Master Virtual Card Guide

Version: 1.1

10 July 2025

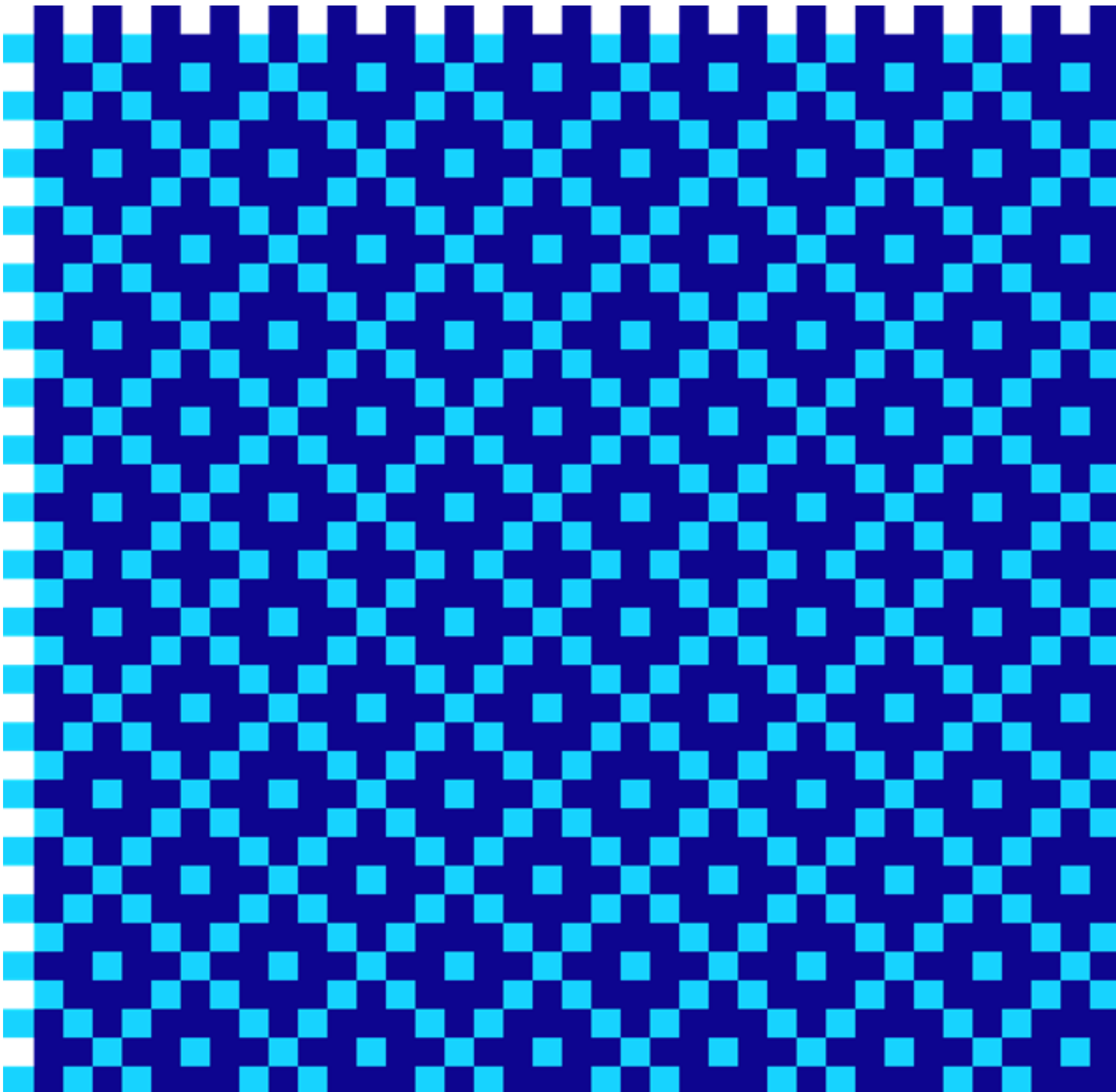
Publication number: MVCG-1.1-7/10/2025

For the latest technical documentation, see the Documentation Portal.

Thredd, Kingsbourne House, 229-231 High Holborn, London, WC1V 7DA

Support Email: occ@thredd.com

Support Phone: +44 (0) 203 740 9682



Copyright

About this Guide

This guide describes the Thredd Master Virtual Card (MVC) functionality and how to implement it on your card program.

Target Audience

This guide is aimed at product managers and business teams who want to use this guide to find out how MVCs work. Developers, who are using our SOAP Web Services or REST-based Cards API, can also refer to this guide for creating and managing MVCs.

What's Changed?

For finding out what's changed since the previous release, see the Document History section.

Other Documentation

Refer to the table below for a list of other relevant documents that you can use with this guide.

Document	Description
Web Services Guide (SOAP)	Provides details of using our SOAP-based Web Services to create and manage cards.
Cards API Website (REST)	Provides details of using our REST-based Cards API to create and manage cards.

Tip: For the latest technical documentation, see the Documentation Portal.

1 Overview

A Master Virtual Card (MVC) is a type of master account card record that you can use to reflect money loaded into a cardholder's account. An MVC is ideal for programs where it is linked to multiple physical and/or virtual cards, as you can transfer money to any of those cards. The MVC card record is restricted to loading and unloading only, and cannot be used like a regular card.

The following diagrams show how an MVC works with multiple virtual and physical cards.

