Health Monitor Events

This section outlines a critical enhancement to Verifone's remote diagnostics solutions. This section provides the necessary background, identifies the problem statement, and offers an overview of the proposed solution. Subsequent sections delve into the operational aspects, including the target software stack required to implement the enhancements, a configuration guide for deployment, and other integration details.

Note

The Payment Application Health Monitor feature is available only for customers with VHQ premium licence.

Background

Remote diagnostics play a crucial role in reducing the total cost of ownership for card payment acceptance devices. Verifone currently offers robust remote diagnostics capabilities through its Terminal Management System (TMS) known as Verifone HQ (VHQ). The existing capabilities include:

- **Software Log Export:** ?Ability to export detailed software logs for analysis.
- **Remote Reboot:**?Capability to remotely reboot the device.
- **Diagnostic Alerts:** ?The device can alert the VHQ server with standard and custom diagnostics, such as low memory warnings, faulty battery notifications, and the number of reboots.

The alert mechanism primarily focuses on the remote diagnosis of hardware issues, facilitating proactive maintenance. However, recent production incidents caused by incorrect software and configuration updates have adversely impacted the devices' ability to process payments. These incidents underscore the need for enhanced remote diagnostics of the application software on the device.

Problem Statement

Despite comprehensive efforts in functional and operational testing of payment applications, discrepancies between test and production environments, along with user errors in terminal management systems, can adversely impact the operational integrity of payment devices. In the worst-case scenario, erroneous software or configuration updates could render devices incapable of processing transactions, resulting in significant revenue loss for merchants.

Currently, operational and functional anomalies are reported by end users, such as store managers, which triggers reactive troubleshooting and root cause analysis by Verifone's technical support and engineering teams. This reactive methodology is suboptimal and highlights the necessity for Verifone to implement proactive health monitoring and anomaly detection for payment devices.

There is a strategic opportunity to augment Verifone's remote diagnostic capabilities to encompass comprehensive health monitoring of application software deployed on payment devices. It is imperative to establish systematic checks and balances to proactively identify and mitigate issues arising from unintentional operational or human errors during remote operations. This monitoring should be integrated with advanced remote diagnostics and automated recovery mechanisms wherever feasible.

Solution

A dedicated diagnostics application, known as the Global Diagnostic Application (GDA), has been developed to collect health monitoring parameters via events from the payment application and transmit them to the Event Service on the Verifone server. GDA functions as a background service, capable of receiving event notifications from any application software on the device. It subsequently forwards these events to the Event Service. Additionally, the application autonomously monitors critical system-level parameters such as signal strength, network availability, and battery level.

The Event Service publishes these events to any subscriber via webhooks, with VHQ being one such subscriber. This architecture enables both VHQ and non-VHQ-based solutions to utilize this remote diagnostics feature.

Intended Audience

This guide is designed for users of the Health Monitor feature with the SCA Payment Application, including:

- **Helpdesk Teams** - Troubleshooting and diagnosing issues based on captured events.
- **Integrators** - Understanding event structures, integration with customer dashboards, and debugging.
- **Customers** - Gaining insights into trending issues in production and being proactive to arrest them. Customers can have access to these payment app health monitors via webhooks. This is available only for customers with VHQ premium licence.

Initialization of Functionality

This section outlines the essential prerequisites and steps required to begin using the Health Monitor feature in your lab environment to start with and then in production.

To get started, ensure that you have a terminal equipped with the required software stack that supports the Health Monitor capability. Once enabled, the terminal will automatically begin transmitting health monitor events to the Event Service on the Verifone server.

The GDA application is responsible for posting these events to the Event Service. Therefore, it is critical that your network allows outbound communication to the Event Service URL, which is specified in the Whitelisting section below. Please ensure that corresponding URL is accessible from your lab and prod environment to avoid communication issues.

To monitor these events on your end, you must register a Webhook with Verifone. This will allow the Health Monitor events to be pushed to your endpoint in near real-time, enabling proactive monitoring and diagnostics.

Finally, this section also covers the relevant configuration parameters on the GDA and SCA to successfully enable and manage the Health Monitor functionality.

Minimum Software Versions

For Terminals on ADK 4.7.x

Component	Version Details
ADK	4.7.42
OTA (For M440/M424 Only)	3.42.2
SCA UGP Solution	4.x.41-1
Global Diagnostic App (GDA)	2.0.0

For Terminals on ADK 4.10.x

Component	Version Details
ADK	4.10.5.2
SCA UGP Solution	4.x.41-1
Global Diagnostic App (GDA)	2.0.0

Installation Sequence for GDA Setup

The Following Sequence is for upgrading existing solutions with latest software mentioned above to have GDA running.

- ADK
- OTA (Valid for M440/M424 Only)
- SCA Solution
- SCA Configuration
- GDA and GDA configuration

Whitelisting URLs/IP Address

The Global Diagnostic App connects to the Verifone Event service to post the events. To ensure seamless connectivity and functionality of the Global Diagnostic App, it is essential to whitelist the following URLs/IP addresses on your network.

URL	Version Details
STAGING URL: cst2.test-gsc.vfims.com/event-service/events/device	Global Diagnostic App uses this end point to connect and post events in the staging environment for customer testing.
PROD URL: gsc.verifone.cloud/event-service/events/device	Global Diagnostic App uses this end point to connect and post events in the Production environment.

