UNIFIED PAYMENTS INTERFACE

COMMON URL SPECIFICATIONS FOR DEEP LINKING AND PROXIMITY INTEGRATION

UPI Linking Specifications Version 1.6 (Draft)

Contents

1 Introduction

The Unified Payment Interface allows payments to be initiated by the payer, or by the payee. In the basic payee initiated flows, the payment request is routed by the initiating application through the NPCI switch to the payer for approval. However, in certain instances, where it is possible to connect with the payer immediately, it is preferred that the payee sends a payment request to the payer, who can then initiate the payment request with his credentials.

This leads to a significantly smoother payment experience. Some examples of these include in-app payments – where the merchant app, may send the request to the PSP app on the same device, instead of a collect request via the PSP network. Another example may be for proximity payments, where the payer and payee are using different devices, but are close enough for the information to be transmitted locally.

This document provides the technical specifications for developers to enable inter application payment requests.

1.1 Usage Examples

Example 1: Seamless in-app payment within the same mobile of the user.

- Ashok is a student and uses a video application (MyStar) that allows buying on-demand movie on his Android phone.
- He banks with DiBank (PSP in this case) and uses their mobile application for Android that has implemented UPI features.
- In MyStar app, Ashok wants to watch a movie for Rs.25.
- MyStar application creates the UPI payment link as per this spec and launches the Android intent with all necessary parameters populated in the URL.
- Since DiBank PSP app is registered to listen to UPI link/intent, it starts the app and takes Ashok straight to pay screen with all values pre-populated from the link/intent.
- Ashok verifies the info on screen and click pay to complete the payment.

Example 2: Proximity payment at a merchant using QR code.

- Mary uses her bank provided UPI application to make payments to nearby grocery store.
- After the purchase, grocery store PoS application generates a dynamic QR code containing the UPI link (as per this spec) with the payment details.
- Mary opens the UPI application on her mobile and scans the QR code on the PoS device or on the bill printed by the PoS.
- UPI application takes her straight to pay screen with all values pre-populated from the link/intent.
- She verifies the info on screen and click pay to complete the payment.
- Both merchant and she gets confirmation instantly.

Note that a real small one person shop could simply print a static QR code containing the payee address and name without having software to generate dynamic QR code with other information such as bill number, amount, etc. In the case of static QR code, customer, after scanning, should enter the amount and then make the payment.

Example 3: DTH payment from home.

- Nadeem subscribes to DTH in his house and wants to make a payment for on demand subscription.
- Nadeem selects the channel and clicks “buy now”.
- DTH shows the details along with a QR code for UPI payment.
- Nadeem opens his UPI application on his mobile and scans the QR code on the TV screen.
- UPI application takes him straight to pay screen with all values pre-populated from the QR code which contained the standard UPI link.
- He verifies the info on screen and click pay to complete the payment.
- He gets a confirmation on his mobile and the TV channel is automatically turned on for him to view.

1.2 Link Specification and Parameters

UPI Deep linking URL spec must be as follows. All PSP applications must mandatorily implement listening to "UPI" links within their mobile applications for QR, intent, NFC, BLE, UHF etc.

upi://pay?parm-name=param-value&param-name=pram-value&...

Where param-name can be any of the valid parameters (based on mandatory vs optional) listed in below table. M-Mandatory, C-Conditional, O-Optional

Parameter name	Data type	Static mode Tags	Dynamic mode Tags	Mapped to UPI API field	Description
pa	String	M	M	Payee-->addr	Payee VPA
pn	String	M	M	Payee-->name	Payee name
mc	String	O	O	Payee-->mcc	Payee merchant code If present then needs to be passed as it is.
tid	String	O	O	Txn -->id	This must be PSP generated id when present. In the case of Merchant payments, merchant may acquire the txn id from his PSP. If present then needs to be passed as it is.
tr	String	O	C	Txn-->refId	Transaction reference ID. This could be order number, subscription number, Bill ID, booking ID, insurance renewal reference, etc. This field is Mandatory for Merchant transactions and dynamic URL generation.
tn	String	O	O	Txn-->note	Transaction note providing a short description of the transaction.
am	String	O	M	Payee--> Amount-->value	Transaction amount in decimal format. If 'am' is not present then field is editable.
mam	String	O	C	Txn -->Rules --> MINAMOUNT	Minimum amount to be paid if different from transaction amount.
cu	String	O	O	Payee--> Amount-->curr	Currency code. Currently ONLY "INR" is the supported value.

