Embedded Payments Technical Review

Useful Testing Tools in Sandbox

- American Express Payment Test App ▶ American Express Payment Test - Apps on Google Play - If you have a second Android device you can use this app to perform Tap to Pay Transactions to test Sandbox payments
- Seems XP Card 03 can trigger a declined transaction from the American Express Payment Test App for now ▶ American Express Payment Test - Apps on Google Play
- XP Card 01 can trigger signature flow for transaction value > $200

Common Cases

App Functionality (POS App)						
Authorisation Flow (POS Portal - POS Server)						
Test Case #	**Test Case**	**Implementation Being Tested**	**Required ✅ (y/n) ❌**	**Test Verification**	**Test in (Sandbox/Production/Both)**	**Test Results/Notes**

AF-1	• User must login to authenticate for POS Portal	• Merchant Admin Auth Flow
		• Merchant Authentication
			✅	☐ User required to authenticate before using POS Portal
				  ☐ When Authenticated, user has access to POS Portal
				  ☐ When not Authenticated, user has restricted access to POS Portal
					• Both	

			• Partner could send video/screen shot evidence of the above		

AF-2	• Has option to authorise POS	• Merchant Admin Auth Flow	✅	☐ There is an option to Authorise POS for Embedded Payments found through the UI
				  ☐ Confirm flow exists
				  ☐ Mentions that user will be redirected to Tyro to complete authorisation
				• Partner could send video/screen shot evidence of the above
					• Both	

AF-3	• Integrates with Tyro Integration Portal	• Merchant Admin Auth Flow • Redirect/Modal to Tyro Integration Portal	✅	☐ Check that once the Authorise POS for Embedded Payments flow has been kicked off by opening the Tyro Integration Portal in a new tab ☐ Check the url contains the POS ID and POS Reference as a query param. The POS ID will be their Client Id. e.g. https://integrate.stg.tyro.com/tap-to-pay?posId=client_id&posReference=pos_reference • Partner could send video evidence of the above	• Production	
AF-4	• Authorising POS and Merchant ID on Tyro Integration Portal updates list of enabled Tap to Pay Locations	• Merchant Admin Auth Flow • `TAPTOPAY_MERCHANT_ID_ENABLED` webhook • Fetch a Tap To Pay Merchant	✅	☐ Displays the correct list of MIDs that are owned by the current Merchant and allow user selection ☐ Displays the correct list of authorised locations for Embedded Payments ☐ Check what was enabled on the Tyro Integration Portal reflects this list • Partner could send video evidence of the above	• Production	

AF-5	• Admin merchant can manage their Embedded Payments trading locations (Sandbox)	• Merchant Admin Auth Flow • Merchant Authentication • List Tap to Pay Locations	✅	• Partner to send screen shots or video that show the following: ☐ How authorised locations are being shown ☐ That there is ability to manage these locations • Partner could send video evidence of the above	• Sandbox	
AF-6	• Admin merchant can choose to use a MID and manage their Embedded Payments trading locations	• Merchant Admin Auth Flow • Merchant Authentication • List Tap to Pay Locations	✅	☐ After completing the process on Tyro Merchant Portal, user navigates back to POS Portal ☐ Merchant can choose a MID to use from a list of MIDs that is owned by the user ☐ Merchant can see locations they are enabled for the chosen MID for Embedded Payments on POS Portal ☐ Their Locations are listed ☐ Shows list of authorised locations ☐ Management options present for each location e.g. user access ☐ Can Update Authorisation for each location ☐ Redirect back to Tyro Integration Portal to complete ☐ Deauthorised Locations not shown in authorised locations list • Partner could send video evidence of the above	• Production	

AF-7	• Admin merchant can manage their Embedded Payments users and devices (Sandbox)	• Merchant Admin Auth Flow - User/Device Management
		• Create a Reader
		• List Readers
		• Merchant Operator Enablement
		• Merchant Operator Disablement
			✅	☐ Partner to send screen shots or video that show the following:
				  ☐ How Admin users can manage devices
				  ☐ How Admin users can create Readers and 1 per device
				  ☐ How Admin users can disable users
				• Partner could send video evidence of the above
					• Sandbox

