

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

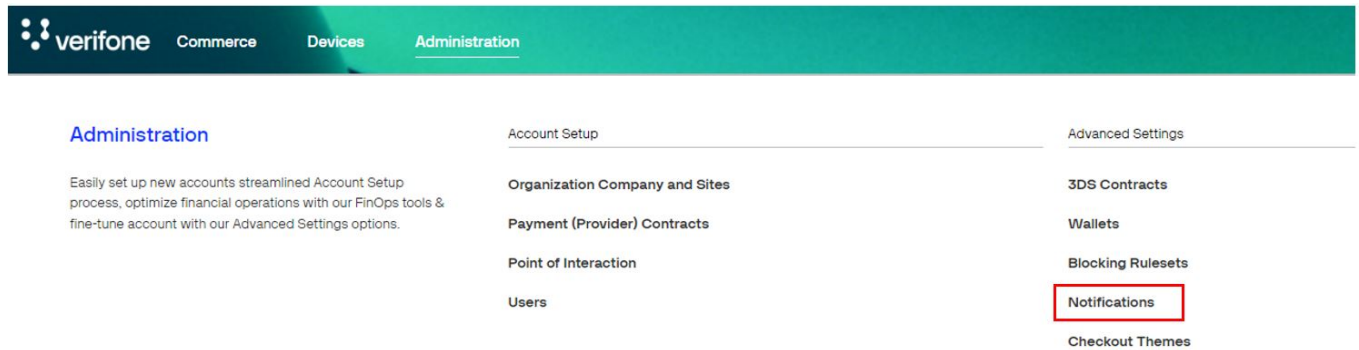
Checkout Events

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**.

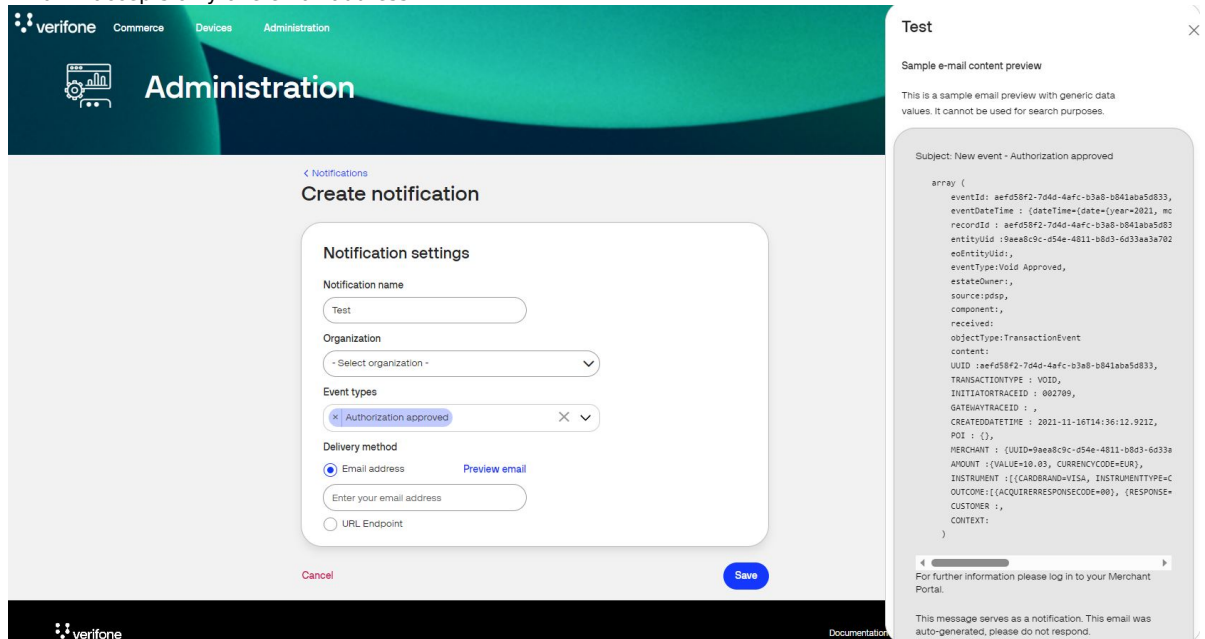
In the **Notifications** page, you can filter the already created notifications by name / email / URL and/or use the **Event type** or the **Status** filters.