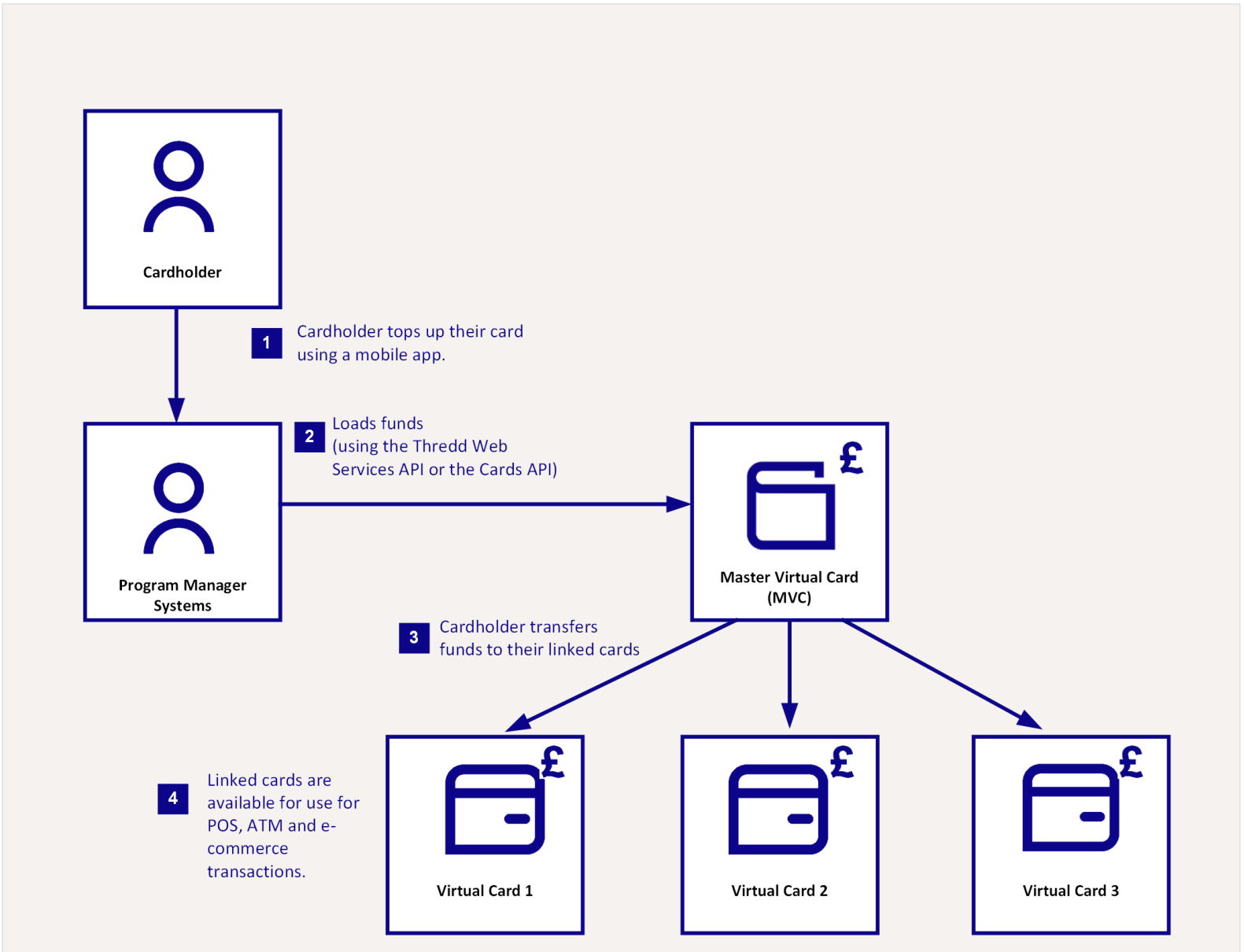


Figure 1: An MVC with Linked Virtual Cards

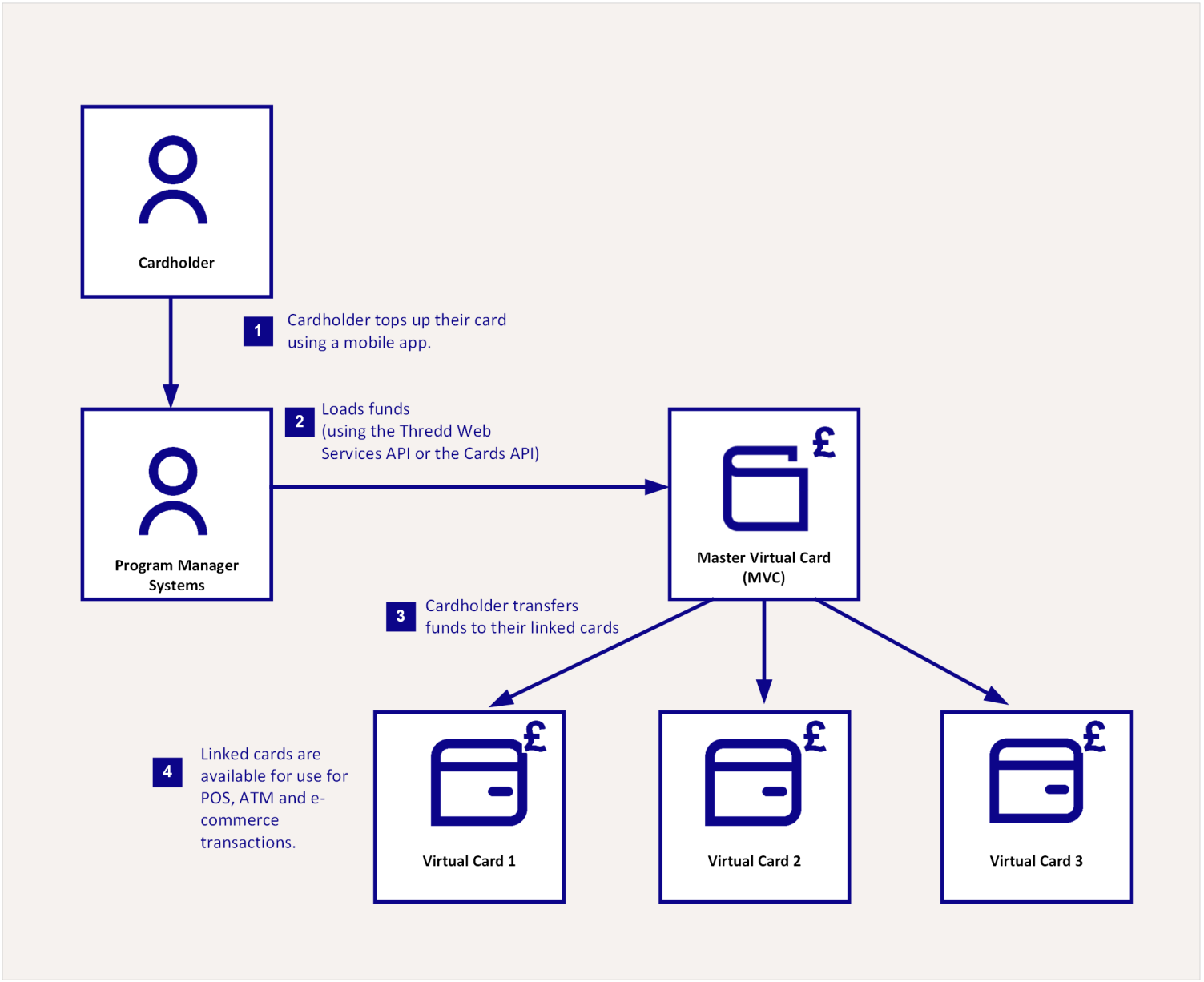


Figure 2: An MVC with Linked Physical Cards

Note that it is also possible to link a single MVC to both virtual and physical cards.

An MVC can link to physical and virtual cards for a particular currency. For example, an MVC can link to cards for US Dollars, while another MVC can link to cards in Euros. The above example shows MVCs set up for Pound Sterling.

An MVC guarantees that the load is limited to the prefunded amount, where the money is loaded onto the MVC. As soon as funds have been authorised or paid into your Issuer Bank account, you can distribute the funds to the linked cards. For additional use cases of MVCs, see Use Case Scenarios.

You can create and manage MVCs using either SOAP web services or the Cards API. Most functions are available across both types of APIs, including, creating, loading and transferring balances.

Note: Creating physical MVCs or performing any type of Point of Sale, e-commerce, ATM or other type of payment transactions are not possible with an MVC. In addition, your Issuer (BIN sponsor) must agree to allowing MVC cards for your program. This is because MVCs are cards with loads, and are included in the Transaction, Balance and other reports sent to you and your Issuer.

1.1 Funding of the MVC

An MVC is a type of prefunding account which allows the Program Manager to load an amount which is not yet assigned to any cardholder. Therefore, these are the funds of the Program Manager and are not counted as e-money.

The Program Manager controls the settlement account and is responsible for covering the actual money from the load into the Issuer's settlement account.

The Program Manager agrees with the Issuer on how they will be informed that funds have arrived in the Issuers Bank account (and by whom). They also agree on who is responsible for reflecting this value on the MVC.

Issuers maintain the following processes for managing MVCs:

- They provide the Program Manager with “view access” to their accounts. This ensures that, when the actual funds are available in the Issuers Bank Account, the Program Manager can then reflect that e-money value on the MVC.

- They advise the Program Manager when these funds are available so that the Program Manager can reflect the value on the MVC. This happens outside of Thredd and is between Program Manager and the Issuer to agree.

1.2 Setting Up Your MVC

When setting up your MVC, two options are available:

1. Create a separate card product for your MVC (recommended). This lets you create a default card configuration and card usage groups that apply only to your MVCs. For example, you can set up a usage group to ensure that no transactions or fees are applied to the MVC. When creating an MVC as a separate card product, it will not expire. Note that when creating a separate product, you select a flag on the PSF form for **card account non expiry**, which means the MVC does not expire.
2. Use the same card product for the MVC and other types of cards (for example, physical or virtual). You should create an MVC to expire after 8 years. Note that once the MVC expires, any cards that are linked to an MVC need to be relinked using the Thredd API: SOAP Link Cards (`Ws_link_cards`) Web Service or REST-based Update Card API (using the `parentCard` field). Using the same card product means that the default product configuration is shared. You need to first set up relevant Card Usage groups for your MVCs and then apply these groups to each MVC when created. Using the same product also requires you to adjust the expiry date of the MVC to match the cards.

You specify configuration options on your Product Setup Form (PSF). For more information see MVC Setup Options.

Note: You are not required to activate an MVC, as it can receive loads without being activated. Remaining inactive provides an extra security measure should the MVC ever be compromised.

Controls

There are 3 types of controls to protect the MVC in a transaction environment.

- **Groups:** Each card product is allocated a default set of attributes or groups. An MVC includes the Velocity Limits group and a Card Usage Group. This ensures that, if the MVC ever was compromised, usage would be restricted to loading and unloading only. With Velocity Limits, the card can only be loaded. This is because Cash and POS limits are set to ZERO. With Card Usage Groups, all usage groups are restricted.
- **Stop Card Going to Print:** When creating an MVC, the Program Manager must stop an MVC going to print by selecting 4 as the `<CreateType>` through SOAP web service or, if using the REST-based Cards API, selecting `cardType` of *MasterVirtual*.
- **PAN & CVV Should Never be Made Known to Cardholder/Corporate Client:** If anyone using an MVC needs a reference, then they should be provided with the MVC's 9 digit token. The MVC's PAN and CVV should never be retrieved or made known to the MVC holder

Note: If an MVC is being created from the same Product ID as other cards, then it is likely that the Program Manager needs to ensure that the correct Velocity Limit Groups and Card Usage groups exist.

1.3 Creating MVCs

You create MVCs using either Thredd's SOAP-based Web Services or the REST-based cards API. For more information, see Using Web Services and Using the Cards API.

Note: You should ensure that you use the correct API environment that you have been provided access to. For example, if you have been given access to the REST environment, use REST instead of SOAP.

1.4 Updating the Account Balance on the MVC

Your MVC can share the balance with other types of cards or have its own separate balance. For more information, see MVC Setup Options > Sharing of Balance.

Example MVC Account Balance Update Flow

The following shows an example workflow for updating an account balance.

1. The customer tops up their account via bank transfer or through accepting an online card payment. Your customer portal or mobile app should provide your cardholders with a means to top up the account. The 'actual' money is authorised and paid to your Issuer's bank account.
2. You use either the Web Services or the Cards API to load funds onto the MVC. Loading funds ensures that the money that has been paid into the account (minus any fees you charge for top-ups or currency conversion).

3. You display the latest MVC balance to the cardholder once loaded.
4. The cardholder transfers funds to any of the cards linked to their MVC wallet.
5. Your mobile app updates the balances on the cards to reflect these transactions.

For additional use cases for updating the account balance, see Use Case Scenarios.

1.5 Alternative Names for an MVC

An MVC can be referred to by these other names.

- iMVC (individual Master Virtual Card)
- Wallet
- Deposit Account
- Prefunding Account
- Holding Account/Pot
- Jam Jar
- Corporate Wallet

1.6 External Host Interface (EHI) Modes

You can use an MVC in Full Service Processing (mode 3). However, it is not applicable to Gateway Processing (modes 1 and 4). In situations where you want to override Thredd's approval or load a card on approval, you can uses Cooperative Processing (mode 2). For more details, see the EHI Guide for XML or the EHI Guide for JSON.

1.7 MFX Functionality

If you are using MFX functionality for multiple currencies, you cannot use an MVC. This is because the MFX functionality uses multiple wallets.

2 Use Case Scenarios

This section explains the use cases for an MVC. These include:

- Prefunding
- Payroll
- Issuer-controlled Loading
- Account Management
- Higher Loading Limits
- Ghost Account or Suspense Account

2.1 Prefunding

In this scenario, the Program Manager uses an MVC to reflect funds that have been preloaded to the Issuer. They can transfer funds from the MVC to cards that are linked to the MVC. However, they are prevented from "paying out" more than what is funded.

