

Payments API Document (JUL 2022)

Contents

Billing Details.....	3
Donor Type.....	3
Payment Object	3
Payment Status	4
Payment Type	4
Payment SubType	4
Payment Items.....	4
Product Types	5
Transaction Fee Product Type	5
Parent Payment Item Correlation.....	6
Offline Payments.....	7
Payments Create.....	7
Example - Minimum Required Fields for request:.....	7
Payment Update	8
Payment Item Add / Update.....	8
Live Payments	10
Payments Create.....	10
Dynamic Drop In Payment UI.....	10
Usage.....	10
Setup	10
Callbacks.....	11
Other Settings	11
Testing.....	12
GoFundraise API.....	12
Example.....	14
Minimum Required Fields for request:.....	14
Synchronous / Asynchronous Workflow.....	14
Example.....	14
Minimum Required Fields for request:.....	14
Synchronous Payments.....	15
Asynchronous Payments.....	17
Payment Status	17

Example.....	17
The Status Object.....	17
Payments Retrieval	18
Entire Payment Object.....	18
Example.....	18
Individual Payment Item Object	19
Example.....	19
Response Objects.....	19
Currency.....	19
Billing.....	19
Payment	20
PaymentItem.....	20
Beneficiary	20
Event	21
Page.....	21

Billing Details

All payments created require the `FirstName`, `LastName`, and `Email` to be completed on the billing details. All other fields are optional but can be recommended in certain circumstances.

```
{
  "BillingDetails": {
    "Organisation": "string",
    "DonorType": "string",
    "UserId": Integer,
    "Title": "string",
    "FirstName": "string",
    "LastName": "string",
    "Email": "string",
    "PhoneNumber": "string",
    "MobileNumber": "string",
    "StreetAddress": "string",
    "Suburb": "string",
    "PostCode": "string",
    "State": "string",
    "Country": "string"
  },
  ...
}
```

Donor Type

The donor type (`DonorType`) determines whether the transaction was made by an individual or organisation. This can help when determining whether to display / receipt the organisation field. When providing an organisation name it is recommended to define the `DonorType` to clearly define if the payment was made on behalf of the organisation or that the user was only associated with the organisation.

Options: null/blank, I – Individual, O - Organisation

Payment Object

The payment object includes all data related to the payment method and items within the payment. The total charged against a payment method is calculated using the Payment items (`PaymentItems`) included in the payment object.

```
{
  ...
  "Payment": {
    "PaymentType": "string",
    "PaymentSubType": "string",
    "PaymentCustomerReference": "string",
    "PaymentNotes": "string",
    "PaymentStatus": "string",
    "ParentPaymentId": Integer,
    "EnableCommunications": false,
    "PaymentToken": "string",
    "AdditionalData": "string",
    "PaymentItems": [
      ...
    ]
  }
}
```

```
}  
}
```

Payment Status

A payment can have the following statuses

- N – Not complete. This transaction is still processing
- A – Active / Successful.
- F – Failed
- R – Refunded. This entire transaction was refunded prior to being remitted to any beneficiaries.

Payment Type

Please note: Additional Payment Types may be added at any time. If using this field, please make sure that you can handle unknown types. Current Payment Types Include:

- CC – Credit Card
- PP – PayPal
- WPG - Workplace Giving
- CASH – Cash / Other
- CHQ – Cheque

Payment SubType

Please note: Additional Payment Sub Types may be added at any time. If using this field, please make sure that you can handle unknown types. Current Payment Sub Types Include:

- Offline – Offline / Other Source
- PayPal - PayPal
- MX – American Express
- Visa - Visa
- Master - Mastercard
- G2G – Good2Give (restricted)

