VISA everywhere
you want to be

Visa Checkout

Getting Started

Effective: January 23, 2019
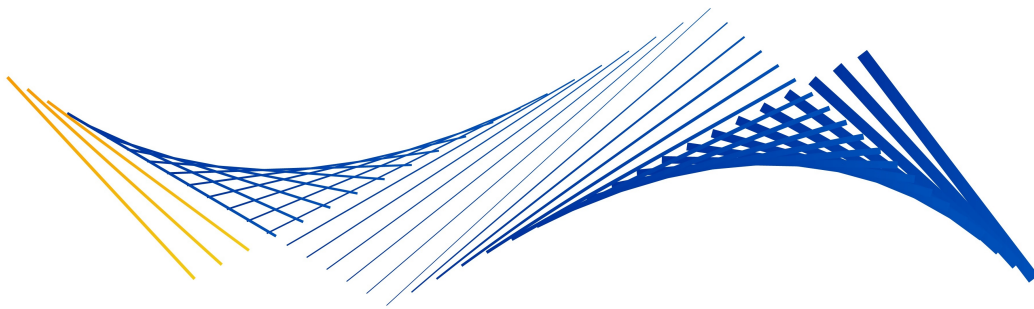
Important Note on Copyright

If you have technical questions or questions regarding a Visa service or capability, contact your Visa representative.

Contents

Introduction to Visa Checkout

1

About Visa Checkout

Consumers are moving to digital payments for a simplified and more convenient payment experience. Visa Checkout is an online and mobile payment service that helps speed up the checkout process, transforming shoppers into buyers. Merchants can easily integrate Visa Checkout into their existing payment processes on their E-commerce site or app and provide consumers with a fast and convenient way to pay. Consumer sign up is simple and secure, the Visa Checkout button works across all devices, and works with any major credit or debit card. Consumers can store their card and shipping information with Visa Checkout so that they only need to enter a username and credential, such as a password or biometric, at checkout.

Visa Checkout can assist merchants with:

- Easy integration

    - Three simple steps add the Visa Checkout button to a merchant's website.

    - A software development kit (SDK) is available for integration with a merchant's app.

- Conversion

    - "Visa Checkout's Stay Signed In and Remember Me" feature makes it even easier for consumers to check out and allows merchants to capture sales that might otherwise be lost.

    - Visa Checkout provides access to biometrics, such as a thumbprint or face recognition, when using a mobile app on devices that support the use of biometrics.

- Fraud management

    - Visa Checkout evaluates risk at every stage of checkout and helps to create a secure foundation for a seamless shopping experience.

    - Visa Checkout's advanced tools include device fingerprinting and step-up authentication to help reduce the risk of fraudulent transactions.

- Visa Account Updater (VAU)

    - Provides merchants with up-to-date account information.

    - May increase authorization approvals.

    - Helps increase sales and consumer retention.

Visa Checkout currently supports:

- Consumers registered in Argentina, Australia, Brazil, Canada, China, Chile, Columbia, France, Hong Kong, India, Ireland, Kuwait, Malaysia, Mexico, New Zealand, Peru, Poland, Qatar, Saudi Arabia, Singapore, South Africa, Spain, United Arab Emirates, United Kingdom, Ukraine, and the United States

- Visa, MasterCard, American Express and Discover cards

About Tokenization

The global payments industry has been adopting payment tokenization as a means to enhance online and mobile transaction security by replacing sensitive payment account information

with payment tokens. A payment token replaces the primary account number (PAN) with a unique token allowing processing by all participants in the payments ecosystem without exposing a consumer's data.

To help protect against unauthorized use, a payment token is limited to a specific domain, such as a particular merchant, device, or channel. Additionally, every e-commerce transaction that is initiated with a token requires a cryptogram to act as the password that ties the token to a customer's PAN. These underlying controls are a key benefit and help to protect against cross-channel fraud.

Merchants and consumers benefit because payment tokenization:

• Allows payments to be processed without exposing account details

• Protects sensitive information from theft and fraud

• Reduces the Payment Card Industry (PCI) scope for merchants

For specific integration and usage information, see the *Visa Checkout Integration Guide*.

Creating a Sandbox Account and Credentials 2

Creating a Sandbox Account

You must first get access to the Visa Developer site to create a sandbox account. This process is as simple as registering your name, email address, organization, and creating a password.

Steps

1. Go to https://developer.visa.com/ to log into the Visa Developer Center

2. Click **Register Here** to create your sandbox account and get started with Visa Developer.

3. Complete the information for the registration form and click **Register**.

Adding a New Project and Obtaining Credentials

Prerequisites

You must have already created a Sandbox account.

Steps

1. On the Dashboard screen, click **Add New Project** to create a new project for Visa Checkout, which must include the Get Payment Data and the Update Payment Information APIs.
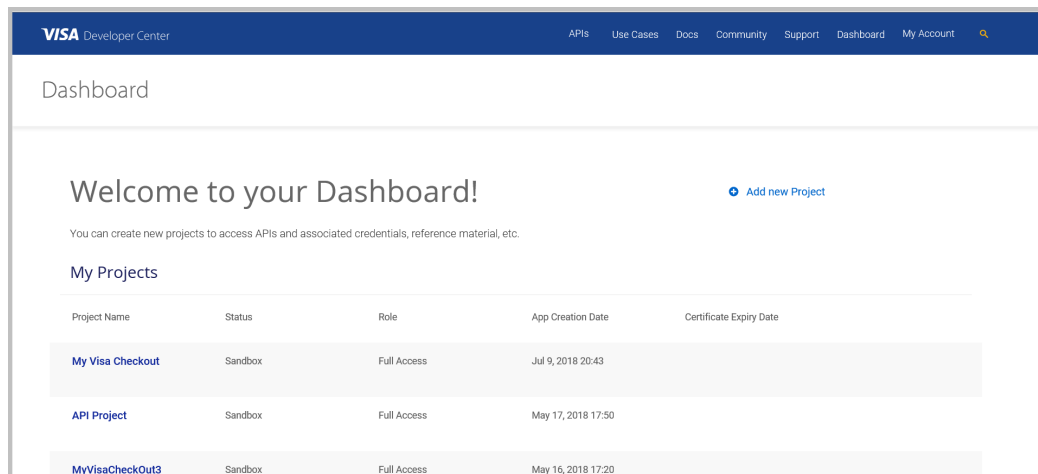
2. In the Dashboard — Add New Project screen, enter the name of the new project in the **Project Name** field and then check **Visa Checkout**.