AF-8	• Admin merchant can manage their Embedded Payments users and devices	• Merchant Admin Auth Flow - User/Device Management • `TAPTOPAY_READER_STATUS_UPDATED` • Create a Reader • List Readers • Merchant Operator Enablement • Merchant Operator Disablement	✅	☐ Can manage Devices ☐ Can create Reader ☐ Single Reader per device ☐ Can deauthorise Merchant ID • Partner could send video evidence of the above	• Production
AF-9	• Only one Reader is used per Device	• Only one reader per device • Get a Reader	✅	☐ Check to see that this is happening through the device management process • Partner could send video evidence of the above	• Production *as sandbox would only have 1 reader
AF-10	• Deauthorise POS and Merchant ID on Tyro Integration Portal updates list of enabled Tap to Pay Locations	• Merchant Authentication • Redirect to Tyro Integration Portal • `TAPTOPAY_MERCHANT_ID_DISABLED` webhook • List Tap to Pay Locations	✅	☐ Displays the correct list of MIDs that are owned by the current Merchant ☐ Displays the correct list of enabled locations for Embedded Payment ☐ Check what was disabled on the Tyro Integration Portal reflects this list • Partner could send video evidence of the above	• Production

POS App

Test Case #	Test Case	Implementation Being Tested	Required ✅ (y/n) ❌	Test Verification	Test in (Sandbox/Prd/Both)	Test Results/Notes
PA-1	• User must authenticate in App to use Embedded Payments	• POS app requires authentication to use the reader	✅	☐ User required to authenticate before using POS App ☐ When Authenticated, user has access to POS App ☐ When not Authenticated, user has restricted access to POS App • Tyro to test	• Both	
PA-2	• App Should not be able to purchase with a 0 amount	• `amountInCents` must be positive and non zero	✅	☐ No ability to start Tap to Pay transaction (purchase) with a 0 dollar amount • Tyro to test	• Both	
PA-3	• App Should not be able to refund a 0 amount	• `amountInCents` must be positive and non zero	✅	☐ No ability to start Embedded Payments transaction (refund) with a 0 dollar amount • Tyro to test	• Both	
PA-4	• App should set the Order id as the transaction reference	• orderId used as transaction ref • transaction ref must be unique	✅	☐ Tyro to check logs/postman that the orderID is being set as the transaction reference ☐ Tyro to check logs/postman that the reference being sent is unique ☐ confirm between Partner and Tyro that these are matching up ☐ Partner to confirm they are giving us a unique reference • Partner could send video evidence of the above	• Both	
PA-5	• App should display the	• Surcharging	✅	Prerequisite: • surcharging enabled for Test MID on Tyro's side	• Both	

Test Case #	Test Case	Implementation Being Tested	Required (y/n)	Test Verification	Test in (Sandbox/Prd/Both)	Test Results/Notes
	amount customer paid including surcharges applied			☐ Tyro to check final amount paid screen after processing a transaction with surcharging applied ☐ Should see the amount as original total plus the surcharge applied		
PA-6	• App should record the total amount plus the final surcharge if one was added	• Surcharging	✅	Prerequisite: • surcharging enabled for Test MID on Tyro's side ☐ Partner to provide evidence they are recording the total amount paid including surcharges applied. Can be in the form of: ☐ Transaction history page ☐ Code snippets	• Both	
PA-7	• App should refund total amount plus any surcharges applied	• Surcharging	✅	Prerequisite: • surcharging enabled for Test MID on Tyro's side ☐ Tyro to check when performing a refund for a transaction with a surcharge applied, that the total includes the surcharge amount	• Both	
PA-8	• Surcharge Added to POS tax invoice	• Surcharging	✅	Prerequisite: • surcharging enabled for Test MID on Tyro's side ☐ Partner to provide evidence that the surcharge has been added to the tax invoice	• Both	

