OSC

Online Services Corp™ Product Services Guide for Prepaid Card Issuers

OSC System™ OEM Issuing Platform

Product Services Guide for Prepaid Card Issuers Ver 1.03 Rel 27.03.25

2025

IMPORTANT NOTICES

Online Services Corp owns the intellectual property in this document exclusively. You acknowledge that you must not perform any act which infringes the copyright or any other intellectual property rights of Online Services Corp and cannot make any copies of this Manual unless in accordance with these terms and conditions.

Without our express written consent, you must not:

- Distribute any information contained in this Manual to the public media or quote or use such information in the public media; or

- Allow access to the information in this Manual to any company, firm, partnership, association, individual, group of individuals or other legal entity other than your officers, directors and employees who require the information for purposes directly related to your business.

The software described in this Manual is supplied under a license agreement and may only be used in accordance with the terms of that agreement.

Online Services Corp, OSC System and the OSC and Online Services Corp logos are trademarks of Online Services Corp.

All third-party product and service names are trademarks or registered trademarks of their respective owners.

SUMMARY OF CHANGES

Not Applicable

RELATED DOCUMENTATION

The following documents and manuals provide information related to the subjects described in this guide.

- *The Issuing Agreement for Prepaid Card Issuers*
- *Online Services Corp Prepaid Card Issuing Overview*

1. OVERVIEW

Online Services Corp has implemented a prepaid card issuing system to enable issuers, payment processors and payment service providers to issue prepaid UnionPay payment cards. This document is specific to EC merchants only.

The service is known as the **OSC System** (or OSC) and provides prepaid payment and account management functions and an on-line administrative platform to a web hosted application.

The purpose of the OSC System is to provide a high function card service that simplifies the card issuing process.

OSC SYSTEM BENEFITS

The OSC System provides an outsourced proposition to issuers, merchants and payment processors (collective referred to as *Payment Service Providers* or PSPs in this document) for card issuing providing:

Significant cost savings – costs incurred for redundant hardware, software, telecommunications and internal operational overheads for an in-house system are avoided.

Faster time to market – OSC makes it possible for a user to be "live" with their service within a much shorter time frame than usual.

Reduced risk – OSC clients avoid many of the risks associated with an in-house system such as high initial and on-going costs; recruitment of skilled technical resources; data security requirements and ability to stay up to date with the latest industry requirements, compliance and trends.

Business transparency – OSC is an "out-sourced" packaged service running on Amazon Web Services. All costs are clear to the merchant or processor and agreed up front.

Reduced burden of issuer administration - OSC provides a web-based issuer administration portal for issuers and processors to manage their payment transactions. The system allows issuers to report on transaction types and perform financial transactions on-line.

Wide applicability - OSC can be used to enable individual issuers regardless of size and number of transactions. It is an ideal solution for the small to medium issuers and merchants; but can also be readily tailored to suit larger enterprises to service their clients.

Rapid issuer payment enablement - OSC deploys a flexible connection mechanism based on 'web-services' communication.

Accepted payment methods - OSC currently supports credit and debit card issuing for UnionPay cards. UnionPay cards are accepted in over 180 countries and territories with more than 10 Million online and e-commerce merchants.

Secure payments - Data passed between the issuer and OSC is encrypted using RSA industry standard cryptography.

Future proofing - OSC will continue to be compliant with new industry initiatives.

Products Provided to Issuers

The card products provided to issuers and processors by OSC through the hosted platform are:

1. Virtual, anonymous, disposable UnionPay branded cards
    - *A virtual card is a UnionPay account issued without a physical plastic card*
    - *The issuer and cardholder are provided with a Card Number, Expiry Date and CVC*
    - *A disposable card can only be loaded once*
    - *When the balance of the account is $0, the card can no longer be used*
    - *The cardholder is not linked to the card – it is anonymous*
    - *No KYC is required, and maximum load values apply*
    - *Anonymous cards are treated like cash - stolen or lost cards cannot be refunded; chargebacks are not accepted; refunds need to raised with the merchant directly*

2. OSC ONLINE PORTAL

The *Issuer Administration* functions are supplied by providing the issuer or processor with secure browser access to the OSC central transaction database to perform day-to-day functions. The user is provided with tools for searching, reporting and editing of client accounts and transactions.

3. CREATING AND LOADING CARD ACCOUNTS

APPLYING AND LOADING

Issuers and PSPs issue and load Card Accounts on the OSC System using a two-step process.

1. **Apply** – Issuers start by applying for a batch of cards. The number of cards in a batch can be up to 100, if the issuer wishes to issue a single card they can apply for a batch of 1 card. Batches are created by the OSC System in response to a request or API call. Once a batch has been created, it can be downloaded as a CSV file containing the card account data.
2. **Load** – Once card accounts have been successfully allocated they can be loaded. Each account is identified on the system by a card token, which is used in the request or API call to load cards.

