Notifications

Overview

You can use the Notifications sections in Verifone Central to search and filter your organisation’s webhooks and email notifications.

The Notification Service allows you to get status updates on certain events/actions you subscribe to, and to automate business processes like order management, accounting, or downloading reports. Data sent in the notifications will be pushed to your email/URL.

Availability

Only users with Merchant Admin, Merchant Cashier or Merchant Supervisor roles can access this section.

Events

Events are the topics a notification would be 'listening' to. The notifications will be triggered as soon as there is an event happening for one of the selected topics.

Transaction Events

Event	Description
TxnAccountVerificationApproved TxnAccountVerificationDeclined	An account/card verification
TxnAuthorisationApproved TxnAuthorisationDeclined	An authorisation or preauthorisation
TxnCaptureApproved TxnCaptureDeclined	A capture or completion
TxnDelayedChargeApproved TxnDelayedChargeDeclined	A single message sale (auth+capture) related to a preauth chain
TxnExtendApproved TxnExtendDeclined	A preauth extend
TxnPreauthIncrementApproved TxnPreauthIncrementDeclined	An increment for a preauth
TxnReauthorisationApproved TxnReauthorisationDeclined	A preauth reauth

Event	Description
TxnRefundApproved TxnRefundDeclined	A refund
TxnRefundPreviewCancelled	A refund preview cancelled transaction
TxnRefundPreviewCustomerApproved	A refund preview customer approved transaction
TxnSaleApproved TxnSaleDeclined	A single message sale (auth+capture)
TxnSaleConfirmed	A confirmation of sale (specific for crypto transactions)
TxnVoidApproved TxnVoidDeclined	A void

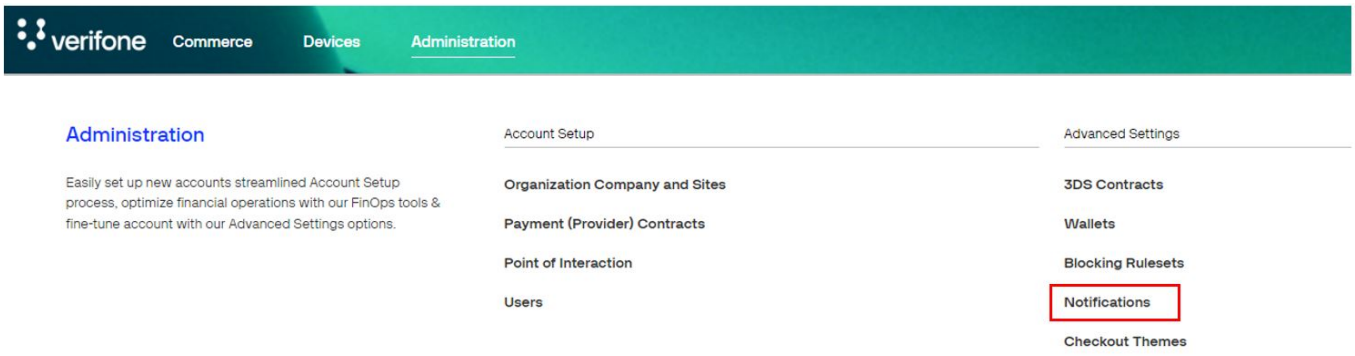
Checkout Events

Event	Description
Checkout - Transaction succeeded	An successful checkout Transaction Success
Checkout - Transaction failed	Checkout transaction failed
Checkout - Card token succeeded	Successful card token transaction
Checkout - Card token failed	Failed card token transaction
Checkout - 3DS authentication succeeded	An successful checkout 3DS transaction
Checkout - 3DS authentication failed	A failed checkout 3DS authentication
Checkout - 3DS lookup failed	3DS lookup failed
Checkout - 3DS lookup succeeded	3DS lookup succeeded
Checkout - SMS delivery succeeded	PBL SMS delivery succeeded
Checkout - Email delivery succeeded	PBL Email delivery succeeded
Checkout - SMS delivery failed	PBL SMS delivery failed
Checkout - Email delivery failed	PBL Email delivery failed

Create notifications in Verifone Central

To use the information in the Notifications section in Verifone Central, follow these steps:

