MDES Token Connect

Token Requestor Implementation Guide and Specifications

21 December 2021

Contents

Summary of changes

This document reflects updates effective since the previously-published version.

Description of Change	Where to Look
Added a new section for Signature Verification and detailed guidance to download and use public keys.	Signature verification
Added duplicate request as result	Display provisioning results
Updated the timing for validity of pushAccountReceipt to 15 mins	Token Connect workflows
Included details for Authentication service support for merchants	Post-Tokenization Authentication for Remote Commerce Programs
Added technical requirement for orphan token	Legal guidelines
Updated the document to include signature support for the Token Connect framework	Throughout the document

Chapter 1 About this document

This document is intended to guide token requestors (merchants, wallets, and commerce platforms) who need to support card or financial account push provisioning using the Mastercard Digital Enablement Services (MDES) Token Connect framework.

The document:

- Explains how the MDES Token Connect framework works
- Provides user experience and implementation guidelines to token requestors
- Contains specification of the exchanges between MDES issuers and MDES token requestors
- Provides an overview of MDES Token Connect implementation process for token requestors
- Describes legal obligations and responsibilities of token requestors with respect to account holder data received from issuers in connection with Token Connect

Audience

The following audiences should review this guide.

- Token requestors and token requestor service providers' business and legal staff
- Token requestors and token requestor service providers' operations staff who need to understand the impact of MDES Token Connect on their operational activities
- Token requestor staff responsible for implementing system and process changes to support the MDES Token Connect framework

Abbreviations and acronyms

This section describes the abbreviations and acronyms used in this guide.

Abbreviation	Description
ACN	Activation Code Notification
API	Application Programming Interface
CIS	Customer Implementation Services
CVC	Card Validation Code
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
JWS	JSON Web Signature
MDES	Mastercard Digital Enablement Services
MTF	Mastercard Test Facility
OS	Operating System

PAN	Primary Account Number
PDF	Portable Document Format (Adobe Acrobat)
PNG	Portable Network Graphics
SVG	Scalable Vector Graphics
TAR	Tokenization Authorization Request
TCN	Tokenization Complete Notification
TER	Tokenization Eligibility Request
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
TRID	Token Requestor Identifier
WID	Wallet Identifier
XML	Extensible Markup Language

Related documentation

This document should be read in conjunction with the latest release of the following documents:

- MDES Wallet Provider Implementation Guide
- MDES API Specification

Chapter 2 Token Connect overview

MDES Token Connect is a framework allowing the connection of MDES issuers with open MDES token requestors (such as wallets storing tokens on the mobile device or an IoT device, commerce platforms or merchants) for issuer-initiated digitization (also known as push provisioning).

Issuer-initiated digitization provides an optimized digitization experience driven by the issuer, initiated from the issuer's app or website. The issuer pushes payment credentials to the token requestor's environment to create a token.

Token Connect features

MDES Token Connect combines the benefits of issuer-initiated digitization with the scalability of an interoperable framework. It removes the need for:

- Multiple proprietary APIs between issuers and token requestors
- Costly one-to-one integration between each issuer and token requestor.

After their connection to MDES Token Connect framework is enabled, an MDES token requestor can accept connection requests from any MDES issuer that has implemented MDES Token Connect.

Compared to a traditional card account provisioning initiated from the token requestor's environment, issuer-initiated digitization or push provisioning provides multiple advantages:

- Offers a convenient user experience where the consumer selects one or more cards proposed by the issuer. It eliminates the burden of manually entering card details (account number, expiry date, CVC2).
- Increases tokenization success rates because issuers initiate push provisioning only for eligible token requestors.
- Provides token requestors with the opportunity to get more visibility for their brand through the issuer's interface and acquire new accounts from the issuer for future transactions.
- Allows issuers to enable Instant Digital Issuance; cardholders can pay with a token as soon as the issuer approves their subscription to a new card and even before the plastic card is shipped or active.
- Provides issuers with the opportunity to offer more services in their banking app/website and increase consumer participation in digital applications.

Consumer experience

Once logged into their issuer's application or website, a consumer is presented with a list of token requestors that are enabled to receive provisioning requests using the Token Connect framework. The cardholder chooses the target token requestor, and then selects one or more card accounts to be pushed to this target.

The consumer is then taken to the token requestor's app or website where they are asked to sign in or sign up. This allows the token requestor to associate the card being pushed with one of their customer accounts. Upon successful consumer log in, the token requestor proceeds with the tokenization of the account.

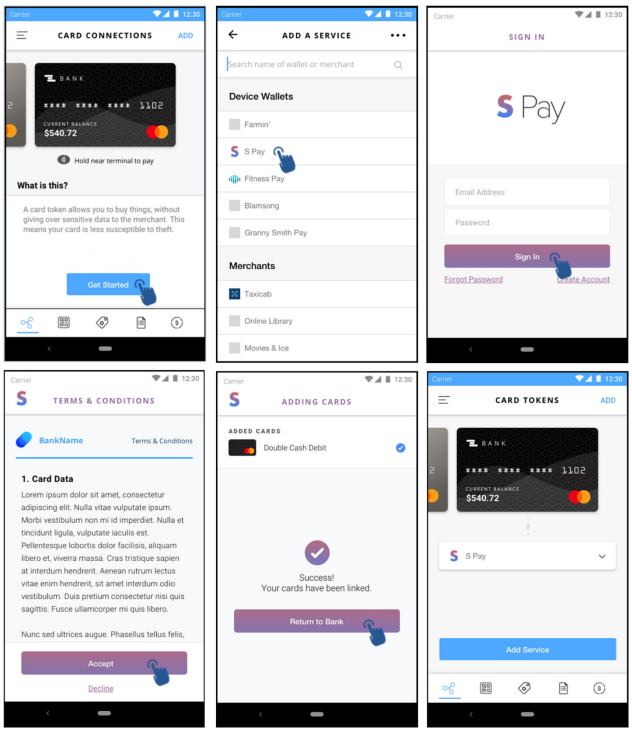
In the Consumer Experience detailed above, when the consumer is redirected to the token requestor app/portal to push the account on his/her profile from the issuer app/portal, the issuer does not pass any PCI & PII information. Only the reference and meta information that helps token requestor to define user experience for the push request and request digitized account from MDES is passed. Communication between issuer and token requestor system is a secure communication using HTTPS, so chances of altering parameter over the air is very difficult. To make sure that token requestor is receiving un-altered data from a legitimate issuer, a signature is included in the redirection. Details of the signature is provided in the Signature Verification topic.

NOTE: Depending on the issuer's configuration, cardholder authentication may be required to activate a new token in device wallets. Depending on a merchant's choice, cardholder authentication may be required before the token is stored.

When the tokenization is completed, the consumer is taken back to the issuer's app or website to continue browsing. For instance, the consumer may be invited to repeat the operation with another token requestor.

The figure below illustrates a possible user experience for tokenization using Token Connect:

Figure 1: User Experience: Using Token Connect

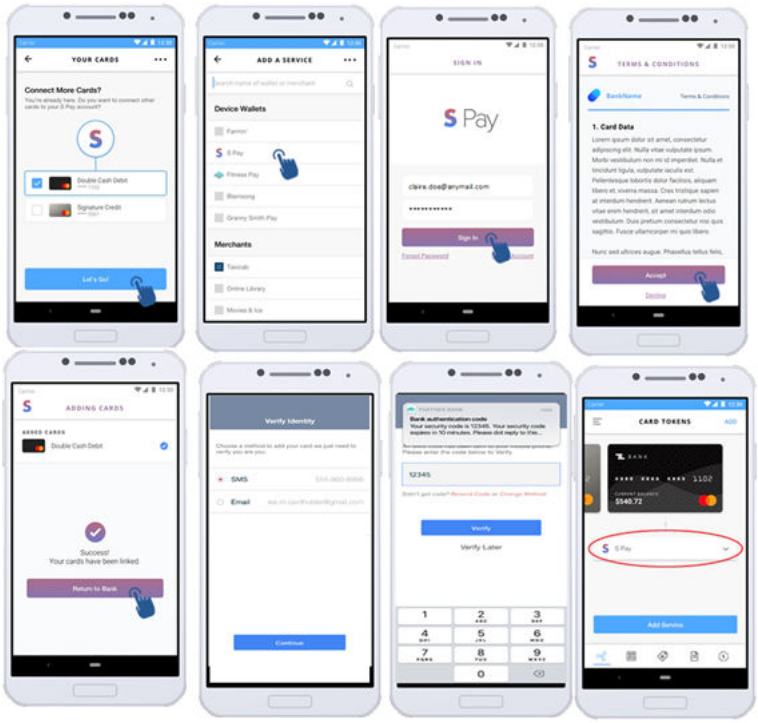


MDES Token Connect may support multiple user experiences such as:

- Web-to-web on mobile as well as desktop devices

- Web-to-app on Android and iOS mobile devices
- App-to-app on Android and iOS mobile devices
- App-to-web on Android and iOS mobile devices

Figure 2: User Experience: Authentication in Token Connect



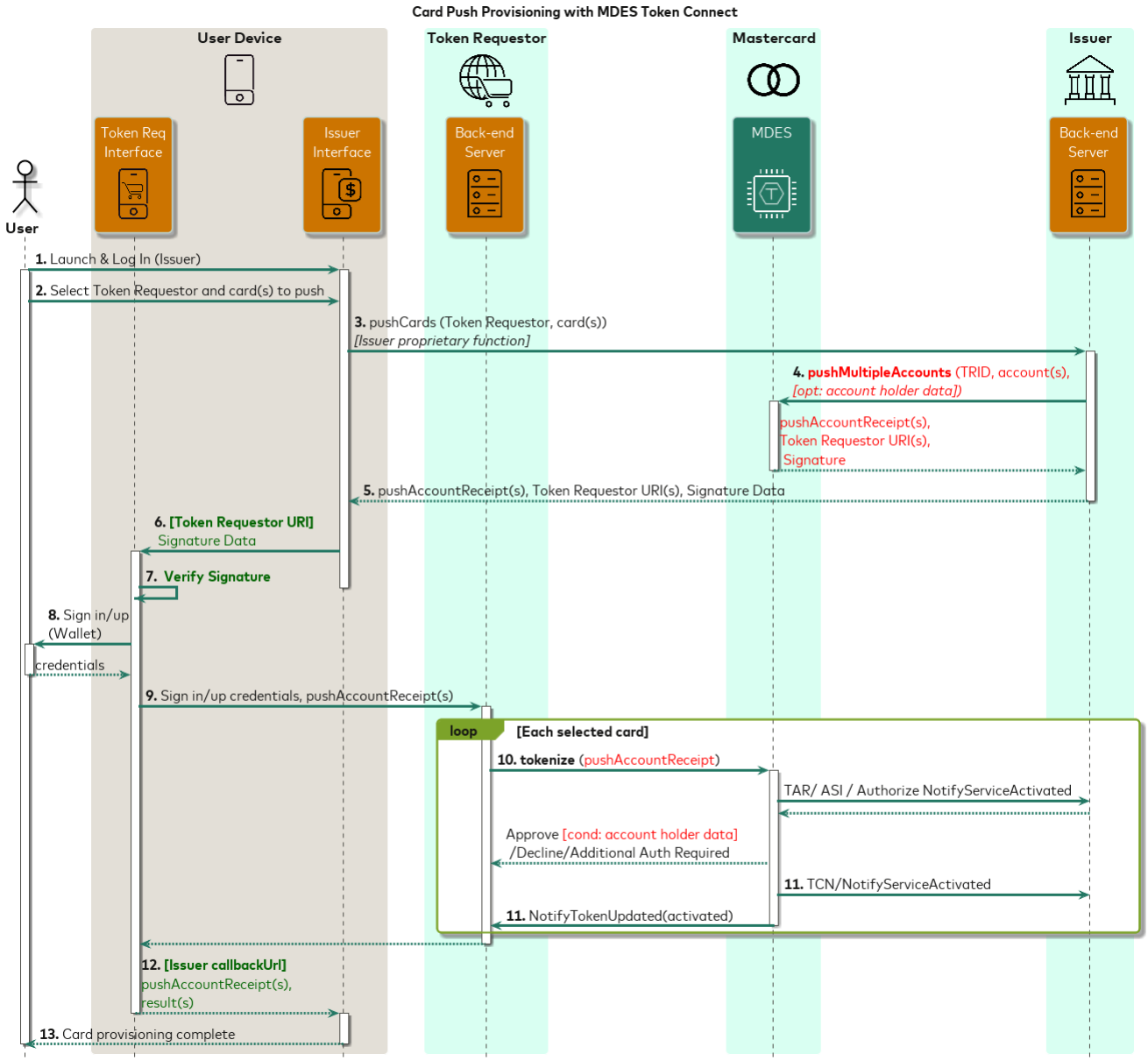
Token Connect workflows

The sequence diagrams below explain the tokenization process using the MDES Token Connect framework.

Merchant token requestor

In this example, the token requestor is a merchant. Cardholder Authentication is not required to activate the token.

Figure 3: Merchant token requestor



1. The consumer is logged in the issuer's app/website.
2. While navigating, they select the target token requestor as well as the card accounts to push.
3. The issuer's app or browser calls the issuer backend with consumer's choice.

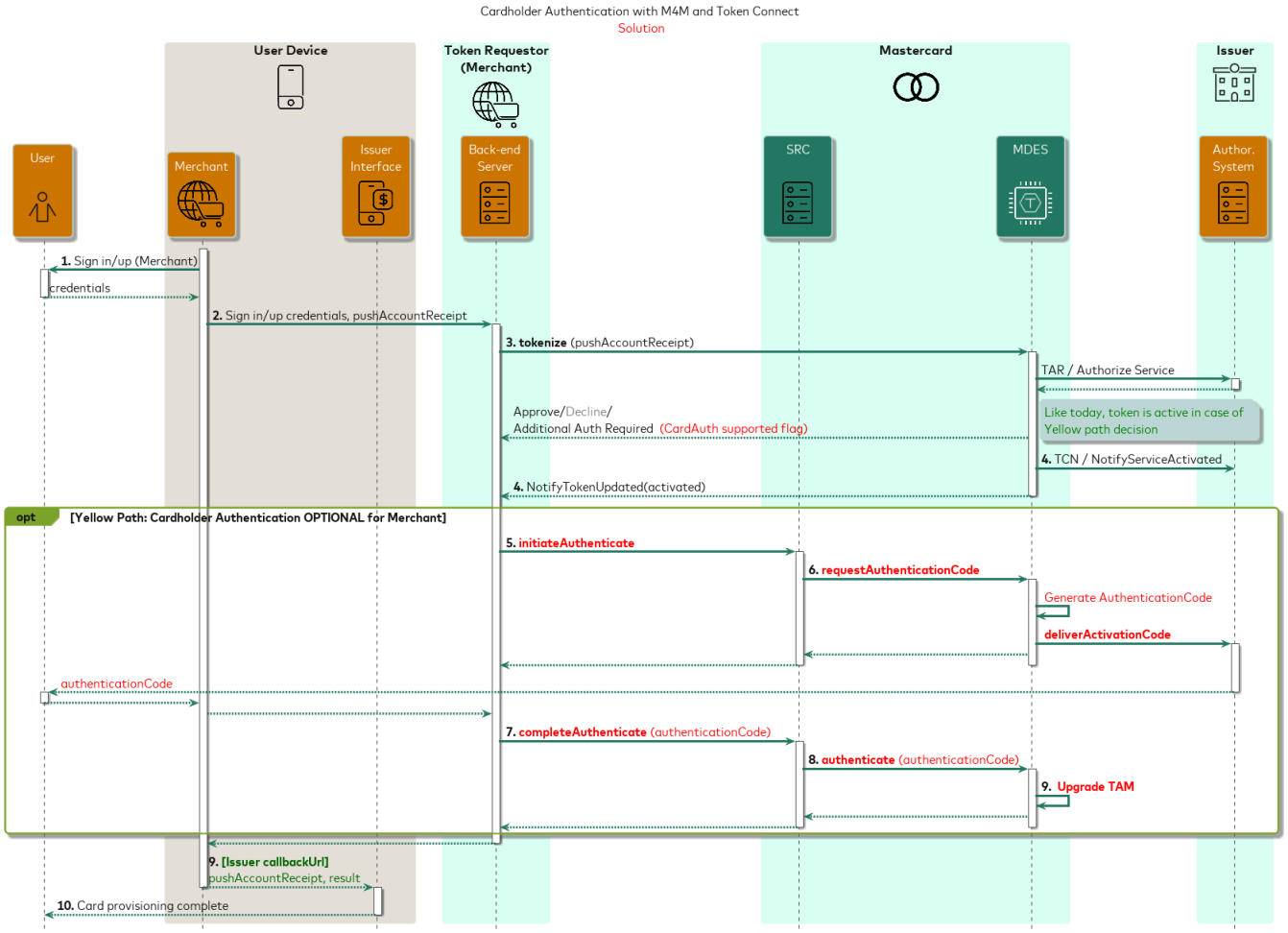
4. Using MDES Token Connect API, the issuer contacts MDES to request the push of the selected account to the target token requestor. The issuer may append account holder data (for instance, billing address) to the request, if supported by the token requestor. If the request is valid, MDES generates and returns a `pushAccountReceipt` – a string voucher valid for the next 15 minutes to be used by the target token requestor for the tokenization of the account. MDES also returns the URIs of the interfaces for this token requestor.
5. The issuer returns the `pushAccountReceipt` and the URIs to the issuer interface on the device.
6. The issuer interface selects the most appropriate URI corresponding to the device environment, and appends dynamic parameters to this URI, among others:
   If a token requestor supports signature verification, then:
   - Append signature value that is received in the response as a `pushAccountData` parameter.

