

# ParadoxLabs Clover Payments: User Manual

Version 1.0 – For Magento® 2.4 – Updated 2024-04-22

## Table of Contents

- ParadoxLabs Clover Payments: User Manual ..... 1
- Installation ..... 3
  - If you purchased from Adobe Commerce/Magento Marketplace ..... 3
  - If you purchased from store.paradoxlabs.com ..... 4
- Updating the Extension ..... 5
  - If you purchased from Magento Marketplace ..... 5
  - If you purchased from store.paradoxlabs.com ..... 5
- Connecting A New Clover Account ..... 6
  - Step 1. Finding the Magento configuration ..... 6
  - Step 2. API Setup ..... 6
- Configuration ..... 11
  - General ..... 11
  - API Setup ..... 11
  - Checkout Settings ..... 12
  - Advanced Settings ..... 13
- Behavior Notes ..... 14
  - User Experience ..... 14
  - Security ..... 14
  - Card Storage ..... 15
- Usage ..... 16
  - Checkout Payment Form ..... 16
  - Order status page ..... 17
  - Customer ‘My Payment Options’ account area ..... 18
  - Admin order form ..... 18
  - Admin order status page ..... 19
  - Admin customer ‘Payment Options’ account area ..... 20
  - Admin transaction info ..... 20
- Frequently Asked Questions & Troubleshooting ..... 22

Is ParadoxLabs Clover Payments PCI Compliance? .....	22
How do I do an online refund from Magento? .....	22
How does this payment method handle currency? .....	22
Technical / Integration Details.....	23
Architecture .....	23
Custom database schema .....	23
Events.....	23
Magento API: REST and SOAP .....	24
Magento API: GraphQL .....	34
How-To: API Checkout Flow.....	41
Support .....	43

## Installation

The installation process differs based on where you purchased our extension.

### If you purchased from Adobe Commerce/Magento Marketplace

**NOTE: You will not be able to install by downloading the extension files from Marketplace.**

The Marketplace download does not include all of the necessary files. You must install using Composer, with the following directions.

#### Step 1: Install

We strongly recommend installing, configuring, and testing all extensions on a development website before installing and using them in production.

If you encounter any problems during this process, please contact [Adobe Commerce Marketplace Support](#).

Note, installing this extension requires familiarity with your server's command line. Ensure your server has composer set up and linked to your Adobe Commerce Marketplace account (including repository repo.magento.com). Then in SSH, from your site root, run the following commands:

```
composer require paradoxlabs/clover:*  
php bin/magento module:enable -c ParadoxLabs-TokenBase ParadoxLabs_Clover  
php bin/magento setup:upgrade
```

If your site is in production mode, you will also need to run these commands to recompile sources:

```
php bin/magento setup:di:compile  
php bin/magento setup:static-content:deploy
```

These commands should load and install the extension packages from the Marketplace repository.

Composer installation is only available for Marketplace purchases.

#### Step 2: Configure

See the configuration section below.

## If you purchased from [store.paradoxlabs.com](https://store.paradoxlabs.com)

**NOTE:** This file upload installation applies **only** to purchases from the ParadoxLabs Store. Marketplace purchases must follow the Marketplace installation directions above.

### Step 1: Upload files

Upload all files within the **upload** folder into the root directory of Magento.

Folder in Download	Folder on Server
/upload/app/	→ /app/

### Step 2: Run Installation

In SSH, from your site root, run the following commands:

```
php bin/magento module:enable -c ParadoxLabs-TokenBase ParadoxLabs_Clover
php bin/magento setup:upgrade
```

These will enable the module, and then run the installation process.

If your site is in production mode, you will also need to run these commands to recompile sources:

```
php bin/magento setup:di:compile
php bin/magento setup:static-content:deploy
```

### Step 3: Configure

See the configuration section below.

## Updating the Extension

All extension updates are free. Just follow these directions to update to the latest version.

### If you purchased from Magento Marketplace

Note, installing/upgrading this extension requires familiarity with your server's command line.

If you installed with composer, you can update using the following commands, in SSH at your site root:

```
composer update paradoxlabs/*  
php bin/magento setup:upgrade
```

This will download and update to the latest extension version compatible with your system.

If your site is in production mode, you will also need to run these commands to recompile sources:

```
php bin/magento setup:di:compile  
php bin/magento setup:static-content:deploy
```

### If you purchased from [store.paradoxlabs.com](https://store.paradoxlabs.com)

#### Step 1: Upload files

Log into your account at [store.paradoxlabs.com](https://store.paradoxlabs.com) and download the latest version.

Open the extension archive and extract it onto your composer.

Upload all files within the **upload** folder into the root directory of Magento.

Folder in Download	Folder on Server
/upload/app/	→ /app/

#### Step 2: Run Update

In SSH, from your site root, run the following commands:

```
php bin/magento setup:upgrade
```

If your site is in production mode, you will also need to run these commands to recompile sources:

```
php bin/magento setup:di:compile  
php bin/magento setup:static-content:deploy
```

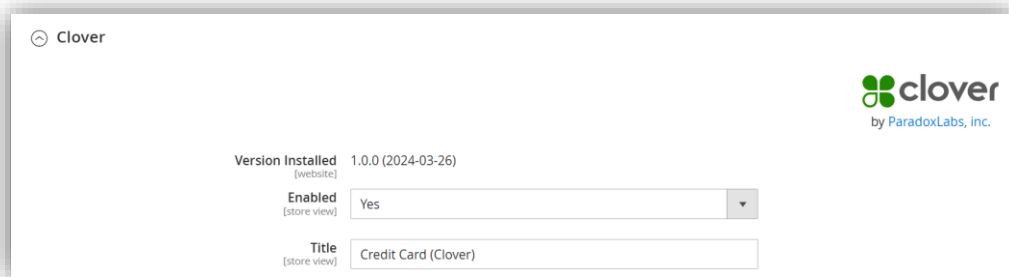
## Connecting A New Clover Account

**Before proceeding:** [Contact Clover](#) to sign up for merchant account if you don't have one already. You will need to go through the account setup and activation process before you can accept real payments.

If you need a sandbox account for testing, you will need to [Create a Developer Account](#).

### Step 1. Finding the Magento configuration

Open your Admin Panel and go to **Admin > Stores > Settings > Configuration > Sales > Payment Methods**. Toward the bottom of the page, you'll find a 'Clover' settings section like this:



When you see this, you're at the right place. This is where you'll enter all of the API credentials, and set additional payment method configuration options. For the top section:

1. Leave 'Enabled' set to No until we're done.
2. If you'd like, change the Title while you're here.

Let's move on to API setup.

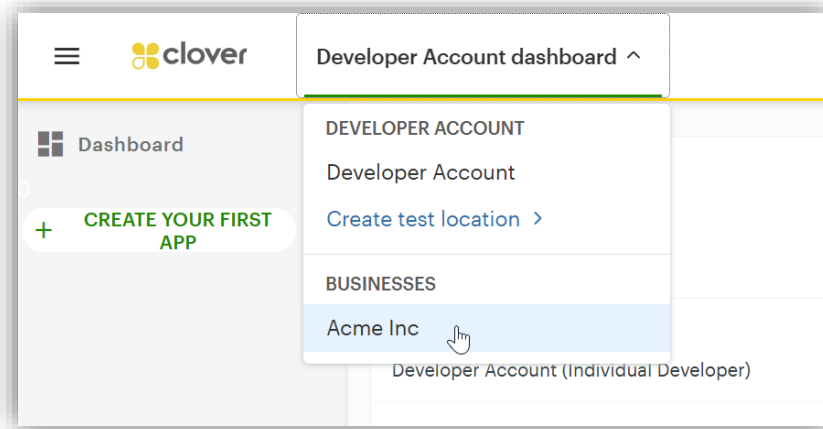
### Step 2. API Setup

The next config section is to connect Magento to your Clover account. You will need the following data:

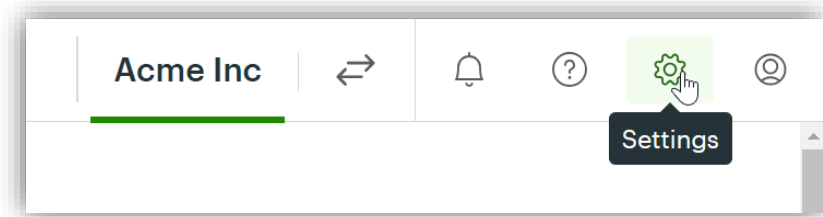
- Merchant ID (MID)
- Public Key
- Private Key

To find these values, first log into your Clover account at <https://www.clover.com> (or <https://www.clover.com/global-developer-home/> for sandbox accounts).

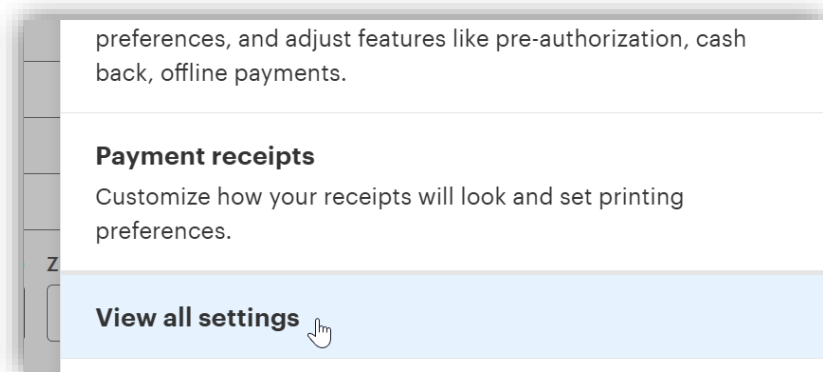
At the page top, click **{your name} dashboard**, and then click into your **Business**.



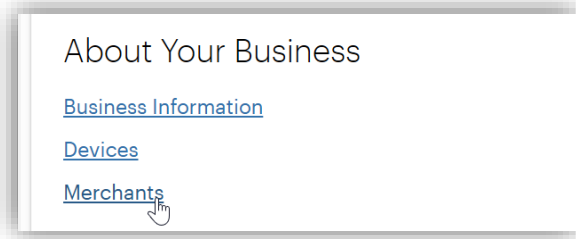
Next, at the top right click the gear icon for **Settings**.