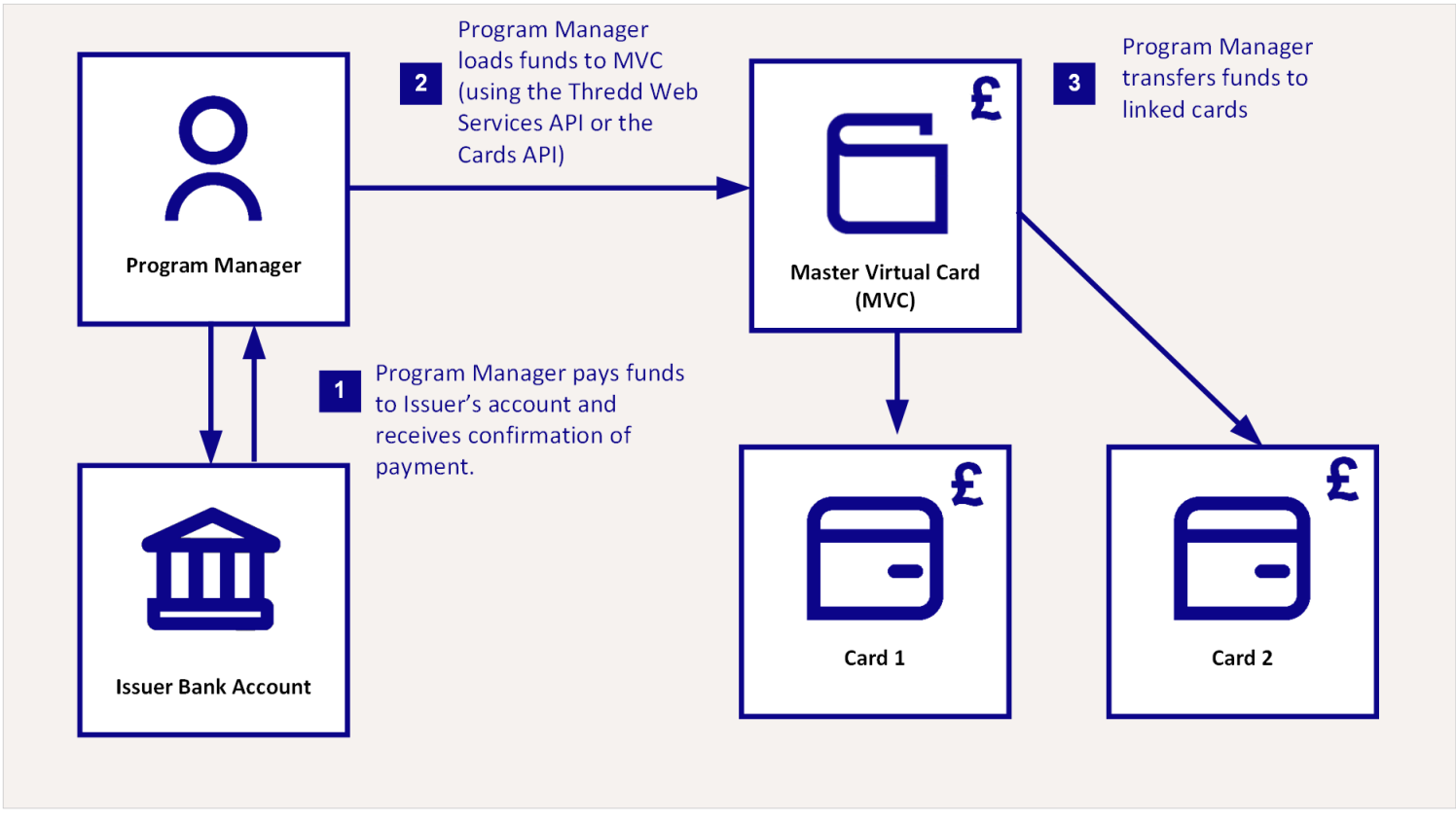


Figure 3: Prefunding Scenario

1. The Program Manager deposits a lump sum of funds with the Issuer. The Program Manager can check that the actual funds have been received. Alternatively, the Issuer (BIN sponsor) can advise the Program Manager that funds have cleared in the Issuer's Bank account.
2. The Program Manager reflects the value of the Issuer's deposit by loading an amount equivalent of the lump sum to the MVC.
3. The Program Manager transfers the funds from the MVC to the cardholder's cards, but is restricted to the balance available on the MVC.

2.2 Payroll

In this scenario, the Program Manager uses an MVC to reflect payroll payments into corporate accounts. The Program Manager may have multiple corporate clients who use the Program Manager's prepaid card programfor payroll. The Corporate Entity is prevented from loading more than what it has actually funded in the MVC.

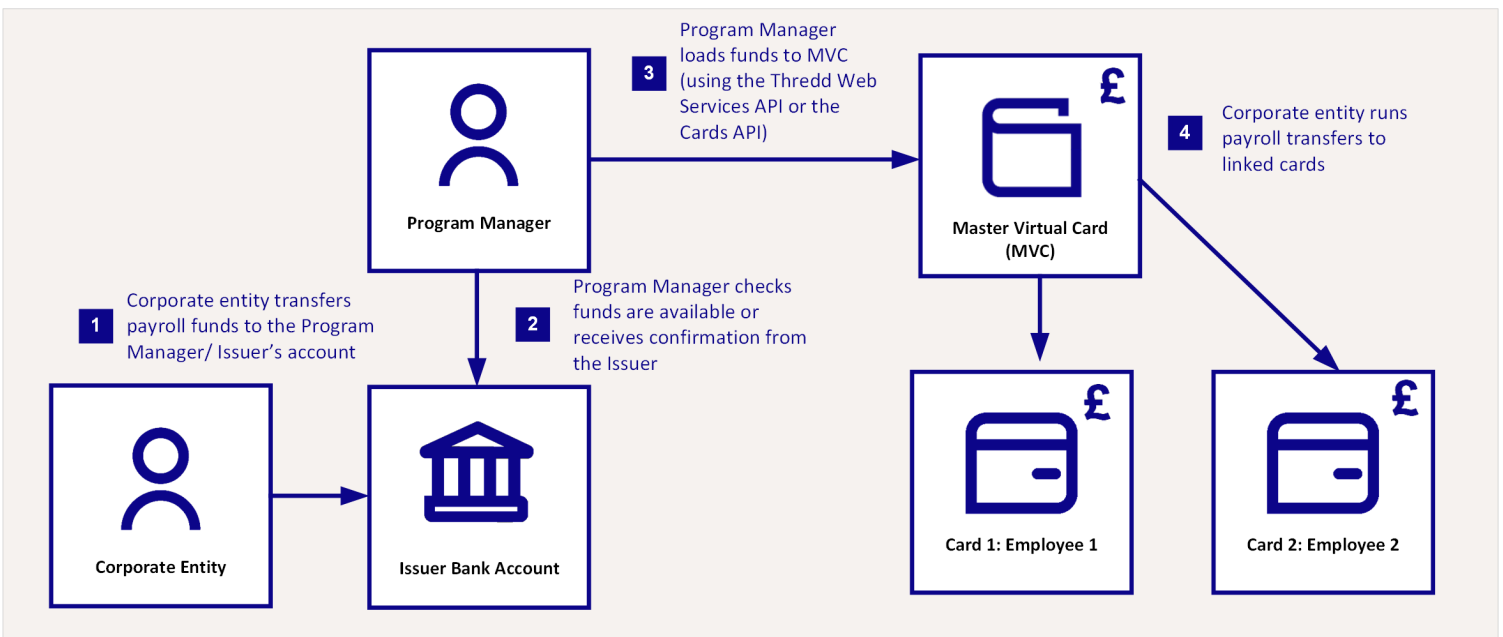


Figure 4: Payroll Scenario

1. The Corporate Entity transfers funds to the Program Manager/Issuer's bank account.
2. The Program Manager checks that funds have been received from the Corporate Entity or receives confirmation of payment from their Issuer.
3. The Program Manager loads the Corporate Entity's MVC.
4. The Corporate Entity can now 'run payroll' and transfer funds to the cardholder's cards.

2.3 Issuer-Controlled Loading

In this scenario, the Program Manager uses an MVC to reflect Issuer-controlled loading of cards. For example, the Issuer can be involved in the loading of the cards.

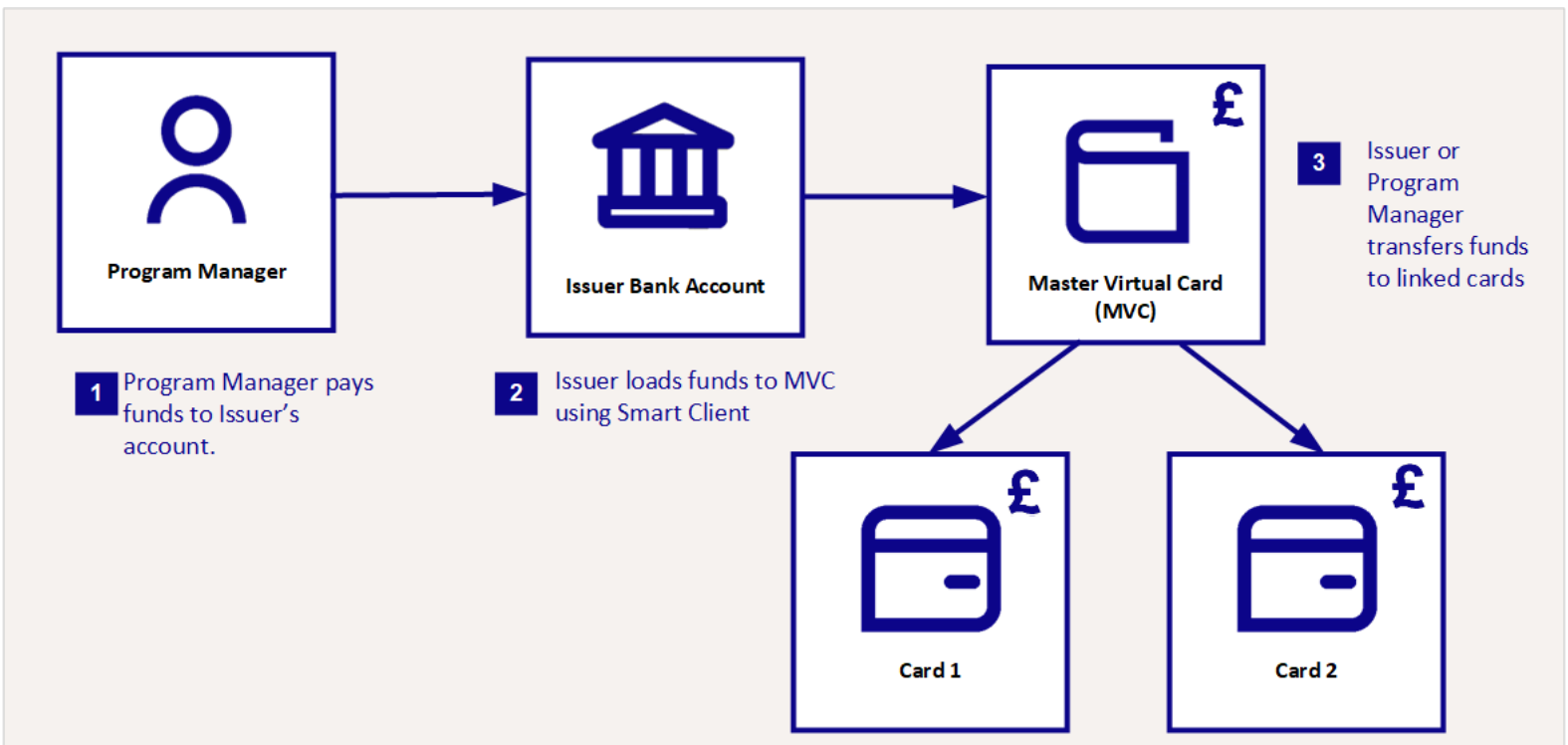


Figure 5: Issuer Scenario

1. The Program Manager transfers funds to the Issuer's bank account.
2. Once the actual funds are reflected in the Issuer's bank account, the Issuer can load the MVC with the value.
3. The Issuer transfers funds from the MVC to the cardholder's cards. Note that the Program Manager can do this task depending on what has been agreed.

