getting Started with the EMV 3-D Secure Mobile SDK

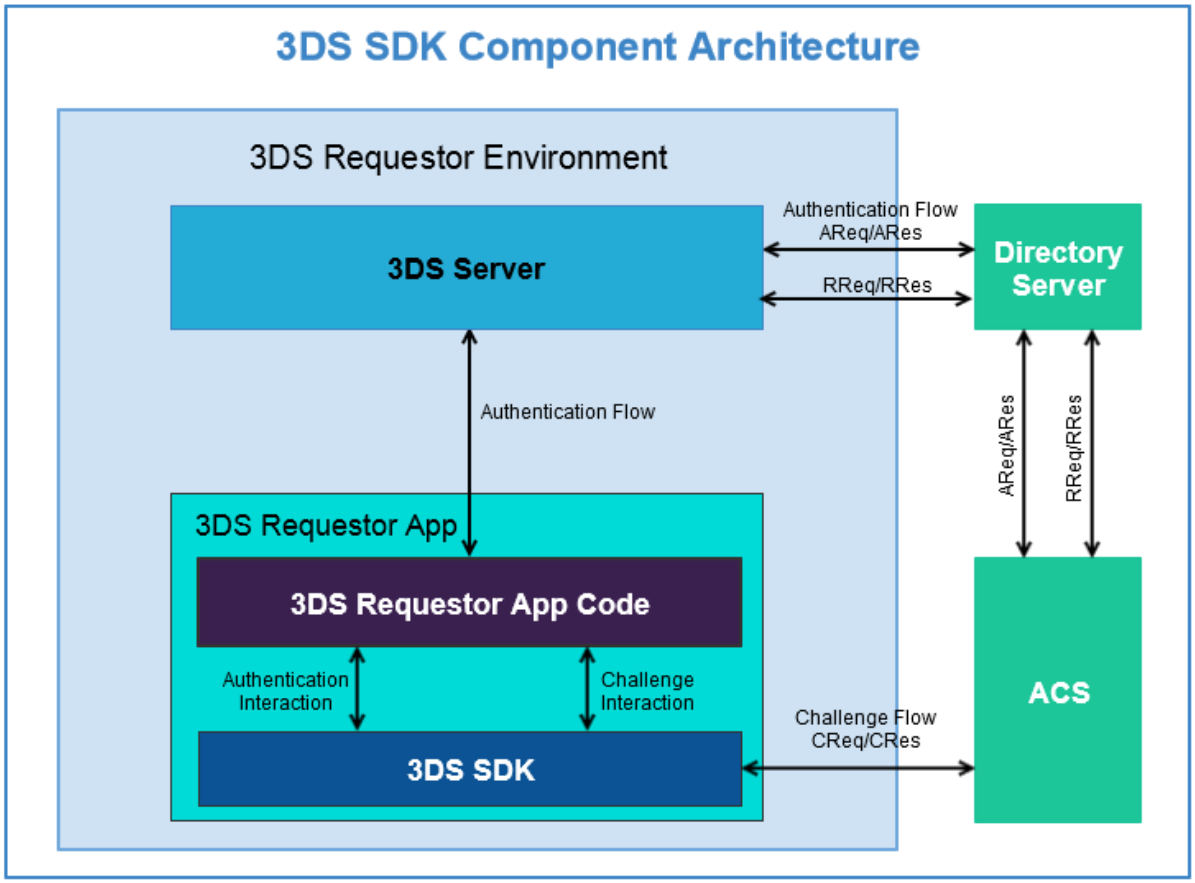
The EMV 3-D Secure Mobile SDK (3DS SDK) is a client-side component of the 3-D Secure ecosystem. When a Cardholder initiates an in-app transaction, the 3DS SDK integrated in the 3DS Requestor App performs operations related to 3-D Secure authentication.

This chapter provides an overview of the 3DS SDK components, lifecycle and flows.

3.1 Component Architecture

Figure 3-1 shows the 3DS SDK component architecture.

Figure 3-1: 3DS SDK Component Architecture



1. The 3DS Requestor App collects the parameters that are required for authentication from the 3DS SDK and initiates the authentication flow.
2. If the authentication flow indicates that no challenge is required, then the Frictionless Flow is applied.
   If the authentication flow indicates that a challenge is required, then the 3DS Requestor App invokes the 3DS SDK to apply the Challenge Flow.
3. The 3DS SDK performs the following steps:

- Communicate with the ACS to initiate the Challenge Flow.
- Display the challenge UI to the Cardholder.
- Collect the Cardholder's challenge response.
- Complete the Challenge Flow.
- Return the challenge response to the 3DS Requestor App.

3.2 Lifecycle

Figure 3-2 provides a high-level view of the lifecycle of the 3DS SDK.

Figure 3-2: High-Level 3DS SDK Lifecycle

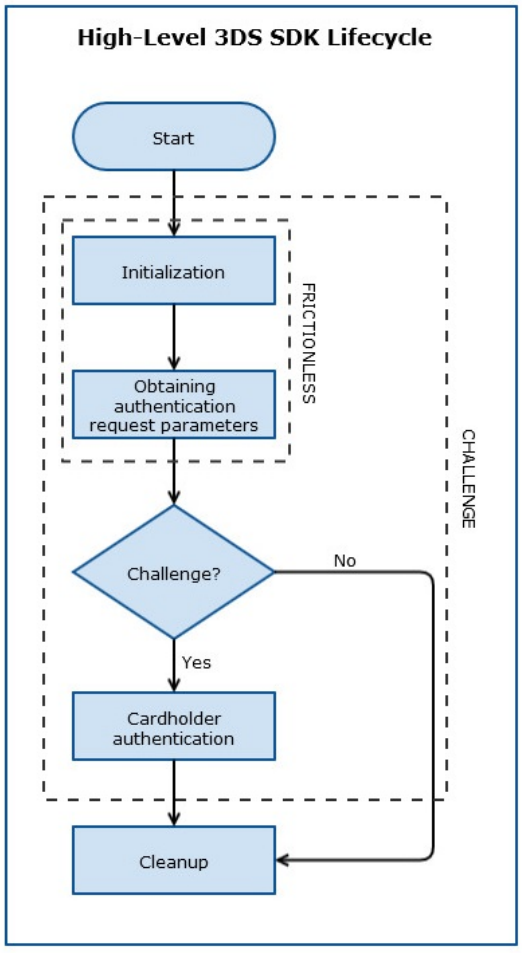


Table 3.1 describes each phase in the lifecycle.

Table 3.1: 3DS SDK Lifecycle Phases

Phase	Description
Initialization	The initialization phase shall take place either during 3DS Requestor App startup as a background task or when a transaction is initiated. In this phase, the SDK shall collect device information against the protocol versions that it supports and perform security checks. This phase shall take place only once during a single 3DS Requestor App session.
Obtaining authentication request parameters	The 3DS SDK shall encrypt the device information, adhering to the protocol version for the transaction, that it collects during initialization and send this information along with the SDK information to the 3DS Requestor App.
Cardholder authentication	If a challenge is required, then the 3DS SDK shall perform cardholder authentication.
Cleanup	The cleanup phase is called only once during a single 3DS Requestor App session to free up resources that are used by the 3DS SDK.

3.3 Authentication Flows

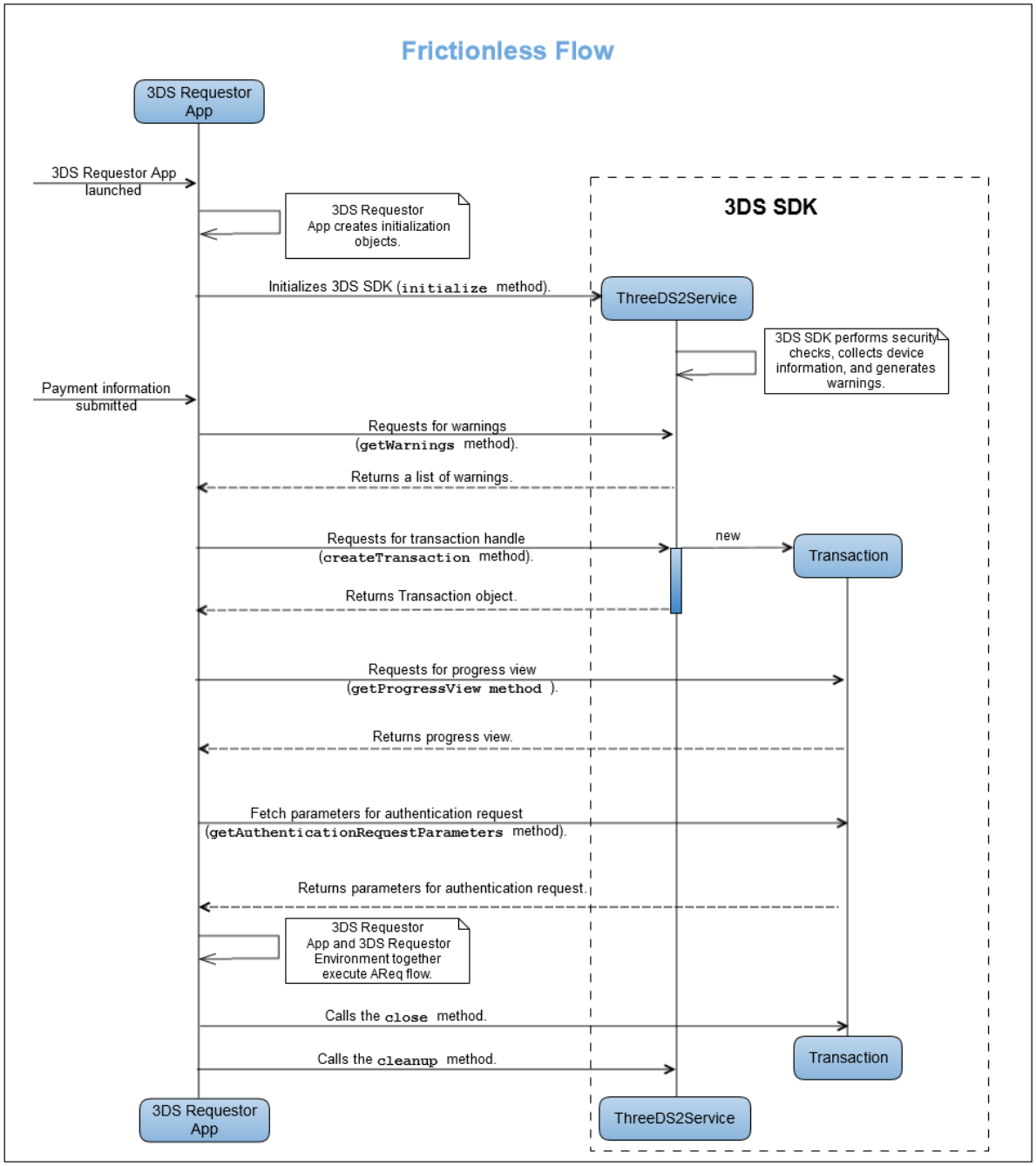
The following sections show the interaction between the 3DS Requestor App and the 3DS SDK code elements during the Frictionless Flow and the Challenge Flow.

Note: In these sections, a diagram showing the flow is followed by a sequence of steps that provide a more detailed description of the flow. If the steps are not in agreement with the diagram at any point, then the steps take precedence over the diagram.

3.3.1 Frictionless Flow

Figure 3-3 shows the interaction between the 3DS Requestor App and the 3DS SDK code elements during the Frictionless Flow.

Figure 3-3: Frictionless Flow



The following steps summarize the events that take place during the Frictionless Flow:

1. The Cardholder launches the 3DS Requestor App.
2. The 3DS Requestor App creates instances of `ConfigParameters`, `locale`, and `UiCustomization` for initialization.
3. The 3DS Requestor App calls the `initialize` method to initialize the 3DS SDK either during App startup as a background task or when a transaction is initiated.
   Note: This method is called only once during a single 3DS Requestor App session.
4. In the `initialize` method call, the 3DS SDK shall:
   [Req 1] Perform security checks and generate warnings for each security check that fails.
   [Req 2] Collect device information.

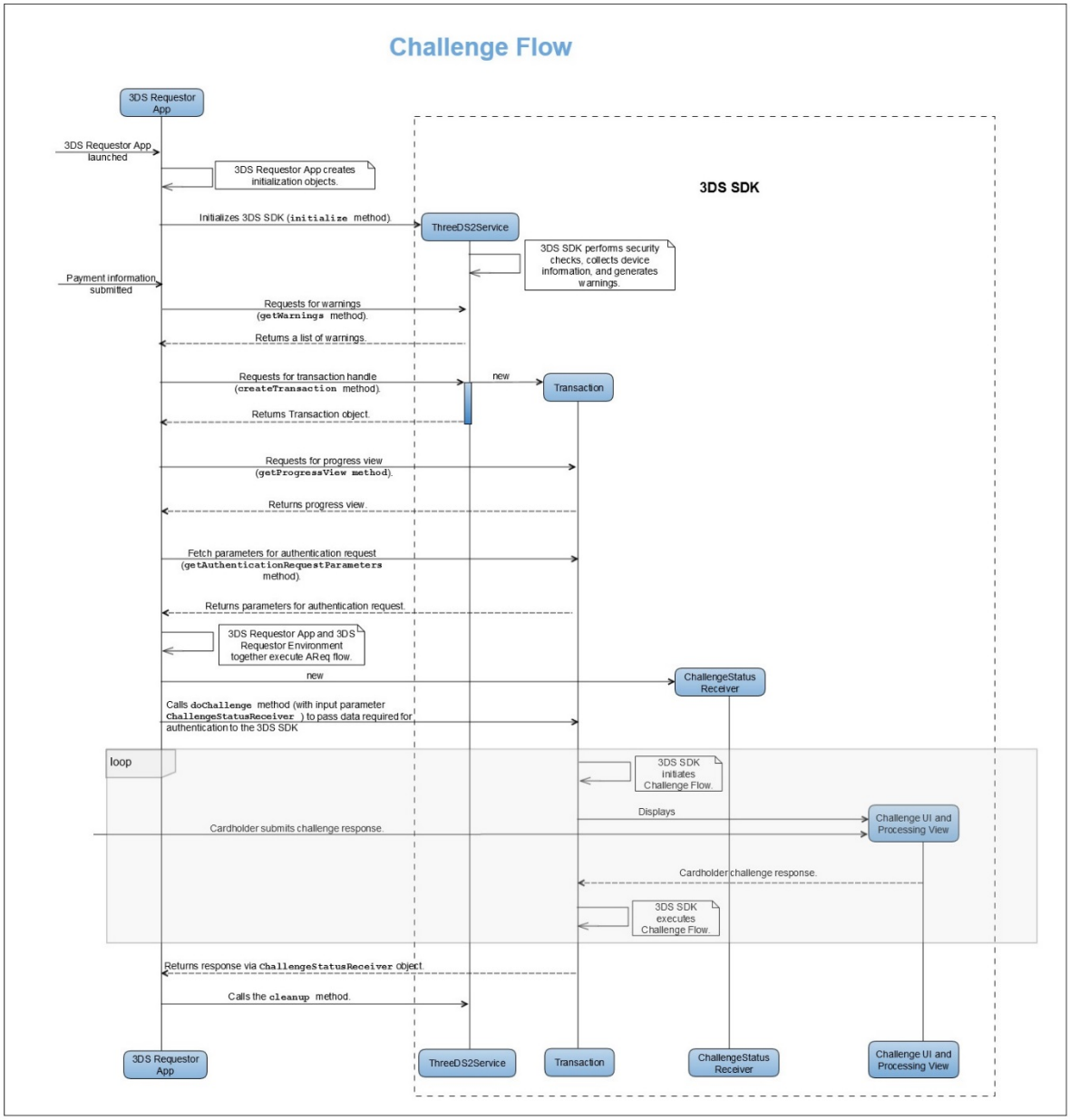
Note: Steps 5 to 18 shall be performed per transaction. There can be multiple transactions in a single 3DS Requestor App session.

5. The Cardholder submits payment information by using the 3DS Requestor App.
6. (Optional) The 3DS Requestor App calls the `getWarnings` method.
7. In the `getWarnings` method call, the 3DS SDK shall:
   [Req 3] Return a List of warnings produced by the 3DS SDK during initialization.
8. (Optional) The 3DS Requestor App may call the `getSDKVersion`.
9. In the `getSDKVersion` method call, the 3DS SDK shall:
   [Req 4] Return the version of the 3DS SDK that is integrated with the 3DS Requestor App.
10. The 3DS Requestor App calls the `createTransaction` method.
11. In the `createTransaction` method call, the 3DS SDK shall:
    [Req 5] Create and return an instance of the `Transaction` interface implementation.
12. The 3DS Requestor App calls the `getProgressView` method.
13. In the `getProgressView` method call, the 3DS SDK shall:
    [Req 6] Return an instance of Progress View (processing screen). The progress view shows the Cardholder that an activity is being processed. The 3DS SDK shall create the Progress View object and return a handle of this object to the app.
14. The 3DS Requestor App calls the `getAuthenticationRequestParameters` method.
15. In the `getAuthenticationRequestParameters` method call, the 3DS SDK shall:
    [Req 7] Return the device information and 3DS SDK information, such as SDK transaction ID, Ephemeral public key, protocol version used and so on.
16. The 3DS Requestor App and the 3DS Requestor Environment together execute the AReq flow.
17. The ARes that is returned indicates that the Frictionless Flow must be applied. Therefore, no further action is required.
18. The 3DS Requestor App calls the `close` method to allow the 3DS SDK to clean up resources that are held by the `Transaction` object.
19. The 3DS Requestor App calls the `cleanup` method to allow the 3DS SDK to free up resources that were used by it.
    Note: The `cleanup` method shall be called only once during a 3DS Requestor App session.