Registering for Webhooks

Customers can register to webhooks from Verifone and receive notifications about specific events. Here's how you can set up and monitor events using webhooks:

1. Registering for Webhooks To start monitoring events, you need to register your webhook URL with Verifone. Follow these steps:
    - Provide your webhook URL: This is the endpoint where you want to receive event notifications.
    - Specify the events: Indicate which events you want to monitor (e.g., Device Registration Failure, Encryption Setup Failure).
    - Authentication: Ensure your webhook endpoint is secured and can authenticate incoming requests from Verifone.
2. Webhook Event Format When an event occurs, Verifone will send a POST request to your webhook URL with the event data. The data will be in JSON format and include details such as event type, timestamp, device information, and specific event data. Here is a sample format:


```
{
  "objectType": "DeviceEvent",
  "eventId": "d94a63e5-3018-4265-a0da-36783d6661ad",
  "eventDateTime": "2025-03-26T05:33:27.001Z",
  "recordId": "0",
  "itemId": "0",
  "entityUid": "76e84daa-c954-4c6a-8f7f-09758c078669",
  "eoEntityUid": "76e84daa-c954-4c6a-8f7f-09758c078669",
  "source": "device",
  "received": "2025-03-26T05:33:28.500035811Z",
  "eventType": "",
  "content": {},
  "deviceType": "",
  "serialNumber": ""
}
```


3. Validating Webhooks in Non-Production Environment: Before deploying webhooks in a production environment, it's crucial to validate them in a non-production setting. Use lab devices to trigger the events and ensure your system correctly processes and responds to these events.
4. Querying Events from Specific Terminals: You can query events on the webhook from specific terminals by using the terminal's serial number or other identifiers. This allows you to filter and monitor events for particular devices, ensuring targeted diagnostics and troubleshooting.
5. Handling Webhook Notifications: When your system receives a webhook notification, it should:
    - Parse the JSON data: Extract relevant information such as event type, timestamp, and device details.
    - Log the event: Store the event data for future reference and analysis.
    - Trigger actions: Based on the event type, initiate appropriate actions such as alerting support teams, updating dashboards, or performing automated diagnostics.
6. Monitoring Dashboard Set up a monitoring dashboard to visualize the events received via webhooks. This dashboard can provide real-time insights into the health of your payment devices, highlight critical issues, and track trends over time.

Note

Please submit a JSD ticket with the webhook URL and the list of events you wish to monitor/register. Verifone support team will set this up so you can start receiving notifications via webhooks.

Related Configurations

Global Diagnostic App

For GDA the following configurations are part of the GDAConfig.ini under section [GDA].

Parameter Name	Description	Allowed Values	Default Value
PRIMARYADDRESS	End point of the Verifone server where the Events from Global diagnostic App are posted.	String of up to 100 characters. • gsc.verifone.cloud/event-service/events/device • cst2.test-gsc.vfims.com/event-service/events/device	cst2.test-gsc.vfims.com/event-service/events/device
THROTTLINGINTERVAL	Number of minutes over which the different terminals in customer estate should wait before posting the events to Verifone server. This helps avoid choke up of the Network Bandwidth during 24-hour reboot.	Numeric (minutes), 0 - 99 minutes	5

Parameter Name	Description	Allowed Values	Default Value
SIGNALTHRESHOLDLIMIT	This parameter defines the signal level in DBM below which the GDA app will notify the Verifone server as a Poor Signal Strength. This applies to GPRS/WiFi Signal	DBM Values: -100 to -1	-70
ENTITYUID	This is Verifone generated Identifier to the particular merchant.	String	Specific value to the merchant

SCA

Parameter Name	Description	Allowed Values	Defa Valu
HEALTHMONITOR	This parameter is used to enable or disable sending SCA application health monitor parameters as diagnostics events messages to GDA (Global Diagnostic App), which will be forwarded to the Verifone server to notify the consumer.	• 0 - Do not Send Events to GDA • 1 - Send Events to GDA	0

Parameter Name	Description	Allowed Values	Defa Valu
SCAPERFMETRIC	SCA application supports a feature of generating performance metric data for each command executed and send the response back to the POS. This parameter also enables Posting this summary as an event to Verifone server. SCAPERFMETRIC parameter is used to enable or disable this feature. This feature will be disabled by default.	• 0 - Does not calculate and return the Performance Metrics. • 1 - Calculate and Return Performance Metric in POS Response and send to GDA if GDA events are enabled	0
HEALTHMONITOR_EXCLCMDS	This parameter is used to set the list of commands excluded from the diagnostics Event Type 'ApplicationException'. The list of commands should be presented as separated by pipe (|).	Alphanumeric Maximum length - 50 Format: [pipe]commandName1[pipe]commandName2[pipe]	Blank