1. Log in to your Verifone Central account.
2. Navigate to **Administration** and click on **Notifications**.



3. The *Notifications* page will be displayed. Click on **Create new notification**.

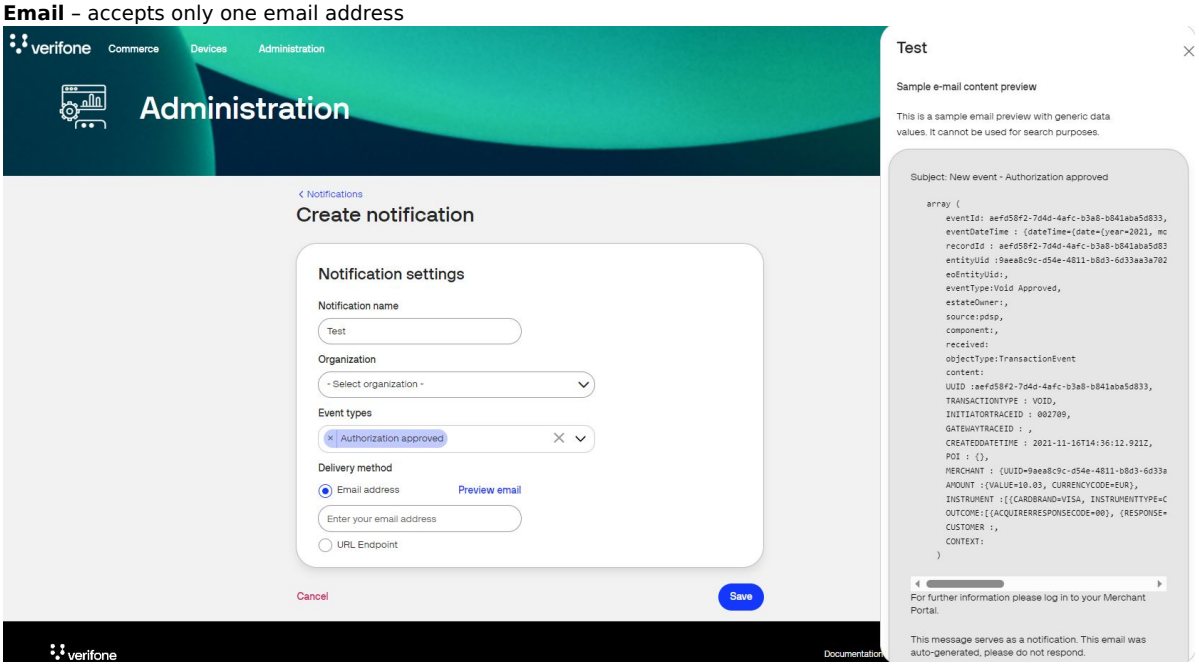
> In the **Notifications** page, you can filter the already created notifications by name / email / URL and/or use the **Event type** or the **Status** filters.

4. On the **Create Notification** page, provide the following information:
   - **Notification name** – used to identify the notification configuration in the listing
   - **Organization(s)**
     - By selecting an organization, you apply a scope to the configuration
     - By selecting the lowest level organization, you will receive messages for the selected organization only

