

Government of India  
Ministry of Housing and Urban Affairs  
Urban Transport Division

Nirman Bhawan, New Delhi  
Dated: April 19, 2021

To

1. Chief Secretaries of all States/ UTs
2. Managing Directors of Metro Rail Corporations

**Subject: Specifications for QR Ticketing System for Transit Applications  
Version 1.1- reg.**

Sir/Madam,

1. This is regarding implementation of National Common Mobility Card (NCMC) ecosystem across India. National Common Mobility Card (NCMC) ecosystem was launched in March 2019 by Hon'ble Prime Minister under the 'Make in India' initiative, which fulfils the vision of "One Nation One Card". National Common Mobility Card (NCMC) ecosystem consists of indigenously developed RuPay Card, Automatic Fare Collection (AFC) System & Software, Validation Terminal, Metro Gate and Interfacing Software with Banking system. This envisages seamless travel across different modes of transport (e.g. Metros, Rail, Bus, Taxi, Auto, Parking etc) besides retail shopping.
2. Final Interface Specification of NCMC ecosystem Version 1.2 (Part IV-VII) was circulated to all State Governments, UT Administration and Metro Companies and also to BRTS Operators and Bus Operators, vide this Ministry's letter of even No. K-14011/5/2019-MRTS-III dated 11.05.2020 for adoption across the country.
3. In addition to the above, Specification for QR Ticketing for Transit Applications Version 1.0 was developed by Centre for Development of Advanced Computing (CDAC) and these draft specifications were circulated for comments to all the State Governments, UT administrations and Public Transport Operators vide this Ministry's letter of even number dated 23<sup>rd</sup> September, 2020.
4. The QR Code Ticketing aims to address the vast base of customers looking for tickets for a single journey/ group journeys by any mode of public transport such as metro, buses and eventually para transit, and for related services such as parking. QR Tickets address three very pertinent requirements for transit operators in our country, viz.



- i. Reducing peak-hour rush
- ii. enabling cashless initiatives and
- iii. Serving as an effective cost-cutting measure by doing away with tokens and reducing paper tickets.

5. Based on the comments and consultations with stakeholders(at workshop held on 22<sup>nd</sup> and 23<sup>rd</sup> Dec, 2020), the specifications have been finalized by C-DAC, as **"Specifications for QR Ticketing System for Transit Applications Version 1.1"** comprising the following parts, viz.

(i) **Part-I :QR Code Data Set-** describes the data structure and elements of the QR payload containing user journey details that is encoded on to the QR image.

(ii) **Part-II : QR Security Scheme** -describes the security features implemented for authentication, data integrity, non-repudiation and confidentiality.

(iii) **Part-III : QR Interface Specification** - describes the various interfaces with various entities that exist in the QR Ticketing System viz. Web, Mobile applications, Ticket Office Machine(TOM), Customer Care, Automatic Fare Collection (AFC), Aggregator (Banking), Validating terminals etc.

6. The **"Specifications for QR Ticketing System for Transit Applications Version 1.1"** are hereby released by this Ministry for adoption across the country. It is requested to ensure adoption of these specifications by all stakeholders such as Road Transport Corporations, Urban Transport Operators, Metro Rail Corporations, BRTS operators etc., for all existing and future operations. The **QR Ticketing System Implementation Guidelines & FAQ V(1.0)** issued by C-DAC are also enclosed for reference.

7. It is issued with the approval of Competent Authority.  
**Encl: As above.**

Yours faithfully

(Lalit Kumar)

Under Secretary to the Govt. of India

ललित कुमार/LALIT KUMAR  
अवर सचिव/Under Secretary  
आवासन और शहरी कार्य मंत्रालय  
Ministry of Housing and Urban Affairs  
भारत सरकार/Govt. of India  
निर्माण भवन, नई दिल्ली-110011  
Nirman Bhawan, New Delhi-110011

Copy to:

1. The Secretary (MoRTH), Secretary (DFS).
2. The Chief Executive Officer, NITI Aayog
3. Director General, STQC, Meity, Electronics Niketan, 6CGO complex, New Delhi.
4. Chief General Manager, SBI, State Bank Bhawan, Madam Cama Road, New Delhi.
5. Managing Director, BEL, Chander Lok, Janpath Rd, New Delhi.
6. Executive Director, CDAC, New Delhi.
7. MD &CEO, NPCI, 10th Floor, The Capital, Mumbai.
8. Director General, BIS, 9 Bahadur Shah Zafar Marg, New Delhi.
9. All BRTS, Road Transport Corporations, City Bus Operators.

Copy also to:

PSO to Secretary (HUA), PPS to JS (AMRUT), PPS to OSD (UT), MoHUA, Nirman Bhawan.

Under Secretary to the Govt. of India

ललित कुमार/LALIT KUMAR  
अवर सचिव/Under Secretary  
आवासन और शहरी कार्य मंत्रालय  
Ministry of Housing and Urban Affairs  
भारत सरकार/Govt. of India  
निर्माण भवन, नई दिल्ली-110011  
Nirman Bhawan, New Delhi-110011





# QR Ticketing System for Transit Operators – Specifications (v1.1)

An Initiative of the  
Ministry of Housing and Urban Affairs (MoHUA), Govt. of India



Prepared by: -

**Centre for Development of Advanced Computing**

(A Scientific Society of the Ministry of Electronics and Information Technology, Govt. of India)

C-56/1, Institutional Area, Sector-62, Noida-201307

## Preface

The QR Ticketing System for Transit Operators is an integrated system of using QR Codes as prepaid tickets to avail various sorts of transit services offered by modern urban transport operators like Metro, Bus, Rental Cars, Rail, Flight, etc. Due to the increased exodus of people moving to cities for work and the busy pace of urban life, consumers nowadays seek the convenience of buying all their amenities from the comforts of their homes and booking urban transport tickets are not an exception. Transport operators also at the same time acknowledging this growing demand and are eager to upgrade their infrastructure to provide seamless and convenient journey experiences to their customers.

The QR Ticketing Solution is a full digital technology solution, tailor-made for a complete mobility-as-a-service offering. The solution is based on present day modern technology standards and infrastructures. Consumer APPs on smart-phones and Webclients on browsers can be provided to enable customers to buy one or multiple single journey QR Code tickets for any kind of urban travel – metro, bus, rail, plane, etc.

QR Tickets address three typical objectives for transport operators in our country

- Reduce peak-hour rush. No more Queues!
- Govt. of India's cashless initiatives – Digital India
- Effective cost-cutting measure – do away with tokens, reduce paper use, etc.

Easier said than done, in order for these objectives to be met and executed successfully, the need of the hour is seamless interoperability of QR Code tickets across heterogeneous operations and services. With that goal in mind, the Ministry of Housing and Urban Affairs (MoHUA) entrusted the job of developing a common standard specification for QR Tickets to Centre for Development of Advanced Computing (CDAC). Although this specification is geared up for nationwide adoption, it has been developed bearing in mind the existing international standards and can address the digital ticketing solution needs of any transit operator around the globe.

## Acknowledgement

At the very outset, Centre for Development of Advanced Computing (CDAC) wishes to extend its deepest thanks to the Ministry of Housing and Urban Affairs (MoHUA) for sanctioning the Research Study/Project titled “QR Ticketing System for Transit Application” per Sanction Order Number K14011/24/2019-UT-IV dated 27th December 2019. CDAC appreciates and expresses its utmost gratitude to MoHUA for its unwavering support and guidance at each and every vertical – be it financial, forging partnerships with other organizations, business co-ordination or otherwise.

CDAC also takes this opportunity to express its sincerest thanks to all the esteemed Public Transport Operators extending the length and breadth of the country including, but not limited to, **Delhi Metro Rail Corporation (DMRC)**, **Bangalore Metro Rail Corporation (BMRC)**, **Brihanmumbai Electric Supply and Transport (B.E.S.T.)**, **Chennai Metro Rail Limited (CMRL)**, **Karnataka State Road Transport Corporation (KSRTC)**, **Mumbai Metro Rail Corporation (MMRC)**, **Mumbai Metropolitan Region Development Authority (MMRDA)**, **Gujarat Metro Rail Corporation (GMRC)**, **Tamil Nadu Transport Department**, **Madhya Pradesh Transport Department**, **Ahmedabad Municipal City Transport Department (AMC)**, **Kochi Metro Rail Corporation** and **Karnataka Urban Land Transport Department** for providing their invaluable comments and suggestions towards improvement of these Specifications. CDAC hereby wishes to extend a special thanks to **DMRC** and **BMRC** for their guidance and knowledge sharing during the development of these Specifications, that greatly helped in giving it its present shape, ready to be adopted as a common National-standard for QR Ticketing in the country.

CDAC further expresses its sincerest thanks to the **National Payments Corporation of India (NPCI)** whose pertinent questions and probes helped in strengthening the architecture of the QR Ticketing System, to **Bharat Electronics Limited (BEL)** for their fantastic support in providing the hardware equipment and associated device drivers required to test QR codes, to **State Bank of India (SBI)** for helping CDAC in solving the financial jigsaw associated with the QR Ticketing System and last but not the least, to **Niti Ayog** for sending their invaluable comments and suggestions.

Since the inception of the project till this present day, these Specifications has gone through several changes and modifications, covering every aspect from – construction of a robust dataset, encoding/decoding and scan times of QR codes, security and design of various interfaces – by taking inputs from external and internal stakeholders. This journey culminated in a brilliant 2-day workshop held over VCon on the 22<sup>nd</sup> and 23<sup>rd</sup> December 2020, which saw as many as 83 participants representing 28 esteemed organizations of our country. CDAC hereby takes the opportunity to say ‘Thank You’ to each and every one of them for their active participation and valuable insights.

Finally, CDAC wishes to express its deepest gratitude to all the team members of the QR Ticketing project for their diligent and sincere efforts and also to the other R&D groups of CDAC Noida who selflessly pitched in to make this deliverable a success.

Jai Hind!

***Never engage in action for the sake of reward. You have the right to work, not to the fruits of your labour.***



## **Part I: QR Code Dataset – QR Ticketing System for Transit Applications**

## CONTENTS

1. Introduction .....	5
2. Scope.....	5
3. Acronyms and Abbreviations .....	5
4. Terms and Definitions .....	6
5. QR Code Dataset Details .....	12
6. Operator Type-specific QR Tickets.....	30
7. QR Image Properties .....	33
7.1. QR encoding method .....	33
7.2. QR Version No.....	33
7.3. Scannability .....	34
7.4. Printing specifications .....	34
7.5. QR Error Correction Code Scheme.....	35
7.6. Illustrative example for a Paper Ticket .....	36
8. References .....	37
Appendix – I (MxN_QR) .....	38
Appendix – II (QR Versioning) .....	43
Appendix – III (QR Dataset Language Codes).....	45

## List of Figures

Figure 1: QR Code Dataset Tree Diagram – I.....	12
Figure 2: QR Code Dataset Tree Diagram – II.....	12
Figure 3: QR Code Dataset Tree Diagram – III .....	13
Figure 4: Big-endian Versus Little Endian .....	15
Figure 5: Sample 2x3_QR Ticket .....	40
Figure 6: Sample 1x1_QR Ticket with Personalized Info.....	41
Figure 7: Sample 1x1_QR Group Ticket with Personalized Info of 2 members .....	42

## List of Tables

Table 4.1: QR Transaction Type .....	8
Table 5.1: QR Code Dataset .....	16
Table 5.2: QR Code Dataset – Security .....	16
Table 5.3: QR Dataset – Version .....	16
Table 5.4: QR Code Dataset – Common Data .....	17
Table 5.5: QR Code Dataset – Dynamic Data .....	18
Table 5.6: QR Dynamic Data – QR Status.....	19
Table 5.7: QR Dynamic Data – Update QR Status at Real-time .....	20
Table 5.8: QR Dynamic Data – Location.....	20
Table 5.9: QR Code Dataset – Ticket Block .....	21
Table 5.10: QR Ticket Block – Personal Info.....	22
Table 5.11: QR Ticket Block – Validator Info.....	23
Table 5.12: QR Ticket Block – Ticket .....	24
Table 5.13: QR Ticket – Group Size .....	25
Table 5.14: QR Ticket – Product ID .....	27
Table 5.15: QR Ticket – Service ID.....	29
Table 6.1: Operator Ticket Block for Transit Applications .....	31
Table 7.1: QR Encoding Scheme .....	33
Table 7.2: Error Correction Scheme vs. Overhead.....	35

## Version History

Date	Version	Author	Comments
17/07/2020	1.0	CDAC	Incorporated comments after presentation external stakeholders – DMRC, BMRC, NPCI, BEL on 24/6/20. Final draft release. This version was circulated to all PTOs, BEL, BIS, Niti Ayog, STQC, etc.
04/02/2021	1.1	CDAC	Amendments related to QR Code Dataset owing to comments received from external stakeholders on V1.0. Also incorporated comments and suggestions received from participants in Workshop conducted by CDAC on 22 <sup>nd</sup> & 23 <sup>rd</sup> Dec 2020.



## 1. Introduction

The initial target for the QR Ticketing System shall be to facilitate the usage of QR Tickets for all kinds of Urban Transport like Metro rail, City buses, Mono rail, Suburban rail, etc. Eventually this system shall include and encompass other industries including, but not limited to, Rail, Air, Museums, Archaeological sites, Hotel reservation, etc. So the QR Code Dataset should be designed with as much foresight as is possible on the current date. This has been the primary goal of the project – to design a specification that can be adopted easily across a large spectrum of service providers.

## 2. Scope

This first part of the specification, **Part I – QR Code Dataset** defines the detailed common and PTO specific data structures, their properties, usage, data types and size. Also described in this part are properties of QR images, printing specifications and other parameters such as error correction level, data density, version etc.

## 3. Acronyms and Abbreviations

AFC	Automatic Fare Collection System
CAFC	Centralized AFC System
ECC	Error Correction Code
GSM	Grams per Square Meter
HID	Human Interface Device
PTO	Public Transport Operator
QR Code	Quick Response Code
RFU	Reserved for Future Use
Stn code	Station Code
TOM	Ticket Office Machine
TRM	Transit Rules Management
TXN Ref. No	Transaction Reference number

## 4. Terms and Definitions

Some important definitions used throughout this specification are described here. These definitions mainly relate to fields of the QR Code Dataset described in this specification.

- **QR Code:** A QR code (short for "Quick Response" code) is a type of 2-dimensional barcode that contains a matrix of square grid arranged in a square over a white background. It can be read by an imaging device such as a camera, and processed using an algorithm until the image can be appropriately interpreted. For the QR Ticketing System, QR codes shall be used to encode single and / or group journey tickets. These tickets shall be issued either in digital form to users' mobile phones or issued as Paper tickets in Station premises.
- **Presence:** Term used to denote if a particular data element in a Dataset is mandatory, optional or conditional. Data elements that denote mandatory functional requirements that must be presented on encoded QR Code.
- **Mandatory:** The Presence of these Data elements is required or mandatory.
- **Optional:** The Presence of these Data elements is not mandatory.
- **Conditional:** The Presence of these Data elements is not mandatory but must be present if some condition is satisfied.
- **Operator ID:** Indicates the unique ID of the operator or PTO. These IDs may be issued either by some centralized controlling authority or maintained by the Ticketing System implementer in agreement with the PTOs.
- **Requester ID:** This term represents the unique ID of Application or TOM used for availing the QR Ticketing Services. This is either the AppID (Application ID) or TomID (Ticket office machine ID).
- **QR Security:** This field describes the Security scheme used in the QR Code. The different types of Security schemes present in the QR Ticketing System is describes in QR Specifications – Part II.
- **QR Dataset Version:** It represents the version of the QR Dataset being used in the QR Ticketing System. It is not to be confused with the QR Version. Details of QR properties and QR Version is given in [Section 7](#).
- **Language Code:** This represents the language in which data is displayed or printed. Refer [Appendix – III \(QR Dataset Language Codes\)](#).
- **QR Ticket Generator ID (TG ID):** Represents the authorized system that generates the actual payload of the QR ticket. The PTO may also choose to be the TG itself. If any operator acts as a TG, then Ticket Generator ID and Operator ID may remain same for that operator.
- **Ticket Serial No:** This represents the unique identifier allocated for each generated QR. This identifier is 8 bytes long, where the first 4 most-significant bytes are used to represent the date of ticket generation and the remaining 4 bytes (or 32 bits) are used to maintain a sequence number. Bits 31 and 30 of the Ticket Serial No. are reserved for the "Requesting Source Indicator" as described in the

table below. The remaining 30 bits (0 – 29) is the actual sequence or counter that has a range from 0 to  $2^{30} - 1 = 1073741823$ . The value 0 is not used. The maximum value of the sequence number is greater than 100 Crores, a very large number. Therefore, there is no need to reset it until it gets exhausted. The sequence number can be stored in persistent storage. But it must be noted here that only one sequence shall be maintained on the TG irrespective of the Requesting Source.

Least significant 4 bytes	Byte 3		Byte2	Byte1	Byte0
Requesting Source	QR Requesting Source Indicator		Bits	Bits	Bits
	Bit 31	Bit 30	29-24	23 – 16	15 – 8
Undefined	0	0	0	0	0
Mobile - M	0	1	X	X	X
Webclient - W	1	0	X	X	X
TOM - T	1	1	X	X	X

Sequence No: Range 1 – 1073741823

The format of ticket serial number when displayed on the screen or printed on paper must be [DDMMYYYYhhmmss][M/W/T][xxxxxxxxxx] where "xxxxxxxxxx" represents the 10-digit sequence number. Zeroes must be prefixed if sequence number is less than 10 digits so that the total string length is always 25 characters. For example, a mobile ticket bought on 21<sup>st</sup> April, 2020 at 18:47:54 hours, with the current sequence counter at 82506 will be displayed as '21042020184754M0000082506'.

When storing the full serial number in database (or for encoding it in the QR), a 'long' data type must be used as follows.

Generation date time (21/04/2020@18:47:54) → represented as seconds since epoch = 1,587,475,074 decimal = 5E9EF282 hex

QR requesting Source= 1 (b01 = M) → Mobile App QR

Sequence number (30 bits) = 82506 = 1424A hex

Datetime in HEX (4 bytes)	QR Type (2 bits)	Sequence in HEX (30 bits)
5E9EF282	01	1424A

The QR Type bit and Sequence can be clubbed into 4 bytes (32 bits) giving the Hex value 4001424A.

Therefore, the final Ticket Serial No can be clubbed into a single ‘long’ data type with value equal to **5E9EF2824001424A** in Hex.

- **QR Generation Date time:** This shall represent the date and time of the instance at which QR code is generated by the TG and is defined as the number of seconds elapsed since the EPOCH – midnight 1<sup>st</sup> January, 1970.
- **QR Update Time:** This element is present on the Dynamic Data branch of the QR Code Dataset and shall represent the current Time in seconds when the QR image is changed. This field will be used in conjugation with the refresh interval set by the App. It is only relevant for APP-based QRs. For Paper QRs this value shall be the time of QR rendering.
- **Transaction Type:** This element shall represent the mode of transaction. The most significant 3 bits of this byte have the **payment mode** and the remaining 5 bits have the QR Type. The field Transaction Type is defined as shown in the table below.

**Table 4.1: QR Transaction Type**

Txn Type	Payment Mode QR – Bits 7 to 5	QR Type – Bits 4 to 0	Txn Type Value	Usage
QR Purchase	b010	b00001	0x41 (65)	TOM, App or TG
QR Transaction	b010	b00010	0x42 (66)	Ticket or Exit
QR Verification	b010	b00011	0x43 (67)	Entry
QR Error	b010	b00100	0x44 (68)	Inspector or exit
QR Duplicate	b010	b00101	0x45(69)	Request for a duplicate of an existing valid Purchased QR ticket.
QR Update	b010	b00110	0x46(70)	Any

- **TXN Ref. No:** Transaction Reference Numbers are generated by an external source, in this case the Financial Institution (Payment Service Provider) integrated with the App Provider. A customer usually gets a range of choices when making online payments –like Credit Card, Debit Card, PayPal, Net Banking, UPI, etc. Each of these payment modes usually have different Txn\_Ref\_No formats. 22 characters is enough to store this value.

As an example, for the Real-time Gross Settlement System (RTGS – online instant payments), the unique transaction reference number is 22 characters long. The structure of the unique number is



“XXXXRCYYYYMMDDnnnnnnnn” where **XXXX** is IFSC (first 4 character) of sending participant, **R** represents RTGS system, **C** represents channel of the transaction, **YYYYMMDD** represents year, month and date of the transaction, **nnnnnnnn** denotes the sequence number [1].

- **Total Fare:** Total Fare (4 bytes) of the entire QR Ticket. Since a QR Code may consist of multiple tickets, the total fare is the sum of all the individual journeys. Denomination is in Paisa.
- **Booking Location:** As the name implies, two fields – ‘Booking Latitude’ and ‘Booking Longitude’ denote the location coordinates of the mobile sent to the TG by the APP when the user requests a QR Ticket. This quantity is an idea to tackle counterfeit QRs and tackle possible misuse of the QR Ticketing System. One scenario of its usage is described in Section 4.3.1.4 of Part III – QR Interface Specifications.
- **Mobile Number:** It is mandatory for Applications (Mobile / Webclient) implementing the QR Ticketing Systems to send the Mobile number of the customer to the Ticket Generator. Since the Mobile number of the customer is encoded in the QR, it will also be very useful for the Customer Care operators to verify/validate and provide assistance to customers for such tickets in case the customer has some problem.
- **QR Dataset Dynamic Data:** The dynamic data element consists of 32 bytes and is used by the APP (mobile) to store current information about the journey.
- **QR Update Datetime:** The APP can use this to refresh the QR image at a pre-defined frequency so that the QR scanned by the validator is always an updated image. This can be useful to prevent copying of QRs or misuse of the QR Ticketing System.
- **QR Status:** This shall represent the status of the QR ticket. Only the status of the most current journey is available here.
- **Location:** The Dynamic Data element can also contain location data – latitude and longitude, each equal to 3 bytes long. If used along with the QR Update Datetime, the location co-ordinates of the mobile can be checked against the location of the validating terminal. For Metro Gate terminals the location will be fixed (stationary), whereas for mobile terminals like handheld ETIMs used in Buses the location of the terminal is usually updated in real-time.
- **Operator-specific Dynamic Data:** These 19 bytes are reserved for the Operator if it wants to maintain some dynamic data of its own.
- **QR Ticket Block:** The QR Ticket Block is divided into two parts static block and dynamic block. The entire *Static block* is optional but the *Dynamic Block* is mandatory.
- **Static block:** Contains the user’s personal information like name, age, etc. Static block is again divided into two elements – Primary Info and Static info.
- **Primary Info:** Contains personal information of the primary person (or user) that is actually requesting the ticket.

- **Secondary Info:** This shall contain the details of other members travelling with the primary person i.e. it also implies that the ticket is a group ticket. Secondary Info is an optional element and its presence is driven by the Group Size and Product ID field.
- **Dynamic block:** Contains information about the tickets in a QR code. It can be said to be the most important part of the QR Dataset.
- **No of Tickets:** The count of the total number of tickets for each operator.
- **Validator Info:** This 1-byte field has a dual purpose. The most-significant nibble (4 bits) denotes the scanner capabilities or options supported by the Operator. On the least-significant nibble, the least significant bit is used to denote if the Operator's Ticket(s) are encrypted or not. The remaining bits are reserved for future use. Refer [Table 5.11](#).
- **Group Size:** This denotes the number of persons in a group ticket. It is not only used to signify the group size of a group ticket but also essential to calculate the total ticket fare. Group size is a 1-byte number. When the 7<sup>th</sup> bit of this byte is set (=1), it signifies that Secondary Info is also present. Default value for group size is 1. Details about using this field are given in [Table 5.13](#).
- **Source Station:** The starting point of a journey. It is a numeric station code and is operator-specific.
- **Destination Station:** The ending point of a journey. It is a numeric station code and is operator-specific.
- **Activation Date time:** It essentially means the date and time of journey. Usually all transit operators have time-tables for the transport services they provide. A user can choose the date and time that he/she wishes to embark on a journey. This field will hold that information.
- **Ticket Fare:** Found in every ticket, it gives the price or value of an individual ticket. Denomination is in Paisa.
- **Duration:** This field can be used as a polymorphic quantity. In some cases, it can signify the actual time of the journey that may be useful for train or rail tickets. But for Metro or Bus operators this field is actually used in a separate sense. It implies the duration for which the ticket remains active *after* it has been scanned. The unit is in minutes. The default value is currently fixed at 180 minutes or 3 hours.
- **Validity:** Validity represents the time period for which the ticket is valid. The unit is in minutes. It denotes the time for which the ticket is valid from the Activation Datetime. Its value is PTO-specific and when not supplied, the default value is taken as 480 minutes or 8 hours. It is quite possible that the ticket whose Activation Date is on one day can actually percolate to the next day.
- **Operator-specific Ticket Data:** The QR Dataset Ticket reserves 8 bytes of data for operators to write any proprietary information of their own into these 8 bytes. For example, some operator may wish to maintain a separate sequence number for its own tickets. In case of Bus Operators, this field may be used to store the Route number, as route numbers are an important parameter for buses.

- **Product ID:** Operators may sell many products like group ticket, student pass, senior citizen pass, etc. All these products are signified by unique identifiers and the field is called Product ID. Refer [Table 5.14](#).
- **Service ID:** A single Operator may provide more than one service. For example, a metro has a primary service that is the Metro rail service. But it may also have some available Parking service that customers can buy along with a journey ticket. All these services are signified by unique identifiers and the field is called Service ID. Refer [Table 5.15](#).

**Convention:** - In [Figure 1](#), [Figure 2](#) and [Figure 3](#) below, the convention ' $(N)$ ', where  $N$  is a number, is used to represent the size of the field in Bytes.

## 5. QR Code Dataset Details

The figure below shows a diagrammatic representation of the QR Code Dataset hierarchical data structure (tree).

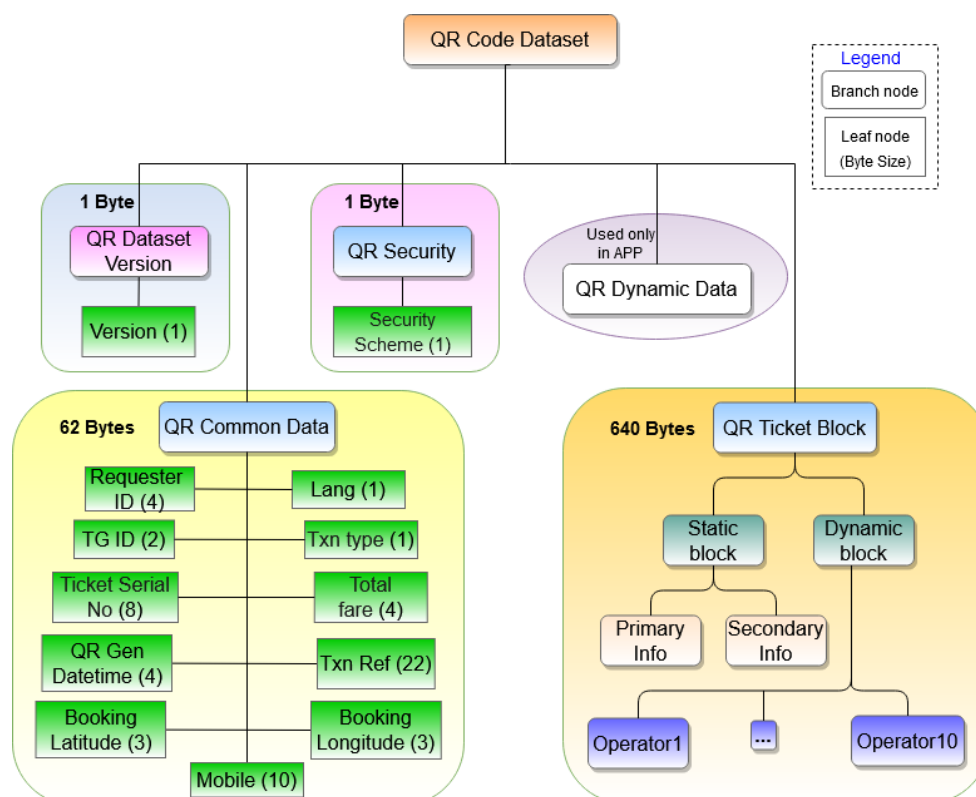


Figure 1: QR Code Dataset Tree Diagram – I

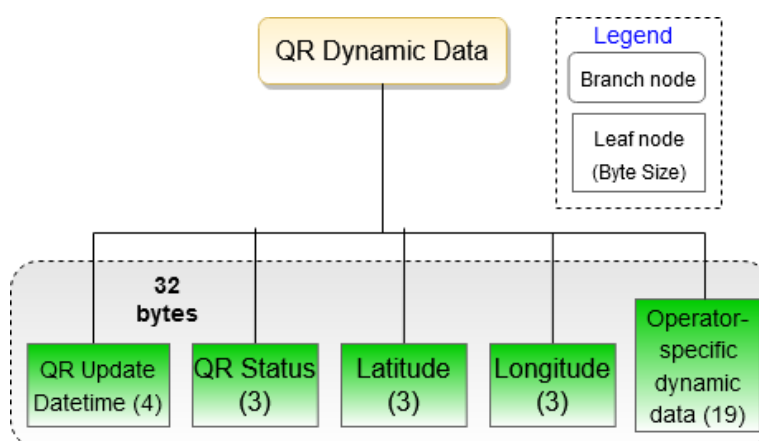


Figure 2: QR Code Dataset Tree Diagram – II



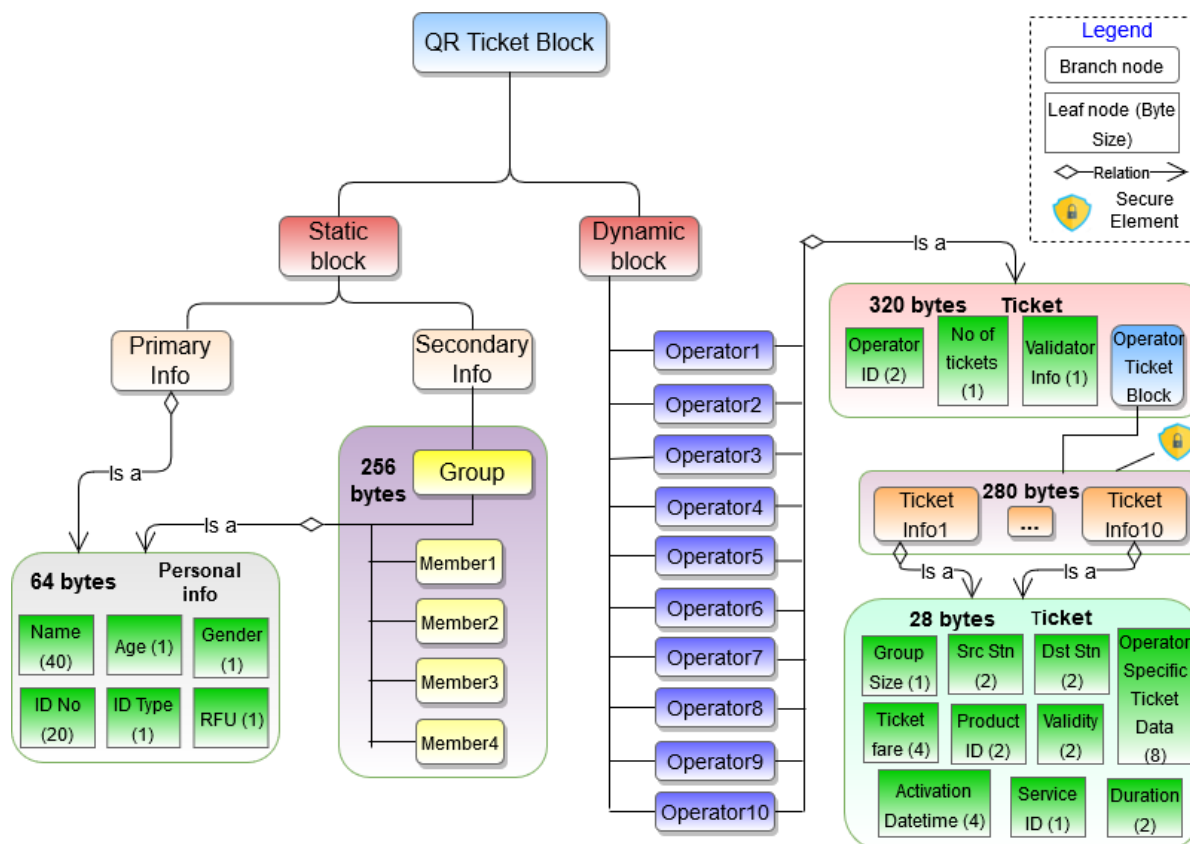


Figure 3: QR Code Dataset Tree Diagram – III

#### Notes: -

1. Primary Info and Secondary Info are both instances of Personal Info. There can be only one instance of Primary Info but since Secondary Info is essentially group information, there can be up to 4 group members and each group member is an instance of Personal Info. How a ticket gets associated with the group is discussed in [Important Note 1](#).
2. The entire Static Block (Primary Info + Secondary Info) is an optional element as most operators (like Bus and Metro operators) seldom need to use personal information of commuters. The Static Block is only likely to be useful for Train or Plane tickets.
3. There is a pool of 10 tickets – ‘TicketInfo1’ through ‘TicketInfo10’. This pool must be shared among all the 10 operators – Operator1 through Operator10.

If one operator uses up the entire pool of 10 tickets, then that QR will be full and cannot include any more tickets. In that case, the *Dynamic Block Size* will be

Size of [OperatorID + NoOfTks + ValidatorInfo + (10\* Tickets)] = 2 + 1 + 1 + (10\*28) = 284 bytes.