   If a token requestor does not support signature verification, then:
   - The `pushAccountReceipt` parameter of the account being pushed. If the token requestor supports multiple cards pushed simultaneously, multiple `pushAccountReceipt` parameters may be appended.
   - The callback URL: the issuer URL that the token requestor calls out upon completion of tokenization to return the consumer into the issuer's interface where they started their journey.

   The issuer interface calls out the resulting URI, and as a result, the consumer is taken to the token requestor's interface (app or website).
7. The token requestor interface asks the consumer to sign in or sign up.
8. The token requestor interface calls out the token requestor backend to complete sign in and to pass the `pushAccountReceipt` parameter.
9. The token requestor sends a tokenization request to MDES. The only difference with a *regular* tokenization initiated from the token requestor's interface is that the enciphered account data is replaced by the `pushAccountReceipt` that was supplied by the issuer. MDES matches the `pushAccountReceipt` with the account data that was supplied by the issuer.
10. When the tokenization is successfully completed, MDES notifies the issuer and the token requestor.
    When multiple accounts are pushed, the tokenization process (steps 9 through 10) is repeated for each `pushAccountReceipt`. If supplied by the issuer, account holder data is transmitted to the token requestor at step 9 (when tokenization is approved).
11. The token requestor appends the list of `pushAccountReceipts`, with their respective results, to the callback URL that the issuer interface had provided. The token requestor calls out the resulting URI, and as a result the consumer is taken back to the issuer's interface (app or website).
12. The issuer interface notifies the consumer about the result of the provisioning.

Merchant token requestor with post tokenization authentication through through Remote Commerce Programs

In this example, the token requestor is a merchant. Cardholder Authentication is required to activate the token.



1. The consumer is logged in the issuer's app/website.
2. While navigating, they select the target token requestor as well as the card accounts to push.
3. The issuer's app or browser calls the issuer back-end with consumer's choice.
4. Using MDES Token Connect API, the issuer contacts MDES to request the push of the selected account to the target token requestor. The issuer may append account holder data (for instance, billing address) to the request, if supported by the token requestor. If the request is valid, MDES generates and returns a pushAccountReceipt – a string voucher valid for the next 15 minutes to be used by the target token requestor for the tokenization of the account. MDES also returns the URIs of the interfaces for this token requestor.

5. The issuer returns the pushAccountReceipt and the URIs to the issuer interface on the device.
6. The issuer interface selects the most appropriate URI corresponding to the device environment, and appends dynamic parameters to this URI, among others:
   If a token requestor supports signature verification, then:
   - Append signature value that is received in the response as a `pushAccountData` parameter.

   If a token requestor does not support signature verification, then:
   - The `pushAccountReceipt` parameter of the account being pushed. If the token requestor supports multiple cards pushed simultaneously, multiple `pushAccountReceipt` parameters may be appended.
   - The callback URL: the issuer URL that the token requestor calls out upon completion of tokenization to return the consumer into the issuer's interface where they started their journey.

   The issuer interface calls out the resulting URI, and as a result, the consumer is taken to the token requestor's interface (app or website).
7. The token requestor interface asks the consumer to sign in or sign up.
8. The token requestor interface calls out the token requestor backend to complete sign in/up and to pass the `pushAccountReceipt`.
9. The token requestor requests tokenization with MDES. The only difference with a *regular* tokenization initiated from the token requestor's interface is that the enciphered account data is replaced by the `pushAccountReceipt` that was supplied by the issuer. MDES matches the `pushAccountReceipt` with the account data that was supplied by the issuer. The issuer receives the tokenization authorization message.
10. If the issuer supports Post Tokenization Authentication Service and looking for authentication for this tokenization; the issuer must set token authorization decision as REQUIRE_ADDITIONAL_AUTHENTICATION and pass available activation methods:
    - TEXT_TO_CARDHOLDER_NUMBER (with masked mobile number as value)
    - EMAIL_TO_CARDHOLDER_ADDRESS (with masked email address as value)
11. If the token requestor and issuer both support Post Tokenization Authentication Service and the decision is REQUIRE_ADDITIONAL_AUTHENTICATION then the token requestor must display authentication screen to consumer to select a way to receive authentication code with retry option.

    NOTE: The `supportsAuthentication` flag value is `true` in the tokenize response if an issuer supports Post Tokenization Authentication Service.

12. On selection of channel to receive authentication code by consumer; the token requestor can initiate authentication using Remote Commerce Programs with ADD_CARD as reason code.

13. MDES generates authentication code and notifies the issuer through Post Tokenization Authenticaiton service (Activation Code Notification or Deliver Activation Code) with the reason code.
14. The issuer sends authentication code to the consumer.
15. After receiving authentication code from the consumer; the token requestor completes the authentication using Remote Commerce Programs.
16. On successful authentication; MDES upgrades Token Assurance Method and Card is added to token requestor.
17. The token requestor appends the list of `pushAccountReceipt`, with their respective results, to the callback URL that the issuer interface had provided. The token requestor calls out the resulting URI, and as a result the consumer is taken back to the issuer's interface (app or website).

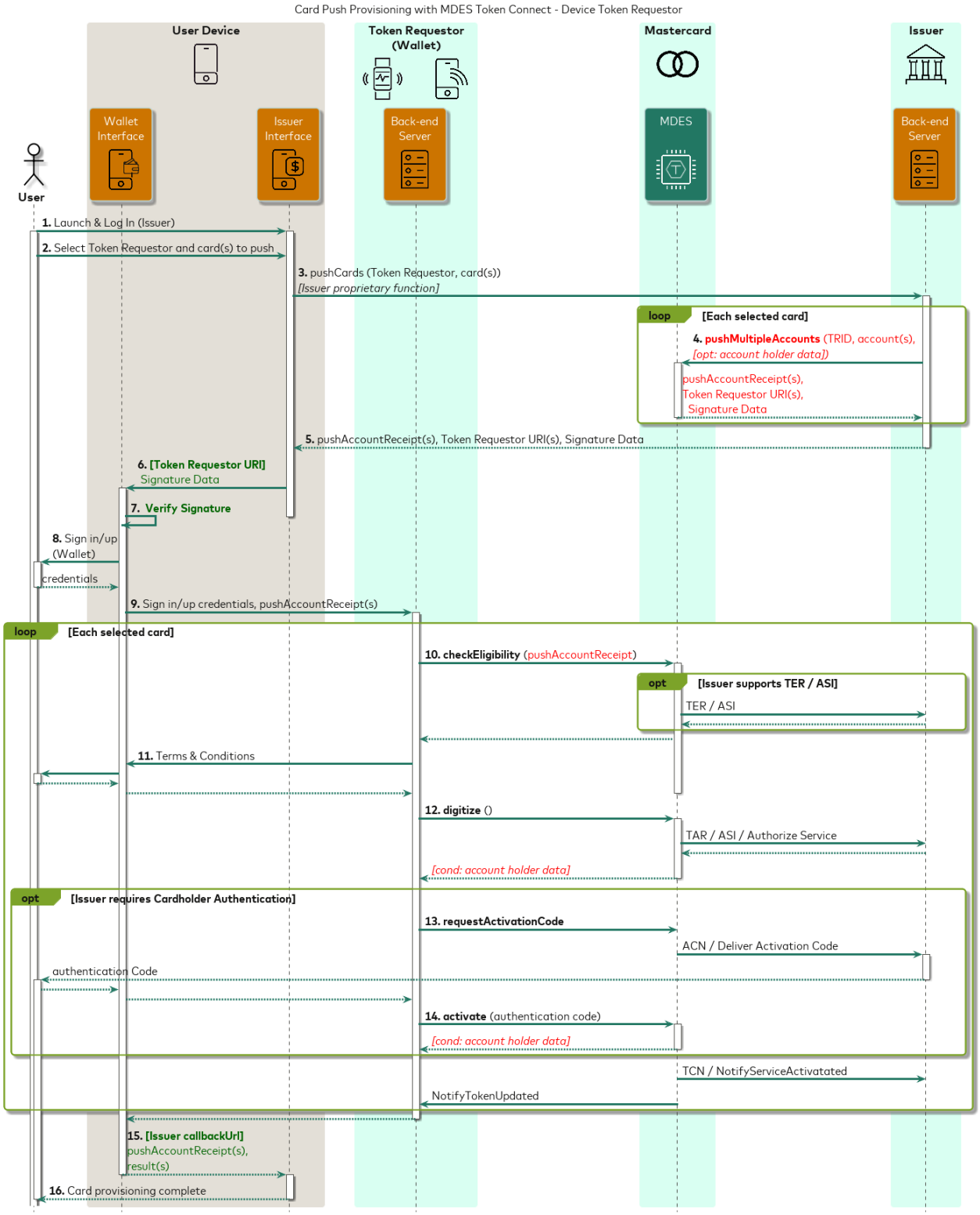
    NOTE: The token requestor will receive an active token even if the decision is REQUIRE_ADDITIONAL_AUTHENTICATION and the token requestor and the issuer both support Authentication Service.

    The token requestors must delete the token if authentication fails (including when the consumer does not complete authentication or authentication code has expired).

Wallet token requestor

In this example, the token requestor is a wallet provider for a device-based wallet. Cardholder Authentication is required to activate the token if the issuer asks for it.

Figure 4: Wallet token requestor



1. The consumer is logged into the issuer's app/website.
2. While navigating, they select the target token requestor as well as the card accounts to push.
3. The issuer app or browser calls the issuer backend with the consumer's choice.
4. Using MDES Token Connect API, the issuer contacts MDES to request the push of the selected accounts to the target token requestor. If the request is valid, MDES generates and returns a `pushAccountReceipt`, a string voucher valid for the next 15 minutes to be used by the target token requestor for the tokenization of the account. MDES also returns the URIs of the interfaces for this token requestor.
5. The issuer backend returns the `pushAccountReceipt` and the URIs to the issuer interface on the device.
6. The issuer interface selects the most appropriate URI corresponding to the device environment, and appends dynamic parameters to this URI, among others:
   If a token requestor supports signature verification, then:
   - Append signature value that is received in the response as a `pushAccountData` parameter.

   If a token requestor does not support signature verification, then:
   - The `pushAccountReceipt` parameter of the account being pushed. If the token requestor supports multiple cards pushed simultaneously, multiple `pushAccountReceipt` parameters may be appended.
   - The callback URL: the issuer URL that the token requestor calls out upon completion of tokenization to return the consumer into the issuer's interface where they started their journey.

   The issuer interface calls out the resulting URI, and as a result, the consumer is taken to the token requestor's interface (app or website).
7. The token requestor interface asks the consumer to sign in or sign up.
8. The token requestor interface calls out the token requestor backend to complete sign in and to pass the `pushAccountReceipt` parameter.
9. The token requestor checks card availability with MDES. The only difference with a *regular* tokenization initiated from the token requestor's interface is that the enciphered account data is replaced by the `pushAccountReceipt` that was supplied by the issuer. MDES matches the `pushAccountReceipt` with the account data that was supplied by the issuer.
10. The consumer accepts the issuer's Terms and Conditions in the token requestor interface.
11. The token requestor checks the card eligibility with MDES, exactly as in a *regular* tokenization. The issuer receives the tokenization authorization message.
12. Note that depending on the issuer's configuration, cardholder authentication may be required to activate the new token.

13. When the tokenization is successfully completed, MDES notifies the issuer and the token requestor.
    When multiple accounts are pushed, the tokenization process (steps 9 through 12) is repeated for each `pushAccountReceipt`.
14. The token requestor appends the list of `pushAccountReceipts`, with their respective results, to the callback URL that the issuer interface had provided. The token requestor calls out the resulting URI, and as a result, the consumer is taken back to the issuer's interface (app or website).
15. The issuer interface notifies the consumer about the result of the provisioning.

Chapter 3 Before you integrate

To integrate with Token Connect, token requestors must fulfill few requirements.

Integration requirements

Integrating with Token Connect requires two main developments for a token requestor:

- Set up the ability for your backend system to request the tokenization of a *pushAccountReceipt* (payment credentials alias) in lieu of encrypted payment credentials. Refer to Integration prerequisites and Integrating with Token Connect for further details.
- Complete changes to your user interfaces to accommodate the consumer journey when receiving a *pushAccountReceipt* from an issuer. Refer to Use cases, Deep linking guidelines, Integrating with Token Connect, and Communication specifications for further details.

These are the primary items to consider when evaluating Token Connect for your business. All other flows are standard to the MDES tokenization process.

For further details on the implementation process for Token Connect, refer to Implementation process.

Integration prerequisites

The support of Token Connect is optional for MDES token requestors.

A token requestor supporting the Token Connect framework needs to provide the following additional setup parameters to MDES:

- An image that consumers will be able to see from their issuer's app/website. This image should be a logo that consumers can associate with the brand.
- A consumer-facing name for their entity, as it will appear to the consumer, next to the logo, in the issuer's app/website. This name may differ from the company's legal name.
- The URIs that the issuer's app/website can call out so as to put the consumer in direct relationship with the token requestor's interface. URIs can be provided for several user interfaces, but at least one is required:
  - URI for Android application interface
  - URI for iOS application interface
  - URI for Web browser interface
- A confirmation whether the token requestor accepts multiple cards (a maximum of five) pushed simultaneously.
- A confirmation whether the token requestor accepts to receive additional account holder information (such as name, billing address, email address, phone number) from the issuer.

- A confirmation whether the token requestor supports authentication services (applicable only to merchants).
- A confirmation whether the token requestor supports signature verification or not.

NOTE: When enabled for Token Connect, an issuer can retrieve the Token Connect parameters of each token requestor for which they have enabled their account ranges from MDES. Issuers can retrieve this information using the MDES Token Connect API for issuers.

We recommend that issuers refresh this information periodically, at their convenience. Issuers can use this information to create a consumer-friendly list of token requestors to which consumers can push their account and build the consumer experience of their app/ website.

Chapter 4 Integrating with Token Connect

Review and follow guidelines and instructions when integrating with Token Connect.

Tokenization guidelines

When checking eligibility or tokenizing, token requestors must use the receipt supplied by the issuer in lieu of the enciphered card or financial account data. The `pushAccountReceipt` is valid for 15 minutes after its issuance by MDES, and it is reserved for the exclusive use of the token requestor for which it was requested. To avoid replay, the receipt can be used successfully only once.

The following requirements are applicable:

- Token requestors must set input parameter *source* to value ACCOUNT_ADDED_VIA_APPLICATION to indicate that the issuer's app or website has triggered the tokenization.
- Token requestors must populate the *deviceIpAddress* value in DecisioningData structure with the IP address of the desktop or mobile device. This will allow the issuer to support device binding, by comparing the IP address of the device hosting the issuer interface and the device hosting the token requestor interface.
- If the consumer has followed a desktop-to-mobile path, the wallet provider must require additional authentication with the reason `Low device score`, as the pairing operation between desktop and mobile represents an additional risk of fraud.
- If the token requestor supports only a single pushed account and receives (by mistake) multiple *pushAccountReceipts* from the issuer, they will process the first one and ignore the subsequent ones.

Some token requestors need to know the acceptance brand of the accounts being pushed by the issuer. This typically concerns wallet applications hosted in the Secure Element (SE) of a device. Token requestors can easily deduce the acceptance brand associated to a *pushAccountReceipt* by checking the 3-character prefix of the *pushAccountReceipt*, as follows:

MCC	Mastercard Credit
DMC	Mastercard Debit/Pay by Account (PBA)
MSI	Maestro
PVL	Private Label

Example: A *pushAccountReceipt* with the value `MCC-C307F0AE-298E-48EB-AA43-A7C40B32DDDE` corresponds to a *Mastercard Credit* card.

