American National Standard
for Financial Services

ANSI X9.150
Payment QR Code Standard

Merchant-Presented QR Codes For Secure Payment

Developed by
Accredited Standards Committee X9, Incorporated
Financial Industry Standards

Date Approved:

American National Standards Institute

This page left intentionally blank

CONTENTS

1. Scope 6

2. Purpose 7

3. Normative References 8

4. Terms and definitions 9

5. Payment QR Code Overview 12

6. QR Code 15
6.1 General Overview of QR Code 15
6.2 QR Code Requirements 15

7. JSON Data Type 18

8. Payment Payload 19
8.1 General Overview of Payment Payload 19
8.2 Payment Payload Request 20
8.3 Payment Payload Response 20
8.4 Payment Payload Requirements 20

9. Payment Notification 41
9.1 General overview and structure 41
9.2 Trigger & timing requirements 41
9.3 Payment Notification Requirements 41

10. Security 44
10.1 Security Overview (JWS) 44
10.1.1 Objects Requiring Digital Signatures 45
10.1.2 Security Flow 45
10.1.3 X9 Financial PKI 46
10.1.4 Alternative Encodings (Informative) 46
10.2 JSON Web Signature Overview (JWS) 46
10.2.1 JWS Structure 46
10.2.2 How Application Data is Transmitted 47
10.3 JWS Protected Header Requirements 47
10.3.1 Standard JOSE Header Fields 47
10.3.2 Custom Header Fields for X9.150 47
10.4 JWS Body Structure 48
10.5 JWS Signature Requirements (RFC 7515) 49
10.5.1 Algorithms 49
10.5.2 Trust & Certificates 49
10.6 Trust Establishment 49
10.6.1 Transport-Layer Authentication 49
10.6.2 Application-Layer Signature Verification 49
10.7 Signature Verification Process 49
10.8 Digital Signature Responsibility 50
10.9 Key Management 50
10.10 Certificate Management 51
10.11 Conformance 51

11. Bibliography 51

Annex (Informative) – Market Practices 52
A.1 QR Code 53
A.3 Payment Notification 56
A.4 JWS Protected Headers 56
A.4.1 Request Header Example JSON: 56
A.4.2 Response Header Example JSON: 57
A.4.3 JWS Body for Payment Payload Request Example 57
A.4.4 JWS Body for Payment Payload Response Example 57
A.4.5 JWS Body for Payment Notification Example 58
A.5 Matching Payer's payment application options to Payee's allowable payment methods 59

A.6 Pacs.008 Mapping (RTP/FedNow) for ID, Tip and Unstructured info 59
A.7 ACH mapping for ID, Tips and Unstructured info 60
A.8 Example: Validations & responsibilities related to the Payment Payload..60
A.11 Payment notification trigger & timing 62

Foreword

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretation should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken to reaffirm, revise, or withdraw this standard no later than five years from the date of approval.

Published by

Accredited Standards Committee X9, Incorporated
Financial Industry Standards
275 West Street, Suite 107
Annapolis, MD 21401 USA
X9 Online http://www.x9.org

Introduction

1. Scope

This ANSI Standard defines the structure, content, and security requirements for merchant-presented, dynamically generated payment QR Codes used to initiate push payments in the United States. It specifies the data elements, message structures, and associated security requirements that govern the QR Code content, Payment Payload Data Model, and Payment Notification needed to support interoperable, secure initiation and processing of QR code-based payments across multiple payment networks and service providers.

Items within scope of this standard includes:

- **QR Code Content:** Defines the structure and data elements of the dynamic QR Code that is presented to the Payer at the point of interaction. To support the purpose of interoperability, the QR Code Content is specified in accordance with, and fully compliant to, the EMV QR Code Merchant Presented Mode Specification.[1]

- **Payment Payload:** Specifies the JSON payload referred by the QR Code Content, including fields that identify the destination account, amounts, and related metadata.

- **Payment Notification:** Defines the structure and delivery of a standardized notification from the Payer's PSP to the Payee's PSP, confirming that a payment has been initiated or completed through the selected payment network.

- **Security:** Establishes requirements for signing and validating QR Code Content, Payment Payload Requests and Responses, and Payment Notifications using cryptographic methods, in conjunction with the complementary X9 security standards.

Items out of scope of this standard includes:
- Consumer-presented QR Codes
- Merchant-presented Static QR Codes
- Card-based (pull) transactions
- How merchants/billers request a QR Code from Payment Service Providers (PSPs), and how Payment Service Providers (PSPs) respond to that request
- Operational infrastructure for clearing and settling payments
- How a Payer chooses to fund a transaction

Note that future versions of this standard may include other items.

2. Purpose

The purpose of this standard is to define a standard approach for QR code based payment initiation. The goal is to enable interoperable, secure, push payment experiences across various networks and devices.

- A push payment is one where the Payer initiates the transfer by sending (pushing) funds directly to the Payee's account (e.g. FedNow, RTP or ACH credit).

- A pull payment is one where the Payee (merchant) initiates a request to pull funds from the Payer's account (e.g. credit card or ACH debit).

[1] EMV® is a registered trademark in the U.S. and other countries and an unregistered trademark elsewhere. The EMV trademark is owned by EMVCo, LLC.

- Networks are payment networks like, but not limited to, RTP, FedNow, ACH, Zelle.
- Devices are things like a merchant POS or a consumer's mobile phone.

3. Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the specific edition cited applies. For undated references, the most recent edition of the referenced document (including any amendments) applies.

1. **[ISO4217]** International Organization for Standardization, "Codes for the representation of currencies and funds." (For currency codes like USD)

2. **[ISO3166-1]** International Organization for Standardization, "Codes for the representation of names of countries and their subdivisions[2] – Part 1: Country codes." (For alpha-2 country codes like US)

3. **[ISO3166-2]** International Organization for Standardization, "Codes for the representation of names of countries and their subdivisions – Part 2: Country subdivision code." (For alpha-2 state codes like WA)

4. **[ISO18245]** International Organization for Standardization, "Retail financial services – Merchant category codes (MCCs)."

5. **[ISO18004]** International Organization for Standardization," Information technology - Automatic identification and data capture techniques - QR Code barcode symbology specification"

6. **[ISO19160-4]** Universal Postal Union, "S42: International Postal Address Components and Templates."

7. **[ISOIEC13239]** International Organization for Standardization/International Electrotechnical Commission, "Information technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures."

8. **[ISO8601]** - International Organization for Standardization, "Date and time — Representations for exchange."

9. **[ISO20022]** International Organization for Standardization, "Financial services — Universal financial industry message scheme (ISO20022)."

10. **[X9.148]** ANSI X9.148, "Quick Response (QR) Code - Protection using Cryptographic Solutions." (Provides guiding principles for QR code security)

11. **[X9.69]** ANSI X9.69, Framework for Key Management Extensions. (Defines methods for generation and control of keys, including key usage controls.)

12. **[X9.73]** ANSI X9.73, Cryptographic Message Syntax (CMS). (Specifies a cryptographic message syntax suitable for protecting financial messages in ASN.1 or XML encodings.)

13. **[X9 SD-34]** Accredited Standards Committee X9 Registry of Approved Cryptographic Resources, SD-34, most recent approved version.

3.14 **[X9 Registry SD-34]** #00002 FIPS 197: Advanced Encryption Standard (AES)

3.15 **[X9 Registry SD-34]** #00003 FIPS 180-4: Secure Hash Standard (SHS)

3.16 **[X9 Registry SD-34]** #00009 FIPS 186-5 Digital Signature Standard (DSS)

3.17 **[X9 Registry SD-34]** #00010 NIST SP 800-57 Recommendation for Key Management – Part 1: General

3.18 **[X9 Registry SD-34]** #00011 FIPS 202 SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

3.19 **[X9 Registry SD-34]** #00013 NIST SP 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation

3.20 **[X9 Registry SD-34]** #00014 FIPS 203 Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)

3.21 **[X9 Registry SD-34]** #00015 FIPS 204 Module-Lattice-Based Digital Signature Standard (ML-DSA)

4. Terms and definitions

4.1 QR Code (Quick Response Code)
A two-dimensional barcode that encodes information such as text, URLs, contact details, or location data. The encoded content can be read by any device with a 2D scanner or camera equipped with decoding software, such as a smartphone.

4.2 Payment QR Code
A QR Code that encodes the data needed to initiate a financial transaction, such as payment amount, Payee details, and due date. When scanned, it triggers a process allowing the Payer to review and confirm the payment without sharing personal account information.

4.3 URL QR Code
A QR Code that encodes a web address (URL), directing the user to a specific online resource when scanned. Commonly used to share static content like menus, product pages, or websites, the same URL QR Code will redirect all users to the same destination.

4.4 Merchant-presented QR Code
A QR Code displayed by a Payee (e.g. merchant or biller) which a Payer (e.g. consumer) scans using a mobile device to initiate a payment.

4.5 Dynamic QR Code
A single-use, merchant-presented QR Code tied to a specific payment transaction. It cannot be reused once paid. Despite the name, a dynamic QR Code may appear static (e.g., printed), but it is considered dynamic because it links to a unique, one-time payment.

4.6 QR Code Content
A text string that contains minimal transaction details such as merchant name, city and other identifiers. The QR Code content is what gets encoded into the QR Code image that a Payer scans.

4.7 QR Code Image
A machine-readable graphic generated from the QR Code data content. The QR Code content is passed to a QR Code library, which generates the QR Code image pattern.

4.8 Payment Payload
A set of payment-related data referenced by a QR Code but stored externally. It includes dynamic details like amount, due date, Payer/Payee info, or account data. Storing this data outside the QR Code keeps it easy to scan and allows updates - like discounts or late fees - based on when the QR Code is scanned.

4.9 Payment URL

The encoded web address inside a QR Code that, when accessed, returns the payment payload with transaction details.

4.10 Payee
The individual or entity that is receiving funds from a customer (Payer). In a QR Code transaction, the Payee presents the QR Code to the Payer to be scanned and paid.

4.11 Payer
The individual or entity that initiates a payment by scanning a QR Code and authorizing a transfer of funds to the Payee's account.

4.12 Payment Service Provider (PSP)
Any entity involved in generating or decoding the QR Code and facilitating the exchange of information to enable money movement. For example, this could be a bank, credit union, mobile banking provider, merchant, merchant service provider, agency of a bank, fintech, third party etc.

4.13 Payee PSP
A Payment Service Provider that supports the Payee by generating the QR Code, hosting the Payload, and receiving the payment. This entity ensures the transaction data is accessible and compliant.

4.14 Payer PSP
A Payment Service Provider that supports the Payer by enabling QR Code scanning, retrieving the Payload, validating security elements, and initiating the payment to the Payee's account via the chosen payment network.

4.15 Payment Payload Request
An HTTP request sent by the Payer's PSP to the Payment URL contained in the QR Code. The request retrieves the associated Payment Payload.

4.16 Payment Payload Response
The HTTP response returned by the Payee's PSP to a Payload Request. It contains the Payload (JSON) along with any applicable metadata, such as digital signatures, timestamps, or error codes.

4.17 Invoicee
The individual or entity legally responsible for paying an amount due, such as a bill, loan, or invoice. While the Invoicee and Payer are often the same, they may differ—for example, a child (invoicee) responsible for tuition that is paid by a parent (Payer). The Invoicee is the party obligated to settle the debt, regardless of who actually makes the payment.

4.18 Creditor
The individual or entity legally entitled to receive payment. This is typically the merchant, biller, or service provider named in the transaction. The Creditor's financial account is the designated destination for funds sent by the Payer.

4.19 Ultimate Creditor
An optional party that is the final beneficiary of the funds, distinct from the initial receiving account. The Ultimate Creditor may be identified when an intermediary account is used—such as when a payment processor or agent receives funds on behalf of the actual merchant or biller.

4.20 Instant Payment
A payment that is initiated, cleared, and settled in real-time or near real-time, typically within seconds, and available to the Payee immediately.

4.21 FedNow®
A real-time payment system operated by the U.S. Federal Reserve, enabling instant account-to-account transfers between financial institutions.

4.22 RTP® (Real-Time Payments)

A real-time payment network operated by The Clearing House (TCH) that allows financial institutions to process instant payments between bank accounts**.**

4.23 ISO (International Organization for Standardization)

An organization that coordinates the creation and publication of International Standards.

4.24 ISO 20022

A collection of 8 (soon to be 9) documents that are the International Standards that describe the processes to create, register and maintain messages for the Financial Community.

4.25 pacs.008 (FItoFICustomerCreditTransfer)

The FI-to-FI Customer Credit Transfer message is part of the ISO 20022 message catalog, maintained by the ISO 20022 Registration Authority. It is used when a Payer’s PSP initiates a customer credit transfer on networks such as FedNow or RTP. The pacs.008 carries debtor/creditor details, account identifiers, amount/currency, and remittance information.

4.26 pacs.002 (FItoFIPaymentStatusReport)

The FI-to-FI Payment Status Report message is part of the ISO 20022 message catalog, maintained by the ISO 20022 Registration Authority. It is used to provide status information on a previously sent payment instruction. On FedNow, a pacs.002 "acknowledgement” goes to the Payer’s PSP and a pacs.002 "advice of credit” goes to the Payee’s PSP. On RTP, pacs.002 provides immediate status updates to both parties.

4.27 Push to Card

A type of payment where funds are sent directly to a Payee’s debit or prepaid card using the card’s credentials (typically the PAN). Unlike traditional card payments, which are pull-based, push-to-card transactions are initiated by the sender and use card networks to deliver funds quickly—often in real time or within minutes.

4.28 ACH Credit Push

An ACH payment initiated by the Payer to send funds to the Payee. Unlike pull-based debits, it's authorized by the Payer. A common example is payroll direct deposit.

4.29 Zelle

An alias-based payment network owned by seven major U.S. banks through Early Warning Services and used by participating financial institutions to send and receive payments using directory identifiers (e.g., email, phone number, tag).

4.30 Payment Notification

A standardized message sent from the Payer’s PSP to the Payee’s PSP after a payment has been initiated. Its purpose is to communicate minimal yet essential information about the payment for the Payee’s PSP to support downstream reconciliation, fulfillment, or customer messaging.

