UNIFIED PAYMENTS INTERFACE

UPI Linking Specifications Version 1.7

(COMMON URL SPECIFICATIONS FOR DEEP LINKING AND PROXIMITY INTEGRATION)

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, payment request is routed by the initiating application through NPCI switch to 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 payment request with his credentials.

This leads to a significantly smoother payment experience. Some examples of these include in-app payments – where merchant app, may send request to PSP app on the same device, instead of a collect request via 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.

1. Ashok is a student and uses a video application (MyStar) that allows buying on-demand movie on his Android phone.
2. He banks with DiBank (PSP in this case) and uses their mobile application for Android that has implemented UPI features.
3. In MyStar app, Ashok wants to watch a movie for Rs.25.
4. MyStar application creates the UPI payment link as per this spec and launches the Android intent with all necessary parameters populated in the URL.
5. 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.
6. Ashok verifies the info on screen and click pay to complete the payment.

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

1. Mary uses her bank provided UPI application to make payments to nearby grocery store.
2. After the purchase, grocery store PoS application generates a dynamic QR code containing the UPI link (as per this spec) with the payment details.
3. 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.
4. UPI application takes her straight to pay screen with all values pre-populated from the link/intent.
5. She verifies the info on screen and click pay to complete the payment.
6. 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.

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

Example 4: Cash Withdrawal from ATM using UPI.

1. Deepak visit the ATM to withdraw cash.
2. Deepak selects channel as UPI and input the amount for withdrawal on ATM display.
3. ATM shows the details along with a QR code for UPI payment.
4. Deepak opens his UPI application on his mobile and scans the QR code on the ATM screen.
5. UPI application takes him straight to pay screen with all values pre-populated from the QR code which contained the standard UPI link.
6. He verifies the info on screen and click pay to complete the payment.
7. He gets a confirmation on his mobile and the ATM dispenses the cash.

Example 5: International merchant payment at merchant location using UPI.

1. Mary uses her bank provided UPI application to make payments to grocery store present aboard.
2. After the purchase, international grocery store PoS application generates a dynamic QR code containing the UPI link (as per this spec) with the payment details.
3. 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.
4. UPI application takes her straight to pay screen with all values pre-populated from the QR.
5. UPI application display info containing foreign currency, FX rate & payable INR value to Mary. Mary verify the information and click pay to complete the payment.
6. Both merchant and she gets confirmation instantly.

Example 6: Purchase of new subscription with mandate (QR Created by Payee)

1. A subscription form has been released by Chandan Co. for various services. To make the payment of the service seamless, Chandan Co. has printed mandate QRs (unique for each form) on their subscription forms.
2. A user who wants to avail the service, fills the form, scans the QR and authorizes the mandate. Finally he submits the form to Chandan Co.

3. After the success purchase of the services, Chandan Co. will execute the mandate and received the funds

Example 7: Purchase of new subscription with mandate (Payee generated Mandate)

1. A cab service applicati7on provides an option for creation of mandate via intent.
2. By selecting 'Mandate create' an app drawer opens up for the user to select the desired app on which the mandate will be created.
3. The user will validate mandate details like payee name, amount, frequency of payment etc. and authorize the payment.
4. Post each ride, the user account will be auto debited.(For Every Debit a Separate Mandate is to be created)

Example 8: Gifting digital Cash vouchers (Mandate QR created by Payer)

1. A corporate is gifting cash voucher to its employee on a festival occasion.
2. It creates UPI-Mandates on the UPI IDs of the employee's
3. QR is generated for the mandates basis the above specification and is shared with the respective employee.
4. When the employee scans the QR, funds are immediately credited to his account.

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, and UHF etc.

2.1 Creation

Followign are the UPI supported format for creation of UPI URL;

1. **Signed QR/intent:** Normal signed QR/intent, url will start with "**upi://pay**"
2. **UPI - Mandate (Create Mandate):** URL will start with "**upi://mandate**" (payee generate mandate string) or "**upi://collect**" (Payer generated Mandate string) and will also be signed as per signing logic defined.
3. **International QR**: URL will start with "**upiGlobal://**" and will be signed as per sigining logic defined.
4. **Bharat QR**: Please refer latest bharatQr specfication for more details.

1. UPI QR/ Intent URL specification

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

2. Mandate URL specification (applicable only for Create mandate)

   upi://mandate?pa=&pn=&mn=&tid=&type=&validitystart=&validityend=&am=&amrule=&recur=&recurvalue=&recurtype=&tr=&url=&cu=&mc=&tn=&sign=&orgid=&mode=&purpose=

3. Mandate URL specification (UPI mandate created by Payer)
   The below specification will be used for mandates which are already authorised by payer and only needs to be executed by payee by simply scanning the QR.

   upi://collect?umn=&am=&tn=&validitystart=&validityend=&amrule=&pa=&sign=&orgid=&mode=&purpose=

4. International QR specification: (applicable only for international QR)

   upiGlobal://pay?purpose=11&param-name=param-value&param-name=param-value&...

Where param-name can be any of the valid parameters (based on mandatory vs optional) listed in below table.