Architecture and Design

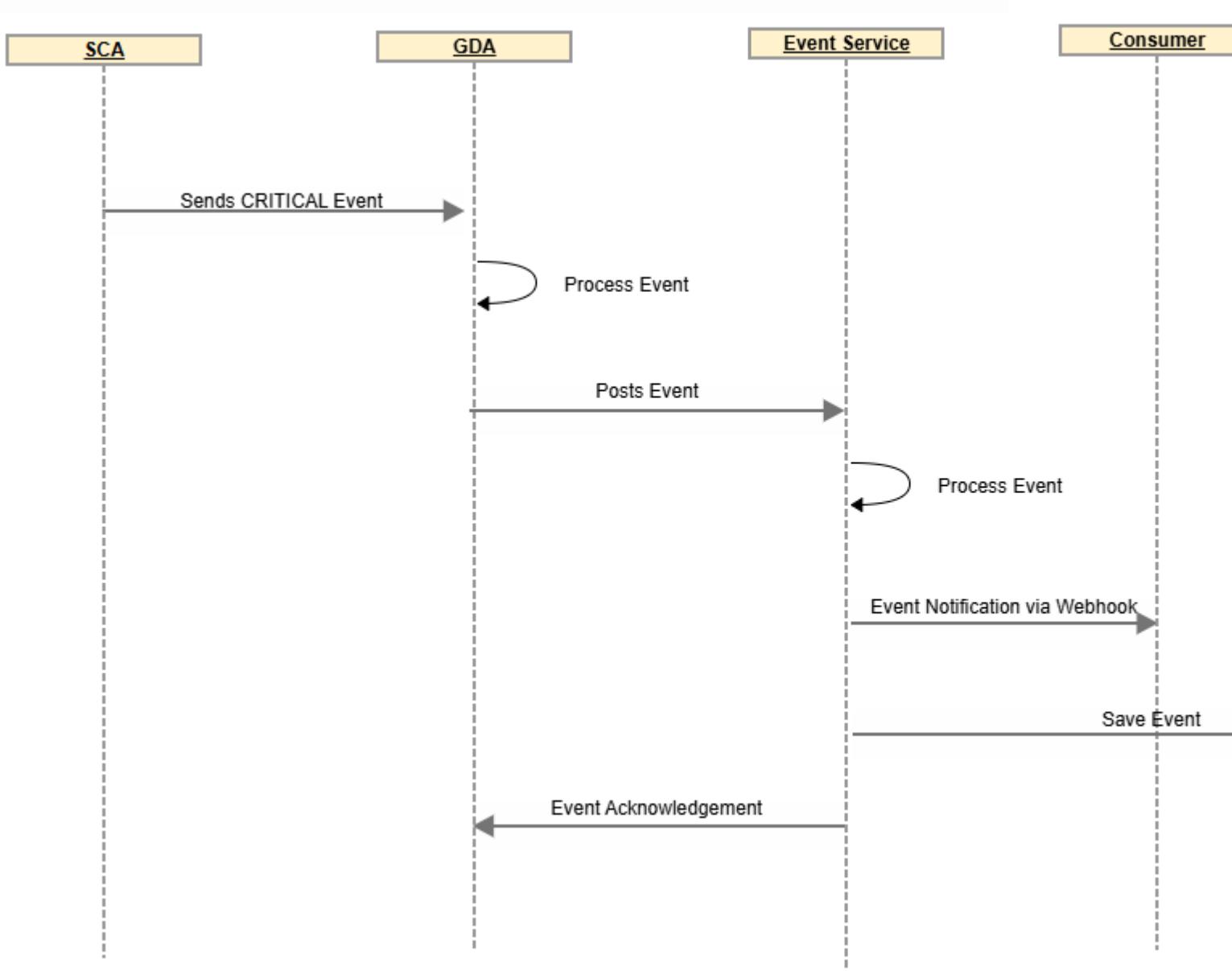
SCA payment application is responsible to identify the occurrence of specific events during the application flows, classify them and appropriately notify the Global diagnostic app regarding the event. The payment application provides the required data along with the event message which will help troubleshoot or gain further understanding regarding the event.

The events are in a JSON Format which are described in the further sections of this document.

There are 21 events as on the publication of this document and these Events can be classified as Critical and Non-Critical events. Critical events are those which may cause the payment application to fail initialization thus rendering it useless or when terminal does not have connectivity to the host causing SAF build up. The Critical events are sent to GDA as soon as they occur, allowing event consumers to act on them immediately. The Non-Critical are sent to GDA of once every 24 hours. This typically occurs after the 24-hour reboot.

These Events posted into Verifone event server will allow customers to register webhooks to notify them directly during the occurrence of the events. This allows customers the flexibility to maintain the required dashboard on their end for monitoring and acting on such events. Alternatively, customers can choose to sign up for sending emails and pulling additional diagnostic data from the terminals during the events.

Solution Architecture Overview



Events Throttling

To reduce the probability of network congestion caused by all terminals in customer estate posting the Diagnostic details at the same time after the 24-hour reboot occurs, the application supports Throttling logic.

The Throttling logic ensures that different terminals post the non-critical events at different instances, and the timing is calculated based on the Serial number as shown below.

If the serial number of the terminal is 321-456-789 and throttling window is set to 600 seconds (10 minutes), then the Throttling time is calculated as 321456789 Mod 600 which sets the throttling window to 189 seconds. The events will be posted 189 seconds after the 24-hour reboot window for such a terminal.

Events Format

Webhook Format:

Following is the generic format for the events that a subscriber will receive as a webhook notification.

Content Type: application/json


```
{
  "objectType": "DeviceEvent",
  "eventId": "d94a63e5-3018-4265-a0da-36783d6661ad",
  "eventDateTime": "2025-03-26T05:33:27.001Z",
  "recordId": "0",
  "itemId": "0",
  "entityUid": "76e84daa-c954-4c6a-8f7f-09758c078669",
  "eoEntityUid": "76e84daa-c954-4c6a-8f7f-09758c078669",
  "source": "device",
  "received": "2025-03-26T05:33:28.500035811Z",
  "eventType": "",
  "content": {
  },
  "deviceType": "",
  "serialNumber": ""
}
```


Where,

Fields	Description
objectType:	Type: string Value: DeviceEvent Type of Event or Object. Events from Health monitor and SCA would send value of DeviceEvent.
eventId:	Type: string <uuid> The internally assigned identifier for the event.
eventDateTime:	Type: string <date-time> The timestamp when the event occurred. This should be in local time with an offset to UTC specified to ensure this can be converted between time zones.
recordId:	Type: string
itemId:	Type: string