4.31 Payment Application

Refers to the software application that passes the QR Code data and performs the functional requirements of the X9.150 standard. This Payment Application is owned by the Wallet Provider (i.e. an issuer banking application or Digital Wallet proprietary application).

4.32 Interoperability

The ability of different financial institutions, payment networks, and technology providers to process QR Code-initiated payments using a common standard.

4.33 NFC

Near field communication, a wireless technology that allows devices to exchange information when they are close together. It is a form of contactless, close proximity, radio communications based on radio-frequency identification (RFID) technology.

4.34 UTC Timestamp
A date and time formatted according to RFC 3339 and expressed in Coordinated Universal Time (UTC).

5. Payment QR Code Overview

A QR Code is a two-dimensional version of a barcode that can encode structured data in machine-readable form and holds significantly more information. It can be scanned using an image-based reader - typically a mobile phone camera - to quickly access the encoded data. A Payment QR Code, as defined in this standard, is a merchant-presented, dynamically generated QR Code used to initiate push payments.

Most QR Codes in the U.S. are URL QR Codes. They are used to share static information like a menu or product page. Payment QR Codes enable financial transactions, allowing money to move directly from one account to another without sharing bank or payment account details.

Unlike URL QR Codes, Payment QR Codes must be scanned from within an authenticated mobile app (e.g., a bank, credit union, or merchant wallet) where the user has logged in and the application itself is recognized as an authorized client. This ensures that both the Payer and the application making the request are trusted before the protected payment payload can be accessed.

The QR Code is separate from the payment network. It’s like a payment request that contains key details, like the Payee’s account, amount, and purpose, that enable payments to be initiated over various networks.

Each payment QR Code includes two linked data sets:
- The QR Code Content with basic information
- The Payment Payload containing full payment details

These data sets are connected by a URL in a field of the QR Code, which points to a secure HTTPS endpoint. When this endpoint is accessed by a payment application, it returns a Payment Payload via JSON with complete payment information.

Once generated, the QR Code content is passed to a QR Code library, which generates the QR Code image pattern. There are many widely available open-source and commercial QR Code libraries. This standard does not require or endorse any specific implementation.
Merchant-presented QR Codes are shown by the Payee (e.g. merchant/biller) and scanned by the Payer.

Dynamic QR Codes are single-use and can be used in-store, online, or for bills. They may include logic for things like discounts or late fees, for example, where a QR Code scanned after a bill’s due date can automatically add a late fee.

Example process flow:

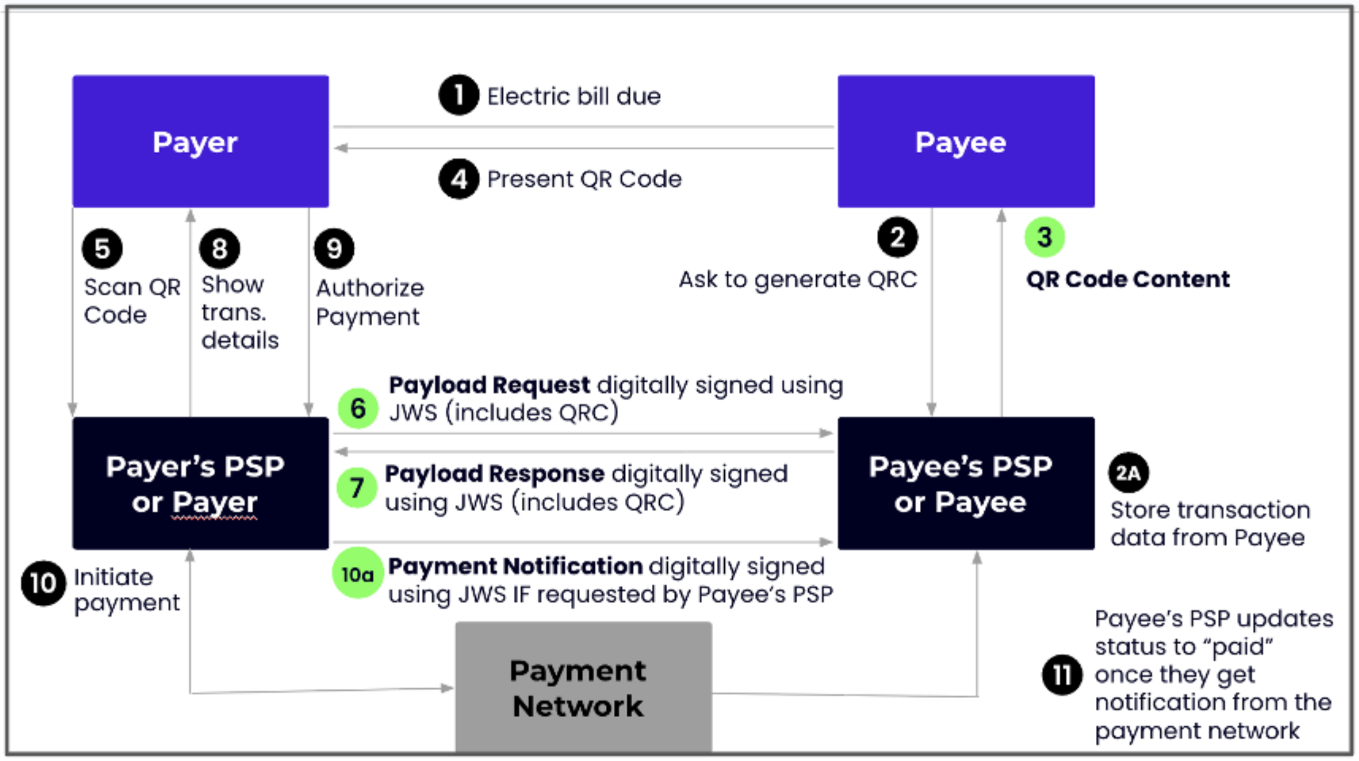
The following diagram demonstrates the typical process flow of a merchant-presented dynamically generated payment QR Code for a biller (flow is the same for a merchant).

This X9.150 Payment QR Code Standard is focused on steps 3, 6, 7 and 10a in the diagram below:

- 3: QR Code content and requirements

- 6: Security related to the Payment Payload request
- 7: Payment Payload content and requirements (and security related to Payment Payload response and QR Code)
- 10a: Payment Notification content and requirements (and security related to this exchange)

Figure 1 - Biller Example


Payer
Payee
1 Electric bill due
4 Present QR Code
5 Scan QR Code
8 Show trans. details
9 Authorize Payment
2 Ask to generate QRC
3 QR Code Content
6 Payload Request digitally signed using JWS (includes QRC)
7 Payload Response digitally signed using JWS (includes QRC)
10a Payment Notification digitally signed using JWS IF requested by Payee's PSP
Payer's PSP or Payer
Payee's PSP or Payee
2A Store transaction data from Payee
10 Initiate payment
Payment Network
11 Payee's PSP updates status to "paid" once they get notification from the payment network

Steps in QR Code Payment – Biller Example

1. **Billing**: Consumer (e.g. Payer) receives an electricity bill after 30 days of usage.

2. **QR Code Request**: Biller (Payee) asks their PSP to generate a QR Code for that specific bill, passing details like amount due, due date, and invoice info.

3. **QR Code Generation**: The Payee's PSP stores the data securely, generates a QR Code per the X9.150 standard. How this content is delivered to the Payee (e.g. as QR Code content to be rendered locally or as an already-rendered image) is implementation specific and out of scope of this specification.

4. **Presentation**: Payee presents the QR Code image to the Payer.

5. **Scan & Authentication**: Payer authenticates (e.g., via login, biometrics etc.) within a mobile app (e.g., bank app or digital wallet) and scans the QR Code.

6. **Payload Request**: The app decodes the QR Code and uses the URL in field 26.01 to request the payload from the Payee's PSP, exchanging digital signatures as defined in the standard.

7. **Payload Response**: After the Payee's PSP validates the Payer PSP's digital signature, the payload is securely delivered to the Payer's PSP. Digital signatures are included in this response to be validated by Payer's PSP.

8. **Display to Payer**: Bill details (e.g. biller name, amount due, fees) are shown to Payer in the mobile app

9. **Authorization**: Payer reviews and approves the payment.

10. **Payment Initiation**: Payer's PSP initiates the payment using any of the payment methods indicated by the Payee's PSP.

10a. **Payment Notification**: IF requested by the Payee's PSP, the Payer's PSP sends payment notification exchanging digital signatures as defined in the standard.

11. **Status Update**: The Payee's PSP updates the status of the QR Code to paid once settled.

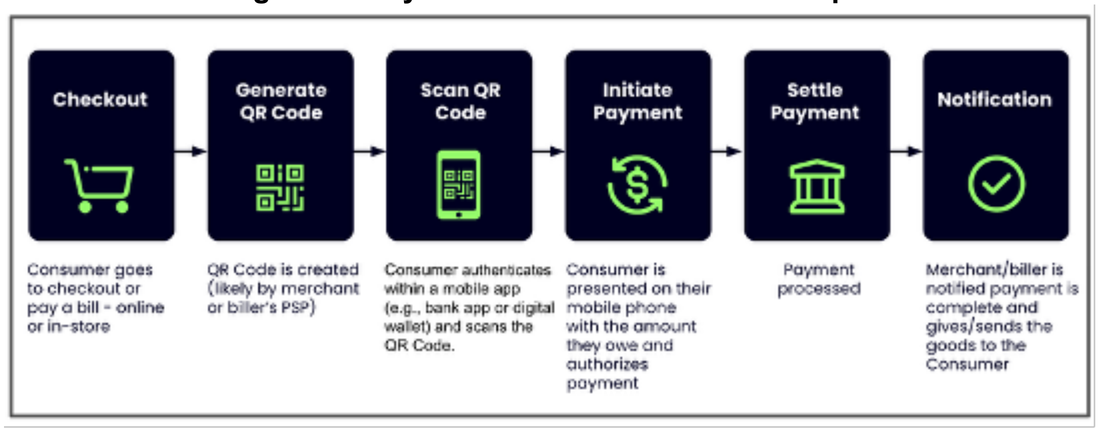
Note: The QR Code and Payload may be created and hosted by any PSP with an X9 digital certificate - typically the biller, their bank or 3rd party processor. Similar to an acquirer in card payments.

Merchant-presented dynamically generated payment QR Codes can be printed on bills, displayed in-store at POS or displayed at online checkout.

Example use cases include payments related to consumer bill pay, e-commerce, in-store retail transactions, events, gas stations, freight delivery and donations just to name a few.

Payment QR Codes enable push payments by embedding the Payee's financial account details. The Payer logs into a payment app, scans the QR Code, reviews the amount, and confirms. The Payer's PSP retrieves the Payee's account info from the QR Code, combines it with the Payer's account details, along with other information needed to initiate payment.

Figure 2 - Payment QR Code Process Example


Checkout
Generate QR Code
Scan QR Code
Initiate Payment
Settle Payment
Notification
Consumer goes to checkout or pay a bill - online or in-store
QR Code is created (likely by merchant or biller's PSP)
Consumer authenticates within a mobile app (e.g., bank app or digital wallet) and scans the QR Code.
Consumer is presented on their mobile phone with the amount they owe and authorizes payment
Payment processed
Merchant/biller is notified payment is complete and gives/sends the goods to the Consumer

Payees can include extra information in the QR Code as needed. For example, an EBT merchant might list purchased items to confirm eligibility, while a utility biller might include a customer ID.

The X9 Payment QR Code is payment network-agnostic - it can be implemented with any payment system, including, but not limited to FedNow, RTP, ACH credit push, push-to-card, and Zelle networks or any number of private or emerging networks. The payment method is selected automatically by the Payer's app, following the merchant's pre-set payment network preferences. (See Annex A.5 for suggested market practices).

While most interactions begin by scanning a QR Code, the same process can be triggered by tapping an NFC tag, clicking a link, in-app communication or future methods like Bluetooth or sound.

6. QR Code

6.1 General Overview of QR Code

The QR Code Content defined herein is based on, and fully compliant with, the *EMV QR Code Merchant Presented Mode Specification* as published by EMVCo. All data elements, structure, and encoding follow the requirements outlined in that specification to ensure interoperability and adherence to global standards. This chapter describes the QR Code Content format to aid reader comprehension and further defines the specific fields that are used for the X9 QR Code and the contents of those fields.

The QR Code itself contains a compact alphanumeric string (not raw bytes) structured using the Tag-Length-Value (TLV) format. Each data item is tagged, sized, and followed by its value (e.g., 53-03-840). It includes mandatory data sets - like the QR Code version, type of QR Code, merchant details, currency, and amount to ensure compatibility and easy scanning.

> *Note: How to Read This Specification*
>
> *Data-element tables indicate each field as M, C, or O. Mandatory (M) fields are always present. Conditional (C) fields are required only when a stated condition applies. Optional (O) fields may be included but are not required.*
>
> *Processing behavior uses requirement keywords per BCP 14 (RFC 2119/RFC 8174). SHALL denotes a requirement; SHOULD denotes a recommendation; MAY denotes optional behavior.*
>
> *When an optional capability is implemented - for example, tips - its related fields SHOULD follow this specification's format and validations. Conditional rules are expressed as: "If <condition> applies, the element SHALL …; otherwise it MAY be omitted."*

6.2 QR Code Requirements

The following data objects **SHALL** be present in the merchant-presented QR code to be parsed by the QR Code reading application.

[00] Payload format indicator
[01] Point of initiation method
[26] Merchant account information
[52] Merchant category code
[53] Transaction currency
[54] Transaction amount
[58] Country code
[59] Merchant name
[60] Merchant city
[63] CRC

While all fields are required, most are informational. The following fields are repeated in the Payload - [52], [53], [54], [58], [59], [60]. For these fields, payment applications **SHOULD** rely on the Payload, not the QR Code.

In the event of any inconsistency between information present in the QR Code and the Payment Payload, the Payload **SHALL** take precedence.

Data in field 26 is not repeated in the Payload. It is essential and indicates the globally unique identifier is org.x9 and includes the Payment Payload URL which tells the Payer's PSP where to retrieve the secure, current payment payload.

The following table describes the format requirements.