NOTE: Token requestors receive the details of the card being pushed (such as card visual, last 4 digits of the FPAN and such details) only after successful digitization.

Token retention restrictions

Here are some token retention restrictions.

- After a successful digitization, token requestors should check that the newly created token doesn't duplicate an existing token for the same user account (and same user device, if applicable). The token requestor may typically use the `panUniqueReference` to detect duplicate tokens. If the new token duplicates an existing token, the token requestor should delete the new token and inform the consumer (with a confirmation) that the card (or account) was already digitized previously, before returning the consumer back to the issuer with a result of CANCELLED for the `pushAccountReceipt`.
- The new token may only be used, retained or otherwise processed only after:
  1. Privity is established with the account holder, such as a result of the successfully completed user registration process; and
  2. There is successful tokenization and completion of the push provisioning Token Connect process as defined by the program, this document and related specifications.
- Token requestor must delete the token using the Delete API if the privity is not established with the account holder

User experience

The consumer should be able to cancel the provisioning and return to the issuer's interface at any moment before the completion of tokenization.

If the token requestor is a wallet provider, like in a standard digitization, the wallet provider must ensure that the consumer accepts the issuer's terms and conditions. If multiple accounts are being pushed and the same terms and conditions apply, the consumer will accept them only once.

Token requestors should not ask for the CVC2 security code, even if a CVC2 is applicable for the card being tokenized. A significant part of the value proposition of Token Connect for the consumer consists of simplifying their digitization experience by avoiding manual entry of card details. The consumer does not need to have the card physically nearby for issuer-initiated digitization. At their convenience, issuers can decide to step up with an additional cardholder authentication.

Upon successful tokenization, the token requestor should give the cardholder an option to select a pushed card as the default payment method.

Receiving issuer account holder data

Some token requestors may be interested in receiving account holder data from the issuer during a push provisioning operation, such as name, billing address, email address, and mobile phone number.

The email address and mobile number may typically be used by the token requestor to reduce friction in case the customer wishes to sign up for a new

account in the token requestor's interface - for instance by displaying an account creation form pre-filled with the information supplied by the issuer. Note that in this case, the token requestor should digitize the `pushAccountReceipt` before requesting sign in/sign up from the consumer, as the issuer-supplied account holder data is provided by MDES only after a successful digitization. (MDES strongly recommends that the token requestor initiates provisioning after an account holder has completed signs in/signs up to avoid privity not established with account holder.)

In some markets, a name and a billing address are required for any stored payment instrument, and token requestors can take advantage of this information supplied by the issuer to reduce friction in facilitating cardholder transactions.

This facility is currently available for MDES **merchants** and **commerce platforms**.

Which data elements can be received from the issuer?

When onboarding for MDES Token Connect, token requestors must declare which of the following account holder data elements they wish to receive from the issuer:

- Name
- Billing Address
- Email Address
- Mobile Phone Number

At their discretion, issuers may elect to transmit any combination of account holder data elements supported by a particular token requestor.

How to retrieve the account holder data?

If the tokenization decision is APPROVED, the token requestor receives the account holder data (if any) from MDES in the response to the *tokenize* request.

To mitigate the risk of identity theft, if the tokenization decision is REQUIRE_ADDITIONAL_AUTHENTICATION or DECLINED, MDES doesn't supply the account holder data.

NOTE: When the issuer provides account holder data, MDES stores this information temporarily for the sole purpose of transmitting this information to the intended token requestor. MDES immediately erases the account holder data once they are successfully retrieved by the token requestor or, if not retrieved, the account holder data are erased at the earliest opportunity.

How to process account holder data?

The token requestor must comply with the privacy and legal obligations outlined below.

There may be situations where the token requestor will receive issuer account holder data that is different from the actual token requestor's customer account data. The token requestor must be able to address such data conflicts. The way

the data conflict is addressed is at the token requestor's discretion (for example: discard issuer data, replace token requestor data, ask consumer).

Streamlined activation via issuer's application/website

When the issuer triggers a token provisioning with Token Connect, the consumer is logged into the issuer's environment (consumer has been authenticated by the issuer). Leveraging this pre-authentication of the cardholder in the issuer's interface, Token Connect provides the opportunity to tokenize cards without any additional step-up provided that the issuer has set up appropriate security controls. Nevertheless, there are still situations where the issuer will take the decision that an additional cardholder authentication is needed before the activation of the token.

NOTE: The support of streamlined activation is mandatory for token requestors (wallet providers) supporting either issuer banking application or issuer website as cardholder authentication methods for token activation. Otherwise, streamlined activation is not applicable.

Cardholder authentication and token activation follow the same principles as for a digitization initiated by the wallet provider. While the consumer is in the token requestor's interface, MDES returns the list of available authentication methods, and the consumer selects their authentication method.

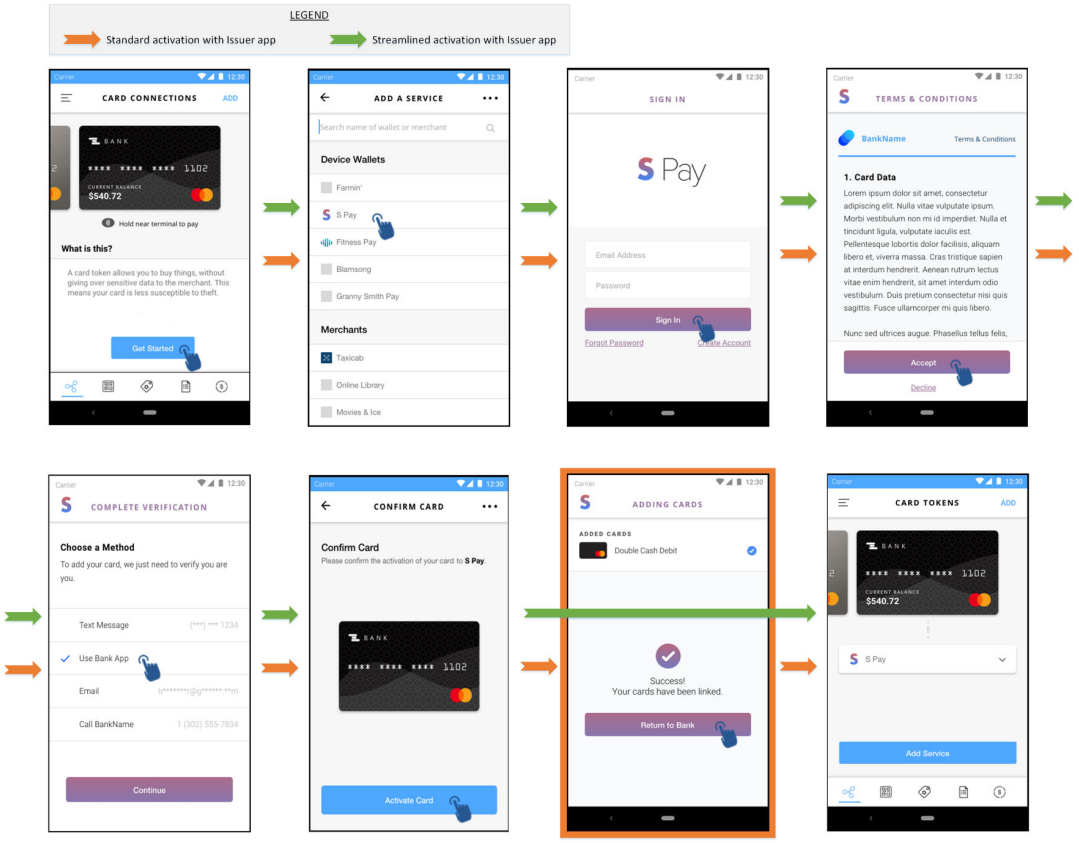
If the cardholder selects an activation through the issuer's app or website, in theory the user experience may become very painful, as the cardholder would be taken from the token requestor environment to the issuer environment (for activation), then back to the token requestor (activation complete), then back again to the issuer (provision through Token Connect complete).

To avoid these back-and-forth interface toggles and to provide a better experience, the issuer may decide to use **streamlined activation** with the issuer's app or website. The issuer will pass additional parameters in the URL to the token requestor which will indicate that if the consumer selects activation via the issuer app (respectively website), the token requestor should NOT try to activate the token using the object supplied by MDES, and should return control to the issuer via the Token Connect callback URL after all tokenizations are completed (in the case of multiple pushed accounts). In this case, the cardholder will activate the token in the issuer's app (respectively website) when driven back there. The issuer may either bypass additional authentication or require the consumer to complete authentication at this point. The issuer backend uses MDES Customer Service API to request the activation of the token to MDES.

See Request query string parameters from issuer to token requestor for more details about the URL query string parameters `completeIssuerAppActivation` and `completeWebsiteActivation` that issuers will use to request streamlined activation from the token requestor.

The figure below illustrates the difference between a standard activation and a streamlined activation using issuer application:

Figure 5: Standard activation streamlined activation diff



Handover to issuer

Using the parameters in the redirection URL, the issuer specifies the desired behaviour after the push provisioning operation. Issuer decides between three options :

- Consumer must return to the isuer interface
- Consumer must stay in the token requestor interface
- Consumer may either stay in the token requestor interface or return to the issuer interface (at token requestor/consumer's convenience)

If the issuer requires a callback to their interface, token requestors shall limit the interaction with the consumer to the very minimum to achieve the card or account provisioning, before sending the consumer back to the issuer's environment.

If the issuer requires a callback to their interface, regardless of whether the provisioning operation was successful or not, the consumer must always be strongly guided to return to the issuer's environment where they started their journey. Typically an obvious **Continue to bank** or **Back to bank** button should be

available when the issuer requires a callback to their interface. If the wallet provider has driven the consumer through a desktop-to-mobile experience, the consumer must be taken back to the issuer's environment on the desktop computer (NOT on the mobile).

The token requestor uses the callback URL supplied by the issuer to return the consumer back to the issuer's environment. The token requestor appends the list of `pushAccountReceipts` and their respective results to the URL to inform the issuer of the digitization outcomes. See Response query string parameters from token requestor to issuer for more details on the format of the parameters appended to the URL.

Display provisioning results

The token requestor should inform the consumer about the provisioning outcome. Likewise, if there is a redirection back to the issuer interface, the token requestor must indicate the result to the issuer.

For each provisioning outcome, the table below provides guidance on messages to be displayed to the consumer and indicates the result to be returned to the issuer:

Provisioning outcome	Suggested message to consumer	Result supplied to issuer
Consumer cancels provisioning operation before tokenization	*"Are you sure you want to cancel the card add operation (Y/N)?"* ➜ Yes – Back to Bank (button) ➜ No (button)	CANCELLED
Token requestor cancels provisioning before tokenization	*"We are sorry, your payment card could not be added. <Optional: Put here the reason, or action needed to fix issue>"* ➜ Back to Bank (button)	CANCELLED
Successful tokenization	*Congratulations! Your payment card has been successfully added!* <Display Card Visual and last 4 digits of FPAN> ➜ Continue to Bank (button)	APPROVED

Tokenization declined	*"Card could not be added. Contact your Bank."* ➔ Back to Bank (button)	DECLINED
Wallets only: Successful tokenization requiring additional cardholder authentication	<Display pending Card Visual and last 4 digits of FPAN> & < Display the usual menu to select an activation method>	-
	<If consumer selects a method requiring Streamlined Activation:> *"You will now be redirected to your Bank application [respectively website] to activate your card."* ➔ Continue to Bank (button)	REQUIRE_ ADDITIONAL_ AUTHENTICATION
	<Else (no Streamlined activation): Execute as usual the activation method selected by consumer – manage retries as usual>	-
	<If authentication successful:> *"Congratulations! Your payment card has been successfully added!"* <Display Card Visual and last 4 digits of FPAN> ➔ Continue to Bank (button)	APPROVED
	"Authentication failed" ➔ Continue to Bank (button)	REQUIRE_ ADDITIONAL_ AUTHENTICATION

Merchants only: Successful tokenization requiring additional cardholder authentication	*Congratulations! Your payment card has been successfully added!* <Display Card Visual and last 4 digits of FPAN><If authentication is not successful:> ➔ Continue to Bank (button)	REQUIRE_ADDITIONAL_AUTHENTICATION
Merchants only Supports authentication and authentication is successful	*Congratulations! Your payment card has been successfully added!* <Display Card Visual and last 4 digits of FPAN> ➔ Continue to Bank (button)	AUTHENTICATED
Merchants only Supports authentication and authentication is failed, and merchant has deleted token	*Congratulations! Your payment card has been successfully added!* <<Display Card Visual and last 4 digits of FPAN> ➔ Continue to Bank (button)	DELETED
Tokenization error	*"Card could not be added. Contact your Bank."* ➔ Back to Bank (button)	ERROR
New token duplicates an existing token for same {card, user account} or {card, user account, user device} (as applicable) – New token is deleted	*"This card is already associated with your wallet/account"* ➔ Continue to Bank (button)	CANCELLED DUPLICATE_REQUEST
Consumer deletes card after successful tokenization	*"Are you sure you want to cancel the card add operation (Y/N)?"* ➔ Continue to Bank (button)	CANCELLED
Issuer hasn't supplied any `pushAccountReceipt`	*"Card could not be added. Contact your Bank."* ➔ Back to Bank (button)	(No result appended to the issuer Callback URL)

Multiple cards/accounts with mixed tokenization results (approved, declined, error...)	*"Congratulations! The following card has been successfully added:* <Display Card Visual and last 4 digits of FPAN> *Some of your cards could not be added. Contact your Bank."* ➔ Continue to Bank (button)	List of results corresponding to each `pushAccountReceipt`.
Received "DUPLICATE_REQUEST" error code in tokenization request for same {card, user account} or {card, user account, user device} (as applicable) –	*This card is already associated with your wallet/ account* ➔ Continue to Bank (button)	DUPLICATE_REQUEST

Deep linking guidelines

With the exception of the few token requestors that do not support any device app (case 1), all other token requestors will have to overcome the difficulty of always offering the simplest consumer experience regardless of the consumer's situation.

- Device app is already installed
- Device app is not yet installed

As the User experience guidelines have shown, there are many situations where the token requestor will have to support a behavior like *If my app is installed then launch it at a specific location, otherwise fallback to my web server at a specific page* (*deep linking*).

Besides, when the app first needs to be installed, for an optimized experience, the initial launch of the app should feature the specific contextual page related to the accounts currently being pushed by the issuer, rather than the generic welcome page (*deferred deep linking*).

However, the techniques to implement such a behavior are complex, as they differ depending on the mobile device OS and the OS versions, the browser and the browser version, and there are many special cases.

Deep linking with Android

Follow these guidelines when implementing deep linking for Android.

App Links

Android **App Links** are generic HTTP or HTTPS URIs to your web server that your Android app, if installed, will intercept to display the related content in-app rather than in a browser. By default, if your app is not installed, the consumer is directed to the URI where they will view the contents in the mobile browser.

Android App Links are a promising tool, yet their biggest drawback is that they are supported exclusively from Android version 6.0 (API level 23). Currently, an Android solution based exclusively on App Links would not work for thirty three percent of Android users, which is of course unacceptable. **Mastercard does not recommend App Links as a unique Android solution** until they are supported more widely.

Custom URI schemes and Intents

For now, token requestors may leverage **Custom URI schemes** that are less powerful, but universally supported. With a custom URI scheme, the URI to open your Android app will look like `yourappname://deep/link/path/to/pushcard/` . The calling entity (app or browser) will use an Android Intent with this URI to launch your app.

The main drawback with Custom URI schemes is that it will work only if the app is installed. The entity calling out the Custom URI scheme must manually implement the fallback to the matching HTTPS web page, instead of being automatically redirected there by the OS. In the context of Token Connect, this implementation burden may fall on the issuer (issuer app source code, script on issuer web server) or on the token requestor (script on filtering landing page).

If you onboard a Custom URI scheme to MDES as Android URI, on Android devices, the issuer's app or website will use an implicit Android intent to try and call out your app. Issuers will use the following parameters:

- Action: android.intent.action.VIEW (Intent.ACTION_VIEW)
- URI: your Custom URI scheme (for example: `yourappname://deep/link/path/to/pushcard/`)

The Manifest File of your app will have to define an intent filter so as to accept the implicit intent requests from apps or from the browser. To support implicit intents and requests from the browser, make sure to include the following lines in the intent filter:

```
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.BROWSABLE" />
```

NOTE: Some browsers automatically convert the uppercase chars in the scheme of a URI to lowercase. Following the recommendation of RFC 3986 (https://tools.ietf.org/html/rfc3986#section-3.1), to avoid interoperability issues during a web-to-mobile experience, token requestors must specify exclusively lowercase custom URI schemes (`yourappname://` instead of `YourAppName://`).