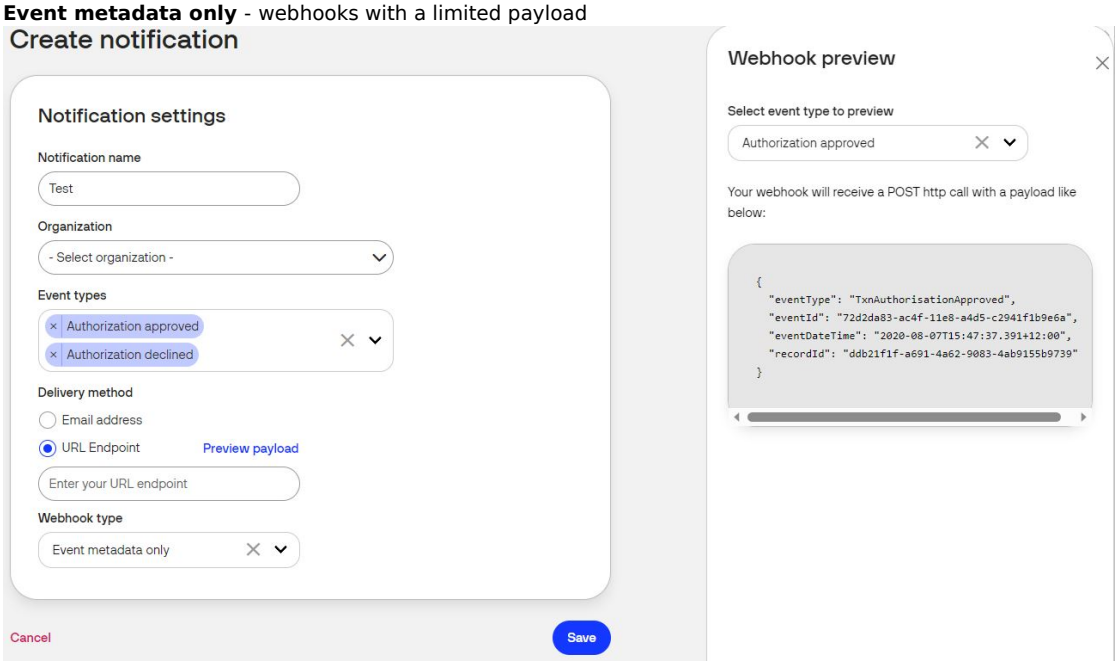
     > Multiple organizations can be selected when creating a notification, if the same notification must be sent to multiple organizations.

   - **Events** – notifications will be triggered as soon as an event takes place for one of the selected transaction status updates (for more details regarding available events check the Events tables above)

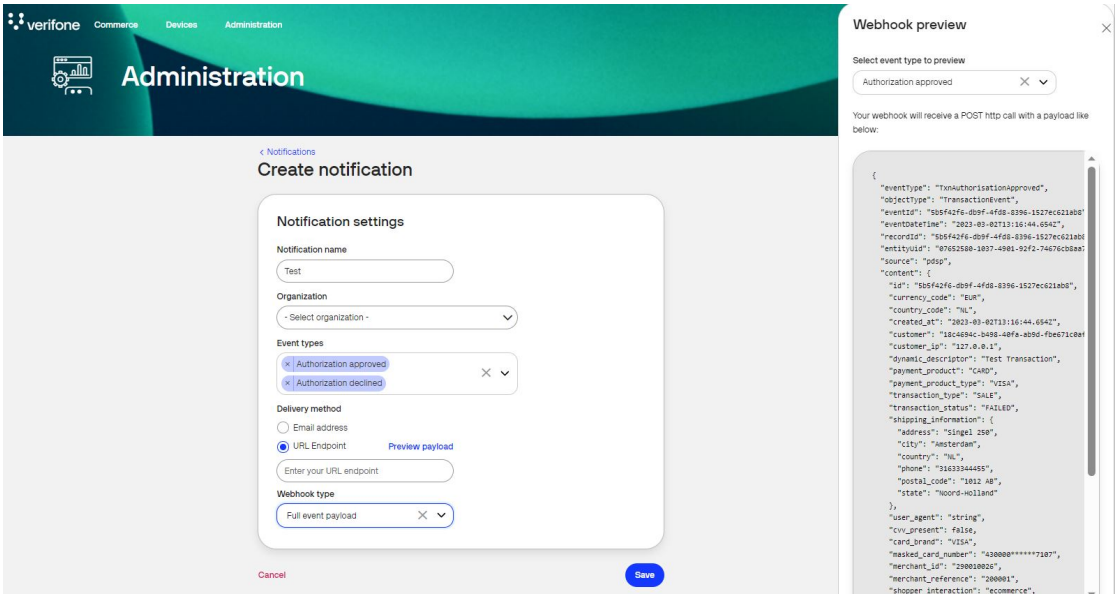
- **Delivery Method** – specifies how notifications will be delivered. Depending on the selected method, Verifone will send either an email with plain text describing the event or a webhook with the transaction payload. Possible values:
  - **Email** – accepts only one email address



  - **URL Endpoint** – accepts only one webhook; selecting this option will provide two other possibilities:
    - **Event metadata only** - webhooks with a limited payload



    - **Full event payload** - webhooks with full content payload (**applicable only for Transaction Events**)



When a new webhook is added, it should be loaded within 60 seconds. However, updates to existing webhooks can take between 10 to 60 minutes depending on server load.