3.3.2 Challenge Flow

Figure 3-4 shows the interaction between the 3DS Requestor App and the 3DS SDK code elements during the Challenge Flow.

Figure 3-4: Challenge Flow



The following steps summarize the events that take place during the Challenge Flow:
Note: Steps 1 to 16 are the same as the steps in the Frictionless Flow.

17. The ARes that is returned indicates that the Challenge Flow must be applied.
18. The 3DS Requestor App creates a callback object. This object implements the `ChallengeStatusReceiver` interface to receive challenge status notification from the 3DS SDK at the end of the challenge process.
19. The 3DS Requestor App calls the `doChallenge` method. One of the parameters of the `doChallenge` method is `ChallengeStatusReceiver`. The 3DS Requestor App passes the callback object created as part of Step 18 using this parameter to the 3DS SDK. Another parameter that is passed by the 3DS Requestor App is a timeout value (in minutes) for the Challenge process.
20. In the `doChallenge` method call, the 3DS SDK shall:
    [Req 66] Start a time counter to measure the time taken by the challenge process.

Note: Within a single doChallenge method call, steps 21 to 23 shall be performed for each CReq/CRes exchange.

21. The 3DS SDK shall:
    [Req 8] Initiate the Challenge Flow by displaying the UI for the challenge screens.
22. The Cardholder responds to the challenge.
23. The 3DS SDK shall:
    [Req 9] Use a graphical element (a processing view) on the Challenge screen to show that the Cardholder's response is being processed.
24. The 3DS SDK shall:
    [Req 10] Call one of the methods (`completed`, `cancelled`, `protocolError` or `runtimeError`) of the `ChallengeStatusReceiver` callback object to return the result of the challenge process to the 3DS Requestor App and clean up resources that are held by the `Transaction` object.
    [Req 67] If a timeout occurs at any point, that is, if the time taken by the challenge process as measured by the time counter (refer Step 20) exceeds the timeout value passed by the 3DS Requestor App (refer Step 19), call the `timedout` method of the `ChallengeStatusReceiver` callback object and clean up resources that are held by the `Transaction` object.

Note: The last step in this flow is the cleanup step which is the same as step 19 in the Frictionless Flow.

3.4 Summary of 3DS SDK Code Elements

The following tables provide a summary of the code elements that shall be included in the 3DS SDK package.

Interface Summary

Table 3.2 summarizes the interfaces that shall be included in the 3DS SDK package.

Table 3.2: Interfaces

Requirement ID	Interface	Description
[Req 11]	`ThreeDS2Service`	This interface shall provide methods to process 3-D Secure transactions. For detailed information, see `Interface ThreeDS2Service`.
[Req 12]	`ChallengeStatusReceiver`	This interface shall provide methods to receive challenge status notifications from the 3DS SDK. For detailed information, see `Interface ChallengeStatusReceiver`.

Requirement ID	Interface	Description
[Req 13]	`Transaction`	An object that implements the Transaction interface shall hold parameters that are required to create AReq messages and to perform the Challenge Flow. For detailed information, see `Interface Transaction`.

Class Summary

Table 3.3 summarizes the classes that shall be included in the 3DS SDK package.

Table 3.3: Classes

Requirement ID	Class	Description
[Req 14]	`ConfigParameters`	This class shall represent the configuration parameters that are required by the 3DS SDK for initialization. For detailed information, see `Class ConfigParameters`.
[Req 15]	`ChallengeParameters`	This class shall hold the data that is required to conduct a challenge process. For detailed information, see `Class ChallengeParameters`.
[Req 16]	`AuthenticationRequestParameters`	This class shall hold transaction data that the 3DS Server requires to create the AReq. For detailed information, see `Class AuthenticationRequestParameters`.
[Req 17]	`UiCustomization`	This class shall provide the functionality required for 3DS SDK UI customization. For detailed information, see `Class UiCustomization`.

Requirement ID	Class	Description
[Req 18]	`Customization`	This class shall serve as a superclass for the `ButtonCustomization` class, `ToolbarCustomization` class, `LabelCustomization` class, and `TextBoxCustomization` class. This class shall provide methods to pass UI customization parameters to the 3DS SDK. For detailed information, see `Class Customization`.
[Req 19]	`ButtonCustomization`	This class shall provide methods for the 3DS Requestor App to pass button customization parameters to the 3DS SDK. For detailed information, see `Class ButtonCustomization`.
[Req 20]	`ToolbarCustomization`	This class shall provide methods for the 3DS Requestor App to pass toolbar customization parameters to the 3DS SDK. For detailed information, see `Class ToolbarCustomization.`
[Req 21]	`LabelCustomization`	This class shall provide methods for the 3DS Requestor App to pass label customization parameters to the 3DS SDK. For detailed information, see `Class LabelCustomization`.
[Req 22]	`TextBoxCustomization`	This class shall provide methods for the 3DS Requestor App to pass text box customization parameters to the 3DS SDK. For detailed information, see `Class TextBoxCustomization`.

Requirement ID	Class	Description
[Req 23]	`ErrorMessage`	This class shall represent an error message that is returned by the ACS to the 3DS SDK or an error message that is generated by the 3DS SDK to be returned to the ACS. For detailed information, see `Class ErrorMessage`.
[Req 24]	`CompletionEvent`	This class shall represent an event that indicates that the challenge process has been completed. For detailed information, see `Class CompletionEvent`.
[Req 25]	`RuntimeErrorEvent`	This class shall represent a run-time error that is encountered during the authentication process. For detailed information, see `Class RuntimeErrorEvent`.
[Req 26]	`ProtocolErrorEvent`	This class shall represent an EMV 3-D Secure protocol-defined error that is returned by the ACS. For detailed information, see `Class ProtocolErrorEvent`.
[Req 27]	`Warning`	This class shall represent a warning produced by the 3DS SDK while performing security checks during initialization. For detailed information, see `Class Warning`.

Exception Summary

Table 3.4 summarizes the exceptions that shall be included in the 3DS SDK package.

Table 3.4: Exceptions

Requirement ID	Exception	Description
[Req 28]	`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For detailed information, see `Class InvalidInputException`.
[Req 29]	`SDKAlreadyInitializedException`	This exception shall be thrown if the 3DS SDK instance has already been initialized. For detailed information, see `Class SDKAlreadyInitializedException`.
[Req 30]	`SDKNotInitializedException`	This exception shall be thrown if the 3DS SDK instance has not been initialized. For detailed information, see `Class SDKNotInitializedException`.
[Req 31]	`SDKRuntimeException`	This exception shall be thrown if an internal error is encountered by the 3DS SDK. For detailed information, see `Class SDKRuntimeException.`

Enum Summary

Table 3.5 summarizes the enum that shall be included in the 3DS SDK package.

Table 3.5: Enum

Requirement ID	Enum	Description
[Req 32]	`Severity`	This enum shall define severity levels of warnings produced by the 3DS SDK. For detailed information, see `Enum Severity`.
[Req 33]	`ButtonType`	This enum shall define the button type. For detailed information, see `Enum ButtonType`.

4 Code Elements of the EMV 3-D Secure Mobile SDK

The 3DS SDK package shall contain code elements that describe and define the contracts between the 3DS Requestor App and the 3DS SDK. This chapter provides detailed information about these code elements.

Note: The information in this chapter is not intended to be specific to any platform or programming language. However, for instructional purposes, Java-based and Android-based code samples have been used to illustrate how to use this information. These code samples can be adapted and used on any mobile platform or programming language.

4.1 Interface ThreeDS2Service

The `ThreeDS2Service` interface is the main 3DS SDK interface. It shall provide methods to process transactions.
The following Java code snippet shows the definition of the `ThreeDS2Service` interface:

```
public interface ThreeDS2Service {

    public void initialize(...)

    public Transaction createTransaction (...)

    public void cleanup(...)

    public String getSDKVersion(...)

    public List<Warning> getWarnings(...)

}
```

Table 4.1 summarizes the methods that shall be provided by the `ThreeDS2Service` interface.

Table 4.1: ThreeDS2Service Interface Methods

Method	Description
`initialize`	Initializes the 3DS SDK instance.
`createTransaction`	Creates an instance of `Transaction` through which the 3DS Requestor App gets the data that is required to perform the transaction.

Method	Description
`cleanup`	Frees up resources that are used by the 3DS Requestor App until it is closed. It shall be called only once during a single 3DS Requestor App session.
`getSDKVersion`	Returns the version of the 3DS SDK that is integrated with the 3DS Requestor App.
`getWarnings`	Returns warnings produced by the 3DS SDK while performing security checks during initialization.

4.1.1 initialize

The 3DS Requestor App calls the `initialize` method at the start of the payment stage of a transaction. The app passes configuration parameters, UI configuration parameters, and (optionally) user locale to this method.

Note: Until the ThreeDS2Service instance is initialized, it shall be unusable.

The following tasks are performed during initialization:

- Security checks
- Collection of device information for all versions of the protocol that the SDK supports. For more information about the device identification parameters that shall be collected, refer to *EMV 3-D Secure SDK - Device Information* (later referred to as *EMV 3DS SDK Device Information* in this document).

Depending on the 3DS Requestor App implementation, a `ThreeDS2Service` instance is called either during 3DS Requestor App startup as a background task or when a transaction is initiated. The state is maintained until the `cleanup` method is called.

The following Android code snippet shows the signature of the `initialize` method:

```
public void initialize(android.content.Context applicationContext,
ConfigParameters configParameters, String locale, UiCustomization
uiCustomization) throws InvalidInputException,
SDKAlreadyInitializedException, SDKRuntimeException
```

initialize Parameters

Table 4.2: initialize Parameters

Parameter	Mandatory?	Description
`applicationContext`	Yes	An instance of Android application context.

Parameter	Mandatory?	Description
`configParameters`	Yes	Configuration information that shall be used during initialization. For more information, see `Class ConfigParameters`.
`locale`	No	String that represents the locale for the app's user interface. For example, the value of locale can be "en_US" in Java. **Note: If this parameter is not provided, then the default device locale is used.**
`uiCustomization`	No	UI configuration information that is used to specify the UI layout and theme. For example, font style and font size. For more information, see `Class UiCustomization`.

initialize Return Value

None.

initialize Exceptions

Table 4.3: initialize Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown in any of the following scenarios: • `configParameters` is null. • The value of `configParameters`, `locale`, or `uiCustomization` is invalid. For more information, see `Class InvalidInputException`.

Exception	Description
`SDKAlreadyInitializedException`	This exception shall be thrown if the 3DS SDK instance has already been initialized. For more information, see `Class SDKAlreadyInitializedException`.
`SDKRuntimeException`	This exception shall be thrown if an internal error is encountered by the 3DS SDK. For more information, see `Class SDKRuntimeException`.

4.1.2 createTransaction

The `createTransaction` method creates an instance of `Transaction` through which the 3DS Requestor App gets the data that is required to perform the transaction.

The 3DS Requestor App calls the `createTransaction` method for each transaction that is to be processed.
When the `createTransaction` method is called:

- The 3DS SDK uses the information adhering to the protocol version passed in the optional `messageVersion` parameter, if it supports the protocol version. If it does not support the protocol version, it generates an `InvalidInputException`. If the `messageVersion` parameter is empty or null, the highest protocol version that the 3DS SDK supports is used. If Challenge Flow is triggered for the transaction, the 3DS SDK uses the same protocol version during the challenge process.
- The 3DS SDK uses a secure random function to generate a Transaction ID in UUID format. This ID is used to uniquely identify each transaction.
- The 3DS SDK generates a fresh ephemeral key pair. This key pair is used to establish a secure session between the 3DS SDK and the ACS subsequently during the transaction.

The following Java code snippet shows the signature of the `createTransaction` method:

```
public Transaction createTransaction(String directoryServerID,
String messageVersion) throws
InvalidInputException,SDKNotInitializedException,
SDKRuntimeException
```

createTransaction Parameters

Table 4.4: createTransaction Parameters

Parameter	Mandatory?	Description
`directoryServerID`	Yes	Registered Application Provider Identifier (RID) that is unique to the Payment System. RIDs are defined by the ISO 7816-5 standard. RIDs are issued by the ISO/IEC 7816-5 registration authority. Contains a 5-byte value. The 3DS SDK encrypts the device information by using the DS public key. This key is identified based on the `directoryServerID` that is passed to the `createTransaction` method. Note: The 3DS SDK shall have the DS Public Keys of all the 3-D Secure participating Directory Servers.
`messageVersion`	No	Protocol version according to which the transaction shall be created.

createTransaction Return Value

This method returns an instance of the `Transaction` interface.

createTransaction Exceptions

Table 4.5: createTransaction Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. This also includes an invalid Directory Server ID or a protocol version that the 3DS SDK does not support. For more information, see `Class InvalidInputException`.

Exception	Description
`SDKNotInitializedException`	This exception shall be thrown if the 3DS SDK instance has not been initialized. For more information, see `Class SDKNotInitializedException`.
`SDKRuntimeException`	This exception shall be thrown if an internal error is encountered by the 3DS SDK. For more information, see `Class SDKRuntimeException`.

4.1.3 cleanup

The `cleanup` method frees up resources that are used by the 3DS SDK. It is called only once during a single 3DS Requestor App session.

The following Android code snippet shows the signature of the `cleanup` method:

```
public void cleanup(android.content.Context applicationContext)
throws SDKNotInitializedException
```

cleanup Parameters

Table 4.6: cleanup Parameters

Parameter	Mandatory?	Description
`applicationContext`	Conditional (Mandatory for Android)	3DS Requestor App context.

cleanup Return Value

None.

cleanup Exceptions

Table 4.7: cleanup Exceptions

Exception	Description
`SDKNotInitializedException`	This exception shall be thrown if the 3DS SDK instance has not been initialized. For more information, see `Class SDKNotInitializedException`.

4.1.4 getSDKVersion

The `getSDKVersion` method shall return the version of the 3DS SDK that is integrated with the 3DS Requestor App. For more information about the 3DS SDK version, refer to 3DS SDK Versioning Requirements.

The following Java code snippet shows the signature of the `getSDKVersion` method:

```
public String getSDKVersion() throws SDKNotInitializedException,
SDKRuntimeException
```

getSDKVersion Parameters

None.

getSDKVersion Return Value

This method returns (as a string) the version of the 3DS SDK that is integrated with the 3DS Requestor App.

getSDKVersion Exceptions

Table 4.8: getSDKVersion Exceptions

Exception	Description
`SDKNotInitializedException`	This exception shall be thrown if the 3DS SDK instance has not been initialized. For more information, see `Class SDKNotInitializedException.`

Exception	Description
`SDKRuntimeException`	This exception shall be thrown if an internal error is encountered by the 3DS SDK. For more information, see `Class SDKRuntimeException`.

4.1.5 getWarnings

The `getWarnings` method shall return the warnings produced by the 3DS SDK during initialization.
The following Java code snippet shows the signature of the `getWarnings` method:

```
public List<Warning> getWarnings()
```

getWarnings Parameters

None.

getWarnings Return Value

This method returns a List of warnings produced by the 3DS SDK during initialization.

getWarnings Exceptions

None.

4.2 Class ConfigParameters

The `ConfigParameters` class shall represent the configuration parameters that are required by the 3DS SDK for initialization.
The following are characteristics of the configuration parameters:

- All related configuration parameters can be placed in a single group.
  Note: A group is not pre-defined. The 3DS SDK implementer can define it to logically group configuration parameters.
- Explicit parameter grouping is optional. If a group name is not provided, then parameters are grouped under a default group.
- Duplicate parameter names cannot be used within a given group or the default group.

The 3DS Requestor App creates a `ConfigParameters` object and sets the required parameter values.
The following Java code snippet shows the definition of the `ConfigParameters` class:

```
public class ConfigParameters {

       public void addParam(...)

       public String getParamValue(...)
```

```
        public String removeParam(...)

}
```

Table 4.9 summarizes the methods that shall be provided by the `ConfigParameters` class.

Table 4.9: ConfigParameters Class Methods

Method	Description
`addParam`	Adds a configuration parameter either to the specified group or to the default group.
`getParam`	Returns a configuration parameter's value either from the specified group or from the default group.
`removeParam`	Removes a configuration parameter either from the specified group or from the default group. It should return the name of the parameter that it removes.

4.2.1 addParam

The `addParam` method shall add a configuration parameter either to the specified group or to the default group.

The following Java code snippet shows the signature of the `addParam` method:

```
public void addParam(String group, String paramName, String
paramValue) throws InvalidInputException
```

addParam Parameters

Table 4.10: addParam Parameters

Parameter	Mandatory?	Description
`group`	No	Group to which the configuration parameter is to be added. **Note: If a group is not specified, then the default group shall be used.**
`paramName`	Yes	Name of the configuration parameter.
`paramValue`	No	Value of the configuration parameter.

addParam Return Value

None.

addParam Exceptions

Table 4.11: addParam Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if `paramName` is null. For more information, see `Class InvalidInputException`.

4.2.2 getParamValue

The `getParamValue` method shall return a configuration parameter's value either from the specified group or from the default group.

The following Java code snippet shows the signature of the `getParamValue` method:

```
public String getParamValue(String group, String paramName) throws
InvalidInputException
```

getParamValue Parameters

Table 4.12: getParamValue Parameters

Parameter	Mandatory?	Description
`group`	No	Group from which the configuration parameter's value is to be returned. **Note: If the group is null, then the default group shall be used.**
`paramName`	Yes	Name of the configuration parameter.

getParamValue Return Value

The `getParamValue` method returns the value of the specified configuration parameter as a string.

getParamValue Exceptions

Table 4.13: getParamValue Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if `paramName` is null. For more information, see `Class InvalidInputException`.