Fields	Description
entityUid:	Type: string <uuid> The Verifone allocated unique id for the entity this webhook/event is for.
eoEntityUid:	Type: string <uuid> The internally assigned identifier for the estate owner/customer for the event.
Source:	Type: string Value: device This internally assigned value indicates that the source of the event is from the Device. All health monitor events will have a source assigned to Device.
Received:	Type: string <date-time> (eventDateTime) This timestamp indicates when the event was received to the Verifone server Database.
eventType:	Type: string The value indicates the type of Event. The various possible event Types is provided in the next section.
content:	Type: object The associated event data. This is optional and if specified will vary according to the event type. Refer to Supported Device Events setion for more details on the Event Formats of all device events.
deviceType:	Type: string This value indicates the Device model from which the event has originated. Different Values supported - M400, M440, M424, P400, V400, E285, E280, P200
serialNumber:	Type: string. The device hardware serial number.

The table above describes the basic event format to which all events must adhere. The following section provides a detailed overview of each event type and outlines the content fields specific to each event. The other fields are

generic and apply to all events.

Supported Device Events

1. Device Registration Failure

Fields

Event Name:	DeviceRegistrationFailure
Priority:	Critical
Description:	This Critical event is triggered when the terminal is Unable to successfully perform Device Registration
Possible Cause:	This event may occur for the following reasons: • Merchant ID/Terminal ID details are missing during device registration. • Terminal is unable to make a connection to the host for registration. • Terminal has not been onboarded into the Gateway. • MID/TID has not been onboarded on the processor end.
Recommended Action:	• Ensure Merchant ID/Terminal ID details are configured correctly on the terminal in case of direct • Ensure Network Connectivity is intact. • Ensure Gateway/Processor onboarding is intact.

Event Format:	{
	"clientIdentifier": "17345800010001",
	"errorCode": "2029999",
	"failureReason": "Device Registration Failed",
	"merchantIdentifier": "005059233998",
	"responseText": "NOT_AUTHORIZED"
	}

Field Description:	• clientIdentifier - This field includes the CLIENT ID associated with the terminal. This is exclusive • errorCode - This field is the error codes are from the Host response code / Result Code values. • failureReason - Device Registration Failed. • merchantIdentifier - Merchant ID from the Processor/Gateway if available. • responseText - Response Text from the Processor/Gateway if available.

2. Encryption Setup Failure

Fields

Event Name:	EncryptionFailure
Priority:	Critical
Description:	This critical event is triggered when the Encryption Setup fails on the terminal. This is an indication that

Fields

Possible Cause:	This event may occur for the following reasons: • Encryption keys not injected on the terminal. Valid for VSP, ADE, AESDUKPT encryption • VSP Registration failure due to a host setup issue, or incorrect keys or terminal configuratio • VCL Bin update failure. • TRA Keys Download Failure. • Terminal is unable to make a connection during Encryption Registration with processor.
Recommended Action:	• Ensure encryption keys are loaded to the terminal. • Ensure Network connectivity is intact. • Ensure processor setup is accurate. • Ensure VCL Bin Update File configuration matches that at the server.

Event Format:	{
	"clientIdentifier": "17345800010001",
	"errorCode": "2029999",
	"failureReason": "Key Exchange Error",
	"merchantIdentifier": "005059233998",
	"responseText": "NOT_AUTHORIZED"
	}

Field Description:	• clientIdentifier - This field includes the CLIENT ID associated with the terminal This is exclusive • errorCode - This field is the error codes are from the Host response code/Result Code values. • failureReason - Key Exchange Error or VSP BIN UPDATE FAILED. • merchantIdentifier - Merchant ID from the Processor/Gateway if available. • responseText - Response Text from the Processor/Gateway if available.

3. Test Sale Failure

Fields

Event Name:	PreambleSaleFailure
Priority:	Critical
Description:	This critical event is triggered when the terminal fails to successfully complete the Test Sale or Test Voi
Possible Cause:	This event may occur for the following reasons: • Terminal is unable to make a connection to the host for the test sale/test void. • Gateway/Processor Declining the transactions to other issues. • Card Not supported. • Timeout or Cancellation during the test sale card read screen flow.

Fields

Recommended Action:

- Ensure the card used is applicable for the terminal configuration and is a valid card.
- Ensure Network Connectivity is intact.
- Ensure Gateway/Processor configuration allows the usage of the specific card used.
- If the decline originates from the processor, contact the processor or bank to determine the cause.

Event Format:

```
{
        "clientIdentifier": "17875200010001",
        "errorCode": "6",
        "failureReason": "Test Sale Failure",
        "merchantIdentifier": "000091156455",
        "responseText": "TRANS DENIED        ",
        "terminalIdentifier": ""
}
```

Field Description:

- clientIdentifier - This field includes the CLIENT ID associated with the terminal. This is exclusive
- errorCode - This field is the error codes are from the Host response code/Result Code values.
- failureReason - Test Sale Failure or Test Void Failure.
- merchantIdentifier - Merchant ID from the Processor/Gateway if available.
- terminalIdentifier - Terminal ID from the Processor/Gateway if available.
- responseText - Response Text from the Processor/Gateway if available.

4. Canadian Debit MAC Key Setup Failure

Fields

Event Name: MACKeyLoadFailure

Priority: Critical

Description: This Critical event is posted when the terminal is Unable to successfully update the MAC Key necessary

Possible Cause:

This event may occur for the following reasons:

- Failure to download the MAC Working Keys from the Processor/Gateway.
- Absence of the MAC Master Keys on the terminals.
- Incorrect configuration
- Missing Packages needed for Storing/retrieving the MAC Keys.