References

App Links: https://developer.android.com/training/app-links/

Intents and Custom URI scheme: https://developer.android.com/training/basics/intents/

Other tutorial to Android deep linking: https://blog.branch.io/technical-guide-android-deep-linking-uri-schemes/

Deep linking with iOS

Follow these guidelines when implementing deep linking for iOS.

Universal links

The mechanism used by iOS for app-to-app communication, with fallback to a default web URL is called **Universal Links**. Universal links are enabled differently, yet they are used exactly as Android app links. A Universal Link looks like a standard HTTPS link, but it opens a special location in the iOS app that registered it if this app is installed in the iOS phone.

Universal links have been supported since iOS version 9, and is currently supported by more than ninety nine percent of iOS phone users. **Mastercard highly recommends that you exclusively use Universal Links for iOS**, especially when considering the complications that could arise from the usage of Custom URI schemes.

Custom URI schemes

As in the case of Android, iOS supports custom URI schemes, and URIs will look exactly the same as in your Android: `yourappname://deep/link/path/to/pushcard/` . As is the case for Android, Custom URI schemes are supported by all OS versions.

There is however a disqualifying drawback with Custom URI schemes where, in order to know whether a custom URI scheme will resolve on the iOS device, the calling app must register the custom URI schemes that it will attempt to call (for example: `yourappname`) in its `Info.plist` configuration file. In other words, the list of Custom URI schemes that an app can call (for instance, the issuer app) is entirely static and cannot be updated dynamically. This cannot work for Token Connect, as Token Connect relies on a list of token requestor apps that evolve dynamically without needing to regenerate and republish the issuer apps that calls them.

NOTE: Token requestors should not onboard Custom URI schemes to MDES for iOS.

References

Universal links:

https://developer.apple.com/documentation/uikit/core_app/allowing_apps_and_websites_to_link_to_your_content

Custom URI schemes:

https://developer.apple.com/documentation/uikit/core_app/allowing_apps_and_websites_to_link_to_your_content/defining_a_custom_url_scheme_for_your_app

https://developer.apple.com/documentation/uikit/uiapplication/1622952-canopenurl

Deferred deep linking

Deferred deep linking routes consumers to the right app page even if the app is not installed when the link is opened. After a first redirection to the App Store or Play Store to download the app, the consumer is taken to the specific *deferred* content upon first launch of the app.

Operating system	Details
Android	Deferred deep linking may be implemented using the Google Play Referrer. https://blog.branch.io/technical-guide-to-google-play-referrer/ https://developer.android.com/google/play/installreferrer/
iOS	iOS does not support any native deferred deep linking mechanism. Each token requestor should therefore develop their own deferred deep linking mechanism for iOS. https://stackoverflow.com/questions/30352647/how-to-make-deferred-deep-linking

Key takeaways about deep linking

All token requestors supporting MDES Token Connect need to support deep linking so that consumers can be directed to the right content when the issuer calls out the token requestor's interface.

For token requestors belonging to Case 1 and Case 2 categories where device apps are not supported or not needed, the deep linking solution is quite straightforward. For Case 2, the behavior can be summarized to *try to open the app if present on the device, otherwise continue with your web experience*: the implementation may

rely on simple native mechanisms supported respectively by Android (custom URL schemes associated to app links) and iOS (universal links).

For token requestors belonging to Cases 3, 4 and 5, the presence of the device app is mandatory. If the app is not present when the consumer pushes the card, the consumer must be taken to the store to download the right app, and when opened for the first time, the app must display the expected contents in relation to the card being pushed (deferred deep linking). Such implementations are far more complex and full of special cases, with dependencies on OS, OS version, browser and browser version.

Deferred deep linking is however necessary for a correct consumer experience if the device app is required but not pre-installed. **For token requestors belonging to Cases 3, 4 and 5, the support of Deferred Deep Linking is mandatory.**

If the token requestor decides to implement their deep linking solution in-house, they need to be aware of this difficultly, and allocate sufficient time and resources to overcome it. Alternatively, the token requestor may decide to use the turn-key services of a hosted deep link provider to simplify their developments. If you opt for such hosted services, **you should make sure NOT to disclose sensitive information to the third party**, such as `pushAccountReceipts` and an issuer callback URL. An easy workaround is to create a session ID, store it and associate it with the sensitive information from the issuer on your server, before calling the third party provider with your session ID as input. When the provider has taken the consumer to the adequate interface and returns your session ID, you can recover the original sensitive information from the session ID to proceed with the digitization.

References

https://medium.com/@ageitgey/mobile-deep-linking-part-2-using-a-hosted-deep-links-provider-c61fa58d083e

Configure URIs for issuer requests with MDES

MDES gives token requestors the flexibility to configure up to three URIs.

- One URI for their Android app
- One URI for their iOS app
- One URI for the web browser experience

MDES transmits this or these URIs to issuers when they push one or more accounts to the token requestor, before the issuer calls out the token requestor interface.

The presence or absence of an URI informs the issuer that the token requestor may or may not have an Android (resp. iOS) app and a web server to support digitization. Even if your app and web URIs are identical (for instance, in the case of a universal link), both must be filled in (with the same value, in this case) to inform the issuer accurately.

Issuer interfaces behave as follows:

- On a desktop computer, the issuer interface calls the web browser URI.
- On an iOS device, the issuer interface calls the iOS URI (universal link) if available, and if not, the web browser URI (when available).
- On an Android device, the issuer interface tries the Android-specific URI. If the token requestor has not provided such a link or if the link does not resolve, the issuer interface tries the web browser URI (when available).

Token requestors must maintain the URIs onboarded in MDES for Token Connect so that they are functional at all times, for all issuers that have enabled the token requestor. Mastercard reserves the right, after notification to the token requestor, to disable the token requestor from MDES Token Connect framework if it is discovered that the URIs are not/no longer functional, until the token requestor proves with valid test results that they are functional again.

The table below summarizes how token requestors should configure their URIs during MDES onboarding for Token Connect support. For more information on Cases 1 to 5, please refer to Use cases.

	Web browser URI	Android URI	iOS URI
Case 1 No app	Standard HTTPS URL to web server landing page	-	-
Case 2 Optional single app	Standard HTTPS URL to web server landing page	Custom URI scheme, or Standard HTTPS URL (can be the same as web browser URI) that will redirect to custom URI scheme	Universal link (can be the same as web browser URI)
Case 3 Mandatory single app, no device filtering	Conditional: Present only if supporting Desktop-to-Mobile flow, absent otherwise. Standard HTTPS URL to landing page on web server.	Standard HTTPS URL (can be the same as web browser URI) that will redirect to custom URI scheme	Universal link (can be the same as web browser URI)
Case 4 Mandatory single app, device filtering	Standard HTTPS URL (can be the same as web browser URI) that will filter devices and redirect to custom URI scheme	Universal link (can be the same as web browser URI)	

	Web browser URI	Android URI	iOS URI
Case 5 Mandatory app, device filtering, multiple apps	Unique standard HTTPS URL to filtering landing page on web server. The filtering landing page then gives the choice between several device-specific HTTPS deep links that will open the device-specific app if present, or else it will send the consumer to Google Play or App Store to download and then open the app with the right content.		

Post-Tokenization Authentication for Remote Commerce Programs

After tokenization (provisioning) token requestor may want to authenticate the customer.

- If the issuer supports Authentication Service and is required to authenticate for this tokenization, the issuer must set token authorization decision as REQUIRE_ADDITIONAL_AUTHENTICATION and pass one or all of the following activation methods:
  - TEXT_TO_CARDHOLDER_NUMBER (with masked mobile number as value)
  - EMAIL_TO_CARDHOLDER_ADDRESS (with masked email address as value)

  NOTE: Existing version of Authentication Service supports one-time password (OTP) based authentication only and supported delivery channels are SMS (text) message and email address.
- If the token requestor supports Authentication Service and all the following conditions are met:
  - Token requestor wishes to authenticate cardholder
  - Tokenization decision is REQUIRE_ADDITIONAL_AUTHENTICATION
  - `supportsAuthentication` response parameter value is `true`

  Then, the token requestor must display an authentication screen to the consumer to select a way to receive the authentication code with retry option.

- On selection of channel to receive authentication code by consumer, the token requestor initiates authentication using the Request Authentication Code service with ADD_CARD as reason code.
- After receiving authentication code from the consumer, the token requestor completes the authentication using Authentication service.
- On failed authentication (response of authentication service is INCORRECT_CODE_RETRIES_EXCEEDED) token requestor must delete the token using the Delete API. If the authentication code expires or token requestor entry field times out (for example, due to no consumer response), token requestor must delete the token using the Delete API.
- On successful authentication, MDES upgrades Token Assurance Method.
  - Response to the issuer (Callback)
    - On successful authentication, token requestor must pass AUTHENTICATED as the result while redirecting to the issuer system using callback.
      The resulting (raw) URI is:

      ```
      https://www.moonbank.com/pushcallback?session_id=789456123&results
      [MCC-C307F0AE-298E-48EB-AA43-A7C40B32DDDE]= AUTHENTICATED |DWSPMC00000
      0000132d72d4fcb2f4136a0532d3093ff1a45
      ```

    - Consumer authentication failed (response of authentication service is INCORRECT_CODE_RETRIES_EXCEEDED) and token requestor has deleted the token then must pass DELETED as a result while redirecting to the issuer system using callback.
      The resulting (raw) URI is:

      ```
      https://www.moonbank.com/pushcallback?session_id=789456123&results
      [MCC-C307F0AE-298E-48EB-AA43-A7C40B32DDDE]= DELETED |DWSPMC00000000013
      2d72d4fcb2f4136a0532d3093ff1a45
      ```

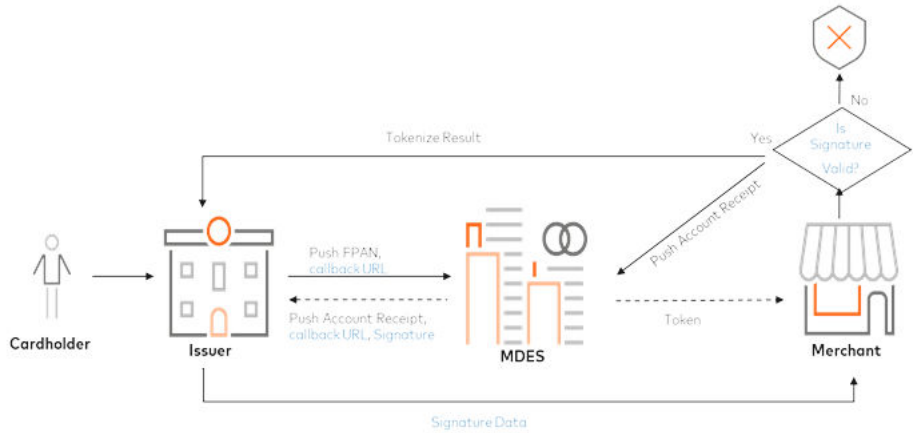
    - There is no change in result value if consumer does not complete authentication.
      The resulting (raw) URI is:

      ```
      https://www.moonbank.com/pushcallback?session_id=789456123&results
      [MCC-C307F0AE-298E-48EB-AA43-A7C40B32DDDE]= REQUIRE_ADDITIONAL_AUTHENT
      ICATION |DWSPMC000000000132d72d4fcb2f4136a0532d3093ff1a45
      ```

  - Authentication Code
    - Authentication after token activation is configured to support three attempts to enter a valid code, after which the code becomes invalid
    - When a token requestor requests to re-send an authentication code, the code with the same expiration date and time is re-sent to the issuer in a new DAC/ACN message, unless the code has expired
    - Authentication code expires in 15 minutes as of now it is not configurable for the issuer.

Signature verification

This section provides information on the signature verification functionality.



If an issuer and token requestor both support signature in Push Account, token requestor will receive the following parameter.

Parameter Name	Description
`pushAccountData`	Mandatory Contains push account data in JWS format

To verify signature token requestor will:

- Retrieve `kid` from JWS Protected Header
- Retrieve public certificate for `kid` (downloaded offline)
- Use public certificate to verify signature for given JWS Payload (before decoding)
- Respond to issuer with HTTP 400 error if signature verification fails

After successful verification of signature, decode JWS Payload and use as normal.

Example URL

```
https://myTR.com/pushAccount?pushAccountData=ew0KImFsZyI6ICJSUzI1
NiIsDQoNCiJraWQiOiAiYXNkZmctcXdlcnR5LXp4Y3ZiIg0KfQ.ew0KDQrCoCJwdX
NoQWNjb3VudFJlY2VpcHQiOiAiTUNDLVNUTC0xMzQzMTNCRi01NTg1LTRFNzEtQUI
yNC1FQ0RCQzI4RjIzRjEiLA0KImlzc3VlckNhbGxCYWNrIjogImh0dHBzOi8vaXNz
dWVyY2FsbGJhY2sudXJsIiwNCiJjYWxsYmFja1JlcXVpcmVkIjogdHJ1ZSwNCiJjb
21wbGV0ZVdlYnNpdGVBY3RpdmF0aW9uIjogdHJ1ZSwNCiJhY2NvdW50SG9sZGVyRG
F0YVN1cHBsaWVkIjogdHJ1ZSwNCiJsb2NhbGUiOiAiZW5fVVMiDQoNCn0.ew0KImF
sZyI6ICJSUzI1NiIsDQoNCiJraWQiOiAiYXNkZmctcXdlcnR5LXp4Y3ZiIg0KfQ
```

JWS Overview

A JWS represents these logical values (each of which is defined in Section 2 of RFC7515):

- JOSE Header
- JWS Payload
- JWS Signature

JWS Compact Serialization Overview

In the JWS Compact Serialization, no JWS Unprotected Header is used. In this case, the JOSE Header and the JWS Protected Header are the same.

In the JWS Compact Serialization, a JWS is represented as the concatenation:

```
BASE64URL(UTF8(JWS Protected Header)) || '.' ||
BASE64URL(JWS Payload) || '.' ||
BASE64URL(JWS Signature)
```

Refer to Section 7.1 of RFC7515 for more information about the JWS Compact Serialization.