However, if there is a QR where all the 10 operators are included, then it essentially means that there is only 1 ticket per operator. The *Dynamic Block Size* in that case will be

10 \* (OperatorID + NoOfTks + ValidatorInfo + Ticket) = 10\* (2 + 1 + 1 + 28) = 10 \* 32 = 320 bytes.

For a complete detailed analysis of calculating QR sizes, please refer to Section 5.1.1.2 of QR Specifications – Part II. The convention used throughout the specifications to represent M operators and N Tickets is **MxN\_QR**. Please refer to [Appendix – I \(MxN\\_QR\)](#) for details.

### **QR Code Dataset Tables**

Before the QR Code Dataset is described it is essential to discuss about storage (specifically memory representation) and display representation of numeric data types in computer systems.

**Storage, Display and Endianness:** For data types that are numeric, it is essential to make a distinction between the Storage format – i.e. how a numeric value is represented in memory – and the Display format – i.e. how the numeric value is shown on the screen or printed on paper. While the display is always shown in typical human-readable text (left to right), the storage order may actually differ from system to system. This is due to the “endianness” of a system. Endianness is the attribute of a system that indicates how a system stores a multi-byte numeric value. It refers to the order or sequence in which data is stored in computer’s memory. It is primarily categorized into Little-endianness and Big-endianness.

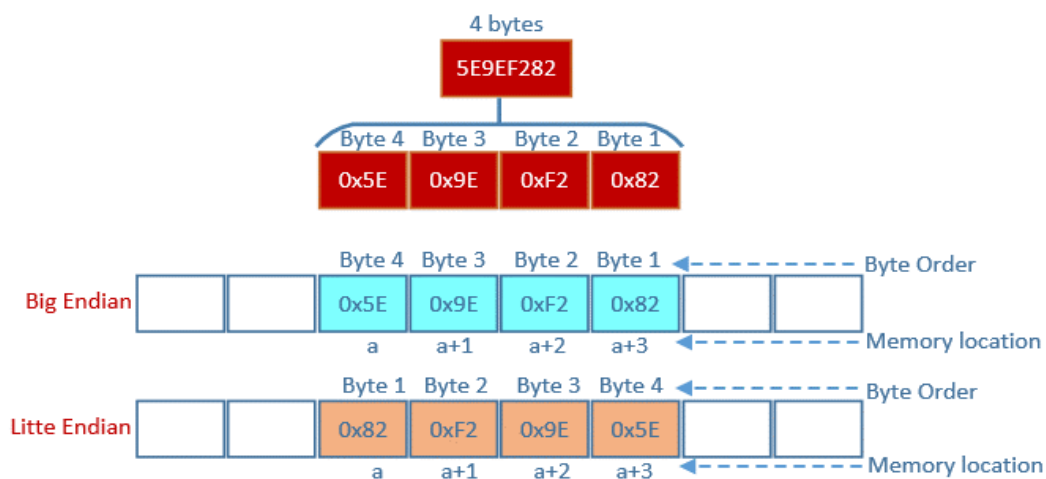
Little-endian systems store the least significant bytes before the most significant bytes. It has left to right ordering in terms of memory address and byte order i.e. low to high.

Big-endian systems, on the other hand, stores in the opposite order. The most significant bytes are stored before the least significant bytes in memory addresses ranging from low to high.

For all practical purposes, every system follows either of these two categories. Endianness comes into play when two systems exchange data in numeric strings or encode numeric data as into images to be read by the target system. This concept shall become clear from the example below.

Consider a multi-byte numeric field like QR Generation Datetime which is a 4-byte element. Let us suppose that QR Generation date time is 21/04/2020@18:47:54. In hexadecimal form, it will be stored in memory as 5E9EF282 Hex. As seen in [Figure 4](#) below, the order of bytes are different in big-endian and little-endian systems. Now say that a little-endian system wants to send these 4 bytes as numeric string. So it accesses them from memory locations ‘a’ through ‘a+3’, then clubs them as a ‘string’ to get “82F29E5E”. The receiving system, which say follows the big-endian format, now takes this value as it is and tries to retrieve the actual date and gets the date to be 14/08/2039@18:08:54 which is probably not what it was expecting.

For consistency, in the QR Ticketing System, only Big-endian format should be followed. Just for trivia, Java generates byte codes in Big-endian format and this order is also known as the Network Byte Order as well as the human reading format (left to right).



**Figure 4: Big-endian Versus Little Endian**

Finally, another concept that needs to be clarified at the very outset is that there are a lot of numeric fields in the QR Code Dataset. Numeric data can be either signed or unsigned. In the QR Ticketing System, all numeric data is unsigned, i.e. all values are greater than or equal to zero.

From here on we describe the QR Code dataset.

The following tables describe the complete layout of the QR Dataset shown in the figures – [Figure 1](#), [Figure 2](#) and [Figure 3](#) depicted above. The dataset is defined after taking into consideration various requirements of Operators for representing their Ticket information.

**Table 5.1: QR Code Dataset**

QR TICKET DATASET			
Tag	Length	Presence	Description
81	1 Byte	Mandatory	QR Security
82	1 Byte	Mandatory	QR Dataset Version
83	62 Bytes	Mandatory	QR Common Data
84	32 Bytes	Mandatory	QR Dynamic Data
85	640 Bytes	Mandatory	QR Ticket Block = Static Block (Optional) + Dynamic Block (Mandatory)

The following tables describe in detail each element or branch of the QR Code Dataset.

**Table 5.2: QR Code Dataset – Security**

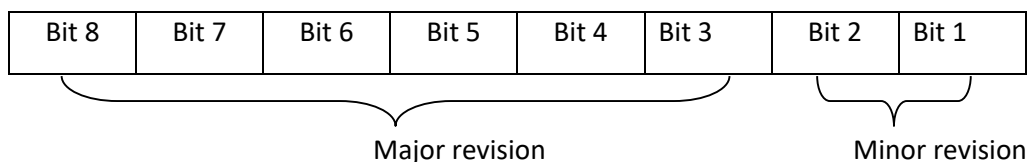
QR Security				
Tag	Total Length	Size (Byte)	Security Scheme	Description
81	1	1	Security scheme	The security scheme adopted for the implementation.

QR Security gives the choices of Security schemes that can be employed in the QR Ticketing Scheme. It is described in detail in Part II – QR Specifications.

**Table 5.3: QR Dataset – Version**

QR Dataset Version				
Tag	Total Length	Size (Byte)	Version	Description
82	1	1	Bits 1 & 2 – Minor revision Bits 3–8 – Major revision	Version number of the entire QR Dataset. Present version number is 0x04.

QR Dataset Version specifies the version of the entire QR Code Dataset. The version number of QR Dataset mentioned in this document is 1.0. Any updates made to any field of the QR Dataset – be it name or size, any addition or removal of fields, after the freeze will mean that the Version value would have to change. Minor changes like name change or size changes will affect only the minor revision whereas major changes like field addition or removal will mean a change in both the major and minor revision. Major revision and minor revisions are described as follows.



The current QR Dataset Version is presently fixed at v1.0 which in binary is <Major – 000001><Minor – 00> = 0x04 in Hex.

**Table 5.4: QR Code Dataset – Common Data**

QR Common Data				
Tag	Total (Length)	Size (Byte)	Element	Data type / Presence
83	62 Bytes	1	Language	Numeric/ Mandatory
		2	QR Ticket Generator ID (TG ID)	Numeric/ Mandatory
		1	Transaction Type	Numeric/ Mandatory
		8	Ticket Serial No/QR Serial No.	Numeric/ Mandatory
		4	QR Generation Date time	Numeric/ Mandatory
		4	Requester ID* (APP ID or TOM ID)	Numeric/ Mandatory
		22	TXN Ref. No	Alpha-Numeric/ Mandatory
		4	Total Fare	Numeric/ Mandatory
		3	Booking Latitude	Numeric/ Optional
		3	Booking Longitude	Numeric/ Optional
		10	Mobile	Alpha-Numeric Only [0-9] allowed/ Mandatory for

				Mobile and Web-based clients, Optional for TOM (Paper) QRs
--	--	--	--	--

Note: - Please refer to the [Terms and Definitions](#) section for detailed description of the elements inside the QR-Common-Data.

\*Requester ID – Only the 2 least significant bytes are used. The 2 most significant bytes are RFU.

**Table 5.5: QR Code Dataset – Dynamic Data**

QR Dynamic Data					
Tag	Total (Length)	Size (Byte)	Data type/ Presence	Element	Description
84	32 Bytes	4	Numeric/ Mandatory	QR Updated Time	Instantaneous time (in secs) will be maintained here. In the mobile App, after some pre-defined time this field will be updated with the current date and time of day. The QR will be then rendered again. It can be useful in curtailing the use of copied QRs. For Paper QRs this element shall simply have the time of QR rendering. Details of such use is provided in the Appendix I of QR Specifications – Part III.
		3	Numeric/ - Optional	QR Status	Status of current QR Ticket as described in <a href="#">Table 5.6</a> .
		3	Numeric/ Optional	Latitude	The latitude expressed in 3 decimal places as described in <a href="#">Table 5.8</a> .
		3	Numeric/ Optional	Longitude	The longitude expressed in 3 decimal places as described in <a href="#">Table 5.8</a> .
		19	Numeric/ Optional	Operator specific Dynamic Data	This space is reserved for the Operator if it wishes to write any specific information of its own.

The Dynamic Data is relevant only for APP QRs and has no significance in Paper QRs. The QR Status is a field that shall be maintained by the APP to indicate the present state of the QR and may be relayed from the Operator backend. It shows the status of individual tickets and is only useful on the activation dates of the QR Tickets i.e. the Date of Journey.

In the QR Ticketing System, there is this provision given to update status of a QR in the mobile in real-time, by making use of the mobile's geo-location. Sending real-time journey updates to mobiles is a desirable feature of the QR Ticketing System. Features like these are described in Appendix III of QR Specifications – Part III.

Status is a 3-byte field – the MSB 2 bytes are for the Operator’s ID. The LSB contains the Ticket Number and its current State. This is essential because there may be multiple tickets in a QR belonging to multiple operators. So the APP should know exactly which ticket number belonging to which Operator is to be updated.

**Table 5.6: QR Dynamic Data – QR Status**

QR Status		
Operator ID (2 Bytes)	Ticket Index(4 bits)	QR State (4 bits)
0-65535	1-10 (only 10 tickets are supported) 11-15 RFU	0-Inactive
		1-Active
		2-Entry
		3-Exit
		4-Tap
		5-Invalid
		6-15 - RFU

Please note that **Ticket Index** is like the position of a Ticket within an Operator’s Ticket Block. Let us assume that the QR Ticket has 3 tickets for Operator ID 135 for three different dates in an order as follows. The App only encodes the ‘QR Status’ field the most recent ticket on the QR Code.

Index	Operator ID	Activation Date
1	135	07/04/2020@12:30:45
2	135	12/04/2020@12:30:45
3	135	14/04/2020@12:30:45

Let us assume that a person buys 3 tickets as shown in the table above with the first ticket being the most recent one (highlighted in blue). Also assume real-time status is sent to the App (refer to [Table 5.7](#) below).

- On **07/04/2020**, the App should update the ‘QR Status’ field as per Sl. No. 1.
- When the commuter uses that ticket the QR State changes from 1-Active to 2-Entry. The QR Status field is updated accordingly – Sl. No. 2.

- Finally, when the commuter uses that ticket at Exit, the QR State changes from 2-Entry to 3-Exit. The QR Status field is updated accordingly – Sl. No. 3. The App may after sometime mark that ticket with QR State = 5-Invalid to prevent misuse.

**Table 5.7: QR Dynamic Data – Update QR Status at Real-time**

Sl. No.	Value	QR Status		
		Operator ID	Ticket Position	QR State
1	Decimal 34569 or 0x008709 Hex	135	1	1
2	Decimal 34570 or 0x00870A Hex	135	1	2
3	Decimal 34571 or 0x00870B Hex	135	1	3

**Table 5.8: QR Dynamic Data – Location**

Location – Latitude & Longitude			
Integer part (1 Byte)	Decimal part (2 Bytes)	Significand (3 Bytes)	Exponent
28	704	28704	$10^{-3}$
77	102	77102	$10^{-3}$

Latitude and longitude of India stretches from (8.438 to 37.627) and (68.773 to 97.254) respectively. We shall represent a latitude and longitude co-ordinate as follows taking an example co-ordinate (28.704, 77.102). We don't store the exponent but applications that need to display the co-ordinates must store it only up to 3 decimal places.



Table 5.9: QR Code Dataset – Ticket Block

QR Ticket Block								
TAG	Length	Size	L0 / Presence	L1	L2	L3	Element	Size
85		0 to 320 Bytes	Static Block / Optional	Primary Info**	–	–	QR Ticket Block – Personal Info	64
				Secondary Info**	Group*	Member1	-do-	64
						Member2	-do-	64
						Member3	-do-	64
						Member4	-do-	64
	Up to 640 Bytes	32 to 320 Bytes	Dynamic Block / Mandatory	Operator 1	–	–	Operator ID	2
					–	–	No of Tickets	1
					–	–	Validator Info	1
					Ticket Block*	Tkt Info 1	QR Ticket Block – Ticket	28
						...		...
						Tkt Info 10	-do-	28
				Operator 2	–	–	Operator ID	2
					–	–	No of Tickets	1
					–	–	Validator Info	1
					Ticket Block*	Tkt Info 1	QR Ticket Block – Ticket	28

						...		...
						Tkt Info 10	-do-	28
				...	...	...	...	...
				Operator 10	-	-	Operator ID	2
					-	-	No of Tickets	1
					-	-	Validator Info	1
					Ticket Block*	Tkt Info 1	QR Ticket Block – Ticket	28
						...		...
						Tkt Info 10	-do-	28

\* Please refer to accompanying notes below [Figure 3](#).

\*\*The entire Static Block is optional. Primary Info and Secondary Info are conditional quantities as described in [Important Note 1](#).

**Table 5.10: QR Ticket Block – Personal Info**

Personal Info (up to 64 bytes)			
Element	Data type / Presence	Size (Bytes)	Description
Name	Alphabetical / Mandatory	Up to 40	Full name of person. Space and dot ('.') are allowed within the name.
Age	Numeric / Mandatory	1	Age of person
Gender	Alphabet / Optional	1	Gender of person (M for Male, F for Female, N for Neutral, T for Transgender and O for Others)
ID number	Alpha-numeric / Optional	Up to 20	The ID proof number mentioned like the Aadhar Card no, PAN No, Passport No, etc.
ID type	Numeric / Optional	1	ID types are defined as follows: 0 = Invalid 1 = Aadhar Card 2 = PAN Card

			3 = Passport No . 4 = Others
RFU	Numeric / Optional	1	Reserved for Future Use

### Validator Info

Please refer to section [Terms and Definitions](#) for a description of Validator Info. It is a multi-purpose field whereby the most-significant nibble specifies the scanning capabilities of the Operator and the other nibble specifies whether that operator's ticket(s) are encrypted or not. This is essential because the Security Scheme employed in the QR Ticketing System may be a hybrid one wherein, some operator's tickets may be encrypted but the others' tickets may not be encrypted.

The scanning capability list of an Operator may be maintained in the Policy DB of the Ticket Generator. It essentially means the Operator has all those scanning options made available to the Customer. Camera scanning (Bit 4) is the default scanning capability.

**Table 5.11: QR Ticket Block – Validator Info**

Validator Info							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0 *	–	–	–	–
1 = WiFi	1 = BLE	1 = NFC	1 = Camera	RFU	RFU	RFU	0 – Unencrypted, 1 – Encrypted

*\*At least one of the bits 4 through 7 must be set. In other words, Validator Info cannot have a value less than 16.*

### Example Usage:

Value = 16 = 0001 0000b = 0x10 → Camera scanning; unencrypted ticket

Value = 49 = 0011 0001b = 0x31 → Camera and NFC scanning; encrypted ticket

Value = 160 = 1010 0000b = 0xA0 → WiFi and NFC scanning; unencrypted ticket

Value = 113 = 0111 0001b = 0x71 → Bluetooth, NFC and Camera scanning; encrypted ticket

### QR Ticket Block → Dynamic Block

The [Table 5.9](#) describes the ‘Dynamic Block’ of a QR Ticket. It is the most important portion of the QR Ticket and is like the heart of the QR Code Dataset. Only as many as can be 10 Operators can be accommodated in a QR Code and the number of available ‘Tickets’ is also 10. The table below defines one instance of a ‘Ticket’.

**Table 5.12: QR Ticket Block – Ticket**

Ticket		
Element	Data type / Presence	Size (Bytes)
Group Size** Refer <a href="#">Important Note 1</a> and <a href="#">Table 5.13</a>	Numeric / Mandatory	1
Source Station	Numeric / Mandatory	2
Destination Station	Numeric / Mandatory	2
Activation Date	Numeric / Mandatory	4
Product ID Refer <a href="#">Important Note 2</a> and <a href="#">Table 5.14</a>	Numeric / Mandatory	2
Service ID Refer <a href="#">Important Note 3</a> and <a href="#">Table 5.15</a>	Numeric / Mandatory	1
Total Fare	Numeric / Mandatory	4
Validity	Numeric / Mandatory	2
Duration	Numeric / Mandatory	2
Operator specific Ticket Data*	Numeric / Optional	8

Note: -Please refer to the [Terms and Definitions](#) section for detailed description of elements in a Ticket

\* The field “Operator Ticket Data” may be used by the Operator to write PTO-specific information. The Operator may provide API to TG which then the TG, during the process of ticket generation, can call so that this information gets written. Some Transit Operators contain the Route ID in their Tickets. Such information can go in this field. Since this field is optional, operators are not forced to fill up this area.

\*\* The field “Group Size” can be a little confusing and so we describe it below.

### Important Note 1: QR Ticketing System – Group Tickets

As seen in [Table 5.9](#), Secondary Info allows for up to 4 group members. But there can be *one and only one group* in a QR. For example, if the QR has 2 group tickets and one ticket uses up 3 out of the 4

available ‘Personal Info’ elements, then it is impossible for the other ticket to specify a separate group with the other remaining Personal Info element. Either it has to be the exact same group as the first group ticket or a completely new QR must be requested.

To understand group tickets, “Group Size” element is of utmost importance. It is possible to have all 10 tickets in a QR as group tickets; however, it is not mandatory to associate Personal Info of group members with those tickets. Only the field ‘A’ in the following table is mandatory. By default, in this implementation ‘group size’ is always equal to 1 i.e. single person or a Standard Product ticket (Refer [Table 5.14](#)). Many operators choose to omit personalized information altogether when issuing tickets. Even for group tickets, only the group size is specified. But when purchasing plane, rail, or hotel reservation group tickets, personalized information of group members may also be necessary. In such cases, associating the members’ personal information with the ticket becomes mandatory. This is how this is determined.

**Table 5.13: QR Ticket – Group Size**

Group Size (N = 1 byte)			
msb (C)	2nd msb (B)	6 least significant bits (A) Actual size of group	Description
0 or 1 [Default = 0] 0 implies that Secondary Info is absent. 1 implies Secondary Info is present	0 or 1 [Default = 0] 0 implies that the Primary Info is absent. 1 implies that the Primary Info is present in the Static Block	0 – Invalid 1 – 63 Default = 1 (single person)	–
0	0	0	Invalid
0	0	1 – 63	Valid – No Information in Static Block
0	1	1 – 63	Valid – Primary Info filled in Static Block
1	0	–	Invalid. It is not correct to have Secondary Info by leaving the Primary Info empty
1	1	1 – 63	Both Primary Info and Secondary Info are filled in Static Block

Field **A** signifies the group size i.e. the ticket is a Group Ticket if the value in this field is greater than 1.

Field **B = 1** implies that the Primary member's information is also provided.

Field **C = 1** implies that some or all the secondary member's information in the group is also provided. It is not possible to leave the Primary member's information empty and having any Secondary member information.

To illustrate the use of field together with the Secondary Info, the Validation rules for Group Size field are laid out here.

Assume that **N** represents the value in the Group Size field.

- **N = 0** is an invalid case since A cannot be 0.
- **If  $1 \leq N < 64$ , then:**  
It means  $B = 0$  and  $C = 0$ . It is the simple case of a single or group ticket without any personalized information. When  $N = 1$  it is a single person ticket.  
**Size of Group =  $A = N$**
- **N = 64** is an invalid case since A cannot be 0.
- **If  $N \geq 65$  and  $N \leq 127$ , then:**  
It means  $C = 0$ ,  $B = 1$  and  $A \geq 1$ . That implies it is either a single person ticket ( $N = 65$ ) or a group ticket and Personal Info of only the primary person is present in the Static block.  
**Size of Group =  $A = N - 64$**
- **N = 128 to N = 192** are invalid as it makes  $C = 1$ , but  $B = 0$ . Having Group member information without Primary member information is not allowed.
- **N = 193** is also invalid as it implies  $B = 1$  and  $C = 1$ , but  $A = 1$ . Having both primary and group member information for a group size of 1 ( $A = 1$ ) is not a valid case.
- **If  $N \geq 194$  and  $N \leq 255$ , then:**  
 $C = 1$ ,  $B = 1$  and  $A > 1$ . This implies this is a group ticket with personal info of both the Primary and Group members are present.  
**Size of Group =  $A = N - 192$**

For a few illustrative examples of some combinations of group and single person tickets in JSON format, please refer to [Appendix – I \(MxN\\_QR\)](#).

### Important Note 2: QR Ticketing System – Product ID – Group Tickets, Passes, Trips

Passes are an integral part of any transit application. PTOs of all types – Bus or Metro operators – often issue many passes like – Senior Citizen Pass, Student Pass, Day Pass, Monthly Pass, etc. In the QR Ticketing System, the Product ID parameter is used to populate Passes. There may be many Passes issued by the PTO and the Product ID field should be used accordingly. Product ID is a 2-byte field – decimal value range 0-65535.

The user must specify the exact date of activation from when the Pass is desired to be used. Products usually have a Validity period. In the QR Ticketing System, the fields “Activation Date” and “Validity” shown in Table 5.12 must be used to maintain date of journey and validity period respectively. The validity is calculated from the date and time mentioned in the Activation Date. Since Validity is a 2-byte field, the maximum possible Validity of Pass in the QR Ticketing System is as follows:

Validity max value = 65535 minutes

Product ID max validity period =  $65535 \div 60 = 1092.25$  hours = 45.5 days.

For passes the maximum value of the Validity field shall be fixed at  $45.5 \times 24 \times 60 = 65520$

Default maximum Validity of every Product or Pass is actually PTO-specific and can be included as a Policy DB parameter. This is discussed in detail in the Section 5.1 on Policy DB parameters in QR Specifications – Part III.

Senior Citizen, Student, Special, Defense and Child Ticket Products or Passes are not as straightforward as the other passes. There is a mandatory requirement of Verification required before these can be used. Please refer to Section 4.1.6 of the QR Ticketing System – Implementation Guidelines for a simple way to implement these products.

**Table 5.14: QR Ticket – Product ID**

Product ID (N = 2 bytes)		
Value (Decimal)	Pass Type	Description
0	Not used	Invalid – must not be used
1	Standard Product	This is the Default Value. Standard Journey Ticket
2	Senior Citizen Pass	Pass for Citizens above the age of 60 with a Validity period of 65520 minutes or 45 days
3	Student Pass	Pass for Students with a Validity period of 65520 minutes or 45 days.
4	Daily Pass	A pass that can be used as a daily ticket from any stop to any stop. The Validity period of this ticket shall be one day only unless the period is explicitly specified by the user in the Validity field.
5	Monthly Pass	A monthly pass that can be used for a period of one month or 43200 minutes

6	Weekend Pass	A pass that would be valid only on Saturdays and Sundays. The Validity period of this ticket shall be 45 days unless the period is explicitly specified by the user in the Validity field.
7	Fixed Station Pass	This Pass is only Valid for the exact Source and Destination stations. The Validity period of this ticket shall be 45 days unless the period is explicitly specified by the user in the Validity field.
8	Defense Personnel Pass	Pass for Armed Forces Personnel with a Validity period of 65520 minutes or 45 days.
9	Special Pass	This Pass is only for use by Differently abled persons. The Validity period of this ticket shall be 45 days unless the period is explicitly specified by the user in the Validity field.
10	Discounted Single Journey	May be used for providing Single Journey Discounted fares. E.g. can be provided for Women, Differently Abled Persons, Children, etc.
11	Trip Pass	May be used for providing a Pass for multiple trips. Trip count value may be filled up in the second byte of this field. Maximum value of trip count is 255.
12	Child Ticket	If a Ticket needs to be marked as Child Ticket i.e. for Minors – children below the age of 18.
13-100	RFU	May not be used. Reserved for future use
101-255	PTO-specific	May be used if PTO want to define more passes for it implementation
256 - 65535	Should only be used for special cases.	Only the LSB must be used for defining “Product ID”. This is the second byte or the MSB. This byte may be made use of in some way if an implementation chooses to. For example, if the ‘Trip Count Pass’ (Product ID = 11) is used, this byte may be used to fill in the trip count value.

### Important Note 3: QR Ticketing System –Service ID – Group Tickets, Passes, Trips

The Service ID field generally specifies the types of services offered by the PTO. For example, a Bus operator may have an AC Bus service or an Express Bus service or a General Fare bus service. The field is a 1-byte field and the Services offered depends on the PTO and must be sent as a Policy DB parameter. However some services are fixed as shown in the [Table 5.15](#) below. Please refer to Section 5.1 of QR Specifications – Part III for definitions and procedures of updating the Policy DB in TG. The general Service ID definitions are described in the table below as follows:



**Table 5.15: QR Ticket – Service ID**

Service ID (N = 1 bytes)		
ID	Service Name	Description
0	Invalid	Not to be used
1	Regular Metro	Regular or Normal Metro Line Service
2	Express Metro	Express Metro Line Service
3	Regular City Bus	Normal Fare Non-AC City Bus Service
4	AC City Bus	AC Fare City Bus Service
5	Regular Inter-City Bus	Normal Inter-city (within state) bus service
6	Express Inter-City Bus	Express Inter-city (within state) bus service
7	Non-stop Inter-City Bus	Non-stop Inter-city (within state) bus service
8	Regular Inter-State Bus	Normal Inter-state bus service
9	Express Inter-State Bus	Express Inter-state bus service
10	Non-stop Inter-State Bus	Non-stop Inter-state bus service
11	Local Train	This is the Indian Railways service used in many cities and towns
12	Sleeper class	Low-fare Indian Railways Service
13	AC 2-tier	High-fare Indian Railways Service
14	AC 3-tier	High-fare (lower than 2-tier AC) Indian Railways Service
15	AC First Class	First Class AC Service
16	Normal First Class	Regular First Class Service
17	AC Chair Car	AC Chair Car Service
18	Normal Chair Car	Regular Chair Car Service
19	Rapid Rail	Modern Rapid Rail Services introduced in big cities

20	Mono Rail	Modern Mono Rail Services introduced in big cities
21	Normal Feeder Bus	Regular Feeder Bus service that is offered for last mile connectivity by many operators
22	Luxury Feeder Bus	AC Feeder Bus service that is offered for last mile connectivity by many operators
23	Regular Hatchback Cab Service	Normal Cab or Taxi service that is offered for last mile connectivity by many operators
24	Shared Cab Service	Shared Cab or Taxi service that is offered for last mile connectivity by many operators
25	Sedan Cab Service	Sedan Cab or Taxi service that is offered for last mile connectivity by many operators
26	Normal Self-Parking Lot	Regular Parking Lot service offered by many operators for its customers to park their private vehicles and continue on public transport
27	Valet Parking Lot	Extra-fee Parking Lot service offered by many operators for its customers to park their private vehicles and continue on public transport
28-99	RFU	Reserved for Future Use
100-255	PTO-specific Service	If the PTO needs to define some service of its own it may use this range

For a few illustrative examples of some combinations of using Service ID in tickets JSON format, please refer to Appendix – I (MxN\_QR).

## 6. Operator Type-specific QR Tickets

It is essential to test the QR Dataset for its readiness in Service-specific applications – e.g. with Operator type as Metro, Bus, Rail, Airplane, etc. as trials for such operators are of higher priority than other operators like hotel or parking, museum or the like. The importance needs to be tested in terms of (a) Completeness of a ticket and (b) Ease of validation for operators.

**Table 6.1: Operator Ticket Block for Transit Applications**

Operator Type	Operator Ticket Block Fields of importance in Travel	Description and Recommendations
Metro and Bus	Name	These fields are usually not of any importance for Metro or Bus travel and should be left empty.
	Age	
	Gender	
	ID Number	
	ID Type	
	Group Size	<ul style="list-style-type: none"> <li>These fields are the main component of a bus or metro ticket and are sufficient to describe a metro or bus journey.</li> <li>Duration can be specified as the period for which the ticket remains valid after a QR has been scanned.</li> <li>Validity field is essential as it specifies for how long a ticket is valid on the activation date. For buses and metro, the ticket is usually a day ticket and the service can be availed at any time on that date from the time of journey.</li> <li>Fare is the total fare or purchase amount of the ticket.</li> <li>Operator Ticket Data is used by the Operator to write operator-specific information.</li> </ul>
	Source Station	
	Destination Station	
	Activation Date	
	Duration	
	Validity	
	Fare	
	Operator-specific Ticket Data	
Rail and Airplane	Name	<ul style="list-style-type: none"> <li>When it comes to air or rail travel, all these fields are of utmost importance. Personal Info is always checked and verified before embarking on a journey.</li> <li>In air and rail travel, ticket is a confirmation but not a guarantee for reservation. Railways provide reservation lists on the date of journey and Airplanes provide boarding passes. Either of these is not part of the ticket itself and hence is not included. That information can be maintained as QR Status information which can be communicated with the APP as and when required.</li> <li>Validity field can essentially mean the time for a full refund in case of a missed trip.</li> </ul>
	Age	
	Gender	
	ID Number	
	ID Type	
	Group Size	
	Source Station	
	Destination Station	
	Activation Date	
	Duration	
	Validity	
	Fare	

## QR Ticketing System – QR Code Dataset

	Operator-specific Ticket Data	<ul style="list-style-type: none"> <li>• Duration of a journey shall be useful in air travel and for rail journey as it always contains the duration of the journey.</li> <li>• Fare is the total fare or purchase amount of the ticket.</li> <li>• Operator-specific Ticket Data is used by the Operator to write operator-specific information.</li> </ul>
--	-------------------------------	--

## 7. QR Image Properties

### 7.1. QR encoding method

Universally all QR codes can support any one of the modes specified in the table below. Current specification mandates only binary mode encoding technique in QR Ticketing System.

- Naturally Kanji cannot be used as it is a multi-byte Japanese character script.
- Numeric also cannot be used as many fields like name, ID No, etc. have alphabets.
- Numeric and Alphanumeric can hold up to 6K bytes and 4.2K bytes respectively for version 40, ECC = M as opposed to Binary mode = 3K bytes. However, as these modes does not support the QR Ticketing implementation because QR payload contains encrypted information and digital signature which has special characters outside the Ascii character set.

For details on QR Code features please refer [\[2\]](#).

**Table 7.1: QR Encoding Scheme**

Mode	Supported
Numeric	No
Alphanumeric	No
Binary	Yes
Kanji	No

### 7.2. QR Version No

Universally standard QR versions run from 1 to 40. Version 1 starts with  $21 \times 21 = 441$  modules and each version increases by 4. Hence, the last version 40 is  $177 \times 177 = 31329$  modules. Higher the version number of a QR image, higher the content to hold. So a Version 40 QR can hold up to 3KB data, including the control information.

#### What Version must be chosen?

It is not recommended to freeze upon a version. Let the generator algorithm decide what the best version is. Modern QR generator libraries help to encode a QR Code as efficiently as possible. However, since the scanning times vary with the size of the QR, the Operator must study the calculated sizes of the full range of QR tickets supported by the QR Ticketing System in Specifications – Part II to decide on the QR dataset best suited for its implementation. Please note here though that the QR Dataset described in this document does not affect scannability as proved by the Scan Test results shown in Appendix-II of Part II – QR Security Scheme.

For printed QRs, if the size has been decided and the scanning distance is also known, then of course one can choose the version and supply that to the QR encoding algorithm.

### 7.3. Scannability

Scannability is the ability to scan and interpret a QR within the desired time interval.

For a QR to be decoded correctly each and every module in the QR must be read. To calculate the right QR Code size for the use case, two things must be considered:

#### A. Scanning Distance:

It is a question of where to put the QR Code and what will be the approximate scanning distance? Depending on this, one can calculate the QR Code size using this very basic standard rule i.e.

$$\text{QR code minimum size} = (\text{Scanning Distance}) / 10$$

Where, size and scanning distance are widths in inches or cm.

It means a **1-inch x 1-inch** image can be accurately scanned from a distance of  $\leq 10$  inches. *In most practical scenarios however, this rule is not very useful at all.*

#### B. QR Version and Size formula

If one knows the size of the QR Code to be printed, then to find the version the advanced thumb rule can be used. This rule might be quite useful for Paper QRs.

$$\text{QR code minimum size} = (\text{Scanning Distance} \times \text{QR Version}) / 250$$

E.g. If size = 3 inch & distance = 6 inch; then rows, cols = 125 or Version 27.

### 7.4. Printing specifications

#### A. Screen or mobile

- **Resolution:** Resolution can be defined as the size of the dots or pixels in the code. The more information a code contains, the smaller the size of the dots become. Moreover, the dots in the code must be large enough to be detected by the scanner.
- **File Format:** Most QR encoding programs like the ones in smartphone apps generates QR in raster format i.e. PNG or JPEG. PNG have generally become the de-facto accepted format as it provides minimal or no loss of data, even when zoomed out.
- **QR Min Size:** The general recommendation for QR code size is that it should never be smaller than 1.3 inches. But in this system the QR ticket information is quite dense having a byte size near about 2Kb. So a 1.3 inch QR is too small to hold that much information. A good recommendation is use a QR of size at least 200px or about 1.75 inch. Please see example [Section 7.6](#).