Table 1 - QR Code Format Requirements

ID	Sub-ID	Field Name	M, C, O	Max Length	Format
00		Payload format indicator	M	02	Numeric Fixed length = 2
01		Point of initiation method	M	02	Numeric Fixed length = 2
26		Merchant account information	M	83	Template Nested subfields
	00	Globally unique identifier	M	06	Alphanumeric Special
	01	Dynamic payment URL	M	77	String
52		Merchant category code	M	04	Numeric Fixed length = 4
53		Transaction currency code	M	03	Numeric Fixed length = 3
54		Transaction amount	M	13	Alphanumeric Special Variable length
58		Country code	M	02	Alphanumeric Special Fixed Length = 2
59		Merchant name	M	15	Alphanumeric Special Variable
60		Merchant city	M	25	Alphanumeric Special Variable
63		CRC checksum	M	04	Alphanumeric Special Fixed Length = 4

Note: The QR Code Content may contain fields in addition to the fields specified above. Applications parsing and reading the QR Code Content must be able to handle the presence of additional fields that may be present beyond the minimum defined above. The application should ignore fields it does not use without error, to ensure forward compatibility and interoperability.

00 - Payload format indicator

SHALL be present and set to 01.

Indicates the version of the EMVCo QR Code specification.

01 - Point of initiation method

SHALL be present and set to 12.

Indicates "dynamic" QR Code.

26 - Merchant account information

Length **SHALL** equal the combined length of all subfields present within the Template.

Example: 26330006org.x90127pay.examplebank.com/txn/123456

26 33 → ID = 26, length = 33

0006org.x9 → SubID 00, length = 06, value = org.x9

0127pay.examplebank.com/txn/123456 → SubID 01, length = 27, value = pay.examplebank.com/txn/123456

Note: The EMV Merchant Present Mode template allows up to 99 characters for Merchant Account Information which use IDs 02–51. The X9.150 profile does not define these additional EMV Merchant Account Information fields, but a QR Code MAY include fields beyond those listed in this table. For Field 26 specifically, the maximum length is determined by the defined subfields (currently 6 + 77 = 83).

26.00 - Globally unique identifier

SHALL be set to org.x9 to indicate conformance with this standard, X9.150.

26.01 - Dynamic payment URL

SHALL contain the host and path of an HTTPS endpoint.

SHALL NOT include the https:// prefix.

The full URL, when resolved, **SHALL** use the HTTPS protocol and point to a secure Payload endpoint as defined in Section 7.

Note: When calling this endpoint, it is recommended to wait at least 3 seconds, but not more than 6 for a response.

Example: pay.examplebank.com/txn/123456

52 - Merchant category code

SHALL be a 4-digit MCC as defined in ISO 18245.

53 - Transaction currency code

SHALL be ISO 4217 numeric.

54 - Transaction amount

SHALL represent the amount in decimal format.

Note: This field **MAY** be 0.00 since there are multiple amount fields in the payload due to late fees and discounts. PSP's are recommended to use the amountDue field in the payload as the authoritative amount.

Example: 0.00

58 - Country code

SHALL be a two-letter country code as defined by ISO 3166-1 alpha-2.

Example: US

59 - Merchant name

SHALL contain a valid and recognizable name for the merchant or biller.

Example: Acme Shoes

60 - Merchant city

SHALL contain the city associated with the merchant or biller.

Example: Chicago

63 - CRC checksum

SHALL be calculated according to [ISO/IEC 13239] using the polynomial ‘1021’ (hex) and initial value ‘FFFF’ (hex). The data over which the checksum is calculated shall cover all data objects, including their ID, Length and Value, to be included in the QR Code, in their respective order, as well as the ID and Length of the CRC itself (but excluding its value).

7. JSON Data Type

The JSON data type defined below specifies the data types and formats used by this standard for the Payment Payload and the Payment Notification for request and response. These shared definitions establish consistent representations for common elements throughout this standard. Field-level requirements reference these types directly.

Table 2 - JSON Data Type

Data Type	Format Requirements
UTC Timestamp	Conforms to ISO 8601 standard Value **SHALL** end with Z and follow YYYY-MM-DDThh:mm:ss[.fff]Z Fractional seconds **MAY** be present (1–3 digits) Examples with and without fractional seconds: "2025-09-30T18:04:00Z", "2025-09-30T18:04:00.123Z"
Date	Conforms to ISO 8601 standard Value **SHALL** contain the simple date format: YYYY-MM-DD Example: "2025-04-30”

Data Type	Format Requirements
String	**SHALL** primarily use Printable ASCII characters, including: ● Letters (A–Z, a–z) ● Numbers (0–9) ● Common Symbols (-, .), and (@) ● No other special characters are permitted unless required by the business use case and allowed by the intermediary Example: "Amazon”
Amount (Minor Units)	**Shall** be a 64 bit integer (see specific field-level requirements on whether can be negative) **Shall** be represented as an integer in the smallest currency unit In order to interpret the value, it’s necessary to know the number of decimal places defined for the specific currency being used.(e.g. cents for USD) Example: 19825 = $198.25
Account Number (Bank)	**SHALL** be a string with a minimum of 4 and a maximum of 17 characters **SHALL** consist only of digits (0–9) AND/OR letters (A–Z, a–z) Value **SHALL** represent the unique primary account identifier for the receiving party at the settlement institution **MAY** be the account number of the creditor OR ultimate creditor **SHALL** be protected - either by tokenization or by encryption using an X9-approved key management methodology as defined in the normative references. ● The encrypted account number results in a ciphertext that **SHALL** be encoded in base64url. Example: "12345678987654321”
Routing Number (ABA RTN)	**SHALL** be a String with exactly 9 digits (0 - 9) Value **SHALL** represent a valid Routing Transit Number (RTN), which uniquely identifies a financial institution for clearing and settlement in the US **MAY** be the account number of the creditor OR ultimate creditor Example: "123456789"

8. Payment Payload

8.1 General Overview of Payment Payload

The payment payload is the set of data elements that the Payer’s PSP requests from the Payee’s PSP after scanning the QR Code.

The payment payload is formatted in JSON and included in the response from the Payee’s PSP using JWS for security (see Section 10 for more details).

8.2 Payment Payload Request

In order to request the payment payload, call the URL indicated in field 26.01 of the QR Code.

Use the POST method including the QR Code content in the body in the JSON format described below.

{"qrCodeContent”:”QRC content itself in base64URL”}

Example:

{"qrCodeContent”:”
MDAwMjAxMDEwMjEyMjY3MzAxNjlodHRwczovL3BnZS1wYXltZW50cy5leGFtcGxlLmNvbS9xc
mMvMTIzRTQ1NjdFODlCMTJEM0E0NTY0MjY2MTQxNzQwMDA1MjA0NDkwMDUzMDM4NDA
1NDA0MC4wMDU4MDJVUzU5MTNTQU4gRlJBTkNJU0NPNjAzMlBBQ0lGSUMgR0FTIEFORCB
FTEVDVFJJQyBDT01QQU5ZNjMwNDY3RkE=”}

Note: Although GET is typically used, HTTP header size limitations combined with the larger size of PQC digital signatures require the use of POST.

8.3 Payment Payload Response

The response to this payment payload request is the JSON containing the Payment Payload data using JWS for security (see Section 10 for more details).

This payload supports a wide range of payment-related information. This includes:

- **QR Code metadata**: Unique payload ID, timestamps, and the QR code contents. (Data fields in orange in section 7.2).
- **Payee details**: Merchant or biller name, contact info, address, and merchant category code. (Data fields in yellow in Section 7.2)
- **Additional data**: Optional fields for reconciliation, key-value pairs, and unstructured information. (Data fields in blue in Section 7.2)
- **Invoice or order details**: Invoice/order numbers, descriptions, due dates, and Payer (invoicee) information. (Data fields in green in Section 7.2)
- **Payment terms**: Amount due, currency, adjustments, discounts, and tipping options. (Data fields in green in Section 7.2)
- **Payment methods**: Supported networks (e.g., FedNow, RTP, ACH credit push, push-to-card, Zelle), associated account credentials, and whether the amount is editable. (Data fields in gray in Section 7.2)

More than one payment network may be included in the payload (FedNow, RTP, ACH, push-to-card, Zelle).

8.4 Payment Payload Requirements

The following table describes the format requirements of each data field in the Payment Payload.

Payment Payload **SHALL** be formatted as a JSON object and encoded in UTF-8.

Table 3 - Payment Payload Format Requirements

Payment Payload ID	Field Name	M,C,O	Type	Max Length	Format	JSON Path
1	Payload id	M	Field	32	String	$.id
2	Revision	M	Field	2	Integer	$.revision
3	QR Code Content	M	Field	1024	String	$.qrCodeContent
4	Created At	M	Field	24	UTC Timestamp	$.createdAt
5	Revised At	M	Field	24	UTC Timestamp	$.revisedAt
6	Sent At	M	Field	24	UTC Timestamp	$.sentAt
7	Valid Until	M	Field	24	UTC Timestamp	$.validUntil
8	Status	M	Field	NA	String (enum)	$.status
9	Payment Notification	O	Field	256	String (HTTPS URL)	$.paymentNotification
10	Creditor	M	Object	-	-	$.creditor
10.1	Creditor Name	M	Field	50	String	$.creditor.name
10.2	Creditor Phone	O	Field	20	String (phone)	$.creditor.phone
10.3	Creditor Email	O	Field	254	String (email)	$.creditor.email
10.4	Creditor Address	M	Object	-	-	$.creditor.address
10.4.1	Creditor Address Line 1	O	Field	60	String	$.creditor.address.addressLine1
10.4.2	Creditor Address Line 2	O	Field	60	String	$.creditor.address.addressLine2
10.4.3	Creditor City	M	Field	40	String	$.creditor.address.city
10.4.4	Creditor State	O	Field	30	String	$.creditor.address.state

Payment Payload ID	Field Name	M,C,O	Type	Max Length	Format	JSON Path
10.4.5	Creditor Postal Code	O	Field	20	String	$.creditor.address.postalCode
10.4.6	Creditor Country	M	Field	2	String	$.creditor.address.country
10.5	Merchant Category Code	M	Field	4	String	$.MCC
10.6	Ultimate Creditor	O	Object	-	-	$.ultimateCreditor
10.6.1	Ultimate Creditor Account	C	Object	-	-	$.ultimateCreditor.account
10.6.1.1	Ultimate Creditor Account id	C	Field	256	String	$.ultimateCreditor.account.id
10.6.1.2	Ultimate Creditor Account Schema Name	C	Field	70	String	$.ultimateCreditor.account.schemaName
10.6.2	Ultimate Creditor Name	C	Field	50	String	$.ultimateCreditor.name
10.6.3	Ultimate Creditor address	C	Object	-	-	$.ultimateCreditor.address
10.6.3.1	Ultimate Creditor Address line 1	O	Field	60	String	$.ultimateCreditor.address.addressLine1
10.6.3.2	Ultimate Creditor Address line 2	O	Field	60	String	$.ultimateCreditor.address.addressLine2
10.6.3.3	Ultimate Creditor City	C	Field	40	String	$.ultimateCreditor.address.city
10.6.3.4	Ultimate Creditor State	O	Field	30	String	$.ultimateCreditor.address.state
10.6.3.5	Ultimate Creditor Postal Code	O	Field	20	String	$.ultimateCreditor.address.postalCode
10.6.3.6	Ultimate Creditor Country	C	Field	2	String	$.ultimateCreditor.address.country
11	Unstructured	O	Field	140	String	$.unstructured
12	Additional Information	O	Object	-	-	$.additionalInformation
12.1	Key	C	Field	30	String	$.additionalInformation.key
12.2	Value	C	Field	218	String	$.additionalInformation.value
13	Bill	M	Object	-	-	$.bill
13.1	Bill Payment Timing	M	Field	-	String (enum)	$.bill.paymentTiming
13.2	Bill Description	O	Field	100	String	$.bill.description
13.3	Bill Order	O	Object	-	-	$.bill.order
13.3.1	Bill Order Number	O	Field	50	String	$.bill.order.number
13.3.2	Bill Order Date	O	Field	10	Date	$.bill.order.date
13.4	Bill Invoice	O	Object	-	-	$.bill.invoice
13.4.1	Bill Invoice Number	O	Field	50	String	$.bill.invoice.number

Payment Payload ID	Field Name	M,C,O	Type	Max Length	Format	JSON Path
13.4.2	Bill Invoice Date	O	Field	10	Date	$.bill.invoice.date
13.4.3	Bill Invoice Due Date	C	Field	24	UTC Timestamp	$.bill.invoice.dueDate
13.4.4	Invoicee	O	Object	-	-	$.bill.invoice.invoicee
13.4.4.1	Invoicee Name	O	Field	50	String	$.bill.invoice.invoicee.name
13.4.4.2	Invoicee Phone	O	Field	20	String (phone)	$.bill.invoice.invoicee.phone
13.4.4.3	Invoicee Email	O	Field	254	String (email)	$.bill.invoice.invoicee.email
13.4.4.4	Invoicee Address	O	Object	-	-	$.bill.invoice.invoicee.address
13.4.4.4.1	Invoicee Address Line 1	O	Field	60	String	$.bill.invoice.invoicee.address.addressLine1
13.4.4.4.2	Invoicee Address Line 2	O	Field	60	String	$.bill.invoice.invoicee.address.addressLine2
13.4.4.4.3	Invoicee Address City	O	Field	40	String	$.bill.invoice.invoicee.address.city
13.4.4.4.4	Invoicee Address State	O	Field	30	String	$.bill.invoice.invoicee.address.state
13.4.4.4.5	Invoicee Address Postal Code	O	Field	20	String	$.bill.invoice.invoicee.address.postalCode
13.4.4.4.6	Invoicee Country	O	Field	2	String	$.bill.invoice.invoicee.address.country
13.5	Bill Amount Due	M	Object	-	-	$.bill.amountDue
13.5.1	Bill Amount	M	Field	18	Integer	$.bill.amountDue.amount
13.5.2	Bill Currency	M	Field	32	String	$.bill.amountDue.currency
13.5.3	Bill Adjustment	O	Object	-	-	$.bill.amountDue.adjustment
13.5.3.1	Bill Adjustment Explanation	C	Field	200	String	$.bill.amountDue.adjustment.explanation
13.5.3.2	Bill Adjustment Amount	C	Field	18	Integer	$.bill.amountDue.adjustment.amount
13.5.3.3	Bill Adjustment Valid Until	C	Field	24	UTC Timestamp	$.bill.amountDue.adjustment.validUntil
13.6	Bill Tip	M	Object	-	-	$.bill.tip
13.6.1	Bill Tip Allowed	M	Field	5	Boolean	$.bill.tip.allowed
13.6.2	Bill Tip Range	O	Object	-	-	$.bill.tip.range
13.6.2.1	Bill Tip Min	C	Field	3	Integer	$.bill.tip.range.min
13.6.2.2	Bill Tip Max	C	Field	3	Integer	$.bill.tip.range.max
13.6.3	Bill Tip Presets	O	Field	-	Array	$.bill.tip.presets