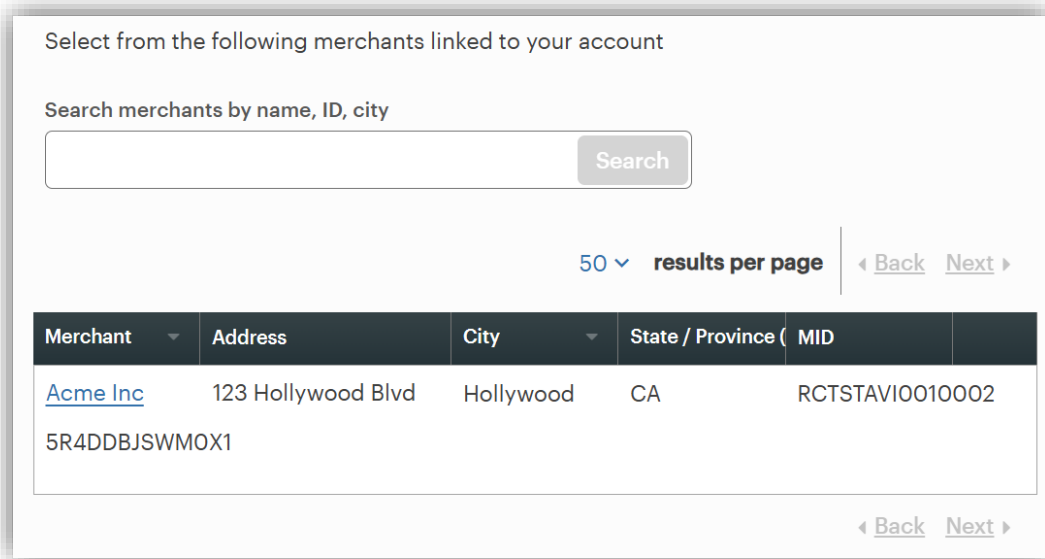
At the bottom, click **View all settings**.



Under **About Your Business**, click **Merchants**.



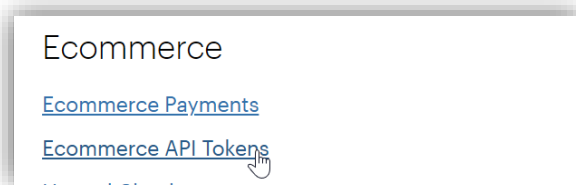
Find the merchant account you want to use for your Magento site on the accounts list. On the right side is your **Merchant ID (MID)**.



Copy the **MID** into your Magento settings.

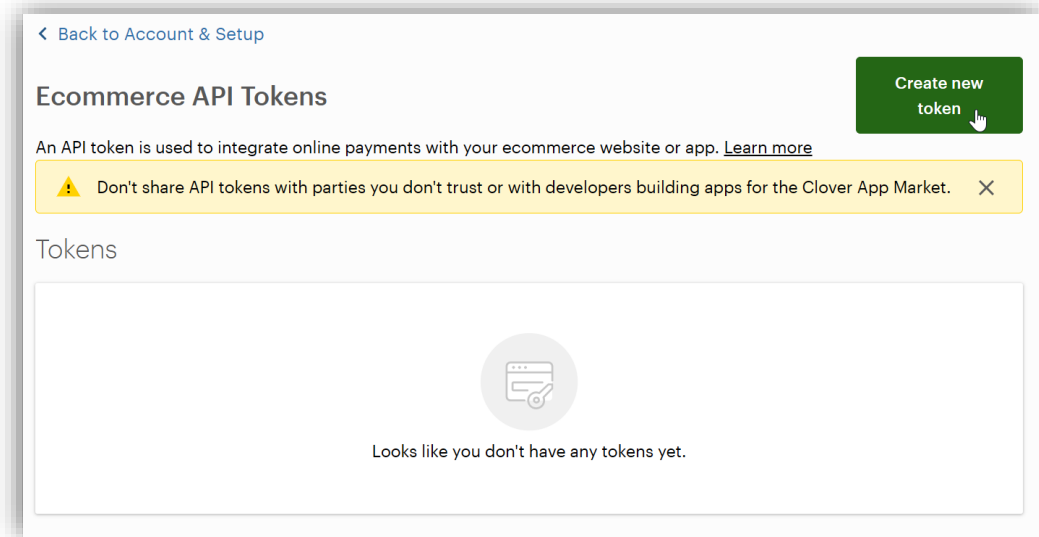
Now go back to the **View all settings** page.

This time, click **Ecommerce API Tokens**:

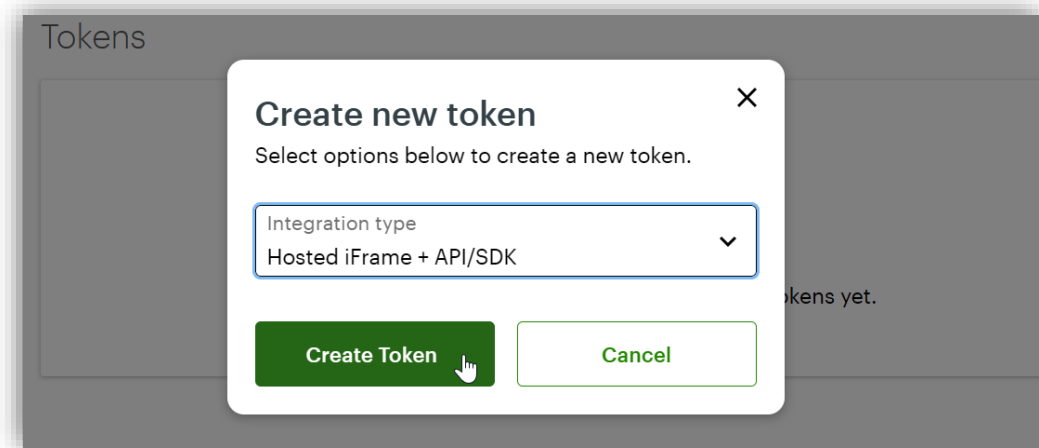


On the Ecommerce API Tokens page, click **Create new token**:

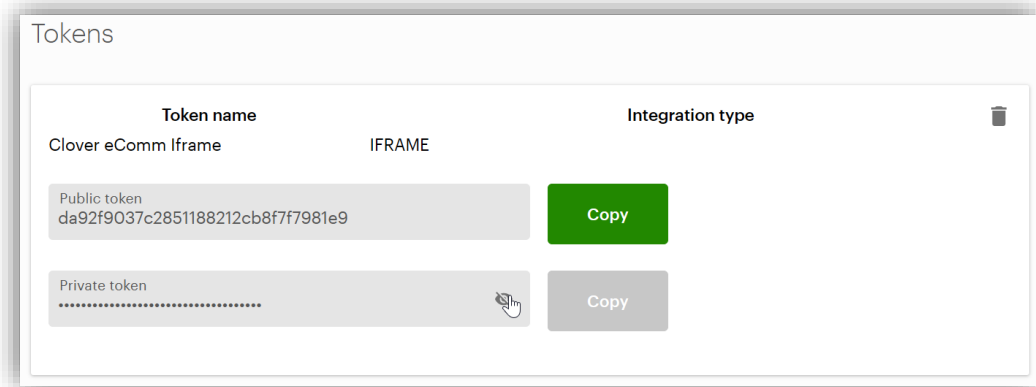




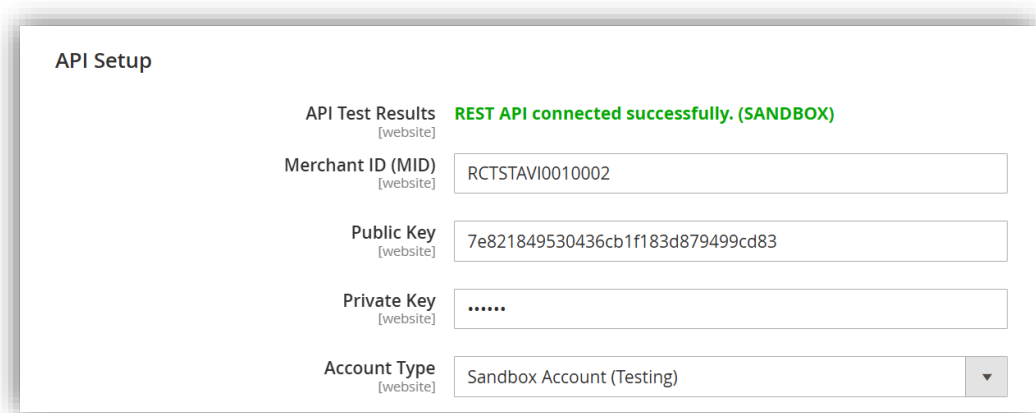
A popup will ask for the integration type. Select **Hosted iFrame + API/SDK**, then **Create Token**.



The next page will give your API keys. Copy the Public token value into Magento settings as your **Public Key**. Then come back, click the eye to show the private token, and copy that into Magento settings as your **Private Key**.



At this point, all of your API Setup fields should be completed. Set your **Account Type** to Sandbox or Production, based on the **API Key Type** you just created.



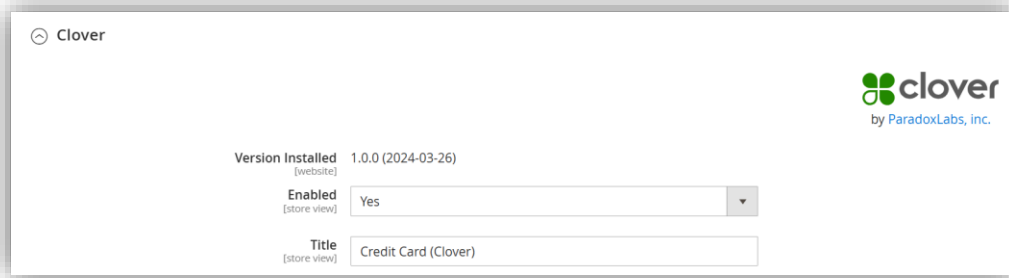
Once you've entered all of the values into Magento, click **Save Config**. Once the page reloads, you should have a green message: **REST API connected successfully**. If you get a red message, verify that you entered the values correctly and didn't miss any part of them.