Note: An Issuer does not use the APIs for loading.

2.4 Account Management

In this scenario, the Program Manager uses the MVC as a type of cardholder master account. As Thredd's main purpose is to process cards, a cardholder is identified by its card, and as such Thredd doesn't have accounts for cardholders. However, if there is a concept of a ‘bank account’ in the Program Manager's system where the balance is held and needs to be maintained on the Thredd system, then you can use an MVC (one per cardholder) to act as the account balance holder. Using an MVC as an account balance holder means that, if the cardholder’s actual card is lost, stolen or blocked, transactions linked to the MVC balance can still occur.

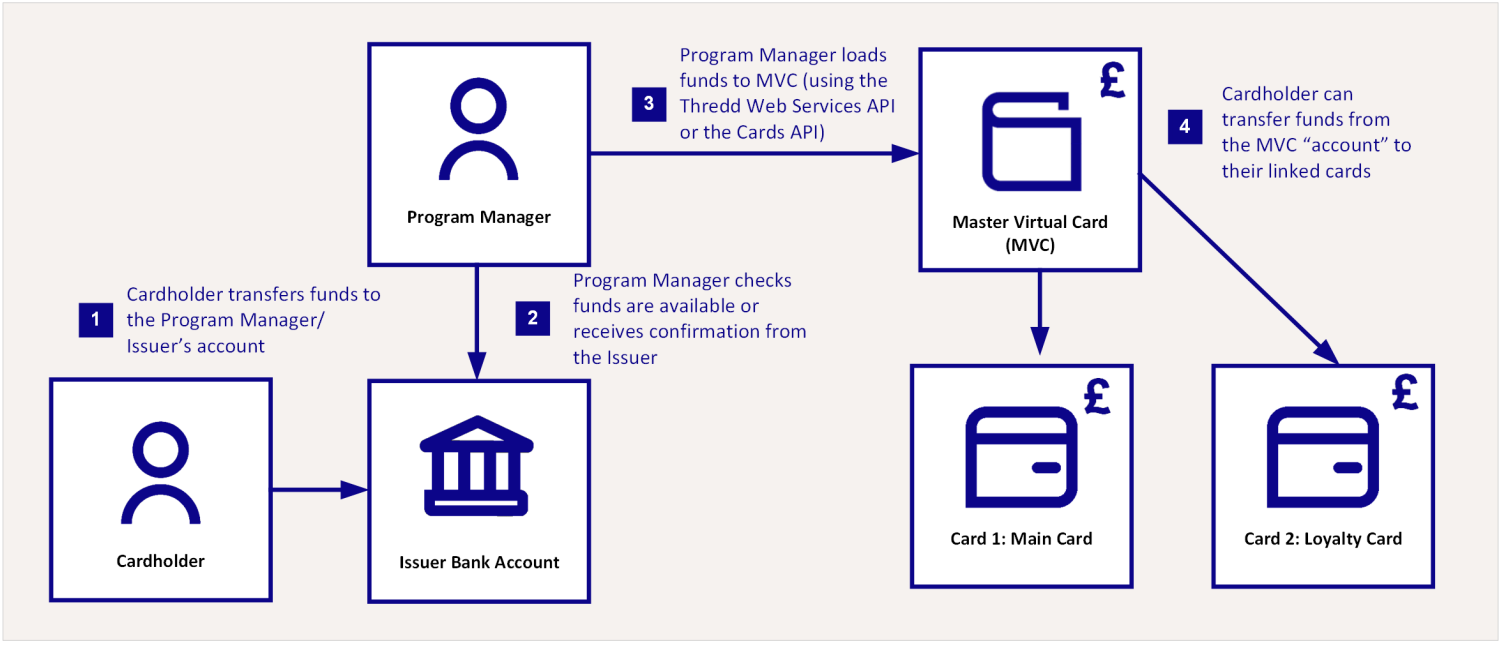


Figure 6: Account Scenario

1. The cardholder deposits money into the Issuer's account, for example, through a bank transfer using a reference or other agreed forms of loading.
2. The Program Manager checks that funds have been received from the cardholder, or receives confirmation of payment from their Issuer.
3. The Program Manager loads the cardholder's MVC with the respective amount upon confirmation that funds have been received in the Issuer's account.
4. The cardholder transfers the MVC balance to their linked cards.

2.5 Ghost or Suspense Accounts

In this scenario, the MVC is used as a Ghost or Suspense account, where it hold funds until the Program Manager knows where to allocate the funds. The Program Manager uses an MVC as a Ghost Account in these situations:

- A cardholder’s card has reached the available load limit.
- A received payment cannot be automatically assigned to a card, for example, the wrong payment reference has been supplied.
- An account is closed, blocked or suspended and funds need to be held for a period until they can be transferred to the cardholder's nominated bank account.

Figure 7: Ghost/Suspense Account Scenario

The Program Manager can use an MVC as both a Ghost Account Suspense Account in Pay-in and Pay-out scenarios.

Pay-in scenario:

1. The Program Manager receives confirmation of a payment which cannot be allocated to a specific card. For example, the card is reported as lost, an incorrect card reference is provided or the maximum allowed balance limits are exceeded.
2. The Program Manager loads the funds to the MVC (using the Cards API).
3. The Program Manager transfers funds to the card. For example, they transfer funds when identifying the correct card or reducing the balance. This ensures that the card's allowed limit is not be exceeded.

Pay-out scenario:

1. The Program Manager transfers the funds to the MVC. This is when a card is blocked or closed on notification of death of the cardholder, or if a card is reported as lost or stolen.
2. The Program Manager unloads the funds from the MVC (using the Thredd API or the Cards API) and returns the funds to the cardholder's nominated bank account.

3 MVC Setup Options

You can specify MPC setup options using the Product Setup Form (PSF). These include setting up product, sub-bin ranges and balances. For more information, discuss your requirements with your Thredd Implementation Manager.

3.1 Product Setup

You can set up an MVC as a separate card product. Alternatively, you can configure an MVC to share the same card product as other card types:

Product setup option	Card Usage Group and Limit Group Setup
Dedicated product for your MVC	Set up a default card **Limit Group**[1] and **Usage Group**[2] for your MVC product. When creating the card, you only need to specify the card product that you are using.
MVCs shares same product as other cards	Set up a default card **Limit Group**[3] and **Usage Group**[4] for MVC-type cards. When creating the MVC, assign the MVC limit and usage groups to the card.

3.2 Sub-Bin Setup

The MVC can share the same sub-bin range as your other types of cards. Alternatively, you could assign a dedicated sub-bin range to the MVC if required.

Note: Since you cannot print or use MVCs for purchases, Thredd recommend you do not use separate BIN ranges, as this reduces the number of available BINs that you can use.

3.3 Balance Setup

Your MVC can share the balance with other types of cards, or have its separate balance.

Balance setup	Usage
Shared Balance	The MVC holds the master balance for all cards. During transactions, a token links cards and an MVC allocates funds to other cards. The linked physical or virtual cards do not hold a balance. To use this option, you require a **Card Linkage Group**[5]. For details, check with your Implementation Manager.
Separate Balance	The MVC and other types of cards maintain their own separate balance. You transfer funds manually from an MVC to the required cards.

Note: If an MVC is on a separate Product ID to your physical or virtual cards (or under another Thredd Scheme/ BIN), you can share balances and transfer funds between the MVC and cards. You can transfer funds automatically or use the Thredd API or Cards API.

Shared Balances

For shared balances, the Card Linkage Group allocates funds from the MVC to be used as transactions, which works in the following ways:

---

[1]Velocity limit group which restricts the frequency and/or amount at which the card can be loaded or unloaded. You can view your current Limit Groups in Smart Client.
[2]Group that controls where a card can be used. For example: POS or ATM.
[3]Velocity limit group which restricts the frequency and/or amount at which the card can be loaded or unloaded. You can view your current Limit Groups in Smart Client.
[4]Group that controls where a card can be used. For example: POS or ATM.
[5]The Linkage Group set up in Smart Client controls various parameters related to linked cards; for details, check with your Implementation Manager.

- Limits can be applied to authorisations, loads, and payments.
- Certain capabilities can be set on the secondary card, such as loading and payments.
- Funds left on the cardholder's cards are swept back to the MVC overnight.
- If there has been a reversal, the funds are returned to the MVC.
- In the case of a refund, scheme rules dictate that refunds need to return to the card that performed the transaction.

When reporting shared balances, note the following:

- The cardholder's card does not have a balance and is reported in the Transaction xml (as the transactions took place on this card.)
- The MVC shows in the Balance xml as it holds the balance.

Note: If the Program Manager wants this sweep to happen immediately (rather than overnight), then this capability needs to be developed on their platform. Alternatively, they can use `WS_BalanceTransfer` to configure their sweeps. `WS_BalanceTransfer` is a dedicated SOAP web service for transferring balances (for more details, see Transferring a Balance)

4 Using Thredd Web Services (SOAP)

This section describes how to use the Thredd SOAP web services API to create and manage your MVCs.

Note: Thredd includes SOAP-based APIs specific to MVCs, for example, Ws_Activate_MVCLoad. However, using these APIs is not mandatory, as you can use APIs that are generic across all card product, such as Ws_CreateCards, Ws_Load and Ws_Unload.

4.1 Creating an MVC

To create an MVC, use the Ws_CreateCard SOAP web service with CreateType set to 4. You can use the web service to create an MVC as a separate product, or an MVC for the same card product. For more details, see Create Card.

You do not need to complete card fulfilment and other non-mandatory details when creating a card.

Example MVC Request

The following is a request for creating an MVC as a separate card product.