POS Server

Test Case #	Test Case	Implementation Being Tested	Required ✅ (y/n) ❌	Test Verification	Test in (Sandbox/Prd/Both)	Test Results/Notes
PS-1	• Client ID and Client Secret stored securely	• Tyro Authentication Flow	✅	☐ Partner to confirm they are storing these details on their server	• Both	

				on POS Server		
PS-2	• POS saves sales receipts for 540 days	• POS receipt compliance with scheme chargeback rules	✅	☐ Partner to confirm they are are storing sales receipts for at least 540 days. This is so that if a chargeback is filed, the merchant is able to provide proof of what was purchased at the time.	• n/a	

Tap To Pay API

Test Case #	Test Case	Implementation Being Tested	Required ✅ (y/n) ❌	Testing Steps Partner/Tyro	Test in (Sandbox/Production/Both)	Test Results/Notes

TA-1	• App can fetch a Tap to Pay Transaction by transaction ID	• Fetch a Tap to Pay Transaction	❌	☐ Check with Partner if they are using this - Tyro will document the usage
				1. Check if this can be visually confirmed in App UI
				  ☐ Perhaps it displays a previous transaction?
				  ☐ Other use cases? Describe
				☐ Tyro to check logs to see these calls are successful i.e. returns 200
				• Tyro to test on Production build
					• Production	

TA-2	• App can list Tap to Pay Transactions by reference	• List Tap to Pay Transactions	❌	☐ Check with Partner if they are using this - Tyro will document the usage
				1. Check if this can be visually confirmed in App UI
				  ☐ Perhaps it displays a list of previous transactions?
				  ☐ Other use cases? Describe
				☐ Tyro to check logs to see these calls are successful i.e. returns 200
				• Tyro to test on Production build
					• Production	

Android SDK Cases

Tap To Pay SDK Functions

Test Case #	Test Case	Implementation Being Tested	Required ✅ (y/n) ❌	Test Verification	Test in (Sandbox/Prd/Both)	Test Results/Notes
TS-1	• Initialise App	• ConnectionProvider() • Create Tap to Pay Connection • TapToPaySdk.createInstance(tyroEnv) • tapToPaySdk.init(activity, callback) • onInitResult() • tapToPaySdk.registerTransactionResultHandler(activity, callback)	✅	☐ Tap to Pay launches successfully No Errors such as: ☐ Attestation error ☐ Invalid connection secret error • Tyro to test	• Both	

TS-2	• App is setting POS Info correctly	• TapToPaySdk.createInstance(tyroEnv)
		• PosInfo
			✅	☐ Partner to send screen shots of code that shows the values being set for **PosInfo** and used accordingly
				  ☐ **posName** matches expected Partner's POS Name
				  ☐ **posVendor** matches what is expected of POS Vendor
				  ☐ **posVersion** matches a versioning format e.g. x.xx.xx or x.x
				  ☐ **siteReference** has some location reference
				☐ Tyro to check this matches what is expected for this POS
				
				Example PosInfo
				
				```\n1 val POS_INFO =\n2 PosInfo(\n3 posName = "Demo",\n4 posVendor = "Tyro Pa\n5 posVersion = "1.0",\n6 siteReference = "Sy\n7)\n```
					• Both	

TS-3	• Process Payment - Success (Sandbox)	• TransactionRequest(type, amount, reference)
		• tapToPaySdk.startTransaction(activity, transactionRequest - PURCHASE transaction type)
			✅	☐ Payment can be processed
				  ☐ Can tap card to pay
				  ☐ Success (App to display)
				☐ Tyro to check logs
				• Tyro to test
					• Sandbox	

		• tapToPaySdk.registerTransactionResultHandler(activity, callback) • Result Displayed				