**Congratulations!** You've completed connecting your new Magento extension to your Clover account. Please see the next section for information on the rest of the configuration options in Magento.

## Configuration

Open your Admin Panel and go to **Admin > Stores > Settings > Configuration > Sales > Payment Methods**. Toward the bottom of the page, you'll find a 'Clover' settings section like the below.

### General



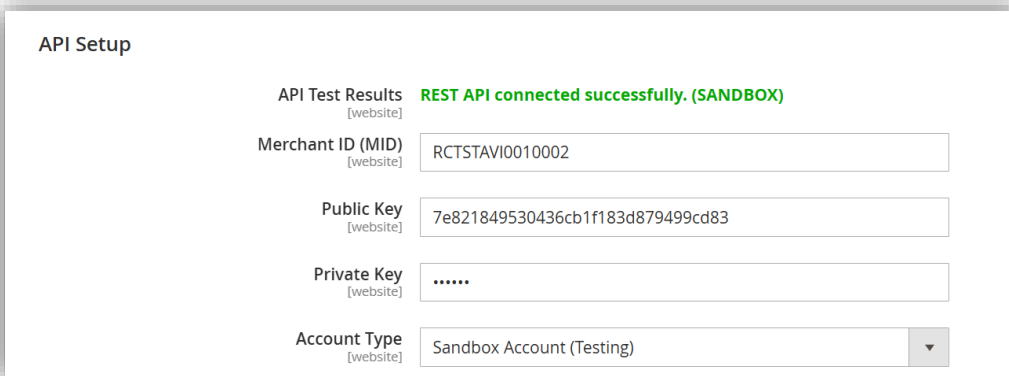
The screenshot shows the 'Clover' configuration page. At the top left is a back arrow and the text 'Clover'. At the top right is the Clover logo with the text 'by ParadoxLabs, Inc.'. Below the logo, there are three settings:

- Version Installed** [website]: 1.0.0 (2024-03-26)
- Enabled** [store view]: Yes (dropdown menu)
- Title** [store view]: Credit Card (Clover) (text input field)

- **Version Installed:** This tells you the version of our extension currently installed on your website. Please include this in any support requests.
- **Enable:** Yes to enable the payment method. If disabled, you will still be able to invoice/refund existing orders, but it will not show up as a payment option during checkout.
- **Title:** This controls the payment option label on checkout and order status pages.

### API Setup

This section connects your store to Clover for payment processing: Authorizations, captures, refunds, etc. You won't be able to process payments without it.



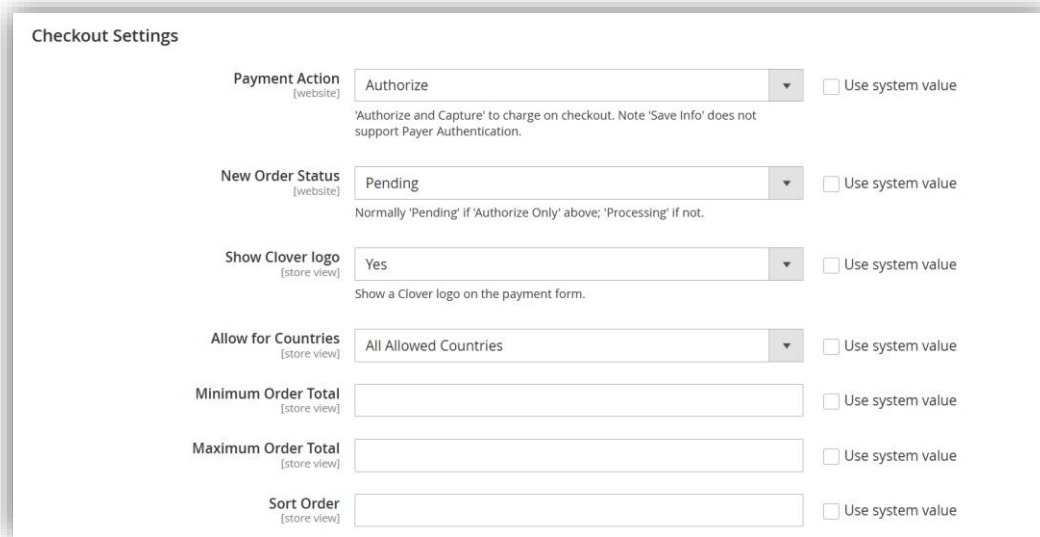
The screenshot shows the 'API Setup' configuration page. At the top left is the text 'API Setup'. Below it, there are several fields and a message:

- API Test Results** [website]: REST API connected successfully. (SANDBOX) (green text)
- Merchant ID (MID)** [website]: RCTSTAVI0010002 (text input field)
- Public Key** [website]: 7e821849530436cb1f183d879499cd83 (text input field)
- Private Key** [website]: ..... (text input field)
- Account Type** [website]: Sandbox Account (Testing) (dropdown menu)

- **API Test Results:** Once you've completed and saved these settings, we will connect to Clover to verify that the connection works successfully. If we cannot connect to Clover, or your credentials are incorrect, this will tell you with a red message. Correct the error, then reload the page and it should show **REST API connected successfully.**
- **Merchant ID, Public Key, Private Key:** Please see Connecting A New Clover Account: [Step 2. API Setup](#) for details on finding and configuring these API credentials.

- **Account Type:** If Sandbox, all requests will be made to Clover’s test APIs, and no actual payments will be processed. If you want to test, you must have a sandbox Merchant ID and API Key. You can create them at: <https://developer.fiserv.com>

## Checkout Settings



**Checkout Settings**

**Payment Action** (website):   Use system value  
Authorize and Capture' to charge on checkout. Note 'Save Info' does not support Payer Authentication.

**New Order Status** (website):   Use system value  
Normally 'Pending' if 'Authorize Only' above; 'Processing' if not.

**Show Clover logo** (store view):   Use system value  
Show a Clover logo on the payment form.

**Allow for Countries** (store view):   Use system value

**Minimum Order Total** (store view):   Use system value

**Maximum Order Total** (store view):   Use system value

**Sort Order** (store view):   Use system value

- **Payment Action:** Choose from the following options.
  - **Save info (do not authorize):** This will require customers to enter a credit card on checkout, and store that credit card in Clover. The credit card will be validated in the process. No funds will be captured or held from the credit card upon checkout. Invoicing the order will perform a standalone authorize+capture transaction, but is not guaranteed to go through (funds may not be available).
  - **Authorize:** This will authorize the order amount upon checkout, allowing for manual invoicing and capture of the funds later. The authorized funds will be held (reserved) for about a week depending on your processor. If you do not invoice within that time, the authorization will expire, and invoicing will perform a standalone authorize+capture transaction instead (which is not guaranteed to go through).
  - **Authorize and Capture:** This will capture all funds immediately when an order is placed. Payment processors strongly recommend not capturing funds unless/until you are within three days of fulfilling/shipping the order.
- **New Order Status:** Set this to your desired initial status for orders paid via Clover. Default Magento behavior is 'Pending' for Authorize Only, and 'Processing' for Authorize and Capture.
- **Show Clover logo:** If yes, checkout will display a 'Clover' logo above the payment form.
- **Allowed for Countries:** This setting allows you to limit which countries are allowed to use this payment method.
- **Minimum Order Total:** This setting allows you to set a minimum order value for the payment option. For instance, set to 5 to only allow credit card checkout for orders of \$5 or more.
- **Maximum Order Total:** This setting allows you to set a maximum order value for the payment option. For instance, set to 1000 to only allow credit card checkout for orders of \$1000 or less.

- **Sort Order:** This setting allows you to change the order of payment options on checkout. Enter a number for this and all other payment methods according to the order you want them to display in.

## Advanced Settings

**Advanced Settings**

<p><b>Form Style</b> <small>[store view]</small></p>	<pre>{   "body": {"fontFamily": "Roboto, Open Sans, sans-serif",     "fontSize": "16px"},   "input": {"background": "transparent", "padding": "9px",     "height": "34px", "fontSize": "16px", "border": "1px solid #ccc", "borderRadius": "4px", "lineHeight": "1.28"} }</pre> <p>Use this to change the font and color of the Clover credit card fields. Must be valid JSON. <a href="#">See documentation for details.</a></p>	<input checked="" type="checkbox"/> Use system value
<p><b>Allow cards to not be stored</b> <small>[website]</small></p>	<p>Yes</p> <p>If yes, customers can choose whether to save their credit card during checkout.</p>	<input checked="" type="checkbox"/> Use system value
<p><b>Auto-select 'save for next time'</b> <small>[website]</small></p>	<p>Yes</p> <p>If yes, will be selected by default during checkout.</p>	<input checked="" type="checkbox"/> Use system value
<p><b>Reauthorize on Partial Invoice</b> <small>[website]</small></p>	<p>Yes</p> <p>If yes, when you create a partial invoice, we will reauthorize any outstanding balance on the order. This helps guarantee funds, but can cause multiple holds on the card until transactions settle.</p>	<input checked="" type="checkbox"/> Use system value

- **Form Style:** This controls styling of the hosted payment form fields. Use this to change colors, background, font, borders, etc. to fit your checkout form. Most basic CSS styles and selectors are allowed. The styles must be formatted as JSON, with each selector as key for an array of CSS properties and values. The CSS properties should be camelCase, like the default styles. If your styles are not valid JSON, the form will have no borders or colors at all.
- **Allow cards to not be stored:** If yes, customers will have a 'Save for next time' checkbox on checkout. If no, logged in customers will see a message instead: *"For your convenience, this data will be stored securely by our payment processor."* Guests will never be given the option to store a credit card. Note that all cards are always stored in Clover, regardless of this setting or the customer's choice. This is necessary for payment processing. If the order is placed as a guest, or the customer chooses to not save their card, it will be automatically purged from all systems 120 days after its last use. This ensures the info is available for edits, captures, and refunds, but respects the customer's wishes. If a card is 'not saved', it will not display under the customer's saved credit cards (Account > My Payment Data), nor will it be selectable during checkout. Note that as an admin, order 'edit' or 'reorder' will bypass this, always allowing reuse of the original payment info (unless it was since purged).
- **Auto-select 'save for next time':** If yes, the 'save this card for next time' checkbox will be checked by default. If no, customers will have to explicitly select it to store and reuse their card.
- **Reauthorize on Partial Invoice:** If yes, and you invoice part of an order, a new authorization will be created for the outstanding order balance (if any). This helps guarantee funds. Any failure during reauthorization is ignored.