Note that you must include the following:

- The WSID , which must be unique for the request.
- The IssCode as assigned to you by Thredd.
- The LocDate and LocTime to state the precise date and time of the request.
- The CreateType, which must be set to 4 (indicating an MVC).
- The ExpDate, which must be blank.
- ActivateNow should be set to 0

```
<hyp:AuthSoapHeader>
         <hyp:strUserName>********</hyp:strUserName>
         <hyp:strPassword>************</hyp:strPassword>
      </hyp:AuthSoapHeader>
   </soapenv:Header>
   <soapenv:Body>
      <hyp:Ws_CreateCard>
         <hyp:WSID>120120241234</hyp:WSID>
         <hyp:IssCode>PMT</hyp:IssCode>
         <hyp:TxnCode>10</hyp:TxnCode>
         <hyp:ClientCode></hyp:ClientCode>
         <hyp:Title></hyp:Title>
         <hyp:LastName>Ken</hyp:LastName>
         <hyp:FirstName>Davies</hyp:FirstName>
         <hyp:Addrl1></hyp:Addrl1>
         <hyp:Addrl2></hyp:Addrl2>
         <hyp:Addrl3></hyp:Addrl3>
         <hyp:City></hyp:City>
         <hyp:PostCode></hyp:PostCode>
         <hyp:Country></hyp:Country>
         <hyp:Mobile></hyp:Mobile>
         <hyp:CardDesign>1628</hyp:CardDesign>
         <hyp:ExternalRef></hyp:ExternalRef>
         <hyp:DOB></hyp:DOB>
         <hyp:LocDate>2024-01-12</hyp:LocDate>
         <hyp:LocTime>120100</hyp:LocTime>
         <hyp:TerminalID></hyp:TerminalID>
         <hyp:LoadValue>20</hyp:LoadValue>
         <hyp:CurCode>GBP</hyp:CurCode>
         <hyp:Reason></hyp:Reason>
         <hyp:AccCode></hyp:AccCode>
         <hyp:ItemSrc>0</hyp:ItemSrc>
         <hyp:LoadFundsType>0</hyp:LoadFundsType>
         <hyp:LoadSrc></hyp:LoadSrc>
         <hyp:LoadFee>0</hyp:LoadFee>
         <hyp:LoadedBy></hyp:LoadedBy>
         <hyp:CreateImage>0</hyp:CreateImage>
         <hyp:CreateType>4</hyp:CreateType>
         <hyp:CustAccount>0</hyp:CustAccount>
         <hyp:ActivateNow>0</hyp:ActivateNow>
```

```
            <hyp:Source_desc></hyp:Source_desc>
            <hyp:ExpDate></hyp:ExpDate>
            <hyp:CardName></hyp:CardName>
            <hyp:LimitsGroup></hyp:LimitsGroup>
            <hyp:MCCGroup></hyp:MCCGroup>
            <hyp:PERMSGroup></hyp:PERMSGroup>
            <hyp:ProductRef></hyp:ProductRef>
            <hyp:CarrierType></hyp:CarrierType>
            <hyp:Fulfil1></hyp:Fulfil1>
            <hyp:Fulfil2></hyp:Fulfil2>
            <hyp:DelvMethod></hyp:DelvMethod>
            <hyp:ThermalLine1></hyp:ThermalLine1>
            <hyp:ThermalLine2></hyp:ThermalLine2>
            <hyp:EmbossLine4></hyp:EmbossLine4>
            <hyp:ImageId></hyp:ImageId>
            <hyp:LogoFrontId></hyp:LogoFrontId>
            <hyp:LogoBackId></hyp:LogoBackId>
            <hyp:Replacement>0</hyp:Replacement>
            <hyp:FeeGroup></hyp:FeeGroup>
            <hyp:PrimaryToken></hyp:PrimaryToken>
            <hyp:Delv_AddrL1></hyp:Delv_AddrL1>
            <hyp:Delv_AddrL2></hyp:Delv_AddrL2>
            <hyp:Delv_AddrL3></hyp:Delv_AddrL3>
            <hyp:Delv_City></hyp:Delv_City>
            <hyp:Delv_County></hyp:Delv_County>
            <hyp:Delv_PostCode></hyp:Delv_PostCode>
            <hyp:Delv_Country></hyp:Delv_Country>
            <hyp:Delv_Code></hyp:Delv_Code>
            <hyp:Lang></hyp:Lang>
            <hyp:Sms_Required></hyp:Sms_Required>
            <hyp:SchedFeeGroup></hyp:SchedFeeGroup>
            <hyp:WSFeeGroup></hyp:WSFeeGroup>
            <hyp:CardManufacturer></hyp:CardManufacturer>
            <hyp:CoBrand></hyp:CoBrand>
            <hyp:PublicToken></hyp:PublicToken>
            <hyp:ExternalAuth></hyp:ExternalAuth>
            <hyp:LinkageGroup></hyp:LinkageGroup>
            <hyp:VanityName></hyp:VanityName>
            <hyp:PBlock></hyp:PBlock>
            <hyp:PINMailer></hyp:PINMailer>
            <hyp:FxGroup></hyp:FxGroup>
            <hyp:Email></hyp:Email>
            <hyp:MailOrSMS></hyp:MailOrSMS>
            <hyp:AuthCalendarGroup></hyp:AuthCalendarGroup>
            <hyp:Quantity></hyp:Quantity>
            <hyp:LoadToken></hyp:LoadToken>
            <hyp:FeeWaiver></hyp:FeeWaiver>
            <hyp:BlackList></hyp:BlackList>
            <hyp:WhiteList></hyp:WhiteList>
            <hyp:PaymentTokenUsageGroup></hyp:PaymentTokenUsageGroup>
            <hyp:VirtualCardImage></hyp:VirtualCardImage>
            <hyp:ImageSize></hyp:ImageSize>
            <hyp:Url></hyp:Url>
            <hyp:IsNonReloadable>0</hyp:IsNonReloadable>
            <hyp:IsSingleUse>0</hyp:IsSingleUse>
         </hyp:Ws_CreateCard>
      </soapenv:Body>
   </soapenv:Envelope>
```

The following is a request for creating an MVC for the same card product. All the requirements for the fields in the request apply as per the above example. However, note the following:

- The ExpDate should be set to 8 years from creation.
- There must be a LinkageGroup, which is the same as the card product.

In this example, there is product with an ExpDate of 2031-12-31 (assuming a LocDate of 2023-12-31) and a LinkageGroup of GPSIXRSTST.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:hyp="http://www.globalprocessing.ae/HyperionWeb">
   <soapenv:Header>
      <hyp:AuthSoapHeader>
```

```
      <hyp:strUserName>*******</hyp:strUserName>
      <hyp:strPassword>************</hyp:strPassword>
   </hyp:AuthSoapHeader>
</soapenv:Header>
<soapenv:Body>
   <hyp:Ws_CreateCard>
      <hyp:WSID>15012027999</hyp:WSID>
      <hyp:IssCode>PMT</hyp:IssCode>
      <hyp:TxnCode>10</hyp:TxnCode>
      <hyp:ClientCode></hyp:ClientCode>
      <hyp:Title></hyp:Title>
      <hyp:LastName>Ken</hyp:LastName>
      <hyp:FirstName>Davies</hyp:FirstName>
      <hyp:Addrl1></hyp:Addrl1>
      <hyp:Addrl2></hyp:Addrl2>
      <hyp:Addrl3></hyp:Addrl3>
      <hyp:City></hyp:City>
      <hyp:PostCode></hyp:PostCode>
      <hyp:Country></hyp:Country>
      <hyp:Mobile></hyp:Mobile>
      <hyp:CardDesign>1628</hyp:CardDesign>
      <hyp:ExternalRef></hyp:ExternalRef>
      <hyp:DOB></hyp:DOB>
      <hyp:LocDate>2023-12-21</hyp:LocDate>
      <hyp:LocTime>120100</hyp:LocTime>
      <hyp:TerminalID></hyp:TerminalID>
      <hyp:LoadValue>20</hyp:LoadValue>
      <hyp:CurCode>GBP</hyp:CurCode>
      <hyp:Reason></hyp:Reason>
      <hyp:AccCode></hyp:AccCode>
      <hyp:ItemSrc>0</hyp:ItemSrc>
      <hyp:LoadFundsType>0</hyp:LoadFundsType>
      <hyp:LoadSrc></hyp:LoadSrc>
      <hyp:LoadFee>0</hyp:LoadFee>
      <hyp:LoadedBy></hyp:LoadedBy>
      <hyp:CreateImage>0</hyp:CreateImage>
      <hyp:CreateType>2</hyp:CreateType>
      <hyp:CustAccount>0</hyp:CustAccount>
      <hyp:ActivateNow>1</hyp:ActivateNow>
      <hyp:Source_desc></hyp:Source_desc>
      <hyp:ExpDate>2031-12-31</hyp:ExpDate>
      <hyp:CardName></hyp:CardName>
      <hyp:LimitsGroup></hyp:LimitsGroup>
      <hyp:MCCGroup></hyp:MCCGroup>
      <hyp:PERMSGroup></hyp:PERMSGroup>
      <hyp:ProductRef></hyp:ProductRef>
      <hyp:CarrierType></hyp:CarrierType>
      <hyp:Fulfil1></hyp:Fulfil1>
      <hyp:Fulfil2></hyp:Fulfil2>
      <hyp:DelvMethod></hyp:DelvMethod>
      <hyp:ThermalLine1></hyp:ThermalLine1>
      <hyp:ThermalLine2></hyp:ThermalLine2>
      <hyp:EmbossLine4></hyp:EmbossLine4>
      <hyp:ImageId></hyp:ImageId>
      <hyp:LogoFrontId></hyp:LogoFrontId>
      <hyp:LogoBackId></hyp:LogoBackId>
      <hyp:Replacement>0</hyp:Replacement>
      <hyp:FeeGroup></hyp:FeeGroup>
      <hyp:PrimaryToken></hyp:PrimaryToken>
      <hyp:Delv_AddrL1></hyp:Delv_AddrL1>
      <hyp:Delv_AddrL2></hyp:Delv_AddrL2>
      <hyp:Delv_AddrL3></hyp:Delv_AddrL3>
      <hyp:Delv_City></hyp:Delv_City>
      <hyp:Delv_County></hyp:Delv_County>
      <hyp:Delv_PostCode></hyp:Delv_PostCode>
      <hyp:Delv_Country></hyp:Delv_Country>
      <hyp:Delv_Code></hyp:Delv_Code>
      <hyp:Lang></hyp:Lang>
      <hyp:Sms_Required></hyp:Sms_Required>
      <hyp:SchedFeeGroup></hyp:SchedFeeGroup>
      <hyp:WSFeeGroup></hyp:WSFeeGroup>
```