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 a root-level organization, the notifications will send messages for all child organizations
 - 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 has to be sent for 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



The screenshot shows the Verifone Administration interface. The main section is titled 'Create notification' and contains the following fields:

- Notification name:** Test
- Organization:** - Select organization -
- Event types:** Authorization approved
- Delivery method:**
 - Email address [Preview email](#)
 - URL Endpoint

Buttons for 'Cancel' and 'Save' are visible at the bottom of the form.

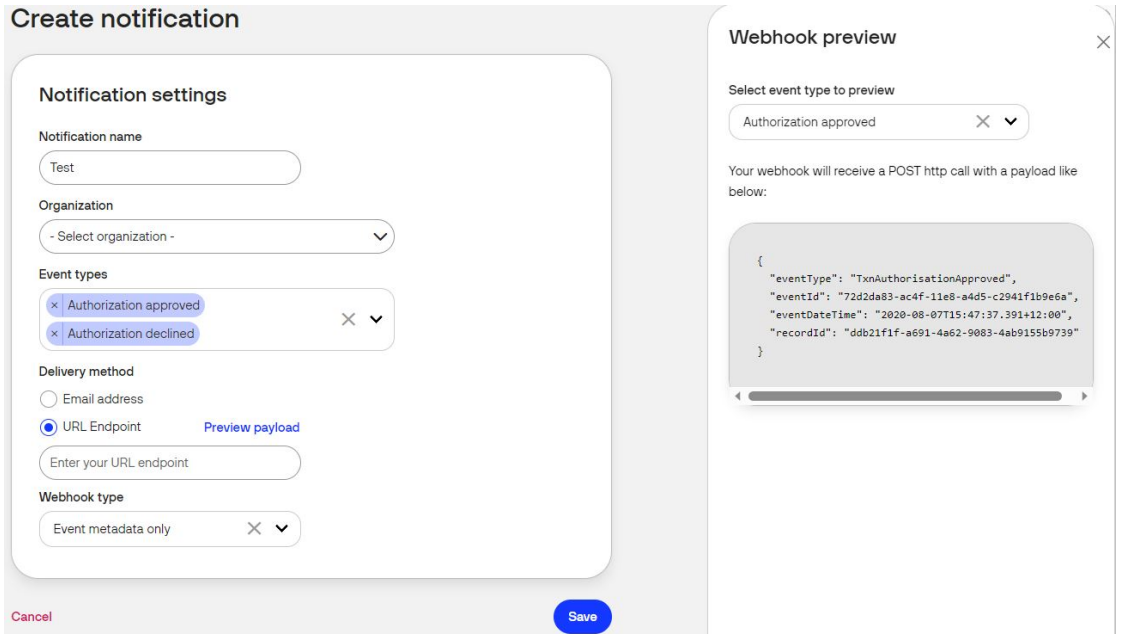
To the right, a 'Test' window shows a sample e-mail content preview. The subject is 'New event - Authorization approved'. The body contains a JSON payload:

```

array (
  eventId: aeF58F2-764d-4afc-b3a8-b841aba50833,
  eventDateTime: {dateTime:(date+year=2021, mc
  recordId : aeF58F2-764d-4afc-b3a8-b841aba5083
  entityUuid : 9aea8c9c-d54e-4811-b803-6d33aa3a702
  eoEntityUuid,
  eventType:Void Approved,
  estateOwner:,
  source:pdsp,
  component:,
  received:,
  objectType:TransactionEvent
  content:
  UUID : aeF58F2-764d-4afc-b3a8-b841aba50833,
  TRANSACTIONTYPE : VOID,
  INITIATORTRACEID : 002709,
  GATEWAYTRACEID : ,
  CREATEDDATEETIME : 2021-11-16T14:36:12.921Z,
  POS : {},
  MERCHANT : {UUID=9aea8c9c-d54e-4811-b8d3-6d33a
  AMOUNT : {VALUE=10.03, CURRENCYCODE=EUR},
  INSTRUMENT : {CARDBRAND=VISA, INSTRUMENTTYPE=C
  OUTCOME : {ACQUIRERRESPONSECODE=00}, {RESPONSE=
  CUSTOMER : ,
  CONTEXT :
)
  
```

Below the JSON, there is a message: 'For further information please log in to your Merchant Portal. This message serves as a notification. This email was auto-generated, please do not respond.'

