evopayments.com

3-D Secure 2.0

Full Feature Integration Guide

Simplifying Payments AROUND THE GLOBE
150+ CURRENCIES ACROSS 50 MARKETS WORLDWIDE

Table of Contents

VERSION HISTORY 2

OVERVIEW 3

WORKFLOW 3

- SIGN ON WITH TOKEN 4
- CHECK FOR 3-D SECURE 2.0 SUPPORT (BROWSER FLOW) 4
  - QueryCardRanges without Serial Number 5
  - QueryCardRanges with Serial Number 5
  - QuerySingleCard 6
- CHECKING IF METHOD DATA IS REQUIRED (METHODCOMPLETIONINDICATOR) 7
  - When MethodCompletionIndicator = 'Completed' 7
  - When MethodCompletionIndicator = 'NotCompleted' 7
  - When MethodCompletionIndicator = 'Unavailable' 7
  - When MethodCompletionIndicator = 'NotSet' 7
- PROTOCOL SUPPORT 8
- INITIAL AUTHORIZE OR AUTHORIZEANDCAPTURE REQUEST 9
  - Browser-Based Frictionless Authentication 9
  - Application-Based Frictionless Authentication 15
  - Challenge Authentication 19
- RESUBMIT WITH CHALLENGE RESPONSE 21
- AUTHORIZATION 27
- FOLLOW-ON TRANSACTIONS 28
  - Authorize and Capture or Authorize and Undo 29
- CARD ON FILE FOR NON-PAYMENT TRANSACTIONS 29
  - Adding Card on File without Processing Payment 29
  - Repeat Card on File Transactions 29

Version History

Version	Date	Description of Changes
V1	25 May 2020	> Initial Version. Browser Flow.
V2	16 June 2020	> .35R2 Updates for Protocol Support, Exemptions and Out of Scope > Updated ProtocolVersion Format
V3	17 June 2020	> Updated SIS URIs and ProtocolVersion > Updated URIs for QueryCardRanges and QuerySingleCard > Updated Failed Out of Scope and Exemption workflow
V4	30 June 2020	> Workflow update for Failed Out of Scope Transaction
V5	01 July 2020	> Added clarification on submitting an Exempted Transaction Request
V6	08 July 2020	> Clarification on Exemption support on the Protocols for each of the supported card brands > Workflow update for Follow-On Transactions
V7	13 July 2020	> Updated RangeAction options for QueryCardRanges with Serial Number
V8	16 July 2020	> Application-Based Frictionless Flow section added > Card on File for Non-Payment Transactions section added > Changed MerchantName field to RequestorName > Changed return status code for out of scope transactions to 65 for MC and Visa transactions

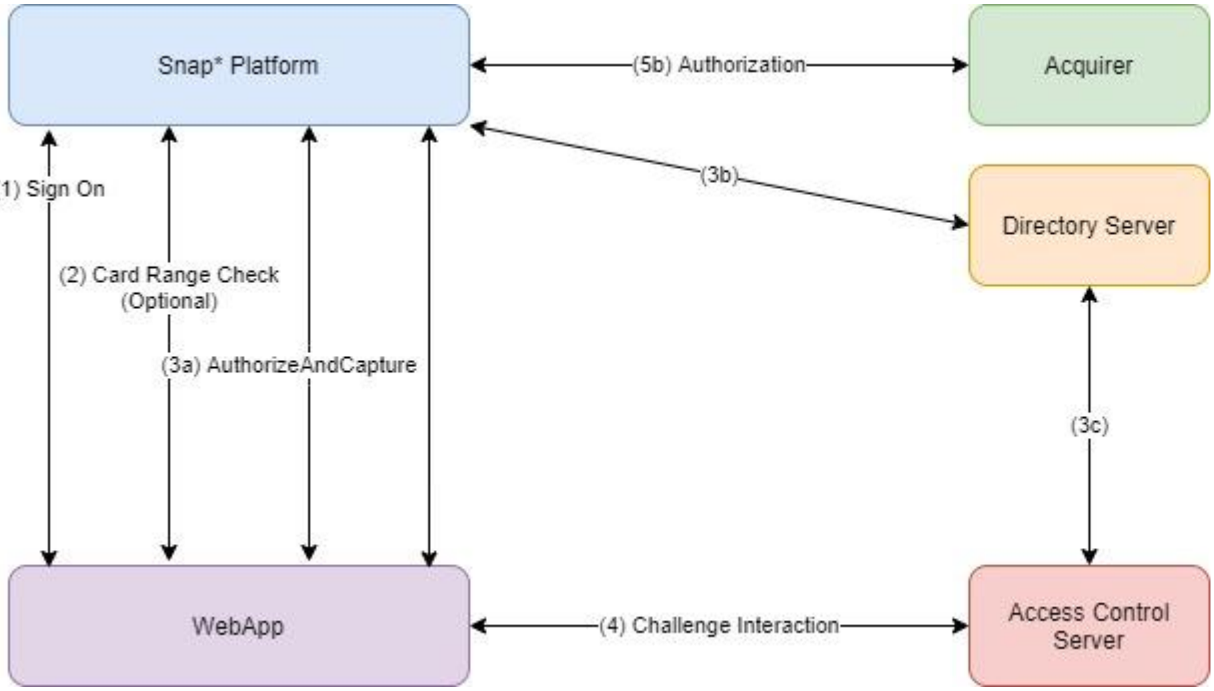
Overview

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.