Payment Payload ID	Field Name	M,C,O	Type	Max Length	Format	JSON Path
14	Payment Methods	M	Object	-	-	$.paymentMethods
14.1	Payment Methods Currency	M	Field	32	String	$.paymentMethods.currency
14.2	Payment Methods Valid Until	M	Field	24	UTC Timestamp	$.paymentMethods.validUntil
14.3	Payment Methods Amount	M	Field	18	Integer	$.paymentMethods.amount
14.4	Payment Methods Editable	O	Object	-	-	$.paymentMethods.editable
14.4.1	Payment Methods Editable Range	C	Object	-	-	$.paymentMethods.editable.range
14.4.1.1	Payment Methods Editable Min	C	Field	18	Integer	$.paymentMethods.editable.range.min
14.4.1.2	Payment Methods Editable Max	C	Field	18	Integer	$.paymentMethods.editable.range.max
14.5	Payment Methods Network	M	Object	-	-	$.paymentMethods.network
14.5.1	FedNow	O	Object	-	-	$.paymentMethods.network.fednow
14.5.1.1	FedNow Routing Number	C	Field	9	String	$.paymentMethods.network.fednow.routingNumber
14.5.1.2	FedNow Account Number	C	Field	17	String	$.paymentMethods.network.fednow.accountNumber
14.5.1.3	FedNow Protection Type	C	Field	-	String (enum)	$.paymentMethods.network.fednow.protectionType
14.5.2	RTP	O	Object	-	-	$.paymentMethods.network.rtp
14.5.2.1	RTP Routing Number	C	Field	9	String	$.paymentMethods.network.rtp.routingNumber
14.5.2.2	RTP Account Number	C	Field	17	String	$.paymentMethods.network.rtp.accountNumber
14.5.2.3	RTP Protection Type	C	Field	-	String (enum)	$.paymentMethods.network.rtp.protectionType
14.5.3	ACH	O	Object	-	-	$.paymentMethods.network.ach
14.5.3.1	ACH Routing Number	C	Field	9	String	$.paymentMethods.network.ach.routingNumber
14.5.3.2	ACH Account Number	C	Field	17	String	$.paymentMethods.network.ach.accountNumber
14.5.3.3	ACH Protection Type	C	Field	-	String (enum)	$.paymentMethods.network.ach.protectionType
14.5.4	Zelle	O	-	-	-	$.zelle

Payment Payload ID	Field Name	M,C,O	Type	Max Length	Format	JSON Path
14.5.5	Visa	O	-	-	-	$.visa
14.5.6	Mastercard	O	-	-	-	$.mastercard
14.5.7	American Express	O	-	-	-	$.americanExpress
14.5.8	Discover	O	-	-	-	$.discover

1 - Payload ID ($.id)

SHALL be a UUID string consisting of exactly 32 hexadecimal characters (`[0-9A-Fa-f]`), with no separators.

SHALL remain unchanged across revisions and **SHALL** uniquely identify the payload.

Identifier assigned when the Payment Payload (JSON) is created.

Example: 123e4567e89B12d3a456426614174000

Note: This field may be used as a reference identifier in underlying payment messages. See Annex A.6 and A.7 for example mappings to specific payment networks.

2 - Revision ($.revision)

SHALL be an integer in the range 0–99.

When the id is first generated, revision **SHALL** be 0.

Each permitted update **SHALL** increment revision by exactly 1.

A payload with status = "ACTIVE" **MAY** be revised.

Once the payload status is "PAYMENT_INITIATED", "PAID", or "CANCELLED", it **SHALL NOT** be revised.

Indicates whether the payload has been updated after creation.

3 - QR Code Content ($.qrCodeContent)

SHALL be present in the payment payload.

Contains the QR Code content (alphanumeric string), encoded in base64URL with maximum length of 1024.

Example QR Code Content: "00020101021226730169pge-payments.example.com/qrc/
123E4567E89B12D3A456426614174000520449005303840540 40.005802US5913SAN
FRANCISCO6032PACIFIC GAS AND ELECTRIC COMPANY630467FA”

Example Base64URL of the QR Code Content above:
"4oCcMDAwMjAxMDEwMjEyMjY3MzAxNjlwZ2UtcGF5bWVudHMuZXhhbXBsZS5jb20vcXJjLzEyM0U
0NTY3RTg5QjEyRDNBNDU2NDI2NjE0MTc0MDAwNTIwNDQ5MDA1MzAzODQwNTQwNDAuMDA1
ODAyVVM1OTEzU0FOIEZSQU5DSVNDTzYwMzJQQUNJRklDIEdBUyBBTkQgRUxFQ1RSSUMgQ0

9NUEFOWTYzMDQ2N0ZB4oCd”

4 - Created At ($.createdAt)

SHALL be a UTC timestamp (see Table 2 - JSON Data Type).

For the initial version (revision = 0), createdAt **SHALL** equal revisedAt

In all cases, createdAt **SHALL** be less than or equal to revisedAt and sentAt.

Marks the date/time when the Payment Payload was created.

5 - Revised At ($.revisedAt)

SHALL be a UTC timestamp (see Table 2 - JSON Data Type).

For the initial version (revision = 0), revisedAt **SHALL** equal createdAt.

For all subsequent revisions (revision > 0), revisedAt **SHALL** be greater than or equal to createdAt.

Marks the time of the most recent modification to the payload.

6 - Sent At ($.sentAt)

SHALL be a UTC timestamp (see Table 2 - JSON Data Type).

sentAt **SHALL** be greater than or equal to revisedAt, ensuring the payload is not considered sent before its most recent update.

Marks the time when the Payload was sent to the Payer’s PSP.

7 - Valid Until ($.validUntil)

SHALL be a UTC timestamp (see Table 2 - JSON Data Type).

SHALL be later than revisedAt and **SHALL** be greater than or equal to sentAt.

Last date/time Payload is valid for payment.

See Annex A.8 for suggested market practices for implementation.

8 - Status ($.status)

SHALL be present.

On Payload creation, status **SHALL** be active.

The field **SHALL** contain only one of the following four exact values:

ACTIVE - payload is available for payment.

PAYMENT_INITIATED - payment has been initiated against the payload.

PAID - payload has been paid.

CANCELLED - payload has been cancelled.

Indicates the lifecycle state of the payload.

Example: "ACTIVE”

See Annex A.9 for suggested market practices for implementation.

9 - Payment Notification ($.paymentNotification)

Shall be a string containing a valid HTTPS URL with 256 maximum characters.

Shall adhere to the syntax dictated by RFC 3986.

If present, the Payer’s PSP **SHALL** POST a JSON notification to the specified URL.

If absent, the Payer’s PSP **SHALL NOT** generate or send a payment notification.

HTTPS endpoint provided by the Payee’s PSP for receiving payment notifications.

Example: "https://paymentNotification.BankABC.com/x9/api/notify”

See Section 9 - Payment Notification for more details.

10 - Creditor ($.creditor)

SHALL be present in the payment payload.

10.1 - Creditor Name ($.creditor.name)

SHALL be string with 50 maximum characters (see Table 2 - JSON Data Type).

10.2 - Creditor Phone ($.creditor.phone)

Shall be a string (phone number) that conforms to E.164 international numbering format.

Shall start with "+” followed by 1 to 15 digits where the first digit after "+” **SHALL NOT** be "0”.

Example: "+15552223333”

10.3 - Creditor Email ($.creditor.emai)

SHALL be a string (email) that conforms with RFC 5322.

Example: "billing@amazon.com”

10.4 - Creditor Address ($.creditor.address)

Object **SHALL** be present in the payment payload.

10.4.1 - Creditor Address Line 1 ($.creditor.address.addressLine1)

MAY be present.

If present, **SHALL** be a string with maximum characters of 60 (see Table 2 - JSON Data Type).

Example: "410 Terry Ave North"

10.4.2 - Creditor Address Line 2 ($.creditor.address.addressLine2)

MAY be present.

Shall be a string with maximum characters of 60 (see Table 2 - JSON Data Type).

Example: "Building 17, Suite 1701”

10.4.3 - Creditor City ($.creditor.address.city)

Shall be a string with maximum characters of 40 (see Table 2 - JSON Data Type).

Example: "Seattle”

10.4.4 - Creditor State ($.creditor.address.state)

MAY be present.

SHALL use the ISO 3166-2, alpha-2 state code where available.

If not available, **SHALL** be a string with 30 maximum characters.

Example: "WA”

10.4.5 - Creditor Postal Code ($.creditor.address.postalCode)

MAY be present.

String **SHALL** conform to ISO 19160-4, allowing for country-specific variations.

Allows 1 to 20 alphanumeric characters, spaces, or hyphens.

Example: "98109”

10.4.6 - Creditor Country ($.creditor.address.country)

SHALL be a two-letter country code as defined by ISO 3166-1 alpha-2.

SHALL be exactly 2 uppercase letters (A-Z).

Example: "US”

10.5 - Merchant Category Code ($.MCC)

SHALL be present and conform to the ISO 18245 Code standard.

SHALL be exactly 4 digits.

Example "5661”

10.6 - Ultimate Creditor ($.ultimateCreditor)

SHALL be present in the payment payload IF payment is first collected by an intermediary before funds are received by a final party.

10.6.1 - Ultimate Creditor Account ($.ultimateCreditor.account)

SHALL be present whenever an ultimateCreditor is identified.

10.6.1.1- Ultimate Creditor Account id ($.ultimateCreditor.account.id)

SHALL be present whenever an ultimateCreditor is identified.

SHALL be encoded as a string with 256 maximum characters (see Table 2 - JSON Data Type).

Value **SHALL** uniquely distinguish the ultimate creditor within the platform or intermediary's environment.

Example: "425-516-8008”

10.6.1.2 - Ultimate Creditor Schema Name ($.ultimateCreditor.account.schemaName)

SHALL be present whenever an ultimateCreditor is identified.

SHALL be a string with 70 maximum characters (see Table 2 - JSON Data Type).

Indicates the type of identifier carried in ultimateCreditor.accountId.

Example: "phone number”

10.6.2 - Ultimate Creditor Name ($.ultimateCreditor.name)

SHALL be present whenever ultimateCreditor is identified.

If present, **SHALL** be a string with 50 maximum characters (see Table 2 - JSON Data Type).

Example: "ABC Shoes”

10.6.3 Ultimate Creditor Address ($.ultimateCreditor.address)

SHALL be present whenever an ultimateCreditor is identified.

Information related to the address of the ultimate creditor.

10.6.3.1 - Ultimate Creditor Address Line 1 ($.ultimateCreditor.address.addressLine1)

MAY be present.

If present, **SHALL** be a string with maximum characters of 60. (see Table 2 - JSON Data Type).

Example: "123 Elm St”

10.6.3.2 - Ultimate Creditor Address Line 2 ($.ultimateCreditor.address.addressLine2)

MAY be present.

If present, **SHALL** be a string with maximum characters of 60 (see Table 2 - JSON Data Type).

Example: "Bldg 1, Suite 2”

10.6.3.3 - Ultimate Creditor City ($.ultimateCreditor.address.city)

SHALL be present whenever ultimateCreditor is identified.

Shall be a string with maximum characters of 40 (see Table 2 - JSON Data Type).

Example: "Seattle”

10.6.3.4 - Ultimate Creditor State ($.ultimateCreditor.address.state)

MAY be present.

If present, **SHALL** use the ISO 3166-2, alpha-2 state code where available.

If present, and ISO 3166-2 not available, **SHALL** be a string with 30 maximum characters.

Example: "WA”

10.6.3.5 - Ultimate Creditor Postal Code ($.ultimateCreditor.address.postalCode)

MAY be present.

String **SHALL** conform to ISO 19160-4, allowing for country-specific variations.

Allows 1 to 20 alphanumeric characters, spaces, or hyphens.

Example: "98109”

10.6.3.6 - Ultimate Creditor Country ($.ultimateCreditor.address.country)

SHALL be present whenever ultimateCreditor is identified.

SHALL be a two-letter country code as defined by ISO 3166-1 alpha-2.

SHALL be exactly 2 uppercase letters (A-Z).

Example: "US”

11 - Unstructured ($.unstructured)

MAY be present.

If present, **SHALL** be string with 50 maximum characters (see Table 2 - JSON Data Type).

Carries Payee-related reference information for reconciliation, reporting, or downstream processing.

Example: "ABC Shoes order #12345 via Amazon”

Note: This field may be used to carry additional remittance information in underlying payment messages. See Annex A.6 and A.7 for suggested market practices for implementation

12 - Additional Information ($.additionalInformation)

MAY be present.

There is no limit to the number of key/value pairs.

Includes key/value pairs the Payee wants to flow with the payment - possibly shown to Payer.

12.1 - Key ($.additionalInformation.key)

SHALL be present IF the Additional Information object is present.

If present, **SHALL** be string with 30 maximum characters (see Table 2 - JSON Data Type).

A unique identifier that specifies the meaning, purpose, or data element represented in the value field of the payment payload.

Example: "CustomerServiceRef"

12.2 - Value ($.additionalInformation.value)

SHALL be present IF the Additional Information object is present.

If present, **SHALL** be string with 218 maximum characters (see Table 2 - JSON Data Type).