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



Create notification

Notification settings

Notification name: Test

Organization: - Select organization -

Event types:

- Authorization approved
- Authorization declined

Delivery method:

- Email address
- URL Endpoint [Preview payload](#)

 Enter your URL endpoint:

Webhook type: Event metadata only

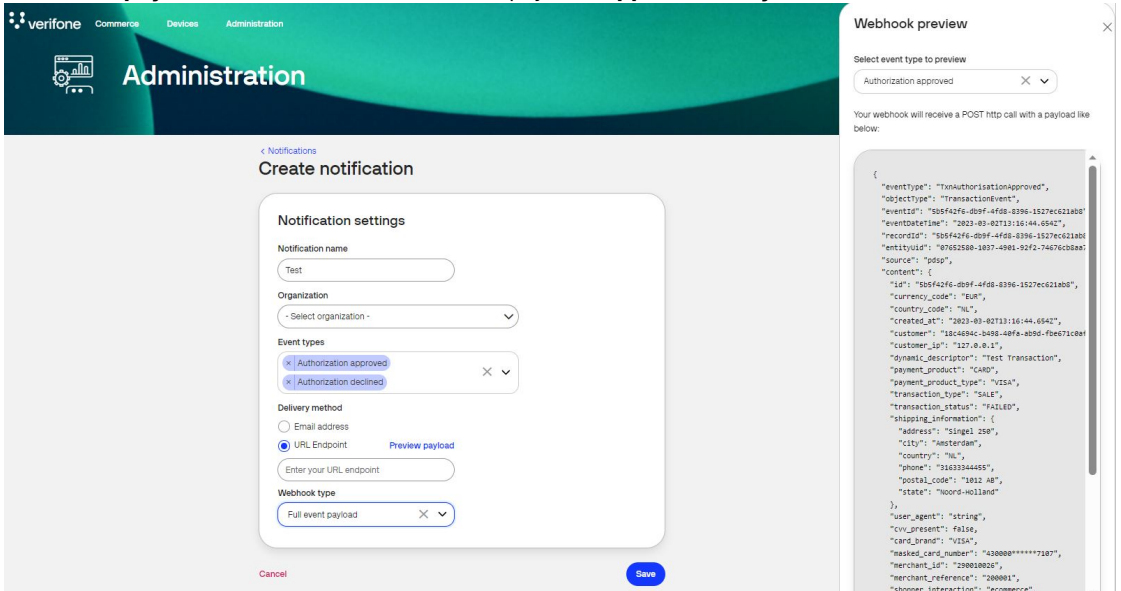
Webhook preview

Select event type to preview: Authorization approved

Your webhook will receive a POST http call with a payload like below:

```
{
  "eventType": "TxnAuthorisationApproved",
  "eventId": "72d2da83-ac4f-11e8-a4d5-c2941f1b9e6a",
  "eventDateTime": "2020-08-07T15:47:37.391+12:00",
  "recordId": "ddb21f1f-a691-4a62-9883-4ab9155b9739"
}
```

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



Administration

Create notification

Notification settings

Notification name: Test

Organization: - Select organization -

Event types:

- Authorization approved
- Authorization declined

Delivery method:

- Email address
- URL Endpoint [Preview payload](#)

 Enter your URL endpoint:

