evopayments.com

3-D Secure 2.0

3-D Secure as a Service Integration Guide

©2018 EVO Payments, Inc.

Table of Contents

Version History

Version	Date	Description of Changes
V1	11 May 2020	Initial version
V2	27 May 2020	> Removed PaymentAuthorizationResponse from required fields on QueryAuthenticationResponse > Updated Requests to include only Required fields > Added TransactionStatus to BankcardTransctionResponse > Changed ChallengeCancellationIndicator to a String > Added clarification on MethodCompletionIndicator > Updated Challenge Required by triggering off of TransactionStatus = "Challenge Required" > Removed inaccurate references to Authorize and AuthorizeAndCapture in the Method Data section > Removed Fallback functionality that will be added in a later release. > Added workflow diagram
V3	16 June 2020	> Added clarification on Frictionless Flow > Updated diagram for 3DSaaS Flow > Updated QueryAuthenticationResponse to QueryAuthenticationResults as it was implemented > Updated Request URIs > Updated Authenticate to include ThreeDSMerchantData > .35R2 Updates for Protocol Support and Exemptions > Updated ProtocolVersion Format
V4	17 June 2020	> Updated SIS URIs and ProtocolVersion > Updated request body for Card Range calls > Updated Failed Exemption workflow
V5	01 July 2020	> Added clarification on submitting an Exempted Transaction Request
V6	13 July 2020	> Updated RangeAction options for QueryCardRanges with Serial Number
V7	16 July 2020	> Application-Based Frictionless Authentication section added > Card on File for Non-Payment Transactions

		section added > Changed MerchantName field to RequestorName
V8	21 August 2020	> Card on File for Payment Transactions section added > Added clarification on retrieving Method Data > Updated Application-Based Frictionless Authentication sample Response
V9	06 October 2020	> Clarification on Exemptions for Card Brands added
V10	20 October 2020	> Decoupled Authentication section added > Data Only Authentication section added > New QueryCardRanges requests added
V11	1 December 2020	> Attempts Server Fallback scenario added > Fallback from version 2.2 to 2.1 scenario added

Overview

3-D Secure is a protocol developed to make online payments more secure through password authentication and cardholder verification. The new Card Scheme mandates are the next wave of 3-D Secure that will bring additional eCommerce security to EMV. These updates will be supported for the eService and TRON front-ends.

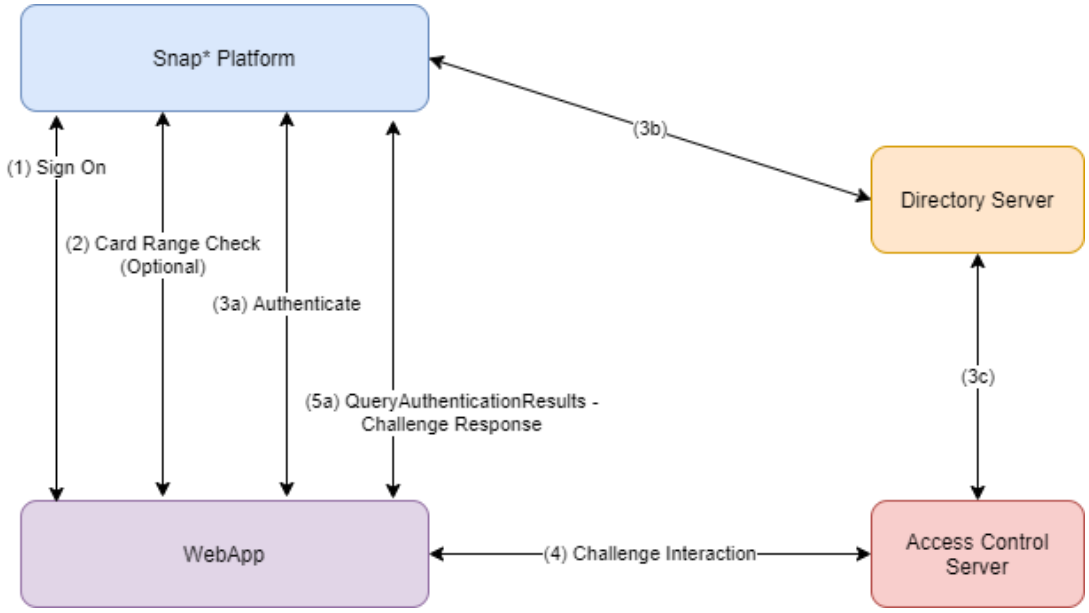
3-D Secure as a Value Added Service

For non-transacting Merchants, 3-D Secure is offered as a standalone value-added service. Merchant Applications will be able to submit 3-D Secure transactions solely for 3-D Secure Authentication purposes, in which the transaction will eventually be processed outside of the Snap* Platform. API calls have been added to the Snap* API. The Authentication request and the QueryAuthenticationResults request will only be used in the event that a Challenge is required.

Workflow

This workflow outlines the process for an integrator that is utilizing 3-D Secure as a standalone value-added service, in which the transaction will eventually be processed outside of the Snap* Platform. The workflow is broken out into two distinct parts: Authentication, which is done through Snap*, and Authorization, which is done elsewhere.

1. An initial Sign On call is made to receive credentials to process
2. The Merchant Application will identify if Method Data is required (options are detailed below)
3. Authenticate is called to start Authentication process
4. If Challenge is required, Merchant Application and Access Control Server complete a challenge
5. Snap* sends Authentication results to Merchant App



Authentication

For Authentication, the Merchant Application follows a similar workflow as if the payment were going to be processed through the Snap* Platform, beginning with a Sign On.

Sign on With Token

Merchants will need to implement the SignOnWithToken API request to get a SessionToken for the Snap* platform. See the example requests and responsesbelow:

SOAP SignOnWithToken Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Body>
<SignOnWithToken xmlns="http://schemas.evosnap.com/CWS/v2.0/ServiceInformation">
<identityToken>PHNhbWw6QXNzZXJ0aW9uIE1ham9yVmVyc2lvbj0…</identityToken>
</SignOnWithToken>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP SignOnWithToken Response

```
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:Body><SignOnWithTokenResponse
xmlns="http://schemas.evosnap.com/CWS/v2.0/ServiceInformation"><SignOnWithTokenResult>PHNhbW
w6QXNz…<
/SignOnWithTokenResult>
</SignOnWithTokenResponse>
</s:Body>
</s:Envelope>
```

REST SignOnWithToken Request

RequestUri	https://api.cipcert.goevo.com/2.1.35/REST/SIS.svc/token
Method	GET

Set Username as the Identity Token on the HTTP Authentication header. The Request body is empty.

REST SignOnWithToken Response

A long SessionToken is returned on the response. This will be required on subsequent calls.

Check for 3-D Secure 2.0 Support

The 3-D Secure protocol requires the Card Range of a transaction be checked for 3-DS 2.0 support prior to Authentication. Card Range Data contains which versions of 3-D Secure the card(s) support, as well as

indicates if Method Data is required for the transaction. Method Data is additional information about a Cardholder's environment that is obtained by the Access Control Server via the Merchant Application. Snap* offers two options to have the Card Range Data results returned to the Merchant.

First, Merchants can manage their own cache of Card Range Data and receive all the updates since the last query using the QueryCardRanges operation. If the Merchant is using their own cache, they must query their own cache for 3-DS support before sending the initial Authentication request. This approach is optimal as it reduces the individual transaction time due to not having to query Snap* for this information on each transaction.

To initiate a Merchant cache, Merchants should query without a unique serial number. This will return all known Card Ranges as well as a unique serial number. Future queries should use the previously returned serial number to receive only Card Range updates since the last query.

QueryCardRanges without Serial Number

RequestUri	https://api.cipcert.goevo.com/2.1.35/REST/ ThreeDSecure.svc/3ds/querycard/ranges/{CardBrand}
Method	POST
Authentication	Session token set as Username on Authentication header

Response

```
{ "CardRanges":[{
                  "RangeAction": "A",
                  "RangeStart": "4000000000000000",
                  "RangeEnd": "4100000000000000",
                  "AcsStartProtocolVersion": "2.1.0",
                  "AcsEndProtocolVersion": "2.2.0",
                  "ThreeDsMethodUrl": "https://some.ds.url/" }],
                  "SerialNumber": "1"
}
```

QueryCardRanges with Serial Number

RequestUri	https://api.cipcert.goevo.com/2.1.35/REST/ ThreeDSecure.svc/3ds/querycard/ranges/{CardBrand}?serialNumber={serialNumber}
Method	POST
Authentication	Session token set as Username on Authentication header

Response

```
{ "CardRanges":[{
                  "RangeAction": "A",
                  "RangeStart": "4000000000000000",
                  "RangeEnd": "4100000000000000",
                  "AcsStartProtocolVersion": "2.0",
                  "AcsEndProtocolVersion": "2.1",
                  "ThreeDsMethodUrl": "https://some.ds.url/" }],
                  "SerialNumber": "2"
```

```
}
```

The response will contain the RangeAction for the associated Card Range. RangeAction has three options: A “Add”, D “Delete”, or M “Modify”. “Modify” excludes any modification of the RangeStart or RangeEnd.