4.2.3 removeParam

The `removeParam` method shall remove a configuration parameter either from the specified group or from the default group. It should return the name of the parameter that it removes.

The following Java code snippet shows the signature of the `removeParam` method:

```
public String removeParam (String group, String paramName) throws
InvalidInputException
```

removeParam Parameters

Table 4.14: removeParam Parameters

Parameter	Mandatory?	Description
`group`	No	Group from which the configuration parameter is to be removed. **Note: If `group` is null, then the default group shall be used.**
`paramName`	Yes	Name of the configuration parameter.

removeParam Return Value

The `removeParam` method should return the name of the parameter that it removes.

removeParam Exceptions

Table 4.15: removeParam Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if `paramName` is null. For more information, see `Class InvalidInputException`.

4.3 Interface ChallengeStatusReceiver

A callback object that implements the `ChallengeStatusReceiver` interface shall receive challenge status notification from the 3DS SDK at the end of the challenge process. This receiver object may be notified by calling various methods.

Depending on the result of the challenge process, the 3DS Requestor App may display a message or redirect the Cardholder to a screen in the app.

The following Java code snippet shows the definition of the `ChallengeStatusReceiver` interface:

```
public interface ChallengeStatusReceiver {

        public void completed (...)

        public void cancelled (...)

        public void timedout (...)

        public void protocolError(...)

        public void runtimeError(...)

}
```

Table 4.16 summarizes the methods that shall be provided by the `ChallengeStatusReceiver` interface. Each method corresponds to an event that can take place during the authentication process.

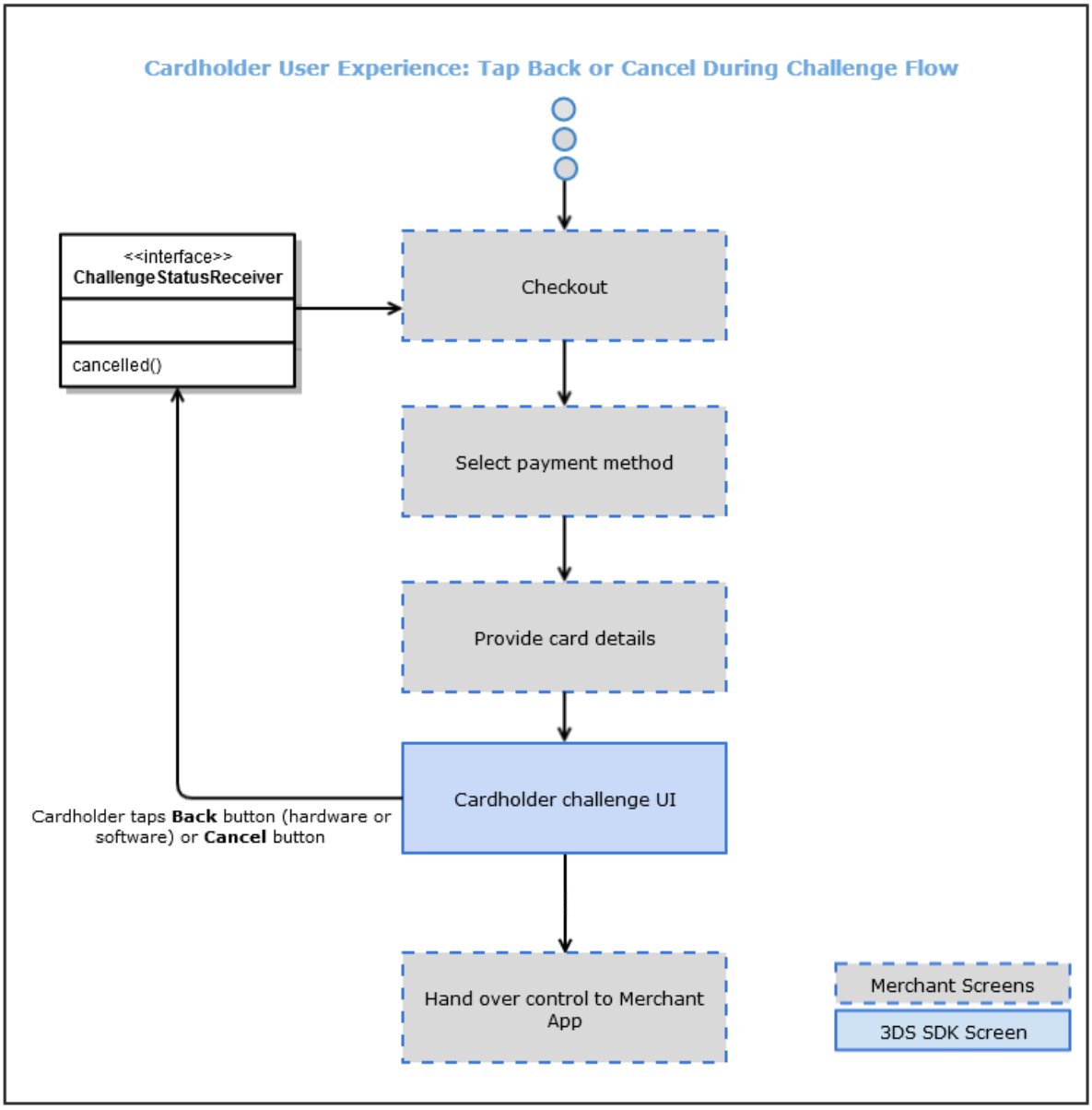
Table 4.16: ChallengeStatusReceiver Interface Methods

Method	Description
`completed`	Called when the challenge process (that is, the transaction) is completed. When a transaction is completed, a transaction status shall be available.
`cancelled`	Called when the Cardholder selects the option to cancel the transaction on the challenge screen.
`timedout`	Called when the challenge process reaches or exceeds the timeout interval that is specified during the `doChallenge` call on the 3DS SDK.
`protocolError`	Called when the 3DS SDK receives an EMV 3-D Secure protocol-defined error message from the ACS.

Method	Description
`runtimeError`	Called when the 3DS SDK encounters errors during the challenge process. These errors include all errors except those covered by the `protocolError` method.

Figure 4-1 shows the Cardholder user experience when the Cardholder taps the Back or Cancel buttons in the Challenge UI.

Figure 4-1: Cardholder User Experience: Tap Back or Cancel During Challenge Flow



The following steps summarize the events that take place when the Cardholder taps the Back or Cancel buttons in the Challenge UI:

1. At the checkout stage of the transaction, the Cardholder selects the Card payment method.
2. The Cardholder provides the card details.
3. The SDK presents the Challenge UI.
4. If the Cardholder taps the **Back** button (hardware or software) or **Cancel** button, the `cancelled` method shall be called and control shall return to the 3DS Requestor App.

4.3.1 completed

The `completed` method shall be called when the challenge process is completed. When a transaction is completed, the transaction status shall be available.

The following Java code snippet shows the signature of the `completed` method:

```
public void completed(CompletionEvent completionEvent)
```

completed Parameters

Table 4.17: completed Parameters

Parameter	Mandatory?	Description
`completionEvent`	Yes	Information about completion of the challenge process. For more information, see `Class CompletionEvent`.

completed Return Value

None.

completed Exceptions

None.

4.3.2 cancelled

The `cancelled` method shall be called when the Cardholder selects the option to cancel the transaction on the challenge screen.

Before sending notification about the cancelled event to the 3DS Requestor App, the 3DS SDK shall end the challenge flow. The app displays subsequent screens after it receives notification about this event.

The following Java code snippet shows the signature of the `cancelled` method:

```
public void cancelled()
```

cancelled Parameters

None.

cancelled Return Value

None.

cancelled Exceptions

None.

4.3.3 timedout

The `timedout` method shall be called when the challenge process reaches or exceeds the timeout specified during the `doChallenge` call on the 3DS SDK. On timeout, the SDK shall make a best effort to stop the challenge flow as soon as possible.

Before sending notification about the timed out event to the 3DS Requestor App, the 3DS SDK shall end the challenge flow. The app displays subsequent screens after it receives notification about this event.

The following Java code snippet shows the signature of the `timedout` method:

```
public void timedout()
```

timedout Parameters

None.

timedout Return Value

None.

timedout Exceptions

None.

4.3.4 protocolError

In the 3DS SDK context, a protocol error is any error message that is returned by the ACS. The `protocolError` method shall be called when the 3DS SDK receives such an error message. The 3DS SDK sends the error code and details from this error message as part of the notification to the 3DS Requestor App.

Note: A protocol error is not covered by the runtimeError method. For information about errors covered by the runtimeError method, refer to Class RuntimeErrorEvent.

Before sending notification about the Protocol Error event to the 3DS Requestor App, the 3DS SDK shall end the challenge flow. The app displays subsequent screens after it receives notification about this event.

The following Java code snippet shows the signature of the `protocolError` method:

```
public void protocolError(ProtocolErrorEvent protocolErrorEvent);
```

protocolError Parameters

Table 4.18: protocolError Parameters

Parameter	Mandatory?	Description
`protocolErrorEvent`	Yes	Error code and details. For more information, see `Class ProtocolErrorEvent`.

protocolError Return Value

None.

protocolError Exceptions

None.

4.3.5 runtimeError

The `runtimeError` method shall be called when the 3DS SDK encounters errors during the challenge process.

Note: A run-time error is not covered by the protocolError method. For information about errors covered by the protocolError method, refer to Class ProtocolErrorEvent.

Before sending notification about the run-time error event to the 3DS Requestor App, the 3DS SDK shall end the challenge flow. The app displays subsequent screens after it receives notification about this event.

The following Java code snippet shows the signature of the `runtimeError` method:

```
public void runtimeError(RuntimeErrorEvent runtimeErrorEvent)
```

runtimeError Parameters

Table 4.19: runtimeError Parameters

Parameter	Mandatory?	Description
`runtimeErrorEvent`	Yes	Error code and details. For more information, see `Class RuntimeErrorEvent.`

runtimeError Return Value

None.

runtimeError Exceptions

None.

4.4 Interface Transaction

An object that implements the `Transaction` interface shall hold parameters that the 3DS Server requires to create AReq messages and to perform the Challenge Flow.

The following Android code snippet shows the definition of the `Transaction` interface:

```
public interface Transaction {

      public AuthenticationRequestParameters
            getAuthenticationRequestParameters(...)

     public void doChallenge(...)

     public ProgressDialog getProgressView(...)

     public void close(...)

}
```

Table 4.20 summarizes the methods that shall be provided by the `Transaction` interface.

Table 4.20: Transaction Interface Methods

Method	Description
`getAuthenticationRequestParameters`	Returns device and 3DS SDK information to the 3DS Requestor App.
`doChallenge`	Initiates the challenge process.
`getProgressView`	Returns an instance of Progress View (processing screen) that the 3DS Requestor App uses.
`close`	Cleans up resources that are held by the Transaction object.

4.4.1 getAuthenticationRequestParameters

When the 3DS Requestor App calls the `getAuthenticationRequestParameters` method, the 3DS SDK shall encrypt the device information that it collects during initialization and send this information along with the SDK information to the 3DS Requestor App. The app includes this information in its message to the 3DS Server.

The 3DS SDK encrypts the device information by using the DS public key. This key is identified based on the `directoryServerID` that is passed to the `createTransaction` method. The 3DS SDK can use A128CBC-HS256 or A128GCM as the encryption algorithm. For more information about 3DS SDK encryption, refer to Section 6.2.2, "Function I: 3DS SDK Encryption to DS" in the *EMV 3DS Protocol Specification.*

The 3DS SDK shall generate an ephemeral key pair that is required for subsequent communication with the ACS if a challenge must be applied. For more information, refer to 3DS SDK – ACS Secure Channel.

The `getAuthenticationRequestParameters` method shall be called for every transaction.

The following Java code snippet shows the signature of the `getAuthenticationRequestParameters` method:

```
public AuthenticationRequestParameters
getAuthenticationRequestParameters()
```

getAuthenticationRequestParameters Parameters

None.

getAuthenticationRequestParameters Return Value

This method returns an `AuthenticationRequestParameters` object that contains device information and 3DS SDK information.

getAuthenticationRequestParameters Exceptions

None.

4.4.2 doChallenge

If the ARes that is returned indicates that the Challenge Flow must be applied, the 3DS Requestor App calls the `doChallenge` method with the required input parameters. The `doChallenge` method initiates the challenge process.

Note: The doChallenge method shall be called only when the Challenge Flow is to be applied.

When the `doChallenge` method is called, control of the app is passed to the 3DS SDK. At this point:

- The 3DS SDK shall start a time counter to measure the overall time taken by the challenge process.
- The 3DS SDK shall check if the CA public key (root) of the Directory Server CA (DS-CA) is present, based on the `directoryServerID` that was passed to the `createTransaction` method.
- The 3DS SDK shall use the CA public key of the DS-CA to validate the ACS signed content JWS object. Based on the information included in the JWS object, the algorithm used to perform the validation can be PS256 or ES256.

- The 3DS SDK shall complete the Diffie-Hellman key exchange process according to JWA (RFC 7518) in Direct Key Agreement mode using curve P-256. The output of this process is a pair of CEKs.
- The 3DS SDK shall use the CEKs to encrypt the CReq messages and decrypt the CRes messages.

For more information about the algorithms used for validation, and the CEKs, refer to the "3DS SDK Secure Channel Set-Up" section in Section 6.2.3, "Function J: 3DS SDK—ACS Secure Channel Set-Up" in the *EMV 3DS Protocol Specification.*

The 3DS SDK shall display the challenge to the Cardholder. The following steps shall take place during the challenge process:

- The 3DS Requestor App's current screen shall be closed either before the challenge screen is launched or before the `ChallengeStatusReceiver` callback is invoked by the 3DS SDK. This is to prevent the Cardholder from revisiting the card details screen using the Back button during the challenge process. For more information about the user experience when the Cardholder taps the Back button, refer to Figure 4-1.
- The 3DS SDK shall exchange two or more CReq and CRes messages with the ACS.
- The 3DS SDK shall send the challenge status back to the 3DS Requestor App by using the `ChallengeStatusReceiver` callback functions.
- The 3DS SDK shall clean up resources that are held by the `Transaction` object.

At any point of time, if the time taken by the challenge process (as measured by the time counter) exceeds the timeout value passed by the 3DS Requestor App, then the 3DS SDK shall call the `timedout` method of the ChallengeStatusReceiver callback object and clean up resources that are held by the Transaction object.

The following Android code snippet shows the signature of the `doChallenge` method:

```
public void doChallenge(android.app.Activity currentActivity,
ChallengeParameters challengeParameters,
ChallengeStatusReceiver challengeStatusReceiver, int timeOut)
throws InvalidInputException
```

doChallenge Parameters

Table 4.21: doChallenge Parameters

Parameter	Mandatory?	Description
`currentActivity`	Conditional (mandatory on Android)	The Android activity instance that invoked `doChallenge`.

Parameter	Mandatory?	Description
`challengeParameters`	Yes	ACS details (contained in the ARes) required by the 3DS SDK to conduct the challenge process during the transaction. The following details are mandatory: • 3DS Server Transaction ID • ACS Transaction ID • ACS Reference Number • ACS Signed Content
`challengeStatusReceiver`	Yes	Callback object for notifying the 3DS Requestor App about the challenge status. For more information, see Interface ChallengeStatusReceiver.
`timeOut`	Yes	Timeout interval (in minutes) within which the challenge process must be completed. The minimum timeout interval shall be 5 minutes.

doChallenge Return Value

None.

doChallenge Exceptions

Table 4.22: doChallenge Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. A timeout interval of less than 5 minutes is also treated as invalid input. For more information, see `Class InvalidInputException`.

4.4.3 getProgressView

The `getProgressView` method shall return an instance of Progress View (processing screen) that the 3DS Requestor App uses. The processing screen displays the Directory Server logo, and a graphical element to indicate that an activity is being processed.
The ProgressView object is created by the 3DS SDK.

The following Android code snippet shows the signature of the `getProgressView` method:

```
public ProgressDialog getProgressView(android.app.Activity
currentActivity) throws InvalidInputException
```

getProgressView Parameters

Table 4.23: getProgressView Parameters

Parameter	Mandatory?	Description
`currentActivity`	Conditional (mandatory on Android)	An Android activity instance.

getProgressView Return Value

This method returns a `ProgressDialog` object.

getProgressView Exceptions

Table 4.24: getProgressView Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.4.4 close

The `close` method is called to clean up resources that are held by the `Transaction` object. It shall be called when the transaction is completed.

Note: This method is required to be called only when the `doChallenge` method is not called in the transaction.

The following are some examples of scenarios in which the `close` method is called:

- Frictionless transaction.
- The ACS recommends a challenge, but the Merchant overrides the recommendation and chooses to complete the transaction without a challenge.

The following Android code snippet shows the signature of the `close` method:

```
public void close()
```

close Parameters

None.

close Return Value

None.

close Exceptions

None.

4.5 Class UiCustomization

The `UiCustomization` class shall provide the functionality required to customize the 3DS SDK UI elements. An object of this class holds various UI-related parameters.

The following Java code snippet shows the definition of the `UiCustomization` class:

```
public class UiCustomization {

    public enum ButtonType {VERIFY, CONTINUE, NEXT, CANCEL, RESEND}
    public void setButtonCustomization(…)
    public void setToolbarCustomization(…)
    public void setLabelCustomization(…)
    public void setTextBoxCustomization(…)
    public ButtonCustomization getButtonCustomization()
    public ToolbarCustomization getToolbarCustomization()
    public LabelCustomization getLabelCustomization()
    public TextBoxCustomization getTextBoxCustomization()
    }
```

Table 4.25 summarizes the methods that shall be provided by the `UiCustomization` class.