2.2 URL Parameter

The parameters are defined as follow;

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
1	**ver**	Txn→ QR→ qVer	String	Fixed values	M	M	It is used to specify the QR version ver=01	Common
2	**mode**	Txn→ InitiationMode	String	Fixed values	M	M	01 = Static QR Code (Offline) 02 = Static Secure QR Code (Offline) 04 = Intent 05 = Secure Intent 06 = NFC (Near field communication) 07 = BLE (Bluetooth) 08 = UHF (ultra-high frequency) 13 = Static Secure QR Mandate (Offline) 14 = Restricted 15 = Dynamic QR Code (Offline) 16 = Dynamic Secure QR Code (Offline) 17 = Dynamic Secure QR Mandate (Offline) 18 = ATMQR (Dynamic) 19 = Online STATIC QR Code 20 = Online STATIC Secure QR Code 21 = Online Static QR Mandate 22 = Online Dynamic QR Code 23 = Online Dynamic Secure QR Code 24 = Online Dynamic Secure QR Code Mandate	Common

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
3	**purpose**	Txn→ purpose	String	Fixed values	C	C	01 – SEBI 02 – AMC 03 – Travel 04 – Hospitality 05 – Hospital 06 – Telecom 07 – Insurance 08 – Education 09 – Gifting 10 – BBPS 11 – Global UPI 12 – Metro ATM QR 13 – Non-metro ATM QR 14 – SI 15 – Corporate disbursement Mandatory in string only w.r.t. above mentioned use case defined by NPCI, else optional If present, payer PSP to pass as it is, else payer PSP to populate '00-default'	Common
4	**orgid**	Head→ orgId In case of International QR Payee → Institution → net	String	6 - 12	C	C	Org Id of the acquiring bank/ PSP. In case of UPI Global QR (purpose code - 11), this param shall contain International Institution ID Mandatory in string only w.r.t. use case of signing & international, else optional	Common
5	**tid**	Txn→id	String	35	O	O	This must be PSP generated id when present. In case of Merchant payments,	Common

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
							merchant may acquire the txnId from Payee PSP. If present, payer PSP has to pass this param as Transaction ID, however if the payee PSP doesn't populate the payer PSP to populate the same.	
6	**tr**	Txn→ refid	String	Max: 35	O	M	Transaction reference Id. This could be order number, subscription number, Bill Id, Booking Id, Insurance Renewal reference etc.	Common
7	**tn**	Txn→ Note	String	Max: 50	O	O	Transaction Note providing a short description of the transaction.	Common
8	**category**	Txn→ refCategory	String	Fixed values	C	C	Mandatory in case URL is present 01 – Advertisement 02 – Invoice	Common
9	**url**	Txn→ refurl	String	Max: 99	O	O	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.	Common

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
10	**pa**	Payee→ addr	String	Max: 255	M	M	Payee UPI ID	Common
11	**pn**	Payee→ name	String	Max: 99	M	M	Payee name	Common
12	**mc**	Payee→ mcc	Numeric	4	M	M	Payee merchant Code. 1. Merchant presented QR must contain non zero MCC i.e. 'XXXX' assigned by the acquirer. Person presented QR contains MCC '0000'	Common
13	**am**	Payee & Payer→ Amount→ Value	Numeric	2 digits should come after the decimal.	O	M	Transaction amount in decimal format. If 'am' is not present (static) or If 'mam' is populated and 'am' value is non-zero (dynamic) then 'am' field is editable. It should always be a final debit amount in INR. e.g. (Amount="100.00") If "Purpose = 11", this param is optional	Common
14	**mam**	Txn→ Rules→ MINAMOUNT	Numeric	2 digits should come after the decimal.	O	O	Minimum amount to be defined by payee PSP. It should always be an INR value greater than zero. E.g.(Amount ="100.00")	Common
15	**cu**	Payee→ Amount→ Curr	String	3	O	O	Indicates the currency code of the transaction. A 3-digit alpha code value, as defined by [ISO8583].	Common

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
							This value will be used by the mobile application to display a recognizable currency to the consumer whenever an amount is being displayed or whenever the consumer is prompted to enter amount. This currency field is used for settlement	
16	**mid**	Payee→ Merchant →Mid	String	Max: 20	O	O	Merchant ID shall be passed in this tag.	Common
17	**msid**	Payee → Merchant → Sid	String	Max: 20	O	O	Store id will be passed in this field	Common
18	**mtid**	Payee→ Merchant → Tid	String	Max: 20	O	O	Terminal Id shall be passed in this tag.	Common
19	**cc**	Payee→ Institution → conCode	Numeric	2	C	C	Indicates the country code of the merchant. [A 2-character alpha value, as defined by ISO 3166-1] only mandatory, if purpose="11"	International
20	**enTips**	Tip enTips=Y	String	Fixed values	O	O	Indicates whether the consumer should be prompted to enter tip. 1. If enTips=Y means, payer psp should prompt for tip amount on the app. 2. PSP to capture this value and add the same to the amount	Common