This concludes the payment method's configuration options.

## Behavior Notes

For the most part this is a standard Magento payment method, with the addition of the advanced Stored Card functionality and everything related to that. However, there are some things worth note:

### User Experience

All Clover credit card forms in the extension use an iframe hosted form. That greatly enhances security, but limits how much control you have over the payment form. For instance, the credit card form includes a 'Zipcode' field that is redundant to the billing address. There is no way to prefill or disable that field. We have to include it for transactions to process correctly.

The screenshot shows a payment form for Clover credit cards. At the top, there is a radio button selected for 'Credit Card (Clover)'. Below this, the form is titled 'Payment Information' and features the Clover logo. A dropdown menu labeled 'Add new card' is present. The main form fields include:
 

- 'Credit Card Number' with a sub-label 'Card Number' and a text input field.
- 'Expiration' with a sub-label 'MM/YY' and a text input field.
- 'Security Code' with a sub-label 'CVV' and a text input field.
- 'Zipcode' with a sub-label 'Zip' and a text input field.

 At the bottom left, there is a checked checkbox for 'Save for next time'. A blue 'Place Order' button is located at the bottom right of the form.

### Security

The Iframe Hosted Checkout form will always be active. There is no option to turn it off and revert to 'normal' credit card input. All credit card details (CC number, expiration date, CCV) are entered into an iframe hosted by Clover, and transmitted directly to Clover's servers. At no time will credit card numbers ever touch your server for this payment method. This is ideal for PCI compliance.

Some data about credit cards is kept in your database. This includes the card type, last 4 digits, expiration date, and identifiers, among other data. All of this data is available via the Clover API, and no part of it is considered 'Confidential Cardholder Data' in the context of PCI compliance.

If you intend to collect payments from other sources using the Magento API, you will need to use a Clover API to tokenize the credit card, then store that card and token in Magento as a TokenBase Card, via the REST or GraphQL API. See the Technical / Integration Details section at the end for more information. This payment method is incapable of processing a transaction or storing a card using raw credit card details; it will not accept them under any circumstances.

## Card Storage

Credit cards will always be stored in Clover to allow advanced functionality, even if the customer chooses not to save upon checkout. If the customer chooses not to save, or is a guest, the card will not show up for reuse within Magento. In this case, the card will be automatically purged from Magento after 120 days past its last use. This ensures the info is available for edits, captures, and refunds, but respects the customer' wishes.

There is no automatic card duplicate prevention: If a customer enters the same credit card as a new card 3 times, it will be stored within Clover 3 times. There is no limit to the number of cards/tokens on a Clover account.

## Usage

There isn't much to using this extension in practice: It's a standard Magento payment method, and all interfaces should be self-explanatory. Here's a rundown of what you get:

### Checkout Payment Form

The frontend payment form lets you choose/enter billing address and credit card. You can choose an existing card (if any) from the dropdown, or to add a new one.

**LUMA**

Shipping **Review & Payments**

Select a new payment method

Credit Card (Clover)

Payment Information clover

Add new card

Credit Card Number  
4111 1111 1111 1111

Expiration: 05/29 Security Code: 999

Zipcode: 90210

Save for next time

**Place Order**

Order Summary	
Cart Subtotal	\$105.00
Shipping Flat Rate - Fixed	\$10.00
Tax	\$8.00
<b>Order Total</b>	<b>\$115.00</b>
2 Items in Cart	

Ship To:

Veronica Costello  
6145 Honey Bluff Parkway

Credit card type is detected automatically, and any format errors are immediately displayed below the input field.

Credit Card Number

4111 1111 1111 1110

Card number is invalid

Expiration: 05/29 Security Code: 999

Zipcode: 90210

If the customer has stored cards, their most recent one will be selected by default:




Select a new payment method

Credit Card (Clover)

Payment Information

American Express XXXX-0005



[Place Order](#)

**Order Summary**

Cart Subtotal	\$105.00
Shipping Flat Rate - Fixed	\$10.00
Tax	\$8.00
<b>Order Total</b>	<b>\$115.00</b>

## Order status page

My Account

**My Orders**

My Downloadable Products

My Wish List

---

Address Book

Account Information

Stored Payment Methods

---

My Product Reviews

Newsletter Subscriptions

My Payment Options

My Subscriptions

---

Compare Products

You have no items to compare.

---

My Wish List

You have no items in your wish list.

### Order # 000001443 PENDING

March 19, 2024

[Reorder](#) [Print Order](#)

Items Ordered

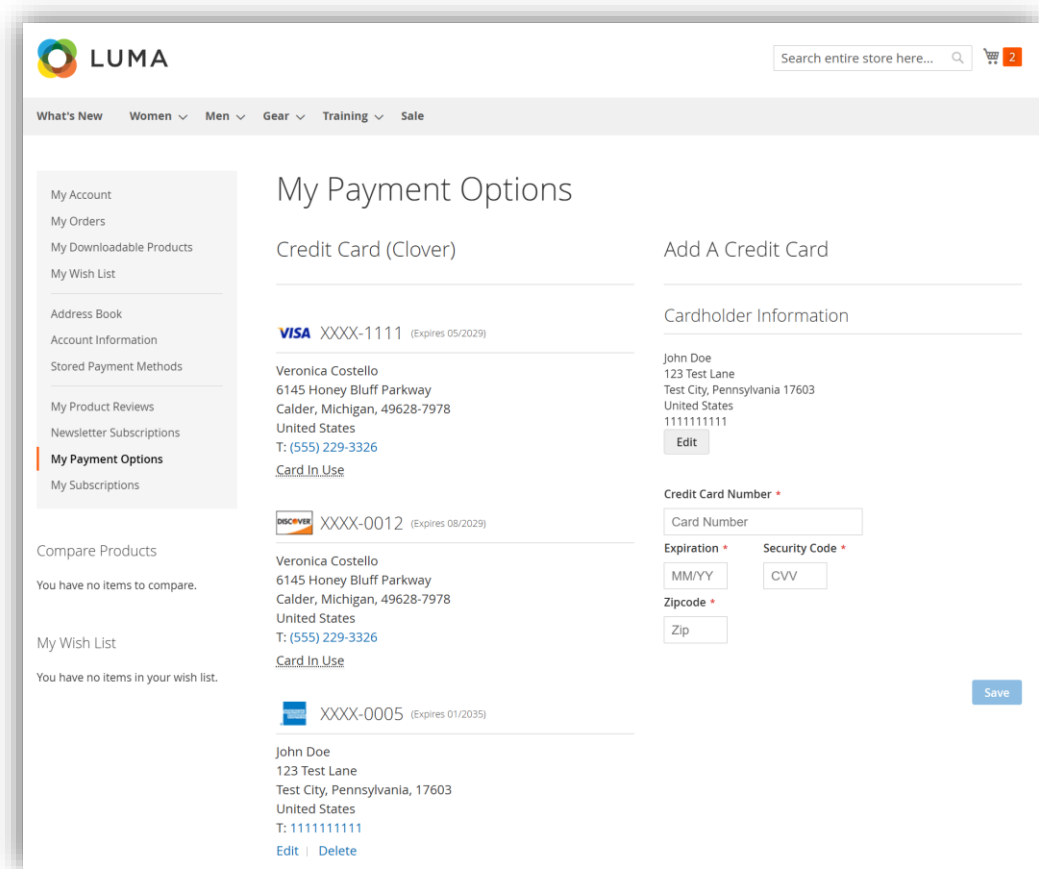
Product Name	SKU	Price	Qty	Subtotal
Sprite Foam Yoga Brick	24-WG084	\$5.00	Ordered: 1	\$5.00
Subtotal				\$5.00
Shipping & Handling				\$5.00
<b>Grand Total</b>				<b>\$10.00</b>

Order Information

<b>Shipping Address</b>	<b>Shipping Method</b>	<b>Billing Address</b>	<b>Payment Method</b>
John Doe ParadoxLabs 8 N Queen St Lancaster, Pennsylvania, 17603 United States T: 123-456-0000	Flat Rate - Fixed	Veronica Costello 6145 Honey Bluff Parkway Calder, Michigan, 49628-7978 United States T: (555) 229-3326	Credit Card (Clover)
			<b>Credit Card Type</b> Discover
			<b>Credit Card Number</b> XXXX-0012

## Customer 'My Payment Options' account area

The My Payment Options section allows customers to see their stored cards, add, edit, and delete.



Like on checkout, the address must be entered and confirmed before payment info can be entered. This means all card adding/editing is a two step process.

When editing a card, the full payment data will have to be reentered for any changes.

Note that cards associated with an open (uncaptured) order cannot be edited or deleted. They will display a 'Card In Use' message in place of the buttons. As soon as all orders paid by the card are completed, the 'Edit' and 'Delete' buttons will appear.

To prevent abuse, the Payment Data section will only be available to customers after they have placed a successful order. If a customer attempts to access the page before then, they'll be redirected to the Account Dashboard with the message, "My Payment Data will be available after you've placed an order." Also to prevent abuse, if a customer receives errors trying to save a card five times, they will be blocked from access for 24 hours with the message, "My Payment Data is currently unavailable. Please try again later." Both of these behaviors can be adjusted or disabled if necessary; please contact us if you have a problem.


## Admin order form

The admin form has the same options and behavior as frontend checkout.