Webhook type: Full event payload

Webhook preview

Select event type to preview: Authorization approved

Your webhook will receive a POST http call with a payload like below:

```
{
  "eventType": "TxnAuthorisationApproved",
  "objectType": "TransactionEvent",
  "eventId": "5b542f6-db9f-4f6b-8396-1527ec621ab0",
  "eventDateTime": "2023-03-02T13:16:44.654Z",
  "recordId": "5b542f6-db9f-4f6b-8396-1527ec621ab0",
  "entityId": "67652588-1837-4981-92f2-74676cb0aa",
  "source": "pdp",
  "content": {
    "id": "5b542f6-db9f-4f6b-8396-1527ec621ab0",
    "currency_code": "EUR",
    "country_code": "NL",
    "created_at": "2023-03-02T13:16:44.654Z",
    "customer": "18c4694c-b498-48fa-ab90-fbe671c0bf",
    "customer_ip": "127.0.0.1",
    "dynamic_descriptor": "test transaction",
    "payment_product": "CARD",
    "payment_product_type": "VISA",
    "transaction_type": "SALE",
    "transaction_status": "FAILED",
    "shipping_information": {
      "address": "Singel 250",
      "city": "Amsterdam",
      "country": "NL",
      "name": "3103304455",
      "postal_code": "1012 AB",
      "state": "noord-holland"
    }
  },
  "user_agent": "string",
  "civ_present": false,
  "card_brand": "VISA",
  "masked_card_number": "430000****7187",
  "merchant_id": "298918026",
  "merchant_reference": "280001",
  "shopper_interaction": "ecommerce"
}
```

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
<code>eventType</code>	String	—	☐	☐	☐	The event type for which the notification was sent. E.g., <code>TxnSaleDeclined</code>
<code>objectType</code>	String	—	☐	☐	☐	In a transaction event this would be <code>TransactionEvent</code> , and in a Checkout event this would be <code>StandardEvents</code> .
<code>eventId</code>	String	UUID	☐	☐	☐	In transactions events this would be the Transaction ID, and in Checkout events this would be the Checkout ID.
<code>itemId</code>	String	UUID	☐	☐	☐	In transactions events this would be the Transaction ID. The <code>itemId</code> is not applicable to Checkout events.
<code>recordId</code>	String	UUID	☐	☐	☐	In transactions events this would be the Transaction ID, and in Checkout events this would be the Checkout ID.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
<code>entityUid</code>	String	UUID	☐	☐	☐	The Verifone assigned organization identifier.
<code>eventDateTime</code>	Datetime	YYYY-MM-DDThh:mm:ss.msZ	☐	☐	☐	The date and time at which the event occurred.
<code>source</code>	String	—	☐	☐	☐	The source of the event information.
<code>content</code>	Object	—	☐	☐	☐	The associated event data. This is optional and if specified will vary according to the event type.
<code>content.id</code>	String	UUID	☐	☐	☐	The Transaction ID.
<code>content.currency_code</code>	String	three-letter ISO 4217 alphabetic currency codes	☐	☐	☐	A three-letter alphabetic code that represents the currency used for the transaction.
<code>content.country_code</code>	String	2-letter ISO 3166 alpha-2 country code	☐	☐	☐	A 2-letter ISO 3166 alpha-2 country code representing the consumer's address.
<code>content.created_at</code>	Datetime	YYYY-MM-DDThh:mm:ss.msZ	☐	☐	☐	The date and time the transaction creation.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
<code>content.customer_ip</code>	String	32-bit number	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	A 32-bit number that identifies a host on a TCP/IP network of the consumer.
<code>content.dynamic_descriptor</code>	String	—	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	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.
<code>content.amount</code>	Float	—	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	The transaction amount.
<code>content.payment_product</code>	String	—	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	The type of product used for payment. E.g., <code>CARD</code> , <code>KLARA</code> , <code>SWISH</code> , <code>CRYPTO</code> , etc.
<code>content.payment_product_type</code>	String	—	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	The brand of the payment type used for payment. E.g., <code>VISA</code> , <code>MASTERCARD</code> , <code>AMEX</code> , etc.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
<code>content.processor_reference</code>	String	—	☐	☐	☐	Reference identifying the transaction, as provided by the processor.
<code>content.transaction_type</code>	String	—	☐	☐	☐	The transaction type, such as <code>SALE</code> , <code>AUTHORIZATION</code> , <code>PREAUTH</code> , etc.
<code>content.transaction_status</code>	String	—	☐	☐	☐	The current status of the transaction. E.g., <code>AUTHORISED</code> , <code>CAPTURED</code> , <code>SETTLED</code> , <code>CANCELLED</code> , etc.
<code>content.reason_code</code>	String	—	☐	☐	☐	A reason code assigned by the acquiring platform; '0000' in case of success.
<code>content.arn</code>	String	—	☐	☐	☐	The acquirer reference number (ARN), generated by the acquirer at the time of clearing for card transactions.
<code>content.authorization_code</code>	String	—	☐	☐	☐	The credit card authorization code represents the five or six numbers generated by the issuing bank.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
<code>content.shipping_information</code>	Object	—	☐	☐	☐	An optional object that includes the consumer's shipping information.
<code>content.shipping_information.address</code>	String	—	☐	☐	☐	The shipping (street) address.
<code>content.shipping_information.city</code>	String	—	☐	☐	☐	The shipping city.
<code>content.shipping_information.country</code>	String	A 2-letter ISO 3166 alpha-2 country code	☐	☐	☐	The shipping country.
<code>content.shipping.phone</code>	String	—	☐	☐	☐	The shipping phone number.
<code>content.shipping.postal_code</code>	String	—	☐	☐	☐	The shipping postal code.
<code>content.shipping.state</code>	String	—	☐	☐	☐	The shipping state.
<code>content.user_agent</code>	String	—	☐	☐	☐	The full user agent string of the device the customer used to submit the transaction.
<code>content.cvv_present</code>	Boolean	—	☐	☐	☐	True if the card was used with a CVV.
<code>content.rrn</code>	String	—	☐	☐	☐	The payment processor's retrieval reference number.