Table 4.25: UiCustomization Class Methods

Method	Description
`setButtonCustomization`	Sets the attributes of a `ButtonCustomization` object for a particular button type. For more information, see `Class ButtonCustomization`.

Method	Description
`setButtonCustomization`	Sets the attributes of a `ButtonCustomization` object for an implementer-specific button type. For more information, see `Class ButtonCustomization`.
`setToolbarCustomization`	Sets the attributes of a `ToolbarCustomization` object. For more information, see `Class ToolbarCustomization`.
`setLabelCustomization`	Sets the attributes of a `LabelCustomization` object. For more information, see `Class LabelCustomization`.
`setTextBoxCustomization`	Sets the attributes of a `TextBoxCustomization` object. For more information, see `Class TextBoxCustomization`.
`getButtonCustomization`	Returns a `ButtonCustomization` object. For more information, see `Class ButtonCustomization`.
`getButtonCustomization`	Returns a `ButtonCustomization` object for an implementer-specific button type. For more information, see `Class ButtonCustomization`.
`getToolbarCustomization`	Returns a `ToolbarCustomization` object. For more information, see `Class ToolbarCustomization`.
`getLabelCustomization`	Returns a `LabelCustomization` object. For more information, see `Class LabelCustomization`.
`getTextBoxCustomization`	Returns a `TextBoxCustomization` object. For more information, see `Class TextBoxCustomization`.

4.5.1 setButtonCustomization

The `setButtonCustomization` method shall accept a `ButtonCustomization` object along with a predefined button type. The 3DS SDK uses this object for customizing buttons.

Note: The 3DS SDK implementer shall maintain a dictionary of buttons passed via this method for use during customization.

The following Java code snippet shows the signature of the `setButtonCustomization` method:

```
public void setButtonCustomization (ButtonCustomization
buttonCustomization, ButtonType buttonType) throws
InvalidInputException
```

setButtonCustomization Parameters

Table 4.26: setButtonCustomization Parameters

Parameter	Mandatory?	Description
`buttonCustomization`	Yes	A `ButtonCustomization` object.
`buttonType`	Yes	The `ButtonType`.

setButtonCustomization Return Value

None.

setButtonCustomization Exceptions

Table 4.27: setButtonCustomization Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.5.2 setButtonCustomization

This method is a variation of the setButtonCustomization method.
The `setButtonCustomization` method shall accept a `ButtonCustomization` object and an implementer-specific button type. The 3DS SDK uses this object for customizing buttons.

Note: This method shall be used when the SDK implementer wants to use a button type that is not included in the predefined Enum `ButtonType`.
The SDK implementer shall maintain a dictionary of buttons passed via this method for use during customization.

The following Java code snippet shows the signature of the `setButtonCustomization` method:

```
public void setButtonCustomization (ButtonCustomization
buttonCustomization, String buttonType) throws
InvalidInputException
```

setButtonCustomization Parameters

Table 4.28: setButtonCustomization Parameters

Parameter	Mandatory?	Description
`buttonCustomization`	Yes	A `ButtonCustomization` object.
`buttonType`	Yes	Implementer-specific button type.

setButtonCustomization Return Value

None.

setButtonCustomization Exceptions

Table 4.29: setButtonCustomization Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.5.3 setToolbarCustomization

The `setToolbarCustomization` method shall accept a `ToolbarCustomization` object. The 3DS SDK uses this object for customizing toolbars.

The following Java code snippet shows the signature of the `setToolbarCustomization` method:

```
public void setToolbarCustomization(ToolbarCustomization
toolbarCustomization) throws InvalidInputException
```

setToolbarCustomization Parameters

Table 4.30: setToolbarCustomization Parameters

Parameter	Mandatory?	Description
`toolbarCustomization`	Yes	A `ToolbarCustomization` object.

setToolbarCustomization Return Value

None.

setToolbarCustomization Exceptions

Table 4.31: setToolbarCustomization Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.5.4 setLabelCustomization

The `setLabelCustomization` method shall accept a `LabelCustomization` object. The 3DS SDK uses this object for customizing labels.

The following Java code snippet shows the signature of the `setLabelCustomization` method:

```
public void setLabelCustomization (LabelCustomization
labelCustomization) throws InvalidInputException
```

setLabelCustomization Parameters

Table 4.32: setLabelCustomization Parameters

Parameter	Mandatory?	Description
`labelCustomization`	Yes	A `LabelCustomization` object.

setLabelCustomization Return Value

None.

setLabelCustomization Exceptions

Table 4.33: setLabelCustomization Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.5.5 setTextBoxCustomization

The `setTextBoxCustomization` method shall accept a `TextBoxCustomization` object. The 3DS SDK uses this object for customizing text boxes.

The following Java code snippet shows the signature of the `setTextBoxCustomization` method:

```
public void setTextBoxCustomization (TextBoxCustomization
textBoxCustomization) throws InvalidInputException
```

setTextBoxCustomization Parameters

Table 4.34: setTextBoxCustomization Parameters

Parameter	Mandatory?	Description
`textBoxCustomization`	Yes	A `TextBoxCustomization` object.

setTextBoxCustomization Return Value

None.

setTextBoxCustomization Exceptions

Table 4.35: setTextBoxCustomization Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.5.6 getButtonCustomization

The `getButtonCustomization` method shall return a `ButtonCustomization` object for a specified button type.

The following Java code snippet shows the signature of the `getButtonCustomization` method:

```
public ButtonCustomization getButtonCustomization (ButtonType
buttonType) throws InvalidInputException;
```

getButtonCustomization Parameters

Table 4.36: getButtonCustomization Parameters

Parameter	Mandatory?	Description
`buttonType`	Yes	A pre-defined list of button types. For more information, see `Enum ButtonType`.

getButtonCustomization Return Value

This method returns a `Button Customization` object.

getButtonCustomization Exceptions

Table 4.37: getButtonCustomization Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.5.7 getButtonCustomization

The `getButtonCustomization` method shall return a `ButtonCustomization` object for an implementer-specific button type.

The following Java code snippet shows the signature of the `getButtonCustomization` method:

```
public ButtonCustomization getButtonCustomization (String
buttonType) throws InvalidInputException;
```

getButtonCustomization Parameters

Table 4.38: getButtonCustomization Parameters

Parameter	Mandatory?	Description
`buttonType`	Yes	Implementer-specific button type.

getButtonCustomization Return Value

This method returns a `ButtonCustomization` object.

getButtonCustomization Exceptions

Table 4.39: getButtonCustomization Parameters

Exception	Description
`InvalidInputExcepti on`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.5.8 getToolbarCustomization

The `getToolbarCustomization` method shall return a `ToolbarCustomization` object for a toolbar.

The following Java code snippet shows the signature of the `getToolbarCustomization` method:

```
public ToolbarCustomization getToolbarCustomization();
```

getToolbarCustomization Parameters

None.

getToolbarCustomization Return Value

This method returns a `ToolbarCustomization` object.

getToolbarCustomization Exceptions

None.

4.5.9 getLabelCustomization

The `getLabelCustomization` method shall return a `LabelCustomization` object.

The following Java code snippet shows the signature of the `getLabelCustomization` method:

```
public LabelCustomization getLabelCustomization()
```

getLabelCustomization Parameters

None.

getLabelCustomization Return Value

This method returns a `LabelCustomization` object.

getLabelCustomization Exceptions

None.

4.5.10 getTextBoxCustomization

The `getTextBoxCustomization` method shall return a `TextBoxCustomization` object.

The following Java code snippet shows the signature of the `getTextBoxCustomization` method:

```
public TextBoxCustomization getTextBoxCustomization()
```

getTextBoxCustomization Parameters

None.

getTextBoxCustomization Return Value

This method returns a `TextBoxCustomization` object.

getTextBoxCustomization Exceptions

None.

4.6 Class Customization

The `Customization` class shall serve as a superclass for the `ButtonCustomization` class, `ToolbarCustomization` class, `LabelCustomization` class, and `TextBoxCustomization` class. This class shall provide methods to pass UI customization parameters to the 3DS SDK.

The following Java code snippet shows the definition of the `Customization` class:

```
public class Customization {
    public void setTextFontName(…)
    public void setTextColor(…)
    public void setTextFontSize(…)
    public String getTextFontName()
    public String getTextColor()
    public int    getTextFontSize()
 }
```

Table 4.40 summarizes the methods that shall be provided by the `Customization` class.

Table 4.40: Customization Class Methods

Method	Description
`setTextFontName`	Sets the font type for a UI element.
`setTextColor`	Sets the text color for a UI element.
`setTextFontSize`	Sets the font size for a UI element.
`getTextFontName`	Returns the font type for a UI element.
`getTextColor`	Returns the text color for a UI element.
`getTextFontSize`	Returns the font size for a UI element.

4.6.1 setTextFontName

The `setTextFontName` method shall set the font type for a UI element.

The following Java code snippet shows the signature of the `setTextFontName` method:

```
public void setTextFontName (String fontName) throws InvalidInputException
```

setTextFontName Parameters

Table 4.41: setTextFontName Parameters

Parameter	Mandatory?	Description
`fontName`	Yes	Font type for the UI element.

setTextFontName Return Value

None.

setTextFontName Exceptions

Table 4.42: setTextFontName Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.6.2 setTextColor

The `setTextColor` method shall set the text color for a UI element.

The following Java code snippet shows the signature of the `setTextColor` method:

```
public void setTextColor (String hexColorCode) throws
InvalidInputException
```

setTextColor Parameters

Table 4.43: setTextColor Parameters

Parameter	Mandatory?	Description
`hexColorCode`	Yes	Color code in Hex format. For example, the color code can be “#999999”.

setTextColor Return Value

None.

setTextColor Exceptions

Table 4.44: setTextColor Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.6.3 setTextFontSize

The `setTextFontSize` method shall set the font size for a UI element.

The following Java code snippet shows the signature of the `setTextFontSize` method:

```
public void setTextFontSize (int fontSize) throws
InvalidInputException
```

setTextFontSize Parameters

Table 4.45: setTextFontSize Parameters

Parameter	Mandatory?	Description
`fontSize`	Yes	Font size for the UI element.

setTextFontSize Return Value

None.

setTextFontSize Exceptions

Table 4.46: setTextFontSize Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.6.4 getTextFontName

The `getTextFontName` method shall return the font type for a UI element.

The following Java code snippet shows the signature of the `getTextFontName` method:

```
public String getTextFontName()
```

getTextFontName Parameters

None.

getTextFontName Return Value

The `getTextFontName` method returns the font type as a string.

getTextFontName Exceptions

None.

4.6.5 getTextColor

The `getTextColor` method shall return the hex color code for a UI element.

The following Java code snippet shows the signature of the `getTextColor` method:

```
public String getTextColor()
```

getTextColor Parameters

None.

getTextColor Return Value

The `getTextColor` method returns the hex color code as a string.

getTextColor Exceptions

None.

4.6.6 getTextFontSize

The `getTextFontSize` method shall return the font size for a UI element.

The following Java code snippet shows the signature of the `getTextFontSize` method:

```
public int getTextFontSize()
```

getTextFontSize Parameters

None.

getTextFontSize Return Value

The `getTextFontSize` method returns the font size as an integer.

getTextFontSize Exceptions

None.

4.7 Class ButtonCustomization

The `ButtonCustomization` class shall provide methods for the 3DS Requestor App to pass button customization parameters to the 3DS SDK. This class shall extend the `Customization` class. The methods that are inherited from the `Customization` class can be used to work with button labels.
The following Java code snippet shows the definition of the `ButtonCustomization` class:

```
public class ButtonCustomization extends Customization {

    public void setBackgroundColor(…)
    public void setCornerRadius(…)
    public String getBackgroundColor()
    public int getCornerRadius()

}
```

Table 4.47 summarizes the methods that shall be provided by the `ButtonCustomization` class.

Table 4.47: ButtonCustomization Class Methods

Method	Description
`setBackgroundColor`	Sets the background colour of the button.
`setCornerRadius`	Sets the radius of the button corners.
`getBackgroundColor`	Returns the background colour of the button.
`getCornerRadius`	Returns the radius of the button corners.

The `ButtonCustomization` class shall inherit the following methods from the `Customization` class:

- setTextFontName
- setTextColor
- setTextFontSize
- getTextFontName
- getTextColor
- getTextFontSize

4.7.1 setBackgroundColor

The `setBackgroundColor` method shall set the background colour of the button.

The following Java code snippet shows the signature of the `setBackgroundColor` method:

```
public void setBackgroundColor(String hexColorCode) throws
InvalidInputException
```

setBackgroundColor Parameters

Table 4.48: setBackgroundColor Parameters

Parameter	Mandatory?	Description
`hexColorCode`	Yes	Colour code in Hex format. For example, the colour code can be “#999999”.

setBackgroundColor Return Value

None.

setBackgroundColor Exceptions

Table 4.49: setBackgroundColor Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.7.2 setCornerRadius

The `setCornerRadius` method shall set the radius of the button corners.

The following Java code snippet shows the signature of the `setCornerRadius` method:

```
public void setCornerRadius(int cornerRadius) throws
InvalidInputException
```

setCornerRadius Parameters

Table 4.50: setCornerRadius Parameters

Parameter	Mandatory?	Description
`cornerRadius`	Yes	Radius (integer value) for the button corners.

setCornerRadius Return Value

None.

setCornerRadius Exceptions

Table 4.51: setCornerRadius Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.7.3 getBackgroundColor

The `getBackgroundColor` method shall return the background colour of the button.

The following Java code snippet shows the signature of the `getBackgroundColor` method:

```
public String getBackgroundColor()
```

getBackgroundColor Parameters

None.

getBackgroundColor Return Value

The `getBackgroundColor` method returns the background colour code (as a string) of the button.

getBackgroundColor Exceptions

None.

4.7.4 getCornerRadius

The `getCornerRadius` method shall return the radius of the button corners.

The following Java code snippet shows the signature of the `getCornerRadius` method:

```
public int getCornerRadius()
```

getCornerRadius Parameters

None.

getCornerRadius Return Value

The `getCornerRadius` method returns the radius (as an integer) of the button corners.

getCornerRadius Exceptions

None.

4.8 Class ToolbarCustomization

The `ToolbarCustomization` class shall provide methods for the 3DS Requestor App to pass toolbar customization parameters to the 3DS SDK. This class shall extend the `Customization` class. The methods that are inherited from the `Customization` class can be used to work with toolbar labels.
The following Java code snippet shows the definition of the `ToolbarCustomization` class:

```
public class ToolbarCustomization extends Customization {

    public void setBackgroundColor(…)
    public String getBackgroundColor()
    public void setHeaderText(…)
    public String getHeaderText()
```

```
    public void setButtonText(…)
    public String getButtonText()

}
```

As seen in the definition, the `ToolbarCustomization` class shall provide the methods listed in Table 4.52.

Table 4.52: ToolbarCustomization Class Methods

Method	Description
`setBackgroundColor`	Sets the background colour for the toolbar.
`getBackgroundColor`	Returns the background colour for the toolbar.
`setHeaderText`	Sets the header text of the toolbar.
`getHeaderText`	Returns the header text of the toolbar.
`setButtonText`	Sets the button text of the toolbar.
`getButtonText`	Returns the button text of the toolbar.

The `ToolbarCustomization` class shall inherit the following methods from the `Customization` class:

- setTextFontName
- setTextColor
- setTextFontSize
- getTextFontName
- getTextColor
- getTextFontSize

4.8.1 setBackgroundColor

The `setBackgroundColor` method shall set the background colour for the toolbar.

The following Java code snippet shows the signature of the `setBackgroundColor` method:

```
public void setBackgroundColor(String hexColorCode) throws
InvalidInputException
```

setBackgroundColor Parameters

Table 4.53: setBackgroundColor Parameters

Parameter	Mandatory?	Description
`hexColorCode`	Yes	Colour code in Hex format. For example, the colour code can be "#999999".

setBackgroundColor Return Value

None.

setBackgroundColor Exceptions

Table 4.54: setBackgroundColor Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.8.2 getBackgroundColor

The `getBackgroundColor` method shall return the background colour for the toolbar.

The following Java code snippet shows the signature of the `getBackgroundColor` method:

```
public String getBackgroundColor()
```

getBackgroundColor Parameters

None.

getBackgroundColor Return Value

The `getBackgroundColor` method returns the background colour code (as a String) for the toolbar.

getBackgroundColor Exceptions

None.

4.8.3 setHeaderText

The `setHeaderText` method shall set the header text of the toolbar.

The following Java code snippet shows the signature of the `setHeaderText` method:

```
public void setHeaderText (String headerText) throws
InvalidInputException
```

setHeaderText Parameters

Table 4.55: setHeaderText Parameters

Parameter	Mandatory?	Description
`headerText`	Yes	Text for the header.

setHeaderText Return Value

None.

setHeaderText Exceptions

Table 4.56: setHeaderText Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.8.4 getHeaderText

The `getHeaderText` method shall return the header text of the toolbar.

The following Java code snippet shows the signature of the `getHeaderText` method:

```
public String getHeaderText()
```

getHeaderText Parameters

None.

getHeaderText Return Value

The `getHeaderText` method returns the header text (as a String) of the toolbar.

getHeaderText Exceptions

None.

4.8.5 setButtonText

The `setButtonText` method shall set the button text of the toolbar.

The following Java code snippet shows the signature of the `setButtonText` method:

```
public void setButtonText(String buttonText) throws
InvalidInputException
```

setButtonText Parameters

Table 4.57: setButtonText Parameters

Parameter	Mandatory?	Description
`buttonText`	Yes	Text for the button. For example, “Cancel”.

setButtonText Return Value

None.

setButtonText Exceptions

Table 4.58: setButtonText Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.8.6 getButtonText

The `getButtonText` method shall return the button text of the toolbar.