On initial release, the CardType values supported will be ‘MasterCard’ and ‘Visa’. It is recommended to call QueryCardRanges daily for the most up to date Card Range Data.

Optionally, Merchants can query the Snap* Card Range Cache for the specific card being used in an individual transaction by the card number using the QuerySingleCard operation. The response will contain which versions of 3-D Secure 2.0 the card supports, as well as if Method Data is required for the transaction. This call will need to be made before any 3-D Secure attempt with an Authenticate transaction and will increase overall transaction time.

QuerySingleCard

RequestUri	https://api.cipcert.goevo.com/2.1.35/REST/ThreeDSecure.svc/3ds/querycard/single/{cardType}/{cardNumber}
Method	POST
Authentication	Session token set as Username on Authentication header

Request

```
{
   "AcquirerBIN": "654321",
   "MerchantCategoryCode": "1234",
   "Country": "USA",
   "MerchantBankId": "bank id",
   "MerchantId": "888888",
   "RequestorName": "MerchantName",
   "MerchantNumber": "TEST_NUMBER",
   "MerchantUrl": "http://www.evosnap.com",
   "MethodNotificationUrl": "http://methodnotification.url",
   "Name": "who are you",
   "NotificationUrl": "http://somenotification.url",
}
```

Response with ThreeDsMethodUrl

```
{
     "CardRange": {
               "AcsStartProtocolVersion": "2.0",
               "AcsEndProtocolVersion": "2.1",
               "ThreeDsMethodUrl": "https://some.ds.url/"
                  },
     "MethodData": {
               "ThreeDsMethodData": "somestring",
               "ServerTransactionId": "00000000-0000-0000-0000-000000000000"
                  }
}
```

Response without ThreeDsMethodUrl

```
{
```

```
     "CardRange": {
                "AcsStartProtocolVersion": "2.1.0",
                "AcsEndProtocolVersion": "2.2.0",
                "AcsInfoInd": "01,02,03,04",
                "ThreeDsMethodUrl": ""
                   },
     "MethodData": {

                 "ThreeDsMethodData": null,
                 "ServerTransactionId": null
                     }
}
```

If ThreeDsMethodUrl is populated in the response, Method Data is required. To exchange Method Data, post the ThreeDsMethodData to the ThreeDsMethodUrl.

Request

```
  <form name="frm" method="POST" target="iframe" action="https://some.ds.url/">
    <input type="hidden" name="threeDSMethodData"
value="eyJ0aHJlZU…pb25VUkwiOiJodHRwczovL2Zha2VzaXRlLmJsYWgifQ==">
  </form>
```

Response

```
  <form name="frm" method="POST" action="http://themerchantnotificationURL.url">
    <input type="hidden" name="threeDSMethodData"
value="eyJ0aHJlZU…g5YWE3LWFhNDItMjY2My03OTFiLTJhYzA1YTU0MmM0YSJ9">
  </form>
```

For more information on retrieving Method Data, please refer to card brand documents on ACS functionality.

At the time of a purchase, Snap* checks whether the card is supported for 3-D Secure 2.0 and if Method Data is required for the transaction. Merchants may query for Card Range Data outside of normal transaction processing to keep their Card Range Cache up to date or they may query Snap* on each transaction to check for 3-D Secure 2.0 support.

Checking if Method Data is required (MethodCompletionIndicator)

The Card Range support query will also indicate if Method Data is required for that card. Method Data is additional information about a Cardholder's environment that is obtained by ACS via the Merchant environment. The MethodCompletionIndicator field is set on the Merchant's Authenticate request to Platform. This field indicates if ACS has collected the Method Data (if applicable) from the Merchant. The possible values and their meanings are detailed in the sections below:

When MethodCompletionIndicator = 'Completed'

A MethodCompletionIndicator value of 'Completed' indicates that the ACS successfully collected the applicable method data. If the MethodCompletionIndicator value is set to 'Completed' on the Merchant's

Authenticate request, a ServerTransactionID is required for authentication. If a merchant does not set it, Snap* will assign one for them.

When MethodCompletionIndicator = 'NotCompleted'

This response indicates that the response from the ACS to the merchant was not received within 10 seconds.

When MethodCompletionIndicator = 'Unavailable'

A MethodCompletionIndicator value of 'Unavailable' indicates that there is no method data for the ACS to collect. If the MethodCompletionIndicator value is set to 'Unavailable' on the Merchant's Authenticate request, Platform will do a check against its own Card Range Cache to determine that method data is in fact not supported. If there is no ThreeDSMethodURL for the card, no fault or corresponding error message will be thrown. However, if a ThreeDSMethodURL is defined for the card, 'Unavailable' is not the correct MethodCompletionIndicator value, and, in this case, an error is thrown stating "Method data is supported for this card. Update the MethodCompletionIndicator and try again.", thus ending the transaction workflow.

When MethodCompletionIndicator = 'NotSet'

A MethodCompletionIndicator value of 'NotSet' may indicate that the card supports 3-D Secure 1.0 rather than 2.2. If the MethodCompletionIndicator value is set to 'NotSet' on the Merchant's Authenticate request, Platform will do a check against its own Card Range Cache to determine concretely if 3-D Secure 2.2 is supported.

If the card exists within Platform's stored 3-D Secure 2.2 CardRanges, this indicates that the card is in fact 3-D Secure 2.2 enrolled, and 'NotSet' is not the correct MethodCompletionIndicator value. Correct values for 3-D Secure 2.2 enrolled cards include 'Completed' 'NotCompleted' and 'Unavailable'. In this case, an error is thrown, saying "3-D Secure 2.2 is supported for this card. Update the MethodCompletionIndicator and try again."

Protocol Support

Due to the many roles in the Authentication workflow, Snap* has added logic to submit the highest mutually supported protocol in an Authentication request. This will guarantee the highest chance of a successful Authentication.

The Merchant Application will only need to have knowledge of the highest version they support and submit that value in the ProtocolVersion field on the request. The current release supports the following protocols: 1.0, 2.1 and 2.2. The Snap* Platform will execute protocol management based on this field, as well as Issuer and Card Range support. For informational purposes, the following flow defines protocol support.

First, if the Merchant Application has not been updated to support any 2.0 workflows, Snap* will continue to support the 3-D Secure 1.0 Authentication for those Merchants. Support for 1.0 only will be identified in the request by setting ProtocolVersion to 'v1_0', Is3DSecure to 'true', and SupportsProtocolVersion1 to 'true'. Previous endpoint integration to 3-D Secure 1.0 will not be affected by these additional fields.

When a Merchant Application upgrades to the 2.0 workflow through the Snap* Platform, they will submit ProtocolVersion as 'v2_X_0', X being defined as the highest minor version the Merchant Application would like to support.

Authentication without 2.0 Support

If the Issuer does not yet support 2.0, there are still a few options for Authentication.

1. If the Merchant Application still supports 1.0 (indicated by SupportsProtocolVersion1 set to 'true'), the Authentication will fall back to the existing 3-D Secure 1.0 functionality. This will be the default fallback if the Merchant Application submits a 3-D Secure 2.0 request but the Issuer does not support 2.0. A successful Authentication will be returned as TransactionStatus 'SuccessfullyAuthenticated'.

2. If the Merchant Application does not support 1.0 but the Merchant is registered for Data Insights with MasterCard (indicated by SupportsDataOnly set to 'true'), Snap* will send the transaction for 2.0 Data Only Authentication to the DS. The Data Insights program is available to Merchants who are not required to support SCA, but would like the DS to do a risk analysis on their transaction and submit that with the Authorization. This transaction type does not reach the ACS. A successful Data Only Authentication will be returned as TransactionStatus 'UnableToAuthenticate', and ProcessedAsDataOnly will be 'true'.

3. If the Merchant Application does not support 1.0 or Data Insights, but the DS supports the Attempts Server, Snap* will send the transaction for 2.0 Authentication to the DS Attempts Server. The Attempts Server is a product provided by the card brands to act as an Authenticator on behalf of the Issuer until the Issuer supports 2.0. The TransactionStatus in this workflow will be returned as 'AttemptsProcessingPerformed'.

4. If the Merchant Application does not support 1.0, the Merchant Application is not registered for MasterCard's Data Insights, and the DS does not support the Attempts Server, the transaction will return with ErrorCode '9000' and ErrorDescription "Unable to process 3-D Secure. Issuer does not support compatible protocol."

Fallback from 2.2 to 2.1

If the Merchant Application tries to process as 2.2 and only supports version 2.1, the transaction will be processed using the 2.1 protocols. Any fields that included on the Request but are not part of the 2.1 protocol will be dropped.

The following fields are not supported in version 2.1 and will be dropped from the Request:

- BrowserJavaScriptEnabled
- EMVPaymentTokenSource
- WhiteListStatus
- WhiteListSource
- DecoupledMaxTimeout
- DecoupledRequestIndicator

The following fields have accepted values that exist for 2.2 but not 2.1:

Field Name	Values
RequestorChallengeIndicator	05 = NoChallengeRequestedRiskAnalysis 06 = NoChallengeRequestedDataShare 07 = NoChallengeRequestedStrongAuth 08 = NoChallengeRequestedWhitelist 09 = ChallengeRequestedWhitelist
RequestorAuthMethod	07 = FIDOAssurance 08 = SRCAssurance

Assuming the transaction is successfully authenticated using the 3DS 2.1 protocol, the ProtocolVersion on the Response will be updated to indicate the transaction was processed as 2.1 (rather than 2.2 as indicated on the original Request).