## B. Printing Paper

- **Contrast:** The first essential requirement for QR printing is to have a good contrast between background and foreground. Black prints over a White or Cream background should serve the purpose well.
- **QR Margin:** The margin is a clear area around a symbol where nothing is printed. QR Code requires a four-module wide margin at all sides of a symbol.
- **Paper Quality:** Paper quality should be rigid and stiff enough to hold between the thumb and the other fingers. A good benchmark would be an airplane boarding pass ticket (BPT1-standard given at the counter). Good quality BPT paper varies from 100 to 250 GSM.

## C. Devices

- **Printers:** The best option for printers is BPT printers available in Parallel port / Ethernet / RFID modes. BPT printers can print high quality Bar/QR codes on 20-250 GSM papers. Such printers are used in airports all over the world. But then they would also require high resolution scanners. If a normal printer is used it is to be ensured that at least 400 dpi or 400 dp25.4 mm resolution where 1 inch = 25.4mm
- **Scanners:** A good quality Standard Type scanner should be the very minimum requirement for a scanner. We saw in the example that our modules were sized at 0.254mm. So our scanner has to have a resolution higher than that – anything between 0.2 to 0.25 mm

## 7.5. QR Error Correction Code Scheme

There are four ECC schemes viz. L, M, Q and H. All the schemes have different error correction capability which provides data recovery from damaged QR image as defined in the table below. Higher the error correction capability, higher the overhead (denser) and takes less input data as referred in the table.

This specification does not mandate any specific error correction schemes which may be designed by system manufacturer or operators. *However, it is recommended to use error correction scheme “L” or “M” as a good compromise between density and recoverability.*

For more details on Error Correction Scheme, please refer [\[3\]](#).

**Table 7.2: Error Correction Scheme vs. Overhead**

ECC Scheme	Error Correction Level in percentage of damage	Overhead in Comparison to ECC-L scheme	Density (Fuzzy qualifiers)
L	7%	0%	Not dense
M	15%	15%-30%	Somewhat dense

Q	25%	45%-80%	Very dense
H	30%	100%-135%	Extremely dense

## 7.6. Illustrative example for a Paper Ticket

**Ticket Size:** The dimensions for an airplane boarding pass are 3 x 7 inch, but our recommendation for the ticket dimension is 2.5 x 3.5 inch with the QR displayed on the top and the text below it as shown below. Another recommendation is to print a ticket rounded to 3 x 4 inch.

### Assume:

Printer resolution= 400(dpi or dp25.4 mm) or 4 dots per 0.254mm

QR Data size = 650 bytes (binary)

ECL = M (standard < 15% correction)

Minimum Version = 20 (97 x 97)

Margin size =  $8 \times 0.254 \sim 2$  mm or 1 mm on either side

Scanning distance = 6 inches or ½ feet

Now using the advanced thumb rule given in [Section 7.3 → B](#)

QR print size =  $(6 \times 97) / 250 = 2.3$  inches

Some margin (0.2 inches) is kept on all sides.

We can print the ticket text like Serial No, Name (if present) and journey stations, etc.in small size, about 9 or 10 dpi.





## 8. References

S. No	References
1.	RBI transaction Standard- <a href="https://rbidocs.rbi.org.in/rdocs/RTGS/PDFs/RTGSB111013_2013.pdf">https://rbidocs.rbi.org.in/rdocs/RTGS/PDFs/RTGSB111013_2013.pdf</a>
2.	<i>"QR Code features"</i> - QR encoding scheme
3.	ISO/IEC 18004-2015 – QR standard Definition , Error correction levels, etc. – <a href="https://www.iso.org/obp/ui/#iso:std:iso-iec:18004:ed-3:v1:en">https://www.iso.org/obp/ui/#iso:std:iso-iec:18004:ed-3:v1:en</a>
4.	QR Specifications – Parts II & III

## Appendix – I (MxN\_QR)

### QR Dataset with Multiple Operators and Multiple Tickets

The QR Ticket Block as shown in [Figure 3](#) is a pool of only 10 tickets. These 10 tickets must be shared among the pool of 10 Operators. Considering the fact that one operator can provide more than one service and each service can have multiple products, the Ticket Pool can become a very complex structure with a plethora of different combinations. The present structure has been branched only up to the level of Operator ID; with Product ID and Service ID fields being flattened within the ticket itself.

A QR ticket must have at least 1 ticket+1 Operator and can have up to 10 tickets+10 Operators and everything in between. So how many of these combinations are possible? For example, if all 10 tickets are used from the pool among how many operators can those tickets be divided? This is considered a hard problem in mathematics and is known as the “Partitioning Problem” in number theory. There are no formulas to deal with this problem, but only approximations – a famous one was given by the genius Indian mathematician of the 20<sup>th</sup> century, Srinivasa Ramanujam. Smaller partitions up to 500 or so have so far been calculated with software programs.

### QR Ticket Pool Denotation –MxN\_QR

We devise a way to denote the number of Operators & Tickets in a QR Ticket as follows:

Let M = No. of Operators in a QR Ticket;  $1 \leq M \leq 10$

And N = No. of Tickets;  $1 \leq N \leq 10$

The convention to denote a QR, to be used throughout this specification, is **MxN\_QR**.

For example, if a QR Ticket is denoted as **5x7\_QR**, it means the QR is for 5 operators and consists of 7 tickets. Obviously, out of the 5 operators, one or two of them will have more than 1 ticket within its block. But it is not possible to tell immediately which operators have more than ticket. This can only be determined after reading the ‘No of Tickets’ field.

When  $M = N$ , then each operator can have only 1 ticket each and furthermore M can never be greater than N.

The table below shows the total number of partitions of tickets. Partitions are not permuted so there is no difference between 1+2 and 2+1 and so on.

Table AN 1: MxN\_QR Partitions

MxN_QR Partitions				
No. of Tickets (N)	Possible No. of Operators (M)	Ticket Arrangements (Tickets per Operator)	Denotation (MxN)	Total No. of combinations
1	1	1	1x1	1
2	1, 2	2, 1+1	1x2, 2x2	2
3	1, 2, 3	3, 1+2, 1+1+1	1x3, 2x3, 3x3	3
4	1, 2, 3, 4	4, 3+1, 2+2, 2+1+1, 1+1+1+1	1x4, 2x4 (2 lots), 3x4, 4x4	5
5	1, 2, 3, 4, 5	5, 4+1, 3+2, 3+1+1, 2+2+1, 2+1+1+1, 1+1+1+1+1	1x5, 2x5 (2), 3x5 (2), 4x5, 5x5	7
6	1, 2, ... , 6	...	...	11
7	1,2, ... , 7	...	...	15
8	1, 2, ... , 8	...	...	22
9	1, 2, ... , 9	...	...	30
10	1, 2, 3, 4, ... , 10	10, 1+9, 2+8, ... , 1+1+1+1+...+1	1x10, 2x10 (5), 3x10 (8), 4x10 (9), 5x10 (7), 6x10 (5), 7x10 (3), 8x10 (2), 9x10, 10x10	42

#### Ticket Request JSON files:

Request for QR Ticket from the APP or from TOM must be sent in the form of JSON or XML objects. *It is recommended to use JSON.* QR Request and Response scenarios are described in great detail in Specification Parts II and III. Here we show some simple examples of QR tickets – like a **2x3\_QR** Ticket and a couple of **1x1\_QR** tickets to illustrate the use of groups, personal information and products and services.

```
{
  "Requester_ID": "15",
  "TXN_Ref_No": "ABCDRC2020050788888123",
  "Language": "0",
  "TXN_Type": "65",
  "TXN_Date": "12/06/2020@18-47-57",
  "PSP_Specific_Data": "Mode=IMPS;ServiceFee=1$",
  "Total_Fare": "240",
  "Customer_Mobile": "9201345678",
  "TicketBlock": {
    "Dynamic_Block": {
      "OperatorID1": {
        "OpID": "10",
        "NoOfTickets": "2",
        "Validator_Info": "16",
        "TicketInfo": {
          "TicketInfo1": {
            "Grp_Size": "1",
            "Src_Stn": "06",
            "Dest_Stn": "24",
            "Activation_Date": "24/06/2020@18-20-45",
            "Product_Id": "01",
            "Service_Id": "01",
            "Ticket_Fare": "45",
            "Validity": "480",
            "Duration": "180"
          },
          "TicketInfo2": {
            "Grp_Size": "1",
            "Src_Stn": "24",
            "Dest_Stn": "06",
            "Activation_Date": "24/06/2020@20-30-00",
            "Product_Id": "01",
            "Service_Id": "01",
            "Ticket_Fare": "45",
            "Validity": "480",
            "Duration": "180"
          }
        }
      },
      "OperatorID2": {
        "OpID": "135",
        "NoOfTickets": "1",
        "Validator_Info": "16",
        "TicketInfo": {
          "TicketInfo1": {
            "Grp_Size": "1",
            "Src_Stn": "25",
            "Dest_Stn": "47",
            "Activation_Date": "27/06/2020@15-30-30",
            "Product_Id": "01",
            "Service_Id": "03",
            "Ticket_Fare": "150",
            "Validity": "240",
            "Duration": "90"
          }
        }
      }
    }
  }
}
```

**Figure 5: Sample 2x3\_QR Ticket**

Here for the first operator, ID = 10 the operator is shown as Metro operator. The first ticket has Service ID = 0 and the second ticket has Service ID = 1. Recall from [Table 5.15](#), it means the first one is a ticket for a Normal Metro Line and the second one is for an Express Metro Line.

For the second operator, ID = 135 the operator is shown as Bus operator. The only ticket has Service ID = 2. From [Table 5.15](#), it means it is a Regular City Bus ticket. The Product ID for all the tickets is 1. From [Table 5.14](#), it means a Standard Product.

```
{
  "Requester_ID": "15",
  "TXN_Ref_No": "ABCDRC2020050788888123",
  "Language": "0",
  "TXN_Type": "65",
  "TXN_Date": "12/06/2020@18-47-57",
  "PSP_Specific_Data": "Mode=IMPS;ServiceFee=1%",
  "Total_Fare": "45",
  "Customer_Mobile": "9201345678",
  "TicketBlock": {
    "Static_Block": {
      "Primary_Info": {
        "Name": "Ravi Kumar",
        "ID_No": "AENPC9863L",
        "ID_Type": "02",
        "Age": "20",
        "Gender": "M"
      }
    },
    "Dynamic_Block": {
      "OperatorID1": {
        "OpID": "10",
        "NoOfTickets": "1",
        "Validator_Info": "16",
        "TicketInfo": {
          "TicketInfo1": {
            "Grp_Size": "65",
            "Src_Stn": "06",
            "Dest_Stn": "24",
            "Activation_Date": "24/06/2020@18-20-45",
            "Product_Id": "01",
            "Service_Id": "01",
            "Ticket_Fare": "45",
            "Validity": "480",
            "Duration": "180"
          }
        }
      }
    }
  }
}
```

**Figure 6: Sample 1x1\_QR Ticket with Personalized Info**

In this example, it is seen that Grp\_Size = 65. Recall from [Table 5.13](#), it means A = 1, B = 1 and C = 0. In binary representation 65 = 0b01000001. Since B = 1, it implies that the Personal Info of the primary person is included in the QR Ticket seen under the “Static Block” element.

```
{
  "Requester_ID": "15",
  "TXN_Ref_No": "ABCDRC2020050788888123",
  "Language": "0",
  "TXN_Type": "65",
  "TXN_Date": "12/06/2020@18-47-57",
  "PSP_Specific_Data": "Mode=IMPS;ServiceFee=1%",
  "Total_Fare": "90",
  "Customer_Mobile": "9201345678",
  "TicketBlock": {
    "Static_Block": {
      "Primary_Info": {
        "Name": "Ravi Kumar",
        "ID_No": "AENPC9863L",
        "ID_Type": "02",
        "Age": "20",
        "Gender": "M"
      },
      "Secondary_Info": {
        "Member1": {
          "Name": "Veera Rakum",
          "ID_No": "BENPC9863M",
          "ID_Type": "02",
          "Age": "41",
          "Gender": "M"
        }
      }
    },
    "Dynamic_Block": {
      "OperatorID1": {
        "OpID": "10",
        "NoOfTickets": "1",
        "Validator_Info": "16",
        "TicketInfo": {
          "TicketInfo1": {
            "Grp_Size": "194",
            "Src_Stn": "06",
            "Dest_Stn": "24",
            "Activation_Date": "24/06/2020@18-20-45",
            "Product_Id": "01",
            "Service_Id": "01",
            "Ticket_Fare": "45",
            "Validity": "480",
            "Duration": "180"
          }
        }
      }
    }
  }
}
```

**Figure 7: Sample 1x1\_QR Group Ticket with Personalized Info of 2 members**

In this example, it is seen that Grp\_Size = 194. Again recall from [Table 5.13](#), it means A = 2, B = 1 and C = 1. In binary representation 194 = 0b11000010. Since B and C are both 1, it implies that the Personal Info of both Primary and Group members are included in the QR Ticket as seen in the “Static Block”.

## Appendix – II (QR Versioning)

### QR Versioning: -

In our QR Ticketing System only binary mode will be supported. Binary mode does not automatically support UTF-8 (multi-byte) encoding, but in our system we will add support for UTF-8 characters. The following table shows the character capacities of binary encoding w.r.t. the Error Correction Levels.

**Table AN 2: Character Capacities in Binary Mode**

QR Version	L	M	Q	H
1	17	14	11	7
2	32	26	20	14
3	53	42	32	24
4	78	62	46	34
5	106	84	60	44
6	134	106	74	58
7	154	122	86	64
8	192	152	108	84
9	230	180	130	98
10	271	213	151	119
11	321	251	177	137
12	367	287	203	155
13	425	331	241	177
14	458	362	258	194
15	520	412	292	220
16	586	450	322	250
17	644	504	364	280
18	718	560	394	310
19	792	624	442	338
20	858	666	482	382
21	929	711	509	403
22	1003	779	565	439
23	1091	857	611	461
24	1171	911	661	511
25	1273	997	715	535
26	1367	1059	751	593
27	1465	1125	805	625
28	1528	1190	868	658
29	1628	1264	908	698
30	1732	1370	982	742
31	1840	1452	1030	790
32	1952	1538	1112	842

# QR Ticketing System – QR Code Dataset

33	2068	1628	1168	898
34	2188	1722	1228	958
35	2303	1809	1283	983
36	2431	1911	1351	1051
37	2563	1989	1423	1093
38	2699	2099	1499	1139
39	2809	2213	1579	1219
40	2953	2331	1663	1273



## Appendix – III (QR Dataset Language Codes)

Language Codes as per the Govt. of India VIII Schedule

**Table AN 3: Official Languages of India**

Sl. No.	Language	Code (1 byte)
1.	English	0
2.	Hindi	1
3.	Bengali	2
4.	Marathi	3
5.	Telugu	4
6.	Tamil	5
7.	Gujarati	6
8.	Urdu	7
9.	Kannada	8
10.	Odiya	9
11.	Malayalam	10
12.	Punjabi	11
13.	Sanskrit	12
14.	Assamese	13
15.	Maithili	14
16.	Santali	15
17.	Kashmiri	16
18.	Nepali	17
19.	Sindhi	18
20.	Dogri	19
21.	Konkani	20
22.	Manipuri	21

QR Ticketing System – QR Code Dataset

23.	Bodo	22
24.	RFU	23-255

\*\*\*\*\* End of Part I \*\*\*\*\*

## **Part II: Security Scheme – QR Ticketing System for Transit Applications**

## Contents

1. Introduction .....	5
2. Acronyms and Abbreviations .....	6
3. Terms and definitions .....	6
4. Security Algorithms – Levels, definitions and implementation .....	8
5. SecureQR – Securitization of QR Codes in QR Ticketing System .....	10
5.1. SecureQR-Alpha .....	10
5.1.1. Phases of SecureQR-Alpha .....	11
5.1.1.1. One-Time Phase Activities .....	11
5.1.1.2. Repeating Phase Activities .....	12
5.1.2. SecureQR-Alpha Algorithm Design .....	24
5.1.3. SecureQR-Alpha Dataflow Example .....	26
6. References .....	34
Appendix I – SQDSR Format .....	35
Appendix II – QR Code Scan Times .....	37
Appendix III – Cryptography Theoretical Definitions .....	45
Appendix IV – Simple Secret Key Sharing Mechanism .....	48

## List of Figures

Figure 1: SecureQR-Alpha Security Level I/O Diagram .....	10
Figure 2: SecureQR-Alpha Component-Attribute Model.....	11
Figure 3: Ticket Generator – CA – Operator Chain of Trust.....	12
Figure 4: Producing QR-friendly text.....	13
Figure 5: QR Ticket Generation Flowchart.....	24
Figure 6: QR Ticket Validation Flowchart.....	25
Figure 7: 2x2_QR Ticket Request JSON File .....	27
Figure 8: Specifications Mitsunami Part Number M/USBOV5640V3 .....	38
Figure 9: Specifications Zebra MS4717 Fixed Mount Imager .....	39
Figure 10: Best Case Scan Times for 1x1_QR through 1x10_QR .....	43
Figure 11: Worst Case Scan Times for 1x1_QR through 1x10_QR.....	43
Figure 12: Best & Worst Case Validation Times 1x1_QR through 1x10_QR on HID A.....	44

## List of Tables

Table 4.1: Security Level Spectrum.....	8
Table 4.2: QR Security Element.....	9
Table 5.1: SQDSR Size with Overhead for QR Dataset.....	15
Table 5.2: QR Code Security Plaintext Size .....	16
Table 5.3: QR Code Dataset Version Plaintext Size.....	16
Table 5.4: QR Code Common Data Block Plaintext Size .....	16
Table 5.5: QR SVC Size with Overhead.....	17
Table 5.6: QR Code Dynamic Data Plaintext Size.....	17
Table 5.7: QR Code Static Block Plaintext Size.....	17
Table 5.8: QR Ticket Block Plain & Cipher Text Size.....	18
Table 5.9: Unencrypted QR Payload Minimum and Maximum Size .....	19
Table 5.10: QR Versions for Unencrypted QR Tickets with Digital Signature .....	20
Table 5.11: Encrypted QR Payload Minimum and Maximum Size.....	20
Table 5.12: QR Versions for Encrypted QR Tickets .....	21
Table 5.13: QR Code Ciphertext Message.....	22
Table 5.14: QR Data and Image Rendering.....	32

**Version History**

Date	Version	Author	Comments
18/07/2020	1.0	CDAC	Changes to document after external stakeholder review comments. Final Draft release is being prepared. Changes mostly in sizes as new elements have been added to dataset. Version circulated for external stakeholders' review by MoHUA.
14/02/2021	1.1	CDAC	Amendments related to QR Code Dataset owing to comments from external stakeholders on V1.0. Also incorporated comments and suggestions received from participants in Workshop conducted by CDAC on 22 <sup>nd</sup> & 23 <sup>rd</sup> Dec 2020.

## 1. Introduction

In today's world of hi-tech and advanced smart phones and IoT devices (all things connected to the Internet), security of embedded systems is of paramount importance for all organizations. Additionally, due to the involvement of sensitive and confidential information that travels through these systems and information highways makes it imperative for providers to deploy state-of-the-art security measures. In order to offer smart services to their customers, service providers often have to deploy large numbers of complex embedded systems on public networks. That indirectly makes those systems potential targets of vicious security attacks. Protecting the customer's private data is entirely the service provider's responsibility and must be done so by employing adept security measures to protect both data and systems that access the data.

This Part II of the QR Specifications discusses in detail the measures undertaken to secure the QR Ticket. It is implemented from the point of view of the following four tenets of cryptography –

- **Authentication** – A signed content or digital signature can only be verified using the data originator's public key
- **Integrity** – The ticket generator creates the digital signature and no one can alter the content during transit so integrity of data is maintained
- **Non-repudiation** – The signature is generated by the sender so authorship of the signature cannot be disputed or denied
- **Confidentiality** – The ticket generator encrypts the message before transmitting it thereby protecting the privacy of the message content

Digital Signature takes care of three aspects – authenticity, integrity and non-repudiation, but not confidentiality unless the message is encrypted explicitly.

## 2. Acronyms and Abbreviations

AES	Advanced Encryption Standard
HMAC	Keyed-Hash Message Authentication Code
KMS	Key Management System
MAC	Message Authentication Code
PKCS	Public Key Cryptography Standard
QR Code	Quick Response Code
RFU	Reserved for Future Use
RSA	Rivest-Shamir-Adleman algorithm
SHA	Secure Hash Algorithm
SQDSR	<b>SecureQR DataSet Representation</b> format
TOM	Ticket Operating/Office Machine
TRM	Transit Rules/Risk Management

## 3. Terms and definitions

The following definitions are used throughout this document:

- **QR Code:** Quick-response code. Used for Single journey or Group tickets and shall be available in both smart phones and paper media. For details about QR technologies please refer [\[2\]](#).
- **Key:** A cryptographic key is a string of data (mostly numeric encoded) that is used to lock or unlock cryptographic functions. The entire process of information security using cryptography is heavily based on usage of cryptographic keys
- **Secret key:** *Secret-key* refers to data keys that are either entirely private or kept as a “shared secret” among trusted parties. Sharing secret keys is often considered problematic in security schemes. An elegant method of sharing secrets is shown in [Appendix IV – Simple Secret Key Sharing Mechanism](#).
- **Hash function:** A mathematical function that maps a string of arbitrary length (up to a predetermined maximum size) to a fixed length string.
- **Hash value:** The result of applying a hash function to data called a message digest. A hash value is sometimes also referred to as a digest.



- **Symmetric cryptography:** In this form of cryptography, the sender and receiver share the same key (shared secret) to encrypt and decrypt messages.
- **Asymmetric cryptography:** In this form of cryptography, the sender and receiver use separate keys for encryption and decryption. This form of cryptography takes advantage of mathematical functions like totients and modulus to form relationship between the sender and receiver's keys.
- **Totient function:** In number theory, totient function, specifically Euler's Totient Function [7] counts the positive numbers up to a number 'n' that are relatively prime to 'n'. It is denoted by the  $\phi(n)$ . Totient functions are widely used in RSA-based algorithms (asymmetric algorithm).
- **Modular multiplicative inverse:** In number theory, a modular multiplicative inverse [8] of an integer A is an integer X such that the product A.X is congruent to 1 w.r.t. the modulus m. Modular multiplicative inverse is widely used in RSA-based algorithms (asymmetric algorithm).
- **Base64 encoding and decoding:** Base64 is only an encoding-decoding technique and has nothing to do with security. In the QR Ticketing System, all messages sent from the TG will be Base64 encoded. The character set of Base64 is [A-Z], [a-z], [0-9], [+/-]. It increases size of encoded data by 33.3% and adds some extra padding bytes. Most Base64 encoding algorithms also insert a newline after the 76<sup>th</sup> encoded byte and that option should be preferably disabled before encoding. In this document, "plaintext" and "Base64 text" have been used interchangeably.
- **Root Certificate Authority (CA):** The root CA is an entity that is the "root of trust" in a PKI implementation and is responsible for authenticating identities in the ecosystem.
- **Intermediate CA:** Also called the subordinate CA, the intermediate CA is certified by a root CA for specific uses as defined by the certificate policy. Digital certificates are typically issued and signed by sub-CAs.
- **Certificate Policy:** Essentially, the certificate policy is the security specification that defines the structure and hierarchy of the PKI ecosystem, as well as policies surrounding the management of keys, secure storage and handling of keys, revocation, and certificate profiles/formats, among other details.
- **X.509:** In cryptography, X.509 is a standard defining the format of public key certificates. X.509 certificates are used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure protocol for browsing the web.
- **Certificate Database:** The certificate database stores certificate records.
- **Revocation Services:** Revocation services are servers that post updated Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP) responders that use CRLs and respond to revocation lookup checks for devices that cannot process CRLs themselves.
- **Digital Certificate:** The digital certificate is a "digital identity" embedded in a device that provides secure authentication capability between devices and servers, allowing access to services, files or other remote resources. This is typically issued by the sub-CA.

- **Ciphertext:** In the QR Ticketing System, the term “Ciphertext” refers to the response message that the Ticket Generator sends for a Ticket Request. It does not imply that the message is completely secure – it may be plaintext, partially encrypted or fully encrypted depending on the chosen security scheme.

## 4. Security Algorithms – Levels, definitions and implementation

In order to classify the security measures employed in a particular QR Ticketing System implementation w.r.t. the four tenets of security, a system of security level spectrum is introduced here. This spectrum denotes the strength or fragility of the security implementation. The [Table 4.1](#) below describes the colour-coded security level spectrum.

**Table 4.1: Security Level Spectrum**

Security Level	Colour Code	Functions	Properties
No Security	0-Red	None	Authenticity – ✗ Integrity – ✗ Non-repudiation – ✗ Confidentiality – ✗
Low Security	1-Amber	HASH comparison (checksum)	Authenticity – ✗ Integrity – ✓ Non-repudiation – ✗ Confidentiality – ✗
Medium Security	2-Blue	Cryptographic checksum (encryption + hash)	Authenticity – ✗ Integrity – ✓ Non-repudiation – ✗ Confidentiality – ✓
Medium-high Security	3-Parrot	Digital Signing, Message not encrypted or partially encrypted	Authenticity – ✓ Integrity – ✓ Non-repudiation – ✓ Confidentiality – ✗ (None or Partial)
High Security	4-Green	Digital Signing, Message fully encrypted	Authenticity – ✓ Integrity – ✓ Non-repudiation – ✓ Confidentiality – ✓

[Table 4.2](#) describes the QR Security Element of the QR Dataset. Although only 1 byte in length, this field actually indicates a lot about the security strength of the QR Ticketing System. It is like a handshake token between the Ticket Generator and the Validator. The Ticket Generator employs a numbered security scheme (or algorithm) on the QR Data and writes that number on this field. The Validator then uses that number to determine how to verify the QR ticket. Every security scheme also has a colour-level indicator according to the [Table 4.1](#) above. It is only provided here for convenience of announcing the security scheme. Furthermore, it may also be used visually when encoding the QR Ticket, if some implementer wishes to issue coloured QR Tickets.

Table 4.2: QR Security Element

QR Security				
Tag	Length	Byte	Security Scheme with Level	Description
82	1	1	0(0x00) – No security	QR Data (payload) sent as plaintext.
			1(0x01) – Checksum message	Checksum generated with SHA-256 Hash function. Payload data is sent as plaintext and verification is only comparison of hashes.
			2(0x02) – Cryptographic checksum	Achieved using symmetric cryptography. Message authentication codes using AES256 will be calculated on payload. The checksum of message is also sent along. Ensures integrity and confidentiality.
			3(0x03) – RSA-based security algorithm	Achieved using asymmetric cryptography. Digitally signed message but does not guarantee confidentiality.
			4(0x04) – Hybrid implementation: – SecureQR-Alpha	Same as #3 above except some parts of the message is encrypted with operator-specific encryption algorithm. Some confidentiality of data is lost.
			5(0x05) – Hybrid implementation: – SecureQR-Beta	Same as #4 above, but an additional security measure is employed to encrypt the remainder of the message. This is a combined 2-fold encryption scheme (Operator-specific and TG) and provides the highest security level.
			6(0x06) – Hybrid implementation: – SecureQR-Gamma	Same as #3 above. Besides digital signature, the entire message is encrypted by the Ticket Generator using its own security algorithm. Provides the highest security level.
			7-200 (0x07-0xC8)	User-defined
			201-255 (0xC9-0xFF) – RFU	Reserved for Future Use.

*The implementation of **Security Scheme 4 – SecureQR-Alpha** is given in this document. However, any scheme can be implemented by PTO/TG as per their requirement. They may use their own scheme and define it in 7-200 as shown in the table above.*

## 5. SecureQR – Securitization of QR Codes in QR Ticketing System

As shown in [Table 4.2](#), SecureQR security scheme has three variants – Alpha, Beta and Gamma. Here we describe only the Alpha variant.

### 5.1. SecureQR-Alpha

The SecureQR-Alpha scheme accepts a Security Level-0 QR Plaintext Dataset as input and elevates it to a Security Level-3 Ciphertext object. That Ciphertext object is then used by the APP or TOM to render a QR image. For verification the exact opposite happens – the original QR Plaintext (Level 0) is retrieved from the QR Ciphertext (Level 3).

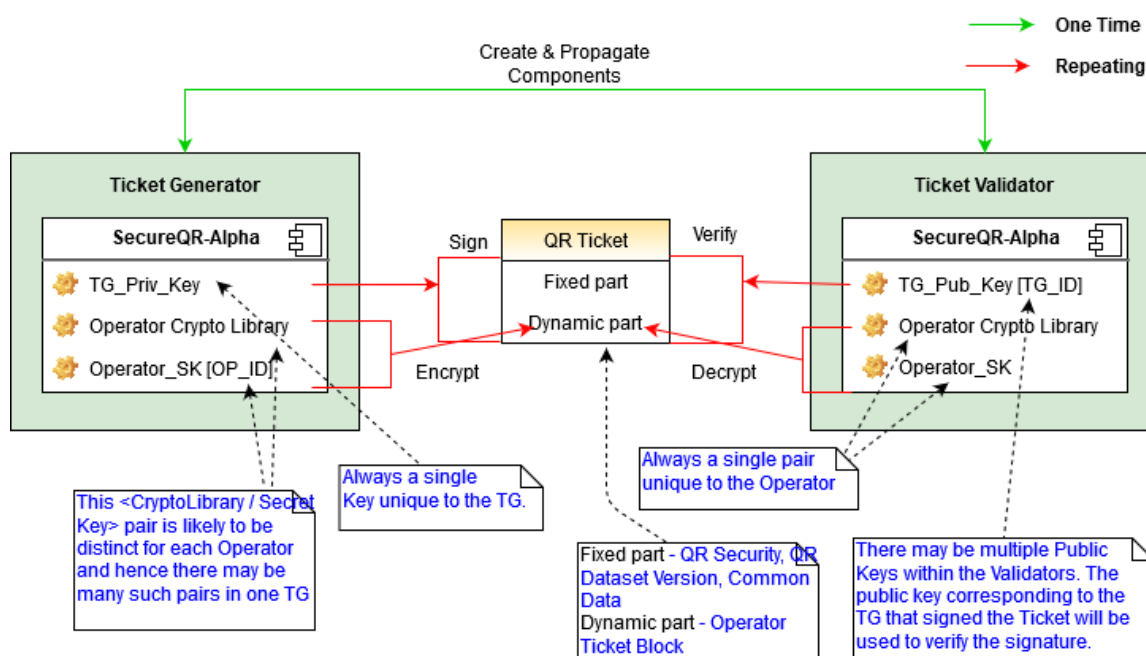


**Figure 1: SecureQR-Alpha Security Level I/O Diagram**

### 5.1.1.1. Phases of SecureQR-Alpha

SecureQR-Alpha consists of activities that need to be handled in two chronological phases – a one-time phase and a repeating phase. We describe them in detail below.

The diagram below shows the component-attribute model of the Ticket Generator and Ticket Validator (Operator). The accompanying notes describe the cardinality of the components in a multi-TG/multi-Operator architecture. Distinction is also made between One-time and Repeating phase activities.



**Figure 2: SecureQR-Alpha Component-Attribute Model**

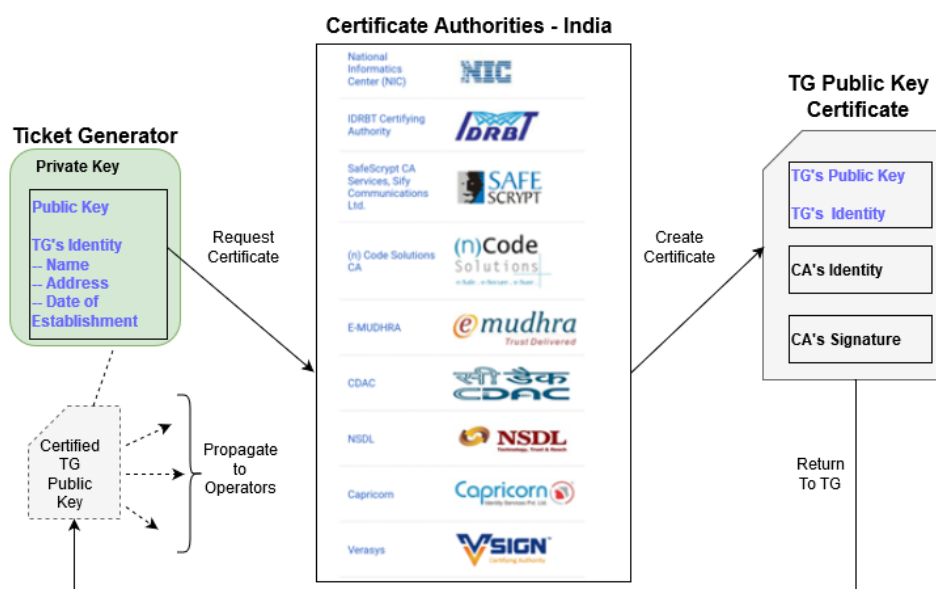
#### 5.1.1.1.1. One-Time Phase Activities

- Creation & propagation of RSA private and public key pair. The Private Key is used for Digital Signing and the Public Key for Signature Verification. This private-public key pair is specific to the TG. In the validators, the public key of the specific TG that signed the QR shall be used for verifying the signature.
- The TG shall bear the responsibility of
  - Creating the RSA private and public key pair<sup>[1]</sup>. We denote these in notational form as – <TG\_Priv\_Key[TG\_ID]>, <TG\_Pub\_Key[TG\_ID]>
  - Propagating the public key to all the operators that have subscribed to it. This is a critical activity required to establish trust relationship between Ticket Generators and Operators. All TGs must create this “chain of trust” by getting their public key certificate issued by Root CA. Getting public keys certified by CAs requires paying an annual fee. Nevertheless, it must be

<sup>1</sup>RSA key pair creation and propagation is common practice and shall not be described in detail here.

borne as establishing a chain of trust is a mandatory requirement in the QR Ticketing ecosystem. The “IT Act of 2001” recognizes the organizations shown in [Figure 3](#) below as the well-known Certificate Authorities in India that can issue certificates.