Workflow

To begin a 3-D Secure 2.0 eCommerce transaction, the Merchant Application follows the existing workflow for processing payments through the Snap* Platform. Additional steps are present if either the Issuer or the Merchant requires a Challenge.

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



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 responses below:

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 (Browser Flow)

The 3-D Secure protocol requires the Card Range of a transaction be checked for 3-D Secure 2.0 support prior to processing a transaction. 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-D Secure 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/ /SIS.svc/3ds/querycard/ranges/{{merchantProfileId}}/{{serviceId}}?cardType={{cardtype}}
Method	GET
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/ /SIS.svc/3ds/querycard/ranges/{{merchantProfileId}}/{{serviceId}}?cardType={{cardtype}}
	&serialNumber={{3DSSerialNumber}}

Method	GET
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.

If ThreeDsMethodUrl is populated in the response, Method Data is required. For more information on retrieving Method Data, please refer to card brand documents on ACS functionality.

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

QuerySingleCard

RequestUri	https://api.cipcert.goevo.com/2.1.35/Rest/SIS.svc/3ds/querycard/single/{{merchantProfileId}}/{{serviceId}}?cardType={{cardType}}&cardNumber={{cardNumber}}
Method	GET

Response with ThreeDsMethodUrl

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

Response without ThreeDsMethodUrl

```
{
     "CardRange": {
               "AcsStartProtocolVersion": "1.0",
               "AcsEndProtocolVersion": "1.3",
               "ThreeDsMethodUrl": ""
                   },
     "MethodData": {
                "ThreeDsMethodData": "",
                "AcsUrl": "https://some.acs.url/",
                "ServerTransactionId": "00000000-0000-0000-0000-000000000000"
                    }
}
```

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 the ACS via the Merchant Application environment. The MethodCompletionIndicator field is set on the Merchant Application's Authorize or AuthorizeAndCapture request to Platform. This field indicates if the ACS has collected the Method Data (if applicable) from the Merchant Application. 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 Application's Authorize or AuthorizeAndCapture request, a ServerTransactionID is required for Authentication. If a Merchant Application 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 Application 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 Application's Authorize or AuthorizeAndCapture 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 2.0. If the MethodCompletionIndicator value is set to 'NotSet' on the Merchant Application's Authorize or AuthorizeAndCapture request, Platform will do a check against its own Card Range Cache to concretely determine if 3-D Secure 2.0 is supported.

If the card exists within Platform's stored 3D Secure 2.0 Card Ranges, this indicates that the card is in fact 3-D Secure 2.0-enrolled, and 'NotSet' is not the correct MethodCompletionIndicator value. Correct values for

3-D Secure 2.0-enrolled cards include ‘Completed’, ‘NotCompleted’ and ‘Unavailable’. In this case, an error is thrown, citing “3DS 2.0 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. 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."

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

Initial Authorize or AuthorizeAndCapture Request

The initial Authorize or AuthorizeAndCapture request is where the process of submitting a 3-D Secure transaction actually begins. This initial request will contain all the new required and conditional 3-D Secure 2.0 fields. There are four possible options on the response:

1. If the response returns as SuccessfullyAuthenticated, the transaction is automatically sent for Authorization.

2. If the response returns as AttemptedProcessingPerfomed, the transaction is automatically sent for Authorization.

3. If the response returns as NotAuthenticated, AuthenticationRejected or UnableToAuthenticate, a Decline response is returned to the Merchant Application, and they can choose to reattempt the transaction as non-3-D Secure.

4. The final response option is Challenge Required.

Browser-Based Frictionless Authentication

Frictionless Authentication without Method Data (Authorized 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 for that card (i.e. no ThreeDsMethodUrl on CardRange), 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 Authorize or AuthorizeAndCapture call with Is3DSecure set to 'true', MethodCompletionIndicator set to 'Unavailable', and other required 3-D Secure 2.0 fields.
- Authorize or AuthorizeAndCapture response will contain the Authorization approval details.