```
            <hyp:CardManufacturer></hyp:CardManufacturer>
            <hyp:CoBrand></hyp:CoBrand>
            <hyp:PublicToken></hyp:PublicToken>
            <hyp:ExternalAuth></hyp:ExternalAuth>
            <hyp:LinkageGroup>GPSIXRSTST</hyp:LinkageGroup>
            <hyp:VanityName></hyp:VanityName>
            <hyp:PBlock></hyp:PBlock>
            <hyp:PINMailer></hyp:PINMailer>
            <hyp:FxGroup></hyp:FxGroup>
            <hyp:Email></hyp:Email>
            <hyp:MailOrSMS></hyp:MailOrSMS>
            <hyp:AuthCalendarGroup></hyp:AuthCalendarGroup>
            <hyp:Quantity></hyp:Quantity>
            <hyp:LoadToken></hyp:LoadToken>
            <hyp:FeeWaiver></hyp:FeeWaiver>
            <hyp:BlackList></hyp:BlackList>
            <hyp:WhiteList></hyp:WhiteList>
            <hyp:PaymentTokenUsageGroup></hyp:PaymentTokenUsageGroup>
            <hyp:VirtualCardImage></hyp:VirtualCardImage>
            <hyp:ImageSize></hyp:ImageSize>
            <hyp:Url></hyp:Url>
            <hyp:IsNonReloadable></hyp:IsNonReloadable>
            <hyp:IsSingleUse></hyp:IsSingleUse>
         </hyp:Ws_CreateCard>
      </soapenv:Body>
   </soapenv:Envelope>
```

Example Response

Thredd returns a PublicToken for the MVC in the response. You should use the PublicToken to link to an MVC. The following is an example response where the MVC is for the same card product.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <Ws_CreateCardResponse xmlns="http://www.globalprocessing.ae/HyperionWeb">
         <Ws_CreateCardResult>
            <WSID>15012028999</WSID>
            <IssCode>PMT</IssCode>
            <TxnCode>10</TxnCode>
            <PublicToken>112366247</PublicToken>
            <ExternalRef/>
            <LocDate>2024-01-12</LocDate>
            <LocTime>120100</LocTime>
            <ItemID>6160057433</ItemID>
            <ClientCode>0</ClientCode>
            <SysDate>2024-01-15</SysDate>
            <ActionCode>000</ActionCode>
            <LoadValue>20.00</LoadValue>
            <IsLive>true</IsLive>
            <StartDate>01/24</StartDate>
            <ExpDate>12/24</ExpDate>
            <CVV>390</CVV>
            <MaskedPAN>9999999465355698</MaskedPAN>
            <WebServiceResult/>
         </Ws_CreateCardResult>
      </Ws_CreateCardResponse>
   </soap:Body>
</soap:Envelope>
```

4.2 Loading an MVC

Loading allows funds to be available on an MVC. However, you can also use an MVC to load funds for a physical or virtual card. For further details on using these Thredd API web services, see the Web Services Guide.

Load an MVC

You use Ws_Load to load an MVC with funds. If you have loaded funds already, you can also use this web service to top-up an initial amount. For more details, see Load. Note the following:

- You must include the PublicToken of the MPC.
- You set the transaction code (TxnCode) to 1 or 20.

The following is an example request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:hyp="http://www.globalprocessing.ae/HyperionWeb">
   <soapenv:Header>
      <hyp:AuthSoapHeader>
         <hyp:strUserName>********</hyp:strUserName>
         <hyp:strPassword>**********</hyp:strPassword>
      </hyp:AuthSoapHeader>
   </soapenv:Header>
   <soapenv:Body>
      <hyp:Ws_Load>
         <hyp:WSID>202418011234578</hyp:WSID>
         <hyp:IssCode>PMT</hyp:IssCode>
         <hyp:TxnCode>20</hyp:TxnCode>
         <hyp:ClientCode></hyp:ClientCode>
         <hyp:AuthType>1</hyp:AuthType>
         <hyp:PAN></hyp:PAN>
         <hyp:Track2></hyp:Track2>
         <hyp:PublicToken>112367416</hyp:PublicToken>
         <hyp:DOB></hyp:DOB>
         <hyp:CVV></hyp:CVV>
         <hyp:AccCode></hyp:AccCode>
         <hyp:LastName></hyp:LastName>
         <hyp:LocDate>2024-01-18</hyp:LocDate>
         <hyp:LocTime>121100</hyp:LocTime>
         <hyp:TerminalID></hyp:TerminalID>
         <hyp:LoadValue>60</hyp:LoadValue>
         <hyp:CurrCode>GBP</hyp:CurrCode>
         <hyp:LoadFundsType>3</hyp:LoadFundsType>
         <hyp:LoadSrc>2</hyp:LoadSrc>
         <hyp:LoadFee>0</hyp:LoadFee>
         <hyp:SecID>0</hyp:SecID>
         <hyp:SecVal></hyp:SecVal>
         <hyp:SecValPos>0</hyp:SecValPos>
         <hyp:LoadedBy>Ken</hyp:LoadedBy>
         <hyp:Description>MVC Load</hyp:Description>
         <hyp:Sms_Required>0</hyp:Sms_Required>
         <hyp:BrnCode></hyp:BrnCode>
         <hyp:Note>MVC_Load</hyp:Note>
      </hyp:Ws_Load>
   </soapenv:Body>
</soapenv:Envelope>
```

The following is an example response:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <Ws_LoadResponse xmlns="http://www.globalprocessing.ae/HyperionWeb">
         <Ws_LoadResult>
            <WSID>202418011234578</WSID>
            <IssCode>PMT</IssCode>
            <TxnCode>20</TxnCode>
            <PublicToken>112367416</PublicToken>
            <LocDate>2024-01-18</LocDate>
            <LocTime>121100</LocTime>
            <ItemID>6160073972</ItemID>
            <ClientCode/>
            <SysDate>2024-01-18</SysDate>
            <ActionCode>000</ActionCode>
            <WebServiceResult/>
         </Ws_LoadResult>
      </Ws_LoadResponse>
   </soap:Body>
```

```
</soap:Envelope>
```

Activate a Separate Card

If you have created a card that is not yet activated, you can load funds into that card from an MVC. Once loaded, the card is activated. It is important to note that you are activating the physical and/or virtual card and not the MVC.

You use the Ws_Activate_MVCLoad endpoint (see MVC Card Activation and Load). This is an MVC-specific API.

Note the following:

- The PublicToken in Ws_Activate_MVCLoad is the card that you want to activate.
- The MVCToken is the public token of the MVC.
- The card that you are loading must be inactive with ActivateNow set to 0.
- The request must include a load source and currency code (LoadSrc and CurCode).

The following is an example request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:hyp="http://www.globalprocessing.ae/HyperionWeb">
   <soapenv:Header>
      <hyp:AuthSoapHeader>
         <hyp:strUserName>********</hyp:strUserName>
         <hyp:strPassword>************</hyp:strPassword>
      </hyp:AuthSoapHeader>
   </soapenv:Header>
   <soapenv:Body>
      <hyp:Ws_Activate_MVCLoad>
         <hyp:WSID>2024180101238</hyp:WSID>
         <hyp:IssCode>PMT</hyp:IssCode>
         <hyp:TxnCode>10</hyp:TxnCode>
         <hyp:Title></hyp:Title>
         <hyp:LastName></hyp:LastName>
         <hyp:FirstName></hyp:FirstName>
         <hyp:Addrl1></hyp:Addrl1>
         <hyp:Addrl2></hyp:Addrl2>
         <hyp:City></hyp:City>
         <hyp:PostCode></hyp:PostCode>
         <hyp:Country></hyp:Country>
         <hyp:ActMethod>6</hyp:ActMethod>
         <hyp:AuthType>1</hyp:AuthType>
         <hyp:PAN></hyp:PAN>
         <hyp:PublicToken>112367030</hyp:PublicToken>
         <hyp:DOB></hyp:DOB>
         <hyp:CVV></hyp:CVV>
         <hyp:AccCode></hyp:AccCode>
         <hyp:LocDate>18012024</hyp:LocDate>
         <hyp:LocTime>110200</hyp:LocTime>
         <hyp:MVCPAN></hyp:MVCPAN>
         <hyp:MVCToken>112366842</hyp:MVCToken>
         <hyp:AmtTxn>10</hyp:AmtTxn>
         <hyp:CurCode>GBP</hyp:CurCode>
         <hyp:LoadSrc>30</hyp:LoadSrc>
         <hyp:LoadedBy></hyp:LoadedBy>
         <hyp:Description></hyp:Description>
         <hyp:FeeWaiver></hyp:FeeWaiver>
         <hyp:BrnCode></hyp:BrnCode>
         <hyp:ActivateOrNot>1</hyp:ActivateOrNot>
         <hyp:ExpDate></hyp:ExpDate>
         <hyp:ItemSrc>0</hyp:ItemSrc>
         <hyp:SMSBalance>0</hyp:SMSBalance>
      </hyp:Ws_Activate_MVCLoad>
   </soapenv:Body>
</soapenv:Envelope>
```

The following is an example response:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <Ws_Activate_MVCLoadResponse xmlns="http://www.globalprocessing.ae/HyperionWeb">
         <Ws_Activate_MVCLoadResult>
            <WSID>2024180101238</WSID>
```

```
            <IssCode>PMT</IssCode>
            <TxnCode>10</TxnCode>
            <PublicToken>112367030</PublicToken>
            <MVCToken>112366842</MVCToken>
            <LocDate>18012024</LocDate>
            <LocTime>110200</LocTime>
            <AmtTxn>10.00</AmtTxn>
            <CurCode>GBP</CurCode>
            <IsLive>true</IsLive>
            <AvlBal>30.00</AvlBal>
            <BlkAmt>0.00</BlkAmt>
            <ItemID>6160073922</ItemID>
            <SysDate>2024-01-18</SysDate>
            <ActionCode>000</ActionCode>
         </Ws_Activate_MVCLoadResult>
      </Ws_Activate_MVCLoadResponse>
   </soap:Body>
</soap:Envelope>
```

Load Funds from an MVC to a Card

You load funds to an active physical or virtual card linked to an MVC. You use the Ws_MVCLoad SOAP API web service (for more details see MVC Load). This is an MVC-specific API.