Payment Items

The payment items include all the details about the individual line items that are associated with a payment.

```
{  
  ...  
  "Payment": {  
    ...  
    "PaymentItems": [  
      {  
        "ProductType": 0,  
        "ProductDescription": "string",  
        "Quantity": 0,  
        "UnitPrice": 0,  
        "UnitSplit": [...],  
        "BeneficiaryAccountId": 0,  
        "Classification": "string",  
        "PaymentItemNotes": "string",  
      }  
    ]  
  }  
}
```

```

        "EventCampaignId": 0,
        "FundraisingPageId": 0,
        "FundraisingMessage": "string",
        "Anonymous": false,
        "DisplayOnPage": true,
        "IncludeInTotal": true,
        "ParentPaymentItemId": 0,
        "ParentPaymentItemCorrelation": ,
        "PaymentItemCustomerReference": "string"
    }
]
}
}

```

Product Types

Currently the following product types are available to choose from

- '1' = Tax Deductible Donation
- '2' = Non Tax Deductible Donation
- '3' = Deposit Funds
- '4' = Platform Gratuity
- '5' = Registration
- '6' = Ticket
- '7' = Other
- '8' = Shipping
- '9' = Adjustment
- '10' = Workplace Giving
- '11' = Transaction Fee

Transaction Fee Product Type

The user pays transaction fee product type is very specific to trigger the transaction fees to not be applied across the other items within a payment. We search for the existence of product type 11 within a payment. If this exists, other payment items are not processed for the transaction fees. Please note, transaction fees are calculated by the merchant on the total payment amount, so when calculating the transaction fees a user is required to pay, please ensure you include this additional amount in the cost calculation, otherwise there will be a slight discrepancy between the user charged amount and the actual charged amount.

Subtotal = T
 Transaction Fee = F
 Transaction Rate = R
 User Charge = C

$$C = ((R*T)+F)/(1-R)$$

E.g.

T = 16.50

F = 0.10

R = 0.011

$$C = ((0.011*16.50)+0.1)/(1-0.011)$$
$$= 0.28$$

Please use the Your OWN Beneficiary for this product. The below is an example only.

```
{  
  "EventCampaignId": 10014,  
  "BeneficiaryAccountId": 104,  
  "ProductType": 11,  
  "ProductDescription": "User Pays Transaction Fee",  
  "UnitPrice": 0.28,  
  "Quantity": 1  
}
```

Parent Payment Item Correlation

When linking a payment item with another payment item using the `ParentPaymentItemId` field it is recommended to provide the context of that relationship so that the platform can display it correctly. Option available are:

'Refund'

'ChargebackRefund'

- Used when adding a new item which refunds a previous item so that the original item is negated and no longer displayed on the platform.

'EmployerMatching'

'GeneralMatching'

- Used when a matching donation is made against the original donation, this will display as a secondary line under the original donation and boost the original donations total displayed.

'Referral'

- Used to identify when a donation has been made after the sharing of the initial donation

'Upsell'

- Used to identify when a user has increased their initial donation

'Recurring'

- Used when a user has made donations on a recurring basis to increase their initial donation over time

Offline Payments

The ability to add Offline Payments via the API is restricted and you must contact GoFundraise support to enable the addition of Offline payments on a per event basis.

Payments Create

This endpoint requires Authorization via a Bearer Token

Url: <https://api.gofundraise.com/v1/payments/payment>

Method: POST

Offline payments are restricted to the Payment Types: WPG, CASH, CHQ and Payment Sub Types: Offline

Offline Payments Support the full Billing Details, however, only support partial Payment / Payment Item objects.

Billing Details require `FirstName`, `LastName`, and `Email`.

