Methodology

Overview

ChipDNA Cloud is a method of processing retail payments using an internet connected terminal without the use of an installed SDK.

Since Cloud terminals do not allow for follow up transactions or use of tokenized payment details, you can use this in conjunction with an existing transaction API (Direct Post) to submit follow up transactions (capture, void, refund) or use other gateway services that utilize payment information.

Choosing between Synchronous and Asynchronous Processing

Understanding the difference and choosing between the two available response styles will decide how you use the API, and whether or not you need to use the AsyncStatus Polling API.

- **Synchronous**: You issue your request and wait for a response from the gateway. Due to the customer interaction with the POI device, the wait time could be up to 2 minutes. You will receive a transaction ID and a final transaction condition in the response and do not need to do any follow up AsyncStatus polling to determine the results of the transaction.
    - **Why use it**: Less development, and the API response is identical to a standard Direct Post API response. Parsing the response, if you already use the Direct Post API, will be easy.
    - **Caveats**: Due to internet connectivity issues, if you lose internet connectivity while waiting for the API call, you will not receive the API response. This will require you to query the gateway or log into the control panel to see the status of the transaction.
- **Asynchronous**: You issue your request and immediately get a GUID in return. You will then use the AsyncStatus Polling API to determine the state of the transaction and final condition once the customer's interaction with the POI device is complete.
    - **Why use it**: If you want to ensure that your system has a way to query on a transaction while it is in flight, or if you have many devices in the field, this is the better option as it doesn't require maintaining multiple sessions while a transaction processes. Responses are immediate for the Direct Post API and the AsyncStatus API.
    - **Caveats**: More development due to the addition of the AsyncStatus API, and the response has minimal detail in it. You may need to use the Query API if you wish to pull back more details about the transaction.

Steps to register a terminal and process a transaction

Step 1: Register your terminal using the current **Code** value visible on an internet-connected device.

Step 2: Process payments with your terminal using the returned POI Device ID. If you are using the synchronous method of processing, you will receive a transaction ID and the final condition of the transaction in the standard Direct Post API format. See Step 3 if you are using the asynchronous method of processing.

Step 3 (for Async users only): If you choose to process using the asynchronous method, you will have received a GUID in the response of Step 2. You will use this GUID to poll for the transaction and terminal status until the customer interaction has completed. The final results of the polling will return the final condition and transaction status.

Usage

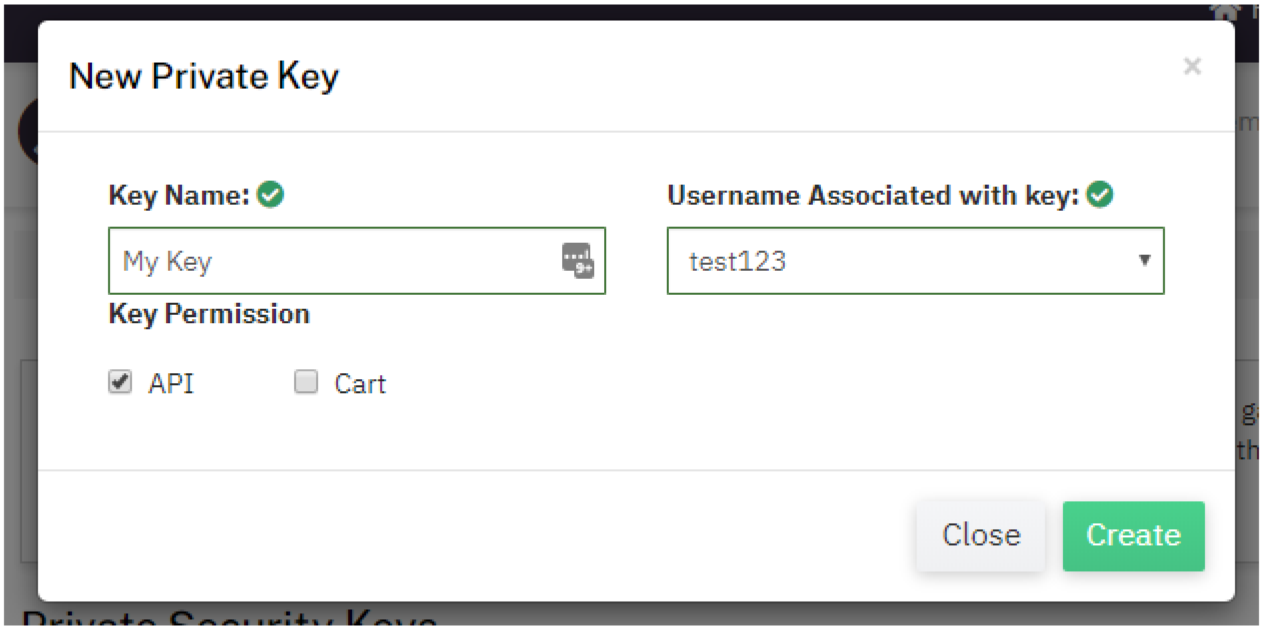
Cloud terminals, referred to hereafter as a POI Device, must be plugged into an ethernet port with internet access. After plugging in the device, it will immediately try to connect to the platform and start rotating through registration codes.

Once your POI Device is registered, you can begin taking payments on it.

Authentication

Authentication is done via a "security key" that you can generate in your merchant control panel under the "Security Keys" settings page. Select "API" for the key type.

This security key can be used with most other APIs (Direct Post, Three-Step, Query) except for Collect.js. You can use an existing API key that has the proper source selected (API.)


