# POS Cloud Integration

# Self-Assessment Checklist

v1.0

|  |  |
| --- | --- |
| **Test Information** | |
| Self-test Date | DD/MM/YYYY |
| Tester Name |  |
| Tester Email |  |
| Test Number (First, Second, etc.) |  |

# Introduction

## Objective

Provide a test plan for an Application to prove that it has correctly integrated with the POSCloud and Terminal so that it might be deployed successfully.

## Approach

For every Application integration, we strongly recommend that all applicable tests specified within this document be executed in their entirety for each test cycle. Each test case must be executed via the Application. Previous experience has warranted these stringent testing requirements, irrespective of the anomalies found in the testing cycles. This ensures that the resulting product has been thoroughly and rigorously tested prior to release.

## Deliverables

1. A completed test script.
2. Log output as described for the different sections which require log messages to be recorded.

## Document Scope

This document only validates that the Application has integrated correctly with the POSCloud for the listed functionality, and it meets a minimum standard for release. Other components, including the Payment Application, Host, etc., are not validated by this document.

The POSCloud functionality measured by this document is not exhaustive. Using APIs and configurations not described in this document must be verified independently.

Each release of the POSCloud is supported for a specific amount of time. If current version of the Application or the POSCloud are no longer supported, the Application should be updated and re-certified using supported versions.

Other documents define how an application must be submitted and deployed to the device; those processes are out-of-scope for this document.

# Contents

* [POS Cloud Integration Test Requirements](#POSCloudIntegrationTestRequirements-POS)
* [Introduction](#POSCloudIntegrationTestRequirements-Int)
  + [Objective](#POSCloudIntegrationTestRequirements-Obj)
  + [Approach](#POSCloudIntegrationTestRequirements-App)
  + [Deliverables](#POSCloudIntegrationTestRequirements-Del)
  + [Document Scope](#POSCloudIntegrationTestRequirements-Doc)
* [Glossary](#POSCloudIntegrationTestRequirements-Glo)
* [General Security Review](#POSCloudIntegrationTestRequirements-Gen)
* [Behavior](#POSCloudIntegrationTestRequirements-Beh)
  + [1. Session Management](#POSCloudIntegrationTestRequirements-1.S)
  + [2. General Behavior](#POSCloudIntegrationTestRequirements-2.G)
* [Payment](#POSCloudIntegrationTestRequirements-Pay)
  + [3. Payment Methods](#POSCloudIntegrationTestRequirements-3.P)
* [4. Error Handling](#POSCloudIntegrationTestRequirements-4.E)
* [Conclusion](#POSCloudIntegrationTestRequirements-Con)

# Glossary

|  |  |
| --- | --- |
| POSCloud | Set of APIs provided by Verifone for the purpose of integrating with a Verifone terminal. |
| Integrator | The party using the POSCloud in an Application to integrate with a Verifone terminal. |
| Application | The software application which consumes the POSCloud API. |
| Terminal | The hardware from Verifone which provides payment acceptance and customer interaction. |
| Payment Application | The software application on the Terminal which interacts with the POSCloud and the Payment Host. |
| Solution Stack | The entire set of software provided by Verifone and installed on the Terminal. |
| Estate Owner | The primary owner or manager of the terminals. |
| Host | The payment host, such as a gateway or processor. |
| Tester | A person working for the integrator to test the Application according to this document. |

# General Security Review

These questions provide a basic overview of the security of the application, but they are not comprehensive. Further documentation might be required for the specific region or Estate Owner.

|  |  |
| --- | --- |
| Is the Application certified by a third-party, such as PCI? |  |
| Does the Application go through any automated review tools, such as static code analysis? |  |
| Does the Application authenticate users? |  |
| Does the Application use secure practices such as mutual TLS server connections? |  |

# Behavior

## 1. Session Management

|  |  |  |  |
| --- | --- | --- | --- |
| **Test #** | **Description** | **Answer** | **P/F** |
| 1.00 | Show that the Application establishes the connection with the Verifone POS cloud CST environment using the actual API key | Provide logs. |  |
| 1.01 | Show that the application is able to handle session time out gracefully | Provide logs. |  |

**Additional Comments:**

## 2. General Behavior

**Note**: The below tests are required to assess the general behavior of the application so that it aligns with industry best practice.

|  |  |  |
| --- | --- | --- |
| **Test #** | **Description** | **Answer (Y/N)** |
| 2.01 | Bank data other than the TID is not shared. |  |
| 2.02 | Does not collect or permit access to any personal information for the cardholder or the merchant (e.g. phone numbers, names, health info). |  |
| 2.03 | Does not imitate the PIN entry screen. |  |
| 2.04 | Does not display or impersonate specific client branding |  |
| 2.05 | Does not display or impersonate Verifone branding |  |
| 2.06 | Does not use another company’s icon or logo |  |
| 2.07 | Does not incorporate any other's licensed content |  |
| 2.08 | Does not display inappropriate content (e.g. gambling, weaponry, adult content) |  |
| 2.09 | Does not display grammatically incorrect or confusing text |  |
| 2.10 | Does not display sporadic behaviors |  |
| 2.11 | Does not violate accessibility standards |  |
| 2.12 | Does not use or collect fingerprint or other biometric data |  |
| 2.13 | Does not involve user tracking or monitoring capabilities |  |