Payment Objects support only the following options:

- `PaymentType` - Required
- `PaymentSubType` - Required
- `PaymentCustomerReference`
- `PaymentNotes`
- `PaymentStatus` - Required
- `ParentPaymentId`
- `EnableCommunications`

Payment Items support only the following options:

- `ProductType` - Required
- `ProductDescription` - Required
- `Quantity` - Required
- `UnitPrice` - Required
- `BeneficiaryAccountId` - Required unless `FundraisingPageId` provided
- `Classification`
- `PaymentItemNotes`
- `EventCampaignId` - Required unless `FundraisingPageId` provided
- `FundraisingPageId`
- `FundraisingMessage`
- `Anonymous`
- `DisplayOnPage`
- `IncludeInTotal`
- `ParentPaymentItemId`
- `ParentPaymentItemCorrelation`
- `PaymentItemCustomerReference`

Example - Minimum Required Fields for request:

```
{
  "BillingDetails": {
    "FirstName": "Test First",
    "LastName": "Test Last",
    "Email": "test@email.com",
  },
  "Payment": {
    "PaymentType": "CASH",
```

```

    "PaymentSubType": "Offline",
    "PaymentStatus": "A",
    "PaymentItems": [
      {
        "EventCampaignId": 12345,
        "BeneficiaryAccountId": 104,
        "ProductType": 1,
        "ProductDescription": "Donation",
        "UnitPrice": 10,
        "Quantity": 1
      }
    ]
  }
}

```

Payment Update

Offline Payments can be altered after the initial creation as they are marked as non financial payments. Live payments cannot be altered after creation.

This endpoint requires Authorization via a Bearer Token

Url: <https://api.gofundraise.com/v1/payments/payment/{paymentid}>

Method: PUT

Any Billing or Payment fields that are passed through on this request will be updated. You cannot update payment items from this endpoint.

Example usage is to convert a pledge donations from status 'N' to status 'A' when funds are received for the pledge.

Payment Item Add / Update

Offline payments can have payment items added or altered after the initial creation as they are marked as non-financial payments. Live payments cannot have additional payment items added or updated.

This endpoint requires Authorization via a Bearer Token

Url: <https://api.gofundraise.com/v1/payments/payment/{paymentid}>

Method: POST

This endpoint can receive an array of payment items as per normal offline payment abilities.

Example

```

{
  "PaymentItems": [
    {
      "EventCampaignId": 12345,
      "BeneficiaryAccountId": 104,
      "ProductType": 1,
      "ProductDescription": "Donation",
      "UnitPrice": 10,
      "Quantity": 1
    }
  ]
}

```



```
]
}
```

This endpoint requires Authorization via a Bearer Token

Url: <https://api.gofundraise.com/v1/payments/paymentitem/{paymentitemid}>

Method: PUT

This endpoint can update a specific payment item as for normal offline payment abilities.

Example:

```
{
  "UnitPrice": 15,
  "Quantity": 2
}
```

Live Payments

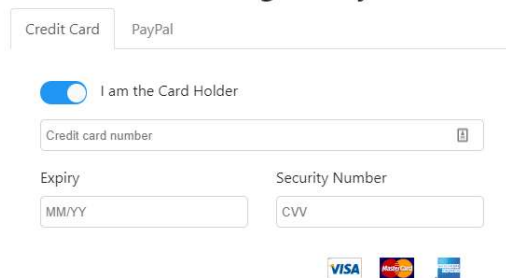
The ability to add Live Payments via the API is **restricted** and you must contact GoFundraise support to enable the addition of Offline payments on a per event basis.

Payments Create

Dynamic Drop In Payment UI

The Dynamic drop in UI has been extended to allow for both Credit Card and PayPal options to be completed through the front end. All updates from our side will be automatically applied to your platform.

Auto loaded from gateway demo



Fees:

```
[{"merchant":"Master","transactionRate":0.011,"transactionFee":0.2},
{"merchant":"MX","transactionRate":0.0198,"transactionFee":0.2},
{"merchant":"Visa","transactionRate":0.011,"transactionFee":0.2},
{"merchant":"PayPal","transactionRate":0.021,"transactionFee":0.2}]
```

An Example of the New Dynamic Drop in UI that should be utilized during this project can be found online here: <https://cdn.gofundraise.com.au/all/js/test/payment-module/AutoLoadDemo.html>

Usage

There are 2 Main areas that need to be addressed when implementing the new Dynamic Drop In UI: Setup and Callback actions. The callback actions should be used by your front end to capture / process data returned by the Dynamic Drop In UI.