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

Request

```
  {
```

```
 "$type": "AuthorizeAndCaptureTransaction,
http://schemas.evosnap.com/CWS/v2.0/Transactions/Rest",
 "transaction":
 {
   "$type":
"BankcardTransactionPro,http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
   "TenderData": {
   "$type":
"BankcardTenderDataPro,http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
      "CardData": {
           "CardType":"2",
           "CardholderName": "Johnny Cardholder2",
           "PAN": "4024007108478834",
           "Expire":"1225",
           "ChipConditionCode":"9"
      },
      "CardholderIdType":"NoEAuth"
    },
 "TransactionData": {
  "$type":
"BankcardTransactionDataPro,http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard/Pro",
    "CashBackAmount":"5.5",
    "EntryMode":"Keyed",
    "GoodsType":"DigitalGoods",
    "InternetTransactionData": {
             "IpAddress": "127.0.0.1",
             "SessionId": "12345",
             "BrowserAcceptHeader":"1",
             "BrowserJavaEnabled":"True",
             "BrowserJavaScriptEnabled":"True",
             "BrowserLanguage": "en-US",
             "BrowserScreenColorDepth": "16",
             "BrowserScreenHeight": "400",
             "BrowserScreenWidth": "300",
             "BrowserTimeZone":"+000",
             "BrowserUserAgent":"2"
    },
    "InvoiceNumber": "12345",
    "OrderNumber": "333",
    "SignatureCaptured":false,
    "TipAmount":1.24,
    "Amount":1.00,
    "CurrencyCode":"USD",
    "TransactionDateTime":"2014-10-06T20:49:14Z",
    "Reference": "referenceTest",
    "Is3DSecure":true,
    "CardholderAuthenticationEntity":"None",
    "CardPresence":false,
    "IsQuickPaymentService":false,
    "ThreeDSData": {
      "AuthenticationIndicator":"Payment",
      "ChallengeWindowSize":"0",
      "MethodCompletionIndicator":"Unavailable",
      "PriorTransactionId":null,
      "RequestorAuthMethod":"0",
      "RequestorChallengeIndicator":"0",
      "ServerTransactionId":"e44b34d5-6edc-4f21-a661-434393e4c7e1",
      "TransactionType":"0",
      "WhiteListStatus": 1,
      "PaymentTokenIndicator":"0",
      "SecureCorporatePayment":"0",
      "DecoupledMaxTimeout":"0",
      "DecoupledRequestIndicator":"0",
      "ProtocolVersion":"v2_2_0",
      "SupportsProtocolVersion1":false,
      "RequestorAuthTimestamp": "0001-01-01T00:00:00"
    },
    "ThreeRIIndicator":"NotSet",
    "TransactionStatusIndicator":"NotSet"
   },
   "IsOffline":false
  },
  "ApplicationProfileId":"781738",
  "MerchantProfileId": "CWS TestClient .30 New Profile"
 }
```

Response

```
{
 "AdviceResponse": "NotSet",
 "Amount": 1.00,
 "Status": "Successful",
 "CommercialCardResponse": "NotSet",
 "CardType": "Visa",
 "StatusCode": "1",
 "ReturnedACI": "NotSet",
 "FeeAmount": 0.00,
 "StatusMessage": "APPROVED",
 "ApprovalCode": "MM1TJX",
 "TransactionId": "620AE845CB5B4A58936AE4B32BE0BBB1",
 "AVSResult": null,
 "OriginatorTransactionId": "23128",
 "BatchId": "2022",
 "ServiceTransactionId": "13096542",
 "CVResult": "NotSet",
 "ServiceTransactionDateTime": {
     "Date": "2020-05-22",
     "Time": "18:28:47.518",
     "TimeZone": "-06:00"
 },
 "CardLevel": "",
 "DowngradeCode": "",
 "CaptureState": "Captured",
 "MaskedPAN": "402400XXXXXX8834",
 "TransactionState": "Captured",
 "PaymentAccountDataToken": "620ae845-cb5b-4a58-936a-e4b32be0bbb1d65a3f19-3488-4a04-
a61daae52734c361",
 "IsAcknowledged": false,
 "RetrievalReferenceNumber": "854257425993",
 "Reference": "23128",
 "Resubmit": "NotSet",
 "TransmissionNumber": "620AE845CB5B4A58936AE4B32BE0BBB1",
 "SettlementDate": "0001-01-01T00:00:00",
 "TransactionCode": "",
 "FinalBalance": null,
 "HostMessageId": "",
 "OrderId": "22128",
 "Geolocation": null,
 "CashBackAmount": 0.00,
 "TerminalAccessToken": null,
 "PrepaidCard": "NotSet",
 "Expire": "1225",
 "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,
 "EBTAvailableBalance": {
    "CashAvailableBalance": 0.00,
    "SNAPAvailableBalance": 0.00
 },
 "IndustryType": "Ecommerce",
 "ThreeDSecureInformation": null,
 "ThreeDSInformation": {
    "TransactionStatus": "SuccessfullyAuthenticated",
    "AuthenticationECI": "05",
    "DSTransactionId": "20c03200-339f-4ae1-b7e8-22b6eac1fb0c",
    "IsChallengeMandated": false,
    "ChallengeRequest": null,
    "ChallengeCancellationIndicator": null,
    "TransactionStatusReason": "NotSet",
    "AuthenticationValue": "QmFzZTY0RW5jb2RlZDIwYnl0ZXM=",
    "ACSPublicKey": null,
    "ACSOperatorId": null,
    "ACSReferenceNumber": null,
    "ACSRenderingInterface": "NotSet",
    "ACSRenderingUITemplate": "NotSet",
    "ACSSignedContent": null,
    "ACSTransactionId": "902157de-f4da-46dd-a242-67e2a6852bc3",
    "AuthenticationType": "Dynamic",
    "CardholderInformationText": null,
    "DSReferenceNumber": null,
    "ErrorCode": null,
    "ErrorDetail": null,
    "ErrorDescription": null,
    "AcsUrl": null,
    "MerchantId": null,
    "MessageCategory": "Payment",
    "ProtocolVersion": "v2_1_0",
    "ServerTransactionId": “QmFzZTY0RW5jb2RlZDIwYnl0ZXM=”,
    "WhiteListStatus": "NotSet"
 },
 "SystemTraceAuditNumber": "3URHVZPIU87LOKA",
 "MACTransmissionNumber": ""
}
```