The following Java code snippet shows the signature of the `getButtonText` method:

```
public String getButtonText()
```

getButtonText Parameters

None.

getButtonText Return Value

The `getButtonText` method returns the button text (as a String) of the toolbar.

getButtonText Exceptions

None.

4.9 Class LabelCustomization

The `LabelCustomization` class shall provide methods for the 3DS Requestor App to pass label customization parameters to the 3DS SDK. This class shall extend the Customization class. The methods that are inherited from the `Customization` class can be used to work with non-heading labels in the UI.

The following Java code snippet shows the definition of the `LabelCustomization` class:

```
public class LabelCustomization extends Customization {

    public void setHeadingTextColor(…)
    public void setHeadingTextFontName(…)
    public void setHeadingTextFontSize(…)
    public String getHeadingTextColor()
    public String getHeadingTextFontName()
    public int getHeadingTextFontSize()
}
```

Table 4.59 summarizes the methods that shall be provided by the `LabelCustomization` class.

Table 4.59: LabelCustomization Class Methods

Method	Description
`setHeadingTextColor`	Sets the colour of the heading label text.
`setHeadingTextFontName`	Sets the font type of the heading label text.
`setHeadingTextFontSize`	Sets the font size of the heading label text.
`getHeadingTextColor`	Returns the colour of the heading label text.
`getHeadingTextFontName`	Returns the font type of the heading label text.
`getHeadingTextFontSize`	Returns the font size of the heading label text.

The `LabelCustomization` class shall inherit the following methods from the `Customization` class:

- setTextFontName
- setTextColor
- setTextFontSize
- getTextFontName
- getTextColor
- getTextFontSize

4.9.1 setHeadingTextColor

The `setHeadingTextColor` method shall set the colour of the heading label text.

The following Java code snippet shows the signature of the `setHeadingTextColor` method:

```
public void setHeadingTextColor(String hexColorCode) throws
InvalidInputException
```

setHeadingTextColor Parameters

Table 4.60: setHeadingTextColor Parameters

Parameter	Mandatory?	Description
`hexColorCode`	Yes	Colour code in Hex format. For example, the colour code can be "#999999".

setHeadingTextColor Return Value

None.

setHeadingTextColor Exceptions

Table 4.61: setHeadingTextColor Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.9.2 setHeadingTextFontName

The `setHeadingTextFontName` method shall set the font type of the heading label text.

The following Java code snippet shows the signature of the `setHeadingTextFontName` method:

```
public void setHeadingTextFontName(String fontName) throws
InvalidInputException
```

setHeadingTextFontName Parameters

Table 4.62: setHeadingTextFontName Parameters

Parameter	Mandatory?	Description
`fontName`	Yes	Font type for the heading label text.

setHeadingTextFontName Return Value

None.

setHeadingTextFontName Exceptions

Table 4.63: setHeadingTextFontName Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.9.3 setHeadingTextFontSize

The `setHeadingTextFontSize` method shall set the font size of the heading label text.

The following Java code snippet shows the signature of the `setHeadingTextFontSize` method:

```
public void setHeadingTextFontSize(int fontSize) throws
InvalidInputException
```

setHeadingTextFontSize Parameters

Table 4.64: setHeadingTextFontSize Parameters

Parameter	Mandatory?	Description
`fontSize`	Yes	Font size for the heading label text.

setHeadingTextFontSize Return Value

None.

setHeadingTextFontSize Exceptions

Table 4.65: setHeadingTextFontSize Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.9.4 getHeadingTextColor

The `getHeadingTextColor` method shall return the hex colour code of the heading label text.

The following Java code snippet shows the signature of the `getHeadingTextColor` method:

```
public String getHeadingTextColor()
```

getHeadingTextColor Parameters

None.

getHeadingTextColor Return Value

The `getHeadingTextColor` method returns the hex color code of the heading label text as a string.

getHeadingTextColor Exceptions

None.

4.9.5 getHeadingTextFontName

The `getHeadingTextFontName` method shall return the font type of the heading label text.

The following Java code snippet shows the signature of the `getHeadingTextFontName` method:

```
public String getHeadingTextFontName()
```

getHeadingTextFontName Parameters

None.

getHeadingTextFontName Return Value

The `getHeadingTextFontName` method returns the font type of the heading label text as a string.

getHeadingTextFontName Exceptions

None.

4.9.6 getHeadingTextFontSize

The `getHeadingTextFontSize` method shall return the font size of the heading label text.

The following Java code snippet shows the signature of the `getHeadingTextFontSize` method:

```
public int getHeadingTextFontSize()
```

getHeadingTextFontSize Parameters

None.

getHeadingTextFontSize Return Value

The `getHeadingTextFontSize` method returns the heading text font size as an integer.

getHeadingTextFontSize Exceptions

None.

4.10 Class TextBoxCustomization

The `TextBoxCustomization` class shall provide methods for the 3DS Requestor App to pass text box customization parameters to the 3DS SDK. This class shall extend the `Customization` class. The methods that are inherited from the `Customization` class can be used to set the properties of user-entered text in text boxes.

The following Java code snippet shows the definition of the `TextBoxCustomization` class:

```
public class TextBoxCustomization extends Customization {

    public void setBorderWidth(…)
    public int getBorderWidth()
    public void setBorderColor(…)
    public String getBorderColor()
    public void setCornerRadius(…)
    public int getCornerRadius()

}
```

Table 4.66 summarizes the methods that shall be provided by the `TextBoxCustomization` class.

Table 4.66: TextBoxCustomization Class Methods

Method	Description
`setBorderWidth`	Sets the width of the text box border.
`getBorderWidth`	Returns the width of the text box border.
`setBorderColor`	Sets the color of the text box border.

Method	Description
`getBorderColor`	Returns the color of the text box border in hex colour code.
`setCornerRadius`	Sets the corner radius of the text box corners.
`getCornerRadius`	Gets the corner radius of the text box corners.

The `TextBoxCustomization` class shall inherit the following methods from the `Customization` class:

- setTextFontName
- setTextColor
- setTextFontSize
- getTextFontName
- getTextColor
- getTextFontSize

4.10.1 setBorderWidth

The `setBorderWidth` method shall set the width of the text box border.

The following Java code snippet shows the signature of the `setBorderWidth` method:

```
public void setBorderWidth (int borderWidth) throws
InvalidInputException
```

setBorderWidth Parameters

Table 4.67: setBorderWidth Parameters

Parameter	Mandatory?	Description
`borderWidth`	Yes	Width (integer value) of the text box border.

setBorderWidth Return Value

None.

setBorderWidth Exceptions

Table 4.68: setBorderWidth Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.10.2 getBorderWidth

The `getBorderWidth` method shall return the width of the text box border. The SDK implementer shall ensure that the border exists before this method is called.

The following Java code snippet shows the signature of the `getBorderWidth` method:

```
public int getBorderWidth ()
```

getBorderWidth Parameters

None.

getBorderWidth Return Value

The `getBorderWidth` method returns the width (as an integer) of the text box border.

getBorderWidth Exceptions

None.

4.10.3 setBorderColor

The `setBorderColor` method shall set the color for the border of the text box.

The following Java code snippet shows the signature of the `setBorderColor` method:

```
public void setBorderColor(String hexColorCode) throws
InvalidInputException
```

setBorderColor Parameters

Table 4.69: setBorderColor Parameters

Parameter	Mandatory?	Description
`hexColorCode`	Yes	Color code in Hex format. For example, the color code can be “#999999”.

setBorderColor Return Value

None.

setBorderColor Exceptions

Table 4.70: setBorderColor Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.10.4 getBorderColor

The `getBorderColor` method shall return the color of the text box border. The SDK implementer shall ensure that the border exists before this method is called.

The following Java code snippet shows the signature of the `getBorderColor` method:

```
public int getBorderColor()
```

getBorderColor Parameters

None.

getBorderColor Return Value

The `getBorderColor` method returns the hex color code (as a string) of the text box border.

getBorderColor Exceptions

None.

4.10.5 setCornerRadius

The `setCornerRadius` method shall set the radius of the text box corners.

The following Java code snippet shows the signature of the `setCornerRadius` method:

```
public void setCornerRadius(int cornerRadius) throws
InvalidInputException
```

setCornerRadius Parameters

Table 4.71: setCornerRadius Parameters

Parameter	Mandatory?	Description
`cornerRadius`	Yes	Radius (integer value) for the text box corners.

setCornerRadius Return Value

None.

setCornerRadius Exceptions

Table 4.72: setCornerRadius Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.10.6 getCornerRadius

The `getCornerRadius` method shall return the radius of the text box corners.

The following Java code snippet shows the signature of the `getCornerRadius` method:

```
public int getCornerRadius()
```

getCornerRadius Parameters

None.

getCornerRadius Return Value

The `getCornerRadius` method returns the radius (as an integer) of the text box corners.

getCornerRadius Exceptions

None.

4.11 Class ChallengeParameters

The `ChallengeParameters` class shall hold the parameters that are required to conduct the challenge process.

Note: It is mandatory to set values for these parameters.

The following Java code snippet shows the definition of the `ChallengeParameters` class:

```
public class ChallengeParameters {
      public void set3DSServerTransactionID(...)
      public void setAcsTransactionID(...)
      public void setAcsRefNumber(...)
      public void setAcsSignedContent(...)
```

```
      public String get3DSServerTransactionID(...)
      public String getAcsTransactionID(...)
      public String getAcsRefNumber(...)
      public String getAcsSignedContent(...)
}
```

Table 4.73 summarizes the methods that shall be provided by the `ChallengeParameters` class.

Table 4.73: ChallengeParameters Class Methods

Method	Description
`set3DSServerTransactionID`	Sets the 3DS Server Transaction ID.
`setAcsTransactionID`	Sets the ACS Transaction ID.
`setAcsRefNumber`	Sets the ACS Reference Number.
`setAcsSignedContent`	Sets the ACS signed content. This data includes the ACS URL, ACS ephemeral public key, and SDK ephemeral public key.
`get3DSServerTransactionID`	Returns the 3DS Server Transaction ID.
`getAcsTransactionID`	Returns the ACS Transaction ID.
`getAcsRefNumber`	Returns the ACS Reference Number.
`getAcsSignedContent`	Returns the ACS signed content object.

4.11.1 set3DSServerTransactionID

The `set3DSServerTransactionID` method shall set the 3DS Server Transaction ID. This ID is a transaction identifier assigned by the 3DS Server to uniquely identify a single transaction.

The following Java code snippet shows the signature of the `set3DSServerTransactionID` method:

```
public void set3DSServerTransactionID(String
3DSServerTransactionID)
```

set3DSServerTransactionID Parameters

Table 4.74: set3DSServerTransactionID Parameters

Parameter	Mandatory?	Description
`3DSServerTransactionID`	Yes	Transaction identifier assigned by the 3DS Server to uniquely identify a single transaction.

set3DSServerTransactionID Return Value

None.

set3DSServerTransactionID Exceptions

None.

4.11.2 setAcsTransactionID

The `setAcsTransactionID` method shall set the ACS Transaction ID.

The following Java code snippet shows the signature of the `setAcsTransactionID` method:

```
public void setAcsTransactionID(String acsTransactionID)
```

setAcsTransactionID Parameters

Table 4.75: setAcsTransactionID Parameters

Parameter	Mandatory?	Description
`acsTransactionID`	Yes	Transaction ID assigned by the ACS to uniquely identify a single transaction.

setAcsTransactionID Return Value

None.

setAcsTransactionID Exceptions

None.

4.11.3 setAcsRefNumber

The `setAcsRefNumber` method shall set the ACS Reference Number.

The following Java code snippet shows the signature of the `setAcsRefNumber` method:

```
public void setAcsRefNumber(String acsRefNumber)
```

setAcsRefNumber Parameters

Table 4.76: setAcsRefNumber Parameters

Parameter	Mandatory?	Description
`acsRefNumber`	Yes	EMVCo assigns the ACS this identifier after running the EMV 3-D Secure Testing and Approvals process on the ACS.

setAcsRefNumber Return Value

None.

setAcsRefNumber Exceptions

None.

4.11.4 setAcsSignedContent

The `setAcsSignedContent` method shall set the ACS signed content. This content includes the ACS URL, ACS ephemeral public key, and SDK ephemeral public key.

The following Java code snippet shows the signature of the `setAcsSignedContent` method:

```
public void setAcsSignedContent(String acsSignedContent)
```

setAcsSignedContent Parameters

Table 4.77: setAcsSignedContent Parameters

Parameter	Mandatory?	Description
`acsSignedContent`	Yes	ACS signed content. This data includes the ACS URL, ACS ephemeral public key, and SDK ephemeral public key.

setAcsSignedContent Return Value

None.

setAcsSignedContent Exceptions

None.

4.11.5 get3DSServerTransactionID

The `get3DSServerTransactionID` method shall return the 3DS Server Transaction ID.

The following Java code snippet shows the signature of the `get3DSServerTransactionID` method:

```
public String get3DSServerTransactionID()
```

get3DSServerTransactionID Parameters

None.

get3DSServerTransactionID Return Value

This method returns the 3DS Server Transaction ID as a string.

get3DSServerTransactionID Exceptions

None.

4.11.6 getAcsTransactionID

The `getAcsTransactionID` method shall return the ACS Transaction ID.

The following Java code snippet shows the signature of the `getAcsTransactionID` method:

```
public String getAcsTransactionID()
```

getAcsTransactionID Parameters

None.

getAcsTransactionID Return Value

This method returns the ACS Transaction ID as a string.

getAcsTransactionID Exceptions

None.

4.11.7 getAcsRefNumber

The `getAcsRefNumber` method shall return the ACS Reference Number.

The following Java code snippet shows the signature of the `getAcsRefNumber` method:

```
public String getAcsRefNumber()
```

getAcsRefNumber Parameters

None.

getAcsRefNumber Return Value

This method returns the ACS Reference Number as a string.

getAcsRefNumber Exceptions

None.

4.11.8 getAcsSignedContent

The `getAcsSignedContent` method shall return the ACS signed content.

The following Java code snippet shows the signature of the `getAcsSignedContent` method:

```
public String getAcsSignedContent()
```

getAcsSignedContent Parameters

None.

getAcsSignedContent Return Value

This method returns the ACS signed content as a string.

getAcsSignedContent Exceptions

None.

4.12 Class AuthenticationRequestParameters

The `AuthenticationRequestParameters` class shall hold transaction data that the App passes to the 3DS Server for creating the AReq.

The following Java code snippet shows the definition of the `AuthenticationRequestParameters` class:

```
public class AuthenticationRequestParameters {

      public AuthenticationRequestParameters(…)
      public String getDeviceData()
      public String getSDKTransactionID()
      public String getSDKAppID()
      public String getSDKReferenceNumber()
      public String getSDKEphemeralPublicKey()
      public String getMessageVersion()

}
```

Table 4.78 summarizes the methods that shall be provided by the `AuthenticationRequestParameters` class.

Table 4.78: AuthenticationRequestParameters Class Methods

Method	Description
`AuthenticationRequestParameters`	Constructs an `AuthenticationRequestParameters` object.

Method	Description
`getDeviceData`	Returns a string that represents the encrypted device data.
`getSDKTransactionID`	Returns the SDK Transaction ID.
`getSDKAppID`	Returns the SDK App ID.
`getSDKReferenceNumber`	Returns the SDK Reference Number.
`getSDKEphemeralPublicKey`	Returns the SDK Ephemeral Public Key as a String representation of a JWK object.
`getMessageVersion`	Returns the protocol version that is used for the transaction.

4.12.1 AuthenticationRequestParameters

The `AuthenticationRequestParameters` constructor shall create an object that shall be used by the 3DS Server to obtain authentication parameters for creating the AReq.

The following Java code snippet shows the signature of the `AuthenticationRequestParameters` constructor:

```
public AuthenticationRequestParameters (String sdkTransactionID,
String deviceData, String sdkEphemeralPublicKey, String sdkAppID,
String sdkReferenceNumber, String messageVersion) throws
InvalidInputException
```

AuthenticationRequestParameters Parameters

Table 4.79: AuthenticationRequestParameters Parameters

Parameter	Mandatory?	Description
`sdkTransactionID`	Yes	SDK Transaction ID.
`deviceData`	Conditional	Device data collected by the SDK.
`sdkEphemeralPublicKey`	Yes	SDK Ephemeral Public Key (Qc).
`sdkAppID`	Yes	SDK App ID.
`sdkReferenceNumber`	Yes	SDK Reference Number.

Parameter	Mandatory?	Description
`messageVersion`	Yes	Protocol version that is supported by the SDK and used for the transaction.

AuthenticationRequestParameters Exceptions

Table 4.80: AuthenticationRequestParameters Exceptions

Exception	Description
`InvalidInputException`	This exception shall be thrown if an input parameter is invalid. For more information, see `Class InvalidInputException`.

4.12.2 getDeviceData

The `getDeviceData` method shall return the encrypted device data as a string.

The following Java code snippet shows the signature of the `getDeviceData` method:

```
public String getDeviceData()
```

getDeviceData Parameters

None.

getDeviceData Return Value

This method returns the encrypted device data as a JWE string.
.

getDeviceData Exceptions

None.

4.12.3 getSDKTransactionID

The `getSDKTransactionID` method shall return the SDK Transaction ID. When this method is called, the 3DS SDK uses a secure random function to generate a Transaction ID in UUID format.
For information about the Transaction ID, see the SDK Transaction ID row in Table A.1, “EMV 3-D Secure Data Elements” in the *EMV 3DS Protocol Specification*.