Setup

The Dynamic Drop in UI must be provided with the following data to allow a correct setup:

```
<script src="https://unpkg.com/vue" charset="utf-8"></script>
```

```
<script
src="https://cdn.au.awstest.gfrnetworks.com/all/js/test/payment-
module/gf-payments.min.js" charset="utf-8"></script>
```

```
<link rel="stylesheet"
href="https://cdn.au.awstest.gfrnetworks.com/all/js/test/payment-
module/gf-payments.min.css">
```

```
<gf-payments
  process-button-id="element"
  on-tokenize-success="tokenSuccess"
  on-tokenize-error="tokenError"
  on-token-reset="tokenReset"
  on-fees-received="feesReceived"
  :use-system-defined-gateway="true">
</gf-payments>
```

Callbacks

- on-tokenize-success

Use this callback to collect the token data for the payment method for the API and other display's required on the page, including validation / enabling the submission button.

Example Callback Response:

```
{
  "paymentToken":
  "tokencc_bj_wh2qbd_vw96gv_b7xbz6_w46bk6_py5",
  "paymentType": "Visa",
  "paymentDisplay": "1111",
  "paymentExpiry": "11/2021",
  "additionalData": {
    "cardholderName": "Test Test"
  }
}
```

- on-tokenize-error

Use this callback to trigger any negative validation where required.

- on-token-reset

Use this callback to set negative validation / disable the submission button if the payment token becomes stale OR is modified.

- on-fees-received

Use this callback to display the fees for all available payment methods that have been retrieved for the Beneficiary / Event combination sent through.

Example Callback Response:

```
[
  {
    "merchant": "Master",
    "transactionRate": 0.011,
    "transactionFee": 0.2
  },
  {
    "merchant": "MX",
    "transactionRate": 0.0198,
    "transactionFee": 0.2
  },
  {
    "merchant": "Visa",
    "transactionRate": 0.011,
    "transactionFee": 0.2
  },
  {
    "merchant": "PayPal",
    "transactionRate": 0.021,
    "transactionFee": 0.2
  }
]
```

Other Settings

- event-Id

- beneficiary-id
- process-button-id
This can be any element on your UI which will always contain the latest data from the Dynamic Drop In UI. This is useful to use in case you have mishandled or missed a callback.
Example data on element:

```
<div id="element" data-payment-token="tokencc_bj_76bcdp_mq3tr7_pvm6jj_k8xvwh_hp7" data-payment-type="Visa" data-payment-display="1111" data-payment-expiry="12/2034" data-additional-data=""></div>
```
- :use-system-defined-gateway
This should be set to true when using the Dynamic Drop In UI

```
:use-system-defined-gateway="true"
```

Testing

Test Credit Cards

Visa: 4111 1111 1111 1111 12/34 123

Mastercard: 5555 5555 5555 4444 12/34 123

Amex: 3782 8224 631 0005 12/34 1234

Paypal: aupaypal@test.com gofundraise

GoFundraise API

This endpoint requires Authorization via a Bearer Token

Url: <https://api.gofundraise.com/v1/payments/payment>

Method: POST

Offline payments are restricted to the Payment Types: WPG, CASH, CHQ and Payment Sub Types: Offline

Offline Payments Support the full Billing Details, however, only support partial Payment objects. Payment Items support all options.

Billing Details require FirstName, LastName, and Email.

Payment Objects Support only the following options:

```
PaymentNotes
ParentPaymentId
EnableCommunications
PaymentToken - Required
AdditionalData - Required
```

The PaymentToken and AdditionalData will be returned from the Dynamic Drop In UI and both contain important data for the completion of the payment.