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 is using the Snap* Card Range Cache, Snap* Platform will query the platform cache and return the single Card Range along with the base64-encoded ThreeDsMethodData to be posted to the ACS URL.
- If the Merchant Application has a local cache, the ThreeDsMethodData can be created by generating a unique ServerTransactionId and base64 encoding the ServerTransactionId and Merchant-defined NotificationURL. Please note this ServerTransactionId should be included on the Authorize or AuthorizeAndCapture request.

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

The Merchant Application Authorize request will be identical to the "without Method Data" request, with the exception of setting the MethodCompletionIndicator to 'Completed'. If more information is needed about how Method Data is exchanged through this process, consult the appropriate card schemes or EMVCo specification here.

If the transaction was successfully 3-D Secure-authenticated, the transaction will continue on to Authorization. After Authorization is complete, the response will contain the following additional Authentication information:

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 App 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 and was not sent for Authorization:

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 Authorize or AuthorizeAndCapture 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 Authorize or AuthorizeAndCapture response will contain the Authorization approval details, including AuthenticationValue and AuthenticationECI fields as proof of authentication and SDKResponseInfo. If a Challenge is required in the Authorize or AuthorizeAndCapture 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": 5.5,
        "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,
        "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": 1,
                "UIType": 1
           },
           "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,
     "DeviceChannel": 1
}
     "TransactionStatusIndicator": 0
   },
   "ReportingData": {
          "Comment": "This is a comment",
          "Description": "12345678",
          "Reference": "12345678"
   },
   "IsOffline": false
}
```

Response

```
{
     "AdviceResponse": 0,
     "CommercialCardResponse": 0,
     "ReturnedACI": "NotSet",
     "Amount": 1.00,
     "CardType": 2,
     "FeeAmount": 0.0,
     "Status": 2,
     "StatusCode": null,
     "StatusMessage": "Service temporarily unavailable.",
     "TransactionId": "7B113CAFA216430EAC88C98DFD72E4F6",
     "ApprovalCode": "",
     "AVSResult": null,
     "OriginatorTransactionId": "402",
     "BatchId": "",
     "ServiceTransactionId": "",
     "CVResult": 0,
     "ServiceTransactionDateTime": {
           "Date": null,
           "Time": null,
           "TimeZone": null
     },
       "CardLevel": "",
       "Addendum": null,
       "DowngradeCode": "",
       "CaptureState": 6,
       "MaskedPAN": "402400XXXXXX8834",
       "TransactionState": 6,
       "PaymentAccountDataToken": "7b113caf-a216-430e-ac88-c98dfd72e4f6a3a5b186-71ed-4352-
bc48-8f898d8e4cfb",
       "IsAcknowledged": false,
       "RetrievalReferenceNumber": "",
       "Reference": "84ae9a5a-f4e6-4fbd-8df1-20d208df927a",
       "Resubmit": 0,
       "TransmissionNumber": null,
       "SettlementDate": "0001-01-01T00:00:00",
       "TransactionCode": "",
       "FinalBalance": null,
       "HostMessageId": "",
       "OrderId": "302",
       "Geolocation": null,
       "CashBackAmount": 0.0,
       "TerminalAccessToken": null,
       "PrepaidCard": 0,
       "Expire": "1225",
       "ErrorType": "",
       "AuthorizationServerUrl": "",
       "PaymentAuthorizationRequest": "",
       "ProcessedAs3D": false,
       "EMVDataResponse": null,
       "Level3Added": 0,
       "LastPANDigits": "8834",
       "BatchAmount": 0.0,
       "MessageAuthenticationCode": "",
       "TokenInformation": null,
       "ForcePostCode": "",
       "MerchantId": "123456789012",
       "TerminalId": "001",
       "BankResponseCode": "",
       "InitialEncryptionKeys": null,
       "IsPartialApproval": false,
       "EBTAvailableBalance": {
              "CashAvailableBalance": 0.0,
              "SNAPAvailableBalance": 0.0
       },
       "IndustryType": 2,
       "ThreeDSecureInformation": null,
       "ThreeDSInformation": {
              "TransactionStatus": 0,
              "AuthenticationECI": null,
              "DSTransactionId": null,
              "IsChallengeMandated": false,
              "ChallengeRequest": null,
              "ChallengeCancellationIndicator": null,
              "TransactionStatusReason": 0,
              "AuthenticationValue": null,
              "ACSTransactionId": null,
              "AuthenticationType": 0,
              "CardholderInformationText": null,
              "DSReferenceNumber": null,
              "ErrorCode": null,
              "ErrorDetail": null,
              "ErrorDescription": "Service temporarily unavailable.",
              "AcsUrl": null,
              "MerchantId": null,
                "MessageCategory": 0,
                "ProtocolVersion": 0,
                "ServerTransactionId": null,
                "WhiteListStatus": 0,
                "TokenResult": "",
                "Protocol1": null,
                "SCARequired": false,
                "ReasonForNotHonoringExemption": "",
                "ExemptionControl": 0,
                "SDKResponseInfo": {
                        "ACSOperatorId": "AcsOpId_4138359541",
                        "ACSReferenceNumber": "3DS_LOA_ACS_PPFU_020100_00009",
                        "ACSRenderingType": {
                                "Interface": "Native",
                                "UITemplate": "Text"
                        },
                        "ACSSignedContent": "",
                        "AppId": "",
                        "MaxTimeout": "5",
                        "TransactionId": "1d33eb63-88d4-40fa-8e8c-a9b0265cde95"
                },
                "AuthenticationTimestamp": "0001-01-01T00:00:00",
                "AuthenticationMethod": 1
        },
        "SystemTraceAuditNumber": "",
        "MACTransmissionNumber": ""
}
```

Challenge Authentication

Alternatively, the response can indicate a Challenge is required. This workflow is an extension to Frictionless Authentication (with or without Method Data). When the Challenge workflow is invoked, the initial Authorize call returns a Decline with TransactionStatus 'ChallengeRequired' on the response.

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.

Decline Response to Authorize Call