The data associated with the key, containing the content or parameter required to process the payment or supporting information.

Example: "123456"

13 - Bill ($.bill)

SHALL be present.

Includes information related to the Invoice or order being paid.

13.1 - Bill Payment Timing ($.bill.paymentTiming)

SHALL be present.

SHALL be one of the following values:
- "immediate" - Payment is due at the time the QR Code is presented/scanned.
- "deferred" - Payment is due at a future date/time.

If bill.paymentTiming = deferred, then bill.invoice.dueDate **SHALL** be present.

Indicates whether payment is due immediately or later.

13.2 - Bill Description ($.bill.description)

MAY be present.

If present, **SHALL** be a string with maximum 100 characters (see Table 2 - JSON Data Type).

Bill description that may be shown to the Payer.

Example: "Nike Air Max Size 10"

13.3 - Bill Order ($.bill.order)

MAY be present.

Structured details related to the order being paid.

13.3.1 - Bill Order Number ($.bill.order.number)

MAY be present if Bill Order object is present.

If present, **SHALL** be a string with 50 maximum characters (see Table 2 - JSON Data Type).

Every payload **MAY** include either an invoice or order number or both.

Example: "ORD-2025-04-00123”

13.3.2 - Bill Order Date ($.bill.order.date)

MAY be present.

If present, value **SHALL** be a string that conforms to (see Table 2 - JSON Data Type).

Date order created.

13.4 - Bill Invoice ($.bill.invoice)

MAY be present.

Structured details related to the invoice being paid.

13.4.1 - Bill Invoice Number ($.bill.invoice.number)

MAY be present.

If present, **SHALL** be a string with 50 maximum characters (see Table 2 - JSON Data Type).

Every payload **MAY** include either an order number, invoice number or both.

Example: "INV-2024-05-00123”

13.4.2 - Bill Invoice Date ($.bill.invoice.date)

MAY be present.

If present, value **SHALL** be a string that conforms to Date (see Table 2 - JSON Data Type).

Date invoice created.

13.4.3 - Bill Invoice Due Date ($.bill.invoice.dueDate)

SHALL be present if bill.paymentTiming is NOT immediate.

If present, **SHALL** be a UTC timestamp (see Table 2 - JSON Data Type).

An invoice.number **MAY** be provided without a dueDate.

Indicates date invoice payment is due - time component helpful when Payees and Payers are in different time zones.

13.4.4 - Invoicee ($.bill.invoice.invoicee)

MAY be present.

Information related to the entity responsible for the bill.

13.4.4.1 - Invoicee Name ($.bill.invoice.invoicee.name)

MAY be present.

If present, **SHALL** be string with 50 maximum characters (see Table 2 - JSON Data Type).

Example: "John Doe”

13.4.4.2 - Invoicee Phone ($.bill.invoice.invoicee.phone)

MAY be present.

If present, **SHALL** be a string (phone number) that conforms to E.164 international numbering format.

Shall start with "+” followed by 1 to 15 digits where the first digit after "+”.

SHALL NOT be "0”.

Example: "+15552223333”

13.4.4.3 - Invoicee Email ($.bill.invoice.invoicee.email)

MAY be present.

If present, **SHALL** be a string (email) that conforms with RFC 5322.

Example: "johndoe@email.com”

13.4.4.4 - Invoicee Address ($.bill.invoice.invoicee.address)

MAY be present.

Information related to the address of the entity responsible for the bill.

13.4.4.4.1 - Invoicee Address Line 1 ($.bill.invoice.invoicee.address.addressLine1)

MAY be present.

If present, **SHALL** be a string with maximum characters of 60 (see Table 2 - JSON Data Type).

Example: "456 Oak St.”

13.4.4.4.2 - Invoicee Address Line 2 ($.bill.invoice.invoicee.address.addressLine2)

MAY be present.

If present, **SHALL** be a string with maximum characters of 60 (see Table 2 - JSON Data Type).

Example: "Building 17, Suite 1701”

13.4.4.4.3 - Invoicee Address City ($.bill.invoice.invoicee.address.city)

MAY be present.

If present, **Shall** be a string with maximum characters of 40 (see Table 2 - JSON Data Type).

Example: "Seattle”

13.4.4.4.4 - Invoicee Address State ($.bill.invoice.invoicee.address.state)

MAY be present.

If present, **SHALL** use the ISO 3166-2, alpha-2 state code where available.

If present, and ISO 3166-2 not available, **SHALL** be a string with 30 maximum characters.

Example: "WA”

13.4.4.4.5 - Invoicee Address Postal Code ($.bill.invoice.invoicee.address.postalCode)

MAY be present.

If present, string **SHALL** conform to ISO 19160-4, allowing for country-specific variations.

Allows 1 to 20 alphanumeric characters, spaces, or hyphens.

Example: "98109”

13.4.4.4.6 - Invoicee Country ($.bill.invoice.invoicee.address.country)

MAY be present.

If present, **SHALL** be a two-letter country code as defined by ISO 3166-1 alpha-2.

SHALL be exactly 2 uppercase letters (A-Z).

Example: "US”

13.5 - Bill Amount Due ($.bill.amountDue)

SHALL be present.

Information related to the invoice or order amount to be paid.

13.5.1- Bill Amount ($.bill.amountDue.amount)

SHALL conform to Amount (Minor Units) (See Table 2 - JSON Data Type).

SHALL be >=0.

Invoice or Order amount to be paid inclusive of taxes and additional fees.

Note: It is recommended that if an amount due is negative (e.g. a credit), a QR code SHOULD NOT be generated. Any request for a QR Code with a negative balance SHOULD be rejected.

13.5.2 - Bill Currency ($.bill.amountDue.currency)

SHALL be a string that adheres to ISO 4217 alphabetic currency codes (non-numeric).

Numeric codes are not supported in this standard.

Currency on invoice or order.

Example: "USD”

Note - The data structure of the payload is currency-agnostic and **MAY** also carry non-ISO 4217 digital asset currencies (e.g., USDC, BTC) for early adopters.

13.5.3 - Bill Adjustment ($.bill.amountDue.adjustment)

MAY be present if there are adjustments to the amount due.

Up to 10 allowable adjustments.

13.5.3.1- Bill Adjustment Explanation ($.bill.amountDue.adjustment.explanation)

SHALL be a string with 200 maximum characters (see Table 2 - JSON Data Type).

How the amount due on invoice/order has been adjusted - displayed to the Payer.

Example: "$10 discount for paying by QR Code”

13.5.3.2 - Bill Adjustment Amount ($.bill.amountDue.adjustment.amount)

SHALL conform to Amount (Minor Units) (See Table 2 - JSON Data Type).

MAY be positive or negative.

Example: -1000 (-$10.00 for USD)

13.5.3.3 - Bill Adjustment Valid Until ($.bill.amountDue.adjustment.validUntil)

SHALL be a UTC timestamp (see Table 2 - JSON Data Type).

If adjustment date expired, payment **MAY** proceed without adjustment provided the overall payload (Payment Payload ID 7, Table 3) is still valid.

Last date/time Bill Adjustment is valid.

Note: When an adjustment is present, payment applications **SHOULD** display the adjustment expiration to the Payer. If the Payer scans the QR Code but does not authorize payment until after the adjustment has expired, payment applications **SHOULD** re-fetch the payload (e.g., via the URL at field 26.01) to ensure the amount due reflects current terms.

See Annex A.8 for suggested market practices for implementation.

13.6 - Bill Tip ($.bill.tip)

MAY be present if tips are allowed.

Payment applications that do not support tipping **MAY** ignore the tip-related fields defined in this section.

See Annex A.10 for suggested market practices for implementation.

13.6.1 - Bill Tip Allowed ($.bill.tip.allowed)

SHALL be present.

SHALL be a Boolean value.

SHALL contain only true or false (no quotes).

If true, the payment application **SHOULD** allow the Payer to add a tip.

If false, the payment application **SHOULD NOT** allow the Payer to add a tip.

13.6.2 - Bill Tip Range ($.bill.tip.range)

MAY be present if tips allowed.

Minimum and maximum allowable tip percentages.

Note: Payer-facing applications **MAY** calculate the currency value of the tip by applying the percentage (preset or user-selected) to the Bill Amount Due Amount (Payment Payload ID 13.5.1, Table 3).

13.6.2.1 - Bill Tip Min ($.bill.tip.range.min)

SHALL be present if $bill.tip.range is present

SHALL be an integer.

The value **SHALL** be interpreted as an integer percentage (e.g., 15 for 15%).

SHALL NOT include the percent sign (%) or any decimal points.

The computed total (amount + tip) **SHALL** apply only to the payment instruction sent to the payment network and related payment notification; it **SHALL NOT** be re-encoded in the payload.

When requested, the tip amount paid **SHALL** be included in the payment notification. (see Section 9 Payment Notification).

Minimum allowable tip.

Example: 0

13.6.2.2 - Bill Tip Max ($.bill.tip.range.max)

SHALL be present if $bill.tip.range is present.

SHALL be an integer.

The value **SHALL** be interpreted as an integer percentage (e.g., 15 for 15%).

SHALL NOT include the percent sign (%) or any decimal points.

The computed total (amount + tip) **SHALL** apply only to the payment instruction sent to the payment network and related payment notification; it **SHALL NOT** be re-encoded in the payload.

When requested, the tip amount paid **SHALL** be included in the payment notification. (see Section 9 Payment Notification).

Maximum allowable tip.

Example: 50

13.6.3 - Bill Presets ($.bill.tip.presets)

MAY be present if $.bill.tip is present.

If present, the array **SHALL** contain a minimum of 1 and a maximum of 10 entries, where each entry **SHALL** be a String consisting only of digits (0–9).

Each string entry **SHALL** have a maximum length of 3 characters.

The value **SHALL** be interpreted as an **integer percentage** (e.g., "15" for 15%).

SHALL **NOT** include the percent sign (%) or any decimal points.

When requested, the tip amount paid **SHALL** be included in the payment notification.(see Section 9 Payment Notification).

Optional preset tip percentages presented as selectable options to the Payer.

Example: ["10", "15", "20"]

14 - Payment Methods ($.paymentMethods)

SHALL be present.

Currencies, edits, payment networks Payee and/or their PSP will accept.

14.1 - Payment Methods Currency ($.paymentMethods.currency)

SHALL be a string that adheres to ISO 4217 alphabetic currency codes (non-numeric).

Numeric codes are not supported in this standard.

Currency Payee and/or their PSP will accept.

Example: "USD”

Note: The data structure of the payload is currency-agnostic and **MAY** also carry non-ISO 4217 digital asset currencies (e.g., USDC, BTC) for early adopters.

14.2 - Payment Methods Valid Until ($.paymentMethods.validUntil)

SHALL be a UTC timestamp (see Table 2 - JSON Data Type).

Last date/time Currency (Payment Payload ID 14.1, Table 3) is valid for payment.

See Annex A.8 for suggested market practices.

Note: If USD is the currency, this field will often align with general settlement availability and may not be strictly needed. However, this structure is forward-compatible for future use cases such as currency rate or digital asset settlement windows.

14.3 Payment Methods Amount ($.paymentMethods.amount)

Shall be present.

SHALL conform to Amount (Minor Units) (See Table 2 - JSON Data Type).

SHALL be >=0.

Invoice or Order amount to be paid inclusive of taxes and additional fees AND net of adjustments.

Note: It is recommended that if an amount due is negative (e.g. a credit), a QR code SHOULD NOT be generated. Any request for a QR Code with a negative balance SHOULD be rejected.

14.4 - Payment Methods Editable ($.paymentMethods.editable)

MAY be present.

If present, Payer can edit the amount due.

14.4.1 - Payment Methods Editable Range ($.paymentMethods.editable.range)

SHALL be present if $.paymentMethods.editable is present.

Indicates the minimum and maximum edit allowable to the amount due.

14.4.1.1 - Payment Methods Editable Min ($.paymentMethods.editable.range.min)

SHALL be present if $.paymentMethods.editable is present.

SHALL conform to Amount (Minor Units) (See Table 2 - JSON Data Type).

Value **SHALL** be >=0.

Editable min **SHALL** be ≤ Editable max.

Minimum amount allowed when editing amount due.

Example: 10000 = $100.00 USD

14.4.1.2 - Payment Methods Editable Max ($.paymentMethods.editable.range.max)

SHALL be present if $.paymentMethods.editable is present.

SHALL conform to Amount (Minor Units) (See Table 2 - JSON Data Type).

Editable max **SHALL** be >= Editable min.

Maximum amount allowed when editing amount due.

Example: 20000 = $200.00 USD

14.5. - Payment Methods Network ($.paymentMethods.network)

SHALL be present.

Indicates the allowable payment networks.

Note: There is no limit to the number of payment networks offered. The data structure of the payload is network-agnostic and **MAY** also carry a network not listed below for early adopters.

14.5.1 - FedNow ($.paymentMethods.network.fednow)

SHALL be present if FedNow is an allowable payment method.

14.5.1.1 - FedNow Routing Number ($.paymentMethods.network.fednow.routingNumber)

SHALL be present if FedNow is an allowable payment method.

Value **SHALL** conform to Routing Number (ABA RTN) (See Table 2 - JSON Data Type).

14.5.1.2 - FedNow Account Number ($.paymentMethods.network.fednow.accountNumber)

SHALL be present if FedNow is an allowable payment method.

Value **SHALL** conform to Account Number (Bank) (See Table 2 - JSON Data Type).

SHALL be protected - either by tokenization or by encryption using an X9-approved key management methodology as defined in the normative references.

- The encrypted account number results in a ciphertext that **SHALL** be encoded in base64url.

MAY be the account number of the creditor OR ultimate creditor.

14.5.1.3 - FedNow Protection Type ($.paymentMethods.network.fednow.protectionType)

SHALL be present if FedNow is an allowable payment method.

SHALL have one of the following values:
- "tokenized”
- "encrypted”
- "plaintext”

Indicates the method used to protect the account number.

14.5.2 - RTP ($.paymentMethods.network.rtp)

SHALL be present if RTP is an allowable payment method.

14.5.2.1 - RTP Routing Number ($.paymentMethods.network.rtp.routingNumber)

SHALL be present if RTP is an allowable payment method.