Examples

- URL

```
https://myTR.com/pushAccount?pushAccountData=ew0KImFsZyI6ICJSUzI1
NiIsDQoNCiJraWQiOiAiYXNkZmctcXdlcnR5LXp4Y3ZiIg0KfQ.ew0KDQrCoCJwdX
NoQWNjb3VudFJlY2VpcHQiOiAiTUNDLVNUTC0xMzQzMTNCRi01NTg1LTRFNzEtQUI
yNC1FQ0RCQzI4RjIzRjEiLA0KImlzc3VlckNhbGxCYWNrIjogImh0dHBzOi8vaXNz
dWVyY2FsbGJhY2sudXJsIiwNCiJjYWxsYmFja1JlcXVpcmVkIjogdHJ1ZSwNCiJjb
21wbGV0ZVdlYnNpdGVBY3RpdmF0aW9uIjogdHJ1ZSwNCiJhY2NvdW50SG9sZGVyRG
F0YVN1cHBsaWVkIjogdHJ1ZSwNCiJsb2NhbGUiOiAiZW5fVVMiDQoNCn0.ew0KImF
sZyI6ICJSUzI1NiIsDQoNCiJraWQiOiAiYXNkZmctcXdlcnR5LXp4Y3ZiIg0KfQ
```

- Three logical values separated by . delimiter:

```
ew0KImFsZyI6ICJSUzI1NiIsDQoNCiJraWQiOiAiYXNkZmctcXdlcnR5LXp4Y3ZiI
g0KfQ.ew0KDQrCoCJwdXNoQWNjb3VudFJlY2VpcHQiOiAiTUNDLVNUTC0xMzQzMTN
CRi01NTg1LTRFNzEtQUIyNC1FQ0RCQzI4RjIzRjEiLA0KImlzc3VlckNhbGxCYWNr
IjogImh0dHBzOi8vaXNzdWVyY2FsbGJhY2sudXJsIiwNCiJjYWxsYmFja1JlcXVpc
mVkIjogdHJ1ZSwNCiJjb21wbGV0ZVdlYnNpdGVBY3RpdmF0aW9uIjogdHJ1ZSwNCi
JhY2NvdW50SG9sZGVyRGF0YVN1cHBsaWVkIjogdHJ1ZSwNCiJsb2NhbGUiOiAiZW5
fVVMiDQoNCn0.ZEJqZnRKZVo0Q1ZQLW1COTJLMjd1aGJVSlUxcDFyX3dXMWdGV0ZP
RWpYaw
```

- Jose Header
  - BASE64URL(UTF8(JWS Protected Header)) :

    ```
    ew0KImFsZyI6ICJSUzI1NiIsDQoNCiJraWQiOiAiYXNkZmctcXdlcnR5LXp4Y3ZiIg0KfQ
    ```

  - JWS Protected Header

    ```
    {
    "alg": "RS256",
    "kid": "asdfg-qwerty-zxcvb"
    }
    ```

- JWS Payload
  - BASE64URL(JWS Payload):

```
ew0KDQrCoCJwdXNoQWNjb3VudFJlY2VpcHQiOiAiTUNDLVNUTC0xMzQ
zMTNCRi01NTg1LTRFNzEtQUIyNC1FQ0RCQzI4RjIzRjEiLA0KImlzc3VlckNhbGxC
YWNrIjogImh0dHBzOi8vaXNzdWVyY2FsbGJhY2sudXJsIiwNCiJjYWxsYmFja1Jlc
XVpcmVkIjogdHJ1ZSwNCiJjb21wbGV0ZVdlYnNpdGVBY3RpdmF0aW9uIjogdHJ1ZS
wNCiJhY2NvdW50SG9sZGVyRGF0YVN1cHBsaWVkIjogdHJ1ZSwNCiJsb2NhbGUiOiA
iZW5fVVMiDQoNCn0
```

- JWS Payload

```
{
"pushAccountReceipts": ["MCC-STL-134313BF-5585-4E71-AB24-ECDBC28F23F1", "MCC-
STL-37fa3300-828f-11eb-8dcd-0242ac130003"],
"callBackURL": "https://issuercallback.url",
"completeWebsiteActivation": true,
"accountHolderDataSupplied": true,
"locale": "en_US
}
```

- JWS Signature
  - BASE64URL(JWS Signature) :

```
ZEJqZnRKZVo0Q1ZQLW1COTJLMjd1aGJVSlUxcDFyX3dXMWdGV0ZPRWpYaw
```

  - JWS Signature:

```
dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

NOTE: If an issuer and token requestor both support signature functionality, the token requestor will receive these parameters as part of the JWS Payload in `pushAccountData`. However, if an issuer does not support signature functionality, the token requestor will receive data in previous or existing format (regardless of the whether the token requestor supports signature verification or not).

Download public key to verify signature

1. Use the `getAsset` API to download public key.
   - New type is introduced: `application/jwk+json`
   - Sample Request

```
https://api.mastercard.com/mdes/assets/mtf/1/0/asset/202106021718
13-MDES-token-connect-stage1
```

   - Sample Response

```
{
"mediaContents": [
{
"data": "ewogICJrdHkiOiAiUlNBIiwKICAieDV0I1MyNTYiOiAiSjE2dld1NGZq
SF9SR0E3d18zSDJOZXhIbWV6M2ZHVEtCRXZpU3JaMTdDNCIsCiAgImUiOiAiQVFBQ
iIsCiAgInVzZSI6ICJzaWciLAogICJraWQiOiAiMjAyMTA2MDIxNzE4MTMtTURFUy
10b2tlbi1jb25uZWN0LXN0YWdlMSIsCiAgIng1YyI6IFsKICAgICJNSUlGV0RDQ0E
wQ2dBd0lCQWdJSUR6TUxXbTR2ZDBJd0RRWUpLb1pJaHZjTkFRUxCUUF3Z2JneEV6
QVJCZ29Ka2lhSmtcL0lzWkFFWkZnTmpiMjB4R2pBWUJnb0praWFKa1wvSXNaQUVaR
mdwdFlYTjBaWEpqWVhKa01Rc3dDUVlEVlFRR0V3SkNSVEVkTUJzR0ExVUVDaE1VVF
dGemRHVnlRMkZ5WkNCWGIzSnNaSGRwWkdVeEd6QVpCZ05WQkFzVEVrTnZjbkJ2Y21
```

```
GMFpTQlRaV04xY21sMGVURThNRG9HQTFVRUF4TXpUV0Z6ZEdWeVEyRnlaQ0JVVTFR
Z1FYQndiR2xqWVhScGIyNGdTVzVtY21GemRISjFZM1IxY21VZ1UzVmlJRU5CSUVje
U1CNFhEVEl4TURZd01qSXlNVGd6TVZvWERUSTFNRFl3TVRJeU1UZ3pNVm93Z1oweE
lqQWdCZ05WQkFNTUdVMUVSVk10ZEc5clpXNHRZMjl1Ym1WamRDMXpkR0ZuWlRFeEV
UQVBCZ05WQkFzTUNFTnNhV1Z1ZERBeE1TNHdMQVlEVlFRS0RDVk5ZWE4wWlhKRFlY
SmtJRWx1ZEdWeWJtRjBhVzl1WVd3Z1NXNWpiM0p3YjNKaGRHVmtNUlF3RWdZRFZRU
UhEQXRUWVdsdWRDQk1iM1ZwY3pFUk1BOEdBMVVFQ0F3SVRXbHpjMjkxY21reEN6QU
pCZ05WQkFZVEFsVlRNSUlCSWpBTkJna3Foa2lHOXcwQkFRUZBQU9DQVE4QU1JSUJ
DZ0tDQVFFQWxaT0pKM2xsSmN3ZGNjSWlcL3NHNEZJTW1cLzRiekJ5RUVRQ0VoZDFz
bzEya3dOZkRLS01uN1FcL25YTGFOWkc4dExSeHdaUmxHMURtdVRjeFJKOUVTeDJiY
0lpcDM3bHhNRk9qNW1jU255SVpNUnFCRmZVdXlNbFNYazlTNm1sMHpwcE54XC9oeV
wvNUlVXC9pNmVBM1ZSVlZhYVFVRnlIN0dzK2JwU0p5Q1FPQWZ4em9yZ2w0N0ZMbjJ
Lc1JhXC9EOHVnVUwxMkRQUEhtNDAwY3JET2syXC9xUFhFUjR5Q0RVUDVNdTR0N3VW
dFZ6UTZYQlowZzZ6bEMrRkR0Y0V3cHhuMUlmdXJQellwZ3JpbkhkeEdWTk41aGhhT
1wvb1dqMmVnOHFBVDE0Z0NPQmtvaCtqNGd0eGo4TUttZmdLN05NZ2RwN0UxMEJQOU
pYMW9RVDN5enZmajlKVFlBUUlEQVFBQm8zOHdmVEFPQmdOVkhROEJBZjhFQkFNQ0F
LQXdEQVlEVlIwVEFRSFwvQkFJd0FEQWRCZ05WSFNVRUZqQVVCZ2dyQmdFRkJRY0RB
UVlJS3dZQkJRVUhBd0l3SFFZRFZSME9CQllFRkhTeUd4SytHbGpSUER0Uk9WZlJjU
3dRMFd4TE1COEdBMVVkSXdRWU1CYUFGSzlRQnY4VWdrTElWakcwNmJUYml2T3djcn
hTTUEwR0NTcUdTSWIzRFFFQkN3VUFBNElDQVFCNEs2a3JrUHRiUHBaSGYwblBLWWx
ZT2dObzAwZkVONEthY25xVlZ1ZnVPZXdOK0NwMFZ3dHBsZHIyWTRVb0YzSHFYMFN2
djdmYmxVODNETVwvQkFwWFpTbktdWVNQUM3YkJyT1dSWkhBbjB6eFcySDA2bnNyQ
3pVRXMyYnlXOElGRFwvNUJORUF5Y0FkNk4wXC9XKzdsdE9ZWWpcL2dLVmJBTklPbk
5PTkJ4eDVqOHlrditJSGpLaWZMXC9NOFpuYm85VmxvXC9XTHJEM2grR0NscWtkcnF
hM1FzME9xOTc4VUNZazl6andUb0FmQk5hbTBMa2M0UGE3eGorZlZmbng4UlpzXC82
RlN0T0h6bkdMZDV3WTFndGdqTGdSVTF1a2xsYkgyUkpsT3lBWk1ZamlLbVVLK3RyV
ERnVmtHeHNwbW9iOUZLQyttMmJKVmxBSk03Q3B0VmszM0trQkhzNVZYZXRjQTkrRX
VCcVBtM0NzSHRhaWhpbDBOcmxlMWxIUFwvSHcwcm1aVDlieGVlckd3WHhXaHh5R1N
zTkNmNVRQeWpETWxEbjhrUUNUUFBhZnRVeHpIc3FZZ21TZ1p3SVRHSGhcL3Blck1j
b01FNEdqTlwvblwvQVJaNG9iN0ZMckF5NFwvYzFWdHNHNEx4b2ZDRVUxekJ0b1FKU
TlGNWFMMFwvV0NMZlN4aDl3UVhCY2g4c3hVQUlWSDh5bHFoaTNqXC91aGo5cmo4NV
ZZbkJyWEgrSEV3RzBLdXUzQ1wvTkx4T21aaHVYT0IzdVgzVnIyaWFYZXllTHh5dmV
2ZFYxN2Jvekp3dWlyaVE4TGRXcTVpR0hZVjlyOWdIVFBReHJQSm5oMEdMbUQ1Z0c4
TUpYQnpPN1hya0ZxMGNpb1MyeW53SkRPcnFPM2RDelpcL3pPN2pqWWF2blFBPT0iC
iAgXSwKICAia2V5X29wcyI6IFsKICAgICJ2ZXJpZnkiCiAgXSwKICAiYWxnIjogIl
JTMjU2IiwKICAibiI6ICJsWk9KSjNsbEpjd2RjY0lpX3NHNEZJTW1fNGJ6QnlFRVF
DRWhkMXNvMTJrd05mREtLTW43UV9uWExhTlpHOHRMUnh3WlJsRzFEbXVUY3hSSjlF
U3gyYmNJaXAzN2x4TUZPajVtY1NueUlaTVJxQkZmVXV5TWxTWGs5UzZtbDB6cHBOe
F9oeV81SVVfaTZlQTNWUlZWYWFRVUZ5SDdHcy1icFNKeUNRT0FmeHpvcmdsNDdGTG
4yS3NSYV9EOHVnVUwxMkRQUEhtNDAwY3JET2syX3FQWEVSNHlDRFVQNU11NHQ3dVZ
0VnpRNlhCWjBnNnpsQy1GRHRjRXdweG4xSWZ1clB6WXBncmluSGR4R1ZOTjVoaGFP
X29XajJlZzhxQVQxNGdDT0Jrb2gtajRndHhqOE1LbWZnSzdOTWdkcDdFMTBCUDlKW
DFvUVQzeXp2Zmo5SlRZQVEiCn0",
"type": "application/jwk+json"
}
]
}
```


- Data parameter contains base 64 encoded public key in JWK format.
- Base 64 decoded value will be similar to:


```
{
"kty": "RSA",
"x5t#S256": "J16vWu4fjH_RGA7w_3H2NexHmez3fGTKBEviSrZ17C4",
"e": "AQAB",
"use": "sig",
"kid": "20210602171813-MDES-token-connect-stage1",
"x5c": [
"MIIFWDCCA0CgAwIBAgIIDzMLWm4vd0IwDQYJKoZIhvcNAQELBQAwgbgxEzARBgoJk
iaJk\/IsZAEZFgNjb20xGjAYBgoJkiaJk\/IsZAEZFgptYXN0ZXJjYXJkMQswCQYDV
QQGEwJCRTEdMBsGA1UEChMUTWFzdGVyQ2FyZCBXb3JsZHdpZGUxGzAZBgNVBAsTEkN
vcnBvcmF0ZSBTZWN1cml0eTE8MDoGA1UEAxMzTWFzdGVyQ2FyZCBUU1QgQXBwbGljY
XRpb24gSW5mcmFzdHJ1Y3R1cmUgU3ViIENBIEcyMB4XDTIxMDYwMjIyMTgzMVoXDTI
```