The description of the API calls to apply for and load card accounts is detailed below.

ISSUER BALANCE

Each issuer or PSP maintains a balance with their aggregator or OSC. This balance is funded by the issuer before they can load any card accounts; once the balance has been established the issuer is able to load value onto their card accounts until the balance has been reduced to zero.

When the balance is completely depleted the issuer will need to top up their account in order to load value onto their card accounts.

The process for loading their accounts and managing their balances is explained to individual issuers when they contract to issue card accounts on the OSC System.

4. THE OSC API INTERFACE

The issuer and card management are accessible to issuers and PSPs using a Web-Services style connection mechanism, where messages are exchanged using JSON encoded resources via RESTful APIs over an encrypted channel. This ensures a simpler integration and allows OSC to support several integration environments, such as Microsoft COM and .NET, JSP and PHP, etc.

The following sections describe the API requests PSPs can initiate to manage their cardholders, and enable payments through the OSC *Payment Management and Processing* service.

OSC maintains two separate systems – test and production, with different URLs. The endpoint for your API request should be set accordingly. While this document will use the web address *https://www.onlineservicescorp.com/issuer/*, the actual endpoint address may be different and will be provided at sign up.

API requests are made by

- sending data to the OSC System Gateway for action, for example to allocate new cards on the cardholders management system. These API calls are made using POST or PUT methods.

- requesting data from the OSC System Gateway, for example to retrieve cardholder details or card information from the issuer client management system or query payment transactions. These API calls are made using the GET method.

THE OSC SECURITY KEYS

The OSC System maintains unique security keys for each issuer or PSP connected to the system.

The Issuer RSA Keys

Issuer Signature Keys – the issuer RSA signature keys are used solely to sign and verify API requests and responses sent to, and received from, the OSC System. Two key pairs are used to sign and verify data exchanged between the issuer and the OSC System.

The issuer or PSP will create a Private/Public RSA key pair and send the Public key to OSC. This Private key will be used to sign data sent to the OSC System; the signature is verified using the issuer's Public key.

OSC will provide the issuer with a Public key which is used to verify the signature sent to the issuer from the OSC System, where it was created using the OSC Private key.

SHA-256 is used to generate a hash value for the data to be signed, the hash is then signed using RSA with a Private key. The signature is then based64 encoded.

Signatures are base64 decoded before being verified using RSA with SHA256.

Issuer Data Key - The issuer or PSP will create a unique Private/Public RSA key pair and send the Public key to OSC. This Public key will be used to encrypt sensitive data such as card account numbers, sent from the OSC System; the issuer decrypts the data using their related Private key.

Data is RSA encrypted using a Public key and then base64 encoded.

The encrypted data should be base64 decoded before being decrypted using a Private key.

The RSA keys used on the test and production platforms are different and unique and cannot be interchanged.

The use of the signature and encryption keys is described below.

An example of PHP sample source code to sign, verify, encrypt and decrypt data can be found by clicking on the source code icon below. Note that this is an example of how RSA functions can be used, it is a not a recommendation for how to implement cryptography in your own applications.

The Merchant API Key – the merchant API key is used to authenticate the issuer on all API calls made to the OSC System, and is provided to the issuer at sign up. This key is common across the test and production platforms for each issuer; but unique to the issuer within the system.

The merchant API key authenticates the origin of each API call through its inclusion in the request header "x-api-key" of the API request.

For example, in JavaScript the usage of the merchant API key may be as follows:

```
// Allocate a Request Object

xhttp = new XMLHttpRequest();

// Set the URL and request headers

xhttp.open( method, endpoint ...);
xhttp.setRequestHeader( "x-api-key", unique_API_key );
```

The issuer RSA keys, and the API key should be kept secure.

OSC API RESPONSES

All OSC API Responses include a status code, signifying whether the call succeeded. Additionally, the response will usually include relevant data or a status message and potentially more detailed information.

The data retrieved from the server is in a JSON object.

Format of a Successful API Request

An example of a successful OSC API request

```
{
"status": 200,
"response":
    {
    "message":"SUCCESSFUL"
    },
“Optional Data”: {}
}
```

Format of an Unsuccessful API Request

An example of the JSON object sent in response to an unsuccessful OSC API request is as follows.

```
{
"status": 400,
"response":
    {
    "message": "Error Message",
    "detail": "Optional More Data"
    }
}
```

An example of executable JavaScript sample source code to manage the data sent from the server in response to an unsuccessful request can be found by clicking on the source code icon below. Note that this is an example of how an error condition can be processed, it is a not a recommendation for how to implement API processing in your own applications.