Recommended Action:

- Ensure the processor is properly setup for returning the MAC Keys for the terminal.
- Ensure Network Connectivity is intact.
- Ensure MAC Master keys are injected on the terminal.
- Ensure application configurations for MACing functionality is proper.

Fields

Event Format:

```
{
        "clientIdentifier": "17884700010001",
        "errorCode": "43",
        "failureReason": "MAC KEY Update Request Error",
        "merchantIdentifier": "700000209468",
        "responseText": "Check Open Batch and Processor to Verify Tran
}
```

Field Description:

- clientIdentifier - This field includes the CLIENT ID associated with the terminal. This is exclusive
- errorCode - This field is the error codes are from the Host response code / Result Code values. Thi
- failureReason - MAC KEY Update Request Error.
- merchantIdentifier - Merchant ID from the Processor/Gateway if available.
- responseText - Response Text from the Processor/Gateway if available.

5. Application Abrupt Restarts or Abrupt Terminal Reboots

Fields	**Description**
Event Name:	AppAbruptRestarts
Priority:	Critical
Description:	This Critical event is posted when the application detects the terminal rebooted or application restarted i idle screen.
Possible Cause:	This event may occur for the following reasons: • Application Exception that may force the merchant to reboot the terminal. • Application not responding to the commands from the POS forcing merchant to reboot. • Application restarts when ADK kills and restarts the application due to memory issues. • Installation of packages occurring during the transaction flow and consequently restarting th
Recommended Action:	• Capture the required diagnostics from the terminal such as Terminal logs. • Identify from the store if there was any reason terminal was manually restarted or any power failu • Capture POS Logs. • Check for any VHQ installations scheduled for the incorrect time.

Event Format:

```
{
        "Data": "Last Active Operation before terminal rebooted was :
}
```

Field Description: Data - This field indicates the state of the terminal before the unexpected reboot occurred. It provides the terminal was within an Active session or not.

6. Host Connection Status

Fields

Event Name: HostConnectionStatus

Fields

Priority:	Critical
Description:	This Critical event is posted when the application detects that the connection to the payment gateway or successful connection to the host.
Possible Cause:	This event may occur for the following reasons: • Loss of connectivity to the host/processor. • SAF error code from the Gateway indicating a loss of connection to the end point. • Network loss.
Recommended Action:	• Capture the required diagnostics from the terminal such as Terminal logs. • Validate the network availability to the terminal.

Event Format:	{
	"Data": " Terminal Could not connect to cert.api.vfipayna.com"
	"errorCode": "2",
	"failureReason": "Host Connection Failed",
	"status": "ConnectionFailure",
	"transactionIdentifier": "000262|1.00|400296******4803|DEBIT|V
	}

Field Description:	• Data - ◦ If the status is Connection Restored - “Terminal Connection to Host restored” ◦ If the status is Connection Failure - “Terminal Could not connect to”, followed by the • errorCode - This field is the error codes are from the Host response code / Result Code values. Thi • failureReason - Host Connection Failed - Only if status is ConnectionFailure • Status - ConnectionFailure or ConnectionRestored • Transaction Identifier - Invoice | Transaction Amount (Two decimals implied) | Masked PAN | Pa ◦ Payment Type - CREDIT, DEBIT, EBT, GIFT, FSA, PRIV_LBL ◦ Card Brand - VISA, MASTERCARD, DISCOVER, AMEX, UNIONPAY, JCB, DEB

7. Multiple SAF Retry

Fields

Event Name:	SAFMultipleRetry
Priority:	Critical
Description:	This Critical event is posted when the application detects that the same SAF Transaction is being forwar be investigated.
Possible Cause:	This event may occur for the following reasons: • End Processor/Gateway has rejected the transaction previously and returned a SAF error co • Error on the terminal causing failure to update the status of the SAF transaction even if it is

Fields

Recommended Action:
- Capture the required diagnostics from the terminal such as Terminal logs.
- Check the availability of the end processor if using a Gateway solution.
- Check from the processor/gateway to ensure the transaction is not already successfully processed.

Event Format:

```
{
        "Data": "SAF Forward Exceeded",
        "ErrorCode": "43",
        "ResponseText": " Communication error: Can't Connect to the Ho
        "transactionIdentifier":
        "123456|232|222360******0203|CREDIT|MASTERCARD|0009"
}
```

Field Description:
- Data - SAF Forward Exceeded.
- ResponseText - Response from the Host causing the SAF to be retried.
- ErrorCode - Error Code causing the retry of the SAF transaction.
- Transaction Identifier - Invoice | Transaction Amount (Two decimals implied) | Masked PAN | Pa
    - Payment Type - CREDIT, DEBIT, EBT, GIFT, FSA, PRIV_LBL
    - Card Brand - VISA, MASTERCARD, DISCOVER, AMEX, UNIONPAY, JCB, DEB
    - SAF_NUM - Locally generated SAF number for this particular transaction based on v

8. Application Crash or Hang

Fields

Event Name: ApplicationException

Priority: Critical

Description: This Critical event is posted when there is an exception, such as an application crash or hang during proc

Possible Cause:

This event may occur for the following reasons:
- Incorrect implementation at some stack on the terminal.
- Memory corruptions
- Deadlock conditions

Recommended Action: Capture the required diagnostics from the terminal such as Terminal logs and capture the screen on the t

Event Format:

```
{
        "AppName": "SCA",
        "data": "Suspecting a hang situation here as the application h
        "failureReason": "ApplicationHang"
}
{
        "data": "SCA App with main Thread[2661023744] is shutting down
        "failureReason": "ApplicationCrash",
        "signal": "6"
}
```