- When the TG receives its Public Key Certificate from the CA, it propagates it to the respective operators that are involved with it.



**Figure 3: Ticket Generator – CA – Operator Chain of Trust**

- iii. Upon receipt of the Public Key Certificate from the TG, the Operator shall bear the responsibility of
  - Copying the CA Public Key and the TG Public Key Certificate in all their validating terminals
- iv. The SecureQR-Alpha scheme employs operator-specific encryption with an algorithm provided by the operator. If this encryption algorithm uses secret keys, then the operator shall bear the responsibility of
  - Providing the encryption/decryption algorithm library to all their trusted TGs
  - Installing the encryption/decryption algorithm library in all their validating terminals
  - Creating and propagating the secret key to all their trusted TGs denoted here as Operator\_SK[OP\_ID]
  - Copying the secret key created above in all the validating terminals

#### 5.1.1.2. Repeating Phase Activities

Activities in the repeating phase constitute to form the main business use case of the QR Ticketing System i.e. creation and validation of QR Tickets. These are described in detail here.

**Ticket Generator – Creating the response to ticket request****A. Creating a QR-friendly payload**

In order to render a QR ticket, the payload must first be transformed into a *QR-friendly* form keeping in mind the size and scannability requirements of a QR. Transforming a plaintext QR payload into a Ciphertext block is broadly a 2-fold process – encryption and Base64 encoding. Both these processes increase the size of payload. Although encryption itself does not increase the size by much—probably some padding bytes at the end, Base64 encoding, on the other hand, increases it by at least 33.3%.

*So why do we need Base64 encoding?*

The short answer to this question is – To make the encrypted text transport-safe. We take an example using “openssl” to establish this fact.

```

djbaruah@dj-baruah-laptop:~/qrcode/gen$ cat qrraw
%0|19|2F|5EA6AD3E|18|3A98|1|1|5A%
djbaruah@dj-baruah-laptop:~/qrcode/gen$ ls -l qrraw
-rw-r--r-- 1 djbaruah djbaruah 34 May 20 16:53 qrraw
djbaruah@dj-baruah-laptop:~/qrcode/gen$ openssl aes-256-cbc -salt -e -in qrraw -out
qrraw.enc -k 1234
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
djbaruah@dj-baruah-laptop:~/qrcode/gen$ cat qrraw.enc
Salted [gibberish]
Ya[...][gibberish]
djbaruah@dj-baruah-laptop:~/qrcode/gen$ ls -l qrraw.enc
-rw-r--r-- 1 djbaruah djbaruah 64 May 23 11:12 qrraw.enc
djbaruah@dj-baruah-laptop:~/qrcode/gen$ cat qrraw.enc | base64 -w 0 > qrraw.enc.bas
e64
djbaruah@dj-baruah-laptop:~/qrcode/gen$ cat qrraw.enc.base64
U2FsZGVkX180tMTJpNiD8nNZQHrshJ4o8rVM0JaZ2oMzOmtApZYQGSzysUQ1tcyefSVtjQvwihz91K1Zu
2+w==
djbaruah@dj-baruah-laptop:~/qrcode/gen$ ls -l qrraw.enc.base64
-rw-r--r-- 1 djbaruah djbaruah 88 May 23 11:13 qrraw.enc.base64
djbaruah@dj-baruah-laptop:~/qrcode/gen$

```

1. Example text ( 34 bytes)

2. AES-256 encryption without Base64 encoding

3. Result of encryption. Gibberish!!

4. Size of encrypted text = 64 bytes. AES-256 uses 64-byte cipher blocks so 30 bytes are padded

5. Base64 encoding of encrypted text

6. QR-friendly text!

7. Increased size 88 bytes

**Figure 4: Producing QR-friendly text**

As seen in [Figure 4](#), there is a remarkable difference between the output of encryption (step 3) and that of converting to Base64 encoded text (step 6). Gibberish like seen after step 3 cannot be easily transported simply by embedding it as-is within a JSON object. But the same when encoded with Base64 can be easily done so.

**Calculation of sizes – Encrypted text and Base64 text:**

We have noted earlier that both encryption and Base64 encoding increases the size of a string. Encryption algorithms always cipher by blocks. So if the cipher block is of 64-bytes, as seen in [Figure 4](#), any text which is less than or equal to 64 bytes will produce a Ciphertext of size equal to 64 bytes. If the sample size is 75



bytes, the encrypted text will become 128 bytes and so on. The sample text string we used is of size 34 bytes Step 1, [Figure 4](#). After encryption, this size becomes 64 bytes Step 4, [Figure 4](#).

Base64-encoding normally increases the text size by 33.3%. As seen in the same example, after encoding the encrypted text with Base64, we get a text of size 88 bytes Step 7, [Figure 4](#). Base64 encoding produces 4 bytes for every 3 bytes of text (hence the  $4/3 = 33.3\%$  increase!). When the sample size is not a multiple of 3 (like in our example it is 64), the next higher multiple 66 is taken i.e. 64 becomes 66. Then 66 is divided by 3 and then multiplied by 4. This is confirmed in Step 7  $\rightarrow 66/3 * 4 = 88$ . The two '=' sign characters seen in the encoded text (step 6) are the two extra padded bytes ( $64 + 2 = 66$ ). When a number is not a multiple of 3, there can either be 1 or 2 '=' signs at the end. The system designer or programmer need not worry about these details as Base64 encoding and decoding algorithms automatically takes care of these intricacies, but this concept will come in handy when we calculate the various sizes of a QR payload later on in this document.

## B. Creating the QR Code Ciphertext

First, we must understand the QR Code Dataset structure itself. The QR Code Dataset is a complex hierarchical data structure with many branches and sub-branches as described in the Part I – QR Code Dataset.

Elements that constitute a QR:

- The nodes “QR Security”, “QR Dataset Version” and “QR Common Data” are present in every QR Code. We shall refer to these three nodes as QR\_SVC for the rest of the document.
- The node “QR Dynamic Data” is relevant only for APP-based QRs. For Paper-based QR's issued at the TOMs, this element may be omitted.
- The “QR Ticket Block  $\rightarrow$  Static Block” node is optional and may or may not be present in a QR Code.
- The ticket information – “QR Ticket Block  $\rightarrow$  Dynamic Block” is the heart of the QR and must be present in every QR code. Every QR must have at least one journey ticket.

As seen in Appendix I of Specifications Part I – QR Code Dataset, there are a number of combinations of tickets ranging from 1x1\_QR to 10x10\_QR.

To represent such complex combinations and to do so economically, by preserving the hierarchical structure, an entirely new format, Secure QR Dataset Representation format (SQDSR format) is being proposed in this document. Please refer to [Appendix I – SQDSR Format](#) for a detailed description of this format. Implementers of the QR Specifications are required to use the SQDSR format before encoding QR images. There is a tiny overhead involved in this format. [Table 5.1](#) below shows the additional characters (overhead) involved in encoding the QR Payload in SQDSR format.



**Table 5.1: SQDSR Size with Overhead for QR Dataset**

Name	Overhead Size	Additional characters
QR Security	2	{}
QR Dataset Version	2	{}
QR Common Data	12	{     }
QR Dynamic Data	6	{    }
QR Static Block (Personal Info only)	8	(<    >)
QR Static Block (Secondary Info – 4 members)	26	<                >
QR Static Block (Personal Info + Secondary Info – 4 members)	8 + 26 = 34	(<    ><                >)
Dynamic Block (Ticket Info only) – 1x1_QR	9 + 13 = 22	{{<    [%     ] >}}
Dynamic Block (Ticket Info only) – 1X10_QR	9 + 13*10 = 9 + 130 = 139	{{<    [%     ] >...up to 10 times...[%     ] >}}
Dynamic Block (Ticket Info only) – 10X10_QR	(4 + 5*10) + (13*10) = 4 + 50 + 130 = 184	{{<    [%     ] >...up to 10 times...<    [%     ] >}}

Fields of QR Dataset either have numeric or alphanumeric data type.

#### Storage size versus Representational size of data fields:

The data size of the elements considered here for calculation are all storage sizes meaning they can be stored in the DB or in program with that exact size limit provided the data types are correctly chosen. For instance, 'Operator ID' is 2 bytes and thus can be stored in the DB as a '**short integer**' that always occupies 2 bytes. Similarly, Tkt\_SI\_No, which is of 8 bytes, can be stored as a '**long integer**' data type that has a size of 8 bytes.

The representational or transmission sizes of these numeric fields are however different and are actually bigger than the storage sizes. For example, a **char** data type (=1 byte) may store a value equal to 135. In order to transmit that value, it must be converted to a string for which we need 3 characters (or bytes) – '1', '3' and '5'. If we instead convert the number to hexadecimal, we can do so as 2 Hex characters or 2 bytes.

In the SecureQR-Alpha scheme, all numeric data are converted to Hex characters before they are sent across. This means that the representation sizes of numeric fields will become double their storage sizes. Computation of sizes of numeric elements in this document are based on that same premise, but with some exceptions. As an example, any 1-byte Numeric field can have a decimal value from 0 to 255 or 0x00 to 0xFF in Hex. So that field may hold a value as small as *decimal 10 = Hex A* which is still 1 character but

to keep calculations straightforward we use 2 hex chars consistently. For sending data, however, there is no need to waste QR payload space by sending leading 0's like '0A'. So effectively, the actual QR size may end being much smaller than the calculated sizes as more or less the entire QR Dataset is numeric.

Alphanumeric fields do not need to be converted to HEX as their storage and transmission sizes are same. Change only numeric data to Hex chars (1 byte numeric = 2 Hex chars).

We shall now denote the process of creating the QR Dataset Ciphertext in the following sequential steps

- Plaintext – numeric and alphanumeric
- Plaintext after conversion of numeric data to HEX characters
- Plaintext encoded in SQDSR format.
- Plaintext after encryption (only Operator-specific Ticket Block)
- Digitally Sign the entire payload and finally
- Base64 Encoding producing the Ciphertext

This section describes the above steps in terms of changes to data size.

**Table 5.2: QR Code Security Plaintext Size**

Branch Name	Plaintext (A)	A after HEX conversion (B)	B with SQDSR separators (C1)
QR Security	1	2	$2 + 2 = 4$

**Table 5.3: QR Code Dataset Version Plaintext Size**

Branch Name	Plaintext (A)	A after HEX conversion (B)	B with SQDSR separators (C2)
QR Dataset Version	1	2	$2 + 2 = 4$

**Table 5.4: QR Code Common Data Block Plaintext Size**

Branch Name	Plaintext (A)	A after HEX conversion (B)	B with SQDSR separators (C3)
QR Common Data (Numeric = 30, Alphanum = 32)	$30 + 32 = 62$	$60 + 32 = 92$	$92 + 12 = 104$

In the SQDSR\_Alpha scheme, QR\_SVC is sent as plaintext.

**Table 5.5: QR SVC Size with Overhead**

Tag Name	Plaintext Size of C1, C2, C3
QR_SVC	$4 + 4 + 104 = 112$

**Table 5.6: QR Code Dynamic Data Plaintext Size**

Branch Name	Plaintext (A)	A after HEX conversion (B)	B with SQDSR separators (C)	QR Dynamic Data Size with Overhead
QR Dynamic Data	32	64	$64 + 6 = 70$	70

**Table 5.7: QR Code Static Block Plaintext Size**

Branch Name	Plaintext (A)	A after HEX conversion (B)	B with SQDSR separators (C)	Size with Overhead
QR Static Block (Personal Info only) Numeric = 2 Alphanum = 61	$(2 + 61) = 63$	$4 + 61 = 65$	$65 + 8 = 73$	73
QR Static Block (Secondary Info of 4 members)	$(2+61)*4 = (8 + 244) = 252$	$16 + 244 = 260$	$260 + 26 = 286$	286
QR Static Block (Primary Info + Secondary Info of 4 members)	$63 + 252 = 315 = 10 + 305$	$20 + 305 = 325$	$325 + 34 = 359$	359

When calculating the size for different combinations of the Tickets, we must also take into account the encryption size. Here we assume, the encryption is consistent with some algorithm similar to AES-256 but one that uses 16-byte cipher blocks. That implies that the encryption algorithm tries to split the input and cipher it in blocks of 16 bytes. The last block shall include padded bytes if it is less than 16 bytes. Another point that is to be noted about encryption is that the quantities – Operator ID, No of Tickets and Validator\_Info – are never encrypted or converted to Base64. Also encryption is only performed when the

‘encryption’ bit is set in the Validator\_Info information sent by the PTO as a Policy DB parameter. The dataflow diagrams shown in [Section 5.1.3](#) will make this concept more clearer.

As an example, let us say we have an Operator Ticket as shown below.

{{(<A|1|11|[%1|6|18|5EA7C9F8|1|1|1194|E0|B4|>}}). In this ticket, encryption is set.

Only the parts shown in pink shall be encrypted and then encoded with Base64, the rest of the text is left as it is. With this information intact, we now proceed to calculate the size.

**Table 5.8: QR Ticket Block Plain & Cipher Text Size**

Branch Name	Plaintext (A)	A after HEX conversion (B)	B with SQDSR separators (C)	QR Ticket Size without encryption	QR Ticket Size with encryption*
QR Ticket Block – Single Operator, Single Ticket – 1X1_QR	4 + 28 = 32	8 + 56 = 64	(8+9) + (56+13) = 17 + 69 = 86	86	With 16-byte cipher block size 69 bytes will have 16*5=80 bytes. Then Base64 conversion of this will increase it to 108 bytes.  17 + 108 = 125
QR Ticket Block – Single Operator, 10 Single Tickets – 1X10_QR	4 + 28*10 = 284	8 + 560 = 568	(8+9) + (560+130) = 17 + 690 = 707	707	With 16-byte cipher block size 690 bytes will have 16*44=704 bytes. Then Base64 conversion of this will increase it to 940 bytes.  17 + 940 = 957
QR Ticket Block – 10 Operators, 1 Single Ticket each – 10X10_QR	(4 + 28)*10 = 320	[8 + 56]*10 = 640	640 + 184 = 824 = (80+54) + (130+560) = 134 + 690	824	Each ticket has 69 bytes. With 16-byte cipher block size, they will each have 16*5=80 bytes. After Base64 conversion, each becomes 108 bytes.  134 + 108*10 = 1214

*\*Assuming all tickets are encrypted in case of multiple tickets – worst case scenario.*

### Digital Signing in SecureQR-Alpha Scheme:

Digital Signature is mandatory in the QR Ticketing System. The signing algorithm in SecureQR-Alpha uses SHA-256 hashing. This kind of signing produces a 128-byte Signature hash. Thereafter, encoding with Base64 produces a signature hash equal 172 ( $= 4 * 129/3$ ) bytes. The implementers are free to use a smaller Digital Signature. For example, if the Digital Signature Hash size is 64, then the Base64 Signature size of this shall be reduced to only 88 bytes ( $4 * 66/3$ ). For our illustration we are showing Signature of 172 bytes.

Using the data available from [Table 5.1](#) through [Table 5.8](#), we can now calculate the maximum size of the QR Payload. Dynamic Data gets included by the APP, but we must still account for it in order to calculate the total size of the QR image. Calculation of sizes is given for Tickets without any encryption and then tickets (all) with encryption.

**Table 5.9: Unencrypted QR Payload Minimum and Maximum Size**

Elements	QR Name	QR Block Size (Bytes)
<b>A.</b> [QR_SVC + Dynamic Data] + 1x1_QR	1X1_QR_MIN_WO_ENC	$[112 + 70] + 86 = 182$ $+ 86 = 268$
<b>B.</b> A + [Static Block Primary Info only]	1X1_QR_MID_WO_ENC	$268 + 73 = 341$
<b>C.</b> A + [Static Block Primary & Secondary Info]	1X1_QR_MAX_WO_ENC	$268 + 359 = 627$
<b>D.</b> [QR_SVC + Dynamic Data] + 1x10_QR	1X10_QR_MIN_WO_ENC	$[112 + 70] + 707 =$ $182 + 707 = 889$
<b>E.</b> D + [Static Block Primary Info only]	1X10_QR_MID_WO_ENC	$889 + 73 = 962$
<b>F.</b> D + [Static Block Primary & Secondary Info]	1X10_QR_MAX_WO_ENC	$889 + 359 = 1248$
<b>G.</b> [QR_SVC + Dynamic Data] + 10x10_QR	10X10_QR_MIN_WO_ENC	$[112 + 70] + 824 =$ $182 + 824 = 1006$
<b>H.</b> G + [Static Block Primary Info only]	10X10_QR_MID_WO_ENC	$1006 + 73 = 1079$
<b>I.</b> G + [Static Block Primary & Secondary Info]	10X10_QR_MAX_WO_ENC	$1006 + 359 = 1365$

With Digital Signature Encoding, we can now determine the QR Versions (QR encoding) of the unencrypted QR Tickets.

**Table 5.10: QR Versions for Unencrypted QR Tickets with Digital Signature**

QR Name	QR Block Size without Signature (Bytes)	Signature size (bytes)	QR Payload Size (Bytes) with Signature	QR Version with ECC = M
1X1_QR_MIN_WO_ENC	268	172	440	16
1X1_QR_MID_WO_ENC	341	172	513	18
1X1_QR_MAX_WO_ENC	627	172	799	23
1X10_QR_MIN_WO_ENC	889	172	1061	27
1X10_QR_MID_WO_ENC	962	172	1134	28
1X10_QR_MAX_WO_ENC	1248	172	1420	31
10X10_QR_MIN_WO_ENC	1006	172	1178	28
10X10_QR_MID_WO_ENC	1079	172	1251	29
10X10_QR_MAX_WO_ENC	1365	172	1537	32

Now we shall calculate the QR Payload sizes for tickets when all the Tickets are encrypted.

**Table 5.11: Encrypted QR Payload Minimum and Maximum Size**

Elements	QR Name	QR Block Size (Bytes)
<b>A.</b> [QR_SVC + Dynamic Data] + 1x1_QR	1X1_QR_MIN_WI_ENC	$[112 + 70] + 125 = 182 + 125 = 307$
<b>B.</b> A + [Static Block Primary Info only]	1X1_QR_MID_WI_ENC	$307 + 73 = 380$
<b>C.</b> A + [Static Block Primary & Secondary Info]	1X1_QR_MAX_WI_ENC	$307 + 359 = 666$
<b>D.</b> [QR_SVC + Dynamic Data] + 1x10_QR	1X10_QR_MIN_WI_ENC	$[112 + 70] + 957 = 182 + 957 = 1139$
<b>E.</b> D + [Static Block Primary Info only]	1X10_QR_MID_WI_ENC	$1139 + 73 = 1212$

<b>F.</b> D + [Static Block Primary & Secondary Info]	1X10_QR_MAX_WI_ENC	1139 + 359 = 1498
<b>G.</b> [QR_SVC + Dynamic Data] + 10x10_QR	10X10_QR_MIN_WI_ENC	[112 + 70] + 1214 = 182 + 1214 = 1396
<b>H.</b> G + [Static Block Primary Info only]	10X10_QR_MID_WI_ENC	1396 + 73 = 1469
<b>I.</b> G + [Static Block Primary & Secondary Info]	10X10_QR_MAX_WI_ENC	1396 + 359 = 1755

With Digital Signature Encoding, we can now determine the QR Versions (QR encoding) of the encrypted QR Tickets.

**Table 5.12: QR Versions for Encrypted QR Tickets**

QR Name	QR Block Size without Signature (Bytes)	Signature size (bytes)	QR Payload Size with Signature (Bytes)	QR Version with ECC = M
1X1_QR_MIN_WI_ENC	307	172	479	17
1X1_QR_MID_WI_ENC	380	172	552	18
1X1_QR_MAX_WI_ENC	666	172	838	23
1X10_QR_MIN_WI_ENC	1139	172	1311	30
1X10_QR_MID_WI_ENC	1212	172	1384	31
1X10_QR_MAX_WI_ENC	1498	172	1670	34
10X10_QR_MIN_WI_ENC	1396	172	1568	33
10X10_QR_MID_WI_ENC	1469	172	1641	34
10X10_QR_MAX_WI_ENC	1755	172	1927	37

As can be seen from [Table 5.12](#) above, 1927 bytes is the absolute maximum size of a QR Ticket which is a 10x10 group ticket. This is by far the highest limit that a QR ticket in this QR Ticketing System can stretch. Going any higher will introduce two problems

- Make the QR payload too big to render
- Make the QR unscannable or require extremely high-resolution scanners even when the QR is rendered.

Finally, we can conclude:

**Absolute Minimum QR Payload Size (with Digital Signature no ticket encryption), 1x1\_QR\_MIN\_WO\_ENC = 440 bytes.**

**Absolute Maximum QR Payload Size (with Digital Signature and ticket encryption), 10x10\_QR\_MAX\_WI\_ENC = 1927 bytes.**

The payload message – Ciphertext – size consists of the Digital Signature, the actual QR Data, the text form of the QR Ticket Serial Number and finally the Hash of the entire content. The response that the TG sends as Ciphertext message, does not include the Dynamic Data which is equal to 70 bytes ([Table 5.6](#)). It is actually needed to be appended by the APP. Hence, we subtract 70 from the QR Data when calculating Ciphertext. The text representational form of the **Tkt\_Serial\_No** is sent separately along with the QR Data just in case the APP that needs to render the QR, choses to display it on screen or paper. The QR Code Dataset Tkt\_SI\_No is already present in the QR\_SVC element inside the Common Data element, but the APP does not have any means of retrieving it. Recall from Section 4 of Part I – QR Code Dataset that the display size of Tkt\_SI\_No is 25 bytes. It is not necessary to encode it into Base64 form. The table below shows the size range (1x1\_QR\_WO\_ENC to 10x10\_QR\_WI\_ENC) of the Ciphertext message to be sent back to the APP or TOM. It must be emphasized here that QR Ciphertext is the response from the TG after a QR Ticket is purchased, and not the content with which the QR image shall be encoded. That content is the QR Payload only shown in the table below.

**Table 5.13: QR Code Ciphertext Message**

QR Ciphertext				
Payload Length (chars)	Element		Size (chars)	Description
459 to 1946	QR Payload	QR Signature	172	Content signed by Ticket generator
		QR Data	198 – 1685	Full QR information – security, dataset version, common data and ticket block
	SHA256 or higher hash of QR Payload		64	This is for data integrity check. The SHA256 (or higher) hash of “Requester ID#TG ID# QR_Signature#QR_SVC#QR_Tkt_Block #QR_Tkt_SI_No” is sent along. Here we are assuming a SHA256 Hash.
	QR Serial No		25	QR Ticket Serial No in representational form or display form.



**QR Image Rendering Recommendation:** - A QR that has a size of 1927 bytes as seen above, is likely to be very dense. Hence, it is recommended to use the default lowest settings to encode large QRs (above the size of 1500 bytes) i.e. use ECC = L and Module Size = 4 dots.

*Issuing Paper QR tickets having such large QR size may not be feasible, because the image itself may not be compressible to fit the paper ticket size. Therefore, it makes more sense to give preference to digital QRs. It is also recommended that since TOM belongs to the PTO and is installed in the PTO's premises, it must only be used to issue day to day running daily journeys.*

### 5.1.2. SecureQR-Alpha Algorithm Design

This section describes the flowcharts for the processes of Ticket Generation and Validation mainly from the security point of view.

#### Ticket Generation:

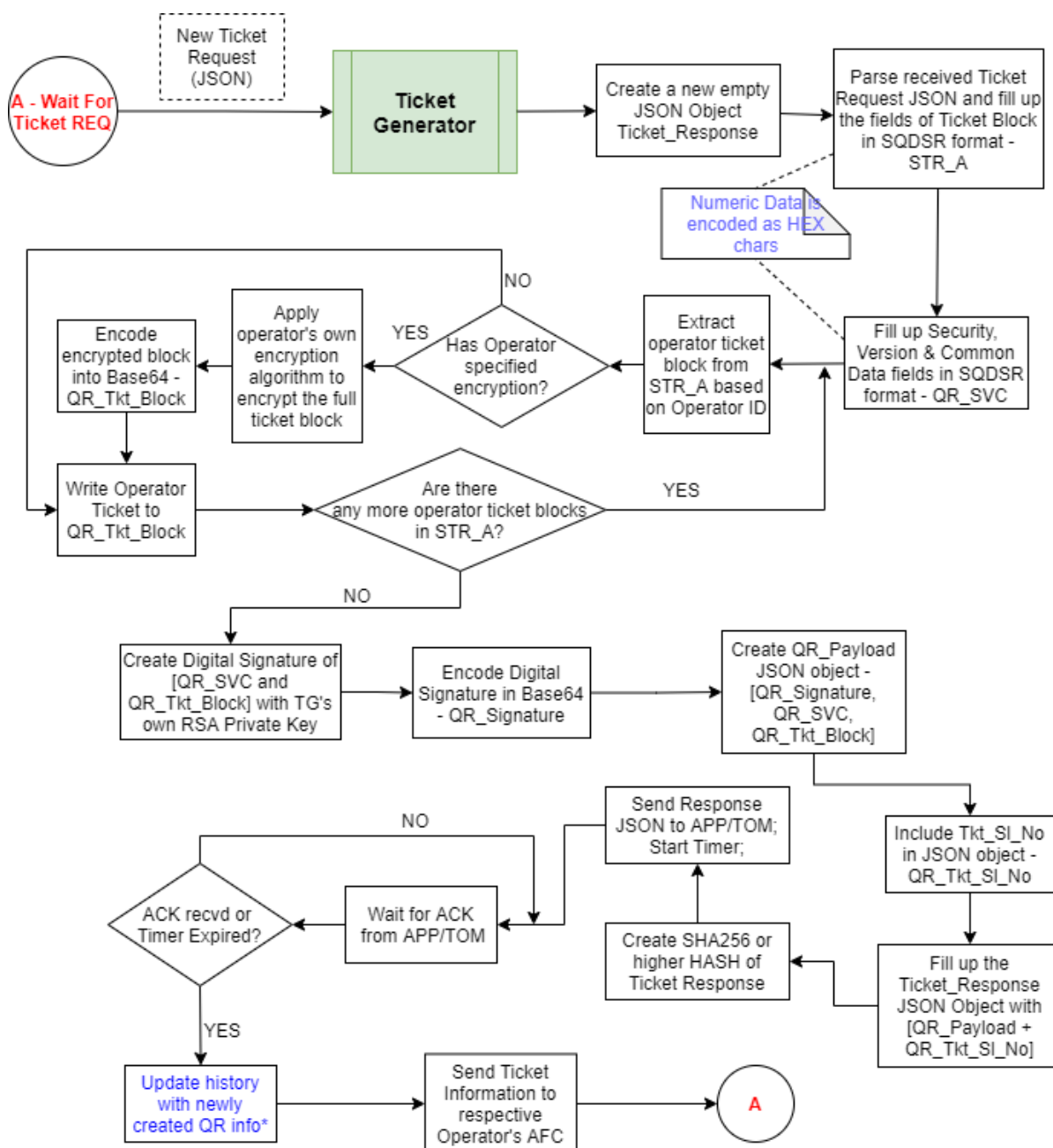
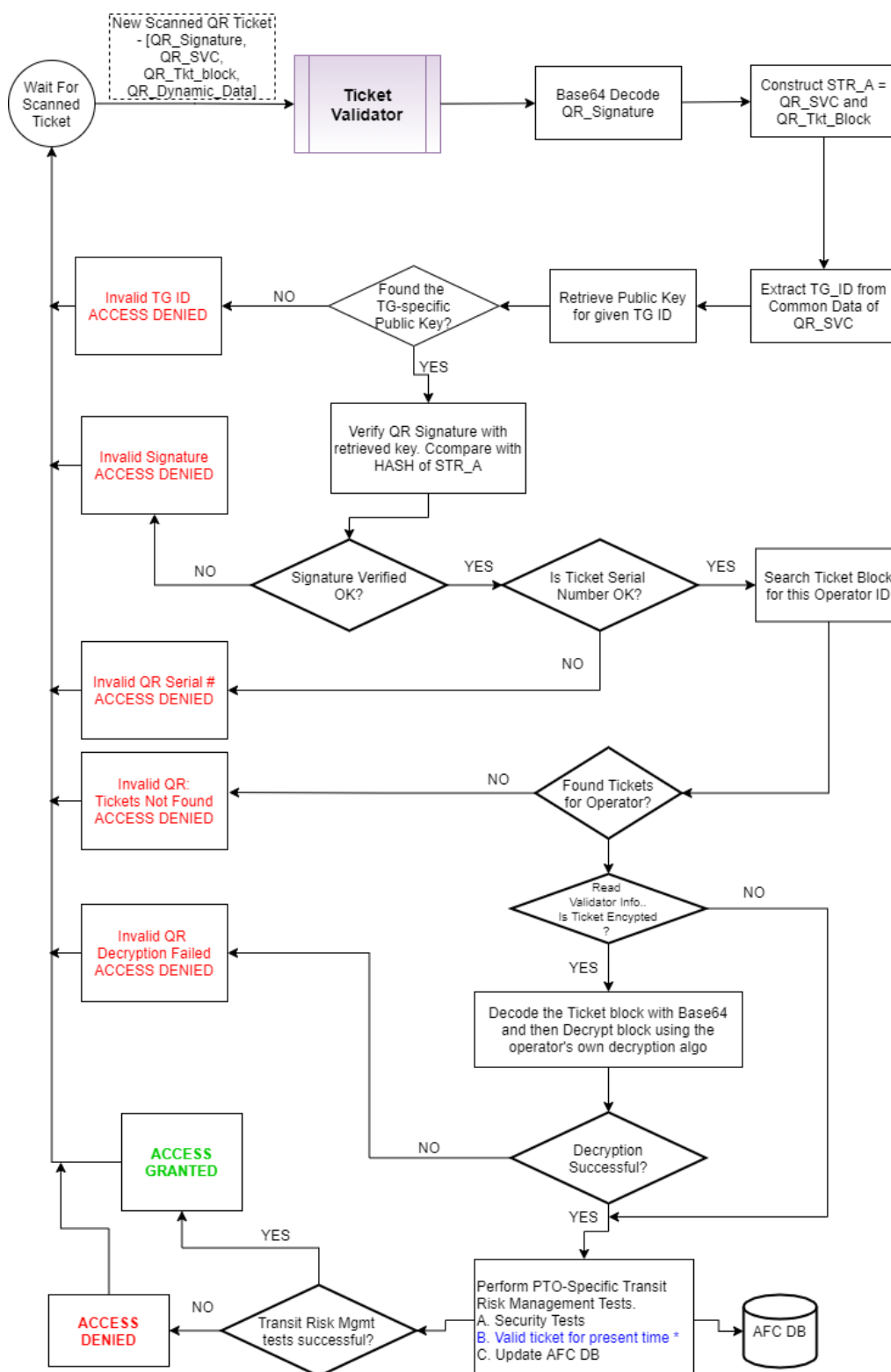


Figure 5: QR Ticket Generation Flowchart

\* The TG maintains record (history) of all the QR tickets it generates. This process is described in detail in Section 4.3.1 of Part III – Interface Specifications.

### Ticket Validation:



**Figure 6: QR Ticket Validation Flowchart**

\* There may be multiple journey information present in a single QR Ticket. However, the Validating Terminal validates only journey – the one that matches the Station code and the falls in the present time window (validity period).

As seen from the Flowchart above, validation is performed completely offline as the Validator does not need to access any other system to determine the validity of a QR Ticket. The step of syncing the QR information with the AFC DB is performed after all the validation steps are completed.

### 5.1.3. SecureQR-Alpha Dataflow Example

Assumptions and hypotheses:

- a. Operator ID = 10 and Operator ID = 135 both having one ticket each.
- b. (Ticket Block → Static Block) is empty i.e. Personal and Secondary Info not present.
- c. Version of QR Dataset = 1.0 => 00000100b = 4
- d. Language = 0 (English), Transaction Type = 65 (QR Purchase)
- e. QR Security => 4 => SecureQR-Alpha
- f. Ticket Generator ID = 23
- g. APP\_ID (or Requester ID) = 15
- h. Ticket Generation DateTime = 21/04/2020@ 18:47:54. This is date of ticket purchase.
- i. Date of journey or Activation Date for Operator ID 135 = 27/04/2020@15:30:30 and for Operator ID 10 = 28/04/2020@11:45:20.
- j. Ticket Validity period = 480 minutes (8 hours) for both operators
- k. Duration = 180 minutes (3 hours) for both operators
- l. Date of validation\* = 27/04/2020
- m. Validating operator → Operator ID = 135
- n. Sequence number at TG = 82506
- o. Mobile Requested Ticket => QR Type Bit = 1 (or M)
- p. Fares must be converted from Rupees to Paisa
- q. Validator Info for both Operator 10 and Operator 135 = 17 (Camera scanning, encryption)
- r. Operator-specific ticket data is not specified

\*Note: - In this example we are assuming that the customer is going to avail the services he has purchased for Operator ID=135 on 27/04/2020. The customer must get his ticket validated within the Ticket Validity period = 480 minutes i.e. between 15:30:30 hours and 23:30:30 hours on 27/04/2020. There is another ticket for 28/04/2020 (for Operator ID = 10) in the QR, which would remain unaffected by this transaction.