VISA Developer Center   APIs   Use Cases   Docs   Community   Support   Dashboard   My Account   🔍

Dashboard - Add New Project

Project Name *

My Visa Project

Project Description

Max 1000 chars.

Offers and Benefits

☐ Visa Merchant Offers Resource Center
Get access to pre-negotiated offers from top merchants around the globe instantly.

☐ Visa Card Eligibility Service
Provide eligibility access rights to exclusive offers and benefits.

☐ Visa Offers Platform (Restricted) ❷
Enhance capabilities to provide promotional offers to card holders.

Commercial

☐ B2B Virtual Account Payment Method
Flexible and secure payment method for B2B market segments using virtual accounts.

☐ Visa Business Data Solutions (Restricted) ❷
Fast and simple integration to Commercial Data.

☐ Visa B2B Connect
A global technology platform designed to enable direct and predictable exchange of value for banks and their corporate clients.

☐ Visa Supplier Matching Service
Drive spend to suppliers that accept Visa Commercial Payments.

Payment Methods

☐ CyberSource Payments
Process payments securely through CyberSource.

☐ Visa Direct
Transfer funds seamlessly, securely and quickly.

☑ Visa Checkout
Deliver safe, simple and speedy online shopping checkout.

Risk and Fraud

☐ Mobile Location Confirmation
Make better fraud risk decisions using geolocation intelligence.

☐ Payment Account Validation
Access multiple methods of ensuring that a payment account is valid.

☐ Visa Risk Manager (Restricted) ❷
Enable issuers to create strategies to help prevent fraud associated with authorization and token provisioning requests.

☐ Visa Travel Notification Service
Incorporate cardholder self-reported travel into your authorization decisions.

3. At the bottom of the Dashboard — Add New Project screen, click **Create Project**.
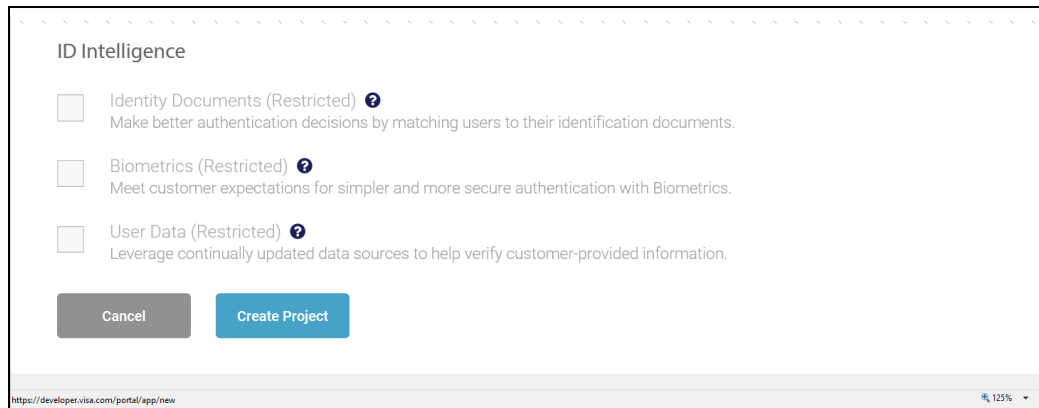
4. In the Dashboard — My Visa Checkout screen, expand **Visa Checkout**.
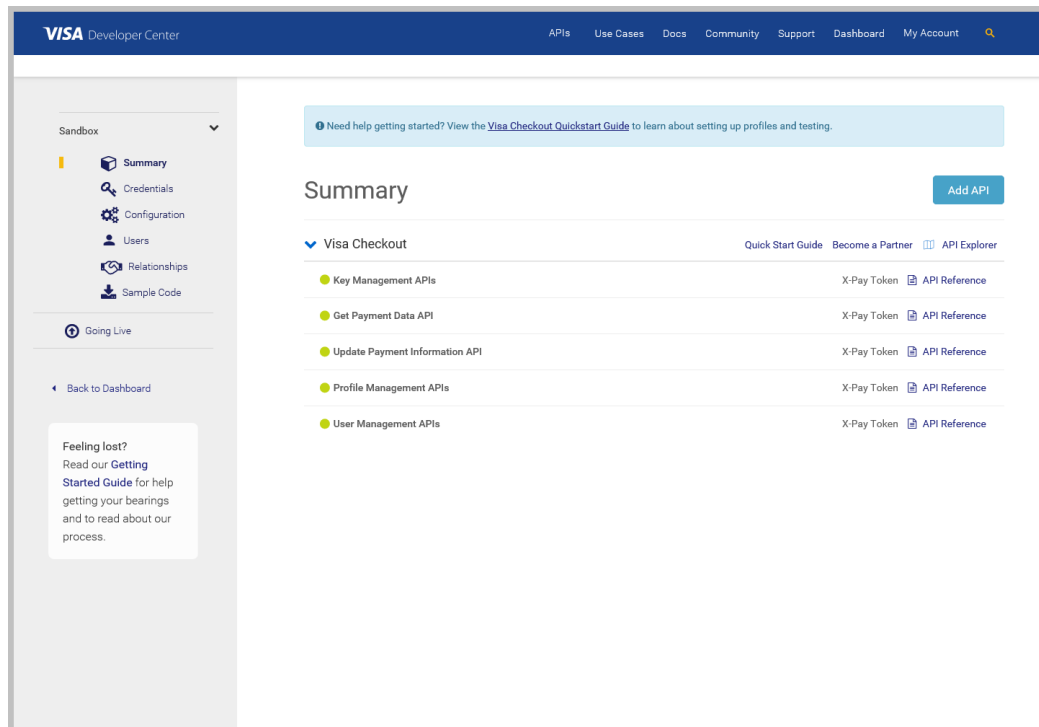
The list includes the Get Payment Data and Update Payment Information APIs. You may need to wait for a few minutes until the radio buttons turn green.