Note that Protocol Support is supported the same way for both Browser and Application-Based workflows.

Initial Authentication Request

The Merchant will send in an initial Authenticate call to Snap*. This initial request will contain all the new required and conditional 3-D Secure 2.0 fields. There are four possible options on the Authenticate response:

1. If the response returns as SuccessfullyAuthenticated, the Authentication data is returned to the Merchant Application to be sent elsewhere for Authorization.

2. If the response returns as AttemptedProcessingPerfomed, the Merchant Application can attempt to process the transaction elsewhere if that provider supports the 3-D Secure attempted workflow.

3. If the response returns as NotAuthenticated, AuthenticationRejected or UnableToAuthenticate, the Merchant Application can attempt to process the transaction elsewhere as non-3-D Secure.

4. The final response option is Challenge Required, which is detailed in the section below.

Browser-Based Frictionless Authentication

Frictionless Authentication without Method Data (Authenticated Response Workflow)

Method Data is additional information about the Cardholder's browser obtained directly by ACS. In this workflow, the ACS does not support/require Method Data (i.e. no ThreeDsMethodUrl exists for the Card Range), and the additional 3-D Secure data is enough for the ACS to authenticate the transaction without further interaction with the Cardholder.

In this scenario:

- Merchant Application either calls QuerySingleCard or queries their local cache to determine if Method Data is supported for the Card Range. Query results indicate ThreeDsMethodUrl is not required for the Card Range by returning null.
- Merchant Application sends in Authenticate call with Is3DSecure set to 'true', MethodCompletionIndicator set to 'Unsupported', and other required 3-D Secure 2.0 fields.
- Authenticate response will contain both AuthenticationValue and AuthenticationECI

Frictionless Authentication with Method Data

In this workflow, the ACS supports Method Data (i.e. ThreeDsMethodUrl is defined on the Card Range). In this scenario:

- Merchant Application either calls QuerySingleCard or queries their local cache to determine if Method Data is supported for the CardRange.
    - If the Merchant Application does not implement a local cache, Snap* Platform will query the Platform cache and return single Card Range along with the base64-encoded ThreeDSMethodData to be posted to the ACS URL.
    - If Merchant Application has local cache, the ThreeDSMethodData can be created by generating a unique 36-character ServerTransactionId string and base64 encoding the ServerTransactionId and Merchant-defined NotificationURL. Please note this ServerTransactionId should be included on the Authenticate request.

The Merchant Application will then interact with the ACS to pull browser information and retrieve Method Data. To do this, Merchants should make a POST to the URL returned in the QuerySingleCard response if they are using the Snap* cache.

The Merchant Application then sends in an Authenticate call with Is3DSecure set to 'true', MethodCompletionIndicator set to 'Completed', and other required 3-D Secure 2.0 fields. If more information is needed about how Method Data is exchanged through this process, consult the appropriate card schemes or EMVCo specification here.

The required 3-D Secure 2.0 field listing will define the specific fields in the ThreeDSData and ThreeDSMerchantData Objects in the sample request below:

RequestUri	https://api.cipcert.goevo.com/2.1.35/REST/ThreeDSecure.svc/{serviceId}
Method	POST
Authentication	Session token set as Username on Authentication header

Request

```
{
"$type": "BankcardTransactionPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
 "TenderData": {
"$type": "BankcardTenderDataPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
    "CardData": {
            "CardType": 2,
```

```
            "CardholderName": "Johnny Cardholder",
            "PAN": "4539797605519795",
            "Expire": "1225",
            "ChipConditionCode": "9",
            "FallbackReason": 0,
            "StrongCardholderAuthSupport": 0
     },
    "CardSecurityData": {
            "CVDataProvided": 0
     },
    "CardholderIdType": 1,
    "TenderType": 0,
    "DeviceTypeIndicator": 0
     },
 "TransactionData": {
            "$type": "BankcardTransactionDataPro,
       http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
        "AccountType": 0,
        "CashBackAmount": 5.50,
        "CustomerPresent": 0,
        "EmployeeId": "1234",
        "EntryMode": 1,
        "GoodsType": 1,
        "InternetTransactionData": {
             "IpAddress": "127.0.0.1",
             "SessionId": "12345",
             "BrowserAcceptHeader": "1",
             "BrowserJavaEnabled": 2,
             "BrowserJavaScriptEnabled": 2,
             "BrowserLanguage": "en-US",
             "BrowserScreenColorDepth": "16",
             "BrowserScreenHeight": "400",
             "BrowserScreenWidth": "300",
             "BrowserTimeZone": "+000",
             "BrowserUserAgent": "2"
        },
        "InvoiceNumber": "12345",
        "OrderNumber": "333",
        "SignatureCaptured": false,
        "TipAmount": 1.24,
        "Amount": 100.00,
        "CurrencyCode": 4,
        "TransactionDateTime": "2014-10-06T20:49:14Z",
        "Reference": "referenceTest",
        "IsPartialShipment": false,
       "FeeAmount": 0.00,
       "PartialApprovalCapable": 0,
       "ScoreThreshold": "scoreThresholdtest",
       "IsQuasiCash": false,
       "TransactionCode": 0,
       "Is3DSecure": true,
       "CardholderAuthenticationEntity": 5,
       "CardPresence": false,
       "IsQuickPaymentService": false,
       "EBTType": 0,
       "AmountTypeIndicator": 0,
       "ThreeDSData": {
              "AuthenticationIndicator": 1,
              "ChallengeWindowSize": 0,
              "MethodCompletionIndicator": 1,
              "RequestorAuthMethod": 0,
              "RequestorChallengeIndicator": 0,
              "ServerTransactionId": "45ec66e7-536e-43dd-827c-24fa3f8cfed1",
```

```
            "TransactionType": 0,
            "WhiteListStatus": 1,
            "PaymentTokenIndicator": 0,
            "DecoupledMaxTimeout": 0,
            "DecoupledRequestIndicator": 0,
            "ProtocolVersion": "v2_2_0",
            "SupportsProtocolVersion1": false,
            "RequestorAuthData": null,
            "RequestorAuthTimestamp": "0001-01-01T00:00:00" },
            "ThreeDSMerchantData": {
                  "AcquirerBIN": "654321",
                  "Country": "USA",
                  "MerchantBankId": "bank id",
                  "MerchantCategoryCode": "1234",
                  "MerchantId": "888888",
                  "RequestorName": "MerchantName",
                  "MerchantNumber": "TEST_NUMBER",
                  "MerchantUrl": "http://www.evosnap.com",
                  "MethodNotificationUrl": "http://methodnotification.url",
                  "Name": "who are you",
                  "NotificationUrl": "http://somenotification.url",
            },
            "ThreeRIIndicator": 0,
            "TransactionStatusIndicator": 0
    },
    "ReportingData": {
            "Comment": "This is a comment",
            "Description": "12345678",
            "Reference": "12345678"
    },
    "IsOffline": false
}
```

Response

If the transaction was successfully Authenticated, a successful BankcardTransactionResponse will be returned to the Merchant Application, indicating Authentication is complete. It will contain the following fields needed for Authorization:

Parameter	Data Type	Description
ACSTransactionID	String	Identifier assigned by the Access Control Server to identify a single transaction.
AuthenticationECI	String	Payment System-specific value provided by the Access Control Server or Directory Server to indicate the results of the attempt to authenticate the Cardholder
AuthenticationType	Enum	Indicates the type of authentication method the Issuer will use to challenge the Cardholder: > *NotSet* > *Static* > *OOB* > *Decoupled* > *Other*
AuthenticationValue	String	Payment System-specific value provided by the Access Control Server or Directory Server using an algorithm defined by Payment System. It is used to provide proof

		of authentication.
ChallengeCancellation Indicator	Enum	Indicator informing the Access Control Server and the Directory Server that the authentication has been canceled. This will only be returned on a QueryAuthenticationResults Response: > *NotSet* > *CardholderCancel* > *RequestorCancel* > *TransactionAbandoned* > *TransactionTimeOut* > *TransactionTimeoutCReqNotReceived* > *TransactionError* > *Unknown*
DSTransactionID	String	Identifier assigned by the Directory Server to identify a single transaction.
MessageCategory	Enum	Identifies the category of the message for a specific use case: > *NotSet* > *NonPayment* > *Payment*
ProtocolVersion	Enum	The Protocol Version Number indicating which protocol was used for Authentication: > *NotSet* > *v1_0* > *v2_1_0* > *v2_2_0*
ServerTransactionId	String	Universally unique transaction identifier assigned by Snap* or the Merchant Application to identify a single transaction. Snap* will define this value if merchant is using their own Card Range Cache.
TransactionStatus	Enum	This value defines the authentication status for validation purposes. It is required for processing: > *SuccessfullyAuthenticated* > *NotAuthenticated* > *UnableToAuthenticate* > *AttemptsProcessingPerformed* > *ChallengeRequired* > *DecoupledAuthenticationRequired* > *AuthenticationRejected InformationalOnly*
WhiteListStatus	Enum	Enables the communication of trusted beneficiary/whitelist status between the Access Control Server, the Directory Server and the 3-D Secure Requestor: > *NotSet*