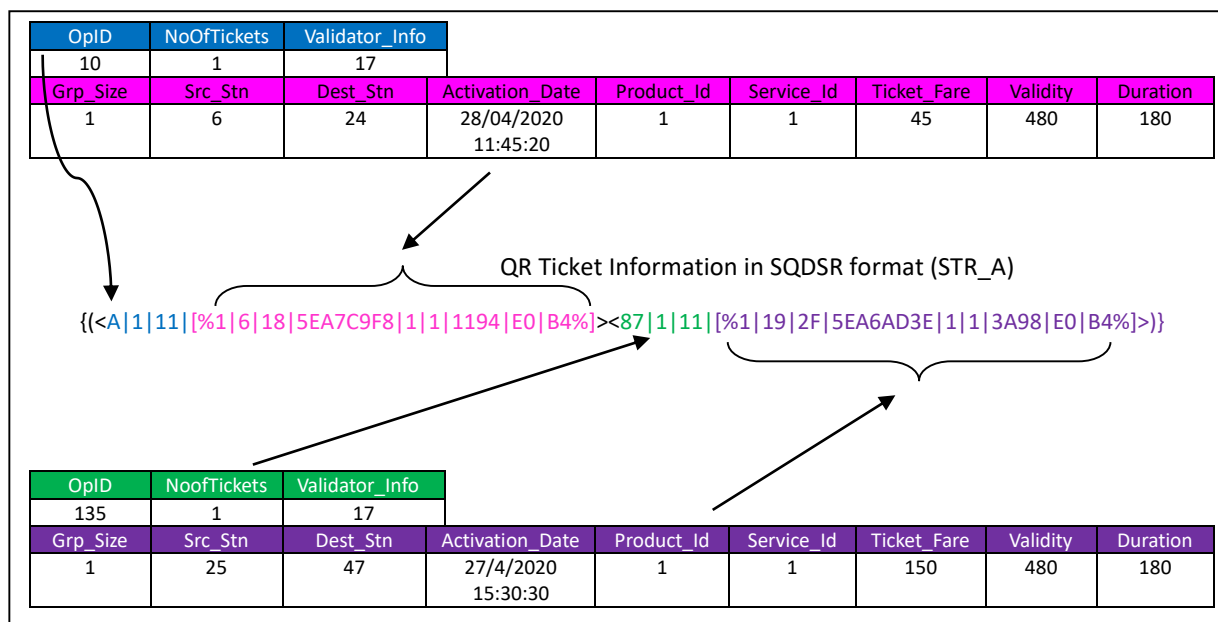
**Ticket Generation:**

**Step 1:** Ticket Request received – 2x2\_QR Ticket. For full details on how the ticket request is sent by the APP, please refer to Section 4.3 of QR Interface Specifications – Part III.

```
{
  "Requester_ID": "15",
  "Language": "0",
  "TXN_Type": "65",
  "Txn_Ref_No": "WXYZRC1204202056789000",
  "Txn_Date": "21/04/2020@18-47-43",
  "PSP_Specific_Data": "78;Merchant15;MO-1;195;IMPS;5467908;Merchant15_20200412_00312",
  "Total_Fare": "195",
  "Customer_Mobile": "9999922130",
  "TicketBlock": {
    "Dynamic_Block": {
      "OperatorID1": {
        "OpID": "10",
        "NoOfTickets": "1",
        "Validator_Info": "17",
        "TicketInfo": {
          "TicketInfo1": {
            "Grp_Size": "1",
            "Src_Stn": "06",
            "Dest_Stn": "24",
            "Activation_Date": "28/04/2020@11-45-20",
            "Product_Id": "01",
            "Service_Id": "01",
            "Tkt_Fare": "45",
            "Validity": "480",
            "Duration": "180"
          }
        }
      },
      "OperatorID2": {
        "OpID": "135",
        "NoOfTickets": "1",
        "Validator_Info": "17",
        "TicketInfo": {
          "TicketInfo1": {
            "Grp_Size": "1",
            "Src_Stn": "25",
            "Dest_Stn": "47",
            "Activation_Date": "27/04/2020@15-30-30",
            "Product_Id": "01",
            "Service_Id": "01",
            "Tkt_Fare": "150",
            "Validity": "480",
            "Duration": "180"
          }
        }
      }
    }
  }
}
```

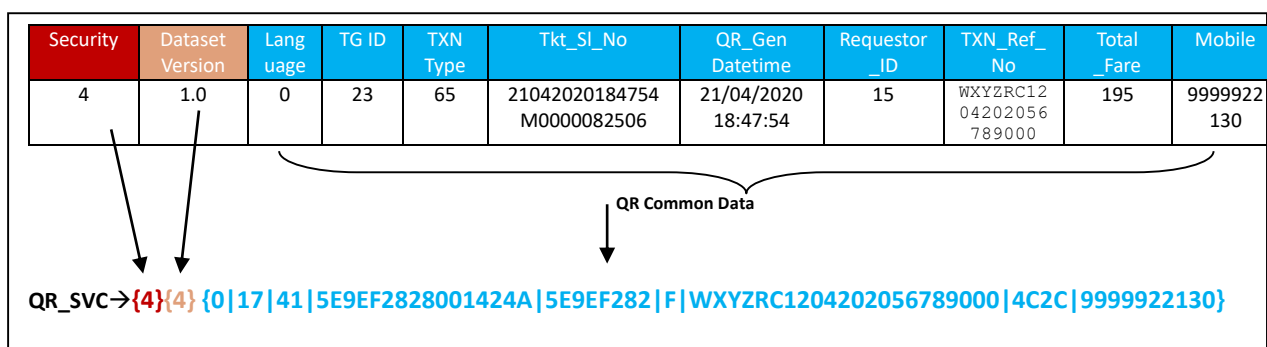
**Figure 7: 2x2\_QR Ticket Request JSON File**

**Step 2:** Parse JSON object and construct QR Payload object in SQDSR format. The figure below shows the Operator-specific Ticket Block in SQDSR format with all numeric values converted to HEX chars.



**Step 3:** Similarly create the QR Security and Common Data blocks.

Notice that the Tkt\_SI\_No becomes 5E9EF2828001424A. This is because Tkt\_SI\_No is 8 bytes (refer to Part I – QR Code Dataset for details). Ticket Generation date time (= 4 bytes => 21042020184754 = 0x5E9EF282), followed by a 1-bit value =1 or M for mobile QR, and then the 31-bit sequence number 82506 in decimal. The least significant 4 bytes altogether = 0x8001424A hex. Therefore, Tkt\_SI\_No becomes the 8-byte hex value 5E9EF2828001424A. When the Serial No is displayed on the screen, it should appear in the form **21042020184754M0000082506**.



**Step 4a:** We have assumed that the Operators use encryption, so now it is time to encrypt the tickets. Both tickets are encrypted separately applying each operator's own algorithm. Here for illustration we use AES-256-CBC and AES-256-ECB with different salts for Operators 10 and 135 respectively. As already seen we must encrypt only the Tickets and not the fields OperatorID, NoOfTickets and Validator\_Info. If these fields are encrypted, the Validator will never be able to find its tickets! The full ticket block of the operator is encrypted in one go and not one ticket at a time. Notice that encryption produces gibberish output. It is advisable not to embed this kind of gibberish text as-is into JSON objects because it might break the JSON

and the SQDSR parsers at the receiver. Also, transmission of such gibberish are often blocked out by many network routers and switches. Stating in other words, only the text between '[ ]', including the square brackets are encrypted.

```
{ (<A|1|11|Salted__y;à9î"s.6}1zãÿ/ú Mē!&-é (dð~É-
ÿMv|í¹á><87|1|11|Salted_•È‡õ•u-
³.,C«¾³°ÛCVWY²†¹•s"6Ù) ¯ÿÿiÿðõ.Eÿê³@1.±â9`P>) }
```

Operator 10(0xA): AES-256-CBC => 16-byte cipher block encryption

Operator 135(0x87): AES-256-ECB => 16-byte cipher block encryption

**Step 4b:** Then we convert the encrypted text from above to Base64 but leave the rest of the text as it is. The encrypted text is the fourth element of the SQDSR form → <OperatorID|NoOfTickets|Validator\_Info|EncryptedTicketBlock>. We need to convert the fourth element into Base64, because the encrypted fourth element is very likely to contain some SQDSR separators like '|', '(', ')', etc. Therefore, if we encode the whole thing and then try to retrieve the original text, the SQDSR parser is likely to break.

```
"QR_Tkt_Block":
"{ (<A|1|11|U2FsdGVkX181mELgdkO86kbbsqMqFTHRDRdmAwgxf1JV56H7BZxNqUVi27P8XRRmQTbud
X7rPb9P3kmlx1l1BQ==><87|1|11|U2FsdGVkX181riFmls14S6vqWU2japrheXL8Icq+SCmQsHifSio1
azq7zMS6ck+GQ4XdrJ65AgUHstxHLgKuuMA==>) }"
```

**Step 5a:** Using the strings QR\_SVC (from step 3) and QR\_Tkt\_Block (from step 4b), a SHA-256 Digital Signature is generated. The signature also produces complete gibberish!

```
{4}{4}{0|17|41|5E9EF2828001424A|5E9EF282|F|WXYZRC1204202056789000|4C2C|999992213
0}{ (<A|1|11|U2FsdGVkX181mELgdkO86kbbsqMqFTHRDRdmAwgxf1JV56H7BZxNqUVi27P8XRRmQTbu
dX7rPb9P3kmlx1l1BQ==><87|1|11|U2FsdGVkX181riFmls14S6vqWU2japrheXL8Icq+SCmQsHifSio
lazq7zMS6ck+GQ4XdrJ65AgUHstxHLgKuuMA==>) }
```

Digital Signing with TG's RSA Private Key<TG\_Priv\_Key>

QR\_SVC & QR\_Tkt\_Block

```
!±",gy$PâO<i3Ú%o|:/Ü#1à%.ÄÄ³q~=1>DÈ{-ŽRhþ$íeið[ÛE+C°HKQ|×ÄÚ€RìÆ´ó-
▣~+BÄXú346G~ÿ/²¹±òe...àlbbIBiÖt%u2t ¼%M⁴f—Ÿ...H
```

**Step 5b:** Next we convert the signature into Base64 as shown below.

```
"QR_Signature":
"TkhsXgFafTiM3dFIJ+YMMCq7MjJuVDiFGjw47ZkvfvjEkmw6wvVRNz1Bn0BYSpKoi8KrARwC
Lgk80vDdi8z2UOoMFpdeDusjSqt07kM73O6z2vY9iD7avHd3izA6vj8E1lkNXK0S1+baSGC94
Yw5JR9AuaPf2XTCA1/28vj7w0k="
```

**Step 5c:** We can convert the Security, Version and Common Data or QR\_SVC from Step 3 into Base64. But we will simply send it as plaintext to reduce the size of encoded QR.

```
"QR_SVC":  
"{4{4}{0|17|41|5E9EF2828001424A|5E9EF282|F|WXYZRC1204202056789000|C3|9999922130}"
```

**Step 5d:** Next we send the Tkt\_SI\_No from Step 3 (=21042020184754M0000082506) to the APP as it is. Note that this quantity is the representational form of the QR Code Dataset → Tkt\_SI\_No.

```
"QR_Tkt_SI_No": "21042020184754M0000082506"
```

**Step 5e:** It is essential to employ a mechanism to ensure the integrity of the QR Payload. For this purpose, the TG shall generate the SHA256 hash of the QR Payload and the quantities – TG ID and Requester ID (also called APP ID). Including two quantities TG ID and Requester ID is essential to randomize the SHA256 hash. QR Payloads can be sent from one and only one TG. The App Provider must know the ID of the TG that sent the response without the TG having to reveal it. The 'Requester ID' is its own ID. Please refer to Section 5.3 of Part III – QR Interface Specifications for details on APP Provider and TG Interface.

```
SHA256 (TG_ID#Requester ID#QR_Tkt_SI_No#QR_SVC#QR_Tkt_Block) →  
  
SHA256 (23#15#21042020184754M0000082506#{4}{4}{0|17|41|5E9EF2828001424A|5E9EF282|F  
|WXYZRC1204202056789000|C3|9999922130}#{(<A|1|11|U2FsdGVkX181mELgdk086kbbsqMgFTHR  
DRdmAwgxf1JV56H7BZxNqUVi27P8XRRmQTbudX7rPb9P3kmlx1i1BQ==><87|1|11|U2FsdGVkX18riFm  
1sl4S6vqWU2japrheXL8Icq+SCmQsHifSiolazq7zMS6ck+GQ4XdrJ65AgUHstxHLgKuuMA==>)}) =>  
  
QR_SHA256: 9836AAB7E41DF5FC654B26FCAA47D882478D359054A45D208B085C4B298D79CE
```

**Step 6:** With all the JSON key-value object pairs ready, the TG now sends the Ticket Response back to the APP as follows. This Ticket Response is the same 'Ciphertext message' shown in Table 5.13 where QR Data → QR\_SVC & QR\_Tkt\_Block.



```
"Ticket Response": {
  "QR_Payload": {
    "QR_Signature": "TkhsXgFafTiM3dFIJ+YMMCq7MjJuVDiFGjw47ZkvfvjEkmw6wvVRNzlBn0B
YSpKoi8KrARwCLgk80vDdi8z2UOoMFpdeDusjSqt07kM73O6z2vY9iD7avHd3izA6vj8E1lkNXK
0Sl+baSGC94Yw5JR9AuaPf2XTCA1/28vj7w0k=",
    "QR_SVC": "{4}{4}{0|17|41|5E9EF2828001424A|5E9EF282|F|WXYZRC1204202056789000
|C3|9999922130}",
    "QR_Tkt_Block": "<A|1|11|U2FsdGVkX18imELgdk086kbbsqMqFTHRDRdmAwgxf1JV56H7BZ
xNqUVi27P8XRrmQTbudX7rPb9P3kmlx1l1BQ==><87|1|11|U2FsdGVkX18riFmls14S6vqWU2j
aprheXL8Icq+SCmQsHifSiolazq7zMS6ck+GQ4XdrJ65AgUHstxHLgKuuMA==>)"
  },
  "QR_Tkt_SI_No": "21042020184754M00000082506"
  "QR_SHA256": "9836AAB7E41DF5FC654B26FCAA47D882478D359054A45D208B085C4B298D79CE",
}
```

### APP renders the QR:

Before we proceed to Validation, it is necessary we show an important intermediate step here – that is how the APP should render the QR after it receives the Ticket Response as shown above. The objective of the APP is to append the Dynamic Data element to the QR Payload.

- The first thing the APP does is it extracts the different elements in the response, namely QR\_Signature, QR\_SVC, QR\_Tkt\_Block, QR\_Tkt\_SI\_No and QR\_SHA256.
- Then it checks the integrity of the Payload.  
APP\_SHA256 →  
SHA256(TG\_ID#Requester\_ID#QR\_Signature#QR\_SVC#QR\_Tkt\_Block#QR\_Tkt\_SI\_No). Then compare QR\_SHA256 and APP\_SHA256
- Assuming the integrity is fine, the APP must then get the local time from its system. Let us assume this date and time is **21/04/2020 18:48:12**. In hexadecimal, this date value is **5E9EF294**.
- The APP must then form the Dynamic Data element but leave the Status and Location elements as 0's or empty. This quantity is also kept as plaintext. We shall keep it simple here. For advanced implementations of this feature, please refer to Appendix III of Part III – Interface Specifications for details.


```
QR_Dynamic_Data → {QR_Update_Time|QR_Status|Latitude|Longitude|
Operator-specific dynamic data} → {5E9EF294||||}
```

- Finally the QR string to encode is created as → QR\_Signature#QR\_SVC#QR\_Tkt\_Block#QR\_Dynamic\_Data. This gives:

```
"TkhsXgFafTiM3dFIJ+YMMCq7MjJuVDiFGjw47ZkvfvjEkmw6wvVRNzlBn0BYSpKoi8KrArwCLgk80vDdi8z2UoMFpdeDusjSqt07kM73O6z2vY9iD7avHd3izA6vj8E1lkNXK0S1+baSGC94Yw5JR9AuaPf2XTCA1/28vj7w0k={4}{4}{0|17|41|5E9EF2828001424A|5E9EF282|F|WXYZRC1204202056789000|4C2C|9999922130}#{(<A|1|11|U2FsdGVkX181mELgdkO86kbbsqMqFTHRDRdmAwgxf1JV56H7BZxNqUVi27P8XRRmQTbudX7rPb9P3kmlx1l1BQ==><87|1|11|U2FsdGVkX18riFm1sl4S6vqWU2japrheXL8Icq+SCmQsHifSiolazq7zMS6ck+GQ4XdrJ65AgUHstxHLgKuuMA==>)}#{5E9EF294|}|}"
```

The length of this full text is 470 bytes. From the Part I – QR Code Dataset – Character Capacities Table, we determine that, with binary mode encoding, the minimum version that can accommodate this string is Version 17 (= 504 bytes, ECC=M). We feed the text to our QR encoding algorithm with parameters ECC = M, mode = Binary, Dots per module = 4 and QR Version = Auto (up to the algorithm to decide). We also enable full verbose mode. The version information returned by the program was exactly 17! This confirmed that the encoding algorithm is correct and consistent with QR theory.

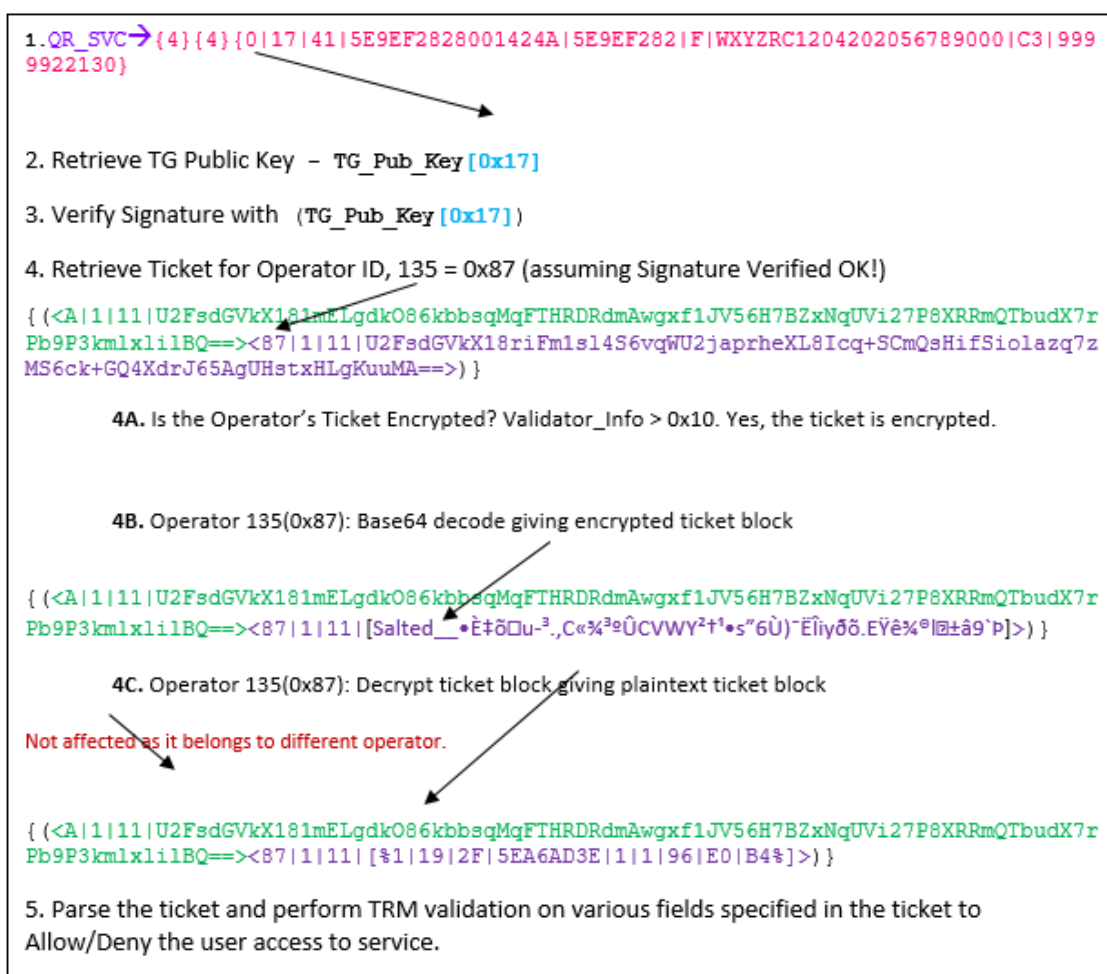
**Table 5.14: QR Data and Image Rendering**

QR Data (APP)	Image
#TkhsXgFafTiM3dFIJ+YMMCq7MjJuVDiFGjw47ZkvfvjEkmw6wvVRNzlBn0BYSpKoi8KrArwCLgk80vDdi8z2UoMFpdeDusjSqt07kM73O6z2vY9iD7avHd3izA6vj8E1lkNXK0S1+baSGC94Yw5JR9AuaPf2XTCA1/28vj7w0k={4}{4}{0 17 41 5E9EF2828001424A 5E9EF282 F WXYZRC1204202056789000 4C2C 9999922130}#{(<A 1 11 U2FsdGVkX181mELgdkO86kbbsqMqFTHRDRdmAwgxf1JV56H7BZxNqUVi27P8XRRmQTbudX7rPb9P3kmlx1l1BQ==><87 1 11 U2FsdGVkX18riFm1sl4S6vqWU2japrheXL8Icq+SCmQsHifSiolazq7zMS6ck+GQ4XdrJ65AgUHstxHLgKuuMA==>)}#{5E9EF294 } }#	

## Ticket Validation:

We will not show all the data flows in the Validation process because the process of Ticket Validation is like undoing the changes applied on the data by the Ticket Generator. It takes place in the exact reverse order and the functions applied are the exact inverses. For example, Step 6 becomes Step 1 where the function becomes extraction of TG\_ID; Step 5 becomes Step 2 i.e. Signature Verification with TG's public key and so on and so forth.

However, there's an important step that is worth mentioning here. The Validator's Step 3 (decryption), which corresponds to Generator's Step 4 (encryption) – is only a single-step activity. Where the TG encrypts all tickets per Operator one after another, the validator only searches through the "Operator" list in the QR\_Tkt\_Block, picks its own ticket block and goes ahead and performs decryption on those tickets. The other tickets that belong to another operator remain unaffected.



## 6. References

Sl. No	References
1.	QR Code Dataset Specifications – Part I, QR Interface Specifications – Part III
2.	ISO/IEC 18004-2015 – QR standard Definition , Error correction levels, etc – <a href="https://www.iso.org/obp/ui/#iso:std:iso-iec:18004:ed-3:v1:en">https://www.iso.org/obp/ui/#iso:std:iso-iec:18004:ed-3:v1:en</a>
3.	RFC- 6234 – Secure Hash Algorithms SHA-2 <a href="https://tools.ietf.org/html/rfc6234">https://tools.ietf.org/html/rfc6234</a>
4.	RFC3826 – Advanced Encryption Security (AES) – <a href="https://it.ojp.gov/NISS/iepd/443">https://it.ojp.gov/NISS/iepd/443</a> , <a href="https://en.wikipedia.org/wiki/Advanced_Encryption_Standard">https://en.wikipedia.org/wiki/Advanced_Encryption_Standard</a>
5.	RSA Cryptography – <a href="https://datatracker.ietf.org/doc/rfc8017/">https://datatracker.ietf.org/doc/rfc8017/</a>
6.	RSA Numbers – <a href="https://en.wikipedia.org/wiki/RSA_numbers">https://en.wikipedia.org/wiki/RSA_numbers</a>
7.	Euler’s Totient Function – <a href="https://en.wikipedia.org/wiki/Euler%27s_totient_function">https://en.wikipedia.org/wiki/Euler%27s_totient_function</a>
8.	Modular Multiplicative Inverse – <a href="https://en.wikipedia.org/wiki/Modular_multiplicative_inverse">https://en.wikipedia.org/wiki/Modular_multiplicative_inverse</a>
9.	Chinese Remainder Theorem – <a href="https://en.wikipedia.org/wiki/Chinese_remainder_theorem">https://en.wikipedia.org/wiki/Chinese_remainder_theorem</a>

## Appendix I – SQDSR Format

### SQDSR – SecureQR DataSet Representation format

In order for the validator to efficiently read and validate the QR Data, the hierarchical structure of the QR Code Dataset must be strictly preserved. JSON or XML are good formats to represent hierarchical data but the problem with these formats is that they are text-heavy. They cannot be used because the keys/tags themselves occupy plenty of space. The same goes for TLV format where just the “tags” and “lengths” will suffocate the QR data space.

For this reason, in the SecureQR Ticketing System, we introduce a new data representation model. This format shall preserve the hierarchy of the data and yet be far more economical w.r.t. space consumption than any one of JSON, XML or TLV formats. It is recommended to follow this structure and the order when encoding QR codes. The QR decoder must then present the SQDSR string as input to the parser in the Validating terminal.

It works as follows:

- i. Enclose a top-level tree object within “{}”. So QR Code dataset becomes: -  
`{QR_Security}{QR_Dataset_Version}{QR_Common}{QR_TicketBlock}{QR_Dynamic_Data}`
- ii. Subsequent sub-branches are enclosed within “()”, “<>”, “[ ]”, “% %”, “^ ^” and “~ ~” pairs – in that very order. The same delimiter pairs could be used but that would make parsing more complex.
- iii. Leaves are separated by pipes – “|”. If a ‘leaf’ is not a mandatory element, then it should be left blank. For example, {ABCD||DEFG||} etc.

Since QR\_TicketBlock consists of the StaticBlock and Dynamic Block, we can collapse the QR\_TicketBlock as follows:

- a. QR\_TicketBlock → {(StaticBlock)(DynamicBlock)}. Please recall that the entire StaticBlock can be empty but DynamicBlock must be present. So, QR\_TicketBlock can also become QR\_TicketBlock→{(DynamicBlock)}.

**A.** If the static block is present then, StaticBlock → (<PrimaryInfo><SecondaryInfo>). Again please recall that if StaticBlock is present, then only the PrimaryInfo is mandatory. So,

- i. SecondaryInfo → <[MemberInfo1][MemberInfo2][MemberInfo3][MemberInfo4]>
- ii. The PrimaryInfo collapses into PrimaryInfo→ Name|Age|Gender|ID\_Type|ID\_No
- iii. Each MemberInfo also collapses similarly, MemberInfo→ Name|Age|Gender|ID\_Type|ID\_No

**B.** Thus a static block with 2 group member info may look like, StaticBlock → (<Name|Age|Gender|ID\_Type|ID\_No><[Name|Age|Gender|ID\_Type|ID\_No][Name|Age|Gender|ID\_Type|ID\_No]>)

**C.** The dynamic block inside the QR\_TicketBlock is mandatory and must always be present, DynamicBlock → (<OperatorID1><OperatorID2><OperatorID3><...><OperatorID10>).

The parser must be written prudently. It should only send so many OperatorID tickets as required. Thus if there are 2 operators in the QR it can simply be (<OperatorID1><OperatorID2>).

- i. Each OperatorID collapses into a common part and then a repeating part

<Op\_ID|No\_of\_Tickets|Validator\_Info|[%TicketInfo1%%TicketInfo2%%TicketInfo3%% ... %%TicketInfo10%]>

Again the parser must be written prudently. It should only send so many TicketInfo as required. Thus if this OperatorID has only 2 tickets, then one can simply send <Op\_ID1|No\_of\_Tickets|Validator\_Info|[%TicketInfo1 %% TicketInfo2%]>.

- ii. Each TicketInfo collapses into several elements like shown (the sequence or order of elements is important).

[%Grp\_Size|Src\_Stn|Dst\_Stn|Activation\_Date|Product\_Id|Service\_Id|Tkt\_Fare|Validity|Duration|Op\_Specific\_Data%]

- D. Thus the Dynamic block with 2 Operators, each having one ticket may look like,

DynamicBlock→(<Op\_ID1|No\_of\_Tickets|Validator\_Info|[%Grp\_Size|Src\_Stn|Dst\_Stn|Activation\_Date|Product\_Id|Service\_Id|Tkt\_Fare|Validity|Duration|Op\_Specific\_Data%]><Op\_ID2|No\_of\_Tickets|Validator\_Info|[%Grp\_Size|Src\_Stn|Dst\_Stn|Activation\_Date|Product\_Id|Service\_Id|Tkt\_Fare|Validity|Duration|Op\_Specific\_Data%]>)

- E. At last we have the QR Dynamic Data element. This one shall look like

QR Dynamic Data → {QR\_Update\_Datetime|QR\_Status|Latitude|Longitude|Operator\_Specific\_Data}

Finally, putting it all together, if we have a QR Code Ticket that consists of 2 Operators each having one ticket each, then a Static block having personal info for primary member and 2 group members, and finally a dynamic data block, the payload can be visualized as 3 main blocks – the QR\_SVC, QR\_TicketBlock and then the QR\_Dynamic Data. When represented in SQDSR format, each block will look as shown below.

QR\_SVC→{Security\_Scheme}{Dataset\_Version}{Language|TG\_ID|Txn\_Type|Tkt\_SI\_No|QR\_Generation\_Dt|Requester\_ID|TXN\_reference\_No|Total\_Fare|Booking\_Latitude|Booking\_Longitude|Mobile}

QR\_TicketBlock→({<Name|Age|Gender|ID\_Type|ID\_No><Name|Age|Gender|ID\_Type|ID\_No|Name|Age|Gender|ID\_Type|ID\_No>}<Op\_ID1|No\_of\_Tickets|Validator\_Info|[%Grp\_Size|Src\_Stn|Dst\_Stn|Activation\_Date|Product\_Id|Service\_Id|Tkt\_Fare|Validity|Duration|Op\_Specific\_Data%]><Op\_ID2|No\_of\_Tickets|Validator\_Info|[%Grp\_Size|Src\_Stn|Dst\_Stn|Activation\_Date|Product\_Id|Service\_Id|Tkt\_Fare|Validity|Duration|Op\_Specific\_Data%]>})

QR\_Dynamic\_Data→{QR\_Update\_Datetime|QR\_Status|Latitude|Longitude|Operator\_Specific\_Data}

The SQDSR format can save up to 60-65% space compared to normal encoding techniques. Experimentally, JSON files as big as 5.8 KB have been encoded in a mere 1.8 KB using this format.

## Appendix II – QR Code Scan Times

### QR Optical Scanner Tests:

The requirement for Scanning and Validation times of QR Tickets is that it should never exceed more than 500 milliseconds.

The Dataset of the QR Ticketing System is complex but is not heavy. The time to scan any QR Code depends on the HID (Human Interface Device).

Validation of a QR Ticket in the QR Ticketing System is a multi-step process. Therefore, the total time to validate a ticket may have many different times as shown below.

- a. **T1** – Total QR transaction time: This is the time taken from the user input that starts QR scanning to the completion of validation of QR data
- b. **T2** – Total Software Time: This is the time taken by the application to process a video frame with valid QR image (valid frame)
- c. **T3** – Time to valid frame: This is the time taken from the start of scan to the arrival of the valid frame
- d. **T4** – Valid frame scan time: This is the time taken to process the valid frame and extract the information encoded in it
- e. **T5** – Time to parse data: This is the time taken to parse the encoded information into its constituent parts and verify the signature
- f. **T6** – Time to validate all the tickets in the parsed data to identify at least one valid ticket.

As seen in the various time variables, it involves both Software and Hardware times.

**T2 = (T4 + T5 + T6)** is the absolute Software time and **T3** is the absolute Hardware time.

Hence, Total Transaction Time, **T1 = T2 + T3**

Looking in another way,

- Parameters **T3** & **T4** are related to the scanner device. Let **T<sub>H</sub> = T3 + T4**
- Parameters **T5** & **T6** are related to business rules validation **T<sub>S</sub> = T5 + T6**

*It is desired that **T<sub>H</sub> + T<sub>S</sub>** should never exceed 500 milliseconds and individually either **T<sub>H</sub>** and **T<sub>S</sub>** should not exceed 250 milliseconds respectively.*

*The maximum size of a live QR Ticket created using the QR Ticketing System Specifications has no bearing on the scanning time but is solely dependent on the HID as shall be evident from the tests results published later in this section. Live QR Tickets were used in the tests performed with 2 HID.*

- **HID A: MITSUNAMI PART NUMBER M/USBOV5640V3 (5MP USB CAMERA MODULE WITH AUTO FOCUS)** – This is only a camera device and not a proper scanner. The QR Image is first focused upon and then decoded. The scanned string is then dumped to a USB buffer from where the QR Ticketing System Validator picks up and performs parsing and ticket validation. In terms of the time variables the device uses up the times **T3** and **T4**.

**KEY Features**

USB 2.0 & 1.1 compliance  
USB video class V1.1 compliance (UVC)  
USB audio class V1.1 compliance (UAC)  
USB high speed, full speed auto switching.

IMAGE SENSOR 5mp OV 5640

PCB SIDE MALE CONNECTOR JST PART # SHR-05V-S-B.

**IMAGE SPECIFICATION**

Illumination: 500+/-50 Lux;  
Color temperature: 5000+/-500 K;  
Characteristics: The length to width=4:3;  
MTF Specification(TV Lines):  
Center:>1100TV Line  
Corner:>800TV Line

**AUTO FOCUS LENS SPECIFICATION**

The focal length of 5M USB Camera is 10cm.

**Figure 8: Specifications Mitsunami Part Number M/USBOV5640V3**

- **HID B: ZEBRA MS4717 FIXED MOUNT IMAGER** – This is a proper scanning device with an optical sensor. A focused beam (orange light) is thrown as soon as the device senses an opaque body in front of the optical device. The scanned string is instantly sent to a character device file (Abstract Control Model) from where the QR Ticketing System Validator picks up and performs parsing and ticket validation. In terms of the time variables the device uses up the times **T3** and **T4**.