**Additional Comments:**

# API TESTING

We recommend using a development terminal to perform the payment tests, configuring the Training settings to simulate the responses to the given scenarios. It is also possible to simulate some of these scenarios against a specific host by using test cards provided by the payment host, or by connecting to a test host and sending specific amounts. It is important to use the training mode or test host with test cards to verify these, instead of using real information.

**Use the correct POS Cloud CST APIs for testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Description** | **Answer** | **Pass/Fail** |
| Sale | 1. Use the API [../poscloud/nexo/payment](http://localhost:9600/poscloud/nexo/payment) 2. Enter the sale request 3. Perform the transaction in the terminal: insert/swipe/tap the card to complete the transaction 4. Based on the response from the terminal/external terminal simulator, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Sale (signature confirmation) | 1. Use the API [../poscloud/nexo/payment](http://localhost:9600/poscloud/nexo/payment) 2. Send across the sale request (signature confirmation) 3. POSCloud sends a signature confirmation Input request to POS. 4. For signature confirmation, POS needs to send a signature confirmation Input response. Until POS sends the signature confirmation, the terminal will keep on waiting. Once the signature confirmation is sent, POSCloud will send the final transaction response. Signature confirmation Input Response looks like below ("ConfirmedFlag" true means signature is confirmed, false means not confirmed) 5. After signature confirmation, POS Cloud will send back the response | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Sale (voice referral) | 1. Use the API: [../poscloud/nexo/payment](http://localhost:9600/poscloud/nexo/payment) 2. Send the Sale (voice Referral) request 3. If the transaction amount ends with **.02**(Ex:-300.02) terminal sends a Voice referral input request for voice referral transactions confirmation. For that, POS needs to send a Voice referral input response for confirmation. 4. After getting a Voice referral Input request POS needs to send a voice referral input response for referral transaction confirmation. Until the voice referral confirmation, the terminal will keep on waiting. Voice referral input confirmation response looks like below ("1234" is the auth code in the example below) 5. POSCloud will return the payment response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Sale (voice referral with Signature) | 1. Use the API: ../poscloud/nexo/payment  2. Enter the Sale (voice referral with signature) request  3. If the transaction amount ends with .02(Ex:-50.02) terminal sends a Voice referral input request for voice referral transactions confirmation. For that, POS needs to send a Voice referral input response for confirmation.  4. After getting a Voice referral Input request POS needs to send a voice referral input response for referral transaction confirmation. Until the voice referral confirmation, the terminal will keep on waiting.  5. POSCloud sends a signature confirmation Input request to POS.  6. For signature confirmation, POS needs to send a signature confirmation Input response. Until POS sends the signature confirmation, the terminal will keep on waiting. Once got the signature confirmation is sent, POSCloud will send the final transaction response. Signature confirmation Input Response looks like below ("ConfirmedFlag" true means signature is confirmed, false means not confirmed)  7. Now, POSCloud will return the payment response | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Refund | 1. Use API: …/poscloud/nexo/payment  2. Send the refund request:  3. Perform the transaction in the terminal, insert/swipe/tap the card to complete the transaction  4. Based on the response from the terminal, the POS Cloud will send back the response | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Abort | 1. Use the API: …/poscloud/nexo/abort  2. Send the abort request.  3. If we perform abort API in the middle of the transaction, the current transaction aborted successfully with 200 success code and payment response we will be able to see abort response from the terminal, the POS Cloud will send back the response | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Get Last Transaction | 1. Use the API: …/poscloud/nexo/transactionstatus  2. Send the last transaction request  3. Based on the response from the terminal, the POS Cloud will send back the response | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Reversal/ Void | 1. Use the API: …/poscloud/nexo/reversal  2. Send the reversal/void request  3. Copy the data present under SaleToPOIData field from Sale response and paste under SaleToPOIData field in reversal request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Login | 1. Use the API: …/poscloud/nexo/login  2. Send the login request.  3. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Logout | 1. Use the API: …/poscloud/nexo/logout  2. Send the logout request  3. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Update Device Name | 1. Use the API: …/poscloud/nexo/updateDeviceName  2. Use method: PUT  3. Send the update device name request.  