Notification payload variables

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`eventType`	String	—	☐	☐	☐	The event type for which the notification was sent. E.g., `TxnSaleDeclined`
`objectType`	String	—	☐	☐	☐	In a transaction event this would be `TransactionEvent`, and in a Checkout event this would be `StandardEvents`.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`eventId`	String	UUID	☐	☐	☐	In transactions events this would be the Transaction ID, and in Checkout events this would be the Checkout ID.
`itemId`	String	UUID	☐	☐	☐	In transactions events this would be the Transaction ID. The itemId is not applicable to Checkout events.
`recordId`	String	UUID	☐	☐	☐	In transactions events this would be the Transaction ID, and in Checkout events this would be the Checkout ID.
`entityUid`	String	UUID	☐	☐	☐	The Verifone assigned organization identifier.
`eventDateTime`	Datetime	YYYY-MM-DDThh:mm:ss.msZ	☐	☐	☐	The date and time at which the event occurred.
`source`	String	—	☐	☐	☐	The source of the event information.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`content`	Object	—	☐	☐	☐	The associated event data. This is optional and if specified will vary according to the event type.
`content.id`	String	UUID	☐	☐	☐	The Transaction ID.
`content.currency_code`	String	three-letter ISO 4217 alphabetic currency codes	☐	☐	☐	A three-letter alphabetic code that represents the currency used for the transaction.
`content.country_code`	String	2-letter ISO 3166 alpha-2 country code	☐	☐	☐	A 2-letter ISO 3166 alpha-2 country code representing the consumer's address.
`content.created_at`	Datetime	YYYY-MM-DDThh:mm:ss.msZ	☐	☐	☐	The date and time the transaction creation.
`content.customer_ip`	String	32-bit number	☐	☐	☐	A 32-bit number that identifies a host on a TCP/IP network of the consumer.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`content.dynamic_descriptor`	String	—	☐	☐	☐	A short transaction description that can be included when creating a transaction via the Virtual Terminal, the Checkout API, or the eCom API. This description might be included in the bank statement issued to the consumer by some card issuers.
`content.amount`	Float	—	☐	☐	☐	The transaction amount.
`content.payment_product`	String	—	☐	☐	☐	The type of product used for payment. E.g., `CARD`, `KLARA`, `SWISH`, `CRYPTO`, etc.
`content.payment_product_type`	String	—	☐	☐	☐	The brand of the payment type used for payment. E.g., `VISA`, `MASTERCARD`, `AMEX`, etc.
`content.processor_reference`	String	—	☐	☐	☐	Reference identifying the transaction, as provided by the processor.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`content.transaction_type`	String	—	☐	☐	☐	The transaction type, such as `SALE`, `AUTHORIZATION`, `PREAUTH`, etc.
`content.transaction_status`	String	—	☐	☐	☐	The current status of the transaction. E.g., `AUTHORISED`, `CAPTURED`, `SETTLED`, `CANCELLED`, etc.
`content.reason_code`	String	—	☐	☐	☐	A reason code assigned by the acquiring platform; '0000' in case of success.
`content.arn`	String	—	☐	☐	☐	The acquirer reference number (ARN), generated by the acquirer at the time of clearing for card transactions.
`content.authorization_code`	String	—	☐	☐	☐	The credit card authorization code represents the five or six numbers generated by the issuing bank.
`content.shipping_information`	Object	—	☐	☐	☐	An optional object that includes the consumer's shipping information.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`content.shipping_information.address`	String	—	☐	☐	☐	The shipping (street) address.
`content.shipping_information.city`	String	—	☐	☐	☐	The shipping city.
`content.shipping_information.country`	String	A 2-letter ISO 3166 alpha-2 country code	☐	☐	☐	The shipping country.
`content.shipping.phone`	String	—	☐	☐	☐	The shipping phone number.
`content.shipping.postal_code`	String	—	☐	☐	☐	The shipping postal code.
`content.shipping.state`	String	—	☐	☐	☐	The shipping state.
`content.user_agent`	String	—	☐	☐	☐	The full user agent string of the device the customer used to submit the transaction.
`content.cvv_present`	Boolean	—	☐	☐	☐	True if the card was used with a CVV.
`content.rrn`	String	—	☐	☐	☐	The payment processor's retrieval reference number.
`content.shopper_interaction`	String	—	☐	☐	☐	The sales channel that was used to capture the transaction. E.g., `ECOMMERCE`, `MAIL`, `PHONE`, `POS`. etc.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`content.stan`	String	—	☐	☐	☐	A number assigned by a transaction initiator (originator) to assist in identifying a transaction uniquely. This property can be used to store the System Trace Audit Number (STAN) as used in the ISO 8583 and AS2805 specifications.
`content.card_brand`	String	—	☐	☐	☐	Same as the payment product type.
`content.merchant_id`	String	—	☐	☐	☐	The identifier assigned to the merchant entity under the Payment Provider Contract.
`content.merchant_reference`	String	—	☐	☐	☐	A reference specified by the merchant to identify the transaction.
`content.poi_id`	String	—	☐	☐	☐	The Verifone assigned ID to the point of the interaction used for the transaction, where applicable.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`content.masked_card_number`	String	—	☐	☐	☐	Masked number of the card used for payment.
`content.payment_summary.captured_amount`	String	—	☐	☐	☐	The amount that was captured out of the total transaction amount.
`content.threed_authentication`	Object	—	☐	☐	☐	3DS authentication information, where applicable. Read our 3-D Secure article for additional information.
`content.threed_authentication.eci_flag`	String	—	☐	☐	☐	The Electronic Commerce Indicator (ECI).
`content.threed_authentication.enrolled`	Boolean	—	☐	☐	☐	The payment card's 3DS enrollment status.
`content.threed_authentication.cavv`	String	—	☐	☐	☐	The Cardholder Authentication Verification Value (CAVV) cryptographic value.
`content.threed_authentication.pares_status`	String	—	☐	☐	☐	The Payment Authentication Response (PARes) status.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
`content.threed_authentication.ds_transaction_id`	String	—	☐	☐	☐	A unique transaction identifier assigned by the 3DS server.
`content.threed_authentication.threeds_version`	String	—	☐	☐	☐	The 3DS version used to authenticate the transaction.