TS-4	• Process Payment - **Success** (Prd)	• Same as above
		• `TAPTOPAY_TRANSACTION_SUCCESS` sent from Tyro
			✅	☐ Payment can be processed
				  ☐ Can tap card to pay
				  ☐ **Success** (App to display)
				☐ Tyro to check logs
				• Tyro to test
					• Production	

TS-5	• Process Payment - **Failed** (Sandbox)	• TransactionRequest(type, amount, reference) • tapToPaySdk.startTransaction(activity, transactionRequest - PURCHASE transaction type) • tapToPaySdk.registerTransactionResultHandler(activity, callback) • Result Displayed	✅	☐ Processing payment with $666.xx ☐ **Failed** status (App to display) ☐ Tyro to check logs • Tyro to test	• Sandbox	
TS-6	• App has option to provide customer with a Receipt	• Provide customer with receipt	✅	☐ Check after a transaction processes that there is an option to provide the customer with a receipt either digital or physical	• Sandbox	

TS-7	• App Can provide **Digital** or **Physical** Receipt	• tapToPaySdk.sendDigitalReceipt(transactionId, email) - For Digital Receipt
		• Provide customer with receipt
			✅	☐ Check after a transaction processes that there is an option to provide the customer with a receipt either digital or physical
				☐ Ensure receipt is using Tyro's customerReceipt content returned in the transaction result
				• Tyro to test
				If Digital Receipt:
				1. After transaction:
				  ☐ Digital receipt option shown that takes an email
				  ☐ Sending Digital receipt results in the given email receiving the Digital receipt
				If Physical Receipt:
				1. After transaction:
					• Production	

Test Case #	Test Case	Implementation Being Tested	Required	Test Verification	Test in (Sandbox/Production/Both)	Test Results/Notes
				☐ Physical receipt option shown that can print a receipt ☐ Printing Physical receipt results in the receipt being printed • Partner could send video evidence of the above		
TS-8	• Refund Payment (Sandbox) - Success	• TransactionRequest(type, amount, reference) • tapToPaySdk.startTransaction(activity, transactionRequest - REFUND transaction type) • tapToPaySdk.registerTransactionResultHandler(activity, callback) • Display Transaction Result	✅	☐ Refund option exists ☐ Processing refund results in ☐ Success status (App to display) ☐ Tyro can check logs • Tyro to test	• Sandbox	
TS-9	• Refund Payment (Prd) - Success	• Same as above • `TAPTOPAY_TRANSACTION_REFUNDED` webhook	✅	☐ Refund option exists ☐ Processing refund results in ☐ Success status (App to display) ☐ Tyro can check logs • Tyro to test	• Production	
TS-10	• Refund Payment - Failed (Sandbox)	• TransactionRequest(type, amount, reference) • tapToPaySdk.startTransaction(activity, transactionRequest - REFUND transaction type) • tapToPaySdk.registerTransactionResultHandler(activity, callback) • Display Transaction Result	✅	☐ Refund option exists ☐ Processing refund results in ☐ Failed status (App to display) ☐ Tyro can check logs • Tyro to test	• Sandbox	

Tyro Onboarding Check

Test Case #	Test Case	Implementation Being Tested	Required ✅ (y/n) ❌	Test Verification	Test in (Sandbox/Production/Both)	Test Results/Notes

TO-1	• Provided App id and Public certificate to Tyro	• Tyro onboarding • Correct Verification Data	✅	☐ Partner to tick this off before going Live	• Both	

iOS SDK Cases

SDK Functionality iOS						
Test Case #	Test Case	Implementation Being Tested	Required ✅ (y/n) ❌	Test Verification	Test in (Sandbox/Production/Both)	Test Results/Notes
SF-1	• Initialise App	• ConnectionProvider() • Create instance of TyroTapToPay • Create Tap to Pay Connection • TyroTapToPaySdk.connect() • Terms and conditions Presented/Accepted (If first time) • Reader Configuration Update (If update required)	✅	☐ Tap to Pay launches successfully No Errors such as: ☐ Attestation error ☐ Invalid connection secret error • Tyro to test	• Both	