New Private Key

Key Name: My Key

Username Associated with key: test123

Key Permission

API [✓] Cart [ ]

Testing credentials

Transactions can be tested using one of two methods. First, transactions can be submitted to any merchant account that is in test mode if you have a staging terminal in hand. Keep in mind that if an account is in test mode, valid credit cards will be approved but **no charges will actually be processed.**

The Payment Gateway demo account can also be used for testing transactions at any time but does not support querying for security reasons. Please use the following security key for testing with this account if you do not wish to use (or do not have) your own account:

security_key:	2F822Rw39fx762MaV7Yy86jXGTC7sCDy

Quick Setup Guide

1. **Your Gateway Account**: Ensure that you have a supported processor on your gateway account if you are using a *live* account. Supported processors are listed below:
2. **Supported Processors**:
    - TSYS - EMV
    - Chase Paymentech NetConnect - EMV
    - Elavon - EMV
    - First Data Rapid Connect Cardnet North - EMV
    - First Data Rapid Connect Nashville North - EMV
    - First Data Rapid Connect FE:Nashville BE:North/Omaha/South Host Capture - EMV
    - First Data Rapid Connect Omaha - EMV
    - Vantiv Core Host Capture - EMV

    Your gateway account should *not* be in test mode in order to connect to the Cloud with a production POI Device. If you have a 'Staging' device, you MUST be in Test Mode or be using a Test Account.

    Contact your account provider to order a POI Device.

3. **Create an API Key in your Gateway Account**: Since using a Cloud enabled POI Device does not allow username/password usage, you must create an API key and use the documented 'security_key' parameter via API. Log into your account and, if you have Administrative access to the account, you should be able to click on 'Options' and then 'Security Keys' to access the correct page. You may also click on 'Settings' in the upper right and choose 'Security Keys' from the dropdown. If you do not have Admin access to your account, please contact an Administrator for your account to assist with this step.

    Once you're on the Security Keys page, you should create a 'Private' Security Key by following the below steps:

    1. Click 'Add a New Private Key'
    2. Enter in the Key "Name" - this is a nickname and can be anything you want.
    3. Choose the 'Username' to associate with this key. This username will control what permissions the API key has access to including transaction types, processors, and services such as the Customer Vault, APIs, and so on.
    4. Choose 'API' as the Key Permission.
    5. Click 'Create'.

    Your new Security Key for registration, deregistration, estate management, and processing is now available in the list of Private Security Keys. You will use the value listed under 'Key', and can ignore the value under 'Key ID'.

4. **Connecting your POI Device to the internet**: Give your POI Device power and internet connectivity via the power adapter and ethernet cord that comes with the device. The ethernet port you connect the device to must have open internet access. If your device

does not display "Unregistered" with a rotating "Code" value upon bootup, it is likely that the network you are connecting to is somehow blocking outside access to the internet. You will need to speak with your IT/Network administrator to open up firewalls/allow access for the device to connect to the Cloud.

You're now ready to register your POI device and start processing! Follow the Device Management Registration and Deregistration documentation to start using your device.

Device Management API

Registration

Before you can use a ChipDNA Cloud POI Device, you must register it to your gateway account using an API key. Doing this will return a device GUID that you can use to process payments using the POI Device.

Deregistration

When you no longer wish to use your POI Device, or need to use it with a different gateway account, you must deregister it.

Endpoint

```
https://centavo.transactiongateway.com/api/v2/devices/register
```

Headers

Every API request must be authenticated using HTTP Bearer Authentication header and include Content-Type

```
Authorization: Bearer {MERCHANT_API_KEY}Content-Type: application/json
```

Device Registration

In this request, if the registration code is valid, the device will be registered and a POI device GUID will be returned.

```
POST /api/v2/devices/register
```

Parameter	Type	Required	Description
registrationCode	string	yes	The value that appears on an internet-connected device when it is not yet registered or has been deregistered.
deviceNickname	string	no	When sent will appear on the POI Device screen and in the UI License Manager, as well as in Estate Management queries.

Example Request

```
curl --request POST \--header "Authorization: Bearer
{MERCHANT_API_KEY} " \--header "Content-Type: application/json" \-d
'{"registrationCode":"8AVJWV","deviceNickname":"My POI Device"}' \
"https://centavo.transactiongateway.com/api/v2/devices/register"
```

Example Response

```
{
"poiDevice": {
"poiDeviceId":"414b0420-b31c-4c76-9fc3-8a4f1a6dcffc",
"deviceNickname":"My POI Device", "deviceLicense":"414b0420-b31c-4c76-
9fc3-8a4f1a6dcffc",
"serialNumber":"3375191PT103344",
"registrationStatus":"registered"
}
}
```

Response Details

poiDeviceId

The POI device id that was returned in the register response.

deviceNickname

The device nickname provided in the request.

deviceLicense

The device license. In most cases this will be the same as poiDeviceId

serialNumber

The serial number of the device. This will never change and is a good tracking system for devices.

registrationStatus

The registration status of the device. This will always be "registered" for a registration response.

Device Deregistration

In this request, if the POI device ID is valid, the deregistration will be successful and the POI device will be returned to an unregistered state.

`DELETE /api/v2/devices/deregister/:poiDeviceId`

Parameter	Type	Required	Description
poiDeviceId	string	yes	The device id returned back as poiDeviceId in the register response.

Example Request

```
curl --request DELETE \--header "Authorization: Bearer
{MERCHANT_API_KEY}" \
"https://centavo.transactiongateway.com/api/v2/devices/deregister/d352da6e-8772-4
```

Example Response

```
{
"registrationStatus":"deregistered"
}
```

Device Estate Management

When you need information on a specific POI device, or all devices associated with your gateway account, you can run a basic query against your entire estate or target a POI Device ID to get information such as the POI Device ID value, the current nickname, dates registered and deregistered, last date used, and more.

Endpoint

```
https://centavo.transactiongateway.com/api/v2/devices/list
```

Headers

Every API request must be authenticated using HTTP Bearer Authentication header and include Content-Type

```
Authorization: Bearer {MERCHANT_API_KEY}Content-Type: application/json
```

Single POI device request

Send the GET request with the POI Device ID and the data for that specific ID will be returned.