For testing you may use the following

Successful Payment

```
"PaymentToken": "fake-valid-nonce",  
"AdditionalData": "{ \"paymentType\": \"CC\", \"clientIpAddress\": \"127.0.0.1\", \"paymentSubType\": \"Visa\", \"referrerUrl\": \"https://www.testing.com\" },
```

Unsuccessful Payment

```
"PaymentToken": "fake-processor-declined-visa-nonce",  
"AdditionalData": "{ \"paymentType\": \"CC\", \"clientIpAddress\": \"127.0.0.1\", \"paymentSubType\": \"Visa\", \"referrerUrl\": \"https://www.testing.com\" },
```

Example

POST /v1/payments/payment

(Authorization token is required to use this endpoint.)

Minimum Required Fields for request:

```
{
  "BillingDetails": {
    "FirstName": "Test First",
    "LastName": "Test Last",
    "Email": "test@email.com",
  },
  "Payment": {
    "PaymentToken": "fake-valid-nonce",
    "AdditionalData": "{\"paymentType\": \"CC\", \"clientIpAddress\": \"127.0.0.1\", \"paymentSubType\": \"Visa\", \"referrerUrl\": \"https://developer.gfrnetworks.com/anary/paymentgateway-ui/index.html#!/UseCase1\"}",
    "PaymentItems": [
      {
        "EventCampaignId": 10014,
        "BeneficiaryAccountId": 104,
        "ProductType": 1,
        "ProductDescription": "Other Item",
        "UnitPrice": 10,
        "Quantity": 1
      },
      {
        "EventCampaignId": 10014,
        "BeneficiaryAccountId": 104,
        "ProductType": 11,
        "ProductDescription": "User Pays Transaction Fee",
        "UnitPrice": 0.28,
        "Quantity": 1
      }
    ]
  }
}
```

Synchronous / Asynchronous Workflow

It is now possible to perform a payment request either synchronously or asynchronously by using a Boolean flag "WaitForCompletion" inside the payment object of your request. By default this is set to true to maintain backwards compatibility with existing synchronous usage.

```
"WaitForCompletion": true,
```

Example

POST /v1/payments/payment

(Authorization token is required to use this endpoint.)

Minimum Required Fields for request:

```
{
  "BillingDetails": {
    "FirstName": "Test First",
```

```

        "LastName": "Test Last",
        "Email": "test@email.com",
    },
    "Payment": {
        "WaitForCompletion": true,
        "PaymentToken": "fake-valid-nonce",
        "AdditionalData": "{\"paymentType\":\"CC\",\"clientIpAddress\":\"127.0.0.1\", \"paymentSubType\":\"Visa\", \"referrerUrl\":\"https://developer.gfrnetworks.com/canary/paymentgateway-ui/index.html#!/UseCase1\"}",
        "PaymentItems": [
            {
                "EventCampaignId": 10014,
                "BeneficiaryAccountId": 104,
                "ProductType": 1,
                "ProductDescription": "User Pays Transaction Fee",
                "UnitPrice": 10,
                "Quantity": 1
            }
        ]
    }
}

```

Synchronous Payments

A synchronous payment, where "WaitForCompletion" is null or true, attempts to perform the entire payment workflow prior to returning a result.

On completion of this process a synchronous payment will return a 201 (Created) result including currency, payment, billing details, and payment items.

```

{
  "Currency": {
    "Symbol": "$",
    "Name": "Australian dollar",
    "Code": "AUD",
    "IsoLanguageCode": "en-AU"
  },
  "BillingDetails": {
    "FirstName": "test",
    "LastName": "user",
    "Email": "test@user.com"
  },
  "Payment": {
    "PaymentId": 1234567,
    "DateCreated": "2021-08-23T16:03:07.6099856",
    "PaymentStatus": "A",
    "PaymentStatusDescription": "SUCCESS",
    "TotalAmount": 1.00,
    "PaymentCustomerReference": "6axbmde8",
    "PaymentType": "CC",
    "PaymentSubType": "VISA",
    "PaymentItems": [
      {