SF-2	• App is setting POS Information correctly	• POSInformation	✅	☐ Partner to send screen shots of code that shows the values being set for **POSInformation** and used accordingly
				    ☐ **name** matches expected Partner's POS Name
				    ☐ **vendor** matches what is expected of POS Vendor
				    ☐ **version** matches a versioning format e.g. x.xx.xx or x.x
				    ☐ **siteReference** has some location reference
				☐ Tyro to check this matches what is expected for this POS
				
				Example POSInformation
				
				``
				1 private let posInformatio
				2 POSInformation(
				3 name: "
				``
					• Both	

```
4                    vendor:
5                    version
6                    siteRef
7                )
```

SF-3	• Incompatible Tap to Pay device throws error	• TyroTapToPaySdk.connect() - failure	✅	1. Use an iPhone that does not support Tap to Pay (< iOS 16.4, < iPhone XS) ☐ Should see error when trying to start Tap to Pay ☐ Handled by App UI • Tyro to test	• Both
SF-4	• Process Payment (Sandbox) - Success	• create instance of PosInformation • Create a TransactionDetail • tapToPaySDK.startPayment(transactionDetail) • Display Transaction Result	✅	☐ Process Tap to Pay transaction with some amount > 0 ☐ App informs user transaction was successful ☐ Tyro to check logs • Tyro to test	• Sandbox
SF-5	• Process Payment (Prd) - Success	• Same as above • `TAPTOPAY_TRANSACTION_SUCCESS` webhook sent from Tyro	✅	☐ Process Tap to Pay transaction with some amount > 0 ☐ App informs user transaction was successful ☐ Tyro to check logs • Tyro to test	• Production
SF-6	• Process Payment - Failed (Sandbox)	• create instance of PosInformation • Create a TransactionDetail • tapToPaySDK.startPayment(transactionDetail) • Display Transaction Result	✅	☐ Process Tap to Pay transaction with some amount > 0 ☐ App informs user transaction Failed ☐ Tyro to check logs • Tyro to test	• Sandbox
SF-7	• App has option to provide customer with a Receipt	• Provide customer with receipt	✅	☐ Check after a transaction processes that there is an option to provide the customer with a receipt either digital or physical	• Sandbox
SF-8	• App Can provide Digital or	• Provide customer with receipt • tapToPaySdk.requestDigitalReceipt(transactionId, email)	✅	☐ Ensure receipt is using Tyro's customerReceipt content returned in the transaction result If Digital Receipt: 1. After transaction:	• Production

	Physical Receipt		☐ Digital receipt option shown that takes an email ☐ Sending Digital receipt results in the given email receiving the Digital receipt • Tyro to test If Physical Receipt: 1. After transaction: ☐ Physical receipt option shown that can print a receipt ☐ Printing Physical receipt results in the receipt being printed • Partner could send video evidence of the above			
SF-9	• Refund Payment (Sandbox) - Success	• create instance of PosInformation • Create a TransactionDetail • tapToPaySDK.startRefund(transactionDetail) • Display Transaction Result	✅	☐ Refund option exists ☐ Processing refund results in Success status (App to display) ☐ Tyro to check logs • Tyro to test	• Sandbox	
SF-10	• Refund Payment (Prd) - Success	• Same as above • `TAPTOPAY_TRANSACTION_REFUNDED` webhook	✅	☐ Refund option exists ☐ Processing refund results in Success status (App to display) ☐ Tyro to check logs • Tyro to test	• Production	
SF-11	• Refund Payment - Failure (Sandbox)	• create instance of PosInformation • Create a TransactionDetail • tapToPaySDK.startRefund(transactionDetail) • Display Transaction Result	✅	☐ Refund option exists ☐ Processing refund results in Failed status (App to display) ☐ Tyro to check logs • Tyro to test	• Sandbox	