COMMON API DATA STRUCTURES

This section defines the data exchanged between the issuer and the OSC System, and stored on the OSC System,

Not all the elements are required for every API call, and some are set by the OSC System. The following tables describe which elements are mandatory for some calls, are optional or read only (as described below in the specific API calls).

R: System supplied, read-only.
M: Mandatory.
O: Optional.

Issuer Data

Each issuer or PSP has a record maintained in the OSC System; this data structure includes the data stored, and the details can be viewed and updated using the API calls described in the documentation below.

Name	Description	R/M/O
address	The business address of the issuer	M
aggregatorID	The ID of the entity with the issuer relationship	R
aggregatorName	The name of the entity with the issuer relationship	R
apiKey	The issuer unique API key (described above)	R
balance	The amount of funds remaining in the issuer account with OSC	R
billingCurrency	The currency in which issuers are billed and settled	M
contact	The name of the primary contact at the issuer	M
created	The date the issuer was setup on the OSC System	R
email	The email address of the issuer primary contact	M
encryptPublicKey	The issuer's RSA Public key used to encrypt sensitive data sent from the OSC System	R
issuerID	The unique ID allocated to the issuer	R
lastUpdate	The date when the issuer record was last updated	R

loadCommission	The fee paid by the cardholder to load value onto their cards (fixed amount)	R
loadFee	The commission paid by the issuer on value loaded onto cardholder cards (percentage)	R
name	The business name of the issuer	M
parent	The ID of the parent organisation of the issuer (if any)	R
shortName	An abbreviation of the issuer business name	M
phone	The phone number of the issuer primary contact	O
signaturePublicKey	The issuer's RSA Public key used to sign data sent from the OSC System	R
status	The status of the issuer can be one of : *Inactive* or *Active*	R
statusDate	The date the status of the issuer was last changed	R
techContact	The name of the issuer's technical contact	O
techEmail	The email of the issuer's technical contact	O
techPhone	The email of the issuer's technical contact	O
url	The address of the issuer's website	O

An example of JavaScript sample source code to create an issuer data structure can be found by clicking on the source code icon below. Note that this is an example of how an issuer data object is structured, it is a not a recommendation for how to implement the APIs in your own applications.

Cardholder Account Data

Each account or card issued has a record maintained in the OSC System; this data structure includes the account data stored, which can be managed using the API calls described in the documentation below.

Card Data

Sensitive card data is included in the Cardholder Account Data and is send from the OSC System to the issuer encrypted using the issuer's Public key. The format of this data is described in the following table.

Name	Description	R/M/O
cvc	The Card Verification Code, often used to authenticate the payment card (a 3 digit number)	R
pan	The Primary Account Number, like that embossed on the front of physical payment cards	R

The structure of the complete Cardholder Account Data is as follows.

Name	Description	R/M/O
added	The date the account was allocated to the issuer	R
batchID	The ID of batch or set of account of which this account is part. A batch may contain any number of account	M
cardData	The card PAN and CVC as described above (Card Data)	R
cardMedium	The type of card issued, always *VIRTUAL* in the case of virtual, disposable UnionPay cards	M
cardState	The status of the account, one of *TO__BE__ACTIVED* for accounts which have been allocated but not loaded or *ACTIVE-LOADED* for accounts which have been loaded	R
cardToken	A unique identifier which identifies the card in the OSC System	R
currency	The 3 digit ISO 4217 numeric currency code for the account's currency	M

expiryDate	The account's expiry date, sometimes also used in online payments	R
loadedBalance	The amount of funds in *currency* loaded on the card	R
maskedPan	A partial view of the Primary Account Number comprising the first 6 and last 4 digits	R

An example of JavaScript sample source code to create a card data structure can be found by clicking on the source code icon below. Note that this is an example of how a card data object is structured, it is a not a recommendation for how to implement the APIs in your own applications.

ISSUER DATA MANAGEMENT INTERFACE CALLS

The following interface calls are available to the issuer or PSP from their applications to access the Issuer Management functions:

1. **Get Issuer Information** – used to return a specific issuer's details stored in the OSC database.
2. **Update Issuer Information** – used to update an existing issuer's information or add issuer information to an existing issuer entry in the OSC database.

Calls are initiated through RESTful APIs to the OSC System. Each API request includes an *action* field which specifies the relevant API call, and all API calls are directed at the same test or production URL.

The data required for all issuer related functions is an *Issuer Object* (described above), serialised as a JSON string. A OSC issuer object holds the details associated with an individual issuer on the system.

The Get Issuer API Call

This call is used to get an issuer's details as they are stored on OSC issuer database.

URLs:

Test
https://www.onlineservicescorp.com/issuer/
Production
https://www.onlineservicescorp.com/issuer/

Request Method: POST

Headers:

The header *'x-api-key'* should be set to the value of the Issuer API Key allocated to the issuer or PSP at sign up.