The following Java code snippet shows the signature of the `getSDKTransactionID` method:

```
public String getSDKTransactionID()
```

getSDKTransactionID Parameters

None.

getSDKTransactionID Return Value

The `getSDKTransactionID` method returns this Transaction ID as a string.

getSDKTransactionID Exceptions

None.

4.12.4 getSDKAppID

The `getSDKAppID` method shall return the SDK App ID. The 3DS SDK uses a secure random function to generate the App ID in UUID format. This ID is unique and is generated during installation and update of the 3DS Requestor App on the Cardholder's device.
For information see the SDK App ID row in Table A.1, "EMV 3-D Secure Data Elements" in the *EMV 3DS Protocol Specification*.

The following Java code snippet shows the signature of the `getSDKAppID` method:

```
public String getSDKAppID()
```

getSDKAppID Parameters

None.

getSDKAppID Return Value

This method returns the SDK App ID as a string.

getSDKAppID Exceptions

None.

4.12.5 getSDKReferenceNumber

The `getSDKReferenceNumber` method shall return the SDK Reference Number.

The following Java code snippet shows the signature of the `getSDKReferenceNumber` method:

```
public String getSDKReferenceNumber()
```

getSDKReferenceNumber Parameters

None.

getSDKReferenceNumber Return Value

This method returns the SDK Reference Number as a string.

getSDKReferenceNumber Exceptions

None.

4.12.6 getSDKEphemeralPublicKey

The `getSDKEphemeralPublicKey` method shall return the SDK Ephemeral Public Key. An ephemeral key pair is used to establish a secure session between the 3DS SDK and the ACS. During each transaction, the `createTransaction` method generates a fresh ephemeral key pair and the `getSDKEphemeralPublicKey` method returns the public key component of the same as a String representation of a JWK object.
For information about the SDK ephemeral public key, see the SDK Ephemeral Public Key row in Table A.1, "EMV 3-D Secure Data Elements" in the *EMV 3DS Protocol Specification*.

The following Java code snippet shows the signature of the `getSDKEphemeralPublicKey` method:

```
public String getSDKEphemeralPublicKey()
```

getSDKEphemeralPublicKey Parameters

None.

getSDKEphemeralPublicKey Return Value

The `getSDKEphemeralPublicKey` method returns the public key component of the ephemeral key pair as a String representation of a JWK object.

getSDKEphemeralPublicKey Exceptions

None.

4.12.7 getMessageVersion

The `getMessageVersion` method shall return the protocol version that is used for the transaction.

The SDK receives the protocol version as a parameter in the `createTransaction` method and determines whether it supports the version.
If the SDK does not receive the protocol version as a parameter in the `createTransaction` method, then it returns the latest version that it supports. For information about protocol version lookup support, refer to ***[Req 68]***.

The following Java code snippet shows the signature of the `getMessageVersion` method:

```
public String getMessageVersion()
```

getMessageVersion Parameters

None.

getMessageVersion Return Value

This method returns the protocol version as a string.

getMessageVersion Exceptions

None.

4.13 Class ErrorMessage

The `ErrorMessage` class shall represent an error message that is returned by the ACS to the 3DS SDK or an error message that is generated by the 3DS SDK to be returned to the ACS. For more information about error messages, refer to Table A.4: Error Code, Error Description, and Error Detail and Table B.10: Error Message Data Elements in the *EMV 3DS Protocol Specification*.

The following Java code snippet shows the definition of the `ErrorMessage` class:

```
public class ErrorMessage {

      public ErrorMessage(...)
      public String getTransactionID()
      public String getErrorCode()
      public String getErrorDescription()
      public String getErrorDetails()

}
```

Table 4.81 summarizes the methods that shall be provided by the `ErrorMessage` class.

Table 4.81: ErrorMessage Class Methods

Method	Description
`ErrorMessage`	Constructs an `ErrorMessage` object.
`getTransactionID`	Returns the Transaction ID.
`getErrorCode`	Returns the error code.
`getErrorDescription`	Returns the error description.
`getErrorDetails`	Returns the error details.

4.13.1 ErrorMessage

The `ErrorMessage` constructor shall create an `ErrorMessage` object.

The following Java code snippet shows the signature of the `ErrorMessage` constructor:

```
public ErrorMessage(String transactionID, String errorCode, String
errorDescription, String errorDetail)
```

ErrorMessage Parameters

Table 4.82: ErrorMessage Parameters

Parameter	Mandatory?	Description
`transactionID`	Yes	Transaction ID.
`errorCode`	Yes	Error code.
`errorDescription`	Yes	Text describing the error.
`errorDetail`	No	Additional error details.

ErrorMessage Exceptions

None.

4.13.2 getTransactionID

The `getTransactionID` method shall return the Transaction ID. The *EMV 3DS Protocol Specification* defines the Transaction ID.

The following Java code snippet shows the signature of the `getTransactionID` method:

```
public String getTransactionID()
```

getTransactionID Parameters

None.

getTransactionID Return Value

This method returns the Transaction ID as a string.

getTransactionID Exceptions

None.

4.13.3 getErrorCode

The `getErrorCode` method shall return the error code.

The following Java code snippet shows the signature of the `getErrorCode` method:

```
public String getErrorCode()
```

getErrorCode Parameters

None.

getErrorCode Return Value

This method returns the error code as a string.

getErrorCode Exceptions

None.

4.13.4 getErrorDescription

The `getErrorDescription` method shall return text describing the error. The *EMV 3DS Protocol Specification* defines error descriptions for a transaction.

The following Java code snippet shows the signature of the `getErrorDescription` method:

```
public String getErrorDescription()
```

getErrorDescription Parameters

None.

getErrorDescription Return Value

This method returns the error description as a string.

getErrorDescription Exceptions

None.

4.13.5 getErrorDetails

The `getErrorDetails` method shall provide error details. The *EMV 3DS Protocol Specification* defines error details for a transaction.

The following Java code snippet shows the signature of the `getErrorDetails` method:

```
public String getErrorDetails()
```

getErrorDetails Parameters

None.

getErrorDetails Return Value

This method returns error details as a string.

getErrorDetails Exceptions

None.

4.14 Class CompletionEvent

The `CompletionEvent` class shall hold data about completion of the challenge process.

The following Java code snippet shows the definition of the `CompletionEvent` class:

```
public class CompletionEvent {
```

```
     public CompletionEvent(...)
     public String getSDKTransactionID()
     public String getTransactionStatus()
}
```

Table 4.83 summarizes the methods that shall be provided by the `CompletionEvent` class.

Table 4.83: CompletionEvent Class Methods

Method	Description
`CompletionEvent`	Constructs an object with the specified inputs.
`getSDKTransactionID`	Returns the SDK Transaction ID. The *EMV 3DS Protocol Specification* defines the SDK Transaction ID.
`getTransactionStatus`	Returns the transaction status that was received in the final CRes.

4.14.1 CompletionEvent

The `CompletionEvent` constructor shall create an object with the specified inputs.

The following Java code snippet shows the signature of the `CompletionEvent` constructor:

```
public CompletionEvent(String sdkTransactionID, String
transactionStatus)
```

CompletionEvent Parameters

Table 4.84: CompletionEvent Parameters

Parameter	Mandatory?	Description
`sdkTransactionID`	Yes	Transaction ID of the 3DS SDK.
`transactionStatus`	Yes	Transaction status that was received in the CRes.

CompletionEvent Return Value

None.

CompletionEvent Exceptions

None.

4.14.2 getSDKTransactionID

The `getSDKTransactionID` method shall return the 3DS SDK transaction ID.
The *EMV 3DS Protocol Specification* defines this transaction ID.

The following Java code snippet shows the signature of the `getSDKTransactionID` method:

```
public String getSDKTransactionID()
```

getSDKTransactionID Parameters

None.

getSDKTransactionID Return Value

This method returns the 3DS SDK Transaction ID as a string.

getSDKTransactionID Exceptions

None.

4.14.3 getTransactionStatus

The `getTransactionStatus` method shall return the transaction status that was received by the 3DS SDK in the final CRes.

The following Java code snippet shows the signature of the `getTransactionStatus` method:

```
public String getTransactionStatus()
```

getTransactionStatus Parameters

None.

getTransactionStatus Return Value

This method returns the transaction status as a String.

getTransactionStatus Exceptions

None.

4.15 Class RuntimeErrorEvent

The `RuntimeErrorEvent` class shall hold details of run-time errors that are encountered by the 3DS SDK during authentication.

Note: A run-time error is not the same as a protocol error. For information about protocol errors, refer to Class ProtocolErrorEvent.

The implementer shall incorporate code that handles run-time errors. The following are examples of run-time errors:

- ACS is unreachable.
- Unparseable message.

- Network issues.

The following Java code snippet shows the definition of the `RuntimeErrorEvent` class:

```
public class RuntimeErrorEvent{

    public RuntimeErrorEvent(...)

    public String getErrorCode()

    public String getErrorMessage()

}
```

Table 4.85 summarizes the methods that shall be provided by the `RuntimeErrorEvent` class.

Table 4.85: RuntimeErrorEvent Class Methods

Method	Description
`RuntimeErrorEvent`	Constructs a `RuntimeErrorEvent` object.
`getErrorCode`	Returns the implementer-specific error code. **Note: As the 3DS SDK implementer, you define this error code.**
`getErrorMessage`	Returns details about the error.

4.15.1 RuntimeErrorEvent

The `RuntimeErrorEvent` constructor shall create an object with the specified inputs.

The following Java code snippet shows the signature of the `RuntimeErrorEvent` constructor:

```
public RuntimeErrorEvent(String errorCode, String errorMessage)
```

RuntimeErrorEvent Parameters

Table 4.86: RuntimeErrorEvent Parameters

Parameter	Mandatory?	Description
`errorCode`	No	Implementer-specific error code. **Note: As the 3DS SDK implementer, you set this error code.**

Parameter	Mandatory?	Description
`errorMessage`	Yes	Error message.

RuntimeErrorEvent Exceptions

None.

4.15.2 getErrorMessage

The `getErrorMessage` method shall return the error message.

The following Java code snippet shows the signature of the `getErrorMessage` method:

```
public String getErrorMessage()
```

getErrorMessage Parameters

None.

getErrorMessage Return Value

This method returns the error message as a string.

getErrorMessage Exceptions

None.

4.15.3 getErrorCode

The `getErrorCode` method shall return the implementer-specific error code. As the 3DS SDK implementer, you define this error code.

The following Java code snippet shows the signature of the `getErrorCode` method:

```
public String getErrorCode() {
    …
}
```

getErrorCode Parameters

None.

getErrorCode Return Value

This method returns the implementer-specific error code as a string. As the 3DS SDK implementer, you define this error code.

getErrorCode Exceptions

None.

4.16 Class ProtocolErrorEvent

In the 3DS SDK context, a protocol error is any error message that is returned by the ACS or an error message that is generated by the 3DS SDK to be returned to the ACS. The `ProtocolErrorEvent` class shall represent an error message of this type. The 3DS SDK sends the error code and details from this error message as part of the notification to the 3DS Requestor App.

Note: This error message is not a run-time error that is encountered by the 3DS SDK. For information about run-time errors, refer to Class RuntimeErrorEvent.

For more information about error messages, refer to Section A.5.5, "Error Code, Error Description, and Error Details" and Table B.10, "Error Message Data Elements" in the *EMV 3DS Protocol Specification*.

The following Java code snippet shows the definition of the `ProtocolErrorEvent` class:

```
public class ProtocolErrorEvent {

    public ProtocolErrorEvent(...)

    public ErrorMessage getErrorMessage()

    public String getSDKTransactionID()

}
```

Table 4.87 summarizes the methods that shall be provided by the `ProtocolErrorEvent` class.

Table 4.87: ProtocolErrorEvent Class Methods

Method	Description
`ProtocolErrorEvent`	Constructs a `ProtocolErrorEvent` object.
`getErrorMessage`	Returns the error message.
`getSDKTransactionID`	Returns the SDK Transaction ID.

4.16.1 ProtocolErrorEvent

The `ProtocolErrorEvent` constructor shall create an object with the specified inputs.

The following Java code snippet shows the signature of the `ProtocolErrorEvent` constructor:

```
public ProtocolErrorEvent(String sdkTransactionID,
ErrorMessage errorMessage)
```

ProtocolErrorEvent Parameters

Table 4.88: ProtocolErrorEvent Parameters

Parameter	Mandatory?	Description
`sdkTransactionID`	Yes	SDK Transaction ID.
`errorMessage`	Yes	Error message.

ProtocolErrorEvent Exceptions

None.

4.16.2 getErrorMessage

The `getErrorMessage` method shall return the error message.

The following Java code snippet shows the signature of the `getErrorMessage` method:

```
public String getErrorMessage()
```

getErrorMessage Parameters

None.

getErrorMessage Return Value

This method returns the error message as a string.

getErrorMessage Exceptions

None.

4.16.3 getSDKTransactionID

The `getSDKTransactionID` method shall return the SDK Transaction ID.

The following Java code snippet shows the signature of the `getSDKTransactionID` method:

```
public String getSDKTransactionID()
```

getSDKTransactionID Parameters

None.

getSDKTransactionID Return Value

This method returns the SDK Transaction ID as a string.

getSDKTransactionID Exceptions

None.

4.17 Class Warning

The `Warning` class shall represent a warning that is produced by the 3DS SDK while performing security checks during initialization.
For information about the security checks, see Table 8.2.

The following Java code snippet shows the definition of the `Warning` class:

```
public class Warning {

    public enum Severity {LOW, MEDIUM, HIGH)
    public Warning(...)
    public String getID()
    public String getMessage()
    public Severity getSeverity()

}
```

Table 4.89 summarizes the methods that shall be provided by the `Warning` class.

Table 4.89: Warning Class Methods

Method	Description
`Warning`	Constructs a Warning object.
`getID`	Returns the warning ID.
`getMessage`	Returns the warning message.
`getSeverity`	Returns the severity level of the warning.

4.17.1 Warning

The `Warning` constructor shall create an object with the specified inputs.

The following Java code snippet shows the signature of the `Warning` constructor:

```
public Warning(String id, String message, Severity severity)
```

Warning Parameters

Table 4.90: Warning Parameters

Parameter	Mandatory?	Description
`id`	Yes	Warning ID.

Parameter	Mandatory?	Description
`message`	Yes	Warning message.
`severity`	Yes	Warning severity level.

Warning Exceptions

None.

4.17.2 getID

The `getID` method shall return the warning ID.

The following Java code snippet shows the signature of the `getID` method:

```
public String getID()
```

getID Parameters

None.

getID Return Value

This method returns the warning ID as a string.

getID Exceptions

None.

4.17.3 getMessage

The `getMessage` method shall return the warning message.

The following Java code snippet shows the signature of the `getMessage` method:

```
public String getMessage()
```

getMessage Parameters

None.

getMessage Return Value

This method returns the warning message as a string.

getMessage Exceptions

None.

4.17.4 getSeverity

The `getSeverity` method shall return the severity level of the warning produced by the 3DS SDK.
The following Java code snippet shows the signature of the `getSeverity` method:

```
public Severity getSeverity()
```

getSeverity Parameters

None.

getSeverity Return Value

This method returns the severity level of the warning as a `Severity` enum type.

getSeverity Exceptions

None.

4.18 Class InvalidInputException

The `InvalidInputException` class shall represent a run-time exception that occurs due to one of the following reasons:

- Parameter value is mandatory, but was not provided.
- Parameter value does not conform to the specified format.
- Parameter value exceeds the maximum limit.
- Parameter value does not meet the minimum length criteria.

The following Java code snippet shows the definition of the `InvalidInputException` class:

```
public class InvalidInputException extends RuntimeException {

    public InvalidInputException(String message, Throwable cause)

}
```

Table 4.91 summarizes the methods that shall be provided by the `InvalidInputException` class.

Table 4.91: InvalidInputException Class Methods

Method	Description
`InvalidInputException`	Creates an `InvalidInputException` object.

4.18.1 InvalidInputException

The `InvalidInputException` constructor shall create an object with the specified error details.
The following Java code snippet shows the signature of the `InvalidInputException` constructor:

```
public InvalidInputException(String message, Throwable cause)
```

InvalidInputException Parameters

Table 4.92: InvalidInputException Parameters

Parameter	Mandatory?	Description
`message`	Yes	Description of the exception.
`cause`	Conditional (mandatory only if the platform supports it)	Cause of the exception.

4.19 Class SDKAlreadyInitializedException

The `SDKAlreadyInitializedException` class shall represent an exception that shall be thrown if the 3DS SDK instance has already been initialized.

The following Java code snippet shows the definition of the `SDKAlreadyInitializedException` class:

```
public class SDKAlreadyInitializedException extends
RuntimeException {

    public SDKAlreadyInitializedException(String message, Throwable
cause) {
      …
    }

}
```

Table 4.93 summarizes the methods that shall be provided by the `SDKAlreadyInitializedException` class.

Table 4.93: SDKAlreadyInitializedException Class Methods

Method	Description
`SDKAlreadyInitializedException`	Creates an `SDKAlreadyInitializedException` object.

4.19.1 SDKAlreadyInitializedException

The `SDKAlreadyInitializedException` constructor shall create an object with the specified error details.

The following Java code snippet shows the signature of the `SDKAlreadyInitializedException` constructor:

```
public SDKAlreadyInitializedException (String message, Throwable
cause)
```

SDKAlreadyInitializedException Parameters

Table 4.94: SDKAlreadyInitializedException Parameters

Parameter	Mandatory?	Description
`message`	Yes	Description of the exception.
`cause`	Conditional (mandatory only if the platform supports it)	Cause of the exception.