Request:

```
GET /api/v2/devices/list/:poiDeviceId
```

Parameter	Type	Required	Description
poiDeviceId	string	yes	The POI device ID value that was returned back in the original registration response.

Example Request

```
curl --request GET \--header "Authorization: Bearer
{MERCHANT_API_KEY}" \
"https://centavo.transactiongateway.com/api/v2/devices/list/d352da6e-8772-4433-9
```

Example Response

```
{
"poiDevices": [
{
"deviceId": "fc572cda-ae8e-4f21-aa53-d2905c30d696", "make":
"ingenico",
"model": "IPP320",
"lastTransactionDate": "2019-11-22T05:52:08Z", "dateRegistered": "2019-
11-21T20:50:08Z",
"dateDeregistered": null,
"serialNumber": "3375191PT103342",
"deviceNickname": "My POI Device",
"registrationStatus": "registered"
}
]
}
```

All POI Devices associated with an account

Send a GET request to the endpoint without a POI Device ID indicated.

```
GET /api/v2/devices/list
```

Example Request

```
curl --header "Authorization: Bearer {MERCHANT_API_KEY}"
 "https://centavo.transactiongateway.com/api/v2/devices/list"
```

Example Response

```
{
"poiDevices": [
{
"deviceId": "fc572cda-ae8e-4f21-aa53-d2905c30d696", "make":
"ingenico",
"model": "IPP320",
"lastTransactionDate": "2019-11-22T05:52:08Z", "dateRegistered": "2019-
11-21T20:50:08Z",
"dateDeregistered": null,
"serialNumber": "3629940PT696190",
"deviceNickname": "My POI Device",
"registrationStatus": "registered"
},
{
"deviceId": "73c3791a-e619-4d9d-a687-913ceb9721fa", "make":
"ingenico",
"model": "IPP320",
"lastTransactionDate": "2019-11-22T05:52:08Z", "dateRegistered": "2019-
11-21T20:50:08Z",
"dateDeregistered": "2019-11-22T20:50:08Z", "serialNumber":
"3375191PT103344",
"deviceNickname": "My POI Device", "registrationStatus":
"deregistered"
},
...
]
}
```

Error Handling

If successful, the response HTTP status code is **200 OK**.

The following HTTP status codes will be returned in the event of various errors:

- **400** - No Platform ID or no API Key was sent in the request.
- **401** - An invalid API Key was sent in the request.

See Error Recover Tips for common errors and ways to resolve them.

Response Details

errors array

An array of error objects describing what went wrong

errors[0].code

The gateway response codes - see further documentation here.

errors[0].refid

A specific error id that can be used for troubleshooting with technical support if necessary.

errors[0].message

A specific message associated with this error. Use the error recovery tips to move past these errors or contact support if necessary.

Example Error

```
{
"errors": [
{
"code": 300,
"refid": 12345,
"message": "Message describing the error"
},
...
]
}
```

Processing

Card Charges (Sale & Auth)

To charge a credit or debit card, you will process a sale or an auth transaction. You can follow these transaction types up with a capture (to allow an auth to settle), a void (to cancel a sale or an auth), or a refund (to reverse a sale or a captured auth.) You can add to the Customer Vault using this transaction type to store payment data while charging a card.

Card Validation

You can also check the validity of a card by processing a validate transaction. This will process a 0.00 transaction against the card details provided. You can add to the Customer Vault using this transaction type to store payment data without charging a card.

Unlinked Credits

If you need to return funds to a customer and do not have an original transaction with which to process a refund, or the customer does not have the original card, you can process an unlinked credit.

Response Style Selection

The response style represents which method you wish to receive responses from the gateway, and will decide whether or not you use the AsyncStatus polling API. There are two options:

- **Synchronous**: You issue your request and wait for a response from the gateway. Due to the customer interaction with the POI device, this could be up to 5 minutes. The transaction will time out after 300 seconds and will need to be restarted. You will receive a transaction ID and a final transaction condition in the response and do not need to do any follow up AsyncStatus polling to determine the results of the transaction.
- **Asynchronous**: You issue your request and immediately get a GUID in return. You will then use the AsyncStatus Polling API to determine the state of the transaction and final condition with that GUID. This will provide you with the response of the customer's interaction with the POI device once it is complete.
  - Related Documentation: AsyncStatus API

Endpoint

```
https://centavo.transactiongateway.com/api/transact.php
```

Headers and Request Data Format

The Direct Post API accepts either **multipart/form-data** or **application/x-www-form-urlencoded**. Depending on which format is used, the **Content-Type** header should be set to the correct value.

Transaction Processing

```
POST /api/transact.php
```

Request Details

Variable Name	Description
security_key*	API Security Key assigned to a merchant account. New keys can be generated from the merchant control panel in Settings > Security Keys. **Note**: Using the 'username' and 'password' variables in the request will result in an error.
poi_device_id*	The registered terminal ID. Provide on transaction types of sale, auth, credit and validate.
response_method	The type of response you wish to see returned. Set to '**synchronous**' to wait for a final transaction ID. Set to '**asynchronous**' to receive an async_status_guid value to poll against. Default: 'synchronous' Values: 'synchronous' or 'asynchronous' **Note**: You must use the AsyncStatus API if you use 'asynchronous' processing. Please see 'Response Style Selection' above for more information. You can see an example of the response you'll get in the examples below.
type	The type of transaction to be processed. Values: 'sale', 'auth', 'credit', or 'validate'
amount	Total amount to be charged. For validate, the amount must be omitted or set to 0.00. Format: x.xx
first_name	Cardholder's first name.
last_name	Cardholder's last name.
company	Cardholder's company.
address1	Card billing address.
address2	Card billing address, line 2.
city	Card billing city.
state	Card billing state. Format: CC
zip	Card billing zip code.
country	Card billing country. Country codes are as shown in ISO 3166. Format: CC
phone	Billing phone number.
email	Billing email address.

order_description	Order description.
orderid	Order Id.
ponumber	Original purchase order.
tax	The sales tax, included in the transaction amount, associated with the purchase. Format: x.xx
merchant_defined_field_#	You can pass custom information in up to 20 fields. Format: merchant_defined_field_1=Value
customer_vault	Associate payment information with a Customer Vault record if the transaction is successful. Values: 'add_customer' or 'update_customer'
customer_vault_id	Specifies a customer vault id. If not set, the payment gateway will randomly generate a customer vault id.
processor_id	If using Multiple MIDs, route to this processor (processor_id is obtained under Settings->Transaction Routing in the Control Panel).