5. Click **Credentials** on the left-hand menu to view the API key and shared secret for your project.

Results

You have finished creating a new project and have the API key credentials.

Next Steps

You will need your API key and associated shared secret to integrate Visa Checkout button on your web page and both the API key and shared secret to make API calls. Keep your shared secret safe; only use your shared secret on a secure server.
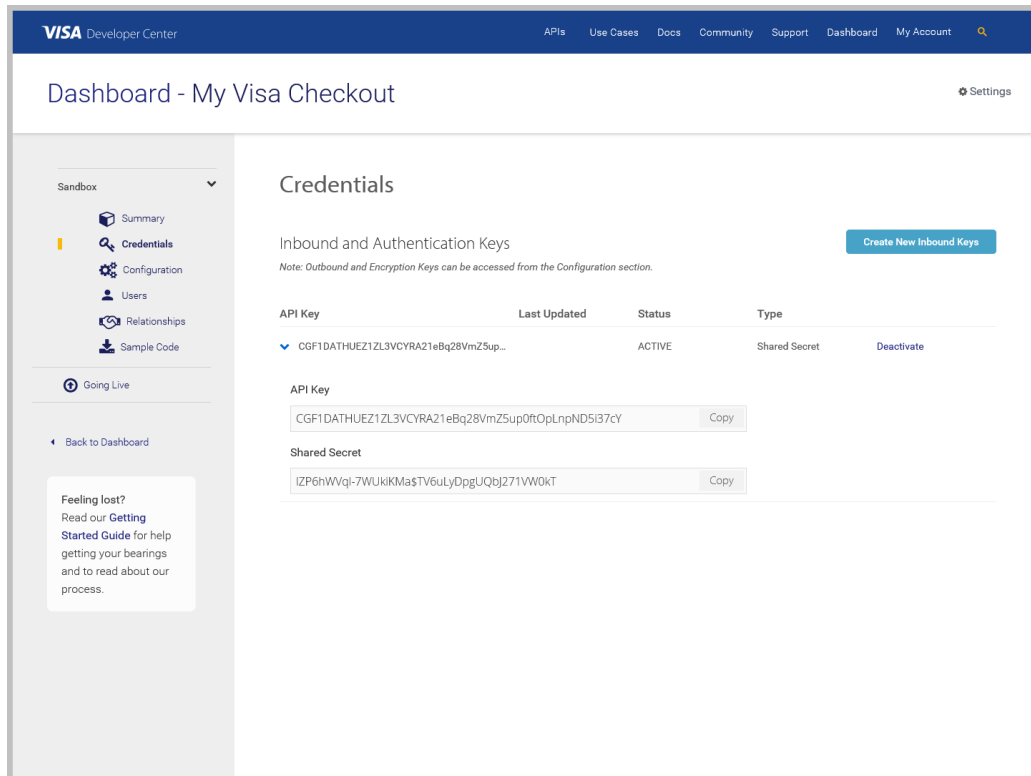
Creating an Encryption Key

You use an encryption key to decrypt the consumer information payload in responses from Visa Checkout.

Prerequisites

You must have already created a Visa Developer account and a project. If you do not see a **Configuration** option in the in the left-navigation menu, contact your Visa Checkout representative; you must use this menu option to create an encryption key.

To create an encryption key:

Steps

1. In the left-hand navigation panel of the Dashboard in the Visa Checkout Development Center, click **Configuration** and expand **Visa Checkout**.
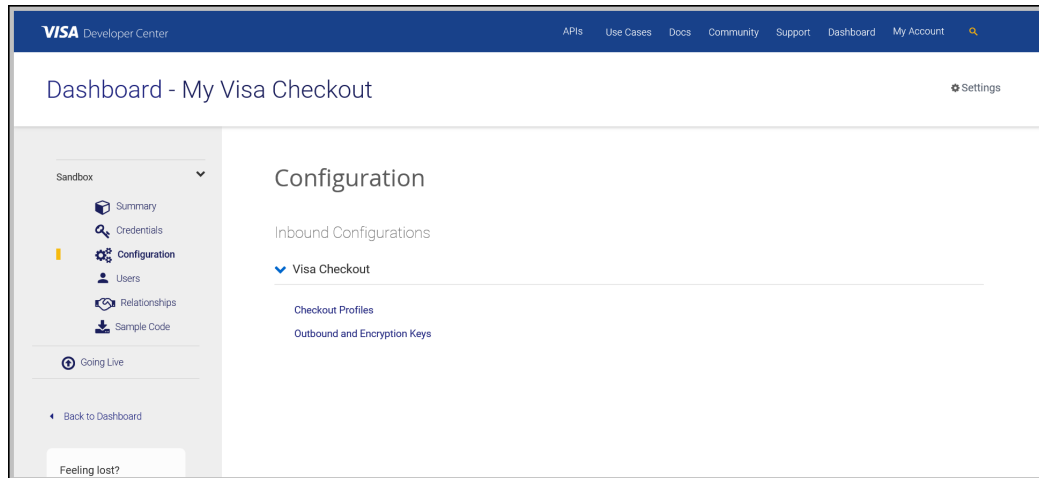
2. Click **Outbound and Encryption Keys**.