```
1MDYwMTIyMTgzMVowgZ0xIjAgBgNVBAMMGU1ERVMtdG9rZW4tY29ubmVjdC1zdGFnZ
TExETAPBgNVBAsMCENsaWVudDAxMS4wLAYDVQQKDCVNYXN0ZXJDYXJkIEludGVybmF
0aW9uYWwgSW5jb3Jwb3JhdGVkMRQwEgYDVQQHDAtTYWludCBMb3VpczERMA8GA1UEC
AwITWlzc291cmkxCzAJBgNVBAYTAlVTMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMII
BCgKCAQEAlZOJJ3llJcwdccIi\/sG4FIMm\/4bzByEEQCEhd1so12kwNfDKKMn7Q\/
nXLaNZG8tLRxwZRlG1DmuTcxRJ9ESx2bcIip37lxMFOj5mcSnyIZMRqBFfUuyMlSXk
9S6ml0zppNx\/hy\/5IU\/i6eA3VRVVaaQUFyH7Gs+bpSJyCQOAfxzorgl47FLn2Ks
Ra\/D8ugUL12DPPHm400crDOk2\/qPXER4yCDUP5Mu4t7uVtVzQ6XBZ0g6zlC+FDtc
Ewpxn1IfurPzYpgrinHdxGVNN5hhaO\/oWj2eg8qAT14gCOBkoh+j4gtxj8MKmfgK7
NMgdp7E10BP9JX1oQT3yzvfj9JTYAQIDAQABo38wfTAOBgNVHQ8BAf8EBAMCAKAwDA
YDVR0TAQH\/BAIwADAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwHQYDVR0
OBBYEFHSyGxK+GljRPDtROVfRcSwQ0WxLMB8GA1UdIwQYMBaAFK9QBv8UgkLIVjG06
bTbivOwcrxSMA0GCSqGSIb3DQEBCwUAA4ICAQB4K6krkPtbPpZHf0nPKYlYOgNo00f
EN4KacnqVVufuOewN+Cp0Vwtpldr2Y4UoF3HqX0Svv7fblU83DM\/BApXZSnKJueMA
C7bBrOWRZHAn0zxW2H06nsrCzUEs2byW8IFD\/5BNEAycAd6N0\/W+7ltOYYj\/gKV
bANIOnNONBxx5j8ykv+IHjKifL\/M8Znbo9Vlo\/WLrD3h+GClqkdrqa3Qs0Oq978U
CYk9zjwToAfBNam0Lkc4Pa7xj+fVfnx8RZs\/6FStOHznGLd5wY1gtgjLgRU1ukllb
H2RJlOyAZMYjiKmUK+trTDgVkGxspmob9FKC+m2bJVlAJM7CptVk33KkBHs5VXetcA
9+EuBqPm3CsHtaihil0Nrle1lHP\/Hw0rmZT9bxeerGwXxWhxyGSsNCf5TPyjDMlDn
8kQCTPPaftUxzHsqYgmSgZwITGHh\/perMcoME4GjN\/n\/ARZ4ob7FLrAy4\/c1Vt
sG4LxofCEU1zBtoQJQ9F5aL0\/WCLfSxh9wQXBch8sxUAIVH8ylqhi3j\/uhj9rj85
VYnBrXH+HEwG0Kuu3C\/NLxOmZhuXOB3uX3Vr2iaXeyeLxyvevdV17bozJwuiriQ8L
dWq5iGHYV9r9gHTPQxrPJnh0GLmD5gG8MJXBzO7XrkFq0cioS2ynwJDOrqO3dCzZ\/
zO7jjYavnQA=="
],
"key_ops": [
"verify"
],
"alg": "RS256",
"n": "lZOJJ3llJcwdccIi_sG4FIMm_4bzByEEQCEhd1so12kwNfDKKMn7Q_nXLaNZG
8tLRxwZRlG1DmuTcxRJ9ESx2bcIip37lxMFOj5mcSnyIZMRqBFfUuyMlSXk9S6ml0zp
pNx_hy_5IU_i6eA3VRVVaaQUFyH7Gs-bpSJyCQOAfxzorgl47FLn2KsRa_D8ugUL12D
PPHm400crDOk2_qPXER4yCDUP5Mu4t7uVtVzQ6XBZ0g6zlC-FDtcEwpxn1IfurPzYpg
rinHdxGVNN5hhaO_oWj2eg8qAT14gCOBkoh-j4gtxj8MKmfgK7NMgdp7E10BP9JX1oQ
T3yzvfj9JTYAQ"
}
```


NOTE: This is the same `getAsset` API that the Token Requestor uses to retrieve a card image; no additional onboarding is required. Refer to RFC 7517 for JWK detail.

2. Asset ids to retrieve keys:

Asset id	Environment	Expiry date
20210927091340-MDES-token-connect-mtf	MTF	Dec 22, 2024
20211006031131-MDES-token-connect-prod	Production	July 13, 2025

3. Token Requestor needs to download this key offline, store it, and use it to verify the signature to ensure that the key is stored with the correct key id.
4. Token Requestor must NOT use the `getAsset` API to retrieve the public key at the time of verify signature.
5. At a time only one key id will be active.
6. `Asset Id` and `kid` (present in JWS Protected Header) are both the same. This will help to retrieve the correct public key from your storage.

7. Renewal/Rotation:
   - Mastercard will notify all token requestors 60 days prior to existing key expiry.
   - Token requestor must download new keys prior to existing key expiring.
   - If Mastercard replaces an existing key with a new key at any point, then Mastercard will immediately inform all token requestors. The token requestors will need to download the new key as soon as possible and ensure it is ready to use.

Communication specifications

The issuer is required to provide the `pushAccountReceipt` parameter to the token requestor. The issuer may also provide a callback URI in the request.

This can be accomplished through several communication channels:

App to app	Communication on a mobile device (Android or iOS) between an issuer application and the token requestor application
App to web	Communication on a mobile device (Android or iOS) between an issuer application and a token requestor's website
Web to app	Communication on a mobile device (Android or iOS) between an issuer website and a token requestor application
Web to web	Communication on the same device (mobile or desktop) between an issuer website and a token requestor website

All information will be passed by query string, both from the issuer to the token requestor and from the token requestor back to the issuer.

Request query string parameters from issuer to token requestor

Parameter Name	Description
`pushAccountReceipts`	**Mandatory**. The `pushAccountReceipts` value obtained by the issuer from MDES, separated by commas. Each `pushAccountReceipt` is a string that consists of a 3-character prefix identifying the product associated with the funding account, concatenated with a universally unique identifier, in the form prefix-UUID. The prefix is needed by some token requestors in preparation of tokenization. The max length of a receipt is 64 characters. A receipt is valid for 15 minutes after its issuance by MDES. Prefix values are: **MCC** Mastercard Credit **DMC** Mastercard Debit **MSI** Maestro **PVL** Private Label Example: Issuer passes 3 `pushAccountReceipts` (for 3 different cards): `pushAccountReceipts=MCC-C307F0AE-298E-48EB-AA43-A7C40B32DDDE,MSI-1E8GTJ94-9D5T-96MO-WV36-56AZN95Y8DUL,DMC-9H2T37SH-RB3O-PI14-QQ9R-321UGR5ZI07A`

Parameter Name	Description

`callbackURL`	**Optional**.
	The URL encoded URL for the token requestor to use to pass control back to the issuer. This needs to be an absolute URL containing the scheme. If the issuer wishes to redirect to their banking app (iOS or Android), the URL must contain a custom URI scheme. Both iOS and Android support custom URI schemes that allow the token requestor's app or the browser to open an application.
	Example 1:
	Issuer Moon Bank expects a redirection to their website at http://www.moonbank.com/pushcallback:
	`callbackURL=https%3A%2F%2Fwww.moonbank.com%2Fpushcallback`
	Example 2:
	Issuer Moon Bank expects a redirection to their iOs or Android bank app at moonbank://pushcallback (custom URI scheme)
	`callbackURL=moonbank%3A%2F%2Fpushcallback`

`callbackRequired`	**Optional** (Default value=true) Boolean value. When callbackURL is supplied, indicates whether the token requestor must call the issuer callback URL after the push provisioning operation. Possible values are: **false** The token requestor may keep the consumer in their interface at the end of the provisioning, or send him/her back to the issuer interface. The consumer may be involved in this choice. **true** Boolean value used to drive the behavior of the token requestor if a digitization request receives a decision ofThe token requestor must return the methods. This field is not case sensitive. Possible values are: Example: `callbackRequired=false`

Parameter Name	Description
`completeIssuerAppActivation`	**Optional**. (Default value= true) **NOTE: This parameter is applicable only for Wallet Providers supporting activation via the issuer's banking application. Other token requestors can ignore it. The token requestor must return the consumer back to the issuer interface at the end of the provisioning.** Boolean value used to drive the behavior of the token requestor if a digitization request receives a decision of REQUIRE_ADDITIONAL_AUTHENTICATION and the cardholder chooses the issuer's mobile app Authentication Method from the list of supplied methods. This field is not case sensitive. Possible values are: **false** **Streamlined Activation**: The token requestor should not execute activation. The token requestor should return control to the issuer app via the callbackURL, after all tokenizations are completed, so that the issuer can perform activation **true** The token requestor should execute activation with the issuer app using the information supplied by MDES. Example: `completeIssuerAppActivation=false`
`completeWebsiteActivation`	**Optional**. (Default value= true) **NOTE: This parameter is applicable only for Wallet Providers supporting activation via the issuer's website. Other token requestors can ignore it.** **false** **REQUIRE_ADDITIONAL_AUTHENTICATION and the cardholder chooses the Streamlined Activation**: The token requestor should not execute activation. The token requestor should return control to the issuer's app via the `callbackURL`, after all tokenizations are completed, so that the issuer can perform activation. **true** The token requestor should execute activation with the issuer website using the information supplied by MDES. Example: `completeWebsiteActivation=false`

Parameter Name	Description
`accountHolderDataSupplied`	**Conditional**. (Default value= false) Boolean value used to indicate if account holder information has been supplied to MDES along with the funding account information. MDES will return the account holder information to the token requestor in certain scenarios. This field is not case sensitive. Possible values are: **false** Account holder data has not been passed to MDES. **true** Account holder data has been passed to MDES for at least one of the `pushAccountReceipts`.

`locale`	**Optional** (no default value)
	Consumer preferred locale (language and country).
	Format:
	Two letter ISO 639-1 language in lowercase, with a underscore (_), followed by two letter ISO 3166-1 country code in uppercase.
	Example 1:
	`locale=en_US`
	Example 2:
	`locale=fr_CA`

Using the parameters in the redirection URL, the issuer specifies the desired behavior after the push provisioning operation. issuer decides between three options:

Behaviour	callbackURL	callbackRequired
Consumer must return to the issuer interface	(Present)	true (or absent)
Consumer must stay in the token requestor interface	(Absent)	N/A
Consumer may either stay in the token requestor interface or return to the issuer interface (at token requestor/consumer's convenience)	(Present)	false

Example

- From Moon Bank app, User has selected 3 cards to push to My Wallet, hence Moon Bank has 3 `pushAccountReceipts` to send to My Wallet.
- MDES has indicated that token requestor My Wallet supports multiple pushed accounts, and expects Token Connect redirections at `https://www.mywallet.com/pushAccount`.
- Callback to the issuer's app is optional, the consumer may stay in the token requestor interface.
- Issuer Moon Bank expects Token Connect responses on their issuer app at callback URL `moonbank://pushcallback`.
- Issuer Moon Bank supports streamlined activation in their Moon Bank app.
- Issuer indicates that consumer is based in the USA and uses English language.

Hence, Moon Bank builds the redirection URL that will send the consumer to the token requestor's interface as follows:

```
https://www.mywallet.com/pushAccount?pushAccountReceipts=MCC-C307
F0AE-298E-48EB-AA43-A7C40B32DDDE,MSI-1E8GTJ94-9D5T-96MO-WV36-56AZ
N95Y8DUL,DMC-9H2T37SH-RB3O-PI14-QQ9R-321UGR5ZI07A&callbackURL=moo
nbank%3A%2F%2Fpushcallback&callbackRequired=false&completeIssuerA
ppActivation=false&completeWebsiteActivation=false&locale=en_US
```

NOTE: If unrecognized query string parameters are present in the URL, token requestors must ignore these parameters.

Upon receiving the `pushAccountReceipts` the token requestor will attempt to perform tokenizations by passing the `pushAccountReceipts` to MDES after identifying and authenticating the user, if needed. MDES will retrieve the funding account information associated with the `pushAccountReceipt` in lieu of it being passed directly by the token requestor.

Response query string parameters from token requestor to issuer

After completion of the tokenization, the token requestor may route the cardholder back to the issuer's website or app by using one of the redirection

methods with the provided callbackURL after appending the tokenization results in the query string.

Description	Parameter Name
The map of results of the tokenization attempts. The `pushAccountReceipt` values must be used as the key to the map and the tokenization decision as the value. In the case of an APPROVED or REQUIRE_ADDITIONAL_AUTHENTICATION result, the `tokenUniqueReference` must be appended to the result separated by a | character. **NOTE: The unsafe characters: ", "], and | must be URL-encoded in the query string, please refer to the example.** Possible values are: **APPROVED** — Any of the following: • The provisioning request has been approved. • The provisioning request required additional authentication of the cardholder, and the authentication has been performed successfully. **REQUIRE_ADDITIONAL_AUTHENTICATION** — The issuer requires additional authentication of the cardholder for this provisioning request, and the authentication hasn't been performed. **DECLINED** — The provisioning request has been declined. **CANCELLED** — The provisioning request has been cancelled by the Cardholder or token requestor. **ERROR** — The provisioning request has resulted in an error.	`results`

If the callbackURL supplied by the already contains one or more query string parameters introduced by a **?** character (for instance, a session identifier), the token requestor keeps these in the redirection URL, and further appends the tokenization results, introduced by an & instead of a ?

In the unlikely case that, by mistake, the wouldn't have supplied any pushAccountReceipt, no results are appended to the callbackURL. The token requestor simply redirects the consumer to the interface, using the callbackURL.

Example 1

Calling back the Android app – single card, no issuer-proprietary parameter:

- Issuer Moon Bank receives MDES Token Connect responses in Android app at `moonbank://pushcallback`.
- Issuer has supplied a unique `pushAccountReceipt` to the token requestor, with value MCC-C307F0AE-298E-48EB-AA43-A7C40B32DDDE. The token requestor has successfully tokenized this `pushAccountReceipt`, creating a token whose `tokenUniqueReference` is `DWSPMC000000000132d72d4fcb2f4136a0532d3093ff1a45`.

The resulting (raw) URI is:

```
moonbank://pushcallback?results[MCC-C307F0AE-298E-48EB-AA43-A7C40
B32DDDE]=APPROVED|DWSPMC000000000132d72d4fcb2f4136a0532d3093ff1a45
```

After URL-encoding of the special characters, the resulting URI called by the token requestor is:

```
moonbank://pushcallback?results%5BMCC-C307F0AE-298E-48EB-AA43A7C40
B32DDDE%5D=APPROVED%7CDWSPMC000000000132d72d4fcb2f4136a0532d3093ff1a45
```

Example 2

Calling back the web server – multiple cards, one issuer-proprietary parameter:

- To avoid an unnecessary consumer login at callback, Moon Bank receives MDES Token Connect responses on their web server at a dynamic address including a session identifier. Moon Bank has supplied this dynamic address to the token requestor as `callbackURL`. For the current session, the desired return URL is:
  ```
  https://www.moonbank.com/pushcallback?session_id=789456123
  ```
- Issuer has supplied 3 `pushAccountReceipts` to the token requestor. One has been tokenized successfully, one requires additional authentication, one has not been tokenized due to a manual cancellation by the consumer.

The resulting (raw) URI is:

```
https://www.moonbank.com/pushcallback?session_id=789456123&results
[MCC-C307F0AE-298E-48EB-AA43-A7C40B32DDDE]=APPROVED|DWSPMC0000000
00132d72d4fcb2f4136a0532d3093ff1a45&results[MSI-1E8GTJ94-9D5T-96M
O-WV36-56AZN95Y8DUL]=REQUIRE_ADDITIONAL_AUTHENTICATION|DWSPMC0000
0000032d72d4ffcb2f4136a0532d32d72d4fcb&results[DMC-9H2T37SH-RB3O-
PI14-QQ9R-321UGR5ZI07A]=CANCELLED
```

After URL-encoding of the special characters, the resulting URI called by the token requestor is:

```
https://www.moonbank.com/pushcallback?session_id=789456123&results
%5BMCCC307F0AE-298E-48EB-AA43-A7C40B32DDDE%5D=APPROVED%7CDWSPMC000
000000132d72d4fcb2f4136a0532d3093ff1a45&results%5BMSI-1E8GTJ94-9D5
T-96MO-WV36-56AZN95Y8DUL%5D=REQUIRE_ADDITIONAL_AUTHENTICATION%7CDW
```