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
							in "am" field for debit while constructing ReqPay API to UPI. This value should be passed in the split tag	
21	**gstBrkUp**	Payee → Amount → Split → name & value	String	Fixed values	O	C	It will contain the break-up of GST amount. Payer PSP to read this amount and restrict display to customer Break up of GST amount GST:amount| CGST:amount| SGST:amount| IGST:amount| CESS:amount| GSTIncentive:amount| GSTPCT:percentage Only mandatory for GST QR	GST
22	**bAm**	Amount→ Split→ base amount & value	Numeric	2 digits should come after the decimal.	O	C	It will contain the purchase amount populated by merchant Only mandatory, if purpose="11"	International
23	**bCurr**	Amount→ Split→ base currency & value	Alpha	3	C	C	Indicates the currency code of the transaction. A 3-digit alpha code value, as defined by [ISO8583]. This value will be used by the mobile application to display a	Interna-tional

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
							recognizable currency to the consumer whenever an amount is being displayed or whenever the consumer is prompted to enter amount. Only mandatory, If purpose="11"	
24	**qrMedium**	Txn→ QR→ qrMedium	Fixed Value	Fixed Value	M	M	This field indicates the Source channel i.e. creation point of the string. 01 - PICK FROM GALLERY 02 - APP 03 – POS 04 – PHYSICAL/ Share Intent mode 05– ATM 06 – WEB To generate the string, code 02 to 06 to be used. Code 01 – Pick from Gallery, applicable only to Payer PSP to identify and pass	Common
25	**invoiceNo**	Payee→ Merchant→ Invoice → Num	Alphanumeric	Max-20	O	C	This field is used to capture the bill/ invoice no. If this tag is present, app should display the value to the customer Only mandatory for GST use case	Common

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
26	**invoiceDate**	Payee→ Merchant→ Invoice→ Date	GMT format		O	C	This field is used to capture the bill invoice date. Only mandatory for GST use case	Common
27	**invoiceName**	Payee→ Merchant→ Invoice→ Name	Alphanumeric	Max-99	O	O	This field is used to track the unique customer name present in the bill If this tag is present, app should display the value to the customer	Common
28	**QRexpire**	Txn→ QR→ expireTs	GMT format		O	O	Dynamic or Static QR – QR expiry date & time	Common
29	**QRts**	Txn→ QR→ Ts	GMT format		O	O	This is the time stamp when the QR was created. In case it is present, App should state that the QR was created on this time stamp	Common
30	**split**	Payee→ Amount→ split → name & value	String	Max: 255	O	O	It will contain the split details of amount. The PSP must compute the Amount value & Tip amount to show the final amount to customer for PIN authorization (one or more split value will come) CONFEE: 30| CONPCT: 2%| TIPS:1| DISCNT: 10%| DISPCT:10| FX:30| MKUP: 5%	Common

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
31	**pinCode**	Merchant→ identifier→ pinCode	alphanumeric	Max: 10	O	O	Zip code or Pin code or Postal code of the merchant.	Common
32	**Tier**	Merchant→ Identifier→ Tier	Fixed Value	Fixed Value	O	O	Denotes the tier of the city on basis of population TIER1| TIER2| TIER3| TIER4| TIER5| TIER6	Common
33	**txnType**	txn→ type	String	Fixed values	O	O	This is txn type which denote type of txn PAY/COLLECT/CREATE/UPDATE/REVOKE/PAUSE/UNPAUSE	Common
34	**consent**	Payee → Consent → name & value	String	String	O	O	The consent type denotes the purpose for which the customer's consent is being taken. This is for specific use cases as may be defined in future. This tag is not applicable for GST	Common
35	**gstIn**	Payee → Consent → name & value	String	String	C	C	The merchant populates GSTIN value under this tag. This tag is applicable for GST QR	GST
36	**mn**	mandate→ name	String	Max - 99	O	C	Mandate name, specifies the purpose of mandate Applicable only for Mandate URL	Mandate
37	**type**	mandate→ type	String	Max - 99	O	C	Future use	Mandate

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
							Applicable only for Mandate URL	
38	**validitystart**	mandate→ validity → start	String		M	C	Defines start time of mandate validity (Format ddmmyyyy) Applicable only for Mandate URL	Mandate
39	**validityend**	mandate→ validity → start	String		M	C	Defines end time of mandate validity (format ddmmyyyy) Applicable only for Mandate URL	Mandate
40	**amrule**	mandate→ Amount→ rule	String		O	C	'MAX' or 'EXACT' rule applied to mandate (Optional, default value to be passed in online message in case amrule is not passed in QR is 'MAX') Applicable only for Mandate URL	Mandate
41	**recur**	Mandate→ Recurrence	String	Fixed values	M	C	Specifies the frequency of mandate debit (ONETIME|DAILY|WEEKLY|FORTNIGHTLY| MONTHLY| BIMONTHLY|QUARTERLY|HALFYEARLY|YEARLY| ASPRESENTED") Applicable only for Mandate URL	Mandate
42	**recurvalue**	Mandate→ Recurrence → rule →value	String	Max-99	O	C	Specifies date along with 'recurtype' for debit (FOR FUTURE USE) Applicable only for Mandate URL	Mandate
43	**recurtype**	Mandate→ Recurrence → rule →type	String	Fixed values	O	C	Can have values: (BEFORE|ON|AFTER), Specifies date along with 'recurvalue' for debit	Mandate