		> *IsWhiteListed* > *IsNotWhiteListed NotEligible* > *PendingConfirmation* > *CardholderRejected* > *StatusUnknown*

If the transaction was not successfully authenticated, the response will include ThreeDSInformation indicating why the Authentication failed:

Parameter	Data Type	Description
TransactionStatusReason	Enum	Provides information on why the transaction status field has the specified value. > *NotSet* > *CardAuthenticationFailed* > *UnknownDevice* > *UnsupportedDevice* > *ExceedsAuthenticationFrequencyLimit* > *ExpiredCard* > *InvalidCardNumber* > *InvalidTransaction* > *NoCardRecord* > *SecurityFailure* > *StolenCard* > *SuspectedFraud* > *TransactionNotPermitted* > *CardholderNotEnrolled* > *TransactionTimeout* > *LowConfidence* > *MediumConfidence* > *HighConfidence* > *VeryHighConfidence* > *ExceedsMaximumChallenges* > *NonPaymentTransactionNotSupported* > *ThreeRITransactionNotSupported* > *ACSTechnicalIssue* > *DecoupledRequiredButNotRequested* > *DecoupledMaxExpiryExceeded* > *DecoupledTimeout* > *CardholderRefusedAuthentication* > *Other*

Application-Based Frictionless Authentication

There is no Method Data within the Application-Based workflow; therefore, the Merchant does not need to consider the CardRangeCache. In this scenario, the Merchant Application interfaces with 3DS SDK to retrieve and encrypt device information and sends in an Authenticate request with SDKInfo fields and other required 3-D Secure 2.0 fields set – the Application specific fields can be found on the Snap* Documentation Portal.

The Authenticate response will contain the Authentication approval details, including AuthenticationValue and AuthenticationECI fields as proof of authentication and SDKResponseInfo. If a Challenge is required in the Authenticate response, the same Challenge workflow is followed as the Browser-Based Authentication and is detailed below.

Request

```
{
"$type": "BankcardTransactionPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
 "TenderData": {
"$type": "BankcardTenderDataPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
    "CardData": {
              "CardType": 2,
              "CardholderName": "Johnny Cardholder",
              "PAN": "4539797605519795",
              "Expire": "1225",
              "ChipConditionCode": "9",
              "FallbackReason": 0,
              "StrongCardholderAuthSupport": 0
     },
    "CardSecurityData": {
              "CVDataProvided": 0
     },
    "CardholderIdType": 1,
    "TenderType": 0,
    "DeviceTypeIndicator": 0
     },
 "TransactionData": {
              "$type": "BankcardTransactionDataPro,
       http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
        "AccountType": 0,
        "CashBackAmount": 5.50,
        "CustomerPresent": 0,
        "EmployeeId": "1234",
        "EntryMode": 1,
        "GoodsType": 1,
        "InternetTransactionData": null,
        "InvoiceNumber": "12345",
        "OrderNumber": "333",
        "SignatureCaptured": false,
        "TipAmount": 1.24,
        "Amount": 100.00,
        "CurrencyCode": 4,
        "TransactionDateTime": "2014-10-06T20:49:14Z",
        "Reference": "referenceTest",
       "IsPartialShipment": false,
       "FeeAmount": 0.00,
       "PartialApprovalCapable": 0,
       "ScoreThreshold": "scoreThresholdtest",
       "IsQuasiCash": false,
       "TransactionCode": 0,
```

```
        "Is3DSecure": true,
        "CardholderAuthenticationEntity": 5,
        "CardPresence": false,
        "IsQuickPaymentService": false,
        "EBTType": 0,
        "AmountTypeIndicator": 0,
"ThreeDSData": {
        "AuthenticationIndicator": 1,
        "ChallengeWindowSize": 0,
        "MethodCompletionIndicator": 2,
        "RequestorAuthMethod": 0,
        "RequestorChallengeIndicator": 0,
        "SDKInfo": {
                "AppId": "a048702f-6bcc-402a-8c22-1c2a362b02c5",
                "DeviceRenderOptions": {
                        "Interface": "Native",
                        "UITypes": ["SingleSelect"]
                        },
                "EncryptedData": "SomeEncryptedData",
                "MaxTimeout": 5,
                "PublicKey":
"eyJrdHkiOiJFQyIsImNydiI6IlAtMjU2IiwieCI6IlRFV0tSenk3S0t3cXZfWVZHbjV5bnBZc28xcVgxRjJnREVWbFB
kSEJzUzgiLCJ5IjoiSGVQQWxYM2laYWNTRTN6aGQ0ZU5WUnVUZ19hSDZNdG9nM0pTU21aV0tBUSJ9",
                "ReferenceNumber": "123",
                "TransactionId": "1d33eb63-88d4-40fa-8e8c-a9b0265cde95"
        },
        "ServerTransactionId": "84ae9a5a-f4e6-4fbd-8df1-20d208df927a",
        "TransactionType": 0,
        "PaymentTokenIndicator": 0,
        "AccountInfo": null,
        "AccountId": null,
        "MerchantRiskInfo": null,
        "DecoupledMaxTimeout": 0,
        "DecoupledRequestIndicator": 0,
        "ProtocolVersion": 3,
        "SupportsProtocolVersion1": true,
        "RequestorAuthData": null,
        "RequestorAuthTimestamp": "0001-01-01T00:00:00",
        "ThreeRIIndicator": 0,
        "IsInterRegionalTransaction": false,
        "IsAnonymousPrepaidTransaction": false,
        "ExemptionInfo": null,
     }
     "ThreeDSMerchantData": {
                "AcquirerBIN": "654321",
                "Country": "USA",
                "MerchantBankId": "bank id",
                "MerchantCategoryCode": "1234",
                "MerchantId": "888888",
                "RequestorName": "MerchantName",
                "MerchantNumber": "TEST_NUMBER",
                "MerchantUrl": "http://www.evosnap.com",
                "MethodNotificationUrl": "http://methodnotification.url",
                "Name": "who are you",
                "NotificationUrl": "http://somenotification.url",
        },
       "TransactionStatusIndicator": 0
    },
    "ReportingData": {
                "Comment": "This is a comment",
                "Description": "12345678",
                "Reference": "12345678"
    },
```

```
    "IsOffline": false
}
```


Response


```
{
    "AdviceResponse": "NotSet",
    "Amount": 1.00,
    "Status": "Successful",
    "CommercialCardResponse": "NotSet",
    "CardType": "Visa",
    "StatusCode": "1",
    "ReturnedACI": "NotSet",
    "FeeAmount": 0.00,
    "StatusMessage": "APPROVED",
    "ApprovalCode": "R7SJKB",
    "TransactionId": "386700F258A04B78AE7415DA5F07C0AE",
    "AVSResult": null,
    "OriginatorTransactionId": "3014",
    "BatchId": "2014",
    "ServiceTransactionId": "36467853",
    "CVResult": "NotSet",
    "ServiceTransactionDateTime": {
        "Date": "2020-09-14",
        "Time": "21:55:35.448",
        "TimeZone": "-06:00"
    },
    "CardLevel": ""
    "DowngradeCode": "",
    "CaptureState": "ReadyForCapture",
    "MaskedPAN": "402400XXXXXX8834",
    "TransactionState": "Authorized",
    "PaymentAccountDataToken": "",
    "IsAcknowledged": false,
    "RetrievalReferenceNumber": "088960526209",
    "Reference": "3014",
    "Resubmit": "NotSet",
    "TransmissionNumber": "386700F258A04B78AE7415DA5F07C0AE",
    "SettlementDate": "0001-01-01T00:00:00",
    "TransactionCode": "",
    "FinalBalance": null,
    "HostMessageId": "",
    "OrderId": "2914",
    "Geolocation": null,
    "CashBackAmount": 0.00,
    "TerminalAccessToken": null,
    "PrepaidCard": "NotSet",
    "Expire": "0230",
    "ErrorType": null,
    "AuthorizationServerUrl": "",
    "PaymentAuthorizationRequest": "",
    "ProcessedAs3D": false,
    "EMVDataResponse": null,
    "Level3Added": "NotSet",
    "LastPANDigits": "8834",
    "BatchAmount": 0.00,
    "MessageAuthenticationCode": "",
    "TokenInformation": null,
    "ForcePostCode": "",
    "MerchantId": "123456789012",
    "TerminalId": "001",
    "BankResponseCode": "",
```

```
    "InitialEncryptionKeys": null,
    "IsPartialApproval": false,
    "IndustryType": "Ecommerce",
    "ThreeDSecureInformation": null,
    "ThreeDSInformation": {
        "TransactionStatus": "SuccessfullyAuthenticated",
        "AuthenticationECI": "05",
        "DSTransactionId": "36949325-12d1-4c14-a222-3f4fb422d454",
        "IsChallengeMandated": false,
        "ChallengeRequest": null,
        "ChallengeCancellationIndicator": null,
        "TransactionStatusReason": "NotSet",
        "AuthenticationValue": "QmFzZTY0RW5jb2RlZDIwYnl0ZXM=",
        "ACSTransactionId": "f948ba27-33aa-471a-aa5e-3501d939932e",
        "AuthenticationType": "Dynamic",
        "CardholderInformationText": null,
        "DSReferenceNumber": null,
        "ErrorCode": null,
        "ErrorDetail": null,
        "ErrorDescription": null,
        "AcsUrl": null,
        "MerchantId": null,
        "MessageCategory": "NonPayment",
        "ProtocolVersion": "v2_1_0",
        "ServerTransactionId": "3480eabc-9b1c-4740-91ea-b9f585a7b7b8",
        "WhiteListStatus": "NotSet",
        "TokenResult": "",
        "Protocol1": null,
        "SCARequired": false,
        "ReasonForNotHonoringExemption": "",
        "ExemptionControl": "NotSet",
        "SDKResponseInfo": {
            "ACSOperatorId": "AcsOpId 4138359541",
            "ACSReferenceNumber": "3DS_LOA_ACS_PPFU_020100_00009",
            "ACSRenderingType": {
                "Interface": "NotSet",
                "UITemplate": "NotSet"
            },
            "ACSSignedContent": null,
            "AppId": "8b4ad245-4a0c-4568-b5be-c1503208a40b",
            "MaxTimeout": 5,
            "TransactionId": "711540ec-967a-487a-aca7-4192f3556514"
        },
        "AuthenticationTimestamp": "2020-09-14T21:55:00+00:00",
        "AuthenticationMethod": "Frictionless"
    },
    "SystemTraceAuditNumber": "C347ZAVKXKB9312",
    "MACTransmissionNumber": ""
}
```