URL Parameters: None
Action: LIST-ISSUER

Get Issuer Request Data

Name	Description	R/M/O
action	LIST_ISSUER	M
payload	The issuer specific data for this call, described below	M
sign	The RSA signature as specified under *The OSC Security Keys*	M

Include the following in the payload field.

Name	Description	R/M/O
aggregatorID	The issuer's unique aggregator ID provided at sign up, allocated by the OSC system	M
issuerID	The issuer's unique ID provided at sign up and stored on the OSC server	M

The *Get Issuer Request* should be sent to the OSC server formatted as a JSON object, using the POST method.

Get Issuer API Call Response

The OSC server will respond including a status code, signifying whether the call succeeded, as described above - *OSC API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The call completed successfully, and the issuer details are returned	*Message* - SUCCESSFUL
400	The aggregator or issuer IDs were invalid or not recognised, and the request failed	*Message* - Invalid Request
400	The signature attached to the request could not be verified	*Message* - Signature verification failed
400	The x-api-key in the message header was not verified	*Message* – Invalid issuer api key
500	A server error occurred, and the request failed	*Message* – Internal processing exception

If the call is successful, the response will include the following.

Name	Description
Message	The response message from the OSC System
payload	The issuer returned data, described below
sign	The RSA signature as specified under *The OSC Security Keys*

The issuer data included in the *payload* field.

Name	Description
issuer	An issuer object, as described above – *Issuer Data*, some of the fields may not be set

Get Issuer - Sample Source Code

An example of JavaScript sample source code to get an issuer can be found by clicking on the source code icon below. Note that this is an example of how a get issuer request can be initiated, it is a not a recommendation for how to implement the APIs in your own applications.

The Update Issuer API Call

This call is used to update an existing issuer's information or add new issuer information to an existing issuer entry in the OSC database.

URLs:

Test
https://www.onlineservicescorp.com/issuer/
Production
https://www.onlineservicescorp.com/issuer/

Request Method: POST

Headers:

The header '*x-api-key*' should be set to the value of the Issuer API Key allocated to the issuer or PSP at sign up.

URL Parameters: None
Action: UPDATE-ISSUER

Update Issuer Request Data

Name	Description	R/M/O
action	UPDATE_ISSUER	M
payload	The issuer data for this call, described below	M
sign	The RSA signature as specified under *The OSC Security Keys*	M

Include the following in the *payload* field.

Name	Description	R/M/O
aggregatorID	The issuer's unique aggregator ID provided at sign up, allocated by the OSC system	M
issuerID	The issuer's unique ID provided at sign up and stored on the OSC server	M
fields	The issuer data to update in an issuer object described above - *Issuer Data.* Include only those field which need to be updated or added	M

The new or updated *issuer data* should be sent to the OSC server formatted as a JSON object, using the POST method.

Update Issuer API Call Response

The OSC server will respond including a status code, signifying whether the call succeeded, as described above - *OSC API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The call completed successfully, and the issuer details have been updated	*Message* - SUCCESSFUL
400	The aggregator or issuer IDs were invalid or not recognised, and the request failed	*Message* - Invalid Request
400	The signature attached to the request could not be verified	*Message* - Signature verification failed
400	The x-api-key in the message header was not verified	*Message* – Invalid issuer api key
400	The issuer data included *Read Only* fields to be added or updated	*Message* - Not allowed to update protected field
500	A server error occurred, and the request failed	*Message* – Internal processing exception

If the call is successful, the response will include the following.

Name	Description
Message	The response message from the OSC System
payload	The issuer returned data, described below
sign	The RSA signature as specified under *The OSC Security Keys*

The issuer data is included in the *payload* field.

Name	Description
issuer	An issuer object, as described above – *Issuer Data*, with the updated issuer data as stored on the OSC issuer database

Update Issuer - Sample Source Code

An example of JavaScript sample source code to update an issuer can be found by clicking on the source code icon below. Note that this is an example of how an update issuer request can be initiated, it is a not a recommendation for how to implement the APIs in your own applications.

5. CARD ACCOUNT MANAGEMENT

The OSC prepaid card management system supports UnionPay international branded virtual, anonymous, disposable cards.

As described in the *Creating and Loading Card Accounts* section above, issuers and PSPs need to have funds deposited in their balance before they can load value unto their prepaid cards.

Issuers can apply for any number of prepaid cards and once these have been allocated, they can be loaded with values up to the maximum defined for the issuer. The card account management APIs described below provide issuers with the ability to apply for, load and manage their cardholders' accounts.

The virtual cards consist of a standard UnionPay 16 digit account number, a Card Verification Code, also known as a Card Verification Number (CVN) and an expiry date. There is no physical plastic issued with this type of account, and they can be used online and at MOTO (mail order telephone order) merchants which accept UnionPay cards.