Email payload sample

```
here is template : RECEIVED:

array (
EVENTID: ,
EVENTDATETIME : ,
ENTITYUID :36559df1-3f8b-492f-adf4-f09313e10c77,
EOENTITYUID:,
EVENTTYPE:Capture Approved,
ESTATEOWNER:,
SOURCE:pdsp,
COMPONENT:,
RECEIVED:
OBJECTTYPE:TransactionEvent
CONTENT:
TRANSACTIONID :5a8a72ae-6257-42c8-a0dd-bfa1d50c7d94,
TRANSACTIONTYPE : CAPTURE,
INITIATORTRACEID : 4187,
GATEWAYTRACEID : ,
CREATEDDATETIME : 2023-04-18T06:36:45.860Z,
POI : {},
MERCHANT : {UUID=36559df1-3f8b-492f-adf4-f09313e10c77, ID=RCTST1000095119, LOCALE={COUNTRYCODE=US},
CONTRACTS=[{MCC=5965, MERCHANTID=RCTST1000095119}]},
AMOUNT :{VALUE=460.06, CURRENCYCODE=USD},
INSTRUMENT :[{CARDBRAND=DINERS, INSTRUMENTTYPE=CARD, MASKEDCARDNUMBER=361859****2226}, {INSTRUMENTTYPE=TOKEN}],
OUTCOME:[{ACQUIRERRESPONSECODE=000}, {RESPONSE=SUCCESS, RESPONSECODE=0000}],
CUSTOMER :,
CONTEXT:{PAYMENTCONTEXT={SALESCHANNEL=ECOMMERCE, ACCOUNTTYPE=CREDIT,
AUTHENTICATIONMETHOD=[SECURE_ELECTRONIC_COMMERCE]}}
)
```

Webhook sample

Event metadata only

```
{
"eventId": "2",
"eventDateTime": "2020-03-23T11:07:28Z",
"recordId": "1",
"eventType": "TxnSaleApproved"
}
```