Challenge Authentication Response

Alternatively, the response can indicate a Challenge is required. This workflow can occur as an extension to Frictionless Authentication, with or without Method Data. When the Challenge workflow is invoked, the initial Authenticate call returns a TransactionStatus of 'ChallengeRequired' on the response in ThreeDSInformation.

The Issuer or the Merchant could request a Challenge for reasons such as the transaction amount is above defined limit or the browser information is not recognized. Examples of Challenges are SMS or email verification.

Response to Original Authenticate Call


```
"ThreeDSInformation": {
         "TransactionStatus": "ChallengeRequired",
         "AuthenticationECI": null,
         "DSTransactionId": null,
         "IsChallengeMandated": true,
         "ChallengeRequest":
         "eyJtZXNzYW...SI6IjAxIn0",
         "ChallengeCancellationIndicator": null,
         "TransactionStatusReason": "NotSet",
         "AuthenticationValue": null,
         "ACSPublicKey": null,
         "ACSOperatorId": null,
         "ACSReferenceNumber": null,
         "ACSRenderingInterface": "NotSet",
         "ACSRenderingUITemplate": "NotSet",
         "ACSSignedContent": null,
         "ACSTransactionId": null,
         "AuthenticationType": "NotSet",
         "CardholderInformationText": null,
         "DSReferenceNumber": null,
         "ErrorCode": null,
         "ErrorDetail": null,
         "ErrorDescription": null,
         "AcsUrl": "https://mockacsds.uat.evopayments.com/CReq",
         "MerchantId": null,
         "MessageCategory": "NotSet",
         "ProtocolVersion": "NotSet",
         "ServerTransactionId": null,
         "WhiteListStatus": "NotSet"
}
```


Decoupled Authentication

Decoupled Authentication is an authentication method whereby Authentication can occur independent from the Cardholder's experience with the Merchant Application (i.e. outside the browser or 3-D Secure SDK) and follows a similar workflow to Challenge Authentication.

In this workflow, the Merchant Application will call Authenticate with the required 3-D Secure fields and two Decoupled Authentication fields set: DecoupledRequestIndicator and DecoupledMaxTimeout. If DecoupledRequestIndicator is set, DecoupledMaxTimeout is required to be set.

On the response, TransactionStatus will return as DecoupledAuthenticationRequired and there will be no AcsUrl or ChallengeRequest to post.

The Merchant will periodically call QueryAuthenticationResults with DecoupledInfo fields set to search for Decoupled Authentication results. The Merchant Application will poll for Decoupled Authentication results until the DecoupledMaxTimeout limit has been reached – this field indicates the maximum amount of time in minutes that the Merchant Application will wait for the results of a Decoupled Authentication transaction and is set by the Merchant on ThreeDSData. If the authentication results have not yet been received and the timeout has not been exceeded, the message will read "Decoupled authentication has not completed". If timeout has been exceeded, message will read "Decoupled Authentication timed out".

Note that Decoupled Authentication is only available for the 3-D Secure 2.2 protocol version and is available for Browser and Application-based workflows.

Request


```
{
"$type": "BankcardTransactionPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
  "TenderData": {
"$type": "BankcardTenderDataPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
       "CardData": {
              "CardType": 2,
              "CardholderName": "Johnny Cardholder",
              "PAN": "4539797605519795",
              "Expire": "1225",
              "ChipConditionCode": "9",
              "FallbackReason": 0,
              "StrongCardholderAuthSupport": 0
       },
       "CardSecurityData": {
              "CVDataProvided": 0
       },
       "CardholderIdType": 1,
       "TenderType": 0,
       "DeviceTypeIndicator": 0
       },
  "TransactionData": {
       "$type": "BankcardTransactionDataPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
       "AccountType": 0,
       "CashBackAmount": 5.50,
       "CustomerPresent": 0,
       "EmployeeId": "1234",
       "EntryMode": 1,
       "GoodsType": 1,
       "InternetTransactionData": {
              "IpAddress": "127.0.0.1",
              "SessionId": "12345",
              "BrowserAcceptHeader": "1",
              "BrowserJavaEnabled": 2,
              "BrowserJavaScriptEnabled": 2,
              "BrowserLanguage": "en-US",
              "BrowserScreenColorDepth": "16",
              "BrowserScreenHeight": "400",
              "BrowserScreenWidth": "300",
              "BrowserTimeZone": "+000",
              "BrowserUserAgent": "2"
        },
       "InvoiceNumber": "12345",
       "OrderNumber": "333",
       "SignatureCaptured": false,
       "TipAmount": 1.24,
       "Amount": 100.00,
       "CurrencyCode": 4,
       "TransactionDateTime": "2014-10-06T20:49:14Z",
       "Reference": "referenceTest",
       "IsPartialShipment": false,
       "FeeAmount": 0.00,
       "PartialApprovalCapable": 0,
       "ScoreThreshold": "scoreThresholdtest",
       "IsQuasiCash": false,
       "TransactionCode": 0,
       "Is3DSecure": true,
       "CardholderAuthenticationEntity": 5,
       "CardPresence": false,
```

```
        "IsQuickPaymentService": false,
        "EBTType": 0,
        "AmountTypeIndicator": 0,
        "ThreeDSData": {
                "AuthenticationIndicator": 1,
                "ChallengeWindowSize": 0,
                "MethodCompletionIndicator": 1,
                "RequestorAuthMethod": 0,
                "RequestorChallengeIndicator": 0,
                "ServerTransactionId": "45ec66e7-536e-43dd-827c-24fa3f8cfed1",
                "TransactionType": 0,
                "WhiteListStatus": 1,
                "PaymentTokenIndicator": 0,
                "DecoupledMaxTimeout": 5,
                "DecoupledRequestIndicator": "true",
                "ProtocolVersion": "v2_2_0",
                "SupportsProtocolVersion1": false,
                "RequestorAuthData": null,
                "RequestorAuthTimestamp": "0001-01-01T00:00:00" },
                "ThreeDSMerchantData": {
                        "AcquirerBIN": "654321",
                        "Country": "USA",
                        "MerchantBankId": "bank id",
                        "MerchantCategoryCode": "1234",
                        "MerchantId": "888888",
                        "RequestorName": "MerchantName",
                        "MerchantNumber": "TEST_NUMBER",
                        "MerchantUrl": "http://www.evosnap.com",
                        "MethodNotificationUrl": "http://methodnotification.url",
                        "Name": "who are you",
                        "NotificationUrl": "http://somenotification.url",
                },
        "ThreeRIIndicator": 0,
        "TransactionStatusIndicator": 0
    },
    "ReportingData": {
        "Comment": "This is a comment",
        "Description": "12345678",
        "Reference": "12345678"
    },
    "IsOffline": false
}
```


Data Only Authentication – *MasterCard Only*

Data Only Authentication offers higher transaction approval rates without the possibility of a Challenge. In this workflow, the Authentication Request is sent to the card scheme, and the card scheme performs risk analysis using the 3-D Secure data provided. The card scheme is responsible for injecting risk assessment data into the Authentication Request, and forwards the request to the issuing bank for approval. Data Only Authentication is only supported by MasterCard.

For Data Only Authentication, the Merchant Application will call Authenticate with BankcardTenderData/CardData/CardType set to Mastercard and BankcardTransactionData/ThreeDSData/ProcessAsDataOnly set to true, and the response will contain the ProcessedAsDataOnly field set as true.