New Response Details

Variable Name	**Description**
async_status_guid*	Will contain a GUID for use with polling. Will only return dynamically when 'response_method' is set to 'asynchronous'. Store and use for retrieving the final status of a transaction using the AsyncStatus API. Format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
platform_id*	DEPRECATED: Will contain a GUID for use with polling. Will only return dynamically when 'response_method' is set to 'asynchronous'. Store and use for retrieving the final status of a transaction using the AsyncStatus API. Format: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
verification_method	A CVM is a means of checking that the user of a card is the genuine cardholder. The returned value will indicate how the customer was verified. Values: 'signature', 'offline_pin', 'online_pin', 'offline_pin_signature', or 'none'
transaction_status_information	A collection of indicators that the terminal will set to show what processing steps have been performed on the current transaction (e.g. Cardholder Verification, Data Authentication). Example: E800.
emv_application_id	Identifies the EMV application that the terminal used on the transaction as described in ISO/IEC 7816-5. Example: A000000003101001.
emv_application_label	The human readable name associated with the AID according to ISO/IEC 7816-5. Example: Visa International.

emv_application_preferred_label	The human readable name associated with the applicationId in the cardholder’s local language. This value will not always be present.

* Only applicable/useful when processing **asynchronous** transactions. Store to poll against using the AsyncStatus API. **Note**: If you are using **synchronous** processing method and you have a custom API response set to include this value, you can safely ignore it and do not need to poll the status API with it.

Examples:

Synchronous Sale: In this request, a transaction ID will be returned and the final state will be returned like a typical Direct Post response.

```
...type=sale&poi_device_id=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
 &response_method=synchronous&amount=1.00&...
```

```
...response=1&responsetext=Approved&authcode=XXXXX&transactionid=XXXXXX
XXXX &type=sale&response_code=100
```

Synchronous Sale & Add to Vault: In this request, a transaction ID and a Customer Vault ID will be returned and the final state will be returned like a typical Direct Post response.

```
...type=sale&poi_device_id=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
 &response_method=synchronous&amount=1.00&customer_vault=add_customer...
```

```
...response=1&responsetext=Approved&authcode=XXXXX&transactionid=XXXXXX
XXXX &type=sale&response_code=100&customer_vault_id=XXXXXXXXX
```

Asynchronous Sale: In this request, a **async_status_guid** will be returned that you must poll against using the AsyncStatus API.

```
...type=sale&poi_device_id=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
 &response_method=asynchronous&amount=1.00&...
```

```
...response=1&responsetext=Request Accepted&...&response_code=101
 &async_status_guid=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

Asynchronous Sale + Add to Vault: In this request, a **async_status_guid** will be returned that you must poll against using the AsyncStatus API. You will not receive a Vault ID in the initial Direct Post API call.

```
...type=sale&poi_device_id=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
 &response_method=asynchronous&amount=1.00&customer_vault=add_customer...
```

```
...response=1&responsetext=Request Accepted&...&response_code=101
 &async_status_guid=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

Since the asynchronous response does not return a transaction ID or the status / condition of the transaction, use the async_status_guid value to poll against the terminal itself. See **'AsyncStatus API'** below for documentation.

Voids / Captures / Refunds

To void a sale or an auth, capture an auth, or refund a transaction, please follow the documented Direct Post methods in a request to your gateway account. You do not need to provide the POI Device ID or response_method variables as there is no device interaction on these types of transactions.

Examples:

Void: In this request, a transaction ID will be returned and the final state will be returned like a typical Direct Post response. You do not need to provide the POI Device ID as there is no device interaction on voids.

```
...type=void&transactionid=XXXXXXXXXX...
```

```
response=1&responsetext=Transaction Void Successful&authcode=XXXXXX
 &transactionid=XXXXXXXXXX&type=void&response_code=100
```

Capture: In this request, a transaction ID will be returned and the final state will be returned like a typical Direct Post response. You do not need to provide the POI Device ID as there is no device interaction on captures. You can specify an amount or not. Not specifying an amount will capture the full amount authorized.

```
...type=capture&transactionid=XXXXXXXXXX&amount=X.XX...
```

```
response=1&responsetext=Transaction Capture Successful&authcode=XXXXXX
 &transactionid=XXXXXXXXXX&type=capture&response_code=100
```

Refund: In this request, a transaction ID will be returned and the final state will be returned like a typical Direct Post response. You do not need to provide the POI Device ID as there is no device interaction on refunds. You can specify an amount or not. Not specifying an amount will refund the full amount authorized.

```
...type=refund&transactionid=XXXXXXXXXX...
```

```
response=1&responsetext=Approved&authcode=XXXXXX&transactionid=XXXXXXXX
XX &type=refund&response_code=100
```

AsyncStatus Polling

If you are integrating to the "asynchronous" version of transaction processing, you will need to use this endpoint to determine whether or not the consumer's interaction with the POI device has completed or not and whether or not the transaction was successful. Using the 'async_status_guid' value you received in an asynchronous transaction response, you can send requests and receive a JSON response.