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type S	URL Type D	Description	Purpose of Fields
							Applicable only for Mandate URL	
44	**rev**	rev = Y/N	String	Fixed values	O	C	Revocable tag can be passed as Y/N. In case this tag is not present in the QR, App should by default pass the value as 'Y' Applicable only for Mandate URL	Mandate
45	**share**	share = Y/N	String	Fixed values	O	C	Share to Payee option can be given as Y/N. In case the tag is not present in the QR, customer will have the option to opt for the same in the app. Applicable only for Mandate URL	Mandate
46	**block**	block – Y/N	String	Fixed values	O	C	this field is used for intimating remitter bank to block the necessary fund against customer account. Applicable only for Mandate URL	Mandate
47	**umn**	Payer→ addr	String	35	O	C	Unique mandate number shared by payer for the payee to initiate the debit. (For Future Use when other type will be included) Applicable only for Mandate URL	Mandate
48	**skip**	-	String	2	O	O	(future use) mandate can be executed for a non – registered mandate i.e. mandate is not present with Payee PSP	Mandate

Sr. No.	Parameter Name	UPI API field	Data Type	Size	URL Type		Description	Purpose of Fields
					S	D		
							Applicable only for Mandate URL	
49	**sign**		String		M	M	Digital Signature needs to be passed in this tag. (It is used for Securing the String)	Common + Secure QR
50	**query**	Txn→ QR→ query	String	Max: 255	O	O	This is for future use in JSON format. We can add multiple fields basis requirements	Common

Following are the definition for the header used in the above table;

1. **S -** Static QR and **D -** Dynamic QR
2. **Table Content**
   a. **M –** Mandatory
   b. **O –** Optional
   c. **C –** Conditional
3. **Purpose of the field:** Usage of the tags
   a. **Common –** Basic fields of UPI and applicable for all QR type
   b. **Mandate –** Only applicable for mandate QR/Intent flow
   c. **GST –** Only applicable for GST functionality
   d. **International Transactions –** Only applicable for International transactions
   e. **Secure QR –** Only applicable for secure signed QR/ intent

All UPI Applications should mandatorily scan all the tags present in the QR/ Intent (Both Optional or Mandatory) and pass. The values should not be edited by the PSP or the Customer (Only Amount Tag to be editable in presence of a MAM Tag).

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

1. If the application and 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 BT, Wi-Fi Direct, NFC, Sound, etc.) to transfer the URL data to customer mobile on which is gets deep linked to their PSP application.
4. While reading a QR, intent, NFC, BLE, UHF etc. all parameters must be read and passed to online message.
5. If any tag is not present it can be dropped or passed as null.

6. **Reading Signed URL string:**

   Following are the verification logic for reading the signed QR/ Intent on UPI.

   a. If OrgId present in URL is non-zeros & MCC=0000, then PSP should call ListKeys for fetching PSP keys.
   b. If OrgId=0000 & mcc is non-zeros, then PSP should call ListVAE with reference of Payee UPI ID for fetching merchant specific key (Verified merchant will fall under this rule)
   c. If OrgId and MCC are non-zeros, then PSP should call first ListVae with payee UPI ID. If no entry is found in listvae then the PSP can call ListKeys with parent orgId to get the keys (mostly non verified merchants will fall under this rule).
   d. Both Orgid and MCC = 0000 such a scenario doesn't exist.

7. 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.

2.3 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.

2.3.1 Merchant Initiated

1. ***Key generation***: Merchant or the acquiring bank on behalf of merchant need to generate a key pair (public and private key with (ECDSA 256 + SHA 256) specifications.

   If Acquiring 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. Merchant and member banks shall also add provision for update of key pairs.

2. ***Key Upload***: Then merchant need to share this public key with its acquiring bank. Then Acquiring bank will upload its 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.

   Banks public key will be available at VAE or List key entry;

   i. List VAE is having the merchant entry and having Signature key, then the key will be used for validating QR/Intent
   ii. List VAE is having merchant entry and no signature key, use parent key for checking signature basis parent Org ID
   iii. No entry in List VAE, use parent key for checking signature basis parent Org ID (non-verified merchant).
   iv. List Keys are used in case of fetching keys for P2P

3. ***Signing of intent:***

   Step 1: The merchant should get required details from customer and form an URN.

```
Sample Inputs

//URN
"upi://pay?ver=01&mode=05&orgid=700004&tid=TESTGST123456qazqwe12345hs2
9hnd1qa2&tr=MerRef123&tn=GST%20QR&category=02&url=https://www.test.com&
pa=merchant@npci&pn=Test%20Merchant&mc=5411&am=100.00&cu=INR&mid=TS
T5432&msid=TSTABC123&mtid=TSTABC1234&gstBrkUp=CGST:08.45|SGST:08.45
&qrMedium=02&invoiceNo=BillRef123&invoiceDate=2019-06-
11T13:21:50+05:30&invoiceName=Dummy%20Customer&QRexpire=2019-06-
11T13:21:50+05:30&QRts=2019-06-
12T13:21:50+05:30&pinCode=400063&tier=TIER1&gstIn=GSTNUM1234567890"
```

```
Sample Output
 //URN
intentURL:
"upi://pay?ver=01&mode=05&orgid=700004&tid=TESTGST123456qazqwe12345hs2
9hnd1qa2&tr=MerRef123&tn=GST%20QR&category=02&url=https://www.test.com&
pa=merchant@npci&pn=Test%20Merchant&mc=5411&am=100.00&cu=INR&mid=TS
T5432&msid=TSTABC123&mtid=TSTABC1234&gstBrkUp=CGST:08.45|SGST:08.45
&qrMedium=02&invoiceNo=BillRef123&invoiceDate=2019-06-
11T13:21:50+05:30&invoiceName=Dummy%20Customer&QRexpire=2019-06-
11T13:21:50+05:30&QRts=2019-06-
12T13:21:50+05:30&pinCode=400063&tier=TIER1&gstIn=GSTNUM1234567890"
```

Step 2: The Specified URN needs to be signed using ECDSA with SHA256 algorithm.