Request

```
{
"$type": "BankcardTransactionPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
   "TenderData": {
"$type": "BankcardTenderDataPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
      "CardData": {
            "CardType": 2,
            "CardholderName": "Johnny Cardholder",
            "PAN": "4539797605519795",
            "Expire": "1225",
            "ChipConditionCode": "9",
            "FallbackReason": 0,
            "StrongCardholderAuthSupport": 0
      },
      "CardSecurityData": {
            "CVDataProvided": 0
      },
      "CardholderIdType": 1,
      "TenderType": 0,
      "DeviceTypeIndicator": 0
      },
   "TransactionData": {
      "$type": "BankcardTransactionDataPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
      "AccountType": 0,
      "CashBackAmount": 5.50,
      "CustomerPresent": 0,
      "EmployeeId": "1234",
      "EntryMode": 1,
      "GoodsType": 1,
      "InternetTransactionData": {
            "IpAddress": "127.0.0.1",
            "SessionId": "12345",
            "BrowserAcceptHeader": "1",
            "BrowserJavaEnabled": 2,
            "BrowserJavaScriptEnabled": 2,
            "BrowserLanguage": "en-US",
            "BrowserScreenColorDepth": "16",
            "BrowserScreenHeight": "400",
            "BrowserScreenWidth": "300",
            "BrowserTimeZone": "+000",
            "BrowserUserAgent": "2"
      },
      "InvoiceNumber": "12345",
      "OrderNumber": "333",
      "SignatureCaptured": false,
      "TipAmount": 1.24,
      "Amount": 100.00,
      "CurrencyCode": 4,
      "TransactionDateTime": "2014-10-06T20:49:14Z",
      "Reference": "referenceTest",
      "IsPartialShipment": false,
      "FeeAmount": 0.00,
      "PartialApprovalCapable": 0,
      "ScoreThreshold": "scoreThresholdtest",
      "IsQuasiCash": false,
      "TransactionCode": 0,
      "Is3DSecure": true,
      "CardholderAuthenticationEntity": 5,
      "CardPresence": false,
      "IsQuickPaymentService": false,
      "EBTType": 0,
```

```
        "AmountTypeIndicator": 0,
        "ThreeDSData": {
                "AuthenticationIndicator": 1,
                "ChallengeWindowSize": 0,
                "MethodCompletionIndicator": 1,
                "RequestorAuthMethod": 0,
                "RequestorChallengeIndicator": 0,
                "ServerTransactionId": "45ec66e7-536e-43dd-827c-24fa3f8cfed1",
                "TransactionType": 0,
                "WhiteListStatus": 1,
                "PaymentTokenIndicator": 0,
                "DecoupledMaxTimeout": 0,
                "DecoupledRequestIndicator": 0,
                "ProtocolVersion": "v2_2_0",
                "SupportsProtocolVersion1": false,
                "RequestorAuthData": null,
                "RequestorAuthTimestamp": "0001-01-01T00:00:00" },
                "ThreeDSMerchantData": {
                        "AcquirerBIN": "654321",
                        "Country": "USA",
                        "MerchantBankId": "bank id",
                        "MerchantCategoryCode": "1234",
                        "MerchantId": "888888",
                        "RequestorName": "MerchantName",
                        "MerchantNumber": "TEST_NUMBER",
                        "MerchantUrl": "http://www.evosnap.com",
                        "MethodNotificationUrl": "http://methodnotification.url",
                        "Name": "who are you",
                        "NotificationUrl": "http://somenotification.url",
                },
                "ThreeRIIndicator": 0,
                "TransactionStatusIndicator": 0,
                "ProcessAsDataOnly": true
        },
        "ReportingData": {
                "Comment": "This is a comment",
                "Description": "12345678",
                "Reference": "12345678"
        },
        "IsOffline": false
}
```


Fallback to Non 3-D Secure Workflow

In this scenario, the Merchant Application calls Authenticate with BankcardTransactionData/ThreeDSData/ProcessAsDataOnly set to false, BankcardTransactionData/ThreeDSMerchantData/SupportsDataOnly set to false and BankcardTransactionData/ThreeDSData/SupportsProtocolVersion1 set to false. The Snap* Platform will send a response back to the Merchant Application with ErrorCode 9000 and ErrorDescription stating "Unable to process as 3DSecure. Issuer does not support compatible protocol."

Attempts Server (as a Fallback)

If an Issuer does not support 2.0 authentication AND the Merchant does not support 1.0 authentication, the

Card Brand can instead support Attempts. Attempts is when the DS makes a decision about the transaction without sending it to the ACS, as would happen in 3-D Secure 2.0 authentication. Attempts is processed the same as 3-D Secure 2.0 authentication, and the authentication requests are the same.

The DS supports Attempts when:

- The PAN is enrolled in 3DS 2.0 (appears in the table CardRange with a 2.x protocol) AND
- The CardRange record's optional field AcsInfoInd contains either "02" or is blank OR
- The PAN is not enrolled in 3DS 2.0 (does not appear in the table CardRange with a 2.x protocol)

If an authentication is attempted and the card is not in the card range, an authentication attempt will still be made, likely resulting in TransactionStatus = AttemptsProcessingPerformed.

QueryAuthenticationResults with Challenge Response

After the Authenticate response indicates a Challenge is required, the Merchant Application must complete the Challenge workflow with the Cardholder.

To initiate the Challenge, the Merchant Application posts the value of the Challenge Request field to the AcsURL that was returned on the Decline response to the original Authenticate. The ACS interacts with the Cardholder directly through a visible iFrame created in the Cardholder's browser and sends the Challenge Response to the Merchant-defined NotificationURL when the Cardholder-ACS interaction is completed. If more information is needed about how the Challenge data is exchanged through this process, consult the appropriate card schemes.

After the Merchant Application receives this Challenge Response, the merchant ***must*** call QueryAuthenticationResults with the Challenge Response data returned from the ACS as a string.

RequestUri	https://api.cipcert.goevo.com/2.1.35/REST/ThreeDSecure.svc/results/{serviceId}
Method	POST
Authentication	Session token set as Username on Authentication header

Request

```
{
     "TransactionId": "8D5E9BC4B84B46A9A1693AC8BF1C4FF7",
     "ChallengeResponse": "challenge response",
 },
```

If the transaction was successfully 3-D Secure authenticated, the response will include the same Authentication fields from BankCardTransactionResponse Table above. If the transaction was not successfully 3-D Secure authenticated, the response will include the same ThreeDSInformation indicating why the Authentication failed.

Exemptions

Snap* offers the benefit of Exemptions for the 3-D Secure as a Service workflow. Exemptions from the Challenge exist for low risk transactions and enable a greater percentage of Frictionless flow transactions. If a transaction qualifies as an exemption, the cardholder is available and known, but a request for no challenge authentication is made. There are six types of exemptions that are defined below:

1. **Whitelisted Merchants**

Cardholders can add Merchants to their whitelist of Merchants either during a Challenge flow or via their online banking application. If the Merchant Application would like to request the Cardholder is prompted to whitelist the Merchant, the following field must be set: BankcardTransactionData/ThreeDSData/RequestorChallengeIndicator is ChallengeRequestedWhitelist.

The Issuer keeps a database of whitelisted Merchants for each Cardholder. If a Merchant is whitelisted, Authentication will not be required. Snap* will return a WhitelistStatus indicating if the Merchant is whitelisted. Snap* will identify Whitelist exempted transactions as any “AuthenticationQueryParameters": { “TransactionId": "8D5E9BC4B84B46A9A1693AC8BF1C4FF7", “ChallengeResponse": "challenge response", }, payment where:

- BankcardTransactionData/ThreeDSData/ExemptionInfo/IsWhitelisted is true and
- BankcardTransactionData/ThreeDSData/RequestorChallengeIndicator is NoChallengeRequestedWhitelist

2. Secure Corporate Payments (B2B) Transactions

For Secure Corporate (B2B) Transactions, Merchant Applications can indicate to the Issuer that the payment is being initiated using a secure process or protocol – for example a physical card used within a secure corporate procurement system or process. Snap will identify Secure Corporate Payment exempted transactions as any payment where:

- BankcardTransactionData/ThreeDSData/ExemptionInfo/IsSecureCorporate is true and
- BankcardTransactionData/ThreeDSData/RequestorChallengeIndicator is NoChallengeRequestedRiskAnalysis

3. Low Value

Any transaction under 30 Euros is exempt from 3-D Secure Authentication. After the fifth consecutive Low Value exempted transaction, Authentication will again be required. Additionally, if the cumulative transaction amount with Low Value exemption exceeds 100 Euros, Authentication will again be required. The Exemption should be used as the last resort.

- BankcardTransactionData/ThreeDSData/ExemptionInfo/IsLowValue is true.

4. Low Risk

The initial release will not include any ability for Snap* to assess risk on behalf of the Merchant. However, the Merchant Application may request the Low Risk exemption based on any risk assessment they have done outside of the Snap* platform. Snap* will identify a Low Risk exempted transactions as any payment where:

- BankcardTransactionData/ThreeDSData/ExemptionInfo/IsLowRisk is true.

5. Recurring/Installment Payments

MasterCard allows the Recurring Payment exemption to be set as a request for Exemption. Snap* will identify a Recurring or Installment exempted transaction as any payment where:

- BankcardTransactionData/ThreeDSData/ExemptionInfo/IsRecurring is true.