Endpoint

```
https://centavo.transactiongateway.com/api/asyncstatus
```

Headers

Every API request must be authenticated using HTTP Bearer Authentication header

```
Authorization: Bearer {MERCHANT_API_KEY}
```

Poll against an Asynchronous Transaction

Using the GUID returned in the 'async_status_guid' on an asynchronous transaction request, perform a GET to receive back JSON formatted response with the most up to date status of the transaction as the customer is interacting with the POI Device.

```
GET /api/asyncstatus/:asyncStatusGuid
```

Path Parameter	Type	Required	Description
asyncStatusGuid	string	yes	The async_status_guid returned from the transaction response. Example: 04249b3a-02f9-4838-b7b5-2bef4d4e7f7d

Example request:

```
curl --request GET --header "Authorization: Bearer {MERCHANT_API_KEY}"
 "https://centavo.transactiongateway.com/api/asyncstatus/70a6272c-4949-4515-956a
```

Example Response:

While the transaction is in progress or 'in flight', the responses will return with little information other than the platform ID and status.

```
{
"transaction": {
"id": null,
"success": false,
"condition": "",
"authCode": ""
},
"platformId": "70a6272c-4949-4515-956a-6e5ae4d5a10c",
"asyncStatusGuid": "70a6272c-4949-4515-956a-6e5ae4d5a10c",
"asyncStatus": "inFlight"
}
```

In the event of an approval (sale or credit), the response would **change** to this:

```
{
"transaction": {
"id": 4869813602,
"success": true,
"condition": "pendingsettlement",
"authCode": "A99BUO"
},
"platformId": "70a6272c-4949-4515-956a-6e5ae4d5a10c",
"asyncStatusGuid": "70a6272c-4949-4515-956a-6e5ae4d5a10c",
"asyncStatus": "interactionComplete"
}
```

Note: An 'auth' or 'validate' action would have a different condition. Auths remain in a 'pending' state and require capturing to settle, and Validates are immediately 'complete' upon approval. In the event of a decline, the response would change to this:

```
{
"transaction": {
"id": 4869813602,
"success": false,
"condition": "failed",
"authCode": ""
},
"platformId": "70a6272c-4949-4515-956a-6e5ae4d5a10c",
"asyncStatusGuid": "70a6272c-4949-4515-956a-6e5ae4d5a10c",
"asyncStatus": "interactionComplete"
}
```

If the transaction is an EMV transaction, the response will include emvMetaData details for creating custom receipts:

```
{"transaction": {"id": 4869813602,"success": true,
"condition": "pendingsettlement","authCode": "A99BUO""emvMetaData": {
"customerVerificationMethod": "signature", "applicationId":
"A0000000031010","applicationLabel": "VISA CREDIT",
 "applicationPreferredName": "CREDITO DE VISA",
"transactionStatusInformation": "E800",}},
"platformId": "70a6272c-4949-4515-956a-6e5ae4d5a10c",
"asyncStatusGuid": "70a6272c-4949-4515-956a-6e5ae4d5a10c",
"asyncStatus": "interactionComplete"
}
```

Vault details will be returned if specified in the transaction request.

```
{
"transaction": {
"id": "281474978429650",
"success": true,
"condition": "pendingsettlement",
"authCode": "SH14L8",
"customerVaultId":"9742353903"
},
"platformId": "86d84df6-e062-49f6-ba81-4b667839e34e",
"asyncStatusGuid": "86d84df6-e062-49f6-ba81-4b667839e34e",
"asyncStatus": "interactionComplete"
}
```

If there was an error it will be in the asyncstatus response.

```
{
"transaction": {
"id": null,
"success": false,
"condition": "",
"authCode": ""
},
"error": {
"code":300,
"refid":"58523101",
"message":"error message will appear here"
},
"platformId": "dd1c7f5c-13b5-4098-82c1-73d3e1bb085f",
"asyncStatusGuid": "dd1c7f5c-13b5-4098-82c1-73d3e1bb085f",
"asyncStatus": "interactionComplete"
}
```

If successful, the response HTTP status code is **200 OK.**

Errors

The following HTTP status codes will be returned in the event of various errors:

-

- **400** - No Platform ID or no API Key was sent in the request.
- **401** - An invalid API Key was sent in the request.
- **404** - The Platform ID provided was not found.

Response Details

transaction object

The transaction object containing a subset of information

transaction.id

The gateway transaction ID that can be used for querying.

transaction.success

Will return true or false depending on the final result of the transaction.

- **true** -indicates a successful transaction (approved)
- **false** - indicates a failed transaction (declined).
  The response will return false also in the event of an error, a cancelled transaction, or when the customer has not yet finished interacting with the POI Device.

transaction.condition

The final condition of the transaction.

- **pendingsettlement** - for successful Sales and Credits
- **pending** - for successful Auths
- **complete** - for successful Validates
- **failed** - for declined transactions

authCode

If the transaction was successful, the auth code will be returned.

customerVaultId

If the transaction request included customer vault parameters, the customer vault ID will be returned.

emvMetaData object

If the transaction was an EMV/Chip Card insert, extra data about the transaction will be returned.

emvMetaData.customerVerificationMethod

A CVM is a means of checking that the user of a card is the genuine cardholder. The returned value will indicate how the customer was verified.