Sample Inputs

```
//Signature with Sha-256
Signature dsa = Signature.getInstance("SHA256withECDSA");
 dsa.initSign(privKey);

String intentURL =
"upi://pay?ver=01&mode=05&orgid=700004&tid=TESTGST123456qazqwe12345hs29hnd1qa2&tr=MerRef123&tn=GST%20QR&category=02&url=https://www.test.com&pa=merchant@npci&pn=Test%20Merchant&mc=5411&am=100.00&cu=INR&mid=TST5432&msid=TSTABC123&mtid=TSTABC1234&gstBrkUp=CGST:08.45|SGST:08.45&qrMedium=02&invoiceNo=BillRef123&invoiceDate=2019-06-11T13:21:50+05:30&invoiceName=Dummy%20Customer&QRexpire=2019-06-11T13:21:50+05:30&QRts=2019-06-12T13:21:50+05:30&pinCode=400063&tier=TIER1&gstIn=GSTNUM1234567890";

System.out.println("\nintentURL:" + intentURL);
byte[] strByte = intentURL.getBytes("UTF-8");
dsa.update(strByte);

//Sign with private key
byte[] realSig = dsa.sign();
```

Sample Output

Private Signing Output

```
3046022100da68400bd3a56c9fd68d21aacfedd6a2f3f87d70cad5bee61efae972d2b
60833022100f463cfa0a3d59a1d0cb7ff3592066275b8832a44f380cd0f34444d04d8
d48a4c
```

Step 3: The output of the signature should be encoded with base 64

```
Sample Input

//Encode signed URL with base 64

String encodedSign =
java.util.Base64.getEncoder().encodeToString(realSig);
```

```
Sample Output
Encoded
Signature:MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1Z
odDLf/NZIGYnW4gypE84DNDzRETQTY1IpM
```

Step 4: The final encoded value should be appended post delimiter “sign=” to original URN

```
Sample Input
//Append with Signed URL
String finalBHIMURL = intentURL+"&sign="+encodedSign;
System.out.println("\nSignature: " + new BigInteger(1, realSig).toString(16));

System.out.println("\nEncoded Signature:" + encodedSign);

System.out.println("\nFinal BHIM URL:" + finalBHIMURL);
```

```
Sample Output
Final Output
URL:
upi://pay?ver=01&mode=05&orgid=700004&tid=TESTGST123456qazqwe12345hs29
hnd1qa2&tr=MerRef123&tn=GST%20QR&category=02&url=https://www.test.com&
pa=merchant@npci&pn=Test%20Merchant&mc=5411&am=100.00&cu=INR&mid=TS
T5432&msid=TSTABC123&mtid=TSTABC1234&gstBrkUp=CGST:08.45|SGST:08.45
&qrMedium=02&invoiceNo=BillRef123&invoiceDate=2019-06-
11T13:21:50+05:30&invoiceName=Dummy%20Customer&QRexpire=2019-06-
11T13:21:50+05:30&QRts=2019-06-
12T13:21:50+05:30&pinCode=400063&tier=TIER1&gstIn=GSTNUM1234567890&sig
n=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/N
ZIGYnW4gypE84DNDzRETQTY1IpM
```

As the above example is an android intent sample initiated by merchant ('merchant@npci' with mcc '5411') hence mode is '**05**' & orgid is '**700004**'

4. ***Key Download:*** Payer PSP server need to download all merchant keys and cache it on their server. This cache needs to be updated daily by the PSP server. 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:***

   Step 1: The signature part need to be extracted and separate the original text from the encrypted code.

   Step: 2 Then the appended value must be decode with base 64

```
Sample Input

//Decode the signed URL with base 64
String[] bhimReceivedURL = finalBHIMURL.split("&sign=");
byte[] decodedSign =
java.util.Base64.getDecoder().decode(bhimReceivedURL[1]);

System.out.println("\nDecoded Signature: " + new BigInteger(1,
decodedSign).toString(16));
```

```
Sample Output
Decoded Signature:
3046022100da68400bd3a56c9fd68d21aacfedd6a2f3f87d70cad5bee61efae972d
2b60833022100f463cfa0a3d59a1d0cb7ff3592066275b8832a44f380cd0f34444d
04d8d48a4c
```

Step 3: The entire intent string excluding the signed part will be passed into verify function along with signature and the public key of merchant for verification.

```
Sample Input
//Initialise verification with public key
dsa.initVerify(pubKey);
//Initialise using plain text bytes
dsa.update(bhimReceivedURL[0].getBytes("UTF-8"));
//Verify against sign
boolean verified = dsa.verify(decodedSign);
System.out.println("\nSignature Verify Result:"+verified);
```

```
Sample Output
Signature Verify Result: true
```

4. *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.