Fields

Field Description:
- AppName - Name of the application which is Hung/not functioning.
- failureReason - ApplicationHang or ApplicationCrash
- Data - For Application Hang, the Field Denotes which command that the terminal is attempting to
- Signal - This field is valid for ApplicationCrash only and denotes the error/signal that occurred ca

9. Payment Ready/ Not Ready State

Fields	Description
Event Name:	PaymentReady
Priority:	Critical
Description:	This Critical event is posted when the application is ready to accept payments or is not in a state to proce deployments or replacements of terminal to know if the terminal has initialized successfully and is ready
Possible Cause:	This event may occur for the following reasons: • Failure to initialize one of the components due to missing configuration/packages. • Failure to obtain the Network. • Absence of Host Configuration on terminal. • Failure to make a connection to the host.
Recommended Action:	• Capture the required diagnostics from the terminal such as Terminal logs if Status indicates NotRe • Ensure required configurations such as Merchant ID/Terminal ID is available on terminal.

Event Format:

```
{
        "Status": "Ready"
}
{
        "Status": "NotReady",
        "failureReason": "PosCommsInitializationInProgress"
}
```

Field Description:
- Status - The status field will indicate whether it is ready or not. Possible values are Ready/NotRea
- failureReason - This field contains the reason why the device is not ready.

10. Host Response Timeout

Fields	
Event Name:	HostResponseTimeout
Priority:	High
Description:	This High Priority event is posted when the application indicates it did not get a response back for the po

Fields

Possible Cause:	This event may occur for the following reasons: • Application Does receive a response from the Gateway/Processor. • Gateway/Processor indicates that it did not receive a response from its endpoint.
Recommended Action:	• Check for any network instability during the time this occurred. • Validate if there were any end processor issues during this time. • Ensure transaction has been reversed.

Event Format:	{
	"Data": "Terminal got timed out from cert.api.vfipayna.com",
	"errorCode": "0",
	"failureReason": "ResponseTimeout",
	"transactionIdentifier": "111111|1.00|651000******0844|CREDIT|
	}

Field Description:	• Data - This field denotes the URL against which the Response Timeout occurred. • ErrorCode - Error Code causing the retry of the Timeout. It would contain TOR error code or no D • failureReason - ResponseTimeout • Transaction Identifier - Invoice | Transaction Amount (Two decimals implied) | Masked PAN | Pa ◦ Payment Type - CREDIT, DEBIT, EBT, GIFT, FSA, PRIV_LBL ◦ Card Brand - VISA, MASTERCARD, DISCOVER, AMEX, UNIONPAY, JCB, DEB ◦ SAF_NUM - Locally generated SAF number for this particular transaction based on v

11. POS Disconnection

Fields

Event Name:	POSDisconnection
Priority:	High
Description:	This High event is posted to notify a Disconnection from the POS. This can signify that the POS and Ter
Possible Cause:	This event may occur for the following reasons: • Network Failure between the POS and the terminal. • POS closing the connection prior to the application responding to a command.
Recommended Action:	• Check for Network instability between the Terminal and POS. • Capture required diagnostics such as terminal logs. • Identify if the POS has closed the connection and capture POS logs. • Ensure the transaction status is correctly updated on the POS.

Fields

Event Format:

```
{
        "InsideSession": "false",
        "Interface": "Bluetooth",
        "PosIdentifier": "C4:C3:6B:6D:13:36,MPOS1234",
        "Reason": "PairingDisconnect"
}
{
        "InsideSession": "true",
        "Interface": "TCP",
        "PosIdentifier": "10.80.10.52",
        "Reason": "AbruptDisconnect"
}
```

Field Description:

- InsideSession - Indicates if the application was within a session when the POS disconnection occu
- Interface - Indicates the interface over which the POS Disconnection has occurred. Possible values
- POS Identifier - Indicates the Bluetooth MAC Address and BlueTooth Name if the POS disconnec
- Reason - Indicates the reason for the POS Disconnection.
- Possible values include PairingDisconnect, AbruptDisconnect.

12. Last Transaction Queries

Fields	Description
Event Name:	LastTransactionQueries
Priority:	High
Description:	This High event is posted when the terminal receives Last Transaction query from the POS. The LAST Tran Command is typically sent to the terminal if there is some disconnect between POS and terminal causing POS to be out of sync.
Possible Cause:	This event may occur for the following reasons: • Host Timeout occurred and terminal/POS do not have the final status of the transaction prompting the POS to initiate LAST_TRAN. • Network glitch causing the Terminal response to get dropped and not ready the POS. POS would initiate the LAST_TRAN to know the final status.
Recommended Action:	• Check for Network instability between the Terminal and POS. • Capture required diagnostics such as terminal logs. • Check if the final status of transaction was finally obtained. • Ensure there is no situation of customer being charged multiple times due to unknown transaction status.
Event Format:	No additional fields.