### Payment & Shipping Information

#### Payment Method

Credit Card (Clover)



Add new card

**Credit Card Number \***

Card Number

**Expiration \***      **Security Code \***

MM/YY      CVV

**Zipcode \***

Zip

Save for next time

### Admin order status page

The admin panel shows extended payment info after placing an order, including transaction ID. This info is not visible to the customer.

#### Address Information

<p><b>Billing Address</b> <a href="#">Edit</a></p> <p>Veronica Costello 6145 Honey Bluff Parkway Calder, Michigan, 49628-7978 United States T: (555) 229-3326</p>	<p><b>Shipping Address</b> <a href="#">Edit</a></p> <p>John Doe Company Name 64 Strawberry Dr Beverly Hills Los Angeles, California, 90210 United States T: 123-456-0000</p>
---	--

---

#### Payment & Shipping Method

<p><b>Payment Information</b></p> <p>Credit Card (Clover)</p> <p>Credit Card Type: <span style="float: right;">Visa</span></p> <p>Credit Card Number: <span style="float: right;">XXXX-1111</span></p> <p>Transaction ID: <span style="float: right;">T0KDv8FHx2SVM</span></p> <p>AVS Response: <span style="float: right;">Pass</span></p> <p>The order was placed using USD.</p>	<p><b>Shipping &amp; Handling Information</b></p> <p><b>Flat Rate - Fixed \$5.00</b></p>
--	--

## Admin customer 'Payment Options' account area

Viewing a customer, you will see an added 'Payment Options' tab. This shows all of the same information with all of the same functionality as the equivalent frontend section. You can use this to view, add, edit, and delete your customers' stored Clover payment data.

The screenshot shows the 'Payment Options' section for a customer named Veronica Costello. On the left is a sidebar with navigation options like 'CUSTOMER VIEW', 'ACCOUNT INFORMATION', 'ADDRESSES', 'ORDERS', 'SHOPPING CART', 'NEWSLETTER', 'BILLING AGREEMENTS', 'PRODUCT REVIEWS', 'WISH LIST', and 'PAYMENT OPTIONS'. The main content area displays a list of stored credit cards:

- VISA XXXX-1111 (Expires 12/2024)**: Veronica Costello, 6145 Honey Bluff Parkway, Calder, Michigan, 49628-7978, United States, T: (555) 229-3326. Card In Use.
- VISA XXXX-1111 (Expires 05/2029)**: Veronica Costello, 6145 Honey Bluff Parkway, Calder, Michigan, 49628-7978, United States, T: (555) 229-3326. Card In Use.
- VISA XXXX-1111 (Expires 05/2029)**: Veronica Costello, 6145 Honey Bluff Parkway, Calder, Michigan, 49628-7978, United States, T: (555) 229-3326. Card In Use.
- DISCOVER XXXX-0012 (Expires 08/2029)**: Veronica Costello, 6145 Honey Bluff Parkway, Calder, Michigan, 49628-7978. Card In Use.

On the right, there is an 'Add A Credit Card' form with the following fields:

- Cardholder Information**: John Doe, 123 Test Lane, Test City, Pennsylvania 17603, United States, 1111111111. Edit button.
- Credit Card Number**: 4111 1111 1111 1111. VISA logo.
- Expiration**: 09/29. **Security Code**: 999.
- Zipcode**: 99999.
- Buttons: Save, Cancel.

## Admin transaction info

Viewing an order, you can also see full transaction info from the 'Transactions' tab.

The screenshot shows the 'Transactions' tab for order #000001436. The page includes a search bar, a 'Reset Filter' button, and a notification that 3 records were found. The table below displays the transaction data:

ID	Order ID	Transaction ID	Parent Transaction ID	Payment Method	Transaction Type	Closed	Created
From							From
To							To
1217	000001436	9VQXX9H7D0ZP	YTF10P2ZCB394	Credit Card (Clover)	Refund	Yes	Mar 19, 2024, 12:29:41 PM
1216	000001436	YTF10P2ZCB394	8BFJVDTF0A6P	Credit Card (Clover)	Capture	Yes	Mar 19, 2024, 12:28:37 PM
1215	000001436	8BFJVDTF0A6P		Credit Card (Clover)	Authorization	Yes	Mar 19, 2024, 12:20:49 PM

Click into a transaction, and you'll see all of the raw transaction data returned by the Clover API. This same data is accessible from the transaction record in code.

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
-

## #YTF10P2ZCB394

### Transaction Data

Transaction ID	YTF10P2ZCB394
Parent Transaction ID	<a href="#">R8FJVDTDF0A6P</a>
Order ID	<a href="#">000001436</a>
Transaction Type	capture
Is Closed	Yes
Created At	Mar 19, 2024, 12:28:37 PM

### Child Transactions

1 records found

ID	Order ID	Transaction ID	Payment Method	Transaction Type	Closed	Created
1217	000001436	9VQXXY9H7D0ZP	Credit Card (Clover)	Refund	Yes	Mar 19, 2024

### Transaction Details

Key	Value
id	YTF10P2ZCB394
amount	1041
tax_amount	41
payment_method_details	card
currency	usd
created	1710865717423
captured	1
ref_num	407900500050
auth_code	OK8631
order	S89AAFV6TFEM
outcome.network_status	approved_by_network
outcome.type	authorized
paid	1
status	succeeded

## Frequently Asked Questions & Troubleshooting

Please search our solution directory for the latest answers to common questions and issues:

<https://paradoxlabs.freshdesk.com/support/solutions>

If your question is not answered there, open a support ticket and we'll help you out.

### Is ParadoxLabs Clover Payments PCI Compliance?

PCI compliance is a complex and multifaceted issue, covering every aspect of your business. We can't guarantee that your business is PCI-compliant. That depends on your server, passwords, business processes, regular security scans, any other payment methods, and a lot more. What we can tell you is that this extension will not prevent you from being PCI compliant. We don't log confidential cardholder data or do anything else that would bring you under scrutiny.

This extension implements **Hosted Checkout** iframes for all credit card forms, and does not support collecting credit card data by any other means. That makes the ParadoxLabs Clover payment method eligible for **PCI v3.2 Self-Assessment Questionnaire A (SAQ A)**, the simplest possible SAQ form.

Note that you **must** have SSL (TLS) enabled on all checkout and login forms, and that this eligibility **only** applies to this specific payment method. Any other payment methods or credit card handling your business may perform will have its own SAQ eligibility, and may require you to complete a more stringent SAQ form (A-EP or D).

For details on the SAQ types and what eligibility means, see "[Self-Assessment Questionnaire Instructions and Guidelines \(3.2\)](#)" (PDF, by PCI Standards Security Council).

### How do I do an online refund from Magento?

In order to process an 'online' refund through Clover, you have to go to the **invoice** you want to refund, and click the 'Credit Memo' button from there.

If you've done that correctly, at the bottom of the page you should see a button that says 'Refund'.

If you only have one button that says 'Refund Offline', it's because you clicked 'Credit Memo' from the order instead of from the invoice.

The reason for this is that the refund needs to be associated with a particular capture transaction. An order can contain any number of capture transactions, but every capture has an invoice that's directly related. You refund an invoice, not an order.

### How does this payment method handle currency?

Transactions are processed in the base currency for the website customers are purchasing from. Any alternate currencies selected on the frontend are converted to the website's base currency by Magento's built-in currency handling, based on conversion rates provided by a web service configured in your Magento Admin Panel.

Magento allows for a separate base currency per website, if configured to do so. In order to define explicit prices in multiple currencies, each currency must have its own website where it is set as the base currency. All currency setup is configured outside of our payment extension settings.

Your Clover account must allow transactions to be processed in your website's base currency(s).

## Technical / Integration Details

### Architecture

The payment method code for this method is `paradoxlabs_clover`.

`ParadoxLabs_Clover` is the payment method module, built heavily on the `ParadoxLabs-TokenBase` module. `TokenBase` defines a variety of interfaces and architecture for handling tokenization and stored cards cleanly.

The payment method class is `\ParadoxLabs\Clover\Model\Method`. This talks to Clover primarily through `\ParadoxLabs\Clover\Model\Gateway` (via REST API), and stores card metadata in instances of `\ParadoxLabs\Clover\Model\Card`. Each of these extends an equivalent abstract class in `TokenBase`, and implements only the details specific to the Clover API.

Card instances are stored in table `paradoxlabs_stored_card`, and referenced by quotes and orders via a `tokenbase_id` column on tables `quote_payment` and `sales_order_payment`.

In all cases, we strongly discourage any customization by editing our code directly. We cannot support customizations. Use Magento's preferences or plugins to modify behavior if necessary. If your use case isn't covered, let us know.

### Custom database schema

- Added table: `paradoxlabs_stored_card`
- Added column: `quote_payment.tokenbase_id`
- Added column: `sales_order_payment.tokenbase_id`

### Events

- `tokenbase_before_load_payment_info` (`method`, `customer`, `transport`, `info`): Fires before preparing method-specific information for the order payment info blocks (frontend, admin, and emails). Use this to add additional information to the payment info block.
- `tokenbase_after_load_payment_info` (`method`, `customer`, `transport`, `info`): Fires before preparing method-specific information for the order payment info blocks (frontend, admin, and emails). Use this to add additional information to the payment info block, or modify what's there by default.
- `tokenbase_before_load_active_cards` (`method`, `customer`): Fires before loading a customer's available stored cards.
- `tokenbase_after_load_active_cards` (`method`, `customer`, `cards`): Fires after loading a customer's available stored cards. Use this to modify cards available to the customer or admin.

## Magento API: REST and SOAP

This module supports the Magento API via standard interfaces. You can use it to create, read, update, and delete stored cards.

If you have a specific use case in mind that is not covered, please let us know.

You can generate new cards by creating a new TokenBase Card with a valid Clover credit card token (and associated card and address data). To place an order with a stored card, pass that card's hash in as `additional_data` -> `card_id`.

Note that raw credit card numbers (`cc_number`) will not be accepted. You must store the card in Clover via their form SDK to obtain a token, then send the token to create an order. See Clover documentation on how to tokenize a credit card in various circumstances.