3. On the OUTBOUND AND ENCRYPTION KEYS tab, click **Create New**.
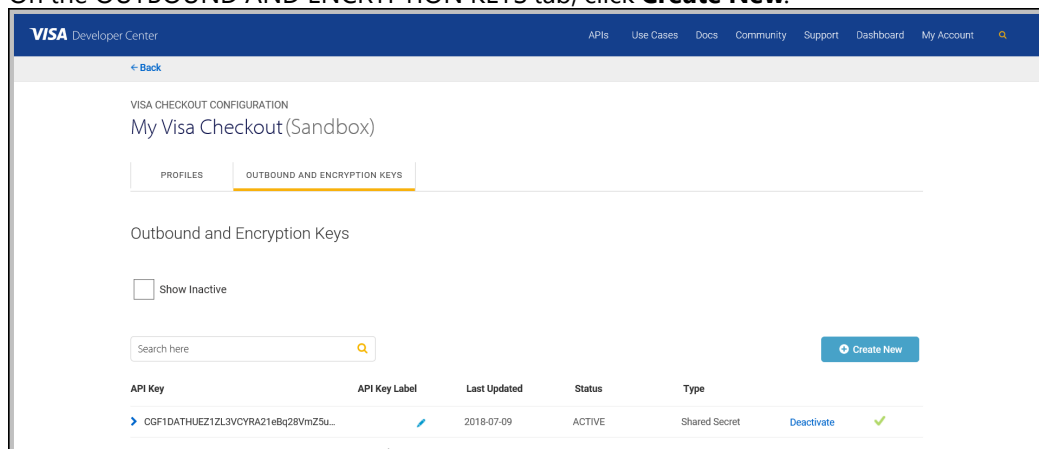
4. Enter an **API Key Label**, such as **Encryption**, to remember the usage of the key.
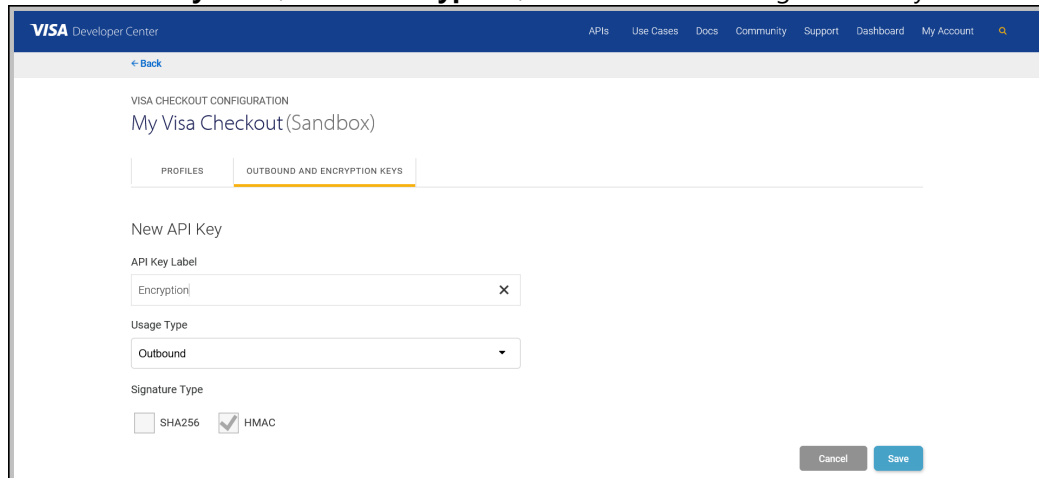
5. Choose `Encryption` from the **Usage Type** drop down menu.

6. Click **Save** to keep the key.

Results

You can open the drop down for your encryption key and save the key value and it's associated shared secret.

Next Steps

You need your encryption key and associated shared secret to integrate the **Visa Checkout** button on your web page, when calling the Get Payment Data API. You also need its associated shared secret to decrypt the consumer information payload. Keep your shared secret safe; only use your shared secret on a secure server.

Quick Start Tutorial 3

Getting Started with the Integration

Integrating Visa Checkout with your website consists of the following steps, as shown below:

Steps

1. Add Visa Checkout buttons to your web pages and include the necessary JavaScript to manage any events that are associated with these buttons.

2. Obtain and decrypt consumer information from a payload that a Visa Checkout event returns.

3. Update Visa Checkout after the payment is complete.

Results

Visa Checkout is very flexible in the ways you can integrate. This tutorial shows only one of those ways, which confines the changes to only web pages, which the exception of handling decryption of the payload on a secure server. It also shows just the minimum effort required, without optional changes to customize Visa Checkout lightbox appearance and behavior. You can integrate Visa Checkout on your pages in a similar way, or you might choose to use Visa Checkout APIs for some of the work. Depending on your configuration, your payment processor or an e-commerce partner might do some or all of the steps involved for you. For complete information about integrating Visa Checkout, see *Visa Checkout Integration Guide*.

Adding a Visa Checkout Button to a Web Page

To enable Visa Checkout from a web page, you must place a **Visa Checkout** button on the page, add JavaScript to submit a request to Visa Checkout when a consumer clicks the button, and handle responses associated with Visa Checkout events.

Prerequisites

Before you start, you must set up a Sandbox account and credentials, consisting of an API key and shared secret, to communicate with Visa Checkout. You must also create an encryption key with its associated shared secret in your Sandbox account. Click **Create Account** in the Visa Developer Center or contact your Visa Checkout representative to get set up.