Once the value in a disposable card account has been depleted, it cannot be reloaded, when a card account which has been loaded has a zero balance it automatically expires.

Anonymous cards are treated like cash - stolen or lost cards cannot be refunded, chargebacks are not accepted and refunds need to raised with the merchant rather than the issuer, PSP or OSC.

SETTLEMENT AND CURRENCY OPTIONS

Settlement

Settlement is the process whereby funds are transferred from cardholder accounts to the merchants via the card scheme (UnionPay in this case) and the merchant's bank. Once funds have been loaded into a cardholder's account, OSC will manage the settlement process on behalf of the issuer.

Currency

Issuers will be allocated a currency in which the funds in the card accounts are denominated. The OSC System allows issuers and PSPs to specify options for :

1. **Account Currency** – This is the currency of the funds loaded into the issuer's card accounts.
2. **Billing Currency** – This is the currency the issuer or PSP will be billed in by their aggregator.

CARD ACCOUNT MANAGEMENT INTERFACE CALLS

The following interface calls are available to the issuer or PSP from their platform to access the OSC card account functions:

1. **Allocate** – request for a number (between 1 and 100) of card accounts from OSC. When a batch of cards are successfully applied for, they will not be loaded with value.
2. **Load** – load a number (between 1 and 100) of cards with different values. Before loading the cards must have been allocated using the allocate call.
3. **Allocate and Load** – issuers who need a number (between 1 and 100) of cards loaded with the same value can use this call to allocate and load the cards required in one API request.
4. **Get Cards** – retrieve all or a subset of an issuer's card accounts from the OSC card account database.
5. **Get Card** – retrieve a specific card account's details from the OSC card account database.
6. **Retrieve Batch** – gets the data of a batch of cards which have previously been allocated and/or loaded from the OSC card account database.

Calls are initiated through RESTful APIs to the OSC System. Each API request includes an *action* field which specifies the relevant API call, and all API calls are directed at the same test or production URL.

The Allocate Card Accounts API Call

This call is used to allocate a number of card accounts, these accounts are exclusively issued to the issuer or PSP and can subsequently be loaded using the load API request.

URLs:

Test
https://www.onlineservicescorp.com/issuer/
Production
https://www.onlineservicescorp.com/issuer/

Request Method: POST
Action: APPLY

Headers:

The header *'x-api-key'* should be set to the value of the Issuer API Key allocated to the issuer or PSP at sign up.

URL Parameters: None

Name	Description	R/M/O
action	APPLY	M
payload	The request data for this call, described below	M
sign	The RSA signature as specified under *The OSC Security Keys*	M

Include the following in the *payload* field.

Name	Description	R/M/O
allocateNumber	The number of card accounts to allocate, a number between 1 and 100	M
batchID	A unique identifier, supplied by the issuer or PSP used to ID this batch of card accounts in the OSC System	M
cardMedium	The type of card requested, always set to VIRTUAL for the UnionPay virtual cards	M
issuerID	The issuer's unique ID provided at sign up and stored on the OSC server	M

reloadable	Flags whether the card is capable of being reloaded. Always set to N for disposable cards	M

The allocate card account request data should be sent to the OSC server formatted as a JSON object, using the POST method.

Allocate Card Accounts API Call Response

The OSC server will respond including a status code, signifying whether the call succeeded, as described above - *OSC API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The call completed successfully, and the requested card accounts have been allocated	*Message* - SUCCESSFUL
400	The issuer ID or another parameter in the request was invalid or not recognised	*Message* - Invalid Request
400	The signature attached to the request could not be verified	*Message* - Signature verification failed
400	The x-api-key in the message header was not verified	*Message* – Invalid issuer api key
500	A server error occurred, and the request failed	*Message* – Internal processing exception

If the call is successful, the response will include the following.

Name	Description
Message	The response message from the OSC System
payload	The returned card account data, described below
sign	The RSA signature as specified under *The OSC Security Keys*

The card account data is included in the *payload* field.

Name	Description
cards	A JSON array of card account records, as described below – *Card Account Records*

Each of the *Card Account Records* are structured as follows:

Name	Description
batchID	The unique batch identifier, supplied by the issuer or PSP in the allocate card accounts request
cardData	The card PAN and CVC, encrypted using the issuer Public key
cardMedium	The type of card as requested by the issuer and allocated by the OSC System
cardState	The status of the account, *TO__BE__ACTIVED* for accounts which have been allocated but not loaded
cardToken	A unique identifier which identifies the card in the OSC System
expiryDate	The account's expiry date, in the format MM/YY

Allocate Card Accounts - Sample Source Code

An example of JavaScript sample source code to allocate card accounts can be found by clicking on the source code icon below. Note that this is an example of how an allocate card accounts request can be initiated, it is a not a recommendation for how to implement the APIs in your own applications.