Note the following:

- The MVCToken is the token of the MVC.
- The NewToken is the card that receives the funds.
- There must be an amount and currency code (AmtTxn and CurCode)

The following is an example request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:hyp="http://www.globalprocessing.ae/HyperionWeb">
   <soapenv:Header>
      <hyp:AuthSoapHeader>
         <hyp:strUserName>********</hyp:strUserName>
         <hyp:strPassword>************</hyp:strPassword>
      </hyp:AuthSoapHeader>
   </soapenv:Header>
   <soapenv:Body>
      <hyp:Ws_MVCLoad>
         <hyp:IssCode>PMT</hyp:IssCode>
         <hyp:MVCToken>114729737</hyp:MVCToken>
         <hyp:NewToken>114729581</hyp:NewToken>
         <hyp:AmtTxn>20</hyp:AmtTxn>
         <hyp:CurCode>GBP</hyp:CurCode>
         <hyp:LoadedBy>ken</hyp:LoadedBy>
         <hyp:Description>transfer</hyp:Description>
      </hyp:Ws_MVCLoad>
   </soapenv:Body>
</soapenv:Envelope>
```

The following is an example response:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <Ws_MVCLoadResponse xmlns="http://www.globalprocessing.ae/HyperionWeb">
         <Ws_MVCLoadResult>
            <IssCode>PMT</IssCode>
            <MVCToken>114729737</MVCToken>
            <NewToken>114729581</NewToken>
            <SysDate>2024-01-18</SysDate>
            <ActionCode>000</ActionCode>
            <AvlBal>55.00</AvlBal>
            <BlkAmt>0.00</BlkAmt>
            <AmtTxn>20.00</AmtTxn>
            <CurCode>GBP</CurCode>
            <ItemID>6160074014</ItemID>
         </Ws_MVCLoadResult>
      </Ws_MVCLoadResponse>
   </soap:Body>
```

```
</soap:Envelope>
```

Unload Funds from a Card to an MVC

If required, you can unload funds back to an MVC from a physical or virtual card using Ws_MVCUnload. This is an MVC-specific API.

Note the following:

- The MVCToken is the token of the MVC.
- The NewToken is the card that sends the funds back.
- There must be an amount and currency code (AmtTxn and CurCode).

The following is an example request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:hyp="http://www.globalprocessing.ae/HyperionWeb">
   <soapenv:Header>
      <hyp:AuthSoapHeader>
         <hyp:strUserName>********</hyp:strUserName>
         <hyp:strPassword>************</hyp:strPassword>
      </hyp:AuthSoapHeader>
   </soapenv:Header>
   <soapenv:Body>
      <hyp:Ws_MVCUnload>
         <hyp:IssCode>PMT</hyp:IssCode>
         <hyp:MVCToken>114729737</hyp:MVCToken>
         <hyp:NewToken>114729581</hyp:NewToken>
         <hyp:AmtTxn>5</hyp:AmtTxn>
         <hyp:CurCode>GBP</hyp:CurCode>
         <hyp:LoadedBy>ken</hyp:LoadedBy>
         <hyp:LoadSrc>0</hyp:LoadSrc>
         <hyp:Description>MVC unload</hyp:Description>
      </hyp:Ws_MVCUnload>
   </soapenv:Body>
</soapenv:Envelope>
```

The following is an example response:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:hyp="http://www.globalprocessing.ae/HyperionWeb">
   <soapenv:Header>
      <hyp:AuthSoapHeader>
         <hyp:strUserName>********</hyp:strUserName>
         <hyp:strPassword>************</hyp:strPassword>
      </hyp:AuthSoapHeader>
   </soapenv:Header>
   <soapenv:Body>
      <hyp:Ws_MVCUnload>
         <hyp:IssCode>PMT</hyp:IssCode>
         <hyp:MVCToken>114729737</hyp:MVCToken>
         <hyp:NewToken>114729581</hyp:NewToken>
         <hyp:AmtTxn>5</hyp:AmtTxn>
         <hyp:CurCode>GBP</hyp:CurCode>
         <hyp:LoadedBy>ken</hyp:LoadedBy>
         <hyp:LoadSrc>0</hyp:LoadSrc>
         <hyp:Description>MVC unload</hyp:Description>
      </hyp:Ws_MVCUnload>
   </soapenv:Body>
</soapenv:Envelope>
```

4.3 Linking Cards to an MVC (Primary and Secondary Cards)

You can link a secondary card to a primary card (the MVC). Using Ws_CreateCard, you can create the cardholder's cards (virtual or physical) which you link to the MVC. You must enter the 9-digit PrimaryToken value of your MVC. You need to provide the PrimaryToken , for example, 987654321, in the correct element:

```
<hyp:PrimaryToken>987654321</hyp:PrimaryToken>
```

See Create Card (Ws_CreateCard).

An MVC can have many secondary cards linked to it, however, it is not possible to cascade cards. Cascading cards is where there are two primary cards that are linked together. However, one of the primary cards is linked to secondary card(s).

4.4 Transferring a Balance

For transferring an amount from the MVC to a cardholder's card, the Program Manager uses Ws_BalanceTransfer. An unload shows on the MVC and a load displays on the cardholder's card. The load to the cardholder's cards is limited to the prefunded amount loaded on the MVC. The load onto the cardholder's card is immediate and the amount is instantly available for use.

Note the following:

- The PublicToken is the MVC.
- The NewToken is the card that receives the funds.
- The TxnCode must be set to 7.
- There must be an amount and currency code (AmtTxn and CurCode). The below example shows 5 GBP as the balance to transfer.
- LoadSource must be set to 74 or 68.

The following shows an example request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:hyp="http://www.globalprocessing.ae/HyperionWeb">
   <soapenv:Header>
      <hyp:AuthSoapHeader>
         <hyp:strUserName>********</hyp:strUserName>
         <hyp:strPassword>************</hyp:strPassword>
      </hyp:AuthSoapHeader>
   </soapenv:Header>
   <soapenv:Body>
      <hyp:Ws_BalanceTransfer>
         <hyp:WSID>202418011236</hyp:WSID>
         <hyp:IssCode>PMT</hyp:IssCode>
         <hyp:TxnCode>7</hyp:TxnCode>
         <hyp:ClientCode></hyp:ClientCode>
         <hyp:AuthType>1</hyp:AuthType>
         <hyp:PAN></hyp:PAN>
         <hyp:Track2></hyp:Track2>
         <hyp:PublicToken>112363814</hyp:PublicToken>
         <hyp:DOB></hyp:DOB>
         <hyp:CVV></hyp:CVV>
         <hyp:AccCode></hyp:AccCode>
         <hyp:LastName>Davies</hyp:LastName>
         <hyp:LocDate>2024-01-18</hyp:LocDate>
         <hyp:LocTime>133400</hyp:LocTime>
         <hyp:TerminalID></hyp:TerminalID>
         <hyp:NewPAN></hyp:NewPAN>
         <hyp:NewToken>112364353</hyp:NewToken>
         <hyp:AmtTxn>5</hyp:AmtTxn>
         <hyp:CurrCode>GBP</hyp:CurrCode>
         <hyp:LoadSrc>68</hyp:LoadSrc>
         <hyp:SecID>0</hyp:SecID>
         <hyp:SecVal></hyp:SecVal>
         <hyp:SecValPos>0</hyp:SecValPos>
         <hyp:Description></hyp:Description>
         <hyp:LoadedBy>Davies</hyp:LoadedBy>
         <hyp:FeeWaiver>2</hyp:FeeWaiver>
         <hyp:BrnCode></hyp:BrnCode>
         <hyp:Fee>0</hyp:Fee>
      </hyp:Ws_BalanceTransfer>
   </soapenv:Body>
</soapenv:Envelope>
```

The following shows an example response.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <Ws_BalanceTransferResponse xmlns="http://www.globalprocessing.ae/HyperionWeb">
```

```
         <Ws_BalanceTransferResult>
            <WSID>202418011236</WSID>
            <IssCode>PMT</IssCode>
            <TxnCode>7</TxnCode>
            <PublicToken>112363814</PublicToken>
            <LocDate>2024-01-18</LocDate>
            <LocTime>133400</LocTime>
            <NewPAN>9999999786646072</NewPAN>
            <ClientCode>0</ClientCode>
            <SysDate>2024-01-18</SysDate>
            <ActionCode>000</ActionCode>
            <AvlBal>15.00</AvlBal>
            <BlkAmt>0.00</BlkAmt>
            <AmtTxn>5.00</AmtTxn>
            <CurCode>GBP</CurCode>
            <ItemID>6160074146</ItemID>
         </Ws_BalanceTransferResult>
      </Ws_BalanceTransferResponse>
   </soap:Body>
</soap:Envelope>
```

Note: Transferring a balance does not require a Card Linkage Group. This is because the cardholder is in control of the balance transfer to the card from the MVC. Setting up primary and secondary cards is also optional for a balance transfer.

4.5 Replacing a New Token

If the MVC needs to be replaced with a new token, you can add the PublicToken and the NewToken details to each of the virtual and physical cards using Ws_link_cards. This ensures that the cards are relinked to the MVC. In this web service example, you enter:

```
<hyp:NewToken>112364353</hyp:NewToken>
```

for the card that the MVC links to.

For the MVC with the new token (to replace the old one), you enter the following:

```
<hyp:PublicToken>112363814</hyp:PublicToken>
```

4.6 Viewing a List of Cards

For viewing a list of cards associated with an MVC, you use the WS_Customer_Enquiry_V2 endpoint. A customer service representative typically checks for virtual or physical cards for an MVC. This endpoint requires you to enter the PublicToken for the MPC.

4.7 Search MVCs for Expiry

Using the Ws_Get_Card_ExpireSoon SOAP API web service, you can search for any MVCs that are to expire in the next month. This is provided that you have set an expiry date to one or more of your MVCs. For more details, see Cards Get Expiring Soon.

4.8 Load Sources

An Issuer (BIN sponsor) can see information on the load source within the various web services. Load sources are optional depending on what an Issuer wants to see reported as a load source or for applying a web service fee.

Code	Name	Description
68	Primary Card	This code is used when MVC and cardholder's cards are linked in a primary and secondary card relationship. The code allows you to transfer funds from the primary to the secondary card only. This is a mandatory setting for linked cards.
74	Master Virtual Card	This code lets you load an individual card from an MVC via balance transfer.
76	MVC Load	This code allows you to load an MVC using Ws_Load.
77	iMVC	This code lets you load an iMVC using Ws_Load.

4.9 Charging Fees

If a Program Manager wants to charge fees for loading, the load source needs to be set up with the corresponding processing code for web service fees.

Code		Description	
68	Primary Card	086: Fees	Primary card
74	Master Virtual Card	074: Fees	Master Virtual Card
76	MVC Load	076: Fees	MVC Load
77	iMVC Load	077: Fees	iMVC Load

5 Using the Cards API

This section describes how to use Thredd's REST-based Cards API to create and manage your MVCs.

The REST API uses the following base URL: *https://cardsapi-uat-pub.globalprocessing.net/api/v1/cards*.

Note: The Cards API does not include endpoints dedicated endpoints specifically for MVCs. However, you can use particular endpoints for cards in MVCs.

5.1 Creating an MVC

To create an MVC, use the Create Card API and set `"MasterVirtual"` as the `"CardType"`.