The extension's custom REST API requests are detailed below. Some response data has been omitted for brevity.

Create and update (POST, PUT) requests take three objects: `card` with primary card data, `address` with address information, and `additional` for card metadata. In responses, `address` and `additional` will be nested within `card` as `address_object` and `additional_object`. This is done for technical reasons. The data formats differ, and not all fields that are returned can be set via API (EG `in_use`, `label`). This means you cannot take a card record and directly post it back to the API to update.

### Integration / Admin-Authenticated API Endpoints

These API requests allow solutions acting with an admin user login, OAUTH authentication, or token-based authentication to take action on any card in the system. Data and behavior are not limited.

#### *GET /V1/tokenbase/:cardId (get one card by ID)*

Example request:

```
GET /rest/v1/tokenbase/1 HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}
```

Example response:

```
{
  "id": 1,
  "in_use": true,
  "additional_object": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2022",
    "cc_exp_month": "12",
    "cc_bin": "400700"
  },
  "address_object": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
```



```

        "firstname": "John",
        "lastname": "Doe"
    },
    "customer_email": "email@example.com",
    "customer_id": 1,
    "customer_ip": "127.0.0.1",
    "profile_id": null,
    "payment_id": "S4FHQMSC7GG9J",
    "method": "paradoxlabs_clover",
    "hash": "f7d085165acdfa0ea6a0b...770111",
    "active": "1",
    "created_at": "2017-08-03 16:31:54",
    "updated_at": "2017-09-20 14:24:14",
    "last_use": "2017-08-03 16:31:54",
    "expires": "2019-06-30 23:59:59",
    "label": "Visa XXXX-0027"
}

```

### GET /V1/tokenbase/search (get multiple cards, with searchCriteria)

Example request:

```

GET /rest/v1/tokenbase/search?searchCriteria[pageSize]=1 HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}

```

Example response:

```

{
  "items": [
    {
      "id": 1,
      // ... other card info
    }
  ],
  "search_criteria": {
    "filter_groups": [],
    "page_size": 1
  },
  "total_count": 51
}

```

See also: [Search using REST APIs](#) (Magento DevDocs)

### POST /V1/tokenbase (create card)

Example request:

```

POST /rest/v1/tokenbase HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}
Content-Type: application/json

```

```

{
  "card": {
    "customer_email": "email@example.com",
    "customer_id": 1,
    "customer_ip": "",
    "profile_id": null,
    "payment_id": "S4FHQMSC7GG9J",
    "method": "paradoxlabs_clover",
    "active": "1",
    "created_at": "2017-08-03 16:31:54",
    "last_use": "2017-08-03 16:31:54",
    "expires": "2019-06-30 23:59:59"
  },
  "address": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [

```

```

    "123 Test Ln."
  ],
  "company": "",
  "telephone": "111-111-1111",
  "postcode": "17603",
  "city": "Lancaster",
  "firstname": "John",
  "lastname": "Doe",
  "vat_id": ""
},
"additional": {
  "cc_exp_month": "01",
  "cc_exp_year": "2023",
  "cc_last4": "0027",
  "cc_type": "VI",
  "cc_bin": "400700",
  "token": "7f58b566-911d-4307-9e51-758391728c9a"
}
}

```

Example response:

```

{
  "id": 95,
  "in_use": false,
  "additional_object": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_country": "US",
    "cc_bin": "400700"
  },
  "address_object": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe",
  },
  "customer_email": "email@example.com",
  "customer_id": 1,
  "customer_ip": "127.0.0.1",
  "profile_id": null,
  "payment_id": "S4FHQMSC7GG9J",
  "method": "paradoxlabs_clover",
  "hash": "9b83d4683f3d3...2309ccd65b",
  "active": "1",
  "created_at": "2017-09-25 17:41:21",
  "updated_at": "2017-09-25 17:41:21",
  "last_use": "2017-08-03 16:31:54",
  "expires": "2023-01-31 23:59:59",
  "label": "visa xxxx-0027"
}

```

*PUT /V1/tokenbase/:cardId (update card)*

Example request:

```

PUT /rest/v1/tokenbase/1 HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}
Content-Type: application/json

```

```
{
  "card": {
    "id": 1,
    "customer_email": "email@example.com",
    "customer_id": 1,
    "customer_ip": "127.0.0.1",
    "profile_id": null,
    "payment_id": "S4FHQMSC7GG9J",
    "method": "paradoxlabs_clover",
    "hash": "f7d085165acdfa0ea6a0b...770111",
    "active": "1",
    "created_at": "2017-08-03 16:31:54",
    "last_use": "2017-08-03 16:31:54",
    "expires": "2019-06-30 23:59:59"
  },
  "address": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe",
    "vat_id": ""
  },
  "additional": {
    "cc_exp_month": "01",
    "cc_exp_year": "2023",
    "cc_last4": "0027",
    "cc_type": "VI",
    "cc_bin": "400700",
    "token": "7f58b566-911d-4307-9e51-758391728c9a"
  }
}
```

Example response:

```
{
  "id": 1,
  "in_use": false,
  "additional_object": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_bin": "400700"
  },
  "address_object": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe",
  },
  "customer_email": "email@example.com",
  "customer_id": 1,
  "customer_ip": "127.0.0.1",
}
```

```

"profile_id": null,
"payment_id": "S4FHQMSC7GG9J",
"method": "paradoxlabs_clover",
"hash": " f7d085165acdfa0ea6a0b...770111",
"active": "1",
"created_at": "2017-09-25 17:41:21",
"updated_at": "2017-09-25 17:41:21",
"last_use": "2017-08-03 16:31:54",
"expires": "2023-01-31 23:59:59",
"label": "visa xxxx-0027"
}

```

### DELETE /V1/tokenbase/:cardId (delete card by ID)

Example request:

```

DELETE /rest/V1/tokenbase/95 HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}

```

Example response:

```
true
```

## Customer Authenticated API Endpoints

These API requests allow authenticated frontend customers to manage their stored cards. This is intended for headless implementations or app integration where card management needs to be exposed outside of Magento's standard frontend.

Customers will only be able to access and manipulate active cards assigned to their specific customer ID.

Note: These requests are disabled by default. You can enable them at **Admin > Stores > Configuration > Sales > Checkout > ParadoxLabs Payment Module Settings > Enable public API**. Only enable this if you use them.

### GET /V1/tokenbase/mine/:cardHash (get one card by hash)

Example request:

```

GET /rest/V1/tokenbase/mine/50b8e326b012e793957215c0361afc4b52434b26 HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}

```

Example response:

```

{
  "id": 1,
  "in_use": true,
  "additional_object": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_bin": "400700"
  },
  "address_object": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
  }
}

```

```

        "postcode": "17603",
        "city": "Lancaster",
        "firstname": "John",
        "lastname": "Doe"
    },
    "customer_email": "email@example.com",
    "customer_id": 1,
    "customer_ip": "127.0.0.1",
    "profile_id": null,
    "payment_id": "S4FHQMSC7GG9J",
    "method": "paradoxlabs_clover",
    "hash": "50b8e326b012e793957215c0361afc4b52434b26",
    "active": "1",
    "created_at": "2017-08-03 16:31:54",
    "updated_at": "2017-09-20 14:24:14",
    "last_use": "2017-08-03 16:31:54",
    "expires": "2023-01-31 23:59:59",
    "label": "visa xxxx-0027"
}

```

### GET /V1/tokenbase/mine/search (get multiple cards, with searchCriteria)

Example request:

```

GET /rest/v1/tokenbase/mine/search?searchCriteria[pageSize]=3 HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}

```

Example response:

```

{
  "items": [
    {
      "id": 1,
      // ... other card info
    }
  ],
  "search_criteria": {
    "filter_groups": [],
    "page_size": 3
  },
  "total_count": 5
}

```

See also: [Search using REST APIs](#) (Magento DevDocs)

### POST /V1/tokenbase/mine (create card)

Example request:

```

POST /rest/v1/tokenbase/mine HTTP/1.1
Host: {host}
Content-Type: application/json
Authorization: Bearer {api_key}

{
  "card": {
    "customer_email": "email@example.com",
    "customer_id": 1,
    "customer_ip": "127.0.0.1",
    "profile_id": null,
    "payment_id": "S4FHQMSC7GG9J",
    "method": "paradoxlabs_clover",
    "active": "1"
  },
  "address": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ]
  }
}

```

```

    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe",
    "vat_id": ""
  },
  "additional": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_bin": "400700",
    "token": "7f58b566-911d-4307-9e51-758391728c9a"
  }
}

```

Example response:

```

{
  "id": 95,
  "in_use": false,
  "additional_object": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_bin": "400700"
  },
  "address_object": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe",
  },
  "customer_email": "email@example.com",
  "customer_id": 1,
  "customer_ip": "127.0.0.1",
  "profile_id": null,
  "payment_id": "S4FHQMSC7GG9J",
  "method": "paradoxlabs_clover",
  "hash": "9b83d4683f3d3...2309ccd65b",
  "active": "1",
  "created_at": "2017-09-25 17:41:21",
  "updated_at": "2017-09-25 17:41:21",
  "last_use": "2017-08-03 16:31:54",
  "expires": "2023-01-31 23:59:59",
  "label": "Visa xxxx-0027"
}

```

**PUT /V1/tokenbase/mine/:cardHash (update card by hash)**

Example request:

```

PUT /rest/v1/tokenbase/mine/50b8e326b012e793957215c0361afc4b52434b26 HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}
Content-Type: application/json

{
  "card": {
    "id": 1,
    "customer_email": "email@example.com",