The Load Card Accounts API Call

This call is used to load a number of card accounts, these accounts must have previously been allocated using the allocate card accounts API call.

URLs:

Test
https://www.onlineservicescorp.com/issuer/
Production
https://www.onlineservicescorp.com/issuer/

Request Method: POST
Action: LOAD

Headers:

The header *'x-api-key'* should be set to the value of the Issuer API Key allocated to the issuer or PSP at sign up.

URL Parameters: None

Name	Description	R/M/O
action	LOAD	M
payload	The card load data for this call, described below	M
sign	The RSA signature as specified under *The OSC Security Keys*	M

Include the following in the *payload* field.

Name	Description	R/M/O
batchID	A unique identifier, supplied by the issuer or PSP used to ID this batch of card accounts in the OSC System	M
currency	The currency of the value to be loaded to the cards	M
issuerID	The issuer's unique ID provided at sign up and stored on the OSC server	M
cards	A JSON array of card load records, as described below – *Card Load Records*	M

Each of the *Card Load Records* are structured as follows:

Name	Description
amount	The amount to be loaded to the card in the specified currency
cardToken	The unique card account identifier which identifies the card allocated by the OSC System

The load card account request data should be sent to the OSC server formatted as a JSON object, using the POST method.

Load Card Accounts API Call Response

The OSC server will respond including a status code, signifying whether the call succeeded, as described above - *OSC API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The call completed successfully, and the card accounts have been loaded	*Message* - SUCCESSFUL
400	The issuer ID or another parameter in the request was invalid or not recognised	*Message* - Invalid Request
400	The signature attached to the request could not be verified	*Message* - Signature verification failed
400	The x-api-key in the message header was not verified	*Message* – Invalid issuer api key
500	A server error occurred, and the request failed	*Message* – Internal processing exception

If the call is successful, the response will include the following.

Name	Description
Message	The response message from the OSC System
payload	The returned card account data, described below

sign	The RSA signature as specified under *The OSC Security Keys*

The card account data is included in the *payload* field.

Name	Description
cards	A JSON array of card account records, as described below – *Card Account Records*

Each of the *Card Account Records* are structured as follows:

Name	Description
batchID	The unique batch identifier, supplied by the issuer or PSP in the allocate card accounts request
currency	The currency of the value loaded to the cards, requested in the card load request
cardState	The status of the account, *ACTIVE-LOADED* for accounts which have been successfully loaded
cardToken	The unique card account identifier which identifies the card allocated by the OSC System
loadedBalance	The amount of funds in the card account after the load, taking into account fees and rebates

Load Card Accounts - Sample Source Code

An example of JavaScript sample source code to load card accounts can be found by clicking on the source code icon below. Note that this is an example of how a load card accounts request can be initiated, it is a not a recommendation for how to implement the APIs in your own applications.

The Allocate and Load Card Accounts API Call

This call is used to load a number of card accounts with the same amount, the accounts will be allocated and then loaded, and there is no need to use the allocate card accounts or load card accounts API calls.

URLs:

Test
https://www.onlineservicescorp.com/issuer/
Production
https://www.onlineservicescorp.com/issuer/

Request Method: POST
Action: APPLY-LOAD

Headers:

The header '*x-api-key*' should be set to the value of the Issuer API Key allocated to the issuer or PSP at sign up.

URL Parameters: None

Name	Description	R/M/O
action	APPLY-LOAD	M
payload	The card data for this call, described below	M
sign	The RSA signature as specified under *The OSC Security Keys*	M

Include the following in the *payload* field.

Name	Description	R/M/O
allocateNumber	The number of card accounts to allocate, a number between 1 and 100	M
amount	The amount to be loaded to the card accounts in the specified currency. All card accounts will be loaded with this same amount	M
batchID	A unique identifier, supplied by the issuer or PSP used to ID this batch of card accounts in the OSC System	M

cardMedium	The type of card requested, always set to VIRTUAL for the UnionPay virtual cards	M
currency	The currency of the value to be loaded to the cards	M
issuerID	The issuer's unique ID provided at sign up and stored on the OSC server	M
reloadable	Flags whether the card is capable of being reloaded. Always set to N for disposable cards	M

The allocate and load card accounts request data should be sent to the OSC server formatted as a JSON object, using the POST method.

Allocate and Load Card Accounts API Call Response

The OSC server will respond including a status code, signifying whether the call succeeded, as described above - *OSC API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The call completed successfully, and the card accounts have been allocated and loaded	*Message* - SUCCESSFUL
400	The issuer ID or another parameter in the request was invalid or not recognised	*Message* - Invalid Request
400	The signature attached to the request could not be verified	*Message* - Signature verification failed
400	The x-api-key in the message header was not verified	*Message* – Invalid issuer api key
500	A server error occurred, and the request failed	*Message* – Internal processing exception

If the call is successful, the response will include the following.