url	String	O	O	Txn→refUrl	This should be a URL when clicked provides customer with further transaction details like complete bill details, bill copy, order copy, ticket details, etc. This can also be used to deliver digital goods such as mp3 files etc. after payment. This URL, when used, MUST BE related to the particular transaction and MUST NOT be used to send unsolicited information that are not relevant to the transaction. url should initiate with http or https.
mode	String (2 digit)	M	M	Txn → initiationMode	00=Default txn 01=QR Code 02=Secure QR Code 04=Intent 05=Secure Intent 06=NFC 07=BLE (Bluetooth) 08=UHF(Ultra High Frequency) 15=SEBI 16,17,18 = future use
sign	String	M	M	-	Base 64 encoded Digital signature needs to be passed in this tag
orgid	String (6 digit)	M	M	-	If the transaction is initiated by any PSP app then the respective orgID needs to be passed. For merchant initiated/created intent/QR '000000' will be used
mid	String (20 digit)	O	O	Payee→ merchant → mid	Merchant id (max 20) shall be passed in this tag
msid	String (20 digit)	O	O	Payee→ merchant → sid	Store id (max 20) shall be passed in this tag
mtid	String (20 digit)	O	O	Payee→ merchant → tid	Terminal id (max 20) shall be passed in this tag
Query	String 'JSON' (max 99 digits)	O	O	(future use)	This is for future use. We can add multiple fields basis requirement.

Developers who are developing merchant applications, mobile apps wanting to initiate UPI payment, should form the URL within their application and then do either of the following:

1. If the application and the PSP UPI application is within the same mobile, then do a deep linking using the URL.
2. Create a QR code within the application and allow customers to scan it and invoke their UPI application.
3. Use alternate transfer protocol (such as BLE, Wi-Fi Direct, NFC, UHF, etc.) to transfer the URL data to customer mobile on which is gets deep linked to their PSP application.
4. Create the URL and allow standard "share" allowing a UPI payment intent to be sent via chat or email. Receiver will click on the link to then invoke their PSP application.

5. While reading a QR, intent, NFC, BLE, UHF etc. all parameters must be read and passed to online message

6. If any tag is not present it can be dropped or passed as null or Null value.

Using a standard data format and URL scheme allows the actual protocol of data transfer to be separated out and thus allowing any transfer protocol to be used to transfer this from one device to another.

1.3 Signature

Signing of intent/QR/NFC/BLE etc. (referred to as intent only in the below section) can be broadly segregated into merchant initiated & PSP app initiated intents. The signing method for both are similar, however verification method for both of them varies.

Merchant can initiate intent from his mobile application, generate signed QR, broadcast signed NFC, BLE etc. from his terminal, POS, exit sensors. All the mentioned protocol for merchant initiated method follow identical process for signing and verification.

Merchant initiated:

1. *Key generation:* Merchant or the acquiring bank on behalf of merchant need to generate a key pair (public and private key). If Acquring bank has generated the key pair then private key can either be shared with merchant for intent generation or can be integrated in SDK directly via API (local storing of key is not recommended). Merchant and member banks shall also add provision for update of key pairs.
2. *Key Upload:* The merchant needs to share this public key with its acquiring bank and acquring bank will upload it's merchant public key on UPI with **Manage VAE** API.

If acquiring bank has generated the key for its merchant then it can directly upload on UPI.

3. *Signing of intent:* The merchant needs to sign the intent with its private key using SHA256 with RSA512 algorithem. The entire content of the string other than the tag "&sign=" need to be pass into the encryption function. E.g. if the intent is:

   'upi://pay?pa=bivek@npci&pn=bivek%20rath&mc=9999&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com&mode=05&orgid=000000&mid=1234&msid=3432&mtid=1212'

   The entire string is passed into encryption function i.e.

   SHA256withRSA512(*upi://pay?pa=bivek@npci&pn=bivek%20rath&mc=9999&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com&mode=05&&orgid=000000&mid=1234&msid=3432&mtid=1212*). Base 64 encoding shall be done to the output.