```

```

"customer_id": 1,
"customer_ip": "127.0.0.1",
"profile_id": null,
"payment_id": "S4FHQMSC7GG9J",
"method": "paradoxlabs_clover",
"hash": "50b8e326b012e793957215c0361afc4b52434b26",
"active": "1",
"expires": "2023-01-31 23:59:59"
},
"address": {
  "region": {
    "region_code": "PA",
    "region": "Pennsylvania",
    "region_id": 51
  },
  "region_id": 51,
  "country_id": "US",
  "street": [
    "123 Test Ln."
  ],
  "company": "",
  "telephone": "111-111-1111",
  "postcode": "17603",
  "city": "Lancaster",
  "firstname": "John",
  "lastname": "Doe",
  "vat_id": ""
},
"additional": {
  "cc_type": "VI",
  "cc_last4": "0027",
  "cc_exp_year": "2023",
  "cc_exp_month": "01",
  "cc_bin": "400700",
  "token": "7f58b566-911d-4307-9e51-758391728c9a"
}
}

```

#### Example response:

```

{
  "id": 1,
  "in_use": false,
  "additional_object": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_bin": "400700"
  },
  "address_object": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe",
  },
  "customer_email": "email@example.com",
  "customer_id": 1,
  "customer_ip": "127.0.0.1",
  "profile_id": null,
  "payment_id": "S4FHQMSC7GG9J",
  "method": "paradoxlabs_clover",
  "hash": "50b8e326b012e793957215c0361afc4b52434b26",
  "active": "1",
  "created_at": "2017-09-25 17:41:21",
  "updated_at": "2017-09-25 17:41:21",
}

```

```

    "last_use": "2017-08-03 16:31:54",
    "expires": "2023-01-31 23:59:59",
    "label": "Visa xxxx-0027"
}

```

### *DELETE /V1/tokenbase/mine/:cardHash (delete card by hash)*

Example request:

```

DELETE /rest/V1/tokenbase/mine/50b8e326b012e793957215c0361afc4b52434b26 HTTP/1.1
Host: {host}
Authorization: Bearer {api_key}

```

Example response:

```
True
```

## Guest API Endpoints

These API requests allow unauthenticated frontend guest users to add and fetch an individual stored card. This is intended for headless implementations or app integration where card management needs to be exposed outside of Magento's standard frontend. Guests are not able to list, edit, delete, or reuse stored cards, so no API requests are exposed for those actions.

Guests will only be able to access and manipulate active cards, by hash, not assigned to any customer ID.

Note: These requests are disabled by default. You can enable them at **Admin > Stores > Configuration > Sales > Checkout > ParadoxLabs Payment Module Settings > Enable public API**. Only enable this if you use them.

### *GET /V1/tokenbase/guest/:cardHash (get one card by hash)*

Example request:

```

GET /rest/V1/tokenbase/guest/50b8e326b012e793957215c0361afc4b52434b26 HTTP/1.1
Host: {host}

```

Example response:

```

{
  "id": 1,
  "in_use": true,
  "additional_object": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_bin": "400700"
  },
  "address_object": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe"
  },
  "customer_email": "email@example.com",
  "customer_id": 0,
}

```



```

"customer_ip": "127.0.0.1",
"profile_id": null,
"payment_id": "S4FHQMSC7GG9J",
"method": "paradoxlabs_clover",
"hash": "50b8e326b012e793957215c0361afc4b52434b26",
"active": "1",
"created_at": "2017-08-03 16:31:54",
"updated_at": "2017-09-20 14:24:14",
"last_use": "2017-08-03 16:31:54",
"expires": "2023-01-31 23:59:59",
"label": "Visa XXXX-0027"
}

```

### POST /v1/tokenbase/guest (create card)

Example request:

```

POST /rest/v1/tokenbase/guest HTTP/1.1
Host: {host}
Content-Type: application/json

{
  "card": {
    "customer_email": "email@example.com",
    "customer_id": 0,
    "customer_ip": "127.0.0.1",
    "profile_id": null,
    "payment_id": "S4FHQMSC7GG9J",
    "method": "paradoxlabs_clover",
    "active": "1"
  },
  "address": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    },
    "region_id": 51,
    "country_id": "US",
    "street": [
      "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe",
    "vat_id": ""
  },
  "additional": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_bin": "400700",
    "token": "7f58b566-911d-4307-9e51-758391728c9a"
  }
}

```

Example response:

```

{
  "id": 95,
  "in_use": false,
  "additional_object": {
    "cc_type": "VI",
    "cc_last4": "0027",
    "cc_exp_year": "2023",
    "cc_exp_month": "01",
    "cc_bin": "400700"
  },
  "address_object": {
    "region": {
      "region_code": "PA",
      "region": "Pennsylvania",
      "region_id": 51
    }
  }
}

```

```

    },
    "region_id": 51,
    "country_id": "us",
    "street": [
        "123 Test Ln."
    ],
    "company": "",
    "telephone": "111-111-1111",
    "postcode": "17603",
    "city": "Lancaster",
    "firstname": "John",
    "lastname": "Doe",
},
"customer_email": "email@example.com",
"customer_id": 0,
"customer_ip": "127.0.0.1",
"profile_id": null,
"payment_id": "s4FHQMSC7GG9J",
"method": "paradoxlabs_clover",
"hash": "9b83d4683f3d3...2309ccd65b",
"active": "1",
"created_at": "2017-09-25 17:41:21",
"updated_at": "2017-09-25 17:41:21",
"last_use": "2017-08-03 16:31:54",
"expires": "2023-01-31 23:59:59",
"label": "Visa xxxx-0027"
}

```

## Magento API: GraphQL

For Magento 2.3.1+, this extension supports the GraphQL API for all customer card management. This is intended for PWA and headless implementations where card management needs to be exposed outside of Magento's standard frontend.

Customers will only be able to access and manipulate active cards assigned to their specific customer ID.

Guests will only be able to access and manipulate active cards, by hash, not assigned to any customer ID.

Note: These requests are disabled by default. You can enable them at **Admin > Stores > Configuration > Sales > Checkout > ParadoxLabs Payment Module Settings > Enable public API**. Only enable this if you use them.

We recommend using the Chrome [Altair GraphQL Client browser extension](#) for browsing your store's GraphQL schema and testing API requests.

### Queries

*tokenBaseCards(hash: String): [TokenBaseCard]*

Get the current customer's stored card(s), if any. Takes a card hash (optional); returns one or more `TokenBaseCard` records. If no hash is given, will return all active cards belonging to the customer.

*tokenBaseCheckoutConfig(method: String!): TokenBaseCheckoutConfig*

Get checkout configuration for the given TokenBase payment method. Takes a TokenBase payment method code, such as `paradoxlabs_clover`; returns a `TokenBaseCheckoutConfig`. This returns all data necessary to render and handle the client-side checkout form. Values mirror what is passed to Magento's standard frontend checkout.

### Mutations

*createTokenBaseCard(input: TokenBaseCardCreateInput!): TokenBaseCard*

Create a new stored card. Takes `TokenBaseCardCreateInput`, returns the new stored `TokenBaseCard` if successful.

*deleteTokenBaseCard(hash: String!): Boolean*

Delete a stored card. Takes a card hash; returns true if successful.

### *updateTokenBaseCard(input: TokenBaseCardUpdateInput!): TokenBaseCard*

Update an existing stored card. Takes `TokenBaseCardUpdateInput`; returns the updated `TokenBaseCard` if successful.

## Data Types

### *TokenBaseCard*

A stored payment account/credit card.

```

type TokenBaseCard {
  hash: String           Card identifier hash
  address: CustomerAddress Card billing address
  customer_email: String Customer email
  customer_id: Int       Customer ID
  customer_ip: String    Created-by IP
  payment_id: String     Card gateway payment ID
  method: String         Payment method code
  active: Boolean        Is card active
  created_at: String     Created-at date
  updated_at: String    Last updated date
  last_use: String       Last used date
  expires: String        Expiration date
  label: String          Card label
  additional: TokenBaseCardAdditional Card payment data
}

```

### *TokenBaseCardAdditional*

Details and metadata for a stored CC/ACH.

```

type TokenBaseCardAdditional {
  cc_type: String       CC Type
  cc_owner: String      CC Owner
  cc_bin: String        CC Bin (CC First-6)
  cc_last4: String      CC Last-4
  cc_exp_year: String   CC Expiration Year
  cc_exp_month: String  CC Expiration Month
  token: String         Clover card token
}

```

### *TokenBaseCheckoutConfig*

Checkout configuration for a TokenBase payment method.

```

type TokenBaseCheckoutConfig {
  method: String       Payment method code
  usevault: Boolean    Are stored cards enabled?
  canSaveCard: Boolean Can cards be saved?
  forceSaveCard: Boolean Is card saving forced?
  defaultSaveCard: Boolean Hash of the default card to select
  logoImage: String    Payment logo image URL (if enabled)
  sdkURL: String       URL for the payment form sdk.js library
  publicKey: String    Clover account public key
  merchantId: String   Clover account merchant ID
}

```

### *TokenBaseKeyValue*

Container for generic key/value data.

```

type TokenBaseKeyValue {
  key: String          Generic key
  value: String        Generic value
}

```

### *TokenBaseCardUpdateInput*

Input for updating a stored card.

```

input TokenBaseCardUpdateInput {
  hash: String!       Card identifier hash to update (required)
  address: CustomerAddressInput Card billing address
}

```

```

customer_email: String      Customer email
customer_ip: String        Created-by IP
method: String             Payment method code
active: Boolean            Is card active
expires: String           Card expiration date (YYYY-MM-DD 23:59:59)
additional: TokenBaseCardPaymentInput  Card payment data
}

```

### TokenBaseCardCreateInput

Input for creating a stored card.

```

input TokenBaseCardCreateInput {
  address: CustomerAddressInput  Card billing address
  customer_email: String!        Customer email (required)
  customer_ip: String           Created-by IP
  method: String!              Payment method code (required)
  active: Boolean              Is card active
  expires: String             Card expiration date (YYYY-MM-DD 23:59:59)
  additional: TokenBaseCardPaymentInput  Card payment data
}

```

### TokenBaseCardPaymentInput

Payment data for a stored card. Note, the specific fields that are relevant depend on the payment method. This data structure is also used for adding payment data to the cart during checkout.

```

input TokenBaseCardPaymentInput {
  save: Boolean                Save the card for later use?
  card_id: String             Card identifier hash to use
  token: String               Clover form token
}

```

## GraphQL Query Examples

Some response data has been omitted for brevity.