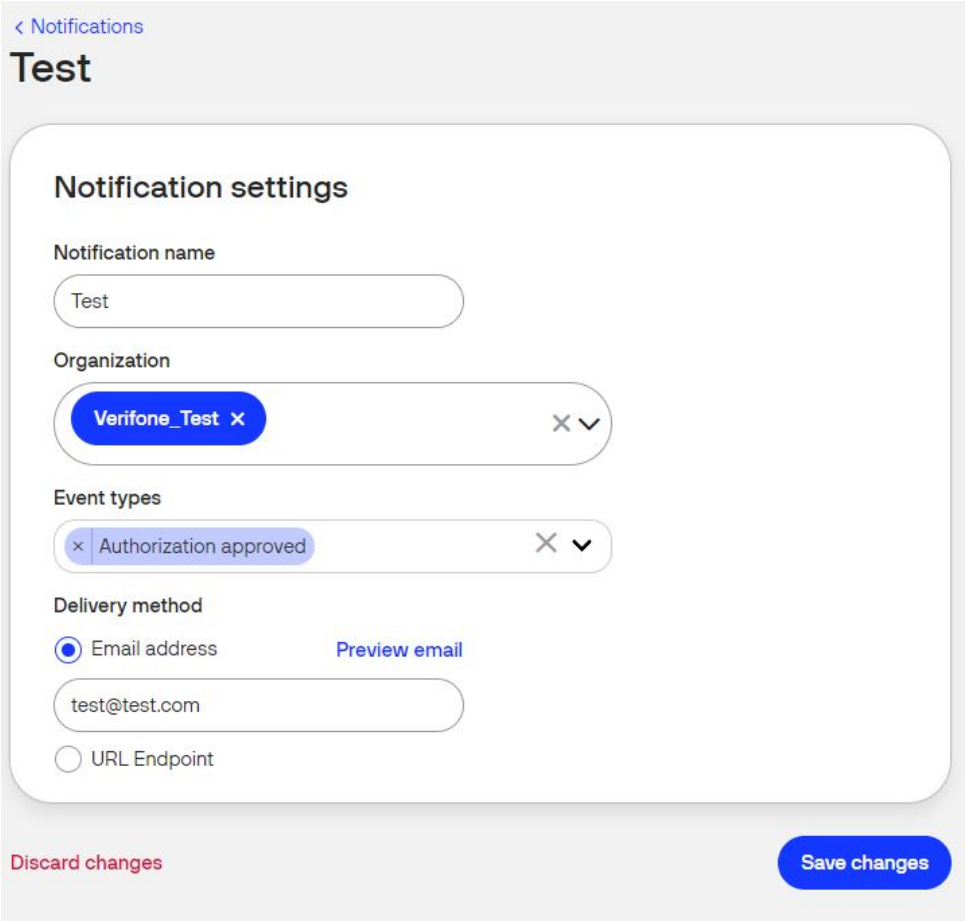

Full event payload


```
Webhook have been called: {
objectType: 'TransactionEvent',
eventId: 'f93fa575-3d2f-4a7f-b182-939c7c6ea610',
eventDateTime: '2023-05-02T12:16:56.077Z',
recordId: 'f93fa575-3d2f-4a7f-b182-939c7c6ea610',
entityUid: '07652580-1037-4901-92f2-74676cb8aa7e',
source: 'pdsp',
eventType: 'TxnAuthorisationApproved',
content: {
id: 'f93fa575-3d2f-4a7f-b182-939c7c6ea610',
currency_code: 'GBP',
created_at: '2023-05-02T12:16:56.077Z',
customer_ip: '127.0.0.1',
dynamic_descriptor: 'TEST AUTOMATION ECOM',
payment_product: 'CARD',
payment_product_type: 'MASTERCARD',
processor_reference: 'BM61NUSABF7:0001030001000206010079000312230502131656000412020000075882',
transaction_type: 'AUTHORISATION',
transaction_status: 'AUTHORISED',
reason_code: '0000',
arn: '020000075882',
authorization_code: '541657',
cvv_present: false,
rrn: '020000075882',
card_brand: 'MASTERCARD',
masked_card_number: '548016******7897',
merchant_id: '290010026',
merchant_reference: '5678',
shopper_interaction: 'ecommerce',
stan: '043026',
threed_authentication: {
eci_flag: '05',
enrolled: 'Y',
cavv: 'MTIzNDU2Nzg5MDEyMzQ1Njc4OTA=',
pares_status: 'Y',
ds_transaction_id: 'ea5e8cb3-69c1-4229-b484-2def1be9952c',
threeds_version: '2.1.0'
},
amount: '11.29',
payment_summary: {}
}
}
```


Update notifications in Verifone Central

Any notification can be edited as desired. The below updates can be applied to a notification:

- Update the notification name
- Add/remove organisation(s)



A notification can be deleted only after being disabled.