```
SPMC00000000032d72d4ffcb2f4136a0532d32d72d4fcb&results%5BDMC-9H2T3
7SH-RB3OPI14-QQ9R-321UGR5ZI07A%5D=CANCELLED
```

Legal guidelines

By opting in to receive account holder data from issuers through Token Connect, a token requestor acknowledges and agrees that:

- Unless the token requestor, or its client, secures a valid legal basis for other uses of the data directly from the account holder, the account holder data may only be used only as follows:
  1. Name and Billing Address may be used for the sole purpose the technical enablement of the cards or financial accounts, being pushed by the issuer, within the user account and for the facilitation of cardholder transactions; and
  2. Email Address and Mobile Phone Number may be used for the sole purpose of creating a new user account, updating or identifying an existing user account.
- The account holder will be provided consumer terms and conditions and an appropriate privacy notice in connection with token requestor's product or service offering leveraging Token Connect;
- Personal account holder information may only be used, retained or otherwise processed only after:
  1. Privity is established with the account holder, such as a result of the successfully completed user registration process; and
  2. There is successful tokenization and completion of the push provisioning Token Connect process as defined by the program, this document and related specifications;
- Token requestor must delete the personal account holder information if the privity is not established with the account holder.
- Token requestor is responsible for providing, or ensuring that its clients provide, all applicable notices and obtaining any necessary consents in order to collect, store and use such account holder data and will comply with any terms and related privacy notice as between token requestor, or token requestor clients, and account holder; and
- Account holder data is provided "as is", without any guarantee of accuracy or completeness. It is the responsibility of the token requestor to validate this information with the consumer.

Chapter 5 Implementation process

Token requestors who want to support Token Connect should follow these steps.

Prepare

Here are guidelines to prepare.

- Read this document.
- Scope your project and find out which case you belong to by answering the following questions:
  - Which interfaces (Web browser - iOS App - Android App) will you use?
  - For iOS/Android apps: is the app supported? If yes, is it mandatory or optional?
  - For iOS/Android app: is device filtering needed?
  - Are you supporting Deferred Deep Linking? Note - Deferred Deep Linking is required when the device App is required (Cases 3-4-5).
  - What behavior do you want for desktop users? Will you reject them? If not, can you tokenize from a desktop browser, or will you need a desktop-to-mobile experience?
  - How many device apps (per mobile OS type) should support Token Connect, one or several?
  - Will you support multiple cards pushed simultaneously?
  - Will you support post tokenization authentication service through Remote Commerce Programs? (This is applicable only to the Merchant program.)
  - Will you support signature validation? (all token requestors must support by January 2023.)
  - For merchants and commerce platforms: will you accept some account holder data elements from the issuer (name, billing address, email address, mobile phone number)?
  - For wallets: does your wallet support token activation via the issuer's app or website? Note - If yes, Streamlined Activation is required.
- Using the User Experience guidelines provided in this document and the check-list supplied in Appendix A: Token requestor user interface design checklist, prepare the sequence of wireframes both for mobile and desktop users. Contact your Mastercard representative, indicate your will to implement Token Connect, and provide your wireframes. Mastercard will review your wireframes against the Design Checklist.
- Prepare the logo icon that you will onboard to MDES. This logo will be displayed to consumers in the issuer's app or website, when the issuer presents the list of potential destinations where consumers can push their accounts to. This is the logo that consumers associate with your (or your client's) brand or wallet. To ensure that token requestors are visually presented in a homogenous way in the issuer's app/website, token requestors logos must meet the following requirements:
  - The token requestor must provide the logo image in SVG and PNG formats. Both formats must correspond to the same image.

  - If the token requestor logo is not originally square, it must be centered on a white square and have a margin on each side (a minimum of five percent).
  - A PNG image must be 192 x 192 pixels. A SVG image must be square, with a reasonable size for an icon (height/width between 192x192 and 400x400 pixels).[1]

Develop

Here are guidelines to develop.

- With the help of the deep linking guidelines provided in this document, implement your deep linking solution in line with the desired user experience. You may implement it in-house, or rely on a third party solution, or have a hybrid solution.
- With the help of the specification piece provided in Communication specifications, implement the reception of issuers' requests with their incoming parameters, and map them with your deep linking solution.
- Per the tokenization guidelines provided in this document, and using the MDES API specification document (Digitization API/Digital Enablement API), implement the changes to use the pushAccountReceipt when calling MDES.
- Per the authentication guidelines provided in this document (and Authentication section) implement the change if you are supporting authentication (applicable only to merchants).
- Per the signature verification guidelines provided in this document, implement the changes to verify signature and retrieve the parameters.
- Implement changes to user activation.
- With the help of the specification piece provided in Communication specifications, implement the response to the issuer with the issuer callback URL and associated output parameters.

Onboard

Contact your Mastercard representative to open a project with our Customer Implementation Service (CIS).

Your CIS project manager will then contact you to set up the MDES Token Connect parameters associated to your token requestor ID (TRID):

- The logo image that consumers will be able to see from their issuer's app/website.

[1] Some browsers allow an easy file conversion from SVG to PNG format: open the SVG file in the browser, and save it as PNG. For a PNG to SVG conversion, you may use online tools such as http://www.aconvert.com (ensure that the original PNG has a sufficient resolution, ideally 400x400 pixels).

NOTE: Token requestor must be authorized to supply the logo to MDES. By supplying the logo, token requestor authorizes Mastercard, or certifies it has obtained its client's authorization for Mastercard, to transfer the logo to each issuer participating in Token Connect for the purpose of displaying eligible token requestors to consumers using the issuer's app or website and related consumer engagement efforts.

- A consumer-facing name for your entity, as it will appear to the consumer, next to the logo, in the issuer's app/website. This name may differ from the company legal name. Although the consumer-facing name may be up to 100 chars, it is recommended to limit it to 25 chars for an optimized display within the issuer's app/website.
- The URIs that the issuer's app/website can call out so as to put the consumer in direct relationship with the token requestor's interface:
  - URI for Android application interface
  - URI for iOS application interface
  - URI for Web browser interface

  See Deep linking guidelines in this document for more details on how to configure your URIs.
- Whether multiple cards (a maximum of five) can be pushed simultaneously or not.
- Whether authentication (authentication service) is supported or not, (this is applicable only to merchants and a merchant must not support multiple cards).
- Whether signature verification is supported or not (token requestor must support this functionality by January 2023).

All these data elements will be transmitted to issuers so that they can build the user experience in their app/website.

NOTE: The above is valid if you are an existing MDES token requestor upgrading to support Token Connect. If you are not yet onboarded to MDES, these operations will be integrated to your regular MDES onboarding.

Test

MDES Token Connect is an interoperable framework enabling MDES issuers to connect with MDES token requestors. To guarantee the interoperability of this framework, Mastercard needs to validate that each token requestor and each issuer correctly implements the Token Connect framework before granting access to the production environment.

In a first phase, you will have the opportunity to debug your implementation of MDES Token Connect in Mastercard Test Facility (MTF) environment. Your CIS project manager will supply you credentials to access the **MDES Token Connect Issuer Simulator** (a web-based tool). Using this tool, you will test various scenarios

on a self-service basis. You will also receive from Mastercard a list of applicable test cases to validate your implementation with the Issuer Simulator.

In a second phase, once you are confident that your implementation is correct, a Mastercard CIS Implementation Engineer will validate your implementation by executing (repeating) the same test cases in MTF with the Issuer Simulator and your implementation. If applicable, you may be required to provide to Mastercard a test user account, mobile devices, wearables, as necessary.

NOTE: This is valid if you are an existing MDES token requestor adding the support of MDES Token Connect. If you are not yet onboarded to MDES, these operations will be integrated to your regular MDES MTF testing.

NOTE: Current Token Connect Issuer testing tool does not have the capability to support the authentication or signature verification functionality. It is advised to test with an actual issuer who supports this functionality.

Launch

As soon as MTF testing is successfully completed, you will be enabled for Token Connect in Mastercard's production environment.

Using the same **MDES Token Connect - Issuer Simulator Tool**, you and the Mastercard CIS Implementation Engineer must validate your implementation in the production environment. You are now ready to process live Token Connect request from issuers.

You have a 30-day warranty after the launch of your project in production. Your CIS Implementation Manager will be available to assist you during this warranty period.

NOTE: Current Token Connect Issuer testing tool does not have the capability to support the authentication or signature verification functionality. It is advised to test with an actual issuer who supports this functionality.

Maintain

Ensure that you keep up-to-date with the changes to the MDES API you use to request digitization, and renew your API keys upon expiry.

As you may publish updates to your website or device applications from time to time, ensure that these changes don't affect negatively the push provisioning functionality. The **MDES Token Connect Issuer Simulator self-service tool** remains available for you. You should use it to test new versions of your website or app against potential regressions (in MTF and Production environments).

Your continuous compliance with the rules and specifications of the MDES Token Connect program is key to guarantee the best experience to all Issuers' cardholders, and to make this program successful. Mastercard reserves the right to enhance the Token Connect specifications, as well as control, on a random or regular basis, that your implementation is fully functional. Check for and implement any updates to this document, and collaborate to fix your implementation when you are asked to do so.

If you need assistance from Mastercard after the 30-day project warranty, contact digital.support@mastercard.com

Chapter 6 Use cases

With MDES Token Connect, after the consumer has selected the target token requestor and the cards to push, the consumer leaves the issuer's user interface and switches to the token requestor's user interface to complete the tokenization process. When complete, the consumer is taken back to the issuer's interface.

The personal situation of the cardholder will influence the user experience in the token requestor's interface. Typically, the experience will differ based on:

- Whether the consumer has the token requestor's app installed on their device or not.
- Whether the consumer has a compatible device or not (if applicable).

Besides, token requestors cover entities as diverse as commerce platforms, device wallets and IoT wallets, and each of them have their own specificities. Hence, during a Token Connect process, the user experience in the token requestors's environment must also be adapted to the token requestors.

Several token requestors parameters will influence the user experience in the token requestors's environment, mainly:

- Whether the presence of the device app (Android or iOS) is needed or not.
- Whether device filtering is needed or not: device-based wallets may need to advise consumers that their device model (e.g. mobile phone or paired IoT device) is not eligible for digitization. In this case, push provisioning must be cancelled, and the consumer must be taken back to the issuer's environment.
- Whether the token requestors supports push provisioning through a unique device app for each operating system (Android, iOS), or through multiple device apps. For instance, for IoT devices, the companion app that pairs the phone with the iOT device may not be the same for all IoT device models of the token requestors.

Most token requestors will fit into one of the five following cases:

Case	Presence of device app (Android, iOS)	Device filtering needed	Number of device apps supporting Token Connect (per OS)	Example token requestor
Case 1	Not supported	No	None	Server-based wallets, commerce platforms, merchants
Case 2	Optional	No	Only one	Server-based wallets, commerce platforms, merchants
Case 3	Mandatory	No	Only one	Device-based wallets supporting online payments (no NFC capability needed)
Case 4	Mandatory	Yes	Only one	Device-based wallets needing NFC capability. IoT wallets using a unique companion app to pair all their IoT device models
Case 5	Mandatory	Yes	Several	IoT wallets using a different companion app to pair each IoT device model

If the issuer requires a callback to their user interface, the following rules apply:

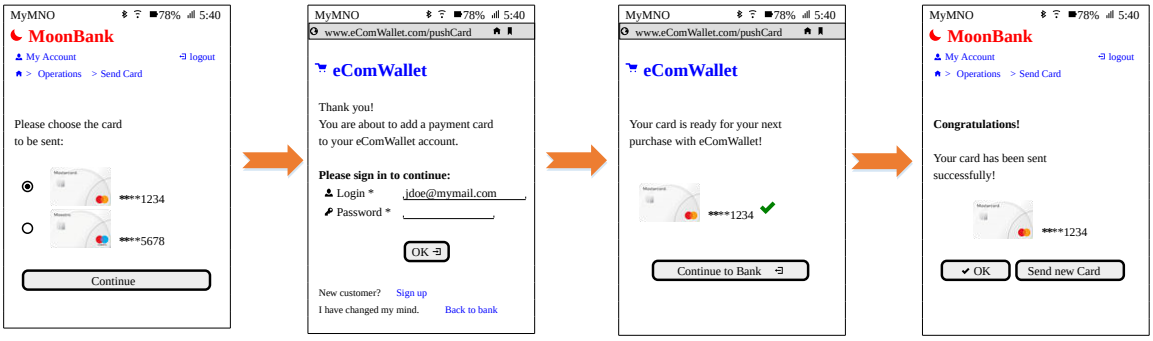