```

```
    "PaymentItemId": "12345678",
    "PaymentItemTotalAmount": 1.00,
    "Quantity": 1,
    "UnitPrice": 1.00,
    "PaymentItemDateCreated": "2021-08-23T06:03:08.1858078",
    "ProductType": 1,
    "ProductDescription": "Tax Deductible Donation",
    "Anonymous": false,
    "DisplayOnPage": true,
    "IncludeInTotal": true,
  }
]
}
```

It is possible for the generation of payment items to fail whilst the transaction itself completes. Please ensure that your implementation can gracefully handle an empty payment item array. If this situation occurs it will be attempted to be rectified during reconciliation and the items generated, or, where rectification is not possible, the transaction will be refunded.

Please note: A maximum timeout period of 29s applies to this endpoint. If the transaction exceeds this time for processing, it will be automatically voided/refunded.

Asynchronous Payments

An asynchronous payment, where "WaitForCompletion" is false, returns a reference GUID which can then be used to retrieve the status of a payment at a later time.

On successful creation of a reference GUID, an asynchronous request will return a 202 (Accepted) result.

```
{
  "ReferenceId": "b2d54513-414a-4cc5-a7a4-5cd8cbf60a8c"
}
```

Payment Status

The payment status endpoint is used to give an "at-a-glance" overview of a payment request and can be used to poll for a response to an asynchronous payment previously made.

Example

GET /v1/payments/paymentstatus/{{ReferenceId}}

(Authorization token is required to use this endpoint.)

```
{
  "PaymentId": null,
  "ReferenceId": "f8e56f3c-bf7a-4729-8804-47cb2137d361",
  "Status": {
    "Status": "Received",
    "StatusCode": null,
    "StatusMessage": null
  }
}
```

The Status Object

There are 3 possible statuses for a payment status

- Received, the initial status from an asynchronous payment request. A PaymentId may or may not be available at this stage
"Status": "Received",
- Success, this indicates that the payment was successfully authorized and submitted for settlement on the payment provider. Further details about the payment can be found by retrieving the payment details using the PaymentId.
"Status": "Success",
- Failure, this indicated that the payment failed to be authorized on the payment provider. Further details about the payment can be found by retrieving the payment details using the PaymentId.
"Status": "Failure",

The "StatusCode" and "StatusMessage" should be used for contextual information only as there is not currently a well known list of responses that can be used for further processing.

Payments Retrieval

It is possible to retrieve payment details using the relevant ids. This may be useful if requiring additional information when using the asynchronous payment flow and the payment status was "Success"

Entire Payment Object

To retrieve the entire payment object including billing, currency, and payment items the payment id is required.

Please note: The existence of a payment object does not indicate a successfully transaction, please use the PaymentStatus field for this detail.

Example

GET /v1/payments/payment/{{PaymentId}}

(Authorization token is required to use this endpoint.)

```
{
  "Currency": {
    "Symbol": "$",
    "Name": "Australian dollar",
    "Code": "AUD",
    "IsoLanguageCode": "en-AU"
  },
  "BillingDetails": {
    "FirstName": "test",
    "LastName": "user",
    "Email": "test@user.com"
  },
  "Payment": {
    "PaymentId": 1234567,
    "DateCreated": "2021-08-23T16:03:07.6099856",
    "PaymentStatus": "A",
    "PaymentStatusDescription": "SUCCESS",
    "TotalAmount": 1.00,
    "PaymentCustomerReference": "6axbmde8",
    "PaymentType": "CC",
    "PaymentSubType": "VISA",
    "PaymentItems": [
      {
        "PaymentItemId": "12345678",
        "PaymentItemTotalAmount": 1.00,
        "Quantity": 1,
        "UnitPrice": 1.00,
        "PaymentItemDateCreated": "2021-08-23T06:03:08.1858078",
        "ProductType": 1,
        "ProductDescription": "Tax Deductible Donation",
        "Anonymous": false,
        "DisplayOnPage": true,
        "IncludeInTotal": true,
      }
    ]
  }
}
```

```
}  
}
```

Individual Payment Item Object

It is possible to retrieve the details about an individual payment item including event, beneficiary, and fundraising page information. To do so a payment item id is required

Please note: The existence of a payment item object does not indicate a successfully transaction, please resolve this detail using the PaymentId field to retrieve the entire payment object for this detail.