Notification failures

Notification failures can occur only for webhook events, when the system fails to notify the merchant (e.g. URL endpoint is not reachable or service is down/time out).

The failed notifications can be viewed in the **Administration** > **Notifications** > select the notification > **Notification Failures** section from Verifone Central.

Webhook signature verification

The authenticity and integrity of the webhook event can be verified by checking the signature provided in the header: x-vfi-jws. This signature is used to validate that Verifone is the sender and that the message has not been tampered with.

This signature is provided as JSON web signature (JWS) using the webhook body as the unencoded payload as described in https://www.rfc-editor.org/rfc/rfc7797.

The public keys used to verify the signatures are provided in a JWKS (JSON web key set) file which can be downloaded from the following URLs.

Environment
Test
Production

To verify the signature of webhook payload the following steps need to be performed.

1. Ensure that the application has loaded the keys from the JWKS file. This file should be cached locally and not downloaded from verifone on each request.
2. When the request is received, convert the http json body to canonicalized form: https://www.rfc-editor.org/rfc/rfc8785.
3. Select the correct key from the JWKS matching the key id from the x-vfi-jws header.
4. Use the x-vfi-jws header and canonicalized body to verify the signature with the selected key.

> If the key id is not found in Step 3 then refresh the JWKS file as a new key may have been added.

Webhook signature verification sample

The following java code verifies a signature as described above and has dependencies on the following libraries:

- https://github.com/erdtman/java-json-canonicalization
- https://bitbucket.org/b_c/jose4j/src/master/

```
/**
* Step 1 - Load signing keys via jwks file
*/

String JWKS_URL = "https://vf11gtostorage1.blob.core.windows.net/test-webhook-sign-keys/test-webhook-sign-
keys.jwks";

// on first startup we need to download the signing keys
// NOTE: this file should be cached locally and not downloaded each time
Path jwksLocalPath = Path.of("./test-webhook-sign-keys.jwks");
if (!Files.exists(jwksLocalPath)) {
try (InputStream in = new URL(JWKS_URL).openStream()) {
Files.copy(in, jwksLocalPath, StandardCopyOption.REPLACE_EXISTING);
}
}
```

```
// load keys from file
String jkws = Files.readString(jwksLocalPath);
JsonWebKeySet jsonWebKeySet = new JsonWebKeySet(jkws);

/**
* Step 2 - Convert webhook json body to canonicalized form
*/

String originalJsonBody = *JSON BODY FROM WEBHOOK*;
JsonCanonicalizer jsonCanonicalizer = new JsonCanonicalizer(originalJsonBody);
String canonicalizedJson = jsonCanonicalizer.getEncodedString();

/**
* Step 3 and 4- Select matching key for x-vfi-jws and validate signature
*/

JsonWebSignature verifierJws = new JsonWebSignature();

// set contents from x-vfi-jws header
verifierJws.setCompactSerialization(detachedContentJws);

// The canonicalized content is the payload
verifierJws.setPayload(canonicalizedJson);

// Pick the key to use for checking the signature. The key selected
// should have the same key id "kid" as the x-vfi-jws header
VerificationJwkSelector jwkSelector = new VerificationJwkSelector();
JsonWebKey jwkSelected = jwkSelector.select(verifierJws, jsonWebKeySet.getJsonWebKeys());

if (jwkSelected == null) {
// if key can't be found then refresh signing keys as
// a new key id may have been added
try (InputStream in = new URL(JWKS_URL).openStream()) {
Files.copy(in, jwksLocalPath, StandardCopyOption.REPLACE_EXISTING);
}
jkws = Files.readString(jwksLocalPath);
jsonWebKeySet = new JsonWebKeySet(jkws);
jwkSelected = jwkSelector.select(verifierJws, jsonWebKeySet.getJsonWebKeys());
}

// set key based on keyid selected from jwks
verifierJws.setKey(jwkSelected.getKey());

// Check the signature
boolean signatureVerified = verifierJws.verifySignature();
```