### MS4717 Technical Specifications

Table 5 MS4717 Technical Specifications

Item	Description
<b>Performance Characteristics</b>	
Sensor Resolution	1280 x 800 pixels
Field of View	42° horizontal, 28° vertical
Pitch/Skew/Roll Tolerance	± 60° / ± 60° / 360°
Aiming Element (LED)	610 nm ± 10 nm
Illumination Element	660 nm ± 5 nm (LED)
Minimum Print Contrast	20% absolute dark/light reflectance
<b>User Environment</b>	
<b>Power Requirements</b>	
Supply Voltage	5.00 V ± 0.5 V
Low Power / Suspend Current Draw	2.5 mA (max)
Idle Current	125 mA RMS (typical)
Operating Current (scan/decode session)	265 mA RMS (typical)
Peak Current	480 mA
Ambient Light Immunity	Total darkness to 10,000 ft. candles (107,369 lux)
<b>Humidity</b>	
Operating	95% RH, non-condensing at 50° C
Storage	85% RH, non-condensing at 70° C

Figure 9: Specifications Zebra MS4717 Fixed Mount Imager

### Benchmarking Test Results for various sized QR Tickets:

Recall that the convention  $QR\_M \times N$  implies there are  $N$  tickets for  $M$  operators. When  $M=N$ , it implies every operator has one ticket each. Also note that the QR Version is the QR Encoding Version and not the QR Code Dataset Version of [Error! Reference source not found.](#)

### Experiment setup:

These tests were performed at the R&D Lab of the Embedded Systems Group at CDAC, Noida.

### Input and Output:

- A total of 100 QR Tickets were used – 10 each of QR\_1x1 (1 operator, 1 ticket) through to QR\_1x10 (1 operator, 10 tickets). The same set of QRs were repeated over both the HIDs. These are live QR Tickets generated using CDAC's own TravelMOZO APP (Webclient and Mobile APP) and QR Ticket Generator (TG) server, hosted on the CDAC Cloud. The chosen PTO was B.E.S.T., Mumbai and the CAFC also belongs to the same.
- For some scans Paper QR Tickets were used while for the rest the QRs were presented on Mobile APP. Scannability of a QR Code is not affected by the medium from which a QR is presented as long as the quality of the code is good.
- Only times **T3** and **T4** are of interest and are published here. **T3 + T4**
- Times – **T5** and **T6** – that are the times to parse and validate tickets respectively, may vary from operator to operator's transit management business rules. Still, the best, average and worst case estimates for **T5 + T6** were performed on the machine where **HID A** was mounted. **T5 + T6**, in any case should never exceed 150-200 milliseconds.

### HID A:

- Scanning distance → 4 – 4.5 inches approx.
- System Specifications

**Static hostname:** am438x-epos-evm  
**Icon name:** computer  
**Machine ID:** d551cd1e29d1438c80c8408d34c7f999  
**Boot ID:** 19670322abe649ce9945171e617759d9  
**Operating System:** Arago 2017.12 (32 bit RTOS by TI)  
**Kernel:** Linux 4.9.69-g3bf1b0f649  
**Architecture:** armv71

### HID B:

- Scanning distance → 9 – 10 inches approx.
- System Specifications

**Static hostname:** am438x-epos-evm  
**Icon name:** computer  
**Machine ID:** 3211474b24144c21bdec93fb908d4543  
**Boot ID:** da12bc4d1004477eb5cce6757338efad  
**Operating System:** Arago 2018.04 (32 bit RTOS by TI)  
**Kernel:** Linux 4.14.32-g079c2ed3b1  
**Architecture:** armv71

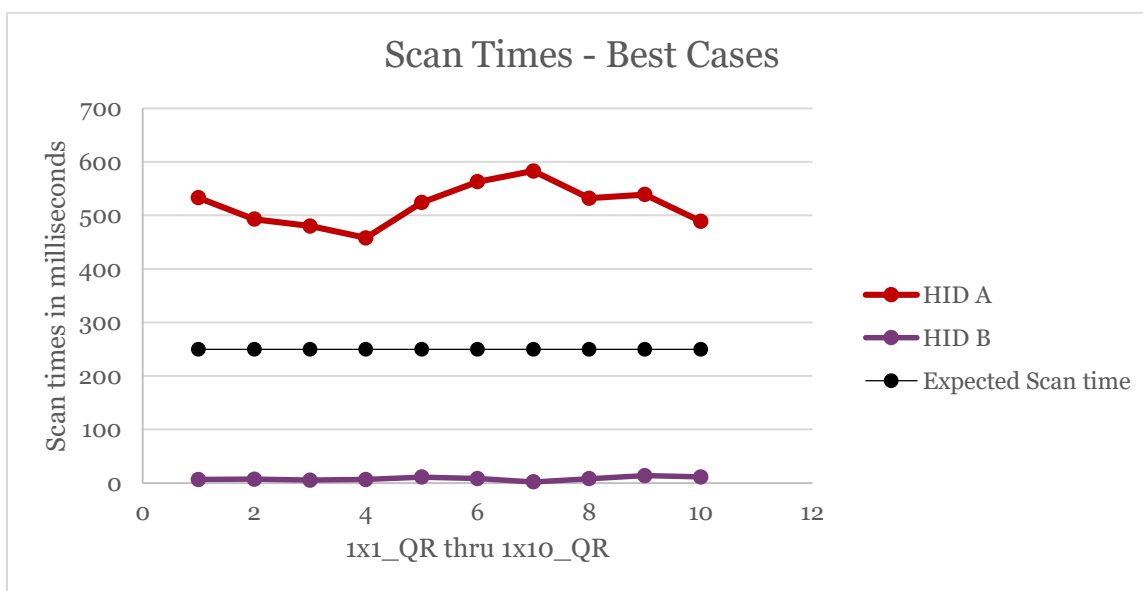
**Table AN 1: QR Ticketing Scan Times**

QR Name	QR Payload Size (bytes) & QR Version	HID A - Scanning Time (T <sub>H</sub> ) milliseconds	HID B - Scanning Time (T <sub>H</sub> ) milliseconds	HID A - Best, Worst T <sub>H</sub> milliseconds	HID B - Best, Worst T <sub>H</sub> milliseconds
1x1_QR	Size: 312 Version: 13	533	6.65	Best: 533 Worst: 887	Best: 6.65 Worst: 20.06
		643	9.81		
		887	13.97		
		638	15.75		
		650	16.16		
		538	16.6		
		602	9.96		
		563	20.06		
		581	14.04		
		550	9.44		
1x2_QR	Size: 358 Version: 14	592	7.27	Best: 493 Worst: 806	Best: 7.27 Worst: 18.33
		528	11.95		
		528	10.33		
		646	12.07		
		493	15.85		

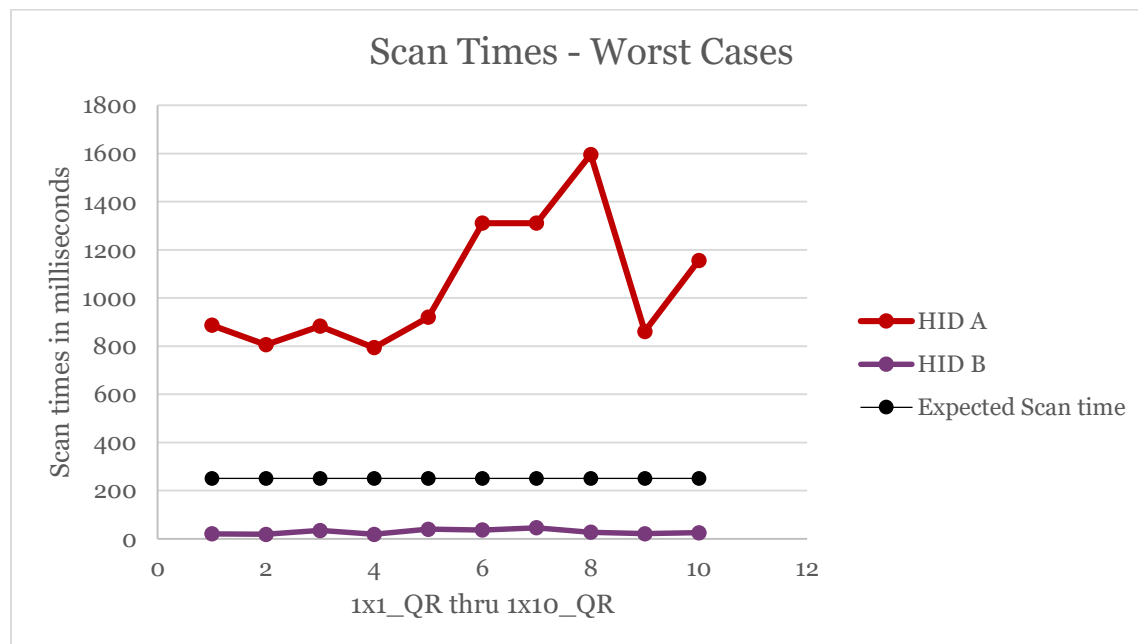
		523	14.57		
		541	18.33		
		564	10.49		
		516	8.09		
		806	8.57		
1x3_QR	Size: 400 Version: 15	508	5.35	Best: 480 Worst: 883	Best: 5.35 Worst: 34.02
		480	13.46		
		527	8.4		
		744	12.17		
		491	10.01		
		541	34.02		
		716	21.69		
		774	16.27		
		883	10.59		
		563	14.09		
1x4_QR	Size: 446 Version: 16	794	8.11	Best: 458 Worst: 794	Best: 6.64 Worst: 18.28
		633	13.63		
		642	11.55		
		550	8.77		
		458	18.28		
		716	11.11		
		675	11.72		
		586	9.39		
		640	6.75		
		540	6.64		
1x5_QR	Size: 492 Version: 17	614	11.76	Best: 524 Worst: 920	Best: 11.05 Worst: 39.74
		920	39.74		
		787	14.36		
		556	11.87		
		555	11.9		
		571	11.7		
		847	14.2		
		524	11.05		
		626	15.41		
		616	16.46		
1x6_QR	Size: 534 Version: 18	584	36.063	Best: 563 Worst: 1311	Best: 8.43 Worst: 36.063
		635	14.23		
		1311	8.59		
		622	13		
		602	10.22		
		563	9.47		
		778	8.43		
		662	31.27		
		635	19.72		
		681	14.06		

1x7_QR	Size: 576 Version: 19	609	17.47	Best: 583 Worst: 1311	Best: 2.11 Worst: 45.46
		607	15.25		
		1153	15.34		
		585	17.4		
		583	45.46		
		584	2.11		
		635	6.524		
		1311	9.01		
		622	13.05		
		602	15.74		
1x8_QR	Size: 622 Version: 19	609	8.068	Best: 532 Worst: 1596	Best: 8.068 Worst: 27.032
		607	23.85		
		1153	20.11		
		585	12.26		
		583	8.61		
		1596	10.83		
		630	19.34		
		532	23.85		
		1257	27.32		
		539	14.21		
1x9_QR	Size: 668 Version: 21	584	15	Best: 539 Worst: 861	Best: 13.69 Worst: 21
		686	21		
		601	18.47		
		539	14.6		
		603	17.41		
		588	16.55		
		600	13.69		
		605	17.86		
		588	13.95		
		861	16.28		
10x10_QR	Size: 712 Version: 22	1005	25.21	Best: 489 Worst: 1156	Best: 11.45 Worst: 25.25
		619	16.53		
		620	15.84		
		489	17.57		
		902	25.25		
		759	14.06		
		1156	14.3		
		658	16.67		
		609	12.4		
		568	11.45		

We plot the Best and Worst case times for both HID<sub>s</sub> against the desired expected threshold of 250 milliseconds. It is clear that **HID A** does not meet desirable threshold in any case whereas HID performs with panache and delivers QR payloads nearly 5 times faster than the desired threshold. This also proves that the QR Dataset is fine and does not affect scannability.



**Figure 10: Best Case Scan Times for 1x1\_QR through 1x10\_QR**



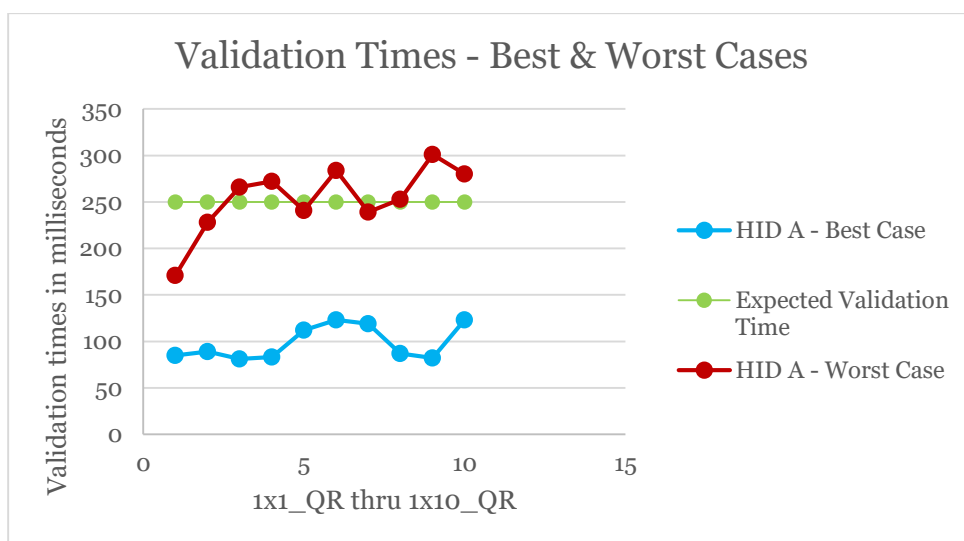
**Figure 11: Worst Case Scan Times for 1x1\_QR through 1x10\_QR**

Recall that the time for business rules validation is  $T_S$  = Time to Parse Data + Time to Validate Ticket. Since the QR Tickets were actual live tickets, some business rules validation tests were performed on them only on **HID A**, including Digital Signature Verification, Tkt\_SI\_No authenticity with AFC System, correctness of journey information like Source, Destination and Journey Time and Ticket Validity, etc. The same set of tickets used for the Scan Time Tests have been used for these tests as well. The results are only a best and worst case times and shown in the table below:

**Table AN 2: QR Ticketing Sample Validation Times**

QR Name	HID A – Best Validation Time ( $T_S$ ) milliseconds	HID A – Worst Validation Time ( $T_S$ ) milliseconds
1x1_QR	85	171
1x2_QR	89	228
1x3_QR	81	266
1x4_QR	83	272
1x5_QR	112	241
1x6_QR	123	284
1x7_QR	119	239
1x8_QR	87	253
1x9_QR	82	301
1x10_QR	123	280

A graphical plot of the same is shown in the figure below. As seen from the graph some of the worst case times – specifically 1x3\_QR, 1x4\_QR, 1x6\_QR, 1x9\_QR and 1x10\_QR – have overshoot the desirable validation threshold of 250 milliseconds.



**Figure 12: Best & Worst Case Validation Times 1x1\_QR through 1x10\_QR on HID A**

## Appendix III – Cryptography Theoretical Definitions

### Definitions of different security schemes

#### HMAC

A cryptographic checksum that results from passing data through a Message Authentication Algorithm is called a MAC. In this standard, the message authentication algorithm is called HMAC or Hashed MAC and the result of applying the HMAC is called the MAC. The message authentication code uses a cryptographic key in conjunction with a hash function.

SHA-2, Secure Hash Algorithm – 2, is a popular security system based on the principles of HMAC cryptography. For a detailed description on SHA-256 and HMAC-SHA, please refer ISO/IEC 9797-2:2011 and RFC 6234 [\[3\]](#).

#### AES

AES or systems are based on the branch of cryptography called Symmetric cryptography that involves the maintenance of a secret private key (shared secret) between all stakeholders that need to exchange the messages intended to be kept secure. The AES or Advanced Encryption Standard cryptographic algorithm has been implemented with both software and with hardware using FPGA (Field Programmable Gate Arrays) and VHDL (Very High-speed Integrated Circuit Hardware Description Language).

For a detailed description on how the AES-256 cryptography (256 bits) refer to the FIPS publication and the Wikipedia page [\[4\]](#).

#### RSA-based security algorithm

RSA-based security systems are based on the branch of cryptography called Asymmetric cryptography that consists of private and public key pairs. The public key is the pair  $[e, n]$  and the private key is the pair  $[d, n]$ . The public key is shared among all the involved parties while each party maintains its own private key. The cryptic message is formed (encrypted) and used (decrypted) by performing some mathematical functions (viz. Totient and Modular Inverse functions) on both the keys. This form of Public Key Cryptography when used in conjunction with digital signatures is called Public Key Infrastructure (PKI). For a more detailed explanation of the algorithm please refer to RFC 8017 [\[5\]](#).

## RSA-based algorithm – description and example

The algorithm consists of 3 phases – key generation, encryption and decryption.

### Key Generation

1. The first phase in using RSA is generating the public/private keys. This is accomplished as follows. Find two random, very large prime numbers  $p$  and  $q$  and calculate  $n=pq$ . Let us assume  $n = 1024$  bits or over 300 decimal digits. It is assumed that the message  $M$  - represented as a number - is smaller than  $n$ . If it's not, we'd have to split the message into multiple block size of a pre-determined block size  $<$  than  $\text{size}(n)$ .
2. Select a small odd integer  $e$  that is relatively prime to  $\phi(n)$ , which is Euler's totient function.

For  $n=p.q$  where  $p$  and  $q$  are primes, we get

$$\phi(n) = (p-1).(q-1)$$

It's recommended to pick  $e$  from a set of known prime values. In practice, common choices for  $e$  are 3, 5, 17, 257 and 65537 ( $2^{16}+1$ ). These particular values are chosen because they are primes and make the modular exponentiation operation faster, having only two bits of value 1. 65537 is the recommended value for  $e$ . Picking this known number does not diminish the security of RSA, and has some advantages such as efficiency.

3. Compute  $d$  as the modular multiplicative inverse of  $e$  modulo  $\phi(n)$ .

At this point we have all we need for the public/private keys. The **public key is the pair  $[e, n]$**  and the **private key is the pair  $[d, n]$** . In practice, when doing decryption, we have access to  $n$  already (from the public key), so  $d$  is remains the only unknown.

### Encryption and Decryption

For encryption, the input is the message block  $M$  and the exponent is  $e$ :

$$\text{Encrypt}(M) = M^e \pmod{n}$$

For decryption, the input is the Ciphertext  $C$  and the exponent is  $d$ :

$$\text{Decrypt}(C) = C^d \pmod{n}$$

### Example

This is an extremely simple example using small numbers you can work out on a pocket calculator.

### Generate Keys

1. Select primes  $p=11$ ,  $q=3$



2.  $n = pq = 11 \cdot 3 = 33$   
 $\varphi(n) = (p-1)(q-1) = 10 \cdot 2 = 20$

3. Choose  $e=3$

Check  $\gcd(e, p-1) = \gcd(3, 10) = 1$  (i.e. 3 and 10 have no common factors except 1),

Check  $\gcd(e, q-1) = \gcd(3, 2) = 1$

Therefore  $\gcd(e, \varphi(n)) = \gcd(e, (p-1)(q-1)) = \gcd(3, 20) = 1$

4. Compute  $d$  such that  $e \cdot d \equiv 1 \pmod{\varphi(n)}$

i.e. compute  $d = (1/e) \bmod \varphi(n) = (1/3) \bmod 20$

This is known as modular inversion. Note that this is not integer division. The modular inverse  $d$  is defined as the integer value such that  $ed \equiv 1 \pmod{\phi}$ . It only exists if  $e$  and  $\phi$  have no common factors.

i.e. find a value for  $d$  such that  $\varphi(n)$  divides  $(ed-1)$

i.e. find  $d$  such that 20 divides  $3d-1$ .

Simple testing ( $d = 1, 2, \dots$ ) gives  $d = 7$

Check:  $ed-1 = 3 \cdot 7 - 1 = 20$ , which is divisible by  $\varphi(n)$ .

5. Public key =  $(n, e) = (33, 3)$

Private key =  $(n, d) = (33, 7)$ .

### Encryption and Decryption

This is actually the smallest possible value for the modulus  $n$  for which the RSA algorithm works. Now say we want to encrypt the message  $m = 7$ .

$\text{Encrypt}(M) = M^e \pmod{n}$

$C = 7^3 \pmod{33} = 343 \pmod{33} = 13$ .

Hence the Ciphertext  $C = 13$ .

To compute decryption, recall

$\text{Decrypt}(C) = C^d \pmod{n}$

$M = 13^7 \pmod{33} = 7$

Several algorithms exist to calculate large exponents i.e.  $M^e$  or  $C^d$  efficiently rather than by brute force. Using the Chinese Remainder Theorem [9] for speeding up exponentiation is also recommended by several experts.

## Appendix IV – Simple Secret Key Sharing Mechanism

### Example – how to compute a “secret key” from a random number

In cryptography, sharing secret keys is commonplace and is an integral part of the security system. Especially in case of symmetric cryptography, the same key needs to be used by both the parties involved in the scheme. The encrypted message can be sent along with some random number which must be enough to derive the final secret key. Here we show a simple yet effective way of generating the secret key from a random number. The following array (table) consists of 1000 random digits (1 – 9) in any random order. A random number is chosen between 1 and 981. This random number is then used as a lookup to index into the table and the next N digits can be used as a secret key.

```
secret_table[1000] = {
2,9,7,4,9,4,4,5,9,2,3,7,8,1,6,4,6,2,8,6,2,8,9,9,8,6,2,8,3,4,8,2,5,3,4,
2,1,1,7,6,7,9,8,2,1,4,8,8,6,5,1,3,2,8,2,3,6,6,4,7,9,3,8,4,4,6,9,5,5,5,
8,2,2,3,1,7,2,5,3,5,9,4,8,1,2,8,4,8,1,1,1,7,4,5,2,8,4,1,2,7,1,9,3,8,5,
2,1,1,5,5,5,9,6,4,4,6,2,2,9,4,8,9,5,4,9,3,3,8,1,9,6,4,4,2,8,8,1,9,7,5,
6,6,5,9,3,3,4,4,6,1,2,8,4,7,5,6,4,8,2,3,3,7,8,6,7,8,3,1,6,5,2,7,1,2,1,
9,9,1,4,5,6,4,8,5,6,6,9,2,3,4,6,3,4,8,6,1,4,5,4,3,2,6,6,4,8,2,1,3,3,9,
3,6,7,2,6,2,4,9,1,4,1,2,7,3,7,2,4,5,8,7,6,6,6,3,1,5,5,8,8,1,7,4,8,8,1,
5,2,9,2,9,6,2,8,2,9,2,5,4,9,1,7,1,5,3,6,4,3,6,7,8,9,2,5,9,3,6,1,1,3,3,
5,3,5,4,8,8,2,4,6,6,5,2,1,3,8,4,1,4,6,9,5,1,9,4,1,5,1,1,6,9,4,3,3,5,7,
2,7,3,6,5,7,5,9,5,9,1,9,5,3,9,2,1,8,6,1,1,7,3,8,1,9,3,2,6,1,1,7,9,3,1,
5,1,1,8,5,4,8,7,4,4,6,2,3,7,9,9,6,2,7,4,9,5,6,7,3,5,1,8,8,5,7,5,2,7,2,
4,8,9,1,2,2,7,9,3,8,1,8,3,1,1,9,4,9,1,2,9,8,3,3,6,7,3,3,6,2,4,4,6,5,6,
6,4,3,8,6,2,1,3,9,4,9,4,6,3,9,5,2,2,4,7,3,7,1,9,7,2,1,7,9,8,6,9,4,3,7,
2,7,7,5,3,9,2,1,7,1,7,6,2,9,3,1,7,6,7,5,2,3,8,4,6,7,4,8,1,8,4,6,7,6,6,
9,4,5,1,3,2,5,6,8,1,2,7,1,4,5,2,6,3,5,6,8,2,7,7,8,5,7,7,1,3,4,2,7,5,7,
7,8,9,6,9,1,7,3,6,3,7,1,7,8,7,2,1,4,6,8,4,4,9,1,2,2,4,9,5,3,4,3,1,4,6,
5,4,9,5,8,5,3,7,1,5,7,9,2,2,7,9,6,8,9,2,5,8,9,2,3,5,4,2,1,9,9,5,6,1,1,
2,1,2,9,2,1,9,6,8,6,4,3,4,4,1,8,1,5,9,8,1,3,6,2,9,7,7,4,7,7,1,3,9,9,6,
5,1,8,7,7,2,1,1,3,4,9,9,9,9,9,9,8,3,7,2,9,7,8,4,9,9,5,1,5,9,7,3,1,7,3,
2,8,1,6,9,6,3,1,8,5,9,5,2,4,4,5,9,4,5,5,3,4,6,9,8,3,2,6,4,2,5,2,2,3,8,
2,5,3,3,4,4,6,8,5,3,5,2,6,1,9,3,1,1,8,8,1,7,1,1,3,1,3,7,8,3,8,7,5,2,8,
8,6,5,8,7,5,3,3,2,8,3,8,1,4,2,6,1,7,1,7,7,6,6,9,1,4,7,3,3,5,9,8,2,5,3,
4,9,4,2,8,7,5,5,4,6,8,7,3,1,1,5,9,5,6,2,8,6,3,8,8,2,3,5,3,7,8,7,5,9,3,
7,5,1,9,5,7,7,8,1,8,5,7,7,8,5,3,2,1,7,1,2,2,6,8,6,6,1,3,1,9,2,7,8,7,6,
6,1,1,1,9,5,9,9,2,1,6,4,2,1,9,8,9,3,8,9,5,2,5,7,2,1,6,5,4,8,5,8,6,3,2,
7,8,8,6,5,9,3,6,1,5,3,3,8,1,8,2,7,9,6,8,2,3,3,1,9,5,2,3 }
```

So for example, if the random number index is as shown above, then the 19 digits (N=19) from that index shall be the secret key. This secret key itself need not be sent over the wire every time a Ciphertext is exchanged. Only the random index needs to be sent. Nobody can guess the secret from the random number so the secret remains tight. The table, however, must be maintained at both the sender and receiver but that again is only a One-time activity.

\*\*\*\*\* End of Part II \*\*\*\*\*

## **Part III: Interface Specification – QR Ticketing Solution for Transit Operators**

## Contents

1. Introduction .....	8
2. Acronyms and Abbreviations .....	8
3. Terms and Definitions .....	9
4. QR Ticketing System Workflow.....	11
4.1. Logical Entities of the QR Ticketing System .....	11
4.2. Architecture .....	13
4.2.1. QR Ticketing System TG and External Entity Communication Standards .....	15
4.3. Segmented Workflows and Interfaces.....	16
4.3.1. Ticketing Client Application (App Provider) and TG Interface .....	16
4.3.1.1. Route Request and Response Scenarios .....	17
4.3.1.2. Fetch Fare Details.....	32
4.3.1.3. User Makes Payment for QR Ticket .....	37
4.3.1.4. Ticket Request to TG .....	46
4.3.1.5. QR Ticket Response from TG.....	49
4.3.1.6. APP Renders QR Ticket image .....	53
4.3.2. TG and AFC System Interface.....	54
4.3.2.1. Purchase QR Transaction Request from TG to AFC .....	57
4.3.2.2. Purchase QR Transaction Response from AFC.....	64
4.3.2.3. Exchange of QR Information with Typical AFC Systems .....	68
4.3.3. QR Payload Media and Validator Interface.....	69
4.3.3.1. QR Payload Transfer to Validator.....	69
4.3.3.2. QR Watchdog in Validating Terminal .....	69
4.3.3.3. Direct API Mode QR Validation in Terminal.....	70
4.3.3.4. QR Verification and Validation in Validating Terminal .....	72
4.3.3.5. Validating Terminal and AFC System Interface .....	78
4.3.4. Optional Futuristic Features .....	78
5. Integrated TG Interface with External Entities .....	79
5.1. PTO Updates Policy DB.....	80
5.1.1. Update Policy Message Exchange .....	84
5.1.2. Generic Fare Calculation Scheme .....	93
5.1.3. Query Policy Message Exchange .....	100
5.2. PSP – APP– TG – AFC Interface .....	103
5.2.1. PSP – APP Provider– TG Interface .....	103
5.2.1.1. PSP Settles Amount with Operators .....	104

## QR Ticketing System – Interface Specification

5.2.1.2.	APP Provider sends MIS report to TG .....	108
5.2.2.	TG sends Payment Details to AFC .....	110
5.2.3.	AFC Acknowledges PTO Payment Details sent by TG .....	115
5.2.4.	Reconciliation of Missed / Delayed Payments.....	117
5.2.5.	Refund and Ticket Cancellation .....	117
5.3.	TG – APP Provider Interface.....	119
5.4.	TG – TOM Interface.....	123
5.4.1.	TG-TOM Workflow – Ticket Request and Response .....	123
5.4.2.	Payment Reconciliation for Paper QR Tickets.....	124
6.	References .....	126
Appendix – I: QR Validation & Customer Care.....		127
Appendix – II: Usage of Different Scanning Media .....		134
Appendix – III: Futuristic QR Ticketing System .....		137
Appendix – IV: File Naming Conventions .....		141

## List of Figures

Figure 1: The QR Ticket Life Cycle .....	11
Figure 2: Single Operator Architecture .....	13
Figure 3: Multi-TG Multi-Operator Architecture .....	15
Figure 4: Fetch Route / Station Information .....	20
Figure 5: Fetch All Routes Request JSON for Operator ID 135 .....	22
Figure 6: Route Info Response JSON from TG for Operator ID 135 .....	27
Figure 7: Fetch All Stations Request JSON for Route 8011 – Operator ID 135 .....	28
Figure 8: Route Response with Stations JSON from TG for Operator ID 135 .....	28
Figure 9: Fetch All Stations Request JSON Operator ID 10 .....	29
Figure 10: Route Response with Stations JSON from TG for Operator ID 10 .....	30
Figure 11: Fetch Fare Details Workflow .....	32
Figure 12: Fetch Fare Request JSON by APP for Operator ID 10 & Operator ID 135 .....	35
Figure 13: Fetch Fare Response JSON by TG for Selected Journeys .....	36
Figure 14: Make Payment and Purchase QR Ticket .....	37
Figure 15: APP Provider sends Payment Details to PSP .....	42
Figure 16: PSP sends Transaction Response to APP Provider .....	45
Figure 17: Customer Selects Journey and Makes Payment .....	46
Figure 18: Ticket Request JSON from APP .....	48
Figure 19: QR_SVC JSON format .....	50
Figure 20: QR Ticket Block JSON .....	51
Figure 21: Ticket Response JSON from TG .....	53
Figure 22: Handle Ticket Response and Render QR Image .....	54
Figure 23: TG prepares Purchased QR for Scheduler .....	55
Figure 24: Operator-specific Transaction File Format (Generic) .....	56
Figure 25: Purchase QR Transaction File qr_12062020185001_23_10.JSON .....	58
Figure 26: Purchase QR Transaction File qr_12062020171512_23_135.JSON .....	59
Figure 27: QR Purchase Transaction Request Operator ID 10 .....	62
Figure 28: QR Purchase Transaction Request Operator ID 135 .....	63
Figure 29: QR Purchase Transaction Response Operator ID 10 .....	65
Figure 30: QR Purchase Transaction Response Operator ID 135 .....	66
Figure 31: Send QR Info to AFC System & Receive Response .....	67
Figure 32: Scanner and Validator Integration (Watchdog) .....	70
Figure 33: Scanner and Validator Integration (Direct mode) .....	71
Figure 34: Integrated TG (TG + Policy DB) Interfaces .....	79
Figure 35: Update Policy ID#1 File qrp_u_06042020142035_10_23.JSON .....	86
Figure 36: Update Policy ID#2 File qrp_u_06042020142036_10_23.JSON .....	87
Figure 37: Update Policy ID#3 File qrp_u_06042020142037_10_23.JSON .....	88
Figure 38: Update Policy ID#4 File qrp_u_06042020142038_10_23.JSON .....	89
Figure 39: Update Policy ID#5 File qrp_u_06042020142039_10_23.JSON .....	90
Figure 40: Update Policy Response for Policy ID#2 – TG to AFC .....	93
Figure 41: Update Policy ID#6 File qrp_u_06042020142040_10_23.JSON .....	98
Figure 42: Query Policy ID#6 Request Data .....	101
Figure 43: Query Policy ID#6 Payload Data .....	102
Figure 44: Query Policy#6 Response File qrpq_07042020180130_10_23 – TG to AFC .....	103
Figure 45: MIS Report – PSP to APP Provider .....	107

Figure 46: MIS Report – APP Provider to TG .....	109
Figure 47: Generic Message Format for Payment Settlement – TG to AFC.....	111
Figure 48: Financial Transaction Payload Data – Operator ID 10 .....	112
Figure 49: Financial Transaction Payload Data – Operator ID 135 .....	113
Figure 50: MIS Report for Operator ID 10 – qrf_13062020020103_23_10.JSON .....	114
Figure 51: MIS Report for Operator ID 135 – qrf_13062020020103_23_135.JSON .....	114
Figure 52: Response from Operator ID 10 for MIS Report .....	116
Figure 53: Response from Operator ID 135 for MIS Report .....	116
Figure 54: Workflow for Refunds and Cancellation .....	118
Figure 55: Payload Data of App Registration Request –App Provider to TG .....	121
Figure 56: Payload Data of App Registration Response – TG to App Provider .....	122
Figure 57: TOM – TG Interface.....	123
Figure 58: Payload Data for Paper QR MIS Report Request – TG to AFC .....	124
Figure 59: Payload Data for Paper QR MIS Report Response – AFC to TG .....	125
Figure 60: Customer Care Workflow Diagram .....	127
Figure 61: QR Scanning Options.....	134
Figure 62: Real-time Status Update on Mobile with Relay Server.....	139
Figure 63: App Receives Status Update from Relay Server .....	140

## List of Tables

Table 4.1: Modes of Communication in QR Ticketing System .....	16
Table 4.2: Data Type Representation Used In Document.....	17
Table 4.3: Data Elements of Route Request .....	18
Table 4.4: Data elements of Route-Station Response .....	23
Table 4.5: Route Details Element of Route-Station Response.....	25
Table 4.6: Data elements of Fetch Fare Request.....	33
Table 4.7: Data elements of Fetch Fare Response.....	34
Table 4.8: Data elements of Make Payment Transaction Request.....	38
Table 4.9: Data elements of Multi-account Payment Request.....	40
Table 4.10: Data elements of Multi-account Payment Response.....	42
Table 4.11: Data elements of QR Ticket Request .....	47
Table 4.12: Data elements of QR Ciphertext .....	52
Table 4.13: Structure for Message Exchange between TG & AFC .....	55
Table 4.14: Length of Payload.....	56
Table 4.15: Structure for Purchase QR Request Payload – TG to AFC.....	60
Table 4.16: Purchase QR Request Message – TG to AFC .....	61
Table 4.17: Structure for Purchase QR Response Payload – AFC to TG.....	64
Table 4.18: Purchase QR Response Message AFC to TG.....	65
Table 4.19: QR Transaction Type Values.....	73
Table 5.1: Update Policy Query Structure .....	80
Table 5.2: Policy Update Request Message from AFC .....	85
Table 5.3: Update Policy Response Structure .....	91
Table 5.4: Update Policy Response Error Codes and Messages .....	92
Table 5.5: Update Policy Response Message – TG to AFC .....	92

Table 5.6: Data elements for Generic Fare Calculation Structure .....	94
Table 5.7: A Simple Distance Matrix – nDistanceMatrix.....	97
Table 5.8: A Simple Fares Matrix – nFaresMatrix .....	99
Table 5.9: Generic Query Policy Request Message Format .....	100
Table 5.10: Generic Query Policy Response Message Format.....	101
Table 5.11: PSP to APP Provider MIS Report Structure .....	105
Table 5.12: Financial Transaction Request Message Format– TG to AFC.....	113
Table 5.13: Financial Transaction Response Message Format – AFC to TG.....	115
Table 5.14: APP Provider Details Structure in Policy DB.....	119
Table 5.15: Structure for App Registration Request Message with TG .....	120
Table 5.16: Structure for App Registration Response Message with TG .....	121
Table 5.17: Message Format for Paper QR MIS Report Request & Response.....	125



### Version History

Date	Version	Author	Comments
7/7/2020	1.0	CDAC	Incorporated comments after presentation external stakeholders – DMRC, BMRC, NPCI and BEL on 24 <sup>th</sup> Jun 2020. This Final Draft version was circulated to all PTOs, BEL, BIS, Niti Ayog, STQC, etc. On 23 <sup>rd</sup> Sep 2020
24/02/2021	1.1	CDAC	Amendments related to QR Code Dataset owing to comments received from external stakeholders on V1.0. Also incorporated comments and suggestions received from participants in Workshop conducted by CDAC on 22 <sup>nd</sup> & 23 <sup>rd</sup> Dec 2020.