To add a Visa Checkout button to a web page:

Steps

1. Add the `onVisaCheckoutReady` JavaScript function to the head and include the `V.init` initialization handler with your API key and encryption key:

```
<head>
 <script type="text/javascript">
 function onVisaCheckoutReady() {
  V.init( {
    apikey: "...",
    encryptionKey: "...",
    ...
    }
   });
   ...
  }
 </script>
</head>
```

Later, you can go back and specify other settings, either in your Sandbox profile or in `V.init`, to customize the appearance of the Visa Checkout lightbox. For more information, see the *Visa Checkout Integration Guide*.

2. Inside `V.init`, specify the amount and currency of the consumer's payment request in the `paymentRequest` properties:

```
paymentRequest: {
 currencyCode: "USD",
 subtotal: "10.00"
}
```

You can also specify amounts for tax, discounts, shipping, etc., or you can include these amounts when you update the payment information after the lightbox closes. It is not necessary to have all information available in advance. The subtotal is the total before these other amounts are included.

3. Add `V.on` event handlers, one for each of the following kinds of events:

Event Handler	Description
`payment.success`	Indicates that the consumer has agreed to the payment request
`payment.cancel`	Indicates that the consumer cancelled the request
`payment.error`	Indicates that an error has occurred

The following code examples demonstrate the event handlers.

```
V.on("payment.success",function(payment)
 {alert(JSON.stringify(payment));});

V.on("payment.cancel",function(payment)
 {alert(JSON.stringify(payment));});

V.on("payment.error",function(payment,error)
```

```
{alert(JSON.stringify(error));});
```

4. Place one or more **Visa Checkout** buttons in the body of the page:

```
<body>
  <img alt="Visa Checkout" class="v-button" role="button"
   src="https://sandbox.secure.checkout.visa.com/wallet-services-web/
    xo/button.png"/>
...
```

You can provide additional properties. Some properties may override those properties that are set in the `V.init` handler; for example, you can restrict the card brands that are accepted for each button. For descriptions of all options, see the *Visa Checkout Integration Guide*.

5. At the end of the body, install the Visa Checkout SDK JavaScript library, `sdk.js`:

```
<script type="text/javascript">
src="https://sandbox-assets.secure.checkout.visa.com/
checkout-widget/resources/js/integration/v1/sdk.js">
</script>
/<body>
```

Results

After completing these steps, you should have a web page whose source looks like this:

```
<html>
 <head>
  <script type="text/javascript">
  function onVisaCheckoutReady() {
   V.init( {
    apikey: "...",
    encryptionKey: "...",
    paymentRequest: {
     currencyCode: "USD",
     subtotal: "10.00"
    }
   });
   V.on("payment.success", function(payment) {
    document.write("payment.success: \n" + JSON.stringify(payment));
   });
   V.on("payment.cancel", function(payment) {
    document.write("payment.cancel: \n" + JSON.stringify(payment));
   });
   V.on("payment.error", function(payment, error) {
    document.write("payment.error: \n" +
     JSON.stringify(payment) + "\n" +
     JSON.stringify(error));
   });
  }
  </script>
</head>

<body>
<img alt="Visa Checkout" class="v-button" role="button"
src="https://sandbox.secure.checkout.visa.com/wallet-services-web/
xo/button.png" />
<script type="text/javascript"
src="https://sandbox-assets.secure.checkout.visa.com/
checkout-widget/resources/js/integration/v1/sdk.js">
```

```
    </script>
    </body>
</html>
```

Next Steps

You can now proceed to test this web page or test one of your own pages that you have modified by these steps.

About Personal Account Numbers for Testing

When you create a personal account in the sandbox for your testing, you must specify at least 1 payment method.

When you need to create a personal account, you can follow the instructions in this document

You can enter any values in this screen that make sense to you, except for a card number. Because card numbers are validated by Visa Checkout logic, the following card numbers are provided for your use in the sandbox:

Brand	Test Card Number
Visa	4005520201264821
Visa (Returns a token)	4622943127010693 (CVV2 121)
MasterCard	5500005555555559
American Express	340353278080900
Discover	6011003179988686

Note

The name on card, expiration date, and security code are not checked in the Visa Checkout sandbox. You can use any valid values.

Contact your Visa Checkout representative for card numbers that return token information.

Testing the Web Page

You can use your web page to start experimenting with Visa Checkout. You must create a consumer account with at least two payment instruments for testing, one of which must be a tokenized payment instrument, meaning that the payload contains a token in place of a PAN.

Prerequisites

You must have account numbers and security codes for payment instruments, such as debit or credit cards, that are associated with a token and a PAN. Contact your Visa Checkout representative for these numbers if they are not listed below.

To test your button:

Steps

1. Invoke your web page and click the **Visa Checkout** button:

If everything is set up properly, the Visa Checkout lightbox appears:

2. Enter your test consumer's email address and password and click **SIGN IN TO VISA CHECKOUT**; or, if you do not yet have a consumer account for testing, click **Continue as a new customer >** and create an account, including your test card numbers and delivery address:

3.   Choose the consumer's card and delivery address.

   a.   Click **Add** to add a payment instrument's card details or use the directional arrows (**<** and **>** to choose an existing payment instrument.

   The cards below return PAN account information. Contact your Visa Checkout representative for card numbers that return token information.

Brand	Test Card Number
Visa	4005520201264821
MasterCard	5500005555555559
American Express	340353278080900
Discover	6011003179988686

   Note

   The name on card, expiration date, and security code are not checked in the Visa Checkout sandbox. You can use any valid values.

   b.   Click **Add** to add a delivery address or use the directional arrows (**<** and **>**) to choose an existing delivery address.