Name	Description
Message	The response message from the OSC System
payload	The returned card account data, described below
sign	The RSA signature as specified under *The OSC Security Keys*

The card account data is included in the *payload* field.

Name	Description
cards	A JSON array of card account records, as described below – *Card Account Records*

Each of the *Card Account Records* are structured as follows:

Name	Description
batchID	The unique batch identifier, supplied by the issuer or PSP in the allocate card accounts request
cardData	The card PAN and CVC, encrypted using the issuer Public key
cardMedium	The type of card as requested by the issuer and allocated by the OSC System
cardState	The status of the account, *ACTIVE-LOADED* for accounts which have been allocated and loaded
cardToken	A unique identifier which identifies the card in the OSC System
currency	The currency of the value loaded to the cards, requested in the card load request
expiryDate	The account's expiry date, in the format MM/YY
loadedBalance	The amount of funds in the card account after the load, taking into account fees and rebates

Allocate and Load Card Accounts - Sample Source Code

An example of JavaScript sample source code to allocate and load card accounts can be found by clicking on the source code icon below. Note that this is an example of how a

allocate and load card accounts request can be initiated, it is a not a recommendation for how to implement the APIs in your own applications.

The Get Card Accounts API Call

This call is used to retrieve a subset, based on date of allocation, of an issuer's card accounts.

URLs:

Test
https://www.onlineservicescorp.com/issuer/
Production
https://www.onlineservicescorp.com/issuer/

Request Method: POST
Action: LIST-CARDS

Headers:

The header *'x-api-key'* should be set to the value of the Issuer API Key allocated to the issuer or PSP at sign up.

URL Parameters: None

Name	Description	R/M/O
action	LIST-CARDS	M
payload	The request parameters for this call, described below	M
sign	The RSA signature as specified under *The OSC Security Keys*	M

Include the following in the *payload* field. If *dateStart* is not set it defaults to the date of the first card account allocated to the issuer on the system. If *dateEnd* is not set it defaults to the date of the most recent card account allocated on the system. If both dates are not set all the card accounts for the issuer will be retrieved.

Name	Description	R/M/O
dateStart	The start date of the subset of cards to retrieve. This date refers to the date the card account was allocated and is in yyyy-mm-dd format	O
dateEnd	The end date of the subset of cards to retrieve. This date refers to the date the card account was allocated and is in yyyy-mm-dd format	O

issuerID	The issuer's unique ID provided at sign up and stored on the OSC server	M

The get card accounts request data should be sent to the OSC server formatted as a JSON object, using the POST method.

Get Card Accounts API Call Response

The OSC server will respond including a status code, signifying whether the call succeeded, as described above - *OSC API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The call completed successfully, and the card account data is returned	*Message* - SUCCESSFUL
400	The issuer ID or another parameter in the request was invalid or not recognised	*Message* - Invalid Request
400	The signature attached to the request could not be verified	*Message* - Signature verification failed
400	The x-api-key in the message header was not verified	*Message* – Invalid issuer api key
500	A server error occurred, and the request failed	*Message* – Internal processing exception

If the call is successful, the response will include the following.

Name	Description
Message	The response message from the OSC System
payload	The returned card account data, described below
sign	The RSA signature as specified under *The OSC Security Keys*

The card account data is included in the *payload* field.

Name	Description
cards	A JSON array of card account records, as described below – *Card Account Records*

Each of the *Card Account Records* are structured as follows:

Name	Description
added	The date and time the card account was first allocated
cardDetails	The card PAN and CVC, encrypted using the issuer Public key
cardState	The status of the account, *TO__BE__ACTIVED* for accounts which have been allocated but not loaded, *ACTIVE-LOADED* for accounts which have been loaded
cardToken	A unique identifier which identifies the card in the OSC System
expiryDate	The account's expiry date, in the format MM/YY
maskedPan	The card account number with the 7th to 12th digits replaced by '*' for e.g. 628888******8888

Get Card Accounts - Sample Source Code

An example of JavaScript sample source code to get card accounts can be found by clicking on the source code icon below. Note that this is an example of how a get card accounts request can be initiated, it is a not a recommendation for how to implement the APIs in your own applications.

The Get Card Account API Call

This call is used to retrieve a specific card account from the OSC card account database.

URLs:

Test
https://www.onlineservicescorp.com/issuer/
Production
https://www.onlineservicescorp.com/issuer/

Request Method: POST
Action: LIST-CARD

Headers:

The header '*x-api-key*' should be set to the value of the Issuer API Key allocated to the issuer or PSP at sign up.