Value **SHALL** conform to Routing Number (ABA RTN) (Table 2 - JSON Data Type).

14.5.2.2 - RTP Account Number ($.paymentMethods.network.rtp.accountNumber)

SHALL be present if RTP is an allowable payment method.

Value **SHALL** conform to Account Number (Bank) (See Table 2 - JSON Data Type).

SHALL be protected - either by tokenization or by encryption using an X9-approved key management methodology as defined in the normative references.

- ○ The encrypted account number results in a ciphertext that **SHALL** be encoded in base64url.

MAY be the account number of the creditor OR ultimate creditor.

14.5.2.3 - RTP Protection Type ($.paymentMethods.network.rtp.protectionType)

SHALL be present if RTP is an allowable payment method.

SHALL have one of the following values:
- "tokenized”
- "encrypted”
- "plaintext”

Indicates the method used to protect the account number.

14.5.3 - ACH ($.paymentMethods.network.ach)

SHALL be present if ACH is an allowable payment method.

14.5.3.1 - ACH Routing Number ($.paymentMethods.network.ach.routingNumber)

SHALL be present if ACH is an allowable payment method.

Value **SHALL** conform to Routing Number (ABA RTN) (See Table 2 - JSON Data Type).

14.5.3.2 - ACH Account Number ($.paymentMethods.network.ach.accountNumber)

SHALL be present if ACH is an allowable payment method.

Value **SHALL** conform to Account Number (Bank) (See Table 2 - JSON Data Type).

SHALL be protected—either by tokenization or by encryption using an X9-approved key management methodology as defined in the normative references.

- ○ The encrypted account number results in a ciphertext that **SHALL** be encoded in base64url.

MAY be the account number of the creditor OR ultimate creditor.

14.5.3.3 - ACH Protection Type ($.paymentMethods.network.ach.protectionType)

SHALL be present if ACH is an allowable payment method.

SHALL have one of the following values:
- "tokenized”
- "encrypted”
- "plaintext”

Indicates the method used to protect the account number.

14.5.4 - Zelle ($.paymentMethods.network.zelle)

Refer to network documentation on required fields and processing requirements.

14.5.5 - Visa ($.paymentMethods.network.visa)

Refer to network documentation on required fields and processing requirements.

14.5.6 - Mastercard ($.paymentMethods.network.mastercard)

Refer to network documentation on required fields and processing requirements.

14.5.7 - American Express ($.paymentMethods.network.americanExpress)

Refer to network documentation on required fields and processing requirements.

14.5.8 - Discover ($.paymentMethods.network.discover)

Refer to network documentation on required fields and processing requirements.

9. Payment Notification

9.1 General overview and structure

The Payment Notification is a standardized message sent from the Payer's PSP to the Payee's PSP after a payment is initiated. Its purpose is to provide essential, payment-specific details needed for reconciliation, QR Code lifecycle management and customer messaging.

A Payment Notification is required only when requested by the Payee's PSP via a paymentNotification URL in the original payload.

The payment notification includes primarily payment-specific details that are determined *after* the original payload is sent. These include the currency used, the payment network, the transaction ID assigned by the network, any tip added, the total amount paid, and the expected payment date. All of this information is tied to the original Payload ID.

9.2 Trigger & timing requirements

- If the payload contains a valid paymentNotification URL, the Payer's PSP **SHALL** send a Payment Notification to that endpoint.
- Note: Network-specific practices for requesting and routing Payment Notifications, including examples for ACH, FedNow, and RTP, are described in the Annex A.11.

9.3 Payment Notification Requirements

If payment notification requested, the Payer's PSP must send a POST call to the Payee's PSP to the URL specified on the payment notification data element in the payment payload.

The following table describes the format requirements of the payment notification JSON request.

Payment Notification **SHALL** be formatted as a JSON object and encoded in UTF-8. The Payee's PSP endpoint **SHALL** return an HTTP status code indicating the outcome: 200 OK (or 204 No Content) for success; an appropriate 4xx/5xx code for failures.

Table 4 - Payment Notification Requirements

Payment Notification ID	Field Name	M, C, O FedNow RTP	M,C,O ACH	Type	Max Length	Format	JSON Path
1	Payload id	M	M	Field	32	String	$.id
2	Payment	M	M	Object	-	-	$.payment
2.1	Payment Amount	M	M	Field	18	Integer	$.payment.amount
2.2	Payment Tip Amount	O	O	Field	18	Integer	$.payment.tipAmount
2.3	Payment Currency	M	M	Field	32	String	$.payment.currency
2.4	Payment Network	M	M	Field	NA	String (enum)	$.payment.network
2.5	Payment Transaction ID	M	O	Field	255	String	$.payment.transactionId
3	Payer	O	O	Object	-	-	$.payer
3.1	Payer	C	C	Field	254	String	$.payer.info
4	Expected Date	O	M	Field	24	UTC Timestamp	$.expectedDate

1 - Payload id ($.id)

SHALL be a UUID string consisting of exactly 32 hexadecimal characters ([0-9A-Fa-f]), with no separators.

Value **SHALL** be an exact match of the id from the original payload.

Identifier that links the payment notification to the original QR Code payload.

Example: 123e4567e89B12d3a456426614174000

2 - Payment ($.payment)

SHALL be included.

Information related to the payment.

2.1 - Payment Amount ($.payment.amount)

Shall be a 64 bit integer with minimum value = 0.

Shall be represented as an integer in the smallest currency unit.

In order to interpret the value, it’s necessary to know the number of decimal places defined for the specific currency being used.(e.g. cents for USD).

Total amount sent for payment.

Example: 19825 = $198.25 in USD

2.2 - Payment Tip Amount ($.payment.tipAmount)

MAY be present if a tip was paid.

If present, **Shall** be a 64 bit integer with minimum value = 0.

Shall be represented as an integer in the smallest currency unit.

In order to interpret the value, it’s necessary to know the number of decimal places defined for the specific currency being used.(e.g. cents for USD).

Example: 1000 - $10.00 in USD

2.3 - Payment Currency ($.payment.currency)

SHALL be a string that adheres to ISO 4217 alphabetic currency codes (non-numeric).

Numeric codes are not supported in this standard.

Currency of the payment.

Example: "USD”

Note: The data structure of the payload is currency-agnostic and **MAY** also carry non-ISO 4217 digital asset currencies (e.g., USDC, BTC) for early adopters.

2.4 - Payment Network ($.paymet.network)

SHALL be a String.

The field **SHALL** contain only one of the following exact, all-uppercase values:
- **FEDNOW**
- **RTP**
- **ACH**
- **ZELLE**

Example: "FEDNOW”

Note: The data structure of the payment notification is network-agnostic and **MAY** also carry a network not listed above for early adopters.

2.5 - Payment Transaction ID ($.payment.transactionId)

SHALL be a String with 255 maximum characters (see Table 2 - JSON Data Type).

For FedNow/RTP, the field **SHALL** include the End-to-End ID for the ISO 20022 payment message.

For ACH, the field **SHOULD** include the Trace Number (ODFI routing number + trace ID).

Unique payment transaction reference id.

Example: "XYZ.USBK.X9aTf72qLm.1”

3 - Payer ($.payer)

MAY be present.

Information related to the Payer.

3.1 - Payer info ($.payer.info)

SHALL be present IF the Payer Object is present.

If present, **SHALL** be string with 254 maximum characters (see Table 2 - JSON Data Type).

Any information related to the Payer.

Example: "JaneDoe@Gmail.com”

4 - Expected Date ($.expectedDate)

SHALL be a UTC timestamp (see Table 2 - JSON Data Type).

For ACH, this **SHALL** be an "effective date” (date money is expected to settle).

Date/time when the payment is expected.

Examples with and without fractional seconds: "2025-09-30T18:04:00Z", "2025-09-30T18:04:00.123Z".

10. Security

10.1 Security Overview (JWS)

If an X9 QR Code is scanned by a mobile phone camera or any unauthenticated application, the information cannot be used meaningfully because the QR Code references protected data that is retrievable only by an authorized payment application over authenticated HTTPS.

An unauthenticated application is any application that has not established a trusted, authorized session with the Payee’s PSP and therefore cannot access the protected Payment Payload.

An authorized payment application is one that has established a secure session and is recognized by the Payer’s financial institution or payment service provider as a trusted client permitted to initiate or access payment information.

While the payment app typically authenticates the Payer (e.g., using biometrics, passcode, or MFA) before a payment is initiated, this user authentication is distinct from the app-level authentication required to retrieve the protected Payment Payload.

Security underpins trust and interoperability; therefore, this standard mandates digital signatures and a shared PKI to guarantee integrity and authenticity across parties in real time.

10.1.1 Objects Requiring Digital Signatures

This standard defines security requirements for four objects:

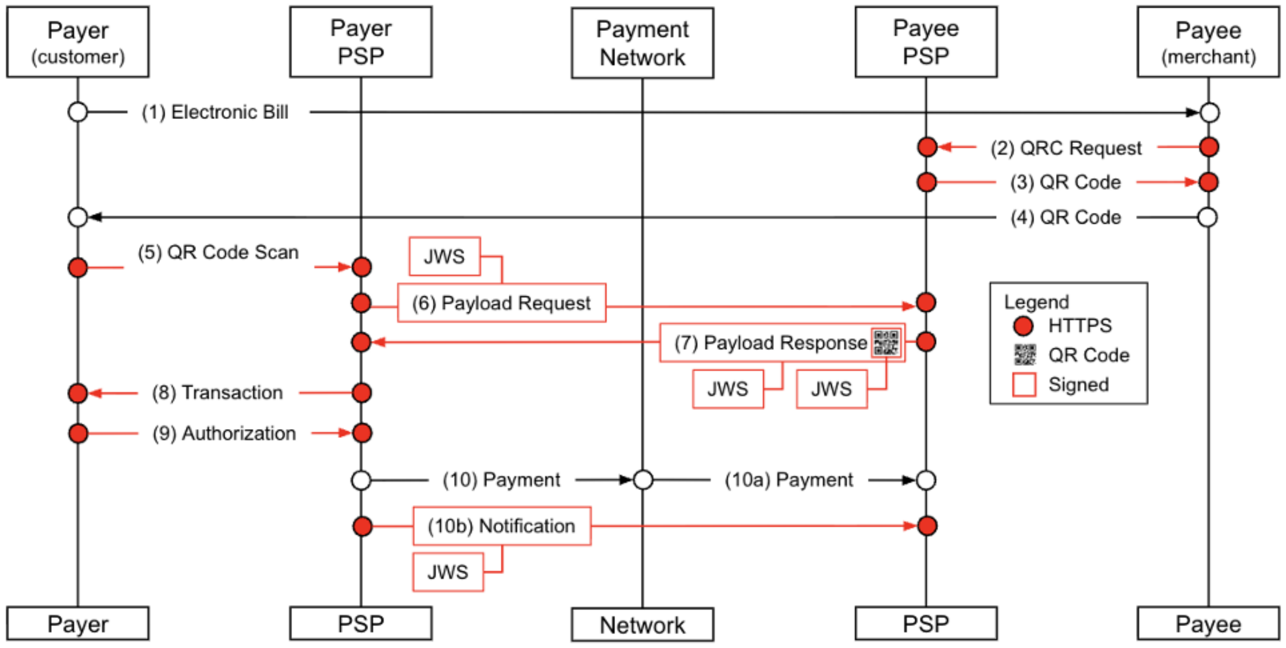
- **Payment Payload Request** — HTTPS request used to retrieve payment details which is digitally signed using JWS (Figure 1, step 6)
- **Payment Payload Response** — HTTPS response containing JSON-formatted payment information which is digitally signed using JWS (Figure 1, step 7)
- **QR Code Content** - included in the HTTPS request and response related to Payment Payload and protected within a digitally signed JWS (Figure 1, step 7)
- **Payment Notification** — HTTPS call which is digitally signed using JWS sent by the Payer's PSP to confirm payment status, if requested (Figure 1, step 10a)

10.1.2 Security Flow

Figure 3 below is a sequence diagram of Figure 1 Biller example.

- The (2) QR Code Request and (3) QR Code Response messages between the Payee PSP and the Payee are assumed to be over a secure channel (e.g. HTTPS).
- (4) The QR Code is presented to the Payer by the Payee and contains reference information (such as a URL and identifier). Its authenticity and integrity are not validated at scan time; instead, the Payer PSP authenticates the QR Code content upon receiving the (7) Payload Response from the Payee PSP, which returns the full QR Code content, protected within a JWS-signed Payment Payload Response, and the encrypted account number.
- (5) When the QR Code is scanned, the Payer's PSP reads the reference information encoded in the QR Code - and uses it to determine the protected endpoint to call in order to retrieve the full Payment Payload.
- The (6) Payload Request message sent by the Payer PSP to the Payee PSP includes a digital signature which the Payee PSP SHALL verify.
- The (7) Payload Response message sent by the Payee PSP to the Payer PSP includes a digital signature which the Payer PSP SHALL verify.
- The (10) Payment messages between the Payer PSP, the Payment Network, and the Payee PSP are out of scope of this standard.
- The optional (10b) Payment notification sent by the Payer PSP to the Payee PSP includes a digital signature which the Payee PSP SHALL verify.

Figure 3 - X9A4 X9.150 Sequence Diagram



10.1.3 X9 Financial PKI

ASC X9 has established the X9 Financial PKI, a certificate authority dedicated to financial services and designed to support the QR Code payment framework. This trusted PKI enables participants to obtain X9-issued certificates, validate certificate chains, and verify digital signatures within a common, governed trust ecosystem. More information about X9 Financial PKI can be found here.

10.1.4 Alternative Encodings (Informative)

This Standard specifies JSON Web Signature (JWS) as the required mechanism for protecting X9.150 messages. Implementations may mutually agree to additionally encapsulate X9.150 messages using alternative formats such as CMS or other ASN.1/XML-based frameworks; however, any such mutually agreed usage is outside the scope of this Standard and is not required for conformance.

When alternative encodings are used:

- All communicating parties should use the same chosen encoding; mixed use (for example, JWS on one side and CMS on the other) will not be interoperable.
- Implementations that choose CMS may use ANSI X9.73 as a reference for encoding conventions.