Example

GET /v1/payments/paymentitem/{{PaymentItemId}}

```
{  
  "PaymentId": 1234567,  
  "PaymentItemId": "12345678",  
  "PaymentItemTotalAmount": 1.00,  
  "Quantity": 1,  
  "UnitPrice": 1.00,  
  "PaymentItemDateCreated": "2021-08-23T06:03:08.1858078",  
  "ProductType": 1,  
  "ProductDescription": "Tax Deductible Donation",  
  "Anonymous": false,  
  "DisplayOnPage": true,  
  "IncludeInTotal": true,  
}
```

Response Objects

Currency

The currency object displays details about the currency the transaction was made in

```
"Currency": {  
  "Symbol": "$",  
  "Name": "Australian dollar",  
  "Code": "AUD",  
  "IsoLanguageCode": "en-AU"  
},
```

Billing

The billing details include the billing information for the transaction

```
"BillingDetails": {  
  "Organisation": "",  
  "DonorType": "",  
  "UserId": null,  
  "Title": "",  
  "FirstName": "test1",  
  "LastName": "user1",  
  "Email": "test@user.com",  
  "PhoneNumber": "",  
  "MobileNumber": "",  
}
```

```
"StreetAddress": "",
"Suburb": "",
"PostCode": "",
"State": "",
"Country": ""
},
```

Payment

```
"Payment": {
  "PaymentId": 1234567,
  "DateCreated": "2021-08-23T16:03:07.6099856",
  "PaymentStatus": "A",
  "PaymentStatusDescription": "SUCCESS",
  "TotalAmount": 1.00,
  "PaymentCustomerReference": "",
  "ParentPaymentId": null,
  "PaymentType": "",
  "PaymentSubType": "",
  "PaymentNotes": "",
  "PaymentItems": [
  ]
}
```

PaymentItem

```
{
  "PaymentItemId": "123456789",
  "PaymentItemTotalAmount": 1.00,
  "Quantity": 1,
  "UnitPrice": 1.00,
  "ParentPaymentItemId": null,
  "PaymentItemCustomerReference": "",
  "PaymentItemDateCreated": "2021-08-23T06:03:08.1858078",
  "ProductType": 1,
  "ProductDescription": "",
  "Classification": "",
  "FundraisingMessage": "",
  "PaymentItemNotes": "",
  "Anonymous": false,
  "DisplayOnPage": true,
  "IncludeInTotal": true,
  "Beneficiary": {},
  "Event": {},
  "Page": {}
}
```

Beneficiary

```
"BeneficiaryAccountId": "104",
"BeneficiaryImagePath": "http://www.au.awstest.gfrnetworks.com//Upload/beneficiary/104/logo.png",
```

```
"BeneficiaryName": "X Demo 01",  
"BeneficiaryUrl": "http://www.au.awstest.gfrnetworks.com/beneficiary/Demo",
```

Event

```
"EventCampaignId": "10014",  
"EventImagePath": "https://cdn.awstest.gfrnetworks.com/Upload/Events/10014/Event1  
32348656632527289.jpg",  
"EventName": "Making A difference",  
"EventUrl": "makingadifference.au.awstest.gfrnetworks.com"
```

Page

```
"PageId": 4567894,  
"PageImagePath": "string",  
"PageTitle": "string",  
"PageCreatorName": "string",  
"PageUrl": "string",  
"PageDonationUrl": "string"
```