```
"AdviceResponse": "NotSet",
 "Amount": 100.00,
 "Status": "Failure",
 "CommercialCardResponse": "NotSet",
 "CardType": "Visa",
 "StatusCode": "3DSChallenge",
 "ReturnedACI": "NotSet",
 "FeeAmount": 0.00,
 "StatusMessage": "3DS challenge required. Resubmit transaction after challenge is
complete.",
 "ApprovalCode": "",
 "TransactionId": "A0E2B5A75B19449F8DADB17927345773",
 "AVSResult": null,
 "OriginatorTransactionId": "23133",
 "BatchId": "",
 "ServiceTransactionId": "",
 "CVResult": "NotSet",
 "ServiceTransactionDateTime": {
    "Date": null,
    "Time": null,
    "TimeZone": null
 },
 "CardLevel": "",
 "Addendum": null,
 "DowngradeCode": "",
 "CaptureState": "CaptureDeclined",

 "MaskedPAN": "402400XXXXXX8834",
 "TransactionState": "CaptureDeclined",
 "PaymentAccountDataToken": "a0e2b5a7-5b19-449f-8dad-b17927345773d7217bf3-4f03-45d0-8ce8-
d6dafa4014f7",
 "IsAcknowledged": false,
 "RetrievalReferenceNumber": "",
 "Reference": "e44b34d5-6edc-4f21-a661-434393e4c7e1",
 "Resubmit": "NotSet",
 "TransmissionNumber": null,
 "SettlementDate": "0001-01-01T00:00:00",
 "TransactionCode": "",
 "FinalBalance": null,
 "HostMessageId": "",
 "OrderId": "22133",
 "Geolocation": null,
 "CashBackAmount": 0.00,
 "TerminalAccessToken": null,
 "PrepaidCard": "NotSet",
 "Expire": "1225",
 "ErrorType": "3DSChallenge",
 "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,
 "EBTAvailableBalance": {
    "CashAvailableBalance": 0.00,
    "SNAPAvailableBalance": 0.00
 },
 "IndustryType": "Ecommerce",
 "ThreeDSecureInformation": null,
 "ThreeDSInformation": {
    "TransactionStatus": "ChallengeRequired",
    "AuthenticationECI": null,
    "DSTransactionId": null,
    "IsChallengeMandated": true,
    "ChallengeRequest":
"eyJtZXNzYWdlVHlwZSI6IkNSZXEiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwidGhyZWVEU1NlcnZlclRyYW5zSUQ
iOiJlNDRiM
zRkNS02ZWRjLTRmMjEtYTY2MS00MzQzOTNlNGM3ZTEiLCJhY3NUcmFuc0lEIjoiNWRhMDUwMWItOWZhZC00MjQzLWEwM
zAtMjA3YTU
1NzU1N2VlIiwiY2hhbGxlbmdlV2luZG93U2l6ZSI6IjAxIn0",
    "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.cipdev2.local/visa",
    "MerchantId": null,
    "MessageCategory": "NotSet",