10.2 JSON Web Signature Overview (JWS)

This standard uses JSON Web Signature (JWS) as defined in RFC 7515 to ensure message integrity and authenticity. JWS is part of the JSON Object Signing and Encryption (JOSE) framework (RFC 7515–7519), which defines how signing metadata is represented in a JWS Protected Header.

10.2.1 JWS Structure

The JWS structure consists of three Base64Url-encoded components:

Base64Url(JWS Protected Header) . Base64Url(JWS Body) . Base64Url(JWS Signature)

Each object **SHALL** be signed as an independent JWS.

10.2.2 How Application Data is Transmitted

In a JWS, the application-level JSON data (such as Payment Payload or Payment Notification) is placed directly in the JWS Body. The JWS Body contains the complete application-level data elements as defined in Section 8 (Payment Payload) or Section 9 (Payment Notification).

Transport and security metadata (such as correlation id's, issued-at, time-to-live timestamps, and http status codes) are included as custom fields in the JWS Protected Header, as permitted by RFC 7515.

The QR Code content is protected within a JWS object exchanged from the Payee PSP to the Payer PSP.

Through this mechanism, the authenticity and integrity of the QR Code content are proven via the JWS signature, even though the QR Code itself carries no embedded signature fields.

When transmitted within a signed JWS, the qrCodeContent represents a verifiable statement from the Payee's PSP, authenticated under its certificate.

10.3 JWS Protected Header Requirements

The JWS Protected Header **SHALL** include both standard JOSE fields and custom X9.150-specific fields for correlation and replay attack prevention.

10.3.1 Standard JOSE Header Fields

These fields are defined by RFC 7515 and are required for signature verification:

Table 5 Standard JOSE Header Fields

Field	Required	Description
alg	Yes	Signature algorithm. SHALL be a value from X9-approved suite (SD-34)
X5u or X5c	Yes	Certificate URL (x5u preferred) or certificate chain (x5c). Used to provide the public key for signature verification
X5t#S256	Yes	SHA-256 thumbprint of the end-entity certificate referenced by x5u or contained in x5c
kid	Yes	Key identifier (free-form string) to aid key rotation
typ	Yes	Type field for content-type clarity. SHALL be "JOSE" or a profile-specific value (e.g. payreq+jws, payresp+jws, paynote+jws, qr+jws).

10.3.2 Custom Header Fields for X9.150

This standard defines the following custom header fields to provide correlation and replay attack prevention:

Table 6 Customer Header Fields

Field	Type	Required	Description
correlationId	String (UUID)	Yes	UUID created by the requester to correlate request and response. The responder **SHALL** echo back the same correlationId in the response.
iat	Integer (Unix timestamp in milliseconds)	Yes	Issued-at timestamp indicating when the JWS was created. Used to validate message age and detect clock skew.
ttl	Integer (Unix time in milliseconds)	Yes	"Time-to-live" duration in milliseconds. The message expires at (iat + ttl). Servers **SHALL** reject any request where (current_time >= iat + ttl).
statusCode	String	Conditional	Required for response only - **SHALL** match the HTTP status code (e.g. 200). Not present in requests.

To ensure that these fields are correctly processed, the JWS Protected Header **SHALL** include the "crit" (critical) parameter identifying iat and ttl as critical headers.

Example JSON request critical parameter below:

"crit": ["correlationId", "iat", "ttl"]

Example JSON response critical parameter below: (Note – statusCode field in response example below)

"crit": ["correlationId", "iat", "ttl", "statusCode"]

If an implementer does not recognize or cannot process any field listed in "crit" it **SHALL** reject the JWS.

Example JWS Protected Headers may be provided in separate, informative implementation guide.

10.4 JWS Body Structure

The JWS Body **SHALL** contain the application-level JSON data directly. The structure depends on the message type. Payment Payload data elements are defined in Section 8. Payment Notification data elements are defined in Section 9.

See Annex A.4 for examples of JWS Body structures.

10.5 JWS Signature Requirements (RFC 7515)

Signatures **SHALL** use JWS (RFC 7515) with a JWS Protected Header as defined in Section 10.3.

10.5.1 Algorithms

Signature algorithms **SHALL** conform to X9 SD-34, including approved PQC (Post-Quantum Cryptography) algorithms as available.

10.5.2 Trust & Certificates

- Signers SHALL use X.509 certificates issued under the X9 Financial PKI
- Verifiers **SHALL** validate the certificate chain to an X9-trusted Certificate Authority (CA)
- Revocation and Validity **SHALL** be enforced per X9.150 requirements

10.6 Trust Establishment

When a Request/Response/Notification is received, trust between the communicating PSPs **SHALL** be established through a two-step verification process:

10.6.1 Transport-Layer Authentication

Both the Payee's PSP and the Payer's PSP **SHALL** verify that the HTTPS connection is authenticated with a valid certificate chaining to a trusted root in the X9 Financial PKI.

Connections that fail certificate validation, chain integrity, or revocation checks **SHALL** be rejected.

10.6.2 Application-Layer Signature Verification

After transport authentication is confirmed, the Payee's PSP **SHALL** verify the JWS contained in the request/response/notification body using the sender's public key.

This verification step ensures the authenticity and integrity of the payload contents, binding the request data to the authenticated sender identity.

10.7 Signature Verification Process

A verifier **SHALL** perform the following steps:

1. Parse the JWS and extract the Protected Header, Body, and Signature.
2. Validate Critical Header Parameters (crit): Check that all fields listed in crit are understood and can be processed. If any field in crit is not recognized, reject the JWS.
3. Validate Custom Header Fields:
    a. Verify that correlationId is a valid UUID format
    b. Verify that iat is a valid Unix timestamp in milliseconds
    c. Verify that ttl is a positive integer within acceptable bounds
    d. For responses, verify that statusCode is present and matches expected HTTP status codes
4. Perform expiration checks of custom header fields (in accordance with Section10.3.2)
    a. The verifier **SHALL** reject any message for which current time >= iat + ttl

   b. The verifier **SHOULD** apply implementation-defined limits on acceptable message age and clock skew when evaluating iat and ttl.

5. Fetch the certificate via x5u (if present) without redirects or authentication, or use x5c
6. Validate certificate validity: Ensure the certificate is not expired or revoked
7. Validate X5t#S256: Verify that X5t#S256 matches the SHA-256 thumbprint of the DER-encoded end-entity certificate
8. Validate certificate path: Verify the certificate path to an X9 Financial PKI trust anchor and check validity/revocation
9. Verify JWS signature: Use the public key in the certificate and the alg parameter specified in the Protected Header
10. Check for duplicate correlationId: The verifier SHOULD verify that this correlationId has not been processed recently.

If any of the above validations fail, the object **SHALL** be rejected.

If verified, transfer verified payload to the application layer.

Note: Caching of certificates via standard HTTP cache headers is RECOMMENDED to avoid downloading the certificate multiple times.

10.8 Digital Signature Responsibility

Table 7 - Digital Signature Responsibility Table

Object	Signed By	Verified By
Payment Payload Request	Payer’s PSP or Payer	Payee’s PSP or Payee
Payload Response	Payee’s PSP Or Payee	Payer’s PSP Or Payer
QR Code Content (within messages)	Both PSP’s or Payer/Payee	Both PSP’s or Payer/Payee
Payment Notification	Payer’s PSP Or Payer	Payee’s PSP Or Payee

10.9 Key Management

- Private signing keys **SHALL** be generated/held by each signer.
- Private signing keys **SHALL NOT** be accessible to X9 or the PKI operator.
- Private keys **SHOULD** be generated and stored in certified HSMs or equivalent secure hardware (encrypted at rest).
- The same entity **MAY** reuse one key/certificate for multiple objects it signs.
- Distinct entities **SHALL** use distinct keys/certs.

Key management methods and controls used with this Standard SHOULD follow industry-accepted practices and the principles defined in relevant X9 key management standards (for example, ANSI X9.69).

Where implementers choose to encapsulate X9.150 messages using Cryptographic Message Syntax (CMS) or other ASN.1/XML-based frameworks for optional or proprietary use cases, they MAY reference ANSI X9.73 for guidance on encoding syntax. Such usage is outside the normative scope of this Standard and does not create additional conformance requirements for X9.150

10.10 Certificate Management

- All signing certificates **SHALL** be issued by the X9 Financial PKI.
- Maximum certificate validity: 18 months.

10.11 Conformance

All certificate issuance, PKI operations, and cryptographic profiles **SHALL** comply with the requirements specified in ANSI X9.150 Financial Services Public Key Infrastructure (PKI) for the Financial Services Industry.

This includes conformance to X9.150's requirements for:

- Certificate formats
- Trust anchors
- Key usage
- Revocation mechanisms
- Cryptographic algorithm profiles

In addition, protection of QR Code Content and associated payloads using this Standard **SHALL** provide integrity and authenticity properties aligned with those defined in ANSI X9.148 Quick Response (QR) Code – Protection Using Cryptographic Solutions.

11. Bibliography

11.1 [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, https://www.rfc-editor.org/info/rfc2119.

11.2 [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words," BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, https://www.rfc-editor.org/info/rfc8174.

11.3 [RFC8941] Nottingham, M., and Kamp, P.-H., "Structured Field Values for HTTP," RFC 8941, DOI 10.17487/RFC8941, February 2021, https://www.rfc-editor.org/info/rfc8941.

11.4 [RFC9530] Backman, A., Jones, M., and Nottingham, M., "Digest Fields," RFC 9530, DOI 10.17487/RFC9530, February 2024, https://www.rfc-editor.org/info/rfc9530.

11.5 [RFC9110] Fielding, R., Nottingham, M., and Reschke, J., "HTTP Semantics," RFC 9110, DOI 10.17487/RFC9110, June 2022, https://www.rfc-editor.org/info/rfc9110.

11.6 [RFC5322] Resnick, P., Ed., "Internet Message Format," RFC 5322, DOI 10.17487/RFC5322, October 2008, https://www.rfc-editor.org/info/rfc5322.1 (For email address syntax)

11.7 [RFC9535] Gössner, S., Normington, G., and C. Bormann, Editors, "JSONPath: Query Expressions for JSON," RFC 9535, DOI 10.17487/RFC9535, February 2024, [https://www.rfc-editor.org/rfc/rfc9535.html).

11.8 [RFC3339] Klyne, G., and A. Mogul, "Date and Time on the Internet: Timestamps," STD 66, RFC 3339, DOI 10.17487/RFC3339, July 2002, https://www.rfc-editor.org/info/rfc3339.

11.9 [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, https://www.rfc-editor.org/info/rfc3986.

11.10 [RFC9421] Backman, A., Richer, J., and M. Sporny, "HTTP Message Signatures," RFC 9421, DOI 10.17487/RFC9421, February 2024, https://www.rfc-editor.org/info/rfc9421.

11.11 [RFC8941] Nottingham, M., and P.-H. Kamp, "Structured Field Values for HTTP," RFC 8941, DOI 10.17487/RFC8941, February 2021, https://www.rfc-editor.org/info/rfc8941.

11.12 [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings," RFC 4648, DOI 10.17487/RFC4648, October 2006, https://www.rfc-editor.org/info/rfc4648.

11.13 [RFC 8259] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format. December 2017. Defines the structure and syntax of JSON, which forms the basis of JWS.

11.14 [RFC 7515] M. Jones, J. Bradley, N. Sakimura. JSON Web Signature (JWS). May 2015. Specifies a means of representing content secured with digital signatures or Message Authentication Codes (MACs) using JSON data structures.

11.15 [RFC 7518] M. Jones. JSON Web Algorithms (JWA). May 2015. Defines the cryptographic algorithms and identifiers used with the JSON Web Signature (JWS) and JSON Web Encryption (JWE) specifications.

11.16 [RFC 7517] M. Jones. JSON Web Key (JWK). May 2015. Defines a JSON data structure for representing cryptographic keys.

11.17 [FIPS180-4] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)," FIPS PUB 180-4, March 2012.

11.18 [E.164] International Telecommunication Union (ITU-T), "The International Public Telecommunication Numbering Plan," February 2010, https://www.itu.int/rec/T-REC-E.164. (for phone syntax)

11.19 [X9.150] Implementation Examples and Future Use Cases Supplemental documentation to this standard that provides implementation examples for 10 scenarios

Annex (Informative) – Market Practices

These Market Practices capture industry stakeholder consensus on recommended practices for implementing ANSI X9.150 in a way that achieves interoperability while remaining rail-agnostic and faithful to the Standard.

A.1 QR Code

Electric Bill Paid via FedNow or RTP

PG&E issues a payment QR Code for Acme Manufacturing’s September electric bill. Acme owes $123.45. The bill is due on Oct 31, 2025, with a $5.00 discount if paid before the due date.

The QR Code is valid until Nov 30, 2025. Payments can be made using either FedNow, RTP or ACH. The payload also includes:

- An unstructured field explaining what the bill is for.
- An additional key/value pair showing a customer reference number.