   The output:
   gynybu6K6ozUrHSySDhBK6rfiRE+VBMLnsxs3d7B/ddeGu43M6sxaY33ZVE4Utc4E1kbCcjgYhfJCUeYLdFp/UKckByYang6C99L337zp4/2mfYyEVg2E+6+G9Y3+RaGYA2iz9cQU43+O0esOuDcc5crmQOLoD+X22D21Dl5IKI=

   This output needs to be appended to the intent:

   *upi://pay?pa=bivek@npci&pn=bivek%20rath&mc=9999&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com&mode=05&orgid=000000&mid=1234&msid=3432&mtid=1212&**sign=gynybu6K6ozUrHSySDhBK6rfiRE+VBMLnsxs3d7B/ddeGu43M6sxaY33ZVE4Utc4E1kbCcjgYhfJCUeYLdFp/UKckByYang6C99L337zp4/2mfYyEVg2E+6+G9Y3+RaGYA2iz9cQU43+O0esOuDcc5crmQOLoD+X22D21Dl5IKI=***

   As the above example is an mobile intent sample initiated by merchant ('bivek@npci' with mcc '9999') hence mode is '**05**' & orgid is '**000000**'.
   *(*sign shall be last tag of the intent)*

4. *Key Download:* Payer PSP server needs to download all merchant keys and cache it on their server. This cache needs to be updated daily by the PSP server (recommended 6 AM). Each public key downloaded must be mapped with the merchant UPI ID for which the key was uploaded.
5. *Searching the key:* Orgid & UPI ID of payee need to be extracted first. As the orgid id '000000', the public key needs to be looked in list VAE cache with UPI ID as search parameter. This key will be used for verifying the signature.
6. *Verifying the intent:* The signature part is extracted and base 64 decoded. The entire intent string (excluding the signed part), signature (value within

“&sign=”) and the public key of merchant is passed into verify function for verification.

Verify((upi://pay?pa=bivek@npci&pn=bivek%20rath&mc=9999&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com&mode=05&orgid=000000&mid=1234&msid=3432&mtid=1212)
,(gynybu6K6ozUrHSySDhBK6rfiRE+VBMLnsxs3d7B/ddeGu43M6sxaY33ZVE4Utc4E1kbCcjgYhfJCUeYLdFp/UKckByYang6C99L337zp4/2mfYyEVg2E+6+G9Y3+RaGYA2iz9cQU43+O0esOuDcc5crmQOLoD+X22D21Dl5IKI=),(**Public_key**))
*(‘&sign=’ shall be not be passed to verify function)

7. *Actions:*
    a. if the verification is successful then the application should bypass passcode page.
    b. If verification is failure either due to corruption or tampering the signature then the intent request must be declined stating ‘intent is tampered or corrupt’.
    c. If signature is not present in intent then the application should show warning message to user that the ‘source of intent could not be verified’ and shall request for passcode to proceed with the payment.

Customer Initiated:

This scenario broadly refers to QRs generated by PSP applications for P2P transactions. In such scenarios public key could not be found in List VAE as the Payee is not a merchant.

1. *Key generation & upload:* The key pair will be generated by the PSP apps and shall be shared with NPCI in offline mode for the keys to get updated in List Keys API. PSP shall have the provision for updating the key pair.
2. *Signing of QR/intent: :* The payee PSP needs to sign the intent with its private key using SHA256 with RSA512 algorithem. The entire content of the string other than ‘&sign=’ is passed into the encryption function. E.g. if the intent is:

   ‘upi://pay?pa=bivekrath@npci&pn=bivek%20rath&mc=0000&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com&mode=02&orgid=123456&mid=1234&msid=3432&mtid=1212’
   then this entire string needs to be passed into encryption function i.e.

   SHA256withRSA512(upi://pay?pa=bivekrath@npci&pn=bivek%20rath&mc=0000&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com&mode=02&&orgid=123456&mid=1234&msid=3432&mtid=1212). Base 64 encoding shall be done to the output.