    "ProtocolVersion": "NotSet",
     "ServerTransactionId": null,
     "WhiteListStatus": "NotSet"
 },
 "SystemTraceAuditNumber": "",
 "MACTransmissionNumber": ""
}
```

Resubmit with Challenge Response

After the Authorize or AuthorizeAndCapture 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. The ACS interacts with the Cardholder directly through a visible iFrame created in the Cardholder's browser and then 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 the Challenge Response, the application must call Resubmit with the Challenge Response data returned from the ACS as a string. This is required to initiate the Authorization part of 3-D Secure processing.

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

```
{
 "$type": "ResubmitTransaction, http://schemas.evosnap.com/CWS/v2.0/Transactions/Rest",
 "transaction": {
      "$type":"Resubmit3DSecure,http://schemas.evosnap.com/CWS/v2.0/Transactions/Bankcard",
      "TransactionId": "DAC5CEA2AF0440D1B70B8BF5C15AC22A",
      "ChallengeResponse":
"eyJ0aHJlZURTU2VydmVyVHJhbnNJRCI6ImEzZjYwY2ExLTg2MGItNDhkYS1iZmQ1LWY1ZGExNGExOGE0YyIsImFjc1R
yYW5zSUQiO
iJhYzgzNTZjNS0xOWY1LTQ2ZTYtYjU3Ni0zYmRmYjI5NGRlN2YiLCJtZXNzYWdlVHlwZSI6IkNSZXMiLCJtZXNzYWdlV
mVyc2lvbiI
6IjIuMS4wIiwidHJhbnNTdGF0dXMiOiJZIn0=",
      "ResubmitReason": “Resubmission”,
      "ProcessAsNon3DSecure": false
 },
 "ApplicationProfileId": "781738",
 "MerchantProfileId": "merchant 123"
}
```

If the transaction was successfully 3-D Secure authenticated, the transaction will continue on to Authorization.

If the transaction was not successfully 3-D Secure authenticated, the response will include the same ThreeDSInformation indicating why the Authentication failed. Depending on the reason for failure, the Merchant Application can choose to attempt that transaction again. This can be done by calling Resubmit and setting Resubmit3DSecure.ProcessAsNon3DSecure to ‘true’.

Out of Scope Transactions

3-D Secure 2.0 has presented options for Merchants to ensure the highest rate of Frictionless transaction processing. They have provided two categories of transactions where a Challenge is unlikely to be required.

First, the benefit of Out of Scope transactions are offered. The Out of Scope identifier represents transactions where the Cardholder is not available for Authentication. Because of this, there is little benefit to performing 3-D Secure Authentication on these transactions and they are considered out of scope for Authentication mandates.

Snap* Platform will bypass 3-D Secure Authentication and submit the transaction for Authorization if a transaction is identified by the Merchant Application as an Out of Scope transaction. The Authorization will identify the transaction as Out of Scope to achieve highest possibility of approval. There are four transaction types that qualify as Out of Scope:

1. Merchant Initiated Transactions, which is existing Snap* functionality, are defined as Out of Scope of Authentication since the merchant is initiating the payment on behalf of the cardholder. The initial MIT transaction where the cardholder is setting up the recurrence will require Authentication. This is the transaction where CardOnFile is First. ThreeRIIndicator will now be an additional required field for 3-D Secure 2.0 MIT transactions. Snap* will identify MIT Out of Scope transactions as any payment where:
    - BankcardTransactionData/CardOnFileInfo/InitiatedBy is Merchant and
    - BankcardTransactionData/CardOnFileInfo/CardOnFile is Repeat and
    - BankcardTransactionData/ThreeDSData/ThreeRIIndicator is NotSet.

2. MOTO transactions, which is existing Snap* functionality, are currently defined on the Merchant Profile. Snap* will identify MOTO transactions as any Merchant that is set up as a MOTO Merchant.

3. Inter-Regional transactions are defined as transactions where the Issuer or Acquirer are not based in Europe are also considered exempt from SCA. Therefore, European businesses will be able to

accept payments from non-European shoppers without problem. Snap* will identify Inter-Regional Out of Scope transactions as any payment where:

- BankcardTransactionData/ThreeDSData/IsInterRegionalTransaction is true.

4. Anonymous Prepaid Transactions are defined as transactions where the card is not tied to a bank account or an individual, but rather to a sum of money, which can originate from cash. For these transactions, the Cardholder is unknown to the Issuer. Snap will identify Anonymous Prepaid out of scope transactions as any payment where:
   - BankcardTransactionData/ThreeDSData/IsAnonymousPrepaidTransaction is true.