- **The consumer must be sent back to the issuer's interface** after tokenization is complete (nominal path). The same applies if the tokenization fails, is declined, or is cancelled by the consumer or by the token requestor while they are in the token requestor's interface. Refer to Handover to issuer and Display provisioning results.
- The issuer delegates a well-defined task to your app/web interface: complete card/account tokenization. **Token requestor must fulfill the issuer's request, and nothing else. Be straightforward in your interactions with the consumer** to guide them to the desired goal (successful tokenization), and **limit the interactions to the very minimum to achieve this goal**. Token requestor may conduct necessary consumer interactions related to customer account management (including device app download, sign up, sign in), as long as eventually the consumer is returned to the issuer interface. Other activities unrelated to tokenization, such as but not limited to, the purchase of goods or services, are not permitted during push provisioning.

These rules and principles are not applicable when the issuer does not supply a callback URL or indicates that the callback is optional.

Case 1 – No app

In this first case, the token requestor does not propose any device app to consumers, as only the web browser interface is available. This may be the choice of some server-based wallets, commerce platforms and merchants. The consumer is hence taken to the token requestor's site during push provisioning.

Figure 6: Token requestor - no app



Case 2 – Optional app

In this second case, the token requestor's device app is not necessary for tokenization as it is typically the case for server-based wallets, commerce platforms and merchants.

- If the token requestor's app is already present on the device, the consumer must be driven directly to the app, as it generally offers an optimized experience on mobile devices.
- If the token requestor's app is not present on the device, the consumer is taken to the token requestor's website through the device browser to complete the digitization. The consumer **should not** be invited to download the mobile app, as this would divert them from the original goal of provisioning the card.

Figure 7: Token requestor - optional app

Case 3 – Mandatory app, no device filtering

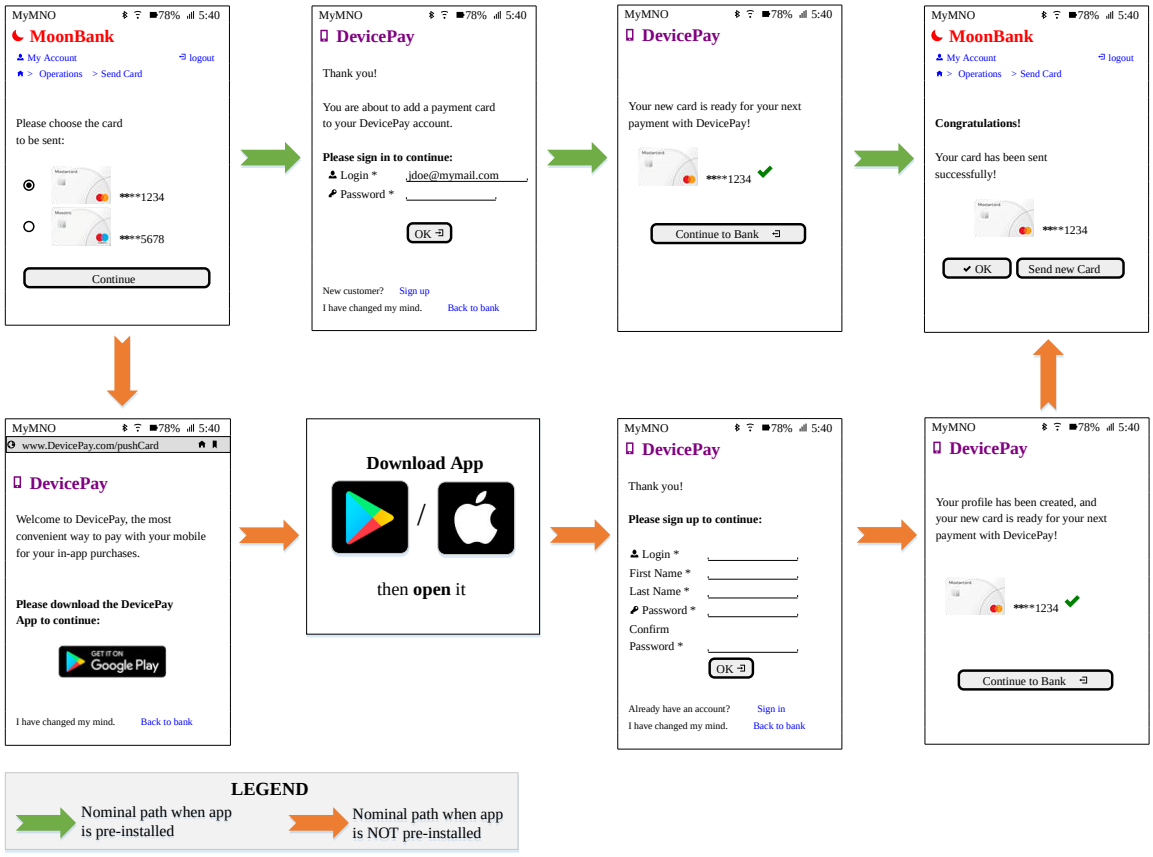
In this third case, the device app is needed for digitization, where all mobile devices are eligible, and the same app applies to all devices using the same operating

system. This typically applies to device-based wallets supporting app-to-app online payments, where NFC capability is optional or not supported.

- If the token requestor's app is already present on the device, the consumer must be driven directly to the app.
- If the token requestor's app is not yet present on the device, the consumer is taken to the token requestor's website through the device browser to download the app. Once downloaded and opened, the app should prompt the consumer **directly** to complete the ongoing card digitization (deferred deep linking).

  NOTE: The consumer may need to sign up.

Figure 8: Token requestor - mandatory app and no device filter



Case 4 – Mandatory app, device filtering, single app

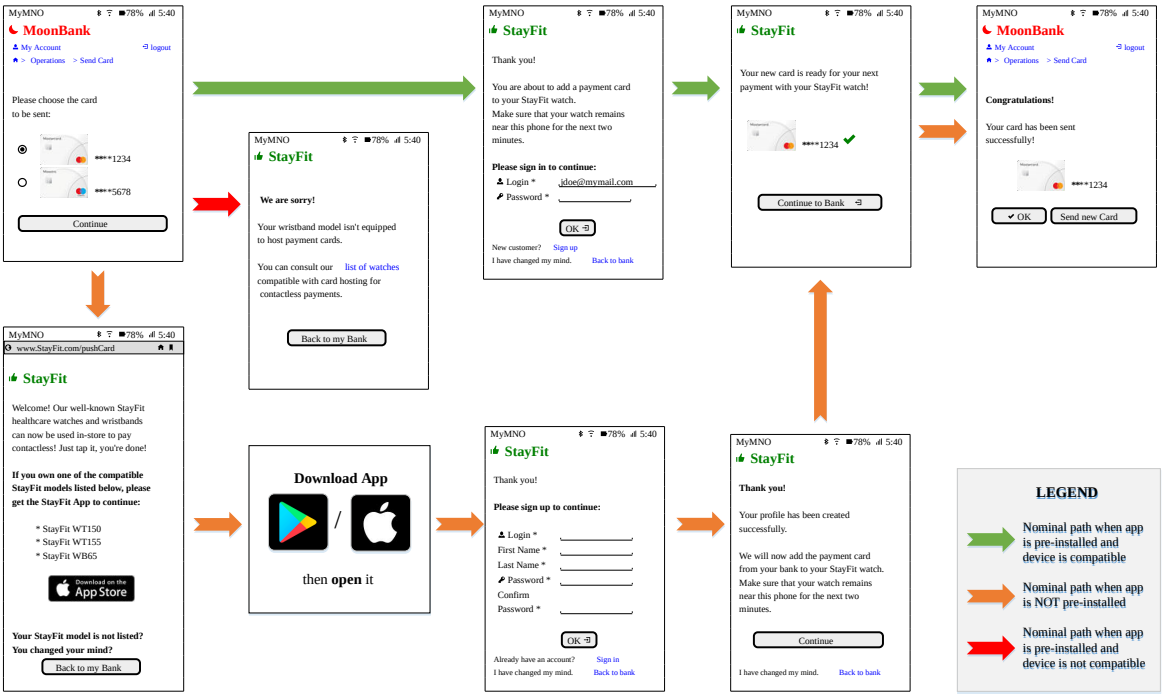
In this fourth case, the device app is needed for the digitization as not all devices are eligible, and the same app applies to all devices with the same operating system. This typically applies to device-based wallets requiring NFC capability as a result of which non-NFC devices must be filtered out, or IoT wallets using a unique

companion app to pair all their IoT device models (including models not supporting payment).

- If the token requestor's app is already present on the device, the consumer must be driven directly to the app. If the mobile or IoT device managed by the app is not compatible with the digitization of the card, the app must **clearly** notify this incompatibility and return the consumer to the issuer's interface.
- If the token requestor's app is not yet present on the device, the consumer is taken to the token requestor's website through the device browser to download the app. The list of compatible devices must be **clearly** featured, and the consumer must have an **easy** way to return to the issuer's interface if their device is not compatible. Once downloaded and opened, the app should prompt the consumer **directly** to complete the ongoing card digitization (deferred deep linking).

  NOTE: The consumer may need to sign up.

Figure 9: Token requestor - mandatory app with device filtering and single app



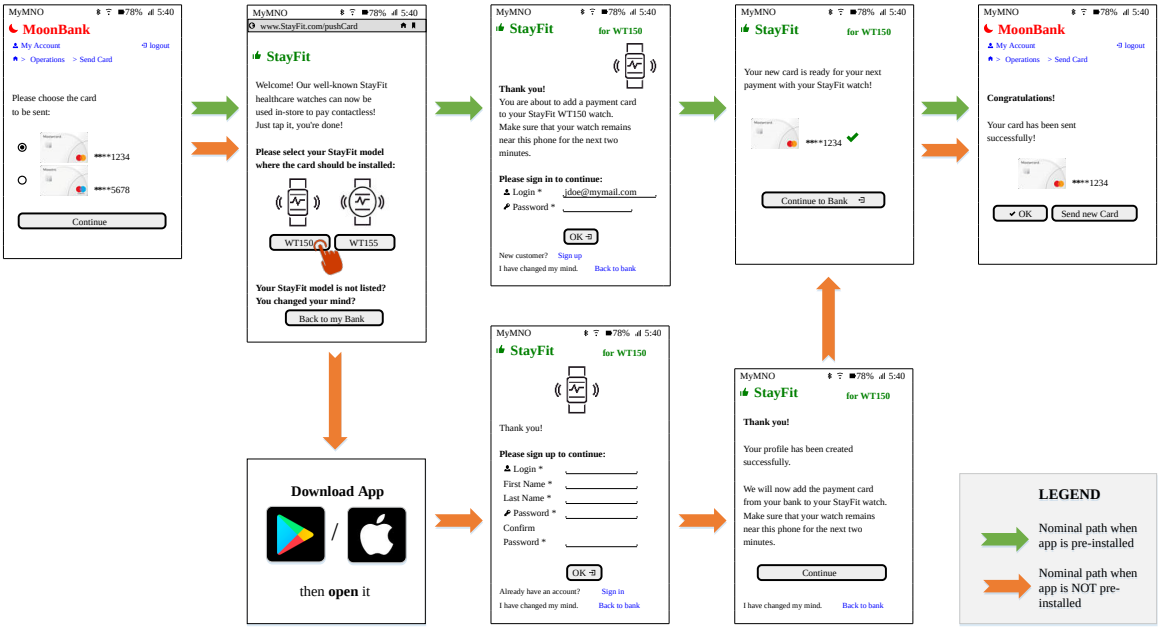
Case 5 - Mandatory app, device filtering, multiple apps

In this fifth case, the device app is needed for the digitization as not all devices are eligible, and the token requestor manages multiple apps per operating system. This typically applies to IoT wallets using a different companion app to pair each IoT device model.

- Regardless of the companion apps potentially installed on the mobile, the consumer is initially taken to the token requestor's website. The list of compatible devices must be **clearly** featured, and the consumer must have an **easy** way to return to the issuer interface if their device is not compatible.
- When the consumer selects a compatible model on the website, if the companion app of this device is already installed, the consumer must be driven directly to the app to complete the tokenization.
- When the consumer selects a compatible model on the website, if the companion app of this device is not installed, the consumer is driven to the app store. Once downloaded and opened, the app should prompt the consumer **directly** to complete the ongoing card digitization (deferred deep linking).

  NOTE: The consumer may need to sign up.

Figure 10: Token Requestor - mandatory app with device filtering and multiple apps



Desktop-to-mobile variant (Cases 3, 4 and 5)

For device-based wallets where a mobile app is needed, when the consumer initiates their journey from a desktop computer, the wallet provider has two possibilities.

1. Present a filtering landing page explaining that this wallet requires a mobile device, and asking them to restart their journey in their bank's environment, from their mobile device. Besides, the consumer must have an **easy** way to return to the issuer's interface (**Back to my bank** button).

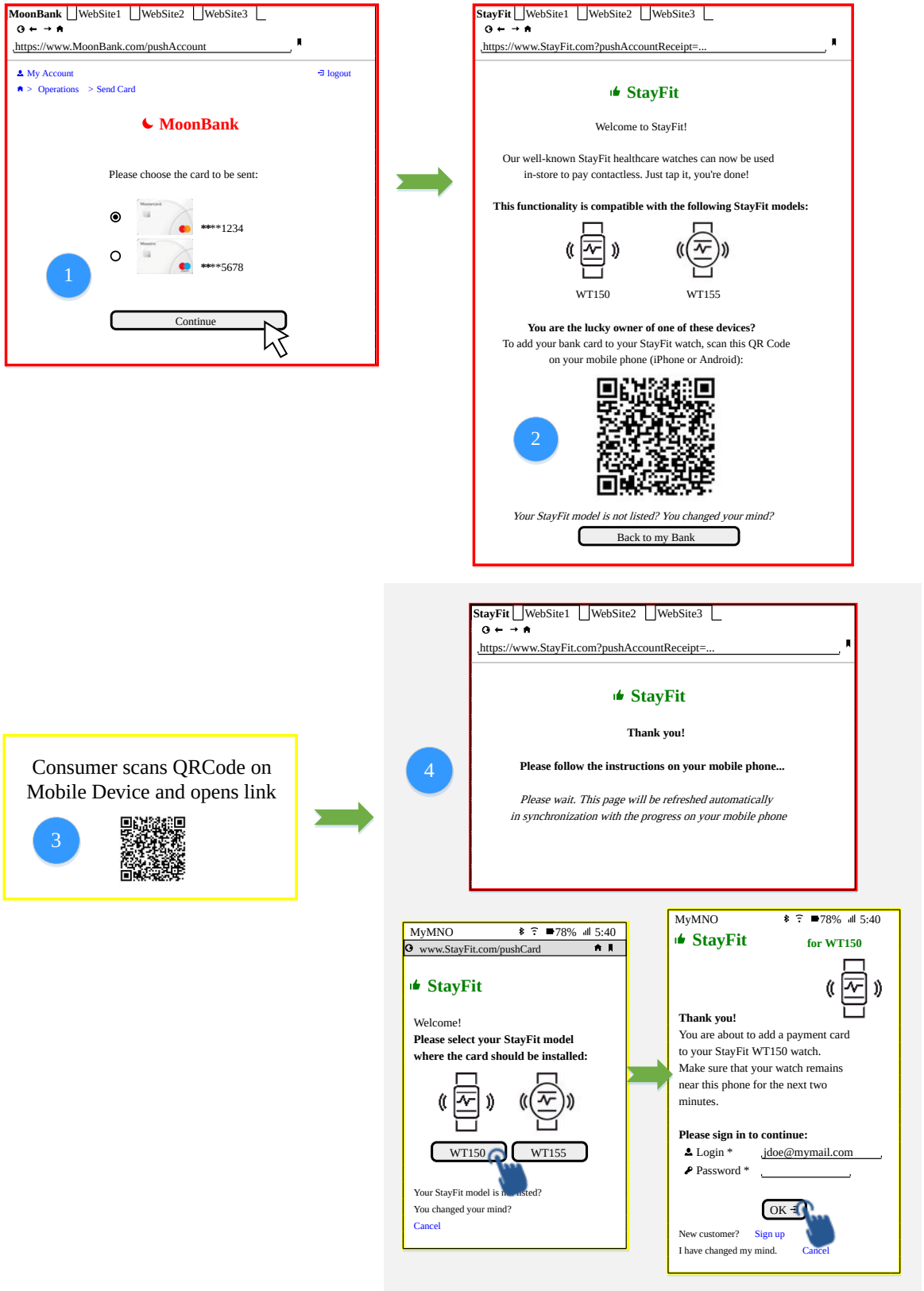
2. Invite the consumer to go on their digitization journey on their mobile device. The token requestor creates a parallel user experience on the mobile device of the consumer. Typical methods for pairing a desktop computer to a mobile include scanning a QR code, or sending a push notification or a SMS text message to a mobile device whose number has been supplied by the consumer.

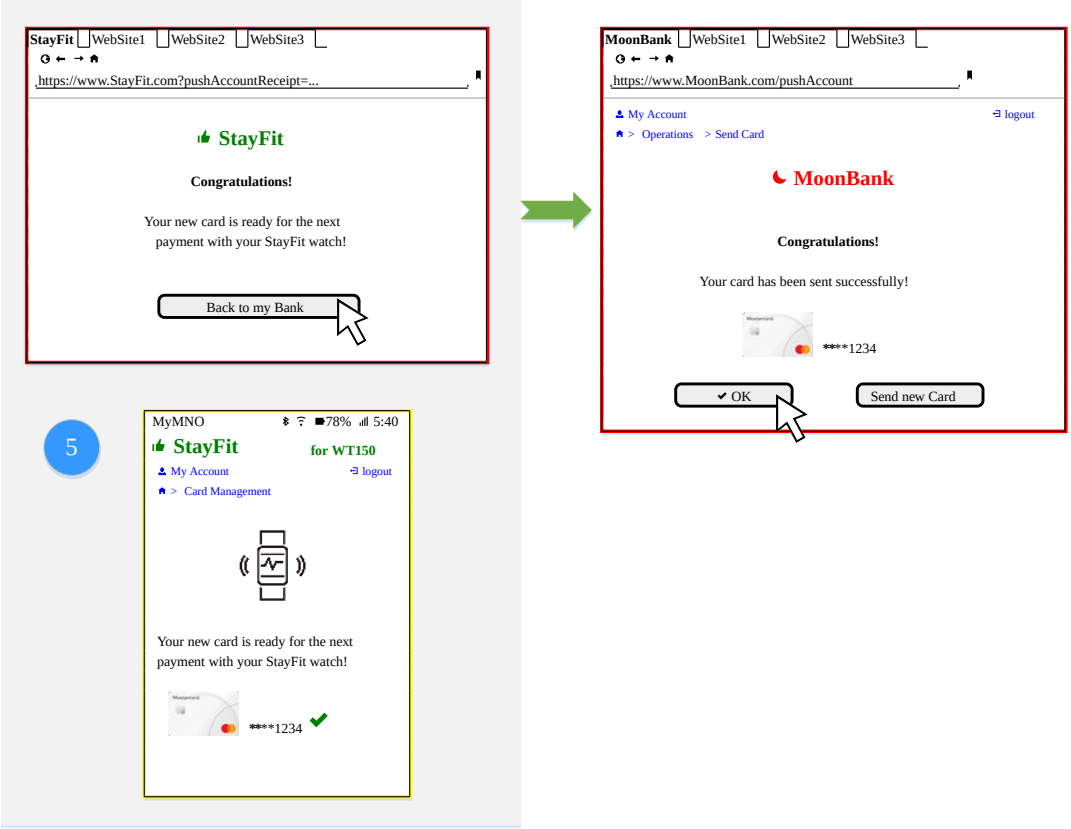
Wallet providers choosing to implement the second option (a desktop-to-mobile experience) must take the following points into consideration:

- A desktop-to-mobile experience brings significantly more complexity, both to your project and to the consumer experience. If you do not legitimately need it (that is, if you are not a device-based wallet provider), it would be best not to implement it.
- The pairing operation between desktop and mobile always carries an additional risk of fraud. For instance, a typo in the mobile number entered by the user, or the computer browser left unattended while displaying a pairing QR code are instances where a fraudster can add their victim's card to their own wallet. For a desktop-to-mobile journey, the wallet provider must systematically recommend additional cardholder authentication when initiating digitization, with reason `Low device score`.
- The callback URL supplied by the issuer when toggling to your web server is valid on the desktop computer. Do not assume that it will be valid on the mobile device as well. At the end of the digitization journey, the consumer must be returned to the issuer's environment in the desktop computer where they started. Hence, it means that you will have to manage two parallel consumer experience threads, one on the computer and another one on the mobile, and the computer thread must be informed of the progress of the mobile device thread (in progress, completed, failed, declined, cancelled…)

The figures in the following pages represent a desktop-to-mobile example flow, with the two parallel synchronized threads on the desktop and on the mobile. For this example, the token requestor belongs to Case 5 (mandatory app, device filtering, multiple apps), and it is assumed that the companion app is already present on the mobile device. The numbers in the blue circles represent the chronological order. Desktop browser views are framed in red, while mobile device views are framed in yellow.

Figure 11: Token requestor - mandatory app with device filtering and desktop to mobile





Appendix A Token requestor user interface design checklist

Token requestors must ensure that they respect the following checklist when creating wireframes for their website and applications for the push provisioning operation.

Any deviation from this checklist must be duly justified and documented to Mastercard:

1. Before the digitization, if the issuer has supplied a callback URL, the consumer shall have the possibility to change their mind and return to the bank website/app. Typically, a discrete *Changed your mind? Back to bank* link should be provided.
2. After the digitization, and regardless of the digitization outcome, if the issuer requires a callback to their user interface, the consumer shall be led back easily to the bank website/app. Typically, an obvious *Return to bank* button should be presented, with no possibility to escape from it.
3. If the issuer requires a callback to their user interface, the interaction between the consumer and the token requestor must be limited to the provisioning of the payment details being pushed. Any other activity unrelated to said digitization, such as, but not limited to, the purchase of goods or services, is not permitted. Account management activity (for instance, sign in, sign up, contact details update, mobile app download) can be conducted, as long as eventually the consumer is easily led back to the bank website/app (see item (2) above).
4. For Case 3/4/5 token requestors (mandatory mobile app), the token requestor must support deferred deep linking: if the app isn't yet present on the mobile device, the consumer must be able to easily download the app, open it, and complete the digitization of the pushed payment details, before being sent back to the bank.
5. For Case 4/5 wallets (device filtering), if the issuer requires a callback to their interface and the consumer device is not eligible, the consumer shall be led back easily to the bank website/app – typically with an obvious *Return to bank* button, with limited or no possibility to escape from it.
6. For wallets supporting a desktop-to-mobile path, upon completion of the digitization, the redirection to the issuer must happen on the desktop computer from which the consumer has triggered the operation.
7. The token requestor shouldn't ask the consumer to enter the CVC2.
8. If the token requestor accepts account holder data from the issuer, they must respect the privacy and legal obligations as outlined in Legal guidelines.
9. For wallets, if the issuer requires additional authentication, the wallet must complete token activation through standard MDES yellow path process.
10. Wallets that support token activation via the issuer's app or website must support streamlined activation as explained in Streamlined activation via issuer's application/website.
11. Token requestors must respect the indications from the table in Display provisioning results with regards to the tokenization results to send to the issuer
12. Token requestors must use messages to inform the cardholder with a meaning similar to those featured in the table from Display provisioning results.

13. Recommendation: Upon successful tokenization, the token requestor should present a screen which gives the cardholder an option to select a pushed card as the default payment method.

Notices

Following are policies pertaining to proprietary rights, trademarks, translations, and details about the availability of additional information online.

Proprietary Rights

The information contained in this document is proprietary and confidential to Mastercard International Incorporated, one or more of its affiliated entities (collectively "Mastercard"), or both.

This material may not be duplicated, published, or disclosed, in whole or in part, without the prior written permission of Mastercard.

Trademarks

Trademark notices and symbols used in this document reflect the registration status of Mastercard trademarks in the United States. Consult with the Global Customer Service team or the Mastercard Law Department for the registration status of particular product, program, or service names outside the United States.

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

Disclaimer

Mastercard makes no representations or warranties of any kind, express or implied, with respect to the contents of this document. Without limitation, Mastercard specifically disclaims all representations and warranties with respect to this document and any intellectual property rights subsisting therein or any part thereof, including but not limited to any and all implied warranties of title, non-infringement, or suitability for any purpose (whether or not Mastercard has been advised, has reason to know, or is otherwise in fact aware of any information) or achievement of any particular result.

Translation

A translation of any Mastercard manual, bulletin, release, or other Mastercard document into a language other than English is intended solely as a convenience to Mastercard customers. Mastercard provides any translated document to its customers "AS IS" and makes no representations or warranties of any kind with respect to the translated document, including, but not limited to, its accuracy or reliability. In no event shall Mastercard be liable for any damages resulting from reliance on any translated document. The English version of any Mastercard document will take precedence over any translated version in any legal proceeding.

Information Available Online

Mastercard provides details about the standards used for this document, including times expressed, language use, and contact information, on the Technical Resource Center (TRC). Go to the Rules collection of the References section for centralized information.