The output:
sQpPJ0YkdEHIV4b1Hme566aEp1XXXqfe9wqaXgUDfhCfSV1MWdgXnfIQcQYBHaZjDbcuIVrhcq11vhmmURYKsDb1ZbbXRlGXxDhIul5etM/EckmiIbpD90njclCyrLmOe6dp5F0rxzXsiTbjvCN8tUFc4fLFZktMnNF3+L8jqHc=

This output needs to be appended to the intent:
*upi://pay?pa=bivekrath@npci&pn=bivek%20rath&mc=0000&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com&mode=02&orgid=123456&mid=1234&msid=3432&mtid=1212***&sign=sQpPJ0YkdEHIV4b1Hme566aEp1XXXqfe9wqaXgUDfhCfSV1MWdgXnfIQcQYBHaZjDbcuIVrhcq11vhmmURYKsDb1ZbbXRlGXxDhIul5etM/EckmiIbpD90njclCyrLmOe6dp5F0rxzXsiTbjvCN8tUFc4fLFZktMnNF3+L8jqHc=***

As the above example is an sample of QR code created by bank PSP app (‘bivekrath@npci’ with mcc ‘0000’) hence mode is ‘**02**’ & orgid is ‘**123456**’
*(*sign shall be last tag of the intent)*

3. *Key Download*: Payer PSP server needs to download all PSP app keys from response list keys and cache it on their server. This cache needs to be updated daily by the PSP server (recommended 6AM). Each public key downloaded must be mapped with the PSP orgID for which the key was uploaded.
4. *Searching the key*: OrgID & UPI ID of payee need to be extracted first. As the orgid id ‘123456’, the public key needs to be looked in List Keys cache with OrgID as the search parameter. This key will be used for verifying the signature.
8. *Verifying the intent*: The signature part is extracted and base 64 decoded. The entire intent string (excluding the signed part), signature (value within “&sign=”) and the public key of merchant is passed into verify function for verification.

   Verify*((upi://pay?pa=bivekrath@npci&pn=bivek%20rath&mc=0000&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com&mode=02&orgid=123456&mid=1234&msid=3432&mtid=1212),(sQpPJ0YkdEHIV4b1Hme566aEp1XXXqfe9wqaXgUDfhCfSV1MWdgXnfIQcQYBHaZjDbcuIVrhcq11vhmmURYKsDb1ZbbXRlGXxDhIul5etM/EckmiIbpD90njclCyrLmOe6dp5F0rxzXsiTbjvCN8tUFc4fLFZktMnNF3+L8jqHc=),(***Public_key))***
   *(‘&sign=’ shall be not be passed to verify function)

5. *Actions:*
   a. if the verification is successful then the application should bypass passcode page.

b. If verification is failure either due to corruption or tampering the signature then the intent request must be declined stating 'intent is tampered or corrupt'.

c. If signature is not present in intent then the application should show warning message to user that the 'source of intent could not be verified' and shall request for passcode to proceed with the payment.

1.4 Bill in the Box

UPI has a provision where the merchants can share bills/invoices with the customers before the transaction is authorised. The bills/invoices can either be downloaded as PDF or can be viewed on a browser. This provision requires the PSP applicatios to display an option called 'view invoice'. This option will be present for collect transactions and intent/QR initiated pay transactions.

When the user scans a QR or triggers an intent carrying URL tag then he will get an option on the app to click that URL. This URL will can redirect him to the invoice which can be either viewed or downloaded. If URL is not passed then 'view invoice' shall be disabled. Similarly for collect requests received on hadset, by clicking the 'view invoice/bill' option the URL can be browsed.

NOTE:

1. **url** :The parameters present in the URL should be passed as it is in the online message by the PSP application

2. **mam:** This parameter is conditional and shall be used to define a minimum amount rule where amount field in PSP app is editable. If mam tag is not present or 'mam=null' or 'mam=' then amount field should **NOT** be editable.

   Note: if a customer enters the value less than value passed in mam then UPI will decline the transaction. To reduce such declines PSP application should not allow entry of amount below mam value.

3. **Null values:** This needs to be handled by PSP as Null value and should not passed into online message directly as a string value "null".

4. **Space**: Space shall be handled as per below method:

   a) While generating/creating/Reading a QR, intent, NFC, BLE, UHF etc. space (" ") should be represented as "%20" and not "%"as to be compliant with existing Internet Standard RFC 3986 section 2.1 Percent-Encoding.