4. It will store device details on the Map. Based on the response from the Map, the POS Cloud will send back the response., which will look something like this:  {  "Result": "SUCCESS",  "ErrorCondition": null,  "AdditionalResponse": "Device name updated successfully"  }  Note: If we try to update deviceName in the Cloud mode we will get the below response  {  "Result": "FAILURE",  "ErrorCondition": "NOTALLOWED",  "AdditionalResponse": "Can't update Device name in Cloud mode"  } | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Reconciliation | 1. Use the API: …/poscloud/nexo/reconciliation  2. Use method: POST  3. Send the reconciliation request.  4. Based on the response from the terminal, the POS Cloud will send back the response | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Print | 1. Use the API: […/poscloud/nexo/print](http://localhost:9600/poscloud/nexo/print) 2. Use method: POST 3. Send the print request 4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Input Text | 1. Use the API: […/poscloud/nexo/input](http://localhost:9600/poscloud/nexo/input) 2. Use the method: POST 3. Send the Input text request. 4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Input confirmation | 1. Use the API: …/poscloud/nexo/input  2. Use the method: POST  3. Send the input confirmation request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Input menu | 1. Use the API: …/poscloud/nexo/input  2. Use the method: POST  3. Send the input menu request  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Display | 1. Use the API: …/poscloud/nexo/display  2. Use the method: POST  3. Send the display request as given below:  **Note**: Based on the InfoQualify(Status, Error, Display, POIReplication) field it will display different types of messages on the terminal.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Card Acquisition | 1. Use the API: …/poscloud/nexo/cardAcquisition  2. Use the method: POST  3. Send the card acquisition request.  4. Based on the response from the terminal, the POS Cloud will send back the response | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Enable Service | 1. Use the API: …/poscloud/nexo/processEnableService  2. Use the method: POST  3. Enable service API used for Card Acquisition end process. Send the request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Pre-Auth | 1. Use the API: …/poscloud/nexo/payment  2. Use the method: POST  3. Send the Pre-auth request  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) |  |
| Pre-Auth completion | 1. Use the API: …:/poscloud/nexo/payment  2. Use the method: POST  3. For the Pre-Auth completion transaction we need to take SaleToPOIData from the Pre-Auth transaction response and need to pass the Pre-Auth completion request for Pre-Auth Completion. (Ex:"SaleToPOIData": " {\"c\":\"31438\",\"p\":\"CREDIT\",\"r\":\"APPROVED\",\"rc\":\"5\",\"ts\":\"SUCCESS\",\"t\":\"4012563997\"}"). Send the pre auth completion request  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) | N/A |
| Pre-Auth Void | 1. Use the API: …/poscloud/nexo/reversal  2. Use the method: POST  3. For the Pre-Auth Void transaction we need to take SaleToPOIData from the Pre-Auth transaction response and need to pass the Pre-Auth Void request for the Pre-Auth void transaction.(Ex:"SaleToPOIData": " {\"c\":\"31438\",\"p\":\"CREDIT\",\"r\":\"APPROVED\",\"rc\":\"5\",\"ts\":\"SUCCESS\",\"t\":\"4012563997\"}"). Send the pre-auth void request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) | N/A |
| Gift Card Activate | 1. Use the API: …/poscloud/nexo/storedValue  2. Use the method: POST  3. Send the gift card activate request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) | N/A |
| Gift Card add value | 1. Use the API: …/poscloud/nexo/storedValue  2. Use the method: POST  3. Send the gift card add value request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) | N/A |
| Gift card cashout/close | 1. Use the API: …/poscloud/nexo/storedValue  2. Use the method: POST  3. Send the gift card cash out / close request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) | N/A |
| Gift Card Balance Inquiry | 1. Use the API: …/poscloud/nexo/balanceInquiry  2. Use the method: POST  3. Send the gift card balance inquiry request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) | N/A |
| Force Transaction | 1. Use the API: …./poscloud/nexo/payment  2. Use the method: POST  3. Send the force transaction request.  4. Based on the response from the terminal, the POS Cloud will send back the response. | Provide logs for both success and failure responses (Request and Response Jason payload) | N/A |

# 4. Error Handling

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test #** | | **Description** | | **Answer** | | | **P/F** |
| 4.1 | | When connection is lost provide some UI to the user that the connection is temporarily unavailable. | | Provide screenshot. |  | |
| 4.2 | | When the POS application is forcefully terminated while a payment is in-progress, confirm that the POS re-initializes and correctly recovers the payment status. | | Provide logs. |  | |

# Conclusion

|  |  |
| --- | --- |
| **Reviewer Information**  **(To be filled out by Verifone)** | |
| **Review Date** | DD/MM/YYY |
| **Reviewer Name** |  |
| **Reviewer Email** |  |
| **Review Number (First, Second, etc.)** |  |
| **Approved or Rejected?** |  |

**Additional Comments:**

For any issues please send your request on Jira tickets or Email to: POS.CLOUDSUPPORT@VERIFONE.com