You do not need to complete card fulfilment and other non-mandatory details in the request.

See the example body below.

Example Request

```
{
     "cardHolder": {
          "title": "Mr",
          "firstName": "Jon",
          "lastName": "Smith",
          "dateOfBirth": "1982-11-03"
     },
     "address": {
          "addressLine1": "32 Western Drive",
          "postCode": "S25 2BZ",
          "country": "GBR"
     },

     "cardType": "MasterVirtual",
     "cardProduct": 10005,
     "designId": "New Card Brand",
     "customerReference": "my ref 12345"
}
```

Example Response

Thredd returns a `publicToken` for the MVC in the response. You should use the `publicToken` to link cards to a Master Virtual Card.

```
{
    "publicToken": "103170223",
    "customerReference": "my ref 12345",
    "embossName": "Mr Jon Smith",
    "maskedPan": "999999******5347",
    "startDate": "2023-11-09",
    "expiryDate": "2028-11-30"
}
```

5.2 Loading and Unloading an MVC

You can use the *Load or Unload Card* API endpoint for transactions to transfer card balances between your MVC and linked cards. In your request, your endpoint should include the `publicToken` of the MVC card in the path. Additionally, the body of the request should include the `publicToken` of the card to where you are transferring the balance. See the example body request below. For further details on using the transactions API, see the Cards API Website > Balance Transfers.

Example Request

```
{
  "transactionType": "TransferFunds",
  "amount": 10,
  "currencyCode": "GBP",
  "toPublicToken": "112233445"
}
```

Example Response


```
{
"Balance": {
  "CurrencyCode": "GBP",
  "ActualBalance": 10.00,
  "BlockedAmount": 0.00,
  "AvailableBalance": 10.00
  }
"Transaction":[
  {
    "PublicToken": 987654321,
    "TransactionID": "string",
    "TransactionType":"Unload",
    "Amount": 10.00,
    "CurrencyCode":"GBP"
    },
    {
    "PublicToken": 112233445,
    "TransactionID": "string",
    "TransactionType":"Load",
    "Amount": 10.00,
    "CurrencyCode":"GBP"
    }
]
}
```


5.3 Linking Cards to an MVC

When creating secondary cards which you want to link to a primary or MVC, you should enter the `publicToken` value of your MVC into the `parentCard` field. See Create Card Field Descriptions for more details.

Note: You need to ensure that the product for the child card has an associated Card Linkage Group. You can check for the associated Linkage Group on the product by using the *Get list of products for a given Program Manager* endpoint. See Get Product for Program Manager for more details.

In the example request, if the `publicToken` of your MVC is 987654321, then the `parentCard` field of the linked card you are creating should have the value of 987654321 in the example request.

```
{
    "CardType":"Physical",
    "CardProduct": 10023,
    "CustomerReference": "CustNo12345A",
    "parentCard": "987654321",
    "CardHolder": {
        "Title": "Mr.",
        "FirstName": "Bruce",
        "MiddleName": "Alan",
        "LastName": "Bloggs",
        "DateOfBirth": "1982-02-19",
        "Mobile": "07755123456",
        "Email": "babloggs@email.com"
    },
    "Address": {
        "AddressLine1": "1007",
        "AddressLine2": "Mountain Drive",
        "AddressLine3": "string",
        "City": "London",
        "PostCode": "GC11 2LD",
        "Country": "GBR"
    },
    "fulfilment": {
        "AddressLine1": "1007",
        "AddressLine2": "Mountain Drive",
        "AddressLine3": "string",
        "City": "London",
        "PostCode": "GC11 2LD",
        "Country": "GBR"
```

```
    },
    "ManufacturingDetails":
    {
      "CardManufacturer": "AB Note",
       "DeliveryMethod": 0,
       "CarrierType": "Type 1",
       "Quantity": 1,
       "Language": "HU",
       "ThermalLine1": "asdeoivh2neriuvnqkr",
       "ThermalLine2" : "asjkfnkerjnkwe",
       "EmbossLine4" : "jdwncwlkejr",
       "VanityName" : "Name Mane",
            "ImageDetails":
            {
                 "ImageId": "An image id",
                 "LogoFrontId" : "A logo ID",
                 "LogoBackId": "Another logo ID"
            }
   }
}
```

5.4 Transferring Funds to a Child Card

You can transfer funds from an MVC to a child card using the *Transactions* endpoint. To transfer funds from the MVC to the child card, the request body must have `transferFunds` set in the `transactionType` field and the child public token set in the `toPublicToken` field. See the below example where £200 is being transferred from the MVC with the publicToken of 103170108 to the child card publicToken 100512652.

```
{
  "transactionType": "TransferFunds",
  "amount": 200,
  "currencyCode": "GBP",
  "toPublicToken": "100512652"
}
```

General FAQs

Q. What options are available for setting up the MVC?

For details of setup options, see MVC Setup Options. For examples of use cases for Master Virtual Cards, see Use Case Scenarios.

Q. Is an MVC a virtual card?

No, an MVC is a type of card record that is used for holding account balances only. You can then transfer balances to and from other types of cards. You cannot use an MVC as a virtual card or to make payments.

Q. Will the card balance automatically sweep back to the MVC?

No, a **Card Linkage Group**[1] is required for sweeping back the balance. For more details, check with your Implementation Manager.

Q. If an MVC is a separate product and on a separate BIN to the secondary cards is it possible to link the cards?

Yes it is possible to link the cards where you can share the balances, and transfer funds between the primary and secondary cards.

Q. Can I get a Physical MVC?

No, a physical MVC does not exist. This is because there is a flag which stops the MVC going to print.

Q. Does an MVC have a full PAN?

Yes, as with other types of cards, an MVC is assigned a full PAN. However, it cannot be used for payment transactions. The PAN must **never** be revealed to the MVC holder (cardholder/corporate client) . If a reference to an MVC is required, they should be provided with the MVC's 9 digit token.

Q. Does the MVC have a start date, an expiry date and a CVV?

Yes, as with other types of cards, the MVC is assigned a start date, an end date and a CVV. However, it cannot be used for payment transactions. The CVV must **never** be revealed to the MVC holder (cardholder/corporate client) . If a reference to an MVC is required, they should be provided with the MVC's 9 digit token.

Q. Should the full card details be shared with the cardholder?

No, only the 9 digit token of the MVC can be shared with the cardholder.

Q. Is it possible to change an MVC into another card type?

No, once the card is created as an MVC, its type cannot be changed.

Q. Is it possible to change an existing card to an MVC?

No, once the card is created as a physical or virtual card, its type cannot be changed to an MVC.

Q. Can I view reports on MVC transactions?

Yes, you can receive details of MVC loads, unloads and balance transfers. These details are provided in your transaction XML reports and External Host Interface (EHI) data. For more information, see the Transaction XML Reporting Guide and External Host Interface Guide.

---

[1]The Linkage Group set up in Smart Client controls various parameters related to linked cards; for details, check with your Implementation Manager.

Glossary

This page provides a list of glossary terms used in this guide.

A

Acquirer

The merchant acquirer or bank that offers the merchant a trading account, to enable the merchant to take payments in store or online from cardholders.

C

Card Linkage Group

The Linkage Group set up in Smart Client controls various parameters related to linked cards; for details, check with your Implementation Manager.

Card Scheme (payment network)

Card scheme or payment network, such as MasterCard or Visa, responsible for managing transactions over the network and for arbitration of any disputes.

E

External Host

The external system to which sends real-time transaction-related data. The URL to this system is configured within per programme or product. The Program Manager uses their external host system to hold details of the balance on the cards in their programme and perform transaction-related services, such as payment authorisation, transaction matching and reconciliation.

G

Ghost account

A temporary account that provides a location for the Program Manager to allocate funds before transferring the funds to a permanent location. A ghost account is otherwise known as a suspense account.

I

Issuer

The card Issuer (BIN sponsor), typically a financial organisation authorised to issue cards. The Issuer has a direct relationship with the relevant card scheme.

L

Limit Group

Velocity limit group which restricts the frequency and/or amount at which the card can be loaded or unloaded. You can view your current Limit Groups in Smart Client.

M

Master Virtual Card (MVC)

A type of Thredd card that is restricted to loading and unloading to a physical or virtual card and cannot be used for e-commerce or in-store transactions. An MVC is used to reflect the value of the ‘actual’ money in the Issuer's bank account. An MVC guarantees that the load is limited to the amount prefunded (i.e. loaded onto MVC) and gives the Program Manager the ability to distribute funds immediately rather than having to wait for notification of each individual load into the Issuer Bank account.

Merchant

The provider of the product or service that the cardholder is purchasing, namely a shop or store. A merchant must have a merchant account, provided by their acquirer, in order to trade. Physical stores use a terminal or card reader to request authorisation for transactions. Online sites provide an online shopping basket and use a payment service provider to process their payments.

Merchant Category Code (MCC)

A unique identifier of the merchant for pinpointing the type of account provided to them by their acquirer.

P

Program Manager

A customer who manages a card program. The programme manager can create branded cards, load funds and provide other card or banking services to their end customers.

S

Smart Client

Smart Client is Thredd's legacy desktop application for managing your cards and transactions on the Thredd Platform.

Suspense Account

A temporary account that provides a location for the Program Manager to allocate funds before transferring the funds to a permanent location. A suspense account is otherwise known as a Ghost Account.

T

Thredd Portal

Thredd Portal is Thredd's new web application for managing your cards and transactions on the Thredd Platform.

U

Usage Group

Group that controls where a card can be used. For example: POS or ATM.

Document History

This section provides details of what has changed since the previous document release.

Version	Date	Reason	Revised by
1.1	11/02/2025	Added a glossary term for Thredd Portal, our new web application for managing your cards and transactions.	WS
	27/06/2024	Updated the company address.	PC
1.0	30/04/2024	Clarification around MVC setup using SOAP web services and REST-based Cards API.	WS
	24/04/2024	Updates to content and graphics to align with taxonomy updates on our Documentation Portal.	KD
	04/04/2024	Moved the Shared Balance sub-section to the Setup Options section.	KD
	31/01/2024	First version	KD

Contact Us

Please contact us if you have queries relating to this document. Our contact details are provided below.

Thredd UK Ltd.

Company registration number 09926803

Support Email: occ@thredd.com

Telephone: +44 (0) 203 740 9682

Our Head Office

Kingsbourne House

229-231 High Holborn

London

WC1V 7DA

Technical Publications

If you want to contact our technical publications team directly, for queries or feedback related to this guide, you can email us at: docs@thredd.com.