4.20 Class SDKNotInitializedException

The `SDKNotInitializedException` class shall represent an exception that shall be thrown if the 3DS SDK has not been initialized.
The 3DS SDK is initialized by calling the `initialize` method on the `ThreeDS2Service` object.

The following Java code snippet shows the definition of the `SDKNotInitializedException` class:

```
public class SDKNotInitializedException extends RuntimeException {

    public SDKNotInitializedException(String message, Throwable
cause)

}
```

Table 4.95 summarizes the methods that shall be provided by the `SDKNotInitializedException` class.

Table 4.95: SDKNotInitializedException Class Methods

Method	Description
`SDKNotInitializedException`	Creates an `SDKNotInitializedException` object.

4.20.1 SDKNotInitializedException

The `SDKNotInitializedException` constructor shall create an object with the specified error details.
The following Java code snippet shows the signature of the `SDKNotInitializedException` constructor:

```
public SDKNotInitializedException (String message, Throwable cause)
```

SDKNotInitializedException Parameters

Table 4.96: SDKNotInitializedException Parameters

Parameter	Mandatory?	Description
`message`	Yes	Description of the exception.
`cause`	Conditional (mandatory only if the platform supports it)	Cause of the exception.

4.21 Class SDKRuntimeException

This exception shall be thrown if an internal error is encountered by the 3DS SDK.

The following Java code snippet shows the definition of the `SDKRuntimeException` class:

```
public class SDKRuntimeException extends RuntimeException {

    public SDKRuntimeException(...)
    public String getErrorCode();
}
```

Table 4.97 summarizes the methods that shall be provided by the `SDKRuntimeException` class.

Table 4.97: SDKRuntimeException Class Methods

Method	Description
`SDKRuntimeException`	Creates an `SDKRuntimeException` object.
`getErrorCode`	Returns an implementer-specific error code.

4.21.1 SDKRuntimeException

The `SDKRuntimeException` constructor shall create an object with the specified error details.

The following Java code snippet shows the signature of the `SDKRuntimeException` constructor:

```
public SDKRuntimeException(String message, String errorCode,
Throwable cause)
```

SDKRuntimeException Parameters

Table 4.98: SDKRuntimeException Parameters

Parameter	Mandatory?	Description
`message`	Yes	Description of the exception.
`errorCode`	No	Implementer-specific error code. **Note: As the 3DS SDK implementer, you define this error code.**
`cause`	Conditional (mandatory only if the platform supports it)	Cause of the exception.

4.21.2 getErrorCode

The `getErrorCode` method shall return the implementer-specific error code.

The following Java code snippet shows the signature of the `getErrorCode` method:

```
public String getErrorCode()
```

getErrorCode Parameters

None.

getErrorCode Return Value

This method returns the error code as a string.

getErrorCode Exceptions

None.

4.22 Enum Severity

The `Severity` enum shall define the severity levels of warnings produced by the 3DS SDK while conducting security checks during initialization.

The following Java code snippet shows the definition of the `Severity` enum:

```
public enum Severity {LOW, MEDIUM, HIGH)
```

Table 4.99 summarizes the severity levels that shall be defined by the `Severity` enum.

Table 4.99: Severity Enum

Severity Level	Description
`LOW`	A low-severity warning.
`MEDIUM`	A medium-severity warning.
`HIGH`	A high-severity warning.

4.23 Enum ButtonType

The `ButtonType` enum shall define the button type.

The following Java code snippet shows the definition of the `ButtonType` enum:

```
public enum ButtonType {VERIFY, CONTINUE, NEXT, CANCEL, RESEND}
```

Table 4.100 summarizes the button types that shall be defined by the `ButtonType` enum.

Table 4.100: ButtonType Enum

Button Type	Description
`VERIFY`	Verify button
`CONTINUE`	Continue button
`NEXT`	Next button
`CANCEL`	Cancel button
`RESEND`	Resend button

5 Message Processing

This chapter provides information about the role of the 3DS SDK in the authentication and challenge flows.

5.1 Authentication

To request authentication for a transaction, the 3DS Requestor App collects some data elements from the 3DS SDK and sends them to the 3DS Server over a secure link. The 3DS Server uses this information to create an Authentication Request (AReq) message that is to be sent to the DS. The DS forwards this message to the ACS.

In Table 5.1, the inclusion criteria can be R (Required) or C (Conditional). The data mentioned here is the minimum set that the 3DS SDK shall provide to the 3DS Requestor App. 3DS SDK implementations have the flexibility to include additional data as required.

Table 5.1: Data Elements Generated by 3DS SDK for Authentication

Data Elements	Inclusion Criterion	Details
Device Information (encrypted)	C	Cardholder's device identification data. If there is no market or regional mandate to restrict sending this information, then this field shall be collected and sent. The data is encrypted using the DS Public Key and is in JWE format. This data element is part of ***[Req 16]*** in Table 3.3.
SDK Reference number	R	Implementer and version of the 3DS SDK that is integrated with the 3DS Requestor App, assigned by EMVCo through the EMV 3-D Secure Testing and Approvals process when the 3DS SDK is approved. This reference number is a security asset of the 3DS SDK, and it shall be securely stored. During each transaction, this reference number shall be securely retrieved by the 3DS SDK and returned to the 3DS Requestor App. This data element is part of ***[Req 16]*** in Table 3.3.

Data Elements	Inclusion Criterion	Details
SDK Transaction ID	R	Transaction identifier assigned by the 3DS SDK to uniquely identify each transaction. The 3DS SDK uses a secure random function to generate a Transaction ID in UUID format. This data element is part of ***[Req 16]*** in Table 3.3.
SDK App ID	R	Unique ID that is generated during installation and update of the 3DS Requestor App on the Cardholder's device. The 3DS SDK uses a secure random function to generate the App ID in UUID format. This data element is part of ***[Req 16]*** in Table 3.3.
SDK Ephemeral Public Key	R	Public key component of the temporary key pair that is generated by the 3DS SDK and used to establish session keys between the 3DS SDK and the ACS. This data element is part of ***[Req 16]*** in Table 3.3.
Message Version	R	Protocol version that is supported by the 3DS SDK and used for the transaction. This data element is part of ***[Req 16]*** in Table 3.3.

If a challenge is requested by the Issuer, then to perform the Challenge Flow, the 3DS SDK uses some data elements from the ARes. The 3DS Requestor App passes these elements to the 3DS SDK.
Table 5.2 lists the minimum set of data elements that are required by the 3DS SDK to perform a Challenge Flow. For more information about each element, refer to Table A.1, "EMV 3-D Secure Data Elements" in the *EMV 3DS Protocol Specification*.

Table 5.2: Data Elements Required by the 3DS SDK for Authentication

Data Elements	Inclusion Criterion	Details
ACS Transaction ID	C	Transaction identifier assigned by the ACS. It is generated only if a challenge is requested. This data element is part of ***[Req 13]*** in Table 3.2.
ACS Signed Content	C	Contains the JWS object created by the ACS for the ARes. This data element is part of ***[Req 13]*** in Table 3.2.
3DS Server Transaction ID	C	The 3DS Server Transaction ID uniquely identifies a transaction within all messages (AReq/ARes, CReq/CRes, and RReq/RRes) that are exchanged during the authentication process. This data element is part of ***[Req 13]*** in Table 3.2.

5.2 Challenge Processing

In the Challenge Flow, the 3DS SDK communicates directly with the ACS over a secure link in order to display the challenge UI to the Cardholder, as directed by the ACS. The 3DS SDK shall support one or both of the following formats to display the challenge content sent by the ACS:

- Native UI by using platform-specific display elements, such as button, textbox, text label and so on.
- HTML UI by using WebView.

The Cardholder's response is encrypted and MACed by the 3DS SDK using session keys (pre-established with the ACS) and then forwarded to the ACS. For details about the Challenge Request (CReq), Challenge Response (CRes) and data elements, refer to the *EMV 3DS Protocol Specification*.

6 Device Identification

Device identification is used to uniquely identify mobile devices in the 3-D Secure ecosystem. The `initialize` method of the `ThreeDS2Service` interface implemented in the 3DS SDK collects the information required for device identification. This information is then sent to the 3DS Requestor App in JSON format. The 3DS Requestor App passes this information to the 3DS Server. The 3DS Server uses this information to create an AReq.

For information about the device identification parameters that are collected by the 3DS SDK, refer to *EMV 3DS SDK Device Information*.

7 User Interface

Note: The information that is provided in this chapter is only a subset of the UI-related information that the 3DS SDK implementer would require while developing the SDK. For detailed information about all other aspects of the UI including security, refer to Chapter 4, "EMV 3-D Secure User Interface Templates, Requirements, and Guidelines" in the *EMV 3DS Protocol Specification*.

The ACS component evaluates the risk of each transaction. If the ACS detects a suspicious transaction or recognizes a situation that requires a challenge for authentication, it advises the 3DS Requestor App to apply the Challenge Flow.
[Req 34] The 3DS SDK shall render the UI for the Challenge Flow in one of the following formats:

- HTML UI, in which the Cardholder challenge is applied by using an HTML-based user interface.
- Native UI, in which the Cardholder challenge is applied by using a native user interface.

The UI format to be displayed by the 3DS SDK is determined based on the ACS UI Type value obtained as part of the CRes message. For information about the ACS UI Type, see the ACS UI Type row in Table A.1, “EMV 3-D Secure Data Elements” in the *EMV 3DS Protocol Specification*.

Note: The implementer shall consider regional Accessibility rules while developing the user interface.

Note: All Device Rendering Options supported shall be supported by the SDK. This is also mentioned in *[Req 314]* in the *EMV 3DS Protocol Specification.*

[Req 35] The 3DS SDK shall support the UI element types listed in Table 7.1.

Table 7.1: UI Element Types

UI Element Type	ACS UI Type	Example
Text	01	SMS OTP field
Single Select	02	Where do you want us to send the OTP? Select one of the following: • Your mobile: **** **** 329 • Your email address: s******k**@g***.com
Multi Select	03	Which cities have you lived in? Portland, Oregon Chicago, Illinois Dallas, Texas

UI Element Type	ACS UI Type	Example
OOB	04	We have sent a link to your email account s******k**@g***.com. Please click that link to authenticate yourself.
HTML	05	Fully-formed HTML snippet that shall be used to display the Challenge UI.

7.1 HTML UI

[Req 36] The HTML UI shall be rendered in a web view controlled by the 3DS SDK.
During the Challenge Flow, the ACS provides the content that is displayed to the Cardholder. This content is provided as a fully formed HTML snippet. The ACS encrypts the HTML snippet and transmits it to the 3DS SDK in the CRes message. The 3DS SDK decrypts the HTML snippet and use a web view to display the content on the mobile device. The 3DS SDK shall display the HTML exactly as provided by the Issuer.
The **ACS HTML** field in the CRes message holds the HTML snippet to be displayed to the Cardholder.

[Req 37] The Cardholder data (response) shall be captured and sent to the ACS in the CReq message. The SDK shall not modify this response data before passing it in the **Challenge HTML Data Entry** field of the CReq message. This field holds the Cardholder's challenge response.
When the Cardholder's response is returned as a parameter string, the form data is passed to the web view instance by triggering a location change to a specified (HTTPS://EMV3DS/challenge) URL with the challenge responses appended to the location URL as query parameters (for example, HTTPS://EMV3DS/challenge?city=Pittsburgh). The web view instance, because it monitors URL changes, receives the Cardholder's responses as query parameters.

[Req 38] The header for the HTML UI pages that are rendered by the 3DS SDK shall not occupy more than 10% of the screen height.

[Req 64] If the CRes message contains the **ACS HTML Refresh** field, then the 3DS SDK shall display the HTML contained in this field when the app is moved to the foreground.

7.2 Native UI

[Req 39] The Native UI shall be rendered and controlled by the 3DS SDK.
The Native UI integrates into the 3DS Requestor App UI to facilitate a consistent user experience. The Native UI has a similar look and feel as the 3DS Requestor's App with the authentication content provided by the Issuer.
This format also allows for Issuer and Payment System branding. The 3DS SDK controls the rendering of the UI such that the authentication pages inherit the 3DS Requestor's UI design elements. The CRes message carries the information that is required to render the UI.
The **Challenge Selection Information** field in the CRes message holds the selection information that is presented to the Cardholder if the challenge type is single select or multi

select. UI text, such as label names, questions, and help text, is sent in a JSON array. The 3DS SDK parses the UI text and then displays it in the user interface.

[Req 40] The Cardholder data (response) shall be captured and sent to the ACS in the CReq message. The **Challenge Data Entry** field in the CReq message holds the Cardholder's challenge response.

[Req 65] If the CRes message contains the **Challenge Additional Information Text** field, then the 3DS SDK shall replace the **Challenge Information Text** and **Challenge Information Text Indicator** fields with the contents of the **Challenge Additional Information Text** field when the app is moved to the foreground.

7.2.1 Input and Output Formats for Native UI

The following sections describe the format of the **Challenge Selection Information** field and **Challenge Data Entry** field for each challenge type.

Single Text Input

The Single Text Input challenge type is used to prompt for and collect a single-text response from the Cardholder.

For example: Enter the OTP that we have sent to your registered mobile number.
In this example, the following are the field formats:
Challenge Selection Information field: No format for this challenge type
Challenge Data Entry field: 432525

Single Select

The Single Select challenge type is used to prompt for and collect the Cardholder's selection of a single item from multiple items.
For example: Where do you want us to send the OTP? Select one of the following:

- Your mobile: **** **** 329
- Your email address: s******k**@g***.com

In this example, the following are the field formats:

Challenge Selection Information field (JSON):

```
{“Challenge Selection Information”: [
                      {"mobile": "**** **** 329"},
                      {"email": " s******k**@g***.com"}
             ]
}
```

Challenge Data Entry field: mobile

Multi Select (Checkbox)

The Multi Select challenge type is used to prompt for and collect the Cardholder's selection of a subset of items from multiple items. For example: Select the cities that you have lived in:

- Chicago, Illinois
- St Louis, Missouri
- Portland, Oregon

In this example, the following are the field formats:

Challenge Selection Information field (JSON):

```
{“Challenge Selection Information”: [
                      {"chicago_illinois": "Chicago, Illinois"},
                      {"st_louis_missouri": "St Louis, Missouri"},
                      {"portland_oregon": "Portland, Oregon”}
             ]
}
```

Challenge Data Entry field (comma-separated):

```
chicago_illinois,portland_oregon
```

Out-of-Band

The Out-of-Band challenge type is used to direct the Cardholder to perform out-of-band authentication. The 3DS SDK shall not collect any information from the Cardholder in this challenge type. Instead, the 3DS SDK displays a user interface containing instructions explaining the authentication process to the Cardholder. These instructions will come from the ACS.

7.2.2 UI Templates for Native UI

[Req 41] The 3DS SDK shall have predefined UI templates for the challenge screens for each challenge type. Based on the Issuer’s choice of template for each challenge type (as determined by the ACS UI type element in the CRes message), the 3DS SDK shall render the UI for the challenge screens. UI elements should be placed in a logical order in the templates. The placement should conform to the standard UI best practices of the country or region where the 3DS SDK will be used.

The 3DS SDK should fine-tune the rendering of the UI on the Cardholder device. The 3DS SDK can optimise the content provided by the Issuer (for example, by removing an extra line feed that would cause scrolling). The formatting provided in the CRes message need not be exactly what is displayed to the Cardholder.

For more information about Native UI templates, refer to Section 4.2.2, “Native UI Templates” in the *EMV 3DS Protocol Specification*.

Note: The use of a carriage return in any UI data element is permitted only as specified in Table A.1, “EMV 3-D Secure Data Elements” in the *EMV 3DS Protocol Specification*.

7.3 UI Elements Customization

[Req 42] The 3DS SDK shall allow customization of the following UI elements on the challenge screens. The information required for UI customization is passed to the 3DS SDK during initialization.

- 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

For more information about UI elements customization, see Class UiCustomization.

8 SDK Security

This chapter describes an overview of the basic security requirements that are to be implemented by a 3DS SDK. The PCI 3DS SDK security requirements are published on the PCI SSC website. Compliance to the PCI 3DS SDK Security Standard is at the discretion of the applicable payment brand.

8.1 Security Goals of the 3DS SDK

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 3DS SDK.

There are three primary security goals for the 3DS SDK:

- Protect sensitive cardholder information while being transferred, being processed, or at rest. A cardholder's response to an authentication challenge is an example of sensitive cardholder information.
- Protect sensitive 3-D Secure system information while being transferred, being 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 3DS SDK.

The 3-D Secure architecture is aimed at addressing most of the threats that stand in the path of these security goals. A securely implemented 3DS SDK can strengthen this line of defence.

There is no guaranteed approach to fully secure an IT system or network. Mobile devices are no exception to this rule. The security requirements are aimed at making attacks on the SDK difficult, expensive and time-consuming for any attacker.

8.2 SDK Initialization Security Checks

Table 8.2 describes the checks that the 3DS SDK shall conduct during initialization. The SDK shall make the result of the checks available as a list of warnings to the 3DS Requestor App and include them in the Device Information JSON data with key as "SW". For more information, refer to *EMV 3DS SDK Device Information*.

Table 8.1: 3DS SDK Initialization Security Checks

Security Warning ID	Description	Severity Level
SW01	The device is jailbroken.	HIGH
SW02	The integrity of the SDK has been tampered.	HIGH

Security Warning ID	Description	Severity Level
SW03	An emulator is being used to run the App.	HIGH
SW04	A debugger is attached to the App.	MEDIUM
SW05	The OS or the OS version is not supported.	HIGH

8.3 3DS SDK Versioning Requirements and Protocol Versioning Support

[Req 58] 3DS SDK implementers shall apply a versioning system for the SDK that they develop to differentiate one version from another. For better 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>.
The version number shall be securely stored in the 3DS SDK code to prevent any modification. It shall be used to determine whether the SDK requires an update. The SDK implementer shall implement the mechanism and frequency of the version check.