   Note: Considering that the current PSP apps are developed to read "%" as space (" "), the Bank PSP should support both "%" and "%20", till such time the ecosystem is aligned to the revision. Hence, backward compatibility should be ensured.

5. Mid, msid, mtid will be used by acquiring bank/merchant for reconciliation/confirmation purposes. This will be echoed in all messages.
6. URLs passed in intent/QR shall begin with http or https.

Response Parameters:

As a standard practice merchant app must check the final status with their server/PSP server.

Following is a recommendation on the data returned from the Bank/PSP app to the merchant app

Parameter Name	Mandatory	Data Type	Description
txnId	M	String	Transaction ID from the online message
responseCode	M	String	UPI Response code
ApprovalRefNo	O	String	UPI Approval reference number (beneficiary CBS)
Status	M	String	Status of the transaction: Acceptable values (**SUBMITTED/SUCCESS/FAILURE**)
txnRef	M	String	Transaction reference ID passed in input

Ex 1:
txnId=abcdefghijklmnopqrstuvwxyz123456789&responseCode=00&ApprovalRefNo=122321&Status=SUCCESS&txnRef=6655443322

Ex 2:
txnId=abcdefghijklmnopqrstuvwxyz123451234&responseCode=ZM&ApprovalRefNo=&Status=FAILURE&txnRef=6655443322

Ex 3:
txnId=abcdefghijklmnopqrstuvwxyz123454321&responseCode=Y1&ApprovalRefNo=null&Status=FAILURE&txnRef=6655443322

The bank application may need to whitelist the Merchant App URL.

2 Implementation Samples

2.1 Hyperlink

The user goes to an ecommerce website (My Star Store) on his mobile phone, and places an order. The website generates a link, which the user can click on, to complete the payment.

As per the specification, the link contains the payee details, the transaction reference (order id), and the amount to be paid.

Example:

upi://pay?pa=nadeem@npci&pn=nadeem%20chinna&mc=0000&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com/orderid=9298yw89e8973e87389e78923ue892&mode=00&sign=aagshd4542bdhhvdshsbvqfqttsvsvsbsjn&orgid=00000&mid=1234&msid=3432&mtid=1212

When the user clicks on the link on his mobile browser, it invokes the local PSP application, where the user can confirm the details, and complete the payment.

Because of the design simplicity, user familiarity with hyperlinks, and the ease of sharing, such links can be generated and shared across multiple communication channels, such as email, chat, and social networks.

2.2 QR Code

QR code consists of black modules arranged in a square pattern on a white background. The information encoded can be made up of four standardized kinds (“modes”) of data (numeric, alphanumeric, byte/binary, Kanji), or by supported extensions virtually any kind of data.

QR codes can be used for proximity payments with UPI. Developers who are developing merchant applications must generate a URL fully compliant to specification in previous section and then create a QR code of that URL.

Example:

upi://pay?pa=nadeem@npci&pn=nadeem%20chinna&mc=0000&tid=cxnkjcnkjdfdvjndkjfvn&tr=4894398cndhcd23&tn=Pay%20to%20mystar%20store&am=10&mam=null&cu=INR&url=https://mystar.com/orderid=9298yw89e8973e87389e78923ue892&mode=00&sign=aagshd4542bdhhvdshsbvqfqttsvsvsbsjn&orgid=00000&mid=1234&msid=3432&mtid=1212

Note to PSPs: Considering the simplicity, openness, and wide acceptance of QR codes and its ability to be printed, displayed on PoS devices, and various screens, etc., PSP applications are encouraged to include a QR code scan option within their UPI application so that customers can use a single app to scan and pay.

2.3 Others

UPI linking is protocol agnostic and hence allows innovative mechanisms between merchant/proximity devices to send a UPI intent to customer phone.

For example, a merchant PoS application could create the UPI link (as per spec in previous section) and then transmit using sound to the customer device. Customer PSP app or a utility app can listen to that sound, convert it back to the link, and then launch the UPI application on customer phone to make the payment.

Note that there can be 3rd party general purpose utility applications that allows users to scan these QR codes, launch the link, allow other innovative transfer protocols using sound, etc. Such apps can work as a proxy utility that sends/receives these links and then launch the appropriate apps that are listening to these intents.