*Note, the example below includes 0.00 in id 54 - Transaction Amount. The Transaction Amount presented to the Payer **SHOULD** come from the payload.*

QR Code Content

```
00020101021226730169pge-payments.example.com/qrc/
123E4567E89B12D3A4564266141740005204490053038405404 0.005802US5913SAN
FRANCISCO6032PACIFIC GAS AND ELECTRIC COMPANY630467FA
```

A.2 Payment Payload JSON Body

```
{
  "id": "a3f19e0c4b2d47ab9c3e5f6071829cde",
  "revision": 0,
  "qrCodeContent":
"4oCcMDAwMjAxMDEwMjEyMjY3MzAxNjIwZ2UtcGF5bWVudHMuZXhhbXBsZS5jb20vcXJjLzEy
M0U0NTY3RTg5QjEyRDNBNDU2NDI2NjE0MTc0MDAwNTIwNDQ5MDA1MzAzODQwNTQwND
AuMDA1ODAyVVM1OTEzU0FOIEZSQU5DSVNDTzYwMzJQQUNJRklDIEdBUyBBTkQgRUxFQ
1RSSUMgQ09NUEFOWTYzMDQ2N0ZB4oCd"
,
 "createdAt": "2025-09-30T18:04:00Z",
  "revisedAt": "2025-09-30T18:30:15Z",
  "sentAt": "2025-09-30T18:30:16Z",
  "validUntil": "2025-11-30T23:59:59Z",
  "status": "active",
  "paymentNotification": "https://payee.example/notify/PN-9f2b8841",
  "creditor": {
        "name": "PACIFIC GAS AND ELECTRIC COMPANY",
        "phone": "+18005551234",
        "email": "payments@pge.example",
        "address": {
        "addressLine1": "77 Beale St",
        "addressLine2": "Suite 2400",
        "city": "San Francisco",
        "state": "CA",
         "postalCode": "94105",
        "country": "US"
        }
    },
    "MCC": "4900",
    "unstructured": "Electric service for Sep 2025. Statement 38201948.",
    "additionalInformation": {
        "key": "customerReference",
        "value": "CR-847292"
    },
    "bill": {
        "paymentTiming": "Deferred",
        "description": "September 2025 service period",
        "invoice": {
        "number": "INV-38201948",
        "date": "2025-09-30",
        "dueDate": "2025-10-31T23:59:59Z",
        "invoicee": {
        "name": "Acme Manufacturing, Inc.",
        "phone": "+14155552671",
        "email": "ap@acme.example",
        "address": {
        "addressLine1": "100 Industrial Way",
        "addressLine2": "Bldg 3",
        "city": "Oakland",
        "state": "CA",
        "postalCode": "94607",
        "country": "US"
        }
        }
```

```
                },
                "amountDue": {
                "amount": 12345,
                "currency": "USD",
                "adjustment": {
                "explanation": "Early payment discount",
                "amount": -500,
                "validUntil": "2025-10-30T23:59:59Z"
                }
                },
                "tip": {
                "tipAllowed": false
                }
        },
        "paymentMethods": {
                "currency": "USD",
                "validUntil": "2025-11-30T23:59:59Z",
                "amount": 11845,
                "network": {
                "fednow": {
             "routingNumber": "121000358",
             "accountNumber": "12345678987654321",
             "protectionType": "plaintext"
                },
                 "rtp": {
             "routingNumber": "026009593",
             "accountNumber": "ACME00112233445",
             "protectionType": "plaintext"

                },
                "ach": {
             "routingNumber": "051000017",
             "accountNumber": "9876543210",
             "protectionType": "plaintext"
           }
         }
       }
     }
```

A.3 Payment Notification

Payment Notification JSON Body

Invoice was paid in full via FedNow - $5.00 discount for early payment

```
{
  "id": "a3f19e0c4b2d47ab9c3e5f6071829cde",
  "payment": {
          "amount": 118.45
          "currency": "USD",
          "network": "FedNow",
          "transactionId": "20250830-ABC123456789"
  },
  "payer": {
          "info": "AccountsPayables@acmemanufacturing.com"
  }
}
```

A.4 JWS Protected Headers

A.4.1 Request Header Example JSON:

```
{
  "alg": "ES256",
  "x5u": "https://pki.x9.org/certs/payer-psp-cert.pem",
  "x5t#S256": "bEU3ODJmZjhmNWQ5YWZhNjE4YTNhNWY3ZDk4ZGQ3MDA",
  "kid": "payer-psp-key-2025-001",
  "typ": "payreq+jws",
  "crit": ["correlationId", "iat", "ttl"],
  "correlationId": "c7b4c6e0-3e2a-4f5b-9d7c-3e2a1b4c6e0a",
  "iat": 1762535778873,
  "ttl": 300000
}
```

In this example:

- Message was issued at timestamp 1762535778873
- TTL is 300000 milliseconds (5 minutes)
- Message expires at 1762535778873 + 300000 = 1762536078873

A.4.2 Response Header Example JSON:

```
{
  "alg": "ES256",
  "x5u": "https://pki.x9.org/certs/payee-psp-cert.pem",
  "x5t#S256": "a1F4NTNhYmM0ZTU2ZGZhNzE5YjRiNmY4ZTk5ZGU4MDE",
  "kid": "payee-psp-key-2025-002",
  "typ": "payresp+jws",
  "crit": ["correlationId", "iat", "ttl", "statusCode"],
  "correlationId": "c7b4c6e0-3e2a-4f5b-9d7c-3e2a1b4c6e0a",
  "iat": 1762535900000,
  "ttl": 300000,
  "statusCode": "200"
}
```

A.4.3 JWS Body for Payment Payload Request Example

```
        {
                                                              "qrCodeContent":
"4oCcMDAwMjAxMDEwMjEyMjY3MzAxNjIwZ2UtcGF5bWVudHMuZXhhbXBsZS5jb20vcXJjLzEyM0U0N
TY3RTg5QjEyRDNBNDU2NDI2NjE0MTc0MDAwNTIwNDQ5MDA1MzAzODQwNTQwNDAuMDA1ODAy
VVM1OTEzU0FOIEZSQU5DSVNDTzYwMzJQQUNJRklDIEdBUyBBTkQgRUxFQ1RSSUMgQ09NUEFO
WTYzMDQ2N0ZB4oCd"
        }
```

A.4.4 JWS Body for Payment Payload Response Example

```
        {
          "id": "123e4567e89B12d3a456426614174000",
          "revision": 0,
                                                              "qrCodeContent":
"4oCcMDAwMjAxMDEwMjEyMjY3MzAxNjIwZ2UtcGF5bWVudHMuZXhhbXBsZS5jb20vcXJjLzEyM0U0N
TY3RTg5QjEyRDNBNDU2NDI2NjE0MTc0MDAwNTIwNDQ5MDA1MzAzODQwNTQwNDAuMDA1ODAy
VVM1OTEzU0FOIEZSQU5DSVNDTzYwMzJQQUNJRklDIEdBUyBBTkQgRUxFQ1RSSUMgQ09NUEFO
WTYzMDQ2N0ZB4oCd",
          "createdAt": "2025-11-14T10:00:00Z",
          "revisedAt": "2025-11-14T10:00:00Z",
          "sentAt": "2025-11-14T10:00:05Z",
          "validUntil": "2025-11-14T16:00:00Z",
          "status": "ACTIVE",
          "paymentNotification": {
```

```
      "url": "https://payee-psp.example.com/payment-notification"
    },
    "creditor": {
      "name": "PACIFIC GAS AND ELECTRIC COMPANY",
      "address": {
        "city": "SAN FRANCISCO",
        "state": "CA",
        "country": "US"
      }
    },
    "MCC": "4900",
    "bill": {
      "paymentTiming": "immediate",
      "amountDue": {
        "amount": 4000,
        "currency": "840"
      }
    },
    "paymentMethods": {
      "currency": "840",
      "amount": 4000,
      "network": {
        "fednow": {
          "routingNumber": "123456789",
          "accountNumber": "9876543210",
              "protectionType": "plaintext"
        },
        "rtp": {
          "routingNumber": "123456789",
          "accountNumber": "9876543210",
              "protectionType": "plaintext"
        }
      }
    }
  }
```

A.4.5 JWS Body for Payment Notification Example

```
  {
    "id": "123e4567e89B12d3a456426614174000",
    "payment": {
      "amount": 4000,
      "tipAmount": 0,
      "currency": "840",
      "network": "fednow",
      "transactionId": "TXN987654321ABC"
    },
    "payer": {
      "name": "John Doe"
    }
  }
```

Note: For Payment Notification responses, the JWS Body may be empty (only the header fields are required).

A.5 Matching Payer's payment application options to Payee's allowable payment methods

- Merchant accepts RTP, FedNow, and ACH credit push
- Payer's payment application supports RTP and ACH only
- On scan, the app reads the payload and shows only supported options: RTP and ACH
- If the Payer chooses Instant (RTP) and authorizes, the payment goes via RTP.
- If RTP fails (e.g., insufficient funds), the app **MAY** prompt the Payer to choose another supported method. *(Note: Some implementations require a re-scan)*

A.6 Pacs.008 Mapping (RTP/FedNow) for ID, Tip and Unstructured info

Excerpt of XML for RTP/FedNow

```
<RmtInf>
 <Strd>
      <RfrdDocAmt>
      <AdjstmntAmtAndRsn>
      <Amt>16.95</Amt>
      <Rsn>TIPS</Rsn>
      </AdjstmntAmtAndRsn>
      </RfrdDocAmt>
      <CdtrRefInf>
      <Tp>
      <CdOrPrtry>
      <Prtry>QRCD</Prtry >
      </CdOrPrtry>
```

```
        </Tp>
        <Ref>3C2D1E0F9A8B76543210FEDCBA987654</Ref>
        </CdtrRefInf>
        <AddtlRmtInf>Thank you </AddtlRmtInf>
        <AddtlRmtInf>Table=7; Check=CHK-4932</AddtlRmtInf>
  </Strd>
</RmtInf>
```

A.7 ACH mapping for ID, Tips and Unstructured info

Example of how ID, Tips and Unstructured information **SHOULD** be included in Addenda Record of ACH.

- QRCD: [32-digit-id]. TIPS:[18-digit-#].UNST:[82-characters allowed]

Note: The example reflects the recommended format for including QR Code reference data within the ACH Addenda Record. NACHA review and confirmation are pending.

A.8 Example: Validations & responsibilities related to the Payment Payload

Validations should be done to determine what is the final amount due to be shown to the Payer and whether to process the payment.

Payee's PSP: Determining final amount due

- When responding to a Payload Request, Payee's PSP **SHOULD** validate if there is an Adjustment in the payload, and if the Bill Adjustment Date is expired.
    - If Adjustment is present, but expired, the adjustment is NOT made to the amount due.
    - If not expired, use Adjustment Amount (Field ID 13.5.3.2 in Table 3) to calculate the final amount due (Requirement 14.3 in Table 3).

Payer's PSP: Determining whether to process the payment and suggested information to show to the Payer.

- The fields Valid Until (Field ID 7 in Table 3) and Status (Field ID 8 in Table 3) are key.
    - Valid Until indicates the last date/time the Payload is valid for payment
    - Status indicates Indicates the lifecycle of the payload - it can be ACTIVE, PAYMENT_INITIATED, PAID or CANCELLED
- When a Payload is requested, the Payee's PSP will respond with the Payload regardless of whether the Valid Until has expired and regardless of the Status
    - It is the responsibility of the Payer's PSP to check the Valid Until. If expired, they **SHOULD NOT** process the payment.
    - It is the responsibility of the Payer's PSP to validate the Status.
        - If PAYMENT_INITIATED, PAID, or CANCELLED - Payer's PSP **SHOULD** indicate this to the Payer and **SHOULD NOT** allow another payment to be made

Other Considerations:

- Payer's PSP**: SHOULD** ignore any adjustment fields in a payload whose validUntil has expired
- The Currency validUntil (Field ID 14.2 in Table 3) exists in the Payload, but is designed to support currency conversion for early adopters (both non-USD and digital assets). If Payer and Payee are using the same currency, the Valid Until of the Payload (Field ID 7 in Table 3) SHOULD equal the Currency validUntil.
- Payee's PSP **SHOULD** periodically purge or deactivate Payment Payloads whose payload-level validUntil has expired. If Payee's PSP receives a request for a payload that has been purged, **SHOULD NOT** be responsible for returning the Payload.

A.9 Status Payment Payload field
The status field manages the lifecycle of the payload. The Payee's PSP manages updates to status based on specific triggering events (discussed below).

Only certain status transitions **SHOULD BE** allowable. Only payloads with a status of PAYMENT_INITIATED may revert to ACTIVE. A payload with status of PAID or CANCELLED **SHOULD NOT** revert to active.

Each transition **SHOULD** be timestamped and auditable.

The following events **SHOULD** trigger a change in payload status.

ACH credit

- On receipt of a payment notification, The Payee's PSP **SHOULD** update status to "PAYMENT_INITIATED" to prevent duplicate payment.

Note: This Standard does not define a network finality event for ACH; a transition to paid is implementation-dependent and out of scope. Some implementers may have QR Codes paid by ACH with "PAYMENT_INITIATED" status indefinitely. Payee's PSP **SHOULD** periodically purge or deactivate.

RTP / FedNow (Receiving Financial Institutions who also manage QR Code lifecycle)

- On initial network acceptance of the credit transfer request (pacs.008), the Payee's PSP **SHOULD** updated status to "PAYMENT_INITIATED"
- On network confirmation of settlement (pacs.002), the Payee's PSP **SHOULD** update status to "PAID."

RTP/FedNow (Non-Receiving Financial Institutions who manage QR Code lifecycle)

- On receipt of a payment notification, The Payee's PSP **SHOULD** update status to "PAYMENT_INITIATED."

Cancellation

- When the Payee or Payee's PSP cancels the payload, status **SHOULD** be updated to "CANCELLED."

Table 8 - Suggested Transitions (informative)

From \ To	active	initiated	paid	cancelled
active	—	✓	✓	✓
initiated	✓	—	✓	✓
paid	✖	✖	—	✖
cancelled	✖	✖	✖	—

A.10 Tipping and amount calculation practices

These guidelines describe a recommended approach to handling tip-related fields in the Payment Payload and mapping them into underlying payment messages.

- Payer-facing applications without the ability to tip SHOULD ignore these fields.
- Payer-facing applications SHOULD validate any user-entered or preset-selected tip such that min ≤ tip ≤ max.
- Payment applications SHOULD display the base amount and any available tipping options and compute the payable total (base + tip) at runtime.
- Payer's PSP SHOULD include the monetary tip amount in the destination network payment message. See Annex A.6 and A.7 for suggested market practices.

A.11 Payment notification trigger & timing

- For ACH, the Payee's PSP **SHOULD** request a payment notification. The Payer's PSP **SHOULD** transmit a Payment Notification without delay after origination (e.g. after they submits ACH entry to the ACH network) and the Payee's PSP **SHOULD** update the status to initiated upon receipt.

- For RTP, FedNow if the Payee's PSP is NOT a receiving financial institution, they **SHOULD** request a payment notification in order to manage the status of the payload. Upon network notification of a successful payment (e.g. receive pacs.002 and funds are available to the receiver/ Payee), Payer's PSP **SHOULD** send the payment notification to the Payee's PSP.