[Req 59] On successful approval by the EMVCo testing and approval process, a unique SDK Reference Number is assigned to the SDK. The SDK Reference Number uniquely identifies the SDK implementer and the protocol version that was tested.
The SDK Reference Number shall be securely stored in the 3DS SDK code to prevent any modification. During authentication, the 3DS SDK shall forward the SDK Reference Number to the 3DS Requestor App. The app then sends the SDK Reference Number to the 3DS Server, which forwards the SDK Reference Number to the DS (in the AReq message) for validation.

[Req 68] The SDK shall maintain a lookup of the protocol versions that it supports and use this lookup to identify the latest version.

It is recommended that the 3DS Requestor App always use the latest available version of the SDK.

8.4 3DS SDK – ACS Secure Channel

[Req 60] The 3DS SDK and the ACS shall apply the Diffie–Hellman key exchange protocol to establish keys for a secure channel for protecting CReq/CRes messages that are exchanged during the Challenge Flow. During a particular transaction, the 3DS SDK shall select the encryption and decryption algorithm (A128CBC-HS256 or A128GCM) for the Challenge Flow. The ACS uses the same algorithm for that transaction.

For more information, refer to Section 6.2.3, “Function J: 3DS SDK – ACS Secure Channel Set-up” and Section 6.2.4, “Function K: 3DS SDK – ACS (CReq, CRes) in the *EMV 3DS Protocol Specification*.

Annex A EMV 3DS SDK Predefined Data and Updates

A.1 3DS SDK Predefined Data

[Req 61] Predefined data refers to data that shall be bundled with the 3DS SDK. Table A.1 describes this predefined data.

Table A.1: Predefined Data

Predefined Data	Description
DS Public Certificate	A public certificate provided by the DS for encryption of device data.
CA Public Certificate of the DS-CA	CA public certificate (root) of the DS-CA. This certificate is used to validate the ACS signed content JWS object.
Card brand logos	Logos of participating card brands.
Region-specific configuration data	This data includes, for example, a flag specifying the device information that should not to be collected due to regional privacy laws.

A.2 Types of Changes That Require 3DS SDK Updates

[Req 62] The following types of changes shall require an update to the 3DS SDK:

- **Changes to the 3DS SDK binary**
  The 3DS SDK binary may change if any functional changes, bug fixes, security fixes, performance fixes, and so on are implemented.
- **Changes to the predefined data in the 3DS SDK binary**
  Refer to Table A.1 for the list of predefined data items.

All updates to the 3DS SDK shall be published through a software patch as part of an update to the 3DS Requestor App through secured and trusted channels, such as Google Play, Apple App Store, and Windows Phone Store.

Annex B EMV 3DS SDK Performance

It is recommended that the implementer ensures that the 3DS SDK is responsive and adheres to the best practices related to mobile device performance parameters. The following are examples of performance parameters:

- SDK binary size
- CPU usage
- Memory usage
- Battery consumption
- Response time of the interface functions

Annex C EMVCo Testing and Approval

[Req 63] The implemeneter shall submit the 3DS SDK to EMVCo for testing and approval. The testing and approval process assesses the functional and security aspects of the SDK against the requirements mentioned in this document and the *EMV 3DS Protocol Specification*.
After the SDK is tested and approved, EMVCo assigns a unique reference number to the SDK. The SDK shall include this reference number in the AReq message.
For more information about testing and approval, refer to the *EMV 3DS Protocol Specification*.

Annex D Code Samples

This annex provides examples that show the code elements of the 3DS SDK.

D.1 Code Sample for iOS

The following sample shows the code elements of the 3DS SDK on iOS:

```
//Create new instance of the VendorThreeDS2ServiceImpl, class that
//implements ThreeDS2Service protocol.
    let threeds2service: ThreeDS2Service =
            VendorThreeDS2ServiceImpl()

    //Prepare input parameters for initialize
    let configParam: ConfigParam = ConfigParam()

    let uiCustomization: UiCustomization = UiCustomization()
    let btnCustomization: ButtonCustomization =
            ButtonCustomization()
    btnCustomization.setTextCollor("#FF00FF")

    let toolbarCustomization: ToolbarCustomization =
            ToolbarCustomization()
    toolbarCustomization. setBackgroundColor ("#FF00FF")

    let lblCustomization: LabelCustomization =
            LabelCustomization()
    lblCustomization.setTextColor("#FF00FF")

    let textboxCustomization: TextBoxCustomization =
            TextBoxCustomization()
    txtboxCustomization.setTextColor("#FF00FF")

    uiCustomization.setButtonCustomization(btnCustomization,
            ButtonType.NEXT)
    uiCustomization. setToolbarCustomization
            (toolbarCustomization)
    uiCustomization.setLabelCustomization(lblCustomization)
    uiCustomization.setTextBoxCustomization(btnCustomization)

    let userLocale: String = "en_US"
    do {

        //Initialize the 3DSSDK
          try threedsservice.initialize(configParam, locale:
                  userLocale,uiCustomization: uiCustomization)

    }catch ThreeDS2Error.InvalidInput(let errorMessage){
    // handle the InvalidInput ErrorType
        return
```

```
    }catch ThreeDS2Error.SDKAlreadyInitialized(let errorMessage){
    // handle the SDKAlreadyInitialized ErrorType
       return
    }catch ThreeDS2Error.SDKRuntime(let errorMessage, let
            errorCode){
    // handle the SDKRuntime ErrorType
       return
    }catch {
        ...
        return
    }


//Create an instance of Transaction

    do {
        let directoryServerID = ...
        let messageVersion = ...
        let transaction: Transaction =
             threeds2service.createTransaction(directoryServerID,
             messageVersion)

    }catch ThreeDS2Error.InvalidInput(let errorMessage){
    // handle the InvalidInput ErrorType
       return
    }catch ThreeDS2Error.SDKNotInitialized(let errorMessage){
    // handle the SDKNotInitialized ErrorType
       return
    }catch ThreeDS2Error.SDKRuntime(let errorMessage, let
             errorCode){
    // handle the SDKRuntime ErrorType
       return
    }catch {
        ...
        return
    }

    //get handle to the progress showing view
    let sdkProgressView: ProgressDialog
    sdkProgressView = try transaction.getProgressView()
    sdkProgressView.start()


//get the Authentication Request Parameters like Device Info,
//SDKAppID and so on from the SDK
    do {
        let authRequestParams: AuthenticationRequestParameters
        authRequestParams = try
             transaction.getAuthenticationRequestParameters()
        let encryptedDeviceInfo: String =
             authRequestParams.deviceData
        let sdkTransactionID: String =
             authRequestParams.sdkTransactionID
        ...
```

```
    } catch ThreeDS2Error.SDKRuntime(let errorMessage, let
             errorCode){
    // handle the SDKRuntime ErrorType
        transaction.close()
        return

    }catch {
        ...
        return
    }

//Challenge Processing

    //Create challenge parameters object
    let challengeParameters = ChallengeParameters()

    //set the parameters to be sent to SDK
    let acsSignedContent = ...
    let acsRefNumber = ...
    let 3DSServerTransactionID = ...
    challengeParameters.acsSignedContent =  acsSignedContent
    challengeParameters.acsRefNumber = acsRefNumber
    challengeParameters.3DSServerTransactionID = 3DSServer
TransactionID
    ...

    // Create an instance of ChallengeStatusReceiver using
    // MerchantChallengeStatusReceiverImpl, class which implements
    // the ChallengeStatusReceiver protocol

    let challengeStatusReceiver  =
            MerchantChallengeStatusReceiverImpl()

    do {
        try transaction.doChallenge(challengeParameters,
               challengeStatusReceiver  : challengeStatusReceiver,
               timeOut: 5)

    }catch ThreeDS2Error.InvalidInput(let errorMessage){
    // handle the InvalidInput ErrorType
        transaction.close()
       return
    }catch {
       ...
       return
    }
    ...

// Custom Progress View Protocol
    public protocol ProgressDialog {
        func start()
        func stop()
    }
```

```
//Class implementing the ProgressDialog protocol
    public class SDKProgressDialog: UIView, ProgressDialog {
        var textLabel: UILabel!
        var activityIndicator: UIActivityIndicatorView!
        ...

   //override required methods


        public func  start(){
       //add “self” as subview to the viewcontroller wanting to
       //show the progress dialog, start the activityIndicator
       //animation here
           activityIndicator.startAnimation()
           ...
       }

        public func stop(){
        //stop the activityIndicator animation here
        //activityIndicator.stopAnimating()remove “self” from
        //super view
          ...
        }
    }

//Class implementing the ChallengeStatusReceiver protocol

    public class MerchantChallengeStatusReceiverImpl:
            ChallengeStatusReceiver {

          public func completed( e: CompletionEvent){
              ...
          }
          public func cancelled(){
              ...
          }

          public func timedout(){
              ...
          }

          public func protocolError(e: ProtocolErrorEvent){
              ...
          }

          public func runtimeError(e: RuntimeErrorEvent){
             ...
          }
    }
```

D.2 Code Sample for Android

The following sample shows the code elements of the 3DS SDK on Android:

```
//Create new instance of VendorThreeDS2ServiceImpl, class that
//implements ThreeDS2Service interface

    ThreeDS2Service threeDS2Service = new
            VendorThreeDS2ServiceImpl();

    //Create the configuration parameter object
    ConfigParam configParam = new ConfigParam(...);

    //Create the UI configuration object
    UiCustomization uiCustomization = new UiCustomization(…);

    ButtonCustomization btnCustomization = new
            ButtonCustomization();
    btnCustomization.setTextColor(“#FF00FF”);

    ToolbarCustomization toolbarCustomization = new
            ToolbarCustomization();
    toolbarCustomization. setBackgroundColor (“#FF00FF”);

    LabelCustomization lblCustomization = LabelCustomization();
    lblCustomization.setTextColor(“#FF00FF”);

    TextBoxCustomization txtboxCustomization =
            TextBoxCustomization()
    txtboxCustomization.setTextColor (“#FF00FF”);

    uiCustomization.setButtonCustomization(btnCustomization,
            ButtonType.NEXT);
    uiCustomization.setToolbarCustomization(toolbarCustomization);
    uiCustomization.setLabelCustomization(lblCustomization);
    uiCustomization.setTextBoxCustomization(txtboxCustomization);

    //Get the device’s locale
    String userLocale = "en_US";

    //Get the android application context
    Context applicationContext = ...;

    //Get the current activity instance
    Activity currentActivity = ...;

    try {

        //Initialize the SDK
          threeDS2Service.initialize(applicationContext,
                configParam, userLocale, uiCustomization);
```

```
    } catch (InvalidInputException e) {
       // handle the InvalidInputException
          return;
    } catch (SDKAlreadyInitializedException e) {
       // handle the SDKAlreadyInitializedException
          return;
    } catch (SDKRuntimeException e) {
      // handle the SDKRuntimeException
         return;
    }


//Create new instance of Transaction
    try {
          String directoryServerID = ...
          String messageVersion = ...
               Transaction transaction =
               threeDS2Service.createTransaction(directoryServerID
               , messageVersion);

    } catch (InvalidInputException e) {
        // handle the InvalidInputException
           return;
    } catch (SDKNotInitializedException e) {
        // handle the SDKNotInitializedException
           return;
    } catch (SDKRuntimeException e) {
        // handle the SDKRuntimeException
           return;
    }

    //get handle to the progress showing view
    ProgressDialog progressDialog;

    try {
          progressDialog =
                transaction.getProgressView(android.app.Activity);

    } catch (InvalidInputException e) {
        // handle the InvalidInputException
          transaction.close();
          return;
    }

//get the Authentication Request Parameters like Device Info,
//SDKAppID and so on from the SDK.
    try {
          AuthenticationRequestParameters authRequestParams =
              transaction.getAuthenticationRequestParameters();
          String encryptedDeviceInfo =
              authRequestParams.getDeviceData();
          String sdkTransactionID =
              authRequestParams.getSDKTransactionID();
          ...
```

```
    }catch (SDKRuntimeException e){
        // handle the SDKRuntimeException
           transaction.close();
           return;
    }

// Challenge Processing

    //Obtain the current activity instance
    //Create challenge parameters object
    ChallengeParameters challengeParameters = new
             ChallengeParameters();

    //Set challenge parameters
    String acsSignedContent = ...
    String acsRefNumber = ...
    String 3DSServerTransactionID = ...
    challengeParameters.setAcsSignedContent(acsSignedContent)
    challengeParameters.setAcsRefNumber(acsRefNumber);

challengeParameters.set3DSServerTransactionID(3DSServerTransaction
ID);
    ...

    //SDK to timeout in 5 minutes
    int timeOut = 5;

    //Begin the challenge flow

    try {
          transaction.doChallenge(currentActivity,
             challengeParameters,
              new ChallengeStatusReceiver () {

          @Override
          public void completed(CompletionEvent e) {

          }

         @Override
         public void cancelled() {

         }

         @Override
         public void timedout() {

         }

         @Override
         public void protocolError(ProtocolErrorEvent e) {

         }
```

```
        @Override
        public void runtimeError(RuntimeErrorEvent e) {
        }

     }, timeOut);

  } catch (InvalidInputException e) {
        // handle the InvalidInputException
        transaction.close();
        return;

  }
  …
```

D.3 Code Sample for Windows Phone

The following sample shows the code elements of the 3DS SDK on Windows Phone:

```
//Create new instance of VendorThreeDS2ServiceImpl, class that
//implements the ThreeDS2Service interface.

    ThreeDS2Service threeDS2Service = new
            VendorThreeDS2ServiceImpl();


    //Create the configuration parameter object
    ConfigParam configParam = new ConfigParam(...);

    //Create the UI configuration object
    UiCustomization uiCustomization = new UiCustomization(...);
    ButtonCustomization btnCustomization = new
            ButtonCustomization();
    btnCustomization.TextColor = “#FF00FF”;

    ToolbarCustomization toolbarCustomization = new
            ToolbarCustomization();
    toolbarCustomization. BackgroundColor = “#FF00FF”;

    LabelCustomization lblCustomization = LabelCustomization();
    lblCustomization.TextColor =“#FF00FF”;

    TextBoxCustomization txtboxCustomization =
            TextBoxCustomization()
    textboxCustomization.TextColor = “#FF00FF”;

    uiCustomization.SetButtonCustomization(btnCustomization,
            ButtonType.NEXT);
    uiCustomization.ToolbarCustomization = toolbarCustomization;
    uiCustomization.LabelCustomization = lblCustomization;
```

```
    uiCustomization.TextBoxCustomization = txtboxCustomization;

    //Get the device’s locale
    String userLocale = "en_US";

    try {

          //Initialize the SDK
          threeDS2Service.Initialize(configParam, userLocale,
                  uiCustomization);

    } catch (InvalidInputException e) {
          // handle the InvalidInputException
          return;
    } catch (SDKAlreadyInitializedException e) {
          // handle the SDKAlreadyInitializedException
          return;
    } catch (SDKRuntimeException e) {
          // handle the SDKRuntimeException
          return;
    }

    //Create new instance of Transaction
    try {

         String directoryServerID = ...
          String messageVersion = ...
          Transaction transaction =
              threeDS2Service.CreateTransaction(directoryServerID,
              messageVersion);

    } catch (InvalidInputException e) {
          // handle the InvalidInputException
          return;
    } catch (SDKNotInitializedException e) {
          // handle the SDKNotInitializedException
          return;
    } catch (SDKRuntimeException e) {
         // handle the SDKRuntimeException
         return;
    }

    //get handle to the progress showing view
    ContentDialog progressDialog;


    progressDialog = transaction.GetProgressView();


//get the Authentication Request Parameters like Device Info,
//SDKAppID and so on from the SDK.
    try {
          AuthenticationRequestParameters authRequestParams =
                  transaction.GetAuthenticationRequestParameters();
```

```
          String encryptedDeviceInfo =
                  authRequestParams.GetDeviceData();
          String sdkTransactionID =
                  authRequestParams.GetSDKTransactionID();
          ...

    }catch (SDKRuntimeException e){
          // handle the InvalidInputException
          transaction.Close();
          return;
    }


// Challenge Processing

    //Create challenge parameters object
    ChallengeParameters challengeParameters = new
            ChallengeParameters();

    //Set challenge parameters
    String acsSignedContent = ...
    String acsRefNumber = ...
    String 3DSServerTransactionID = ...
    challengeParameters.ACSSignedContent = acsSignedContent;
    challengeParameters.AcsRefNumber = acsRefNumber;
    challengeParameters.3DSServerTransactionID =
3DSServerTransactionID;
    ...

    //Create ChallengeStatusReceiver
    ChallengeStatusReceiver challengeStatusReceiver = new
            MerchantChallengeStatusReceiver();
    //SDK to timeout in 5 minutes
    int timeOut = 5;

    //Begin the challenge flow
    try {
          transaction.DoChallenge(challengeParameters,
                  challengeStatusReceiver, timeOut);

    } catch (InvalidInputException e) {
          // handle the InvalidInputException
          transaction.Close();
          return;
    }
    …

//Class implementing the ChallengeStatusReceiver interface

    class MerchantChallengeStatusReceiver :
            ChallengeStatusReceiver {

        public override void Completed(CompletionEvent e) {
        }
```

```
        public override void Cancelled() {
        }

        public override void Timedout() {
        }

        public override void ProtocolError(ProtocolErrorEvent e) {
        }

        public override void RuntimeError(RuntimeErrorEvent e) {
        }
    }
```

***** END OF DOCUMENT *****