## 1. Introduction

So far we have spent all the time in describing the building blocks of the QR Code Ticket in terms of its structure, size, type and security. Now it is time to put all the pieces together. In this Part III – QR Interface Specification, we introduce the various logical entities of the QR Ticketing System and describe the logical process flows of these multiple entities interact, what are the various levels of Application Program Interface (API) required in the eco-system, what kind of messages are exchanged, how these messages are structured and finally how they should be transmitted from one entity to another.

## 2. Acronyms and Abbreviations

<b>AFC</b>	Automatic Fare Collection
<b>API</b>	Application Program Interface
<b>APP</b>	Application
<b>BLOB</b>	Binary Large Objects
<b>DSSL</b>	Datagram Secure Sockets Layer
<b>DTLS</b>	Datagram Transport Layer Security
<b>HTTP</b>	Hyper Text Transfer Protocol
<b>MDR</b>	Merchant Discount Rate
<b>NCMC</b>	National Common Mobility Card
<b>PSP</b>	Payment Service Provider
<b>PTO</b>	Public Transport Operator
<b>QR Code</b>	Quick Response Code
<b>RESTful Web Service</b>	Representational State Transfer Architecture
<b>SLA</b>	Service Level Agreement
<b>SSL</b>	Secure Sockets Layer
<b>TCP</b>	Transmission Control Protocol
<b>TG</b>	Ticket Generator

<b>TLS</b>	Transport Layer Security
<b>TOM</b>	Ticket Operating/Office Machine
<b>UI</b>	User Interface
<b>UPI</b>	Unified Payments Interface
<b>VPN</b>	Virtual Private Network

### 3. Terms and Definitions

The following definitions are used throughout this Standard:

- **QR Code Tickets:** Tickets in the form of QR images shall be used for single journey / Group tickets as an alternative to cash based paper ticket and token. QR code tickets shall be used in mobile phones or in Paper Media.
- **HTTPS:** HTTPS is a secured form of HTTP protocol. HTTPS, HTTP/2.0, HTTP over SSL and HTTP with TLS are all synonymous terms. It was designed to make internet communication between remote entities secure and confidential. Security is implemented in the transport layer, which is TCP for all practical purposes. All modern browsers and mobile applications support HTTPS communication. Applications that facilitate financial transactions are mandated by the RBI to use HTTPS as the communication mode.
- **REST API:** It refers to Representational State Transfer Architecture, a form of software architecture usually used for creating Web services. In RESTful Web Service, requests and responses are usually exchanged in the form of HTML/XML/JSON and the communicating channel is HTTPS.
- **Sockets:** Sockets are an endpoint of communication that usually resides in the Application layer of a packet-switched network, e.g. the Internet. Sockets are a lower level form of application programming than, say, application protocols like HTTP or FTP that are encapsulated over some form of sockets. Application Sockets are mainly of two types – depending on the transport protocol they use – SOCK\_STREAM (uses TCP) and SOCK\_DGRAM (uses UDP).
- **APP Provider:** The smart-phone application referred to throughout the specification as ‘APP’ may be provided by the merchant or PTO itself, or any third party aggregator service. This entity is referred to in this specification as the APP Provider.
- **Payment Service Provider:** A Payment Service Provider, or PSP for short, may be a bank or financial institution that processes all existing payment services e.g. Credit/debit cards, Netbanking, Wallets, etc. on behalf of a merchant. The merchant in this case may also be the APP

Provider. It is the role of the PSP to handle financial transactions and distribution to operators. This distribution of funds is a very important function and is discussed in detail in this document.

- **Integrated TG:** The Ticket Generator or TG is a high-end Server that may have many components tightly integrated with it – for example the Policy DB, the Web Service/Web Socket process, the Relay server to relay real-time status updates to the mobile, AFCs, etc. The main TG process per se implements the actual business logic. All these components and modules together are referred to as the Integrated TG. The terms “TG” and “Integrated TG” are used interchangeably in this document and must not be differentiated as separate entities.
- **PTO and Operator:** The terms PTO or Public Transport Operator or Operator are used interchangeably in this document.
- **Allowlist & Denylist:** In the modern Tech world, terms like Whitelist and Blacklist are considered derogatory by most organizations. Henceforth we introduce the terms ‘allowlist’ and denylist’ to remain up-to-date with the changing world.
- **NCMC-compliant AFC System:** Indigenously developed, inter-operable Automatic Fare Collection (AFC) System which is vendor and bank agnostic. Using a single NCMC card (Debit/Credit/Prepaid Card, issued by any Bank), customers can make payment for any purchases and also use it as a travel card for urban transport services like Metro transit, Bus transit, Toll, Parking etc. across the nation. AFC Systems of such transit operators that support usage of NCMC cards are usually called NCMC-compliant AFC Systems.

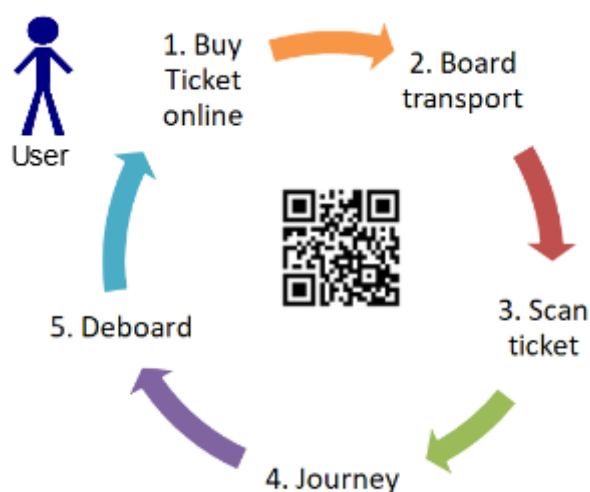
## 4. QR Ticketing System Workflow

In this section we discuss in detail the interface and workflow between all the entities of the QR Ticketing System mainly – TG, App and Validator but along with the other co-operating entities.

### 4.1. Logical Entities of the QR Ticketing System

Here we describe in details all the logical entities that make up the QR Ticketing System as seen in the architecture diagrams above.

The defined entities are logical representations of either a single party or organization and the same party may play the multiple roles.



**Figure 1: The QR Ticket Life Cycle**

**User:** This is the customer. It interacts with the Mobile or Web App to avail QR ticket. In case of a ticket issued by the TOM at the operator premise, the TOM interacts on behalf of this user. The user's roles can be summed up with the diagram above, [Figure 1](#). Note that Steps 2 and 3 may be interchangeable depending on the mode of transport. For example, in case of Metro it is mandatory to get the ticket scanned and verified by a Validating Terminal before access is given into the Operator's premises.

#### **Ticket Generator (TG):**

Ticket Generator is the most sophisticated component of the entire system. All TGs must have a trusted relationship with the operators they serve and the Mobile/Web APP Providers that use its services. The

function of the TG is that it generates the ticket based on various inputs received from the requester, employs security features, and communicates with respective AFC Systems. The TG is tightly coupled with the Policy DB. The TG must be implemented as a Web server coupled with an Application Server (REST API). It should accept all requests like user registration, ticket request, etc. from Smartphone Apps or Webclients on secure channel HTTP over SSL i.e. HTTPS. It can also cater to requests from the Ticket Office if the operator supports Paper-based QRs.

**Policy Database:** Policy database is a centralized repository tightly coupled with the Ticket Generator and stores information related to payments, security, fare tables, etc. for different operators. Only entities trusted by the operator like the TG and Mobile APP can access the Policy DB. Access to Policy DB is only possible with the API provided by the TG.

**PTO:** Public Transport Operator or PTO is the service provider and executes multiple roles within the system. One of the most important roles of the PTO is that it controls the policy parameters including Fares, Security schemes, etc. It can also choose to implement different product policies like discounts for special days, determine the validity period of tickets and even denylisting rogue users. So any updates or changes on those will affect everything from Policy DB to the Mobile Apps.

The three main components of a PTO when seen as a logical entity are:

- **Ticket Authorizer/Validator:** Validates the QR and allows or disallows the commuter to use the facility. The most important requirement of the validator is that it performs the security validation which is the fundamental requirement for offline validation. Please refer to QR Specifications – Part II for details.
- **AFC System:** This is tightly coupled with the AFC DB. As all the tickets (QR) are pre-paid, the primary job of the validator shall be to validate the QR and to update the transit data onto the AFC system. The TG also communicates with the AFC System with information about newly procured QRs. *Communication with the AFC from Validator and TG must take place over a secure TCP channel like SSL/TLS. Application protocols like HTTPS, SFTP, SCP, TCP/IP Sockets, etc. are well-known standard protocols that can be used for the purpose.*
- **Ticket Office Machine or Window** – An optional entity where a person can go and physically buy a QR Ticket in paper form. It uses the same REST API Web service request form as a Mobile APP would when requesting for QR ticket from the TG.

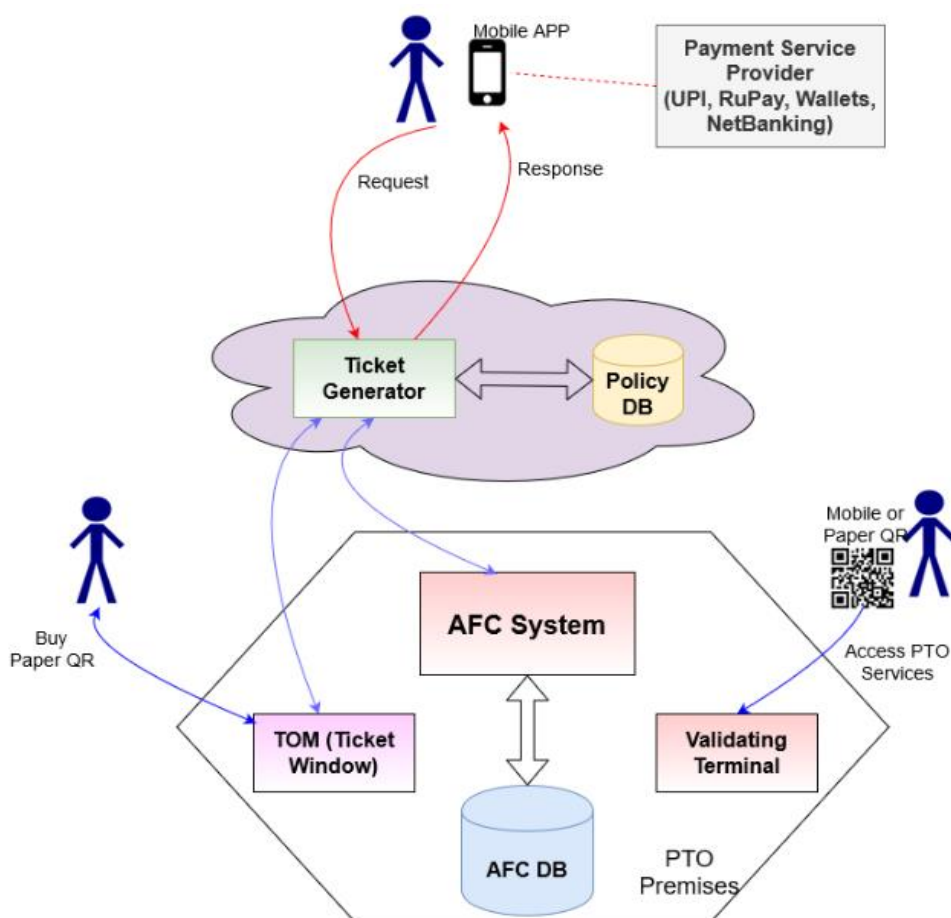
**Mobile (or Webclient) App and App Provider:** The smart phone application or the Webclient application may be provided by the APP Provider and sometimes synonymously referred to as the merchant. Under all circumstances, the APP Provider must have a trust relationship with the TG. APP Provider – TG cardinality relationship described in details in [Section 5.3](#).

## 4.2. Architecture

In this section we shall discuss the two forms of the QR Ticketing System architecture – single operator and multi-operator.

The **Single Operator Architecture** is the basic form that describes the QR Ticketing Solution. The same architecture is then scaled up to cater to multiple operators. The fundamental principle of the Single Operator Architecture is as follows:

- **Single Centralized TG**
- **Single Operator**
- **Single Policy DB**
- **Single PTO Premise - AFC System + AFC DB + Validating terminal + TOM**
- **Single Mobile (or Webclient) APP Provider**

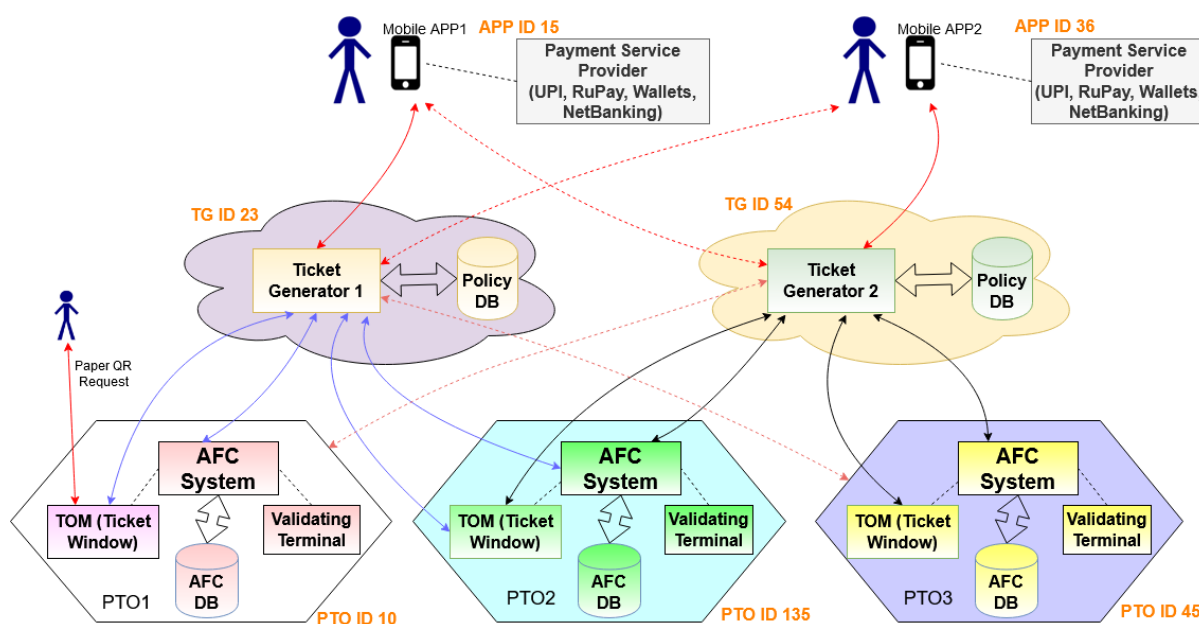


**Figure 2: Single Operator Architecture**

The **Multi-TG Multi-Operator Architecture** is for all practical purposes the actual form that the QR Ticketing Solution is going to be. With this solution, a user would be able to buy and avail services for multiple operators. In [Figure 3](#) below, we only show 2 Ticket Generators and 3 Operators:

- **Multiple Centralized Ticket Generators** – Fundamentally they operate in the same way as the Single operator scheme, except here we conveniently illustrate the possibility that the same TG may associate with multiple operators. Henceforth, TG ID 23 serves Operators with Operator ID 10 and Operator ID 135, whereas TG ID 54 serves Operators with Operator ID 135 and Operator ID 45. Of course there is no restriction on TG ID 23 also serving Operator ID 45, or TG ID 54 also serving Operator ID 10, as shown by the dotted lines.
- **Multiple Operators** – Again fundamentally similar to the single operator architecture, except here we conveniently illustrate the possibility that the same operator may get associated with multiple TGs. As seen in the diagram below, Operator ID 135 is associated with both TG ID 23 and TG ID 54. Again there is no restriction on APP ID 15 associating with TG ID 54 or APP ID 36 associating TG ID 23, as shown by the dotted lines.
- **Single Policy DB** integrated with the TG.
- **AFC System, AFC DB, Validator and TOM** work in the same way as with the single operator architecture.
- **Multiple Mobile (or Webclient) APP Providers**–Mobile APP1 sells services related to the “TG ID 23” operator base, and similarly Mobile APP2 caters only to “TG ID 54” operator base. APP Providers for Mobile APP1 – APP ID 15 and Mobile APP2 – APP ID 36 are different. Many such combinations of Mobile APP and Webclients are also possible.





**Figure 3: Multi-TG Multi-Operator Architecture**

#### 4.2.1. QR Ticketing System TG and External Entity Communication Standards

As seen from Figure 2 and Figure 3, there are a number of external entities that need to communicate with the TG directly like –

- AFC Systems
- App Provider Systems (Webclient or Mobile App)
- Payment Service Provider

Henceforth, looking at these various interfaces, it becomes necessary here to define the communication standards that should be followed by all the entities participating in the QR Ticketing System Ecosystem. Since there may be many choices of communication that an entity can choose from, the external entity needs to opt only one of them. The eventual choice of implementation is a business decision and is left at the discretion of the participation entities. The TG on the other hand must be prepared to support all or as many of these mechanisms.

The table below shows the various mechanisms of communication that may be followed by external entities. The communication is assumed to be bi-directional and the type of data is always plain text or ASCII/Unicode Text which maybe in the form of HTML text, JSON, XML, CSV files or even freeform text. If binary data like encrypted data needs to be exchanged, then it must always be converted into Base64 form.

**Table 4.1: Modes of Communication in QR Ticketing System**

QR Ticketing System Entity	External Entity	Application Protocol / Endpoint	Communication Mechanism*
TG	AFC	<ul style="list-style-type: none"> <li>• HTTP, HTTPS</li> <li>• SSH, SFTP, SCP</li> <li>• Sockets</li> </ul>	<ul style="list-style-type: none"> <li>• TCP/IP and / or its secure variants TLS/SSL</li> <li>• UDP/IP and / or its secure variants DTLS/DSSL</li> </ul>
	PSP	<ul style="list-style-type: none"> <li>• HTTP, HTTPS</li> <li>• SSH, SFTP, SCP</li> <li>• Sockets</li> </ul>	<ul style="list-style-type: none"> <li>• TCP/IP and / or its secure variants TLS/SSL</li> <li>• UDP/IP and / or its secure variants DTLS/DSSL</li> </ul>
	App Provider	<ul style="list-style-type: none"> <li>• HTTP, HTTPS</li> <li>• Sockets</li> </ul>	<ul style="list-style-type: none"> <li>• TCP/IP and / or its secure variants TLS/SSL</li> </ul>

\*The IP part of the communication mechanism can be either the IPv4 or the IPv6 variant. Further, the business needs of the participating entities may demand data to be exchanged over private tunnels like (VPN/IPsec). Discussion of such factors are beyond the scope of this document.

### 4.3. Segmented Workflows and Interfaces

The workflows involving a Mobile APP and the TG are same for both the single and multi-operator models. These flows are divided into many segments and discussed in the sections below. As seen in QR Specifications Parts I and II, data exchange takes place in JSON format.

#### 4.3.1. Ticketing Client Application (App Provider) and TG Interface

\*Please note that the Ticket Purchase Client Application may be the Mobile APP or the Webclient (Web browser APP). Interface with TG for both the clients are exactly the same.

There are several parts in this interface as described in the sections below.

Note: - User registration and management is not part of the Ticket Generator per se but it is local to the APP or Web Service Provider. Registered users will be given the list of PTO IDs and Names by the Application or Web Service Provider.

**Note:** The TG does not maintain any customer information like the username, mobile number or email ID. This is maintained by the App Provider backend system. However, it can still be sent with every request to the TG. In cases of reconciliation, the TG can then use this information for correlation. It is mandatory for online ticket bookings (using Mobile or Webclient Apps) to send the Mobile number along with the Ticket Request. Please refer to [Sections 4.3.1.4](#) and [5.3](#) for details. For TOM bookings – Paper QR – the mobile number is optional.

We shall now describe in the following sections the complete flow of completing a QR Ticket Booking – from requesting route/station information through to QR image rendering.

#### 4.3.1.1. Route Request and Response Scenarios

Only users of a trusted App Provider shall be allowed to buy QR Tickets by using the mobile or Webclient APP. The TG also verifies if the user requesting the 'Routes Info' is using a trusted APP by means of digital signature verification. After this request, the Mobile App or Client shall maintain a session with the TG.

#### Assumption:

Throughout this document we shall only consider one TG with 'TG ID' = 23 and one App Provider with 'Requester ID' = 15. Requester ID is also sometimes referred to as APP ID.

For sake of clarity we first show here the abbreviated forms and descriptions of various data types used throughout this document.

**Table 4.2: Data Type Representation Used In Document**

Abbreviated Data Type	Description
<b>NS</b>	Numeric String – Digits[0-9] only
<b>AS</b>	Alphabetic String – Alphabets [a-zA-Z] only. In special cases like name fields, spaces are allowed.
<b>AN</b>	Alpha-numeric String – Combinations of <b>NS</b> and <b>AS</b> . In special cases like name and address fields, spaces are allowed.
<b>ES</b>	Email String – Standard email form <b>AN@AN</b> . Dot '.', '_', etc. are allowed.
<b>(NS.NS)</b>	Compound numeric fields like Amounts viz. ticket fare, total fare, etc. shall be represented with this Notation. The first <b>NS</b> denotes the Rupees part and the second <b>NS</b> denotes the paisa part included only if it is present. The dot '.' also needs to be specified only if the paisa part is present. Even other fields that require a compound numeric string representation may use this format. For amount fields the separator must always be a '.'. Other fields may use different separators like a ';' or a '^'.
<b>B64S</b>	Base64 String – <b>AN</b> and [+/-], making it a total of 64 possible characters.
<b>DS</b>	Date String – We shall only use one format used throughout this document: DD/MM/YYYY@hh-mm-ss unless otherwise specified explicitly. If two entities having an interface uses different formats then a common format must be agreed upon for that particular interface.

### Processing a Route Request:

The Route Request information sent from the APP is shown in [Table 4.3](#) below.

**Table 4.3: Data Elements of Route Request**

Tag	Field Name	Data type / Presence	Maximum Size (Chars)	Description
Route Request Details	Requester ID	NS / Mandatory	5 Range: 0-65535	The ID of the APP Provider. Requester ID is the same as APP ID for the Mobile APP Provider or the Webclient. For all communications to TG from any Ticket Requesting Client, viz. Mobile APP, Webclient APP or TOM; the term 'Requester ID ' shall be used.
	Operator ID	NS / Mandatory	5 Range: 0-65535	The TG will only send route information details for this Operator/PTO ID and not for all the PTOs. The terms PTO and Operator are used interchangeably in this document.
	Mobile	NS / Conditional	10	Registered 10-digit mobile number of the customer. The customer may choose to login into the Application backend using either his/her registered Username or Mobile Number or Email ID.
	Email ID	ES / Conditional	64	Registered Email ID of the customer. The customer may choose to login into the Application backend using either his/her registered Username or Mobile Number or Email ID.
	Username	AN / Conditional	64	Registered username of the customer. The customer may choose to login into the Application backend using either his/her registered Username or Mobile Number or Email ID.
	Route ID	AN / Optional	8	Route Request is a polymorphic or multi-purpose request message and the "Route ID" field is used for that. This field is sent in the Station Request

## QR Ticketing System – Interface Specification

				message and is used when the Stations related for a particular route is required.
Digital Signature	Route Request Signature	B64S / Mandatory	172	<p>The APP Provider takes a SHA-256 hash of the request details.</p> <p>The APP Provider generates the Digital Signature of the hash (128 bytes) with Private Key.</p> <p>It then creates the Base64 representation of the signature (produces 172 bytes). Please note that a smaller size hash and Digital Signature is also possible.</p>

Please refer to the flowchart below in Figure 4 for the TG's handling of Route Request.

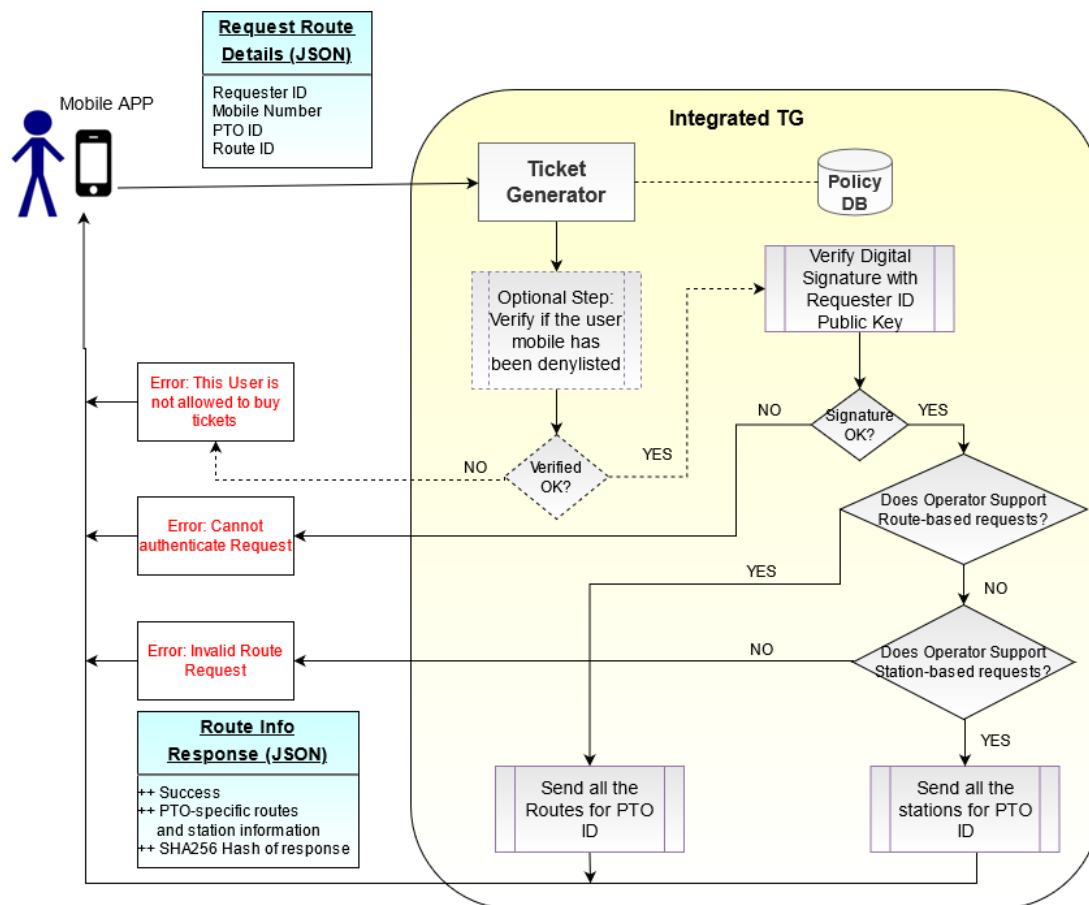


Figure 4: Fetch Route / Station Information

**Notes: -**

- Digital Signature Verification is a mandatory step and must be followed in every request response between APP and TG.
- A 'Denylist' concept is shown here. This is only an optional flow. The PTO can implement a process of marking a user's mobile as 'Denied'. If that happens, then the user cannot buy any tickets until in the future the PTO decides to allow the user again. For this process to work, there needs to be a number of additional interfaces and processes defined between the PTO and the TG. It is beyond the scope of this document.

### **A Note about Data Integrity of Responses from TG:**

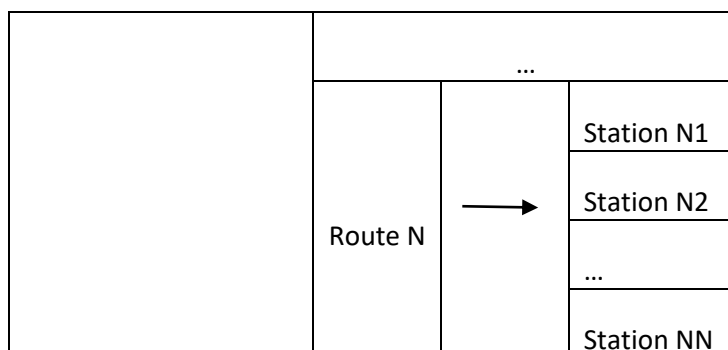
Since it is very much possible that the responses from the TG back to the client application may get tampered i.e. its integrity may be compromised, henceforth the TG sends the SHA256 or higher Hash of the response along with the actual data. The client application must then always take a new Hash of the response with the same algorithm and compare with the Hash sent by the TG.

From here on, we will assume all requests and responses exchanged by the client Application and the TG follow this protocol, irrespective of whether it has been explicitly displayed or not in the data flows.

**Operation Type:** Some PTOs have routes based information whereas others simply define only station-based information. Most Bus Operators in India define a hierarchical structure from Routes to Stations. Some Metro Operators also define a hierarchy of a top-level Line Info and then Stations falling on that Line. However, since Metro networks are inter-connected, operators seldom use Line Info and only specify Stations Info at the top-level. The TG defines a parameter ‘Operation Type’ just to make this aforementioned distinction. It is essential for the Operators to send this value as a Policy DB parameter. The diagrams below further elucidate this concept.

<b>Operation Type* = 0 (Non-route based);</b> Operator uses only Stations Info and does not define routes or line info. Usually for Metro operators, all stations are interconnected with interchanges so Line information is redundant and the system can easily show this information to user.	Station 1
	Station 2
	Station 3
	...
	Station N

<b>Operation Type* = 1 (Route based);</b> Operator uses for Route/Line Info and then under each route stations are defined. Bus operators usually define routes and stops/stations falling under that route.	Route 1	→	Station 11
			Station 12
			...
			Station 1N
	Route 2	→	Station 21
			Station 22
			...
			Station 2N



\* Refer Table 5.1.

### Multiple Route/Station Request(s):

#### Assumptions:

An example with multiple journey tickets shall be followed in this document – 1 Single Journey Ticket for Operator ID = 135 (Operation Type = 1) and 2 Single Journey Tickets for Operator ID = 10 (Operation Type = 0). Only the TG knows if an operator supports Non-route based (Operation Type = 0) or Route-based (Operation Type = 1) operations. Bear in mind that the 'Route Request/Response' is generic and should be used to fetch stations information as well.