4.   Click the **Continue** button to create the payment request:

   The `payment.success` event (`V.on` handler) triggers the response that appears in your web browser. It should look similar to the following one after pretty printing the JSON structure:

```
{
 "encKey": "...",
 "encPaymentData": "...",
 "partialShippingAddress": {
  "countryCode": "US",
  "postalCode": "19106"
 },
 "partialPaymentInstrument": {
  "lastFourDigits": "0693",
  "paymentType": {
   "cardBrand": "VISA",
   "cardType": "DEBIT"
  }
 },
 "paymentMethodType": "TOKEN",
 "callid": "1234567890123456789",
  "vInitRequest": {
    ...
  }
}
```

Note

The response indicates that a tokenized payment instrument was used `"paymentMethodType": "TOKEN"`.

Results

You have successfully tested whether or not the button functions properly.

Next Steps

You are now ready to obtain and decrypt the payload. You will use the value in the `callid` field of the response in Visa Checkout APIs. You may use other fields in the response, such as the last 4 digits of the associated account number printed on the card and partial shipping address, to display order information on your site's payment review page.

Note

The `paymentMethodType` parameter is used to distinguish whether payload contains token or PAN information, which may affect how you will process the payload. Your integration must handle both kinds of payloads. The specific handling that might be required depends on your partner or payment processor.

Obtaining and Decrypting a Payload

When the `payment.success` event occurs, Visa Checkout can return a payload that includes a consumer's information, which is encrypted. You can also obtain the payload using the Get Payment Data API.

Prerequisites

You must decrypt the consumer information payload on a secure server. Ensure that the dynamic encrypted key, `encKey`, and the encrypted payload, `encPaymentData`, from the response are on your secure server. You will also need the shared secret associated with your encryption key (`encryptionKey`) on your secure server.

Note

If you need complete token information or the full account number (PAN) from the payload, your Visa Checkout account must have the necessary permissions from Visa Checkout.

Visa Checkout provides code in several languages that you can use to decrypt the payload using the prerequisite fields. For more information, see the *Visa Checkout Integration Guide*.

Results

After you decrypt the payload, the fields include the following ones for a token:

```
{
  "paymentRequest": {
    "currencyCode": "USD",
    "subtotal": "10"
  },
  "userData": {
    "userFirstName": "Test",
    "userLastName": "Test",
    "userFullName": "Test Test",
    "userName": "testvco1@...com",
    "userEmail": "testvco1@...com",
    "encUserId": "..."
  },
  "creationTimeStamp": 1529600118302,
  "paymentInstrument": {
    "id": "...",
    "lastFourDigits": "0693",
    "tokenInfo": {
      "token": "4895370013106924",
      "tokenRange": "489537001",
      "last4": "6924",
      "expirationDate": {
        "month": "12",
        "year": "2024"
      }
    },
    "cryptogramInfo": {
      "cryptogram": "...",
      "eci": "07",
      "tokenCryptogramType": "TAVV",
      "expirationTimestamp": "2018-09-19T16:55:33.000Z"
    },
    "paymentAccountReference": "V00100130180107108227346159923",
    "billingAddress": {
      "personName": "Test Test",
      "firstName": "Test",
      "lastName": "Test",
      "line1": "124 Market Street",
      "city": "Philadelphia",
      "stateProvinceCode": "PA",
      "postalCode": "19106",
      "countryCode": "US",
      "phone": "5674563456",
      "default": false
    },
    "verificationStatus": "VERIFIED",
    "expired": false,
    "cardArts": {
      "cardArt": [
        {
          "baseImageFileName": "https:....png",
          "height": 105,
          "width": 164
```

```
        }
      ]
    },
    "issuerBid": "14",
    "nickName": "Token 1",
    "nameOnCard": "Test Test",
    "cardFirstName": "Test",
    "cardLastName": "Test",
    "paymentType": {
      "cardBrand": "VISA",
      "cardType": "DEBIT"
    }
  },
  "shippingAddress": {
    "id": "...",
    "verificationStatus": "VERIFIED",
    "personName": "Test Test",
    "firstName": "Test",
    "lastName": "Test",
    "line1": "124 Market Street",
    "city": "Philadelphia",
    "stateProvinceCode": "PA",
    "postalCode": "19106",
    "countryCode": "US",
    "phone": "5674563456",
    "default": false
  },
  "riskData": {
    "advice": "UNAVAILABLE",
    "score": 0,
    "avsResponseCode": "M",
    "cvvResponseCode": "M",
    "ageOfAccount": 16
  },
  "visaCheckoutGuest": false,
  "walletInfo": {
    "walletName": "VISA_CHECKOUT"
  },
  "partialShippingAddress": {
    "countryCode": "US",
    "postalCode": "19106"
  },
  "paymentMethodType": "TOKEN"
}
```

Next Steps

Securely transfer decrypted fields to your partner or payment processor.

Creating an x-pay-token Header

Visa Checkout APIs require an `x-pay-token` header in the request, which includes a timestamp and an SHA-256 hash using your API key's shared secret of the following items: timestamp, resource path (API name), request query string, and complete request body, if it exists, to compute a Hash-based Message Authentication Code (HMAC). You can use this example to test your hashing algorithm for creating an `x-pay-token` header.

Prerequisites

You must first implement an HMAC SHA-256 hashing algorithm, such as one of those shown in the *Visa Checkout Integration Guide*, to compare your hash against this example.

To create a test `x-pay-token` header for comparison with the one in this example, follow these steps.

Important

Strings must not contain white space, including line breaks. The strings shown in the example do not have line breaks; they are shown here only for readability.