Since Out of Scope transactions are submitted for Authorization directly, the initial request does not require the additional 3-D Secure 2.0 values. This is in the rare case the Issuer rejects the Out of Scope attempt – Snap* will return a decline response to the Merchant Application with a status '65' for both MasterCard and Visa transactions. The Merchant Application can then make the decision if they would like to resubmit the transaction without the Out of Scope values. In this case, the Merchant will submit a new Authorize or AuthorizeAndCapture transaction with 3-D Secure Data set. This could result in a Challenge required and the Merchant Application should be prepared to handle this scenario.

Out of Scope transactions are only available for 3-D Secure 2.0 transactions, but are available for both MasterCard and Visa.

Exemptions

Second, the benefit of Exemptions are offered. Exemptions from the Challenge workflow 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 to transact without 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.

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 (or currency equivalent) 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 (or currency equivalent), 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

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

If the Merchant Application supports 2.1,
For Whitelisted and Secure Corporate transactions, MasterCard transactions will be sent directly for Authorization with the Exemption identified. These Exemptions have a high confidence of acceptance. For the remainder of the Exemptions listed above, MasterCard transactions will be sent for Authentication with the Exemption identified. For all Visa Exemptions, transactions will be sent directly for Authorization with the Exemption identified.
If the Merchant Application supports 2.2,
For Whitelisted and Secure Corporate transactions, MasterCard and Visa transactions will be sent directly for Authorization with the Exemption identified. These Exemptions have a high confidence of acceptance. For the remainder of the Exemptions listed above, MasterCard and Visa transactions will be sent for Authentication with the Exemption identified.

Sending directly on the Authorization request has the benefit of eliminating the latency associated with 3-D Secure processing. However, if the Merchant Application does not have a high degree of confidence that the Exemption applies, there may be higher risk of decline as the Authorization does not include the rich 3-D Secure Authentication data set. Additionally, sending directly on the Authorization request is not suitable when there is a delay between Authentication and Authorization, as the Cardholder may no longer be available in the event SCA is required.

The above are the Snap* defaulted workflows; however, the Merchant Application can override those at any time by setting BankcardTransactionData/ThreeDSData/ExemptionInfo/ExemptionControlParam to 'AuthorizationFlow' or 'AuthenticationFlow' accordingly.

Submitting an Exempted Transaction Request

When submitting the initial Authorize or AuthorizeAndCapture 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. The Merchant Application can set either AuthorizationFlow or AuthenticationFlow as described above, otherwise the transaction will automatically proceed with the Snap* default workflows.

In the Authorization workflow, Snap* sends the transaction directly for Authorization. If the Issuer rejects the Exemption, Snap* will return a decline response to the Merchant Application indicating the ReasonForNotHonoringExemption returned from the Issuer. The Merchant Application can then decide to re-process the transaction without the Exemption by calling Resubmit.

In the Authentication workflow, Snap* sends the transaction for Authentication. If the Issuer rejects the Exemption, a challenge will be required and the normal Challenge Authentication flow is executed.

If the issuer accepts the Exemption, Snap* sends the transaction for Authorization as an exempted transaction from SCA. If at this time, the Issuer rejects the transaction with SCARequired, Snap* will send a Decline response back to the Merchant Application indicating the ReasonForNotHonoringExemption. If the Merchant wants to re-process the transaction without the Exemption, the Merchant Application must call Resubmit. A Challenge can be expected here, and the Merchant Application and cardholder must perform the Challenge to continue. After the Challenge is completed, the Merchant Application can call Resubmit with the ChallengeResponse, and the transaction continues through the Authorization workflow.

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,
       "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",
       "PriorTransactionId": "2",
       "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"
               }
 },
}
}
```