2.3.2 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 for the keys to get updated in List Keys API. PSP shall have the provision for updating the key pair.

2. ***Signing of intent***:
   Step 1: The merchant should get required details from customer and form an URN.

```
Sample Inputs
//URN
"upi://pay?ver=01&mode=16&orgid=159991&tid=NPCI8cdd79c7391467cb76988f
e8bbccebc&tr=4894398cndhcd23&tn=P2P%20Transaction&pa=deepak@npci&pn
=Deepak%20Gupta&mc=0000&am=1000.00&cu=INR&qrMedium=02"
```

```
Sample Output
//URN
intentURL:upi://pay?ver=01&mode=16&orgid=159991&tid=NPCI8cdd79c7391467
cb76988fe8bbccebc&tr=4894398cndhcd23&tn=P2P%20Transaction&pa=deepak@
npci&pn=Deepak%20Gupta&mc=0000&am=1000.00&cu=INR&qrMedium=02
```

Step 2: The Specified URN needs to be signed using SHA256 with ECDSA algorithm.

```
Sample Inputs
//Signature with Sha-256
Signature dsa = Signature.getInstance("SHA256withECDSA");
dsa.initSign(privKey);
String intentURL = "
upi://pay?ver=01&mode=16&orgid=159991&tid=NPCI8cdd79c7391467cb76988fe8
bbccebc&tr=4894398cndhcd23&tn=P2P%20Transaction&pa=deepak@npci&pn=De
epak%20Gupta&mc=0000&am=1000.00&cu=INR&qrMedium=02";
System.out.println("\nintentURL:" + intentURL);
byte[] strByte = intentURL.getBytes("UTF-8");
dsa.update(strByte);
//Sign with private key
byte[] realSig = dsa.sign();
```

```
Sample Output
Private Signing Output
3046022100da68400bd3a56c9fd68d21aacfedd6a2f3f87d70cad5bee61efae972d2b
60833022100f463cfa0a3d59a1d0cb7ff3592066275b8832a44f380cd0f34444d04d8
d4
```

Step 3: The output of the signature should be encoded with base 64

```
Sample Input
//Encode signed URL with base 64
String encodedSign = java.util.Base64.getEncoder().encodeToString(realSig);
```

```
Sample Output
Encoded Signature:
MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZI
GYnW4gypE84DNDzRETQTY1IpM
```

Step 4: The final encoded value should be appended post delimiter “sign=” to original URN

```
Sample Input
//Append with Signed URL
String finalBHIMURL = intentURL+"&sign="+encodedSign;
System.out.println("\nSignature: " + new BigInteger(1, realSig).toString(16));
System.out.println("\nEncoded Signature:" + encodedSign);
System.out.println("\nFinal BHIM URL:" + finalBHIMURL);
```

```
Sample Output
Final Output
URL:upi://pay?ver=01&mode=16&orgid=159991&tid=NPCI8cdd79c7391467cb76
988fe8bbccebc&tr=4894398cndhcd23&tn=P2P%20Transaction&pa=deepak@np
ci&pn=Deepak%20Gupta&mc=0000&am=1000.00&cu=INR&qrMedium=02&sign=
MEYCIQDaaEAL06Vsn9aNlarP7dai8/h9cMrVvuYe+uly0rYIMwlhAPRjz6Cj1ZodDLf/
NZIGYnW4gypE84DNDzRETQTY1IpM
```

As the above example is a sample of dynamic QR code created by bank PSP app (‘deepak@npci’ with mcc ‘0000’) hence mode is ‘**16**’ & orgid is ‘**159991**’

3. ***Key Download:***

   Payer PSP server need to download all PSP app keys and cache it on their server. This cache needs to be updated daily by the PSP server. Each public key downloaded must be mapped with the PSP orgID for which the key was uploaded.

4. ***Searching Key***:

   OrgID & UPI ID of payee need to be extracted first. As the orgid id ‘159991’, 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.

5. ***Verifying the intent***:

   Step 1: The signature part need to be extracted and separate the original text from the encrypted code.

Step 2: Then the appended value must be decode with base 64

```
Sample Input
//Decode the signed URL with base 64
String[] bhimReceivedURL = finalBHIMURL.split("&sign=");
byte[] decodedSign =java.util.Base64.getDecoder().decode(bhimReceivedURL[1]);
System.out.println("\nDecoded Signature: " + new BigInteger(1,
decodedSign).toString(16));
```

```
Sample Output
Decoded Signature:
3046022100da68400bd3a56c9fd68d21aacfedd6a2f3f87d70cad5bee61efae972d2b6
0833022100f463cfa0a3d59a1d0cb7ff3592066275b8832a44f380cd0f34444d04d8d4
```

Step 3: The entire intent string excluding the signed part will be passed into verify function along with signature and the public key of merchant for verification.

```
Sample Input
//Initialise verification with public key
dsa.initVerify(pubKey);
//Initialise using plain text bytes dsa.update(bhimReceivedURL[0].getBytes("UTF-
8"));
//Verify against sign
boolean verified = dsa.verify(decodedSign);
System.out.println("\nSignature Verify Result:"+verified);
```

```
Sample Output
Signature Verify Result: true
```

6. *Actions:*

a) If the verification is successful, then the application should bypass passcode.