Steps

1. Use the following data:

    • Shared secret associated with the API key:

`B123456789B123456789B123456789B123456789`

    • Time of the request, as a UNIX UTC timestamp:

`1544444444`

    • API resource path (name of the endpoint):

`payment/data/1234567890123456789`

    • Request query string:

`apikey=A123456789A123456789A123456789A123456789A12345678`
`&encryptionKey=`
`E123456789E123456789E123456789E123456789E12345678`

    • The request body, which is empty in this example.

2. Form your string to be hashed from the concatenated parts, as follows; do not use line breaks or other white space:

    Your concatenated consists of the following items in order: UNIX UTC timestamp, resource path (API name), request query string, and complete request body, if it exists

    The string to be hashed should look like the following one without line breaks or white space:

`1544444444payment/data/1234567890123456789`
`apikey=A123456789A123456789A123456789A123456789A12345678&`
`encryptionKey=E123456789E123456789E123456789E123456789E12345678`

Note

The question mark (`?`) between the resource path and the query parameters is **not** included.

Results

Your hashed string should match the following string:

`D6F356CE717B3EBA9A2463C4575B9626E981951CD4535F7072C5901F8E16A223`

The complete `x-pay-token` looks like this:

`x-pay-token: xv2:1544444444:D6F356CE717B3EBA9A2463C4575B9626E9819...`

Next Steps

In another example that includes a request body, you can change the resource path and request body to specify the Update Payment Info API. The string to hash would be as follows without line breaks or white space:

```
1544444444payment/info/1234567890123456789
apikey=A123456789A123456789A123456789A123456789A12345678
{
  "updateInfo": [
    {
      "payInfo": {
        "reason": "Just_a_test",
        "avsResponseCode": "Y",
        "total": 91.00,
        "currencyCode": "USD",
        "eventStatus": "Success",
        "eventType": "Authorize",
        "payTransId": "123456789"
      }
    },
    {
      "orderInfo": {
        "currencyCode": "USD",
        "discount": 25.00,
        "eventType": "Confirm",
        "giftWrap": 0.00,
        "misc": 0.00,
        "orderId": 123456,
        "promoCode": "VISACHECKOUT25",
        "reason": "Order_placed",
        "shippingHandling": 10.00,
        "subtotal": 100.00,
        "tax": 6.00,
        "total": 91.00
      }
    }
  ]
}
```

Note

As a convenience, you can copy and paste the following string as the request body shown above:

```
{"updateInfo":[{"payInfo":{"reason":"Just_a_test",
"avsResponseCode":"Y","total":91.00,"currencyCode":"USD",
"eventStatus":"Success","eventType":"Authorize",
"payTransId":"123456789"}},{"orderInfo":{"currencyCode":"USD",
"discount":25.00,"eventType":"Confirm","giftWrap":0,"misc":0,
"orderId":123456,"promoCode":"VISACHECKOUT25","reason":"Order_placed",
"shippingHandling":10.00,"subtotal":100.00,"tax":6.00,"total":91.00}}]}
```

White space has already been removed; however, you must remove line breaks from this string.

Your hashed string should match the following string:

```
69340A54D03DF0163DD86B2C0F367BA5EDED800A3D90BD0E125D234F13EADD9D
```

The complete `x-pay-token` looks like this:

```
x-pay-token: xv2:1544444444:69340A54D03DF0163DD86B2C0F367BA5EDED8...
```

Getting Payment Data via API

You can call the Visa Checkout Get Payment Data API from your secure server to obtain the payload. The payload will be encrypted if you request full information, which includes the token's cryptogram or an actual account number, and are eligible to receive full information.

Prerequisites

You must create an `x-pay-token`, which is used in a request header. You must specify your encryption key when calling the Get Payment Data API to receive encrypted payloads.

The following steps show how to obtain the response to the Get Payment Data API:

Steps

1. Specify a `Content-Type` header.

   ```
   Content-Type: application/json
   ```

2. Specify an `Accept` header.

   ```
   Accept: application/json
   ```

3. Specify an `x-pay-token` header.

   ```
   x-pay-token: xv2:1529603146:...
   ```

4. Specify a `GET` operation for the `payment/data/{callid}` endpoint and include your `apikey` and `encryptionKey` query parameters.

   #### Note

   The endpoint and query parameters do not include new lines; these breaks are shown in documentation for readability only.

```
https://sandbox.api.visa.com/wallet-services-web/
payment/data/1234567890123456789?apikey=...&encryptionKey=...
```

5. Execute the fully formed request.

   For testing, you can use cURL to execute the following request from your browser:

```
curl -k -v -X GET \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-H 'x-pay-token: xv2:1529603146:...' \
-d '' "https://sandbox.api.visa.com
/wallet-services-web/payment/data/1234567890123456789
?apikey=...&encryptionKey=..."
```

   #### Note

   The Get Payment Data API does not have a request body.

Results

The JSON response includes the following fields for an encrypted payload containing a tokenized payment instrument:

```
{
  "partialShippingAddress": {
    "countryCode": "US",
    "postalCode": "19106"
  },
  "partialPaymentInstrument": {
    "lastFourDigits": "0693",
```

```
    "paymentType": {
      "cardBrand": "VISA",
      "cardType": "DEBIT"
    }
  },
  "encPaymentData": "...",
  "encKey": "...",
  "paymentMethodType": "TOKEN"
}
```

Next Steps

Decrypt and securely send required payload fields to your partner or payment processor. Contact your partner or payment processor for information about their required fields.

Updating Visa Checkout With the Payment Information

After you accept your consumer's payment, you must update Visa Checkout with the payment details.

Prerequisites

You must create an `x-pay-token`, which is used in a request header.