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

Authorization

After a successful Authentication process is complete, Snap* will automatically set the Authentication values on the Authorize transaction and send it to the appropriate processor for Authorization. The response to the Resubmit will contain the Authentication fields from above with an Authorization response. Below is an example of a ‘Successful’ Authorization response in the Status field.

```
{
 "AdviceResponse": "NotSet",
 "Amount": 100.00,
 "Status": "Successful",
 "CommercialCardResponse": "NotSet",
 "CardType": "Visa",
 "StatusCode": "1",
 "ReturnedACI": "NotSet",
 "FeeAmount": 0.00,
 "StatusMessage": "APPROVED",
 "ApprovalCode": "8BAVBL",
 "TransactionId": "4F0CE321EA1A499697E09D657984AFC1",
 "AVSResult": null,
 "OriginatorTransactionId": "23143",
 "BatchId": "2022",
 "ServiceTransactionId": "42723331",
 "CVResult": "NotSet",
 "ServiceTransactionDateTime": {
     "Date": "2020-05-22",
     "Time": "22:56:14.039",
     "TimeZone": "-06:00"
 },
 "CardLevel": "",
 "Addendum": null
 "DowngradeCode": "",
 "CaptureState": "ReadyForCapture",
 "MaskedPAN": "402400XXXXXX8834",
 "TransactionState": "Authorized",
 "PaymentAccountDataToken": "dac5cea2-af04-40d1-b70b-8bf5c15ac22a7db0823f-d6b7-486e-b6c3-
2bb5d6b3cdd2",
 "IsAcknowledged": false,

 "RetrievalReferenceNumber": "141435311871",
 "Reference": "23143",
 "Resubmit": "NotSet",
 "TransmissionNumber": "4F0CE321EA1A499697E09D657984AFC1",
 "SettlementDate": "0001-01-01T00:00:00",
 "TransactionCode": "",
 "FinalBalance": null,
 "HostMessageId": "",
 "OrderId": "22143",
 "Geolocation": null,
 "CashBackAmount": 0.00,
 "TerminalAccessToken": null,
 "PrepaidCard": "NotSet",
 "Expire": "1225",
 "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,
 "EBTAvailableBalance": {
     "CashAvailableBalance": 0.00,
      "SNAPAvailableBalance": 0.00
 },
 "IndustryType": "Ecommerce",
 "ThreeDSecureInformation": null,
 "ThreeDSInformation": {
     "TransactionStatus": "SuccessfullyAuthenticated",
     "AuthenticationECI": "05",
      "DSTransactionId": "9e270423-2526-44bf-a816-04b5106a37c5",
     "IsChallengeMandated": false,
     "ChallengeRequest": null,
     "ChallengeCancellationIndicator": null,
     "TransactionStatusReason": "NotSet",
     "AuthenticationValue": "MTIzNDU2Nzg5MDA5ODc2NTQzMjE=",
     "ACSPublicKey": null,
     "ACSOperatorId": null,
     "ACSReferenceNumber": null,
     "ACSRenderingInterface": "NotSet",
     "ACSRenderingUITemplate": "NotSet",
     "ACSSignedContent": null,
     "ACSTransactionId": "ac8356c5-19f5-46e6-b576-3bdfb294de7f",
     "AuthenticationType": "Dynamic",
     "CardholderInformationText": null,
     "DSReferenceNumber": null,
     "ErrorCode": null,
     "ErrorDetail": null,
     "ErrorDescription": null,
     "AcsUrl": null,
     "MerchantId": null,
     "MessageCategory": "Payment",
     "ProtocolVersion": "v2_1_0",
     "ServerTransactionId": “QmFzZTY0RW5jb2RlZDIwYnl0ZXM=”,
     "WhiteListStatus": "NotSet"
 },
 "SystemTraceAuditNumber": "EQFRMUMFH32G803",
 "MACTransmissionNumber": ""
}
```

Follow-On Transactions

For Authorize and Capture and Authorize and Undo transaction workflows, Authentication only occurs on the Authorize part of the transaction. However, the 3-D Secure Authentication information returned on the Authorize response are required on the follow-on transaction requests, such as Capture, ReturnById, and Undo.

Authorize and Capture or Authorize and Undo

In this scenario:

- The Merchant Application sends an Authorize transaction with required/conditional 3DS data and BankcardTransactionData/Is3DSecure set to True.
- The Snap* Platform then processes the Authorize transaction with 3-D Secure Authentication and then returns BankcardTransactionResponse/ThreeDSInformation.
- The Merchant Application then sends a Capture or Undo request against the previous Authorize transaction with BankcardTransactionData/Is3DSecure set to False. No 3-D Secure Authentication occurs on this transaction because the Is3DSecure flag is set to False. Note that if the Is3DSecure flag is not set, it will default to False.
- The Snap Platform submits the Capture or Undo request.
- The front-end then maps the 3-D Secure Authentication data on the Capture request.

Card on File for Non-Payment Transactions

There are three 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.
3. Account data can be verified by Merchant Applications before processing a recurring payment.

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 ChallengeRequestedMandated 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 Resubmit for the Challenge completion and will follow the workflow detailed above.

Repeat Card on File Transactions

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 Merchant or 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 ChallengeRequestedMandated.

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

Merchant Verification

Merchant applications may wish to verify account data prior to processing recurring payments.

Because the merchant is requesting account verification and the cardholder is not in session, Requestor Initiated (3RI) authentication is invoked, meaning a Challenge will not be requested.

Merchant applications will call Verify with BankcardTransactionData/Is3DSecure as True, BankcardTransactionData/CardOnFileInfo/CardOnFile as Repeat, BankcardTransactionData/CardOnFileInfo/InitiatedBy as Merchant, BankcardTransactionData/ThreeDSData/AuthenticationIndicator as MaintainCard, BankcardTransactionData/ThreeDSData/ThreeRIIndicator 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.

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. On a Repeat Card on File transaction, the Merchant Application must submit CardOnFileInfo/OriginalTransactionId as the TransmissionNumber from the First Card on File transaction. Field length for TransmissionNumber is expected to be 20 characters.

If Merchant Failed Account Verification

If the Account Verification failed, the TransactionStatus will return with value NotAuthenticated.

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