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.

3 Invoice 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 applications to display an option called 'view invoice'. This option will be present for collect 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. Similarly for collect requests received on handset from a Verified Merchant, by clicking the 'view invoice' option the URL can be browsed.

4 NOTE

1. **url**: The parameters present in the URL should be passed as it is in the online message by the PSP application and it always start from http or https i.e url=https:// or url=http://

   Checks on the URL which are to be performed by the APPS on Security, Source of Generation, Country etc needs to be specified.

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.

5 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)
Status	M	String	Status of the transaction (SUBMITTED/SUCCESS/FAILURE)
txnRef	M	String	Transaction reference ID passed in input
appId	M	String	Payer Application ID, where customer input UPI PIN

Ex 1:

txnId=abcdefghijklmnopqrstuvwxyz123456789&responseCode=00&ApprovalRefNo=122321&Status=SUCCESS&txnRef=6655443322&appId=com.bhim

Ex 2:

txnId=abcdefghijklmnopqrstuvwxyz123451234&responseCode=ZM&ApprovalRefNo=&Status=FAILURE&txnRef=6655443322&appId=com.bhim

Ex 3:

txnId=abcdefghijklmnopqrstuvwxyz123454321&responseCode=Y1&ApprovalRefNo=null&Status=FAILURE&txnRef=6655443322&appId=com.bhim

Mandatorily the final response should be sent back to the initiating Application in case of an Intent.

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

6 Implementation Samples

6.1 Hyperlink (Intent URL)

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?ver=01&mode=05&orgid=700004&tid=TESTGST123456qazqwe1234
5hs29hnd1qa2&tr=MerRef123&tn=GST%20QR&category=02&url=https://www
.test.com&pa=merchant@npci&pn=Test%20Merchant&mc=5411&am=100.00
&cu=INR&mid=TST5432&msid=TSTABC123&mtid=TSTABC1234&gstBrkUp=CG
ST:08.45|SGST:08.45
&qrMedium=02&invoiceNo=BillRef123&invoiceDate=2019-06-
11T13:21:50+05:30&invoiceName=Dummy%20Customer&QRexpire=2019-06-
11T13:21:50+05:30&QRts=2019-06-
12T13:21:50+05:30&pinCode=400063&tier=TIER1&gstIn=GSTNUM123456789
0&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uIy0rYIMwIhAPRj
z6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM
```

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.

6.2 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?ver=01&mode=16&orgId=159992&tid=NPCI8cdd79c7391467cb76988fe8bbccebc&tr=MerRef123&tn=Merchant%20QR&category=02&url=https://www.test.com&pa=merchant@npci&pn=Merhcant&mc=5411&am=100.00&cu=INR&mid=12343&mtid=12343&qrMedium=06&QRexpire=2019-06-11T13:21:55+05:30&sign=**MEYCIQDaaEAL06Vsn9aNlarP7dai8/h9cMrVvuYe+uly0rYIMwlhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM**

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.

Sample QR

1) Merchant Presented Singed Dynamic QR generated on POS

upi://pay?ver=01&mode=16&orgid=159992&tid=NPCI8cdd79c7391467cb76988fe8bbccebc&tr=MerRef123&tn=Merchant%20QR&category=02&url=https://www.test.com&pa=merchant@npci&pn=Merchant%20Name&mc=5411&am=1000.00&mid=12343&msid=56789&mtid=12343&qrMedium=03&QRexpire=2019-06-11T13:21:55+05:30&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM

2) Merchant Presented Singed Dynamic GST QR generated on POS

upi://pay?ver=01&mode=16&orgid=700004&tid=TESTGST123456qazqwe12345hs29hnd1qa2&tr=MerRef123&tn=GST%20QR&pa=merchant@mypsp2&pn=Test%20Merchant&mc=5411&am=100.00&mid=TST5432&msid=TSTABC123&mtid=TSTABC1234&gstBrkUp=CGST:08.45|SGST:08.45&qrMedium=03&invoiceNo=BillRef123&InvoiceDate=2019-06-11T13:21:50+05:30&InvoiceName=Dummy%20Customer&QRexpire=2019-06-11T13:21:50+05:30&pinCode=400063&tier=TIER1&gstIn=GSTNUM1234567890&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM

3) **Merchant Presented Singed Static QR code generated & pasted near counter**

upi://pay?ver=01&mode=02&orgid=159992&tr=MerRef123&tn=Merchant%20QR&pa=merchant@npci&pn=Merchant%20Name&mc=5411&mid=12343&msid=56789&mtid=12343& qrMedium=04&QRexpire=2020-06-11T13:21:55+05:30&pinCode=400063&tier=”Tier1”&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM

4) **Merchant Presented GST Singed Static QR code generated & pasted near counter**

upi://pay?ver=01&mode=02&orgid=159992&tr=MerRef123&tn=Merchant%20QR&pa=merchant@npci&pn=Merchant%20Name&mc=5411&mid=12343&msid=56789&mtid=12343& qrMedium=04&QRexpire=2020-06-11T13:21:55+05:30&pinCode=400063&tier=”Tier1”& gstIn=GSTNUM1234567890&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM

5) Merchant Presented Singed Dynamic QR (International) generated on POS

upiGlobal://pay?ver=01&mode=16&**purpose=11**&orgid=1599910001&tid=NPCI8cdd79c7391467cb76988fe8bbccebc&tr=MerRef123&tn=Global%20Merchant&pa=globalmerchant@npci&pn=Global%merchant&mc=5411&mid=123412&msid=321231&mtid=123123&cc=SG&bAm=36.00&bCurr=SGD&qrMedium=03&invoiceNo=BillRef123&invoiceDate=2019-06-11T13:21:50+05:30&invoiceName=Global%20Merchant&QRexpire=2019-06-11T13:21:50+05:30&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM

6) Merchant Presented Singed Static QR (International) & pasted near counter

upiGlobal://pay?ver=01&mode=02&**purpose=11**&orgid=1599910001&tr=MerRef123&tn=Global%20Merchant&pa=globalmerchant@npci&pn=Global%merchant&mc=5411
&mid=123412&msid=321231&mtid=123123&cc=SG&bCurr=SGD&qrMedium=04&invoiceNo=BillRef123&invoiceDate=2019-06-11T13:21:50+05:30&invoiceName=Global%20Merchant&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM

7) Merchant Presented Singed dynamic mandate QR

upi://mandate?ver=01&mode=24&orgid=159991&tid=NPCI8cdd79c7391467cb76988fe8bbccebc&tr=MerRef123&tn=Mandate%20QR&category=02&url=https://www.test.com&pa=merumntest@npci&pn=Merhcant%20Mandate&mc=5411&am=1000.00&cu=INR&mid=12343&mtid=12343&qrMedium=06&QRexpire=2019-06-11T13:21:55+05:30&txnType=CREATE&mn=Test%20Mandate&type=&validitystart=11062019&validityend=11102020&amrule=MAX&recur=ONETIME&rev=Y&share=Y&block=Y&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM

8) Person presented signed Dynamic QR

upi://pay?ver=01&mode=22&orgid=159991&tid=NPCI8cdd79c7391467cb76988fe8bbccebc&tr=4894398cndhcd23&tn=P2P%20Transaction&pa=deepak@npci&pn=Deepak%20Gupta&mc=0000&am=1000.00&qrMedium=02&sign=MEYCIQDaaEAL06Vsn9aNIarP7dai8/h9cMrVvuYe+uly0rYIMwIhAPRjz6Cj1ZodDLf/NZIGYnW4gypE84DNDzRETQTY1IpM

6.3 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.

7 Appendix