6. **Delegated SCA**

   Delegated SCA is where the transaction is authenticated by a third-party Authenticator who is certified to the individual card brands. Issuers and Acquirers are then able to delegate authentication to these third-party Authenticators. Delegated Authenticators authenticate the Cardholder with two-factor authentication. Authenticator categories include:

   - Device Authenticators (usually biometrics on mobile or PC device)
   - Wallet Authenticators (applications often take advantage of device authenticators)
   - Merchant Authenticators (Merchant Applications that meet SCA requirements as part of normal processing)

Since many existing applications have been using these Authenticators since their creation, the Delegated SCA Exemption is meant to eliminate the need for SCA to be performed twice (leading to poor customer experience). For new applications, Delegated Authentication offers Merchant Applications the ability to take full control of the Challenge flow leading to better customer experience.

If the Merchant Application takes advantage of Delegated Authentication, they can identify the Delegated SCA Exemption by setting: BankcardTransactionData/ExemptionInfo/IsDelegatedSCA is true BankcardTransactionData/ThreeDSData/RequestorChallengeIndicator is NoChallengeRequestedStrongAuth

To submit any of the Exemptions in the 3-D Secure as a Service workflow, ExemptionControl must be set to 'AuthenticationFlow'.

The supported card brands have varying support for each of these Exemptions on the available protocols and will be supported in the following way:

> MasterCard transactions will be available to submit exempted transactions for Authentication in a 2.1 or 2.2 protocol message. Note that MasterCard does not support Whitelisted 2.1 transactions.

> Visa transactions will not be available to submit for Authentication with an Exemption set due to the limitation of the Visa 2.1 implementation. Visa will allow Exemptions only in a 2.2 integration.

Submitting an Exempted Transaction Request

When submitting the initial request for a 3-D Secure 2.0 exempted transaction, the Merchant Application is still required to populate all required and desired conditional 3-D Secure 2.0 fields. This is in case the Issuer rejects the Exemption.

If the Issuer rejects the Exemption, Snap* will return a decline response to the Merchant Application indicating a challenge is required, and the normal Challenge Authentication flow is executed. The Merchant Application will call QueryAuthenticationResults with ChallengeResponse as part of the standard Challenge flow.

If the issuer accepts the Exemption, Snap* responds to the Merchant Application with Transaction Status InformationalOnly and the Issuer ReasonForNotHonoring. The Merchant Application then submits an Authorization outside of Snap*. If the issuer then declines the Exemption, the Merchant Application should call a new Authenticate without Exemption and RequestorChallengeIndicator set to

ChallengeRequestedMandate or ChallengeRequestedPreference. The transaction then goes through the standard Challenge Authentication workflow for 3-D Secure as a Service.

Below is an example of a MasterCard Exemption Request transaction for 2.2.


```
 {
 "$type": "BankcardTransactionPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
 "CustomerData": null,
 "TenderData": {
       "$type": "BankcardTenderDataPro,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
       "EMVData": null,
       "CardData": {
            "CardType": "MasterCard",
            "CardholderName": "Spintax the Green",
            "PAN": "5307808167635130",
            "Expire": "0523"
       }
 },
 "TransactionData": {
       "Amount": 1.00,
       "CurrencyCode": "USD",
       "TransactionDateTime": "2020-05-28T08:01:38",
       "EntryMode": "Keyed",
 "InternetTransactionData": {
       "IpAddress": "127.0.0.1",
       "SessionId": "12",
       "BrowserAcceptHeader": "1",
       "BrowserJavaEnabled": "True",
       "BrowserJavaScriptEnabled": "True",
       "BrowserLanguage": "en-US",
       "BrowserScreenColorDepth": "1",
       "BrowserScreenHeight": "02",
       "BrowserScreenWidth": "02",
       "BrowserTimeZone": "+000",
       "BrowserUserAgent": "02"
 },
 "Is3DSecure": "true",
 "ThreeDSData": {
       "AuthenticationIndicator": "Payment",
       "ChallengeWindowSize": "Size390X400",
       "MethodCompletionIndicator": "Completed",
       "RequestorAuthMethod": "None",
       "RequestorChallengeIndicator": "ChallengeRequestedMandate",
 "ServerTransactionId": "DE7B9338-1AE9-49AD-8390-FFCDEAABB5D9",
       "TransactionType": "CheckAcceptance",
       "PaymentTokenIndicator": "NotSet",
       "AccountId": "",
       "DecoupledMaxTimeout": "0",
       "DecoupledRequestIndicator": "NotSet",
       "ProtocolVersion": "v2_2_0",
       "SupportsProtocolVersion1": "true",
       "RequestorAuthTimestamp": "2020-05-06T21:12:25.047Z",
              "ExemptionInfo": {
                     "ExemptionControl": "AuthenticationFlow",
                     "IsSecureCorporate": "true"
              }
 },
 "ThreeDSMerchantData": {
       "AcquirerBIN": "654321",
       "Country": "USA",
```

```
        "MerchantBankId": "ID",
        "MerchantCategoryCode": "0000",
        "MerchantId": "1",
        "RequestorName": "Acme_Corp",
        "MerchantNumber": "2222",
        "MerchantUrl": "www.evosnap.com",
        "MethodNotificationUrl": "https://www.acs.com/script",
        "Name": "Acme_Corp",
        "NotificationUrl": "www.notification.url",
}
}
}
```

Note that Exemptions are supported the same way for both Browser and Application-Based workflows.

Authorization

After authenticating the 3-D Secure transaction through the Snap* platform, Merchants may submit their transaction to whomever is responsible for Authorization, including the fields returned in the ThreeDSInformation on the Response.

Card on File for Non-Payment Transactions

There are two possible options to manage a card on file without processing a payment:

1. A card can be added to the account.
2. A card on file can be updated on the account.

Challenge Authentication is required when the Cardholder is managing the Cards on an account. Note that all 3-D Secure 2.0 required and conditional fields still need to be sent for these non-payment transactions.

Adding Card on File without Processing Payment

Merchant Applications indicate a card is being added by setting BankcardTransactionData/CardOnFileInfo/CardOnFile to First. Per SCA mandate, all First Merchant initiated transactions will require authentication. Merchant Applications will call Verify with BankcardTransactionData/Is3DSecure as True and BankcardTransactionData/CardOnFileInfo/InitiatedBy as Cardholder. Optional fields that can be set are BankcardTransactionData/ThreeDSData/RequestorChallengeIndicator as NotSet or ChallengeRequestedMandate and BankcardTransactionData/ThreeDSData/AuthenticationIndicator as NotSet or AddCard. If sent as NotSet these fields will default to these values.

Finally, the Merchant Application will call QueryAuthenticationResults for the Challenge completion and will follow the workflow detailed above. ACSTransactionId and DSTransactionId are returned on the BankcardTransactionResponse to be submitted for Authorization outside of Snap*. These values ***must*** be stored to process subsequent Authorizations for this Card on File.

Updating Existing Card on File without Processing Payment

If the card on file is updated by the cardholder, then a new Challenge is required. Again, Merchant Applications will call Verify with BankcardTransactionData/Is3DSecure as True,

BankcardTransactionData/CardOnFileInfo/ CardOnFile as Repeat, BankcardTransactionData/CardOnFileInfo/InitiatedBy as Cardholder and BankcardTransactionData/ThreeDSData/AuthenticationIndicator as MaintainCard, and Amount as Zero.

Repeat Card on File transactions require a reference to the First Card on File transaction. Whether the Merchant Application is using Snap* tokenization or Third Party Tokenization, the Repeat Card on File transaction will require the appropriate reference field to be set. See the Tokenization section below for more details.

Optionally, BankcardTransactionData/ThreeDSData/ RequestorChallengeIndicator can be set as NotSet or ChallengeRequestedMandate.

If the update is initiated by the Merchant, Authentication is not required.

Tokenization

Merchants Using Snap* Tokenization

For Merchant Applications using Snap* tokenization, Repeat Card on File transactions must be tokenized transactions with TenderData/PaymentAccountDataToken set to the PaymentAccountDataToken returned on the First Card on File transaction response.

Merchants Using Third Party Tokenization

The Merchant Application receives the reference ID on their First Card on File transaction response as TransmissionNumber. The Merchant Application must then submit CardOnFileInfo/OriginalTransactionId as the TransmissionNumber from the First Card on File transaction. Field length for TransmissionNumber is expected to be 20 characters.

Note that Card on File transactions are supported the same way for both Browser and Application-Based workflows.

Card on File for Payment Transactions

There are several possible options to manage a card on file while processing a payment:

1. A card on file can be added as part of a single payment.
2. A card on file can be added as part of the first recurring payment.
3. Merchants can initiate a transaction for a recurring payment.
4. A cardholder can initiate a transaction for a recurring payment.
5. A cardholder can update the card on file while processing a payment.

Adding a Card on File as Part of a Single Payment

Merchant Applications indicate a card is being added by setting BankcardTransactionData/CardOnFileInfo/ CardOnFile to First. Per SCA mandate, all First Merchant initiated transactions will require authentication. Merchant Applications will call Authenticate with BankcardTransactionData/Is3DSecure as True, BankcardTransactionData/ThreeDSData/ProtocolVersion as 2.1 or 2.2,

BankcardTransactionData/CardOnFileInfo/InitiatedBy as Cardholder and with an Amount greater than zero. ThreeDSMerchantData fields must also be set on the Authenticate request.