### Fetch card by ID

Example request:

```

{
  tokenBaseCards(hash:"ec431a3e1f9904a35dc083a257cf2585de7b7b6c") {
    label,
    expires,
    hash,
    customer_email,
    customer_id,
    profile_id,
    payment_id,
    method,
    active,
    created_at,
    updated_at,
    last_use,
    address {
      region {
        region_code,
        region,
        region_id
      },
      region_id,
      country_id,
      street,
      company,
      telephone,
      postcode,
      city,
      firstname,
      lastname
    },
    additional {
      cc_type,
      cc_bin,

```

```

    cc_last4,
    cc_exp_year,
    cc_exp_month
  }
}
}

```

Example response:

```

{
  "data": {
    "tokenBaseCards": [
      {
        "label": "Visa XXXX-0027",
        "expires": "2022-12-31 23:59:59",
        "hash": "ec431a3e1f9904a35dc083a257cf2585de7b7b6c",
        "customer_email": "roni_cost@example.com",
        "customer_id": 1,
        "profile_id": null,
        "payment_id": "S4FHQMSC7GG9j",
        "method": "paradoxlabs_clover",
        "active": true,
        "created_at": "2020-02-25 18:05:12",
        "updated_at": "2020-02-25 18:05:12",
        "last_use": null,
        "address": {
          "region": {
            "region_code": "PA",
            "region": "Pennsylvania",
            "region_id": 51
          },
          "region_id": 51,
          "country_id": "US",
          "street": [
            "123 Test Lane",
            ""
          ],
          "company": "",
          "telephone": "1111111111",
          "postcode": "17603",
          "city": "Test City",
          "firstname": "John",
          "lastname": "Doe"
        },
        "additional": {
          "cc_type": "VI",
          "cc_bin": "400700",
          "cc_last4": "0027",
          "cc_exp_year": "2022",
          "cc_exp_month": "12"
        }
      }
    ]
  }
}

```

### Fetch checkout config

Example request:

```

{
  tokenBaseCheckoutConfig(method:"paradoxlabs_clover") {
    method,
    useVault,
    canSaveCard,
    forceSaveCard,
    defaultSaveCard,
    logoImage,
    sdkURL,
    publicKey,
    merchantId
  }
}

```

Example response:

```
{
  "data": {
    "tokenBaseCheckoutConfig": {
      "method": "paradoxlabs_clover",
      "useVault": true,
      "canSaveCard": true,
      "forceSaveCard": false,
      "defaultSaveCard": true,
      "logoImage": "false",
      "sdkURL": "https://example.com/sdk.js",
      "publicKey": "abc123",
      "merchantId": "1000004973"
    }
  }
}
```

### Create card

Example request:

```
mutation {
  createTokenBaseCard(
    input: {
      expires: "2022-12-31 23:59:59",
      customer_ip: "127.0.0.1",
      customer_email: "roni_cost@example.com",
      method: "paradoxlabs_clover",
      active: true,
      address: {
        region: {
          region_code: "PA",
          region: "Pennsylvania",
          region_id: 51
        },
        country_id: US,
        street: [
          "123 Test St.",
          "Apt 9"
        ],
        company: "",
        telephone: "111-111-1111",
        postcode: "12345",
        city: "Testcity",
        firstname: "John",
        lastname: "Doe"
      },
      additional: {
        token: "7f58b566-911d-4307-9e51-758391728c9a",
        save: true
      }
    }
  ) {
    label,
    expires,
    hash,
    customer_email,
    customer_id,
    customer_ip,
    payment_id,
    method,
    active,
    created_at,
    updated_at,
    last_use,
    address {
      region {
        region_code,
        region,
        region_id
      },
      region_id,
      country_id,
      street,
      company,
      telephone,
      postcode,
      city,
      firstname,
      lastname
    }
  }
}
```

```

    },
    additional {
      cc_type,
      cc_bin,
      cc_last4,
      cc_exp_year,
      cc_exp_month
    }
  }
}

```

Example response:

```

{
  "data": {
    "createTokenBaseCard": {
      "label": "Visa XXXX-0027",
      "expires": "2022-12-31 23:59:59",
      "hash": "a5412c321a5de0c1f3964492e0bfba2b48e5984b",
      "customer_email": "roni_cost@example.com",
      "customer_id": 1,
      "customer_ip": "127.0.0.1",
      "payment_id": "S4FHQMSC7GG9J",
      "method": "paradoxlabs_clover",
      "active": true,
      "created_at": null,
      "updated_at": null,
      "last_use": null,
      "address": {
        "region": {
          "region_code": "PA",
          "region": "Pennsylvania",
          "region_id": 51
        },
        "region_id": 51,
        "country_id": "US",
        "street": [
          "123 Test St.",
          "Apt 9"
        ],
        "company": "",
        "telephone": "111-111-1111",
        "postcode": "12345",
        "city": "Testcity",
        "firstname": "John",
        "lastname": "Doe"
      },
      "additional": {
        "cc_type": "VI",
        "cc_bin": "400700",
        "cc_last4": "0027",
        "cc_exp_year": "2022",
        "cc_exp_month": "12"
      }
    }
  }
}

```

### Delete card

Example request:

```

mutation {
  deleteTokenBaseCard(hash:"88bb7dc06faad55c77177446ed83047811234008")
}

```

Example response:

```

{
  "data": {
    "deleteTokenBaseCard": true
  }
}

```

### Place an order

Example request:

```
mutation {
  setPaymentMethodOnCart(
    input: {
      cart_id: "kDy0EkkJmIOxa6H3ceus6MjMasF9lao1"
      payment_method: {
        code: "paradoxlabs_clover",
        tokenbase_data: {
          card_id: "ec431a3e1f9904a35dc083a257cf2585de7b7b6c"
        }
      }
    }
  ) {
    cart {
      selected_payment_method {
        code,
        tokenbase_card_id,
        tokenbase_data {
          cc_type,
          cc_last4,
          cc_exp_year,
          cc_exp_month
        }
      }
    }
  }
  placeOrder(
    input: {
      cart_id: "kDy0EkkJmIOxa6H3ceus6MjMasF9lao1"
    }
  ) {
    order {
      order_number
    }
  }
}
```

Example response:

```
{
  "data": {
    "setPaymentMethodOnCart": {
      "cart": {
        "selected_payment_method": {
          "code": "paradoxlabs_clover",
          "tokenbase_card_id": "ec431a3e1f9904a35dc083a257cf2585de7b7b6c",
          "tokenbase_data": {
            "cc_type": "VI",
            "cc_last4": "0027",
            "cc_exp_year": "2022",
            "cc_exp_month": "12"
          }
        }
      }
    },
    "placeOrder": {
      "order": {
        "order_number": "000000676"
      }
    }
  }
}
```



## How-To: API Checkout Flow

The checkout flow for this extension is a little complex, due to the security mechanisms involved.

### Step 1: Fetch checkout payment parameters

For GraphQL, see the *'Fetch checkout config'* example. For standard checkout's REST requests, these parameters are provided to JS by Magento's frontend layout system.

There are several parameters that are critical to the Clover checkout flow. Most important is the SDK URL, public key, and merchant ID, which are needed for the hosted form fields:

```
"sdkURL": "https://example.com/sdk.js",
"publicKey": "7e821849530436cb1f183d879499cd83",
"merchantId": "RCTSTAVI0010002",
"formStyle": "{...}"
```

The other parameters communicate Magento payment settings and state, and can be used or disregarded as you choose.

### Step 2: Initialize Hosted Form

Once you have the params, load the sdkURL onto the page. Once that's complete, you can instantiate it with the params. For example:

```
this.clover = new Clover(
  this.publicKey,
  {
    merchantId: this.merchantId,
    showSecurePayments: true
  }
);

const elements = this.clover.elements();

var ccNumber = elements.create('CARD_NUMBER', this.formStyle);
var ccExp = elements.create('CARD_DATE', this.formStyle);
var ccCid = elements.create('CARD_CVV', this.formStyle);
var ccZip = elements.create('CARD_POSTAL_CODE', this.formStyle);
ccNumber.mount('#' + this.getCode() + '-cc-number');
ccExp.mount('#' + this.getCode() + '-cc-exp');
ccCid.mount('#' + this.getCode() + '-cc-cid');
ccZip.mount('#' + this.getCode() + '-cc-postcode');

ccNumber.addEventListener('change', this.handleFieldChange.bind(this));
ccNumber.addEventListener('blur', this.handleFieldBlur.bind(this));
ccExp.addEventListener('change', this.handleFieldChange.bind(this));
ccExp.addEventListener('blur', this.handleFieldBlur.bind(this));
ccCid.addEventListener('change', this.handleFieldChange.bind(this));
ccCid.addEventListener('blur', this.handleFieldBlur.bind(this));
ccZip.addEventListener('change', this.handleFieldChange.bind(this));
ccZip.addEventListener('blur', this.handleFieldBlur.bind(this));
```

### Step 3: Tokenize the card

When the customer clicks Place Order, you will need to tell Clover to tokenize the payment details for you. For example:

```
this.clover.createToken()
  .then(this.handleToken.bind(this))
  .catch(this.handleToken.bind(this));
```

```
handleToken: function(result) {
  if (result.errors) {
    var message = result.errors.join('; ');
    window.alert(message);
  } else {
    this.creditCardExpMonth(result.card.exp_month);
    this.creditCardExpYear(result.card.exp_year);
    this.creditCardBin(result.card.first6);
    this.creditCardLast4(result.card.last4);
    this.creditCardType(result.card.brand);

    this.token(result.token);
    this.placeOrder();
  }
}
```

#### Step 4: Receive token and place order

If successful, your handleToken method will be called when the tokenization is completed successfully. Complete the order by passing the token and other card details as the order payment data.

## Support

If you have any questions not covered by this document, or something isn't working right, please open a ticket in our support system: [support.paradoxlabs.com](https://support.paradoxlabs.com)

Support Policy: <https://store.paradoxlabs.com/support.html>

License and Terms of Use: <https://store.paradoxlabs.com/license.html>