```
import java.math.BigInteger;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.spec.ECGenParameterSpec; public class TestECC {
public static void main(String args[]) {
try {
KeyPairGenerator kpg = KeyPairGenerator.getInstance("EC");
System.out.println(kpg.getAlgorithm());
ECGenParameterSpec ecsp = new ECGenParameterSpec("secp256r1");
kpg.initialize(ecsp);  //ecsp
KeyPair kyp = kpg.genKeyPair();
PublicKey pubKey = kyp.getPublic();
PrivateKey privKey = kyp.getPrivate();
System.out.println("\n\n");
//Signature with Sha-256
Signature dsa = Signature.getInstance("SHA256withECDSA"); dsa.initSign(privKey);
String intentURL =
"upi://pay?ver=01&mode=05&orgid=700004&tid=TESTGST123456qazqwe12345hs29
hnd1qa2&tr=MerRef123&tn=GST%20QR&category=02&url=https://www.test.com&p
a=merchant@npci&pn=Test%20Merchant&mc=5411&am=100.00&cu=INR&mid=TST5
432&msid=TSTABC123&mtid=TSTABC1234&gstBrkUp=CGST:08.45|SGST:08.45
&qrMedium=02&invoiceNo=BillRef123&invoiceDate=2019-06-
11T13:21:50+05:30&invoiceName=Dummy%20Customer&QRexpire=2019-06-
11T13:21:50+05:30&QRts=2019-06-
12T13:21:50+05:30&pinCode=400063&tier=TIER1&gstIn=GSTNUM1234567890"
System.out.println("\nintentURL:" + intentURL);
byte[] strByte = intentURL.getBytes("UTF-8");
dsa.update(strByte);
//Sign with private key
byte[] realSig = dsa.sign();
//Append with Signed URL

String finalBHIMURL = intentURL+"&sign="+encodedSign;
System.out.println("\nSignature: " + new BigInteger(1, realSig).toString(16));
System.out.println("\nEncoded Signature:" + encodedSign);
System.out.println("\nFinal BHIM URL:" + finalBHIMURL);

//Decode the signed URL with base 64

String[] bhimReceivedURL = finalBHIMURL.split("&sign=");

byte[] decodedSign = java.util.Base64.getDecoder().decode(bhimReceivedURL[1]);
System.out.println("\nDecoded Signature: " + new BigInteger(1,
decodedSign).toString(16));

//Initialise verification with public key

dsa.initVerify(pubKey);

//Initialise using plain text bytes dsa.update(bhimReceivedURL[0].getBytes("UTF-
8"));

//Verify against sign

boolean verified = dsa.verify(decodedSign);
System.out.println("\nSignature Verify Result:"+verified);
}
catch (Exception e)
{
System.out.print(e);
}
}
}
```