Optional fields that can be set are BankcardTransactionData/ThreeDSData/ RequestorChallengeIndicator as NotSet or ChallengeRequestedMandate and BankcardTransactionData/ ThreeDSData/AuthenticationIndicator as NotSet or AddCard. If sent as NotSet these fields will default to these values.

Finally, the Merchant Application will call QueryAuthenticationResults for the Challenge completion and will follow the workflow detailed above. Authentication data, including ACSTransactionId and DSTransactionId, are returned on the BankcardTransactionResponse to be submitted for Authorization outside of Snap*. These values ***must*** be stored to process subsequent Authorizations for this Card on File.

Repeat Card on File Transactions

Updating Card on File as Part of the First Recurring Payment

This process follows a similar workflow as adding a card on file for a single payment, with one exception. Merchant Applications will call Authenticate with BankcardTransactionData/CardOnFileInfo/CardOnFile as Repeat, BankcardTransactionData/Is3DSecure as True, BankcardTransactionData/ThreeDSData/ProtocolVersion as 2.1 or 2.2, BankcardTransactionData/CardOnFileInfo/InitiatedBy as Cardholder and with an Amount greater than zero. Merchant Applications must also set BankcardTransactionPro/BankcardInterchangeData/BillPayment as Recurring to indicate this is a recurring transaction, and ThreeDSMerchantData fields must be set on the Authentication request.

Repeat Card on File transactions require a reference to the First Card on File transaction. Whether the Merchant Application is using Snap* tokenization or Third Party Tokenization, the Repeat Card on File transaction will require the appropriate reference field to be set. See the Tokenization section below for more details.

Optionally, BankcardTransactionData/ThreeDSData/RequestorChallengeIndicator can be set as NotSet or ChallengeRequestedMandate.

Finally, the Merchant Application will call QueryAuthenticationResults for the Challenge completion and will follow the workflow detailed above. Authentication data, including ACSTransactionId and DSTransactionId, are returned on the BankcardTransactionResponse to be submitted for Authorization outside of Snap*.

Merchant Initiated Transaction

In this workflow, the Merchant application processes a recurring transaction and wants liability to shift to the Issuer. This workflow is supported only for protocol version 2.2. Merchant Applications will call Authenticate with BankcardTransactionData/CardOnFileInfo/ CardOnFile as Repeat, BankcardTransactionData/Is3DSecure as True, BankcardTransactionData/ThreeDSData/ProtocolVersion as 2.2, BankcardTransactionData/CardOnFileInfo/InitiatedBy as Cardholder and with an Amount greater than zero. Merchant Applications must also set BankcardTransactionPro/BankcardInterchangeData/BillPayment as Recurring to indicate this is a recurring transaction and ThreeDSMerchantData fields must also be set on the Authentication request.

Merchant Applications must also set BankcardTransactionData/CardOnFileInfo/InitiatedBy as Merchant, BankcardTransactionData/ThreeDSData/PaymentTokenIndicator as True, BankcardTransactionData/ThreeDSData/ThreeRIIndicator as Recurring, BankcardTransactionPro/BankcardInterchangeData/RecurringExpirationDate as the recurring payment expiration date, and BankcardTransactionPro/BankcardInterchangeData/RecurringFrequency as the interval at which the recurring payment is processed.

As before, Repeat Card on File transactions require a reference to the First Card on File transaction. Whether the Merchant Application is using Snap* tokenization or Third Party Tokenization, the Repeat Card on File transaction will require the appropriate reference field to be set. See the Tokenization section below for more details.

If BankcardTransactionData/CardOnFileInfo/InitiatedBy is set to Merchant, a Challenge is not mandated and the RequestorChallengeIndicator will not default to ChallengeRequestedMandate as it will if InitiatedBy is Cardholder.

Finally, the Merchant Application will call QueryAuthenticationResults for the Challenge completion and will follow the workflow detailed above. Authentication data, including ACSTransactionId and DSTransactionId, are returned on the BankcardTransactionResponse to be submitted for Authorization outside of Snap*.

Cardholder Initiated Transactions

For Cardholder Initiated transactions, the Merchant Application is responsible for setting BankcardTransactionData/ThreeDSData/AccountInfo, which is optional but suggested. This additional information allows the ACS to make risk based decisions with no direct interaction with cardholder for tokenized transaction.

Cardholder Initiated Transactions using Snap* Token

Merchant Applications using a Snap* token will call Authenticate with BankcardTransactionData/CardOnFileInfo/CardOnFile as Repeat, BankcardTransactionData/Is3DSecure as True, BankcardTransactionData/CardOnFileInfo/InitiatedBy as Cardholder, TenderData/PaymentAccountDataToken as the Snap Token, and an Amount greater than zero. ThreeDSMerchantData fields must also be set on the Authenticate request.

As before, Repeat Card on File transactions require a reference to the First Card on File transaction. Whether the Merchant Application is using Snap* tokenization or Third Party Tokenization, the Repeat Card on File transaction will require the appropriate reference field to be set. See the Tokenization section below for more details.

Because the card on file is being updated by the cardholder, a new Challenge is required. The Merchant Application will call QueryAuthenticationResults for the Challenge completion and will follow the workflow detailed above. ACSTransactionId and DSTransactionId are returned on the BankcardTransactionResponse to be submitted for Authorization outside of Snap*.

Cardholder Initiated Transactions using Third Party Token

Merchant Applications using a third party token will call Authenticate with BankcardTransactionData/CardOnFileInfo/CardOnFile as Repeat, BankcardTransactionData/Is3DSecure as True, BankcardTransactionData/CardOnFileInfo/InitiatedBy as Cardholder,

BankcardTenderData/TokenInformation as the third party token information, and an Amount greater than zero.

As before, Repeat Card on File transactions require a reference to the First Card on File transaction. Whether the Merchant Application is using Snap* tokenization or Third Party Tokenization, the Repeat Card on File transaction will require the appropriate reference field to be set. See the Tokenization section below for more details.

Because the card on file is being updated by the cardholder, a new Challenge is required. The Merchant Application will call QueryAuthenticationResults for the Challenge completion and will follow the workflow detailed above. Authentication data, including ACSTransactionId and DSTransactionId, are returned on the BankcardTransactionResponse to be submitted for Authorization outside of Snap*.

Cardholder Updates Card on File as Part of Processing Payment

In this workflow, the Cardholder updates the card on file while processing a payment. Merchant Applications will call Authenticate with BankcardTransactionData/ThreeDSData/ProtocolVersion as 2.1 or 2.2, BankcardTransactionData/CardOnFileInfo/CardOnFile as Repeat, BankcardTransactionData/Is3DSecure as True, BankcardTransactionData/CardOnFileInfo/InitiatedBy as Cardholder, and an Amount greater than zero. ThreeDSMerchantData fields must also be set on the Authentication request.

Optional fields that can be set are BankcardTransactionData/ThreeDSData/RequestorChallengeIndicator as NotSet or ChallengeRequestedMandate and BankcardTransactionData/ThreeDSData/ AuthenticationIndicator as NotSet or MaintainCard.

As before, Repeat Card on File transactions require a reference to the First Card on File transaction. Whether the Merchant Application is using Snap* tokenization or Third Party Tokenization, the Repeat Card on File transaction will require the appropriate reference field to be set. See the Tokenization section below for more details.

Because the card on file is being updated by the cardholder, a new Challenge is required. The Merchant Application will call QueryAuthenticationResults for the Challenge completion and will follow the workflow detailed above. Authentication data, including ACSTransactionId and DSTransactionId, are returned on the BankcardTransactionResponse to be submitted for Authorization outside of Snap*.

Tokenization

Merchants Using Snap* Tokenization

For Merchant Applications using Snap* tokenization, Repeat Card on File transactions must be tokenized transactions with TenderData/PaymentAccountDataToken set to the PaymentAccountDataToken returned on the First Card on File transaction response.

Merchants Using Third Party Tokenization

The Merchant Application receives the reference ID on their First Card on File transaction response as TransmissionNumber. The Merchant Application must then submit CardOnFileInfo/OriginalTransactionId as the TransmissionNumber from the First Card on File transaction. Field length for TransmissionNumber is expected to be 20 characters.

Best Practices

It is highly recommended that Merchant Applications not set AuthenticationIndicator to values other than MaintainCard and VerifyCardholder. This is to avoid setting it incorrectly and the transaction being rejected based on validation. Snap Platform will default AuthenticationIndicator for all use cases outside of MaintainCard and VerifyCardholder.

For merchants that have purchased tokenization (i.e. a Snap PaymentAccountDataToken is returned on the response), all transactions with card data will be defaulted to CardOnFile First. If the Merchant Application specifically sets AuthenticationIndicator to Payment, a validation error will occur as CardonFile First transactions must have AuthenticationIndicator set to AddCard. If the Merchant Application does not specifically set the AuthenticationIndicator, Snap Platform will default the correct value and no validation error will occur.

Note that Card on File transactions are supported the same way for both Browser and Application-Based workflows.