13. SAF Details

Fields

Event Name:	SAFDetails
Priority:	High
Description:	This high priority event is posted by the application on every restart or reboot when there are SAF transa
Possible Cause:	This is a generic event posted during the application startup which would help merchant understand the s
Recommended Action:	• The merchant can monitor this event to understand if the terminal has been in Offline mode and ha • Terminal should typically not be offline for a long time and merchant should watch out for a large • Troubleshoot with the store to ensure network is properly setup and terminal has a valid IP addres • Monitor SAF processing using SAF Query and if transactions are not being processed, work with

Event Format:

```
{
        "SAFDays": "0",
        "declinedAmount": "25840",
        "declinedCount": "317",
        "deferredAmount": "0",
        "deferredCount": "0",
        "notProcessedAmount": "0",
        "notProcessedCount": "0",
        "pendingAmount": "0",
        "pendingCount": "0",
        "preAuthAmount": "0",
        "preAuthCount": "0",
        "processedAmount": "380",
        "processedCount": "7",
        "totalAmount": "26220",
        "totalCount": "324"
}
```

Field Description:

- SAFDays - Number of Days since the Oldest Eligible SAF. All amount values mentioned below a
- declinedAmount - Total Amount of Declined Transactions.
- declinedCount - Total Number of Declined SAF Transactions.
- deferredAmount - Total Amount of Deferred Transactions.
- deferredCount - Total Number of Deferred transactions.
- notProcessedAmount - Total Amount of transactions not processed.
- notProcessedCount - Total number of transactions not processed.
- pendingAmount - Total transaction Amount of currently eligible transactions.
- pendingCount - Total number of transactions still in eligible state.
- preAuthAmount - Total transaction amount of Pre-Authorization transactions.
- preAuthCount - Total number of Pre-Auth transactions.
- processedAmount - Total transaction amount of successfully processed transactions.
- processedCount - Total number of transactions which are successfully processed.
- totalAmount - Total transaction amount of all transactions in SAF.
- totalCount - Total number of transactions in SAF.

14. Duplicate Transactions

Fields

Event Name:	DuplicateTransactions
Priority:	High
Description:	This high priority event is posted by the application whenever it detects the duplicate transaction. This e
Possible Cause:	• Duplicate transaction was attempted since previous transaction result was not obtained by the POS • Valid attempt of purchase by same customer for same transaction amount.
Recommended Action:	• Capture diagnostic data such as terminal logs and POS Logs. • Identify if there are large number of duplicate transactions and check the POS logs if it was a genu

Event Format:

```
{
        "dupTransactionIdentifier": "000932|9999900|374245*****1003|CR
        "failureReason": "DuplicateTransaction",
        "transactionIdentifier": "000933|9999900|374245*****1003|CREDI
}
```

Field Description:

- dupTransactionIdentifier - Transaction details of the previous transaction which has been identifie
- failureReason - Duplicate Transaction.
- transactionIdentifier - Transaction details of current transaction which was not processed due to th
    - Transaction Identifiers - Invoice | Transaction Amount (Two decimals implied) | Mas

15. Network Down

Fields	Description
Event Name:	NetworkDown
Priority:	High
Description:	This high priority event is sent by application whenever it detects a network down event with any of the such as WIFI, GPRS, Bluetooth, USB or Ethernet. This event would signify the health of the Network. N network or stable network would result in a greater number of offline transactions or loss of business du offline state.
Possible Cause:	• Faulty hardware such as network switch, cables etc. • Moving out of WIFI/GPRS ranges and hence resulting in network drop. • Network provider outage due to some technical/environmental issues.
Recommended Action:	Identify the stores where the Network Down events occur the most and check for network stability conc another transaction or if the transactions were re-attempted because status of previous transaction was u

Event Format:

```
{
        "Data": "",
        "Interface": "Wifi",
        "failureReason": "Network Down"
}
```

Fields	Description
Field Description:	• Interface - Indicates the Interface which received the Interface Down even from OS/ADK. This fie following values - WiFi, GPRS, Ethernet, Bluetooth, IP OVER USB. • FailureReason - Network Down

16. Poor Signal Strength

Fields	Description
Event Name:	PoorSignalStrength
Priority:	High
Description:	The application will send this event whenever it detects that the signal strength of the WIFI or GPRS net Refer to Related Configurations section to obtain details on SIGNALTHRESHOLDLIMIT parameter.
Possible Cause:	• Terminal moving out of the recommended operating ranges. • Incorrect placement of the Wireless access points.
Recommended Action:	• Review the signal strength of WIFI / GPRS at different points in the store and plan to add required • Update the configurations of the wireless access points to use a wireless channel which has less tra
Event Format:	`{ "Data": "Poor Signal Strength DBM:-71|sRSSI:3|Signal Percent:2 "Interface": "Wifi" }`
Field Description:	• Data - Pipe “|” separated values mentioned below • Signal strength in DBM | Signal in RSSI | Signal in % • Interface - WiFi or GPRS

17. Low Battery

Fields	Description
Event Name:	LowBattery
Priority:	High
Description:	The application will send this event for battery-operated devices like E285, E280, V400 whenever the battery level falls below the threshold. The default threshold in the ADK level is 25%.
Possible Cause:	• Terminal not charged correctly. • Degraded Battery health.

Fields	Description
Recommended Action:	• Review the LowBattery events from various terminals and identify the battery life. Investigate battery health if the LowBattery occurs often. • Review the process followed by merchants to ensure terminal is placed on charging points at required intervals.
Event Format:	`{ "Data": "Battery Level is 13 %" }`
Field Description:	Data - Denotes the battery level in %

18. Host Declined Transactions

Fields	
Event Name:	HostDeclinedTransactions
Priority:	Medium
Description:	The application will send this event for every declined transaction from the Gateway/Processor. The eve
Possible Cause:	• Valid declines from the issuer due to insufficient funds, incorrect PINs etc. • Gateway/Processor setup or processing issues. • Issues in the message format posted by the terminal.
Recommended Action:	Review the declined transactions and work with the Gateway/Processor to identify the reason for transac
Event Format:	`{ "ErrorCode": "6", "ResponseText": "TRANS DENIED", "failureReason": "TransactionDeclined", "transactionIdentifier": "71234|001|476173******0023|DEBIT|VIS "transactionType": "SALE" }`
Field Description:	• ResponseText - Response from the gateway/processor during a decline. • ErrorCode - Error Code from gateway/processor for transaction decline. • Transaction Identifier - Invoice | Transaction Amount (Two decimals implied) | Masked PAN | Pa ◦ Payment Type - CREDIT, DEBIT, EBT, GIFT, FSA, PRIV_LBL ◦ Card Brand - VISA, MASTERCARD, DISCOVER, AMEX, UNIONPAY, JCB, DEB • Transactiontype - Type of transaction such as SALE, REFUND etc.