- **signature** - The customer either signed digitally or was asked to sign a physical receipt.
- **offlinePin** - The customer entered their pin on the POI Device and it was verified by the

chip.
  - **onlinePin** - The customer entered their pin on the POI Device and it was verified by the Card Issuer.
  - **offlinePinSignature** - The customer entered their PIN on the POI Device and they signed digitally or a physical receipt.
  - **none** - The customer could not be verified by the POI Device.

emvMetaData.applicationId

Identifies the EMV application that the POI Device used on the transaction as described in ISO/IEC 7816-5. Example: A000000003101001.

emvMetaData.applicationLabel

The human readable name associated with the AID according to ISO/IEC 7816-5. Example: Visa International.

emvMetaData.applicationPreferredName

The human readable name associated with the applicationId in the cardholder's local language. This value will not always be present.

emvMetaData.transactionStatusInfo

A collection of indicators that the POI Device will set to show what processing steps have been performed on the current transaction (e.g. Cardholder Verification, Data Authentication). Example: E800.

error object

If the transaction experienced an error, the code, refid, and message will be returned.

error.code object

The static error code indicating an issue. Will be '300'.

error.refid object

The unique identification for the error. This value can be provided to support for further troubleshooting.

error.message object

The human readable message describing the issue with the transaction.

platformId

DEPRECATED: The current GUID being polled against.

asyncStatusGuid

The current GUID being polled against.

asyncStatus

The current status of the transaction on the POI Device itself.

  - **inFlight** - This means the customer is still interacting with the POI Device. There may or may not be a transaction created at this point.

- **cancelledAtTerminal** - This means the cancel button on the POI Device was pressed. No transaction was created.
- **interactionComplete** - This means the transaction has completed, and POI Device interaction has ceased. A new transaction can be started on this device.

POI Device Prompts

There are several variables available to customize the POI Device flow on a per transaction basis. All prompts have a default value (true or false). Not all prompts work together, so see below for an explanation.

- **Keyed Entry**: Using this prompt will provide the cardholder the option to key in their card if desired or if inserting/swiping is not possible for some reason (bad chip, magstripe for example.)
    - **CVV Entry**: When prompting for Keyed entry, specify whether you wish the cardholder to also enter their CVV or not. This is true by default.
- **Signature Prompting**: When inserting/dipping a chip card, signature prompting may be ignored in favor of the chip card communication with the POI Device. For example if the chip card determines that pin entry is required, signature will not be prompted even if sent as 'true'.
- **Amount Confirmation**: Using this prompt will provide the cardholder the option to confirm the amount being charged or not. This is false by default.
- **Tipping**:
    - If prompting for tip, you may set a tip amount via API or allow the customer to enter/select an amount on the POI device itself. Quicktip amount options can be customized as well. See below:
    - **QuickTip Amounts**: When sending custom quicktip amounts via API, poi_prompt_tip must be sent as 'true'.
        - The iPP320 device will display up to 3 custom amounts.
        - The iSC250 device will display up to 4 custom amounts.
        - Values will be displayed in the order they are provided. Values beyond 3-4 (depending on the device) will be ignored. Example: If, on an iPP320, the following is sent: poi_prompt_quicktip_amounts: 1.00,2.00,3.00,4.00 - the '4.00' amount will be ignored as the iPP320 only has space for 3 custom amounts.

POI Device Prompting Request Details

Variable Name	Description
tip	The final tip amount, included in the transaction amount, associated with the purchase. **Format**: x.xx
poi_prompt_tip	When set to 'true', will allow the cardholder to type in a tip amount via the POI Device. When sent in with the 'tip' value, this will show a confirmation of the tip value. **Format**: boolean (true/false) **Default**: false

poi_prompt_quicktip_amounts	Customize the quick tip amount options available when tipping via the POI Device. Invalid when sent with poi_prompt_tip set to 'false'. **Format**: X.XX, comma delimited up to 4 options. Example: 1.50,3.00,4.50
poi_enable_keyed	When set to 'true', provides the option to key-in the card details via the POI Device. **Format**: boolean (true/false) **Default**: false
poi_require_keyed	When set to 'true', forces the cardholder to key-in the card details via the POI Device. This prompt does not allow any other entry method when set. **Note:** Cannot be set to 'true' when poi_enable_keyed is also set to 'true'. **Format**: boolean (true/false) **Default**: false
poi_prompt_cvv	When set to 'false', the CVV code will not be requested from the customer via the POI Device on a keyed transaction. Invalid when sent with poi_enable_keyed set to 'false' or when poi_enable_keyed is not sent. **Format**: boolean (true/false) **Default**: true
poi_prompt_zip	When set to 'true', will enforce zip code/postal entry via the POI Device. **Note**: Entry on the device will override a zip code sent via API in the initial request. **Format**: boolean (true/false) **Default**: false
poi_prompt_amount_confirmation	When set to 'true', will force the user to confirm the amount being charged via the POI Device. **Format**: boolean (true/false) **Default**: false
poi_prompt_signature	When set to 'true', will force the user to sign via the POI Device where supported. When set to 'false', will skip the signature capture step for magstripe transactions. Will be overridden via Chip Card/POI Device interaction when applicable. **Format**: boolean (true/false) **Default**: true
poi_request	If you wish to reset the POI Device before a transaction has been processed, but after the transaction has been requested, you may request a cancellation. This does NOT guarantee the transaction can be cancelled on the device. If the authorization has already begun, you will need to void the transaction after it has completed, assuming it was approved. You must provide the POI Device ID in the request with this prompt. **Value**: 'cancellation'

poi_automatic_fall_forward	When set to 'true', will enable Fall Forward (EMV Contactless upgrade to EMV Contact) in-line via the POI Device where supported. When set to 'false', transactions will need to be restarted when tapped cards that are rejected by the Issuer require a contact/EMV insert to occur. **Format**: boolean (true/false) **Default**: true

Examples:

Customer Input Tip Prompt: With the following request, the specified POI device will prompt the cardholder to enter the amount they wish to tip. This value will be added onto the amount sent with the transaction request. The default QuickTip options will appear with this request.