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
<code>content.shopper_interaction</code>	String	—	☐	☐	☐	The sales channel that was used to capture the transaction. E.g., <code>ECOMMERCE</code> , <code>MAIL</code> , <code>PHONE</code> , <code>POS</code> . etc.
<code>content.stan</code>	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.
<code>content.card_brand</code>	String	—	☐	☐	☐	Same as the payment product type.
<code>content.merchant_id</code>	String	—	☐	☐	☐	The identifier assigned to the merchant entity under the Payment Provider Contract.
<code>content.merchant_reference</code>	String	—	☐	☐	☐	A reference specified by the merchant to identify the transaction.

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

Parameter Name	Type	Format	Email	Full payload	Metadata	Description
content.threed_authentication.pares_status	String	—	☐	☐	☐	The Payment Authentication Response (PAREs) status.
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:

<https://verifone.cloud/docs/portal/administration/notifications>

Updated: 13-Jun-2024

[< Notifications](#)

Test

Notification settings

Notification name

Organization
Verifone_Test ✕ ▾

Event types
✕ Authorization approved ✕ ▾

Delivery method
 Email address [Preview email](#)

 URL Endpoint

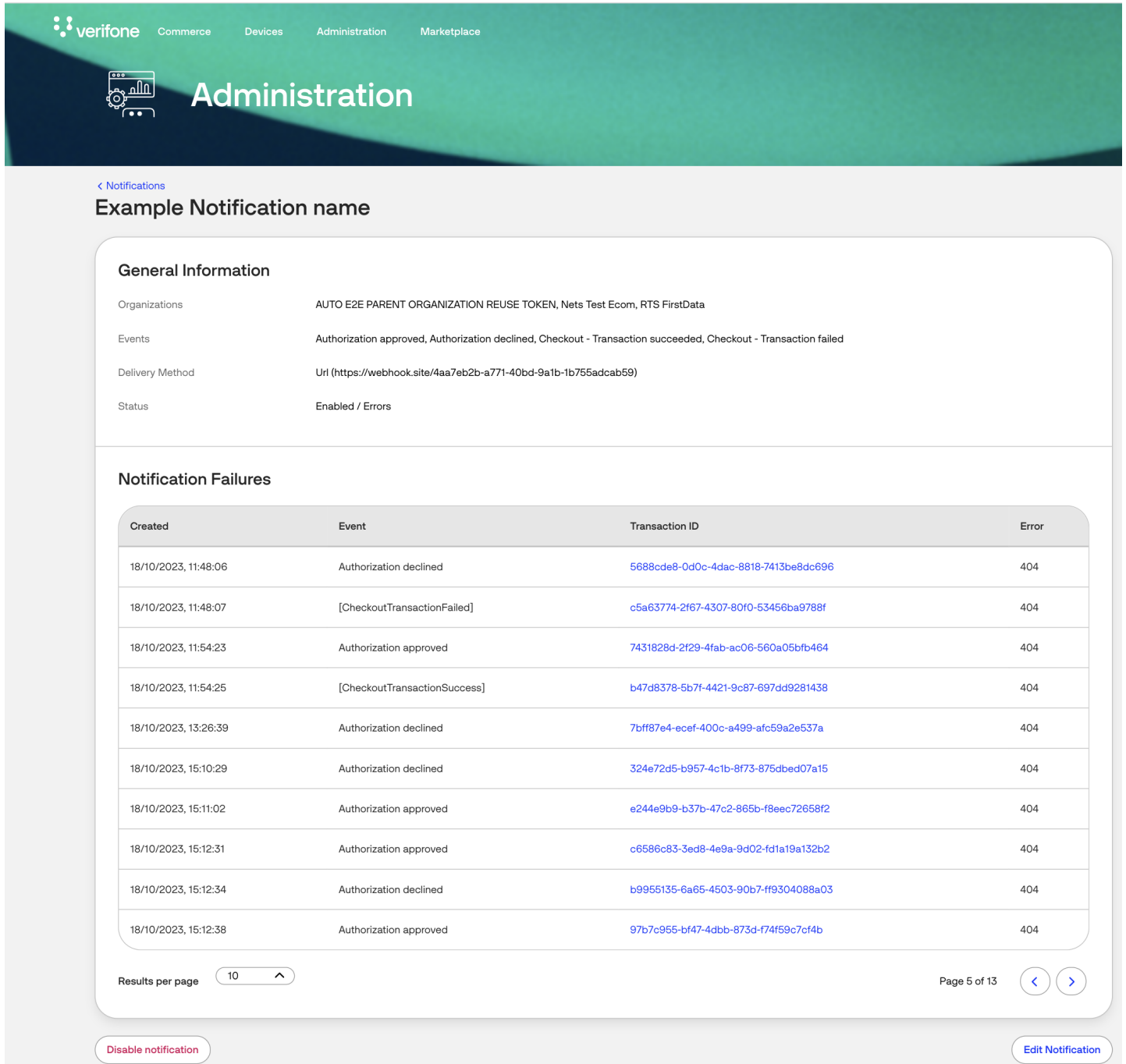
[Discard changes](#) [Save changes](#)

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.



The screenshot shows the 'Administration' section of the Verifone Central interface. The breadcrumb trail is 'Notifications > Example Notification name'. The notification details are as follows:

- General Information**