19. Transaction Performance Time

Fields

Event Name:	TransactionPerformanceTime
Priority:	Medium
Description:	In this event, the application records the minimum, maximum, and average time taken for SALE transac
Possible Cause:	This is a generic event posted during the application startup which would help merchant understand the
Recommended Action:	• Monitor terminals/stores with lower average transaction performance and pull the required diagno • Work on merchant training if the lower performance times are due to process being followed at th

Event Format:


```
{
        "cardTokenTransactions": {
        "avgTime": "2.118",
        "maxTime": "2.118",
        "minTime": "2.118",
        "noOfTransactions": "1",
        "transactionType": "SALE"
        },
        "emvCTLSTransactions": {
        "avgTime": "12.9843",
        "maxTime": "15.567",
        "minTime": "10.375",
        "noOfTransactions": "3",
        "transactionType": "SALE"
        },
        "emvCTTransactions": {
        "avgTime": "17.0627",
        "maxTime": "30.824",
        "minTime": "7.567",
        "noOfTransactions": "10",
        "transactionType": "SALE"
        },
        "manualTransactions": {
        "avgTime": "25.011",
        "maxTime": "27.180",
        "minTime": "22.842",
        "noOfTransactions": "2",
        "transactionType": "SALE"
        },
        "msrTransactions": {
        "avgTime": "21.95",
        "maxTime": "30.101",
        "minTime": "6.851",
        "noOfTransactions": "4",
        "transactionType": "SALE"
        },
        "passThroughTransactions": {
        "avgTime": "9.813",
        "maxTime": "9.813",
        "minTime": "9.813",
        "noOfTransactions": "1",
        "transactionType": "SALE"
        }
}
```

Field Description:

Multiple containers in the content field for this event type will denote the transaction performance across

- avgTime - Average time of all transactions in this category.
- maxTime - Maximum Transaction time.
- minTime - Minimum transaction time.
- noOfTransactions - Number of transactions across which the average was obtained, or max/

The various containers are based on the different card entry modes - cardTokenTransactions, emvCTLS

20. Fallback Transactions

Fields	Description
Event Name:	FallbackTransactions
Priority:	Medium

Description:

The application will send this event whenever it detects the following scenarios:

- It could not read the EMV CT card after the maximum attempts.
- It could not read the EMV CTLS card after the maximum attempts.
- It could not read the MSR card after the maximum attempts.
- The EMV card does not have a matching AID on the terminal.

The failurereason field will indicate the reason for the fallback.

Possible Cause:

- Faulty chip
- Customer removes the CTLS card away from the terminal before the card read can complete.
- Invalid/incorrect configuration on the terminal.
- Damaged magstripe.

Recommended Action:

- Identify failing transactions and possible reasons of failure.
- Pull the required diagnostic OS Logs from the terminal.
- Train the merchants to handle the fallback effectively.
- Ensure terminal configuration is set correctly.

Event Format:

```
{
        "failureReason": "MaxSwipeError",
        "transactionIdentifier": "000191|370000|||"
}
```

Field Description:

- failureReason - MaxSwipeError or EmptyCandidateList or MaxInsertError or MaxTapError
- transactionIdentifier - Transaction details of current transaction.
    - Transaction Identifiers - Invoice | Transaction Amount (Two decimals implied) | | | Ca in this case since the card was not read successfully when the fallback occurred.

21. Card Not Supported

Fields

Event Name:	CardNotSupported
Priority:	Medium
Description:	The application will send this event whenever there is a local decline of the transaction due to any of the • The card is not part of the BIN table (BIN table lookup failure). • The card is not valid (expired card, LUHN check failure). • The payment type field from the POS does not match the card being used. The data field will contain the detailed reason for the decline.
Possible Cause:	• Incorrect card being used by the cardholder compared the card expected by the POS. • Card configuration is not setup correctly on the terminal, such as missing card BIN configuration. • Usage of NON-ISO card for a transaction expecting ISO card to be used.
Recommended Action:	• Identify failing transactions and possible reasons for failure. • Capture required diagnostics such as terminal logs. • Train the merchants to properly choose the payment type for transactions through the POS. • Ensure the terminal configuration is up to date.
Event Format:	`{ "Data": "Expected:DEBIT Actual:CREDIT", "failureReason": "PaymentTypeMismatch", "transactionIdentifier": "000234|1000|476173******0011|CREDIT| }`

Field Description:	• Data - Additional details causing the failure such as Non-ISO Card, Bin Look Up Failed, actual pa
	• failureReason - PaymentTypeMismatch , UnsupportedCard,
	• Transaction Identifier - Invoice | Transaction Amount (Two decimals implied) | Masked PAN | Pa
	  ◦ Payment Type - CREDIT, DEBIT, EBT, GIFT, FSA, PRIV_LBL
	  ◦ Card Brand - VISA, MASTERCARD, DISCOVER, AMEX, UNIONPAY, JCB, DEB