Note: The default QuickTip options are a calculation of the total amount using the following percentages: 15%, 18%, and 20%. The final amounts will appear as dollar amounts on the device screen.

```
...type=sale&poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e
 &poi_prompt_tip=true&amount=11.00
```

Set Tip Prompt: With the following request, the specified POI device will prompt the cardholder to confirm they wish to tip the amount included with the transaction request (in this example, 1.00). This value will be added onto the amount sent with the transaction request.

```
...type=sale&poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e
 &poi_prompt_tip=true&tip=1.00&amount=11.00
```

QuickTip Customization: With all tip prompting, default QuickTip amounts will display along the F1-F4 keys on the POI Device screen. With the following request, the specified POI device will display custom amounts sent in the request instead of the default. This can be provided with or without a specific tip amount, but must be provided with the basic tip prompt of poi_prompt_tip=true.

Note: The iPP320 will show up to 3 custom options, and the iSC250 will display up to 4. Both will show an extra option to enter something other than the set tip or quicktip amounts. With the below example, the customer would be prompted to confirm the 1.00 amount, but the F1-F4 keys will display the provided amounts vs the default. If 'tip=1.00' is omitted, the customer would be prompted to enter an amount, but they could select one of the F1-F4 options if desired.

```
...type=sale&poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e
 &poi_prompt_tip=true&tip=1.00&poi_prompt_quicktip_amounts=1.00,3.00,5.00&...
```

Enabling Keyed Entry: With the following request, the specified POI device will show the standard 'Insert or Swipe' message, but also have the option to 'Enter Card' by selecting the F1 key on the device. If 'Enter Card' is selected, the cardholder will be prompted to enter their card number, the expiration date, and the CVV code from the back of the card. This is set to 'false' by default, so it only needs to be sent if you want this option available for use.

```
...type=sale&poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e
 &poi_enable_keyed=true...
```

Enforcing Keyed Entry: With the following request, the specified POI device will prompt the cardholder to enter their card number, the expiration date, and the CVV code from the back of the card. This is set to 'false' by default, so it only needs to be sent if you want to only allow keyed entry rather than give the 'option' to.

```
...type=sale&poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e
 &poi_require_keyed=true...
```

CVV Prompt w/ Keyed Entry: With the following request, the POI device will not display the CVV field when keying in a card to the POI Device. CVV prompting is 'true' by default.

```
...type=sale&poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e
 &poi_enable_keyed=true&poi_prompt_cvv=false...
```

Signature Confirmation: With the following request, the specified POI device will either enforce a digital signature capture (on iSC250 devices only) or show a message to capture the signature manually if on a device with no signature capture capabilities (ex: iPP320.)

Note: This prompt will only be honored on magstripe transactions as signature capture enforcement is controlled on EMV transactions by the chip itself which cannot be overridden.

```
...type=sale&poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e
 &poi_prompt_signature=true...
```

Amount Confirmation: With the following request, the specified POI device will either enforce a cardholder to "confirm" the amount shown on the POI Device screen or not. This is 'false' by default, so it must be sent in order to show the amount confirmation prompts.

```
...type=sale&poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e
 &poi_prompt_amount_confirmation=true...
```

Device Reset/Cancellation: With the following request, the current transaction request on specified POI device will be cancelled as long as authorization has not already begun.

```
...poi_request=cancellation
 &poi_device_id=0b3043ca-5764-4edc-b720-b652bdf73c9e...
```

```
...response=1 &responsetext=Transaction cancellation requested. If the
transaction cannot be cancelled, you must process a void request once
the transaction is complete. &...&response_code=100...
```

Error Recovery Tips

The ChipDNA Cloud API has several responses associated with it with regards to error messages. Use the following to recover more quickly based on the response message associated with the error.

Device Management

Message/Issue	**Explanation and Potential Recovery**
The Problem: "API key required."	You are not sending an API key for authentication or have not included 'Bearer' where applicable in your header. **The Solution**: Use an existing API key (found on the Security Keys page in a Merchant Gateway Account), or create one. Send a valid Merchant API key in the Authorization header as documented.
The Problem: "Missing required field: 'registration code''	You are not passing the required 'registrationCode' field in the body of your request or are not passing a value into it during device registration. **The Solution**: Use the documented parameter in the body of your request and ensure you are passing a value.
The Problem: "Missing required field: 'poi device id'"	You are not passing the required POI Device ID in your deregistration request. **The Solution**: Ensure you are passing a value in with the /devices/deregister endpoint.
The Problem: Encrypted Device service not available	The gateway account you are attempting to process on does not have the correct 'Encrypted Devices' service enabled on it. **The Solution**: Contact the provider of the gateway account to have them enable this service for you.
The Problem: "Invalid registration code. Please try again."	Occurring exclusively on the registration request, this means that the code provided in the request was not valid at the time of the request. Registration codes expire every 15 minutes, so using an old code will not be possible. **The Solution**: Look at the POI Device screen of the unregistered internet-connected POI Device and use the currently visible code on the screen.

The Problem: "Device Deregistration Unsuccessful" or "Invalid poi device id"	Whether you are attempting to deregister a device or query on it, you are not passing in the correct POI Device ID value. **The Solution**: Query against the /list endpoint without specifying an ID to get a list of all valid POI Devices and resend your request once you acquire the valid ID.
The Problem: "I am seeing my device display 'Registered POI Device' instead of the nickname I provided."	**The Solution**: Check that you are sending the proper parameter 'deviceNickname' in during registration. If you are not, the default 'Registered Device' appears instead. You can deregister your device, and re-register it with the proper parameter/nickname value.
The Problem: "I am not seeing any devices in the poiDeviceID object when I ping the /list endpoint."	**The Solution**: If the response contains no data, this means there are no registered (or deregistered) devices on the account at all. Check that you are using the correct API key for the account you wish to query against.