URL Parameters: None

Name	Description	R/M/O
action	LIST-CARD	M
payload	The request parameters for this call, described below	M
sign	The RSA signature as specified under *The OSC Security Keys*	M

Include the following in the *payload* field.

Name	Description	R/M/O
issuerID	The issuer's unique ID provided at sign up and stored on the OSC server	M
cardToken	The unique identifier allocated to the card account when the account was first created	M

The get card account request data should be sent to the OSC server formatted as a JSON object, using the POST method.

Get Card Accounts API Call Response

The OSC server will respond including a status code, signifying whether the call succeeded, as described above - *OSC API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The call completed successfully, and the card account data is returned	*Message* - SUCCESSFUL
400	The issuer ID or another parameter in the request was invalid or not recognised	*Message* - Invalid Request
400	The signature attached to the request could not be verified	*Message* - Signature verification failed
400	The x-api-key in the message header was not verified	*Message* – Invalid issuer api key
500	A server error occurred, and the request failed	*Message* – Internal processing exception

If the call is successful, the response will include the following.

Name	Description
Message	The response message from the OSC System
payload	The returned card account data, described below
sign	The RSA signature as specified under *The OSC Security Keys*

The card account data is included in the *payload* field.

Name	Description
cards	A JSON array with one element which contains the requested card account's record, as described below – *Card Account Record*

The *Card Account Record* is structured as follows:

Name	Description
added	The date and time the card account was first allocated
cardDetails	The card PAN and CVC, encrypted using the issuer Public key
cardState	The status of the account, *TO__BE__ACTIVED* for accounts which have been allocated but not loaded, *ACTIVE-LOADED* for accounts which have been loaded
cardToken	A unique identifier which identifies the card in the OSC System
expiryDate	The account's expiry date, in the format MM/YY
maskedPan	The card account number with the 7th to 12th digits replaced by '*' for e.g. 628888******8888

Get Card Account - Sample Source Code

An example of JavaScript sample source code to get a card account can be found by clicking on the source code icon below. Note that this is an example of how a get card account request can be initiated, it is a not a recommendation for how to implement the APIs in your own applications.

The Retrieve Batch API Call

This call is used to retrieve a batch of cards previously created using the APPLY, LOAD and APPLY-LOAD API calls.

URLs:

Test
https://www.onlineservicescorp.com/issuer/
Production
https://www.onlineservicescorp.com/issuer/

Request Method: POST
Action: RETRIEVE-BATCH

Headers:

The header *'x-api-key'* should be set to the value of the Issuer API Key allocated to the issuer or PSP at sign up.

URL Parameters: None

Name	Description	R/M/O
action	RETRIEVE-BATCH	M
payload	The request parameters for this call, described below	M
sign	The RSA signature as specified under *The OSC Security Keys*	M

Include the following in the *payload* field.

Name	Description	R/M/O
batchID	The unique batch identifier of the batch of card accounts to retrieve	M
issuerID	The issuer's unique ID provided at sign up and stored on the OSC server	M

The retrieve batch request data should be sent to the OSC server formatted as a JSON object, using the POST method.

Retrieve Batch API Call Response

The OSC server will respond including a status code, signifying whether the call succeeded, as described above - *OSC API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The call completed successfully, and the card account data is returned	*Message* - SUCCESSFUL
400	The batch ID or another parameter in the request was invalid or not recognised	*Message* - Invalid Request
400	The signature attached to the request could not be verified	*Message* - Signature verification failed
400	The x-api-key in the message header was not verified	*Message* – Invalid issuer api key
500	A server error occurred, and the request failed	*Message* – Internal processing exception

If the call is successful, the response will include the following.

Name	Description
Message	The response message from the OSC System
payload	The returned batch data, described below
sign	The RSA signature as specified under *The OSC Security Keys*

The batch data is included in the *payload* field.

Name	Description
cards	A JSON array of card account records, as described below – *Card Account Records*

Each of the *Card Account Records* are structured as follows:

Name	Description
batchID	The unique batch identifier for this batch of card accounts
cardData	The card PAN and CVC, encrypted using the issuer Public key
cardMedium	The type of card as requested by the issuer and allocated by the OSC System
cardState	The status of the account, *TO__BE__ACTIVED* for accounts which have been allocated but not loaded, *ACTIVE-LOADED* for accounts which have been loaded
cardToken	A unique identifier which identifies the card account in the OSC System
currency	The currency of the value loaded to the card account
expiryDate	The card account's expiry date, in the format MM/YY
loadedBalance	The amount of funds in the card account

Retrieve Batch - Sample Source Code

An example of JavaScript sample source code to retrieve a batch can be found by clicking on the source code icon below. Note that this is an example of how a retrieve batch request can be initiated, it is a not a recommendation for how to implement the APIs in your own applications.