Other iOS Cases

Test Case #	Test Case	What is Being Tested	Required ✅ (y/n) ❌	Test Verification	Test in (Sandbox/Production/Both)	Test Results/Notes

OI-1	• In-app merchant education provided	• Apple Requirements	✅	☐ App has education materials present Education Material may be in the form: ☐ in-app merchant education video ☐ In-app payment acceptance tutorials ☐ Online training guides ☐ In-person, online, or in-app training ☐ Help center guidelines • Partner could send video or screen shot evidence of the above	• Both	
OI-2	• Apple terms and conditions accepted using Apple ID	• Using valid Apple ID to accept the terms	✅	☐ Tyro to tick if App appears to launch Embedded Payments	• Both	
OI-3	• Reader reconnects upon app resume	• iOS requires readers to be reconnected on app resume	✅	1. From a suspended state: ☐ App reconnects/resumes Embedded Payments ☐ Tyro can see logs connection created/verified? • Tyro to test	• Both	
OI-4	• Reader updates are handled	• iOS can handle reader updates	✅	Reader Updates happen: ☐ in the background ☐ Progress/Completion status verifiable ☐ Other? Describe • Partner could send video or screen shot evidence of the above	• Both	
OI-5	• Apple Entitlement	• Entitlements have been applied/issued	✅	☐ Partner to confirm they have entitlements from Apple	• Both	

Error Handling Check List

Error Handling Check List

Te st Ca se #	Test Case	Implementation Being Tested	Req uire d ✅ (y/n) ❌	Test Verification	Test in (Sandbox/Pr oduction/Bot h)	Test Results/Notes
EH -1	• Partner is aware of below errors and acknowl edges they are handling them	• Error Handling	✅	☐ Partner to tick this off before going Live ☐ Tyro to tick this off after testing in Sandbox and no unexpected errors ☐ Tyro to tick this off after testing in Production and no unexpected errors	• Both	

SDK Errors

You should be aware of the below errors and handle them accordingly.

- InvalidTransactionParamException
  - `amountInCents` must be positive and non zero
  - `amountInCents` length cannot exceed 8 digits
  - `reference` must not be blank
- INVALID_CONNECTION
  - Connection Secret is invalid
- SDK_VERSION_MISSING
- SDK_UPGRADE_REQUIRED
  - Current App version is < supported version

API Errors

You should be aware of the below errors but your integration should not trigger these.

- Create Tap to Pay Connection
  - 400 - When the provided payload is not valid.
  - 403 - When you don't have the right permissions to create a Connection for the provided Reader ID
  - 404 - When the provided `readerId` does not exist in our system
- Create a Reader
  - 400 - When the provided payload is not valid
  - 403 - When you don't have the right permissions to create a Reader for the provided location or Merchant ID
- List Readers
  - 403 - When you don't have the right permissions to retrieve the Readers for the provided location or Merchant ID
- Get a Reader
  - 403 - When you don't have the right permissions to retrieve a Reader for the provided location or Merchant ID
- Fetch a Tap to Pay Transaction
  - 403 - When you don't have the right permissions to access the transactions
- List Tap to Pay Transactions

- - **403** - When you don't have the right permissions to access the transactions
- List Tap to Pay Locations
  - **403** - When you don't have the right permissions to fetch the locations
  - **404** - When the provided `merchantId` does not exist in our system
- List Tap to Pay Merchants
  - **403** - When you don't have the right permissions to fetch the merchants
- Fetch a Tap to Pay Merchant
  - **400** - When the provided `merchantId` is not valid
  - **403** - When you don't have the right permissions to fetch the merchants
  - **404** - When the provided `merchantId` does not exist in our system

Postman Collection

The attached Postman collection contains examples of the Embedded Payments API that will need to be implemented on your server.

Connect Tap to ...on.json
22 Apr 2024, 05:27 am