Processing

Message	Explanation and Potential Recovery
The Problem: "API key required."	You are not likely sending an API key for authentication. The likely culprit is the use of username/password which is not supported when using a POI Device. **The Solution**: Use an existing API key (found on the Security Keys page in a Merchant Gateway Account), or create one. Use the 'security_key' variable to pass it in and omit username/password credentials.
The Problem: "Only one authentication method may be used."	You are likely sending two types of credentials in your request and must only use one. **The Solution**: When processing with a POI device, use the 'security_key' variable via API to authenticate. Use of the username/password credential set is not supported.
The Problem: Encrypted Device service not available	The gateway account you are attempting to process on does not have the correct 'Encrypted Devices' service enabled on it. **The Solution**: Contact the provider of the gateway account to have them enable this service for you.

The Problem: "Invalid poi_device_id."	The value being provided in the poi_device_id variable is incorrect or invalid. **The Solution**: Query your POI Device estate via API or log into the Merchant Control Panel and check 'Registered Devices' in the License Manager for valid/registered POI Device IDs.
The Problem: "Communication with the POI device has timed out. Please restart transaction."	The transaction has exceeded the 300 second timeout limit and the transaction has been cancelled. **The Solution**: The transaction must be restarted by sending a new transaction request to the POI Device.
The Problem: "The POI device already in use. Please complete the current transaction and try again."	The physical device (as represented by the POI Device ID) is currently in the middle of a transaction and cannot be used. **The Solution**: Target a different POI Device ID or cancel the transaction in progress if necessary.
The Problem: "The transaction was cancelled on the POI device. Please restart transaction."	The transaction was cancelled by the user OR the POI Device itself and will not be successful. **The Solution**: Restart the transaction by sending a new transaction request to the POI Device.

Asynchronous Status Polling

Message	**Explanation and Potential Recovery**
The Problem: "API key required."	You are not sending an API key for authentication or have not included 'Bearer' where applicable in your header. **The Solution**: Use an existing API key (found on the Security Keys page in a Merchant Gateway Account), or create one. Send a valid Merchant API key in the Authorization header as documented.
The Problem: "Platform Id or API Key not found"	Either the platform ID value you've provided to the polling API is old or invalid, your API key is invalid, or your API key in use was not the key that initiated the transaction. **The Solution**: Check that your API key and platform ID provided match the API key used and platform ID value received in the original asynchronous request**.** If you are polling against an old platform ID, use the Query API to retrieve historical information on transactions. The AsyncStatus API is meant for transactions happening in the moment vs a reporting API.

Note: The vast majority of these errors will be accompanied by a REFID code which you should provide to support for troubleshooting if necessary. If you cannot recover from an error for any reason, contact Customer Support for assistance.

For AsyncStatus API issues, provide the **errorRefUUID** value to support for troubleshooting.

VPP Testing Information

ChipDNA Cloud

Overview

The Virtual Pin Pad (VPP) can be used for testing the ChipDNA Cloud API without a device. To do so, use our virtual registration codes which can be submitted to any test gateway account or gateway account that is in test mode. Keep in mind that if an account is in test mode, no actual credit card processing will take place. The VPP will simulate a Visa EMV transaction for the purposes of transaction processing and reporting visibility.

Transaction Testing Credentials

The Payment Gateway demo account can also be used for testing at any time. Please use the following api-key for testing with this account:

api-key:	2F822Rw39fx762MaV7Yy86jXGTC7sCDy

Registration

VPP Registration requests are submitted with the same API calls as a registration request for a physical device, but using the following special registration codes:

Successful Registration:	T00001
Failed Registration:	T00002

All GUIDs returned associated with the VPP (including POI Device IDs and Asynchronous Transaction GUIDs) will always end in 12 zeroes. This will be an indicator that the transaction is a ‘test’ transaction since this will never occur in a live environment.

For example: **3915af04-29fa-49df-88d0-000000000000**

Registration requests using `T00001` will return a virtual (fake) POI Device ID which can be used to process test payments and will appear in estate management polling requests. This POI Device ID can also be used to simulate deregistration of an EMV terminal.

Processing

The POI Device IDs returned via VPP registration can be used in both synchronous or asynchronous processing requests.

Process an approved transaction	Send an amount greater than or equal to 1.00
Process a declined transaction	Send an amount less than 1.00

When using Synchronous processing, you can expect to wait a few seconds before receiving a response. The VPP does not simulate the full five (5) minute wait time, but you will want to plan ahead to expect longer wait times for customer interaction.

VPP AsyncStatus GUIDs will only simulate an immediate ‘interactionComplete’ response versus remaining ‘inFlight’ for customer interaction.

If you would like to test Customer Vault interaction with VPP, simply send the standard customer vault action variables along with your transaction request.

Transaction Contents for VPP Transactions

- Card Brand: Visa
- Card Number: 4111111111111111
- Expiration Date: 0229
- Full Name: JOHN SMITH
    - Cardholder Verification Method: Pin Entered
    - Application ID: A0000000031010
    - Application Label: VISA CREDIT
    - Application Preferred Name: CREDITO DE VISA
    - Transaction Status Info: E800
    - PAN Sequence Number: 01
    - Masked MID: XXXXXXXXXX1234
    - Auth Code: 123456

Deregistering VPP devices

The POI Device ID can be used to deregister the VPP device - exactly like a real POI Device ID.

Triggering Errors in Test Mode

- To cause a declined response, pass an amount less than 1.00.
- To trigger a fatal error message in processing, pass an invalid POI Device ID.
- To cause an unknown AsyncStatus GUID, pass an invalid GUID.
- To cause an empty Estate Management response, pass an invalid GUID.

See Registration and Deregistration, Estate Management, Transaction Processing, and AsyncStatus API documentation for more information on these concepts.

VPP does not support handling POI device prompts at this time, though sending the variables in your request will not cause errors. These prompts require interaction with a physical device though have very little effect on the final responses or API calls.