 - Organizations: AUTO E2E PARENT ORGANIZATION REUSE TOKEN, Nets Test Ecom, RTS FirstData
 - Events: Authorization approved, Authorization declined, Checkout - Transaction succeeded, Checkout - Transaction failed
 - Delivery Method: Url (https://webhook.site/4aa7eb2b-a771-40bd-9a1b-1b755adcab59)
 - Status: Enabled / Errors
- Notification Failures**

Created	Event	Transaction ID	Error
18/10/2023, 11:48:06	Authorization declined	5688cde8-0d0c-4dac-8818-7413be8dc696	404
18/10/2023, 11:48:07	[CheckoutTransactionFailed]	c5a63774-2f67-4307-80f0-53456ba9788f	404
18/10/2023, 11:54:23	Authorization approved	7431828d-2f29-4fab-ac06-560a05bfb464	404
18/10/2023, 11:54:25	[CheckoutTransactionSuccess]	b47d8378-5b71-4421-9c87-697dd9281438	404
18/10/2023, 13:26:39	Authorization declined	7bff87e4-ecf4-400c-a499-afc59a2e537a	404
18/10/2023, 15:10:29	Authorization declined	324e72d5-b957-4c1b-8f73-875dbed07a15	404
18/10/2023, 15:11:02	Authorization approved	e244e9b9-b37b-47c2-865b-f8eec72658f2	404
18/10/2023, 15:12:31	Authorization approved	c6586c83-3ed8-4e9a-9d02-fd1a19a132b2	404
18/10/2023, 15:12:34	Authorization declined	b9955135-6a65-4503-90b7-ff9304088a03	404
18/10/2023, 15:12:38	Authorization approved	97b7c955-bf47-4d8b-873d-f74f59c7cf4b	404

At the bottom of the failures list, there is a 'Results per page' dropdown set to 10, a 'Page 5 of 13' indicator, and navigation arrows. Below the main content area, there are two buttons: 'Disable notification' and 'Edit Notification'.

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://vfl1gtostorage1.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();
```