The following steps show how to obtain the response to the Update Payment Info API:

Steps

1. Specify a `Content-Type` header.

   `Content-Type: application/json`

2. Specify an `Accept` header.

   `Accept: application/json`

3. Specify an `x-pay-token` header.

   `x-pay-token: xv2:1529603146:...`

4. Specify a request body that contains payment and order information. For more information, see the *Visa Checkout Integration Guide*.

5. Specify a `PUT` operation for the `payment/info/{callid}` endpoint and include the `apikey` query parameter.

   #### Note

   The endpoint and query parameters do not include new lines; these breaks are shown in documentation only for readability.

```
https://sandbox.api.visa.com/wallet-services-web/
payment/info/1234567890123456789?apikey=...
```

6. Execute the fully formed request.

   For testing, you can use cURL to execute the following request from your browser:

```
curl -k -v -X PUT \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-H 'x-pay-token: xv2:1530307172:...' \
-d '{"updateInfo":[{"payInfo":{"reason":"...",...}},
{"orderInfo":{"currencyCode":"USD",...}}]}'
```

```
https://sandbox.api.visa.com
/wallet-services-web/payment/info/1234567890123456789?apikey=...
```

Results

If successful, you receive a 200 status message in the response; the response body is empty.

Branding Requirements 4

About Branding Requirements

Visa Checkout includes branding requirements that describe how to use Visa Checkout visual assets on your web site.

Your use of Visa Checkout assets is subject to the license agreement you have entered into for use of Visa Checkout services.

Important

Visa may update or change Visa Checkout assets at any time.

About an Asset's Usage and Placement

The following rules apply to the use and placement of **Visa Checkout** buttons and acceptance marks:

- Do not change the functionality of these assets.
- Do not alter the size, shape, orientation, or any other aspect of the images. In the event that you need an alternative variation, contact your Visa Checkout representative.
- Make sure that Visa Checkout buttons are placed on an equal level with other action items on the page, regardless of the orientation of the page. If the buttons are below the fold, it helps to place an additional button at the top of the page.

About Visa Checkout Button Requirements

The **Visa Checkout** button enables a consumer to use with Visa Checkout on your site. The following rules are specific to the **Visa Checkout** button:

- You can use either a horizontal or stacked implementation with or text between payment options to emphasize and distinguish that users are making a selection:
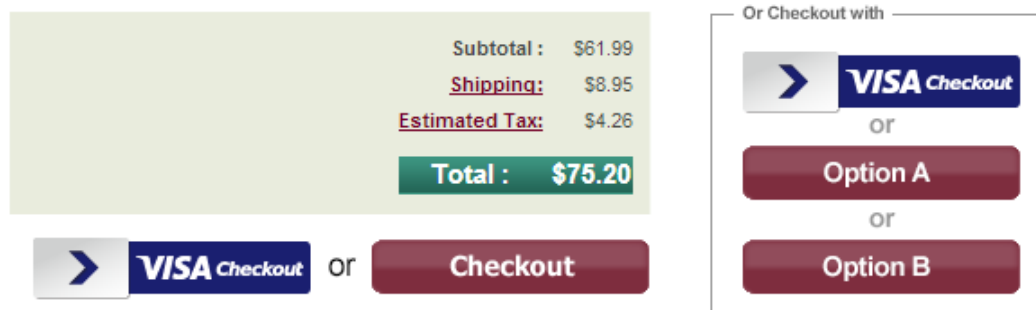
- If you offer guest checkout, pre-fill any account creation form you provide from Visa Checkout after the lightbox closes.
- You should use the standard button unless the background colors on your site are dark, in which case, you can consider using a neutral button to provide greater contrast.

As a best practice, if you require the consumer to sign in to your site, link the consumer's Visa Checkout account to the consumer's account on your site to avoid signing in multiple times by

the consumer during checkout. By linking accounts, you can recognize the consumer's account on your site based on the consumer's successful sign in to Visa Checkout.

About Visa Checkout Acceptance Mark Requirements

You can use the **Visa Checkout** acceptance marks to let consumers know that you accept Visa Checkout on your pages, such as on your payment pages and other pages that display a payment method.

The Visa Checkout acceptance marks are dynamic. They display an image of your customer's default card art with Visa Checkout if a user has previously selected **Remember Me** during sign in.

The following illustration shows the use of the standard acceptance mark, which is static:

Standard Acceptance Mark

The following illustration shows the use of the dynamic acceptance mark, which contains a consumer's card art:

Card Art

Token Branding and User Experience

Tokens cannot be auto-filled/key-entered in a browser. You should work with your Visa Checkout representative to develop consumer-facing messaging about tokens, as needed.

To ensure proper branding and user experience::

• Display the last 4 digits of the account number instead of the last 4 digits of the token.

• Refer to tokens using the name **Digital Account Number** in cases where you must refer to a token.

• Do not display the token expiration date.

• Place token information under **Digital Account Number** and separate token information from other account information, if you must display both the last 4 digits of the token and the token expiration date..

• Advise the consumer to **choose another card** in the event of an error with a token-enabled payment instrument.

Revision History

- Version 2.0, April 29, 2014
- Version 2.1, June 10, 2014
- Version 2.2, July 8, 2014
- Version 2.3, August 5, 2014
- Version 2.4, August 5, 2014
- Version 2.5, October 7, 2014
- Version 2.6, November 11, 2014
- Version 2.7, January 27, 2015
- Version 2.8, March 31, 2015
- Version 2.9, April 28, 2015
- Version 3.4, September 29, 2015
- Version 3.6, January 26, 2016
- Version 3.8, March 22, 2016
- Version 4.1, June 21, 2016
- Version 4.6, January 24, 2017
- Version 5.0 May 2, 2017
- Version 6.4 July 25, 2018
- Version 6.5, August 22, 2018
- Version 6.6, September 26, 2018
- Version 6.7, October 31, 2018
- Version 19.01, January 23, 2019