With these assumptions, now we can send the Route Request to the TG. The user selects Operator 135 and sends the Route Request to the TG. The JSON Route Request is as shown in Figure 5 below.

```
{
  "Route_Request": {
    "Requester_Id": "15",
    "Operator_Id": "135",
    "Mobile": "9201345678",
    "Email": "shriroop@web.in",
    "Username": "roop9201"
  },
  "Route_Request_Signature": {
    "KdkjsahdJYuashb+YMMCq7MjJuVDiFGjw47ZkvfvjEkmw6wvVRNzlBn0BYSpKoi8KrAR
wCLgk8 0vDdi8z2UOoMFpdeDusjSqt07kM73O6z2vY"
  }
}
```

Figure 5: Fetch All Routes Request JSON for Operator ID 135

*The communication between Mobile APP/Ticket Webclient and TG must be over HTTPS and the format of message is through REST API/JSON.*

*The typical way that REST API over HTTPS works is that the Server sends an 'access token' after it has satisfactorily determined that the "Request" it has received is a valid one. The clients must use this 'access*



*token' in all requests sent during the current session with the Server. This step is trivial and implementation specific and hence describing it is not within the scope of this document.*

Now we shall describe the response that the TG sends for the route request.

**Route Response:** The TG will verify the Digital Signature in response to the Route Request, the TG responds with either “Error code” = 0 meaning Success or “Error code” >= 1 meaning Error. When there is an error, the error text will be filled with the exact error message and all the remaining details will be empty. The TG sends the route/station information of PTO/Operator back to the APP. Some other details related to the PTO – like Product ID, Service ID, etc. – are also sent along on this response.

**Table 4.4: Data elements of Route-Station Response**

Field Name	Data type / Presence	Maximum Size (Chars)	Description
Error Code	NS / Mandatory	1	<p>Either 0 (Success) or [1-9] (Error). Application specific errors:</p> <p>1 – Invalid Request</p> <p>2 – Invalid Request Signature</p> <p>3 – User is denied from buying QR Tickets</p> <p>4-6 – RFU</p> <p>7-9 – PTO-specific</p>
Error Text	AN / Optional	64	<p>The error text will contain error message if return code is an error. Some strings defined are:</p> <p><i>Success</i></p> <p><i>Invalid Request</i></p> <p><i>Invalid Request Signature</i></p> <p><i>User is denied from buying QR Tickets</i></p>
Operator ID	NS / Mandatory	5 Range: 1-65535	The ID of the Operator as defined in the QR Ticketing System. All participating entities shall use that same ID throughout.
Operation Type	NS / Mandatory	1	Whether the operator supports non-route based requests (=0) or route-based requests

## QR Ticketing System – Interface Specification

		Range: 1-9	(=1). It is a Policy DB parameter sent by the PTO.
Product ID	NS / Mandatory	3 Range: 0-255	This field is the same field shown inside the operator's Ticket Info element described in the QR Specifications – Part I. <i>Although this is a 2-byte field, presently only 1 byte is used.</i> There may be many products – Standard Product (default), Senior Citizen Pass, Student Pass, etc. as defined in the Product ID Table 5.14 of Part I – QR Specifications. 3 characters only specify the Product ID number. <i>For request-response, when an operator supports many products the string should be separated with semi-colons e.g. "1-Standard;3-Student;4-Daily;5-Monthly" if this operator defines only these products.</i>
Service ID*	NS / Optional	3 Range: 0-255	This field is the same field shown inside the operator's Ticket Info element described in the QR Specifications – Part I. There may be many services offered by the operator – Regular Metro, Express Metro, Feeder Bus, etc. as defined in the Service ID Table 5.15 of Part I – QR Specifications. 3 characters only specify the Service ID number. <i>For request-response, when an operator supports many services the string should be separated with semi-colons e.g. "0-Normal Metro;1-Express Metro;21-Normal Feeder Bus" if this operator offers these services.</i>
Validity	NS / Mandatory	5 Range: 0-65535	This field is the same field defined in a Ticket element and described in the Terms and Definitions section of QR Specifications Part I. Tickets for urban transit operators like Metro or Buses are usually valid for the entire day. However, it is actually operator-specific and consistent for all tickets. The unit is in minutes. Here we have shown it as 480 minutes or 8 hours.
Duration	NS / Optional	5 Range: 0-65535	This field is the same field defined in a Ticket element and described in the Terms and Definitions section of QR Specifications Part I. For urban transit operators like Metro or

			Buses after a ticket is validated till this 'duration' the user can avail the service. When exiting the system, if the duration has elapsed then it is an error. The unit is in minutes. Here we have shown it as 180 minutes or 3 hours.
Route Details	Compound / Mandatory	Variable	Please refer <a href="#">Table 4.5</a> . If an operation is Route-based, then the route details are sent here. Once a particular route is selected and a Station Request is sent, all the stations for that route are sent. If the operation type is non-route based, all the station details are sent and the route details are empty.

**\*Note: - Service ID:** An important note about the Service ID field is as follows – In case of non-route based requests, which for all practical purposes, is all Metro-type operations, this field can be sent along with the information above. However, in case of route-based requests, each route may or may not support a particular service. For example, a particular route may only run a regular service but not an AC service. Henceforth, for route-based operations, the Service ID field must be sent along with the Route details of [Table 4.4](#), even though it is not shown explicitly in the table. The reason for doing this is that the customer must be given the choice of selecting an AC service or a Non-AC service or an Express service, etc.

**Table 4.5: Route Details Element of Route-Station Response**

Parent Element	Child Element / Presence	Field Name	Data Type	Maximum Size (Characters)	Description
Route Response <i>Routes only</i> → (Route-based requests)	Route# 1 Details / Conditional	Route ID	NS	5	Unique ID given to the Route by the PTO
		Route Name	AN	100	Name of route given to the Route by the PTO
	Route# 2 Details / Conditional	Route ID	NS	5	Unique ID given to the Route by the PTO
		Route Name	AN	100	Name of route given to the Route by the PTO
	...				
		Route ID	NS	5	Unique ID given to the Route by the PTO

## QR Ticketing System – Interface Specification

	Route# N Details / Conditional	Route Name	AN	100	Name of route given to the Route by the PTO
Route Response Stations only → (Non- route based requests)	Station# 1 Details / Conditional	Station ID	NS	5	Unique ID given to the Station by the PTO
		Station Name	AN	100	Name of the Station or Stop given by the PTO
	Station# 2 Details / Conditional	Station ID	NS	5	Unique ID given to the Station by the PTO
		Station Name	AN	100	Name of the Station or Stop given by the PTO
	...				
	Station# N Details / Conditional	Station ID	NS	5	Unique ID given to the Station by the PTO
		Station Name	AN	100	Name of the Station or Stop given by the PTO

In a single response, the maximum number of routes (No of Routes), N is set at 5000.

In a single response, the maximum number of stations (No of Stations), N is set at 5000.

If the number exceeds 5000 in either case, the response must be split into multiple responses. Collating such multiple responses in the correct sequence is implementation specific and is the responsibility of the application.

*In the form of a JSON we can now describe the JSON response sent by the TG.*

```

"Route_Response": {
  "Error_Code": "0",
  "Error_Text": "Success",
  "Operator_Id": "135",
  "Operation_Type": "1",
  "Product_Id": "1=Standard;3=Student;4=Daily;5=Monthly",
  "Validity": "480",
  "Duration": "180",
  "NoOfRoutes": "1000",
  "NoOfStations": "0",
  "Route_Details": {
    "Route_1": {
      "Route_Id": "1",
      "Route_Name": "StopName1 to StopNameA",
      "Service_Id": "3-Regular City Bus;4-AC City Bus"
    },
    "Route_2": {
      "Route_Id": "2",
      "Route_Name": "StopName2 to StopNameB",
      "Service_Id": "3-Regular City Bus;4-AC City Bus"
    },
    ...
    "Route_1000": {
      "Route_Id": "1000",
      "Route_Name": "StopName1000 to StopNameZ",
      "Service_Id": "4-AC City Bus"
    }
  }
  "Station_Details": {
  }
}

```

**Figure 6: Route Info Response JSON from TG for Operator ID 135**

After the APP receives the above response from the TG, it must facilitate the following:

- i. Allow user to select the Route from the list of Routes. Here for instance, the Operator ID 135 is a Bus operator and follows a route based approach (Operation type = 1). The APP displays the “Routes” only and the user can then select the route on which he/she wishes to travel. Here assume that Route ID = “8011” is selected.
- ii. When the user selects this Route and the ‘Station Request’ should be sent again – this time specifying Route ID = 8011.

Please note that the Station Request and Response structure is similar to the Route Request/Response scenario.

The request and response dataflow is shown here:

```
{
  "Station_Request": {
    "Requester_Id": "15",
    "Operator_Id": "135",
    "Mobile": "9201345678",
    "Email": "shriroop@web.in",
    "Username": "roop9201",
    "Route_Id": "8011"
  },
  "Station_Request_Signature": {
    "KdkjsahdJYuashb+YMMCq7MjJuVDiFGjw47ZkvfvjEkmw6wvVRNzlBn0BYSpKo
i8KrARwCLgk8 0vDdi8z2UOoMFpdeDusjSqt07kM73O6z2vY"
  }
}
```

**Figure 7: Fetch All Stations Request JSON for Route 8011 – Operator ID 135**

When the request lands at the TG, the TG will respond with all the Stations for this route (8011). In the form of a JSON the response is shown in [Figure 8](#).

```
"Station_Response": {
  "Error_Code": "0",
  "Error_Text": "Success",
  "Route_Id": "8011",
  "Operator_Id": "135",
  "NoOfRoutes": "0",
  "NoOfStations": "1000",
  "Route_Details": {
  },
  "Station_Details": {
    "Station_1": {
      "Station_Id": "1",
      "Station_Name": "StationName1"
    },
    "Station_2": {
      "Station_Id": "2",
      "Station_Name": "StationName2"
    },
    ...,
    "Station_N": {
      "Station_Id": "N",
      "Station_Name": "StationNameN"
    }
  }
}
```

**Figure 8: Route Response with Stations JSON from TG for Operator ID 135**

After the APP receives the above response from the TG, the Application facilitates the user to select the source and destination stations for the journey he/she wishes to embark. Also the user can specify the date and time of journey.

Here let us assume the user makes the following selection for Operator 135:

Operator ID = 135

Route ID = 8011

Source station = 24, Destination Station = 42

Date and time of Journey = 28/06/2020@18:10:00

As stated earlier, we are following an example of a multiple journeys in the same QR. Now we shall describe the requests and responses for Operator ID 10.

## **Fetch Stations for Operator ID 10:**

Now the User selects Operator ID 10 and sends a Route Request. The JSON request is similar to the request shown in Figure 7. The App does not have to know that this operator supports a non-route based operation type. It simply has to send the same first request i.e. the 'Route Request'. The TG knows the operation type and will send the Stations in the response.

```
{
  "Route_Request": {
    "Requester_Id": "15",
    "Operator_Id": "10",
    "Mobile": "9201345678",
    "Email": "shriroop@web.in",
    "Username": "roop9201",
  },
  "Route_Request_Signature": {
    "KdkjsahdJYuashb+YMMCq7MjJuVDiFGjw47ZkvfvjEkmw6wvVRNzlBn0
    BYSpKoi8KrARwCLgk80vDdi8z2UOoMFpdeDusjSqt07kM73O6z2vY"
  }
}
```

**Figure 9: Fetch All Stations Request JSON Operator ID 10**

Since for this request, the Route ID = 0 and Operation Type is 0, the TG responds with the Stations Information for this particular Operator.

The JSON response is similar to the response structure shown in [Figure 8](#).

```

"Route_Response": {
  "Error_Code": "0",
  "Error_Text": "Success"
  "Route_Id": "0",
  "Operator_Id": "10",
  "Operation_Type": "0",
  "Product_Id":
    "1=Standard;2=SeniorCitizen;3=Student;4=Daily;5=Monthly;6=Weekend",
  "Service_Id": "1-Regular;2-Express,21-FeederBus",
  "Validity": "480",
  "Duration": "180",
  "NoOfRoutes": "0",
  "NoOfStations": "1000",
  "Route_Details": {
  },
  "Stations_Details": {
    "Station_1": {
      "Station_Id": "1",
      "Station_Name": "StationName1",
    },
    "Station_2": {
      "Station_Id": "2",
      "Station_Name": "StationName2",
    },
    ...
    "Station_N": {
      "Station_Id": "N",
      "Station_Name": "StationNameN",
    }
  }
}

```

**Figure 10: Route Response with Stations JSON from TG for Operator ID 10**

After the APP receives the above response from the TG, the Application facilitates the user to select the source and destination stations for the journey he/she wishes to embark. Also the user can specify the date and time of journey.

Here let us assume the user makes the following selections for Operator 10:

a. Single Journey 1

Source station = 01, Destination Station = 12

Date and time of Journey = 15/06/2020@15:15:00

Product ID = 1 → Standard Product

Service ID = 1 → Normal Metro Service

b. Single Journey 2



## QR Ticketing System – Interface Specification

Source station = 03, Destination Station = 34

Date and time of Journey = 18/06/2020@11:30:00

Product ID = 1 → Standard Product

Service ID = 1 → Normal Metro Service

Similarly, for Operator 135:

a. Single Journey 1

Source station = 24, Destination Station = 42

Date and time of Journey = 28/06/2020@18:10:00

Product ID = 1 → Standard Product

Service ID = 4 → AC City Bus Service

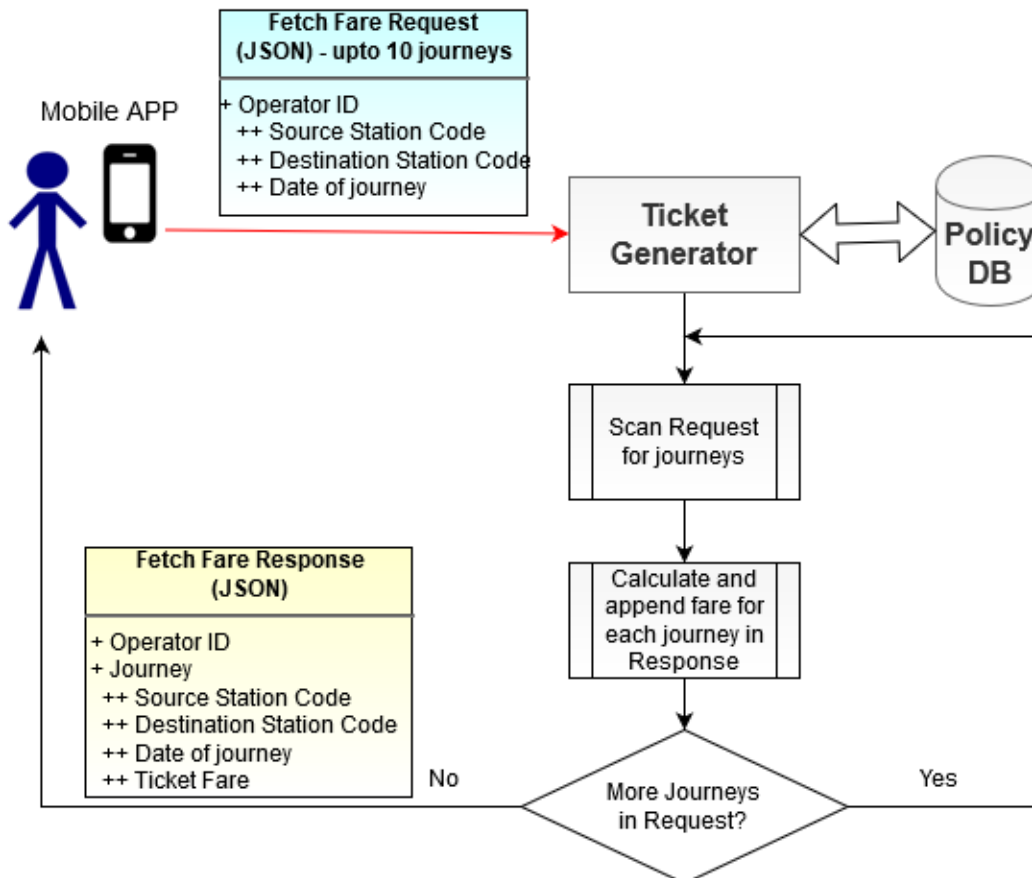
Now that the user has made all the choices this completes the selection of the multiple (3) journeys for the example. The application must now facilitate the User to send a 'Fetch Fare' request in order to allow the user to view the individual ticket fares.

We shall now describe the Fetch Fare Request and Response scenarios exchanged between the Application and the TG.

#### 4.3.1.2. Fetch Fare Details

*The Fetch Fare Request from APP to TG and Response from TG back to the APP is a HTTPS communication sequence.*

Figure 11 below describes the Fetch Fare Request and Response workflow.



**Figure 11: Fetch Fare Details Workflow**

The generic data elements of the Fetch Fare request and Response are shown here first. The data types and sizes of all the fields (except fare) have already been defined in [Table 4.3](#), [Table 4.4](#) and [Table 4.5](#) above.

**Table 4.6: Data elements of Fetch Fare Request**

Parent Element	Child Element /Presence	Child Element/Presence	Field /Presence	Data Type /Maximum Size (Chars)	Description
Fetch Fare Request - Requester ID	Operator-specific journeys - Operator ID /Mandatory (repeating field)	Journey /Mandatory (repeating field)	Source Station /Mandatory	NS /5 Range: 1-65535	Starting point of journey
			Destination Station /Mandatory	NS / 5 Range: 1-65535	Ending point of journey
			Activation Date /Mandatory	DS	Date and time of Journey
			Route ID /Optional	AN / 8	Unique ID given to the Route by the PTO. For PTOs that don't have route information this is not sent
			Group Size /Mandatory	NS / 3 Range: 0-255	No of persons travelling. Default = 1.
			Service ID* /Mandatory	NS / 3 Range: 0-255	Service being purchased. For example AC or Non-AC Bus or

\*Service ID: Please refer to note below [Table 4.4](#).

**Table 4.7: Data elements of Fetch Fare Response**

Parent Element	Child Element /Presence	Child Element /Presence	Field /Presence	Data Type /Maximum Size (Chars)	Description
Fetch Fare Response - Requester ID	Operator-specific journeys - Operator ID /Mandatory (repeating field)	Journey /Mandatory (repeating field)	Source Station /Mandatory	NS / 5	Starting point of journey
			Destination Station /Mandatory	NS / 5	Ending point of journey
			Activation Date /Mandatory	DS	Date and time of Journey
			Route ID / Optional	AN / 8	Unique ID given to the Route by the PTO
			Group Size /Mandatory	NS / 3	No of persons travelling. Default = 1.
			Service ID* /Mandatory	NS / 3	Service being purchased. For example AC or Non-AC Bus or
			Ticket fare /Mandatory	(NS.NS) / (6.2)	Price of this ticket. If it is a group ticket i.e. group size > 1, then this field will be equal to [Single Person Ticket Fare X Group Size]. If the fare has a paisa quantity, then only up to 2 decimal places must be sent like "10.45". If the paisa part is absent then only

					the whole number is sent like "23".
--	--	--	--	--	-------------------------------------

\*Service ID: Please refer to note below [Table 4.4](#).

Continuing with the example we have used thus far, we are requesting fare details for two single person single journey tickets for Operator ID 10 and one single journey ticket for Operator ID 135. The Fetch Fare request in JSON format is as follows:

```

"Fetch_Fare_Request": {
  "Requester_Id": "15",
  "Operator_Specific_JourneyA": {
    "Operator_Id": "10",
    "Journey1": {
      "Src_Stn": "01",
      "Dest_Stn": "12",
      "Journey_Date": "15/06/2020@15-15-00",
      "Grp_Size": "1",
      "Product_Id": "1",
      "Service_Id": "1"
    },
    "Journey2": {
      "Src_Stn": "03",
      "Dest_Stn": "34",
      "Journey_Date": "18/06/2020@11-30-00",
      "Grp_Size": "1",
      "Product_Id": "1",
      "Service_Id": "2"
    }
  },
  "Operator_Specific_JourneyB": {
    "Operator_Id": "135",
    "Journey1": {
      "Route_Id": "8011",
      "Src_Stn": "24",
      "Dest_Stn": "42",
      "Journey_Date": "28/06/2020@18-10-00",
      "Grp_Size": "1",
      "Product_Id": "1",
      "Service_Id": "4"
    }
  }
}

```

**Figure 12: Fetch Fare Request JSON by APP for Operator ID 10 & Operator ID 135**

In response the TG calculates and sends the Fare details for all the journeys.

```
"Fetch_Fare_Response": {
  "Requester_Id": "15",
  "Operator_Specific_JourneysA": {
    "Operator_Id": "10",
    "Journey1": {
      "Src_Stn": "01",
      "Dest_Stn": "12",
      "Journey_Date": "15/06/2020@15-15-00",
      "Grp_Size": "1",
      "Product_Id": "1",
      "Service_Id": "1",
      "Ticket_Fare": "80"
    },
    "Journey2": {
      "Src_Stn": "03",
      "Dest_Stn": "34",
      "Journey_Date": "18/06/2020@11-30-00",
      "Grp_Size": "1",
      "Product_Id": "1",
      "Service_Id": "2",
      "Ticket_Fare": "100"
    }
  },
  "Operator_Specific_JourneysB": {
    "Operator_Id": "135",
    "Journey1": {
      "Route_Id": "8011",
      "Src_Stn": "24",
      "Dest_Stn": "42",
      "Journey_Date": "28/06/2020@18-10-00",
      "Grp_Size": "1",
      "Product_Id": "1",
      "Service_Id": "4",
      "Ticket_Fare": "120"
    }
  }
}
```

**Figure 13: Fetch Fare Response JSON by TG for Selected Journeys**

Note that we have shown only 3 journeys – 2 for Operator ID = 10 and 1 for Operator ID = 135. As mentioned earlier, the maximum possible is 10 journeys. Now that the user has selected the journeys, he/she may proceed to make the payment for the tickets. The client application must facilitate the following:

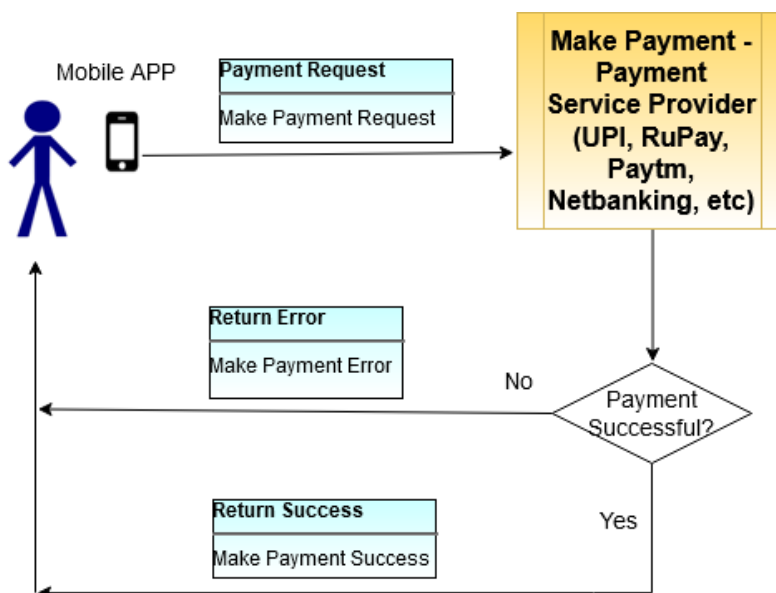
- On the 'Purchase Cart' page, the APP must display the date and time of journey, ticket fare, validity and duration against each journey.
- The APP must show the 'Total Fare' = 'sum of all individual journeys' clearly on the page.
- The APP must show a 'Make Payment' button on the page.
- Clicking the 'Make Payment' button will take the user to the Payment Service Provider interface.

The user only pays once and is not burdened to pay separately for multiple operator services.

The 'Make Payment' scenario is described in the next section.

### 4.3.1.3. User Makes Payment for QR Ticket

The APP sends the payment details to the Payment Service Provider, which then facilitates the direct deduction of funds from the user's bank account or wallet or any other mode of payment that the payment service provider facilitates.



**Figure 14: Make Payment and Purchase QR Ticket**

Assumptions and the 'Make Payment' process may be viewed as follows:

- The APP Provider (hence the application) has a secure interface with some bank/financial institution –this entity is the 'Payment Service Provider'.
- Assume that the date on which the user makes the purchase is 12/06/2020@18:47:00 hours
- The APP Provider has a 'make payment' facility that is invoked when the user proceeds to pay for the tickets.

The full payment information must be sent by the App Provider to the Payment Service Provider or PSP for short. Along with user payment information, the payment distribution fields must also be sent to the PSP.

Firstly, the payment transaction request that the APP Provider sends to the PSP can be summed up as follows:

**Table 4.8: Data elements of Make Payment Transaction Request**

Field Name	Field Description	Field Presence	Data Type /Length (chars)
Merchant ID	Unique ID created for every merchant at the time of registration. Merchant ID is given to the APP Provider by the Aggregator or the PSP. We assume this value is 'Merchant15'	M	AN / 20
Operating Mode	There are usually two operating modes — 'Domestic' and 'International'.  The values can be: <b>DOM</b> (for domestic) and <b>INTL</b> (for international).	M	AS / 10
Merchant Country	This is the country where the merchant is based in.	M	AS / 10
Merchant Currency	This is the transaction currency. E.g. in India, it should be 'Indian Rupees' or 'INR'	M	AS / 10
Transaction Amount	This is the transaction/order amount excluding the PSP commission and applicable taxes.  Values up to 2 decimal places are allowed.  We shall not account for PSP commission and applicable taxes here for simplicity.	M	(NS.NS) / (17.2)
Other Details	This is a 'free text' field the merchant can add any additional information relevant to the transaction.  The APP Provider must use this field to send other important details as follows which shall be useful for reconciliation. Every quantity must be separated with a semi-colon ';' <p><i>PSP_ID;</i> <i>APP_ID;</i> <i>TG_ID;</i> <i>Customer Mobile No;</i></p>	O	AN / 1000



# QR Ticketing System – Interface Specification

	<i>Operator IDs from the 5<sup>th</sup> item onwards (there maybe multiple operators like operator1;operator2, etc.)</i>		
Aggregator ID	This is the unique code for the Aggregator/PSP with whom the merchant integrates. For all practical purposes, the Aggregator and PSP are the same. For illustration we shall use the constant string value 'PSPEPAY' for the Aggregator ID field throughout this document.	M	AN / 15
Merchant Order Number	The APP Provider must create a unique 'order number' for each transaction.	M	AN / 100
Payment Mode	<p>The mode in which payment is made by selecting one of the payment options.</p> <p>The value can be any one of the following:</p> <ul style="list-style-type: none"> <li>● <b>NB</b> (Netbanking)</li> <li>● <b>CC</b> (Credit Card)</li> <li>● <b>DC</b> (Debit Card)</li> <li>● <b>IMPS</b> (Mobile Banking)</li> <li>● <b>WALLET</b></li> <li>● <b>PC</b> (Prepaid Cards)</li> <li>● <b>CASH</b></li> <li>● <b>POS</b></li> <li>● <b>NEFT/RTGS</b></li> <li>● <b>PAYPAL</b></li> <li>● <b>UPI</b></li> </ul>	O	AS / 10
Access Medium	This is the medium over which the payment page was accessed. The value to be passed in this field is: <b>ONLINE</b>	M	AS / 10

## QR Ticketing System – Interface Specification

Transaction Source	This indicates the source from where the transaction originated. The value to be passed in this field is: <b>ONLINE</b>	M	AS / 10
Success URL	The PSP may need to post the response 'SUCCESS' response on a HTTPS URL	O	AN / 200
Failure URL	The PSP may need to post the response 'FAILURE' response on a HTTPS URL	O	AN / 200

Secondly, the APP Provider must give the breakup of all the payments to the service providers – including its own MDR amount, the Amount for the TG, and the amounts for the PTOs. This can be described by [Table 4.9](#). Please note that this entire structure is a repeating structure.

**Table 4.9: Data elements of Multi-account Payment Request**

Field Name	Field Description	Field Presence	Data Type /Length (Chars)
Merchant Order Amount	This is the amount that needs to be settled in one of the multi accounts, mentioned in the 'Account Identifier' field.	M	NS / 40
Currency	Same as in <a href="#">Table 4.8</a>	M	AS / 10
Account Identifier	This is the unique identifier configured by the merchant at PSP to identify the Bank account in which the merchant order amount needs to be settled. Each of the multi accounts will have a unique identifier.	M	AN/100

For the example we have followed thus far, we have

- Total value of service, Total Amount → ₹300/-. Let the Payment mode chose by customer = IMPS

## QR Ticketing System – Interface Specification

- 2 tickets for Operator ID 10 worth ₹180/-
- 1 ticket for Operator ID 135 worth ₹120/-
- Assume that both APP Provider and the TG have 1 MDR Account each as shown below. Also assume the Service Fee charged by both accounts is 1% of the Total amount.
  - **APP\_Prov\_MDR\_Acc (No: 3344 last 4 digits)**
  - **TG\_MDR\_Acc (No: 1729 last 4 digits)**
- Also assume that both the operators – PTO ID 10 and PTO ID 135 also have 1 Corporate Account each as shown below.
  - **PTO\_10\_Corp\_Acc (No: 9187 last 4 digits)**
  - **PTO\_135\_Corp\_Acc (No: 6528 last 4 digits)**
- For sake for completeness, assume that the PSP institution ID or PSP\_ID = **78** and Aggregator ID = **'PSPEPAY'**.

*It is not mandatory that the APP Provider must send this information to the PSP using JSON in a REST API architecture over HTTPS. It depends entirely on the kind of interface the APP Provider has with the PSP. But here we shall assume that communication channel is one of REST API / JSON message exchange as has been used throughout the document. Nonetheless, it is a mandatory requirement that all Payment Information from APP Provider to the Payment Service Provider or vice versa to be exchanged over a secure communication channel like HTTPS and preferably with some encryption/decryption mechanism that can be agreed beforehand between the APP Provider (Merchant) and the PSP (Aggregator).*

```
{
  "Customer_Payment_Details": {
    "Merchant_Id": "Merchant15",
    "Merchant_Country": "INDIA",
    "Merchant_Currency": "INR",
    "Merchant_Order_No": "MO-1",
    "Operating_Mode": "DOM",
    "TXN_Amount": "300.00",
    "Aggregator_Id": "PSPEPAY",
    "Payment_Mode": "IMPS",
    "Access_Medium": "ONLINE",
    "TXN_Source": "ONLINE",
    "Other_Details": "78;15;23;9201345678;10;135"
  }
  "Multiaccount_Payment_Details": {
    "App_Provider_MDR_Account": {
      "Merchant_Order_Amount": "3.00"
      "Currency": "INR",
      "Account_Id": "xxxxxxxx3344"
    }
    "TG_MDR_Account": {
      "Merchant_Order_Amount": "3.00"
      "Currency": "INR",
      "Account_Id": "xxxxxxxx1729"
    }
    "PTO_10_Corp_Account": {
      "Merchant_Order_Amount": "176.40"
      "Currency": "INR",
      "Account_Id": "xxxxxxxx9187"
    }
    "PTO_135_Corp_Account": {
      "Merchant_Order_Amount": "117.6"
      "Currency": "INR",
      "Account_Id": "xxxxxxxx6528"
    }
  }
}
```

**Figure 15: APP Provider sends Payment Details to PSP**

In response to the above request, the PSP shall send the transaction details – status of transaction, transaction numbers, etc. as seen below in [Table 4.10](#).

**Table 4.10: Data elements of Multi-account Payment Response**

Field Name	Field Description	Field Type	Data Type /Length (Chars)
Merchant ID	Same value that is sent in the Transaction Request. This will be sent back in plain text.	M	NS / 10

## QR Ticketing System – Interface Specification

Merchant Order Number	This is the same 'Merchant Order Number' that was sent in the transaction request packet.	M	AS / 100
TXN_Ref_No	This is the unique Reference ID generated at PSP that is used to identify the transaction. This will be generated only when all the required fields are passed. This will not be generated in case of validation failures.	M	AS / 20
Transaction Status	This is the status of the payment transaction. It can have any of these values:  SUCCESS, FAILURE.	M	AS / 7
Error Message/Reason	This is a text description for the 'Error Code' if the Status is FAILURE.	M	AS / 200
Transaction Amount	Same as in <a href="#">Table 4.8</a>	M	(NS.NS) / (17.2)
Currency	Same as in <a href="#">Table 4.8</a>	M	AS / 10
Payment Mode	Same as in <a href="#">Table 4.8</a>	M	AS / 10
Other Details	<p>Same value that is sent in the Transaction Request.</p> <p>These fields were included in the request and sent back as-is:</p> <p>PSP_ID; APP_ID; TG_ID; Customer Mobile No;</p> <p><i>Note: Now populate the Operator/PTO Id's used in the QR Ticket.</i></p> <p>Operator_IDs (can be multiple operators e.g. Operator1;Operator2, etc.)</p>	M	AN / 1000

Gateway Code	The unique code for the Gateway/Bank, which is used to make the payment. We assume this is equal to 'BNK_GC_12'	M	AN / 10
Bank Reference Number – BNK_TRN	This is the unique reference number generated by the Bank/Payment Gateway for identifying the transaction.	M	AN / 40
Transaction Date	Date and time at which the transaction took place. Only present if the transaction is successful. Format must be agreed between APP Provider and PSP.	C	DS
Country	This is the same 'Merchant Country' that was sent in the transaction request packet.	M	AS / 10
CIN	<p>This is the 'Challan Identification Number (CIN)' generated at PSP for a government merchant.</p> <p>The format is as per agreement of PSP and APP Provider. In our example we will separate each quantity with a '_': MerchantID (7CHAR) '_ ' YYYYMMDD '_ ' running sequence (5 CHAR).</p> <p>Note: The CIN will be generated only for successful transactions. For a failed transaction, this value will be 'NA'</p>	M	AN / 10
Total Fee	This is the total Fee and applicable taxes on this transaction. They may be sent separated by some symbol like Carat (^) or Semi-colon (;). E.g. 200^2 or 200;2. This Fee must be borne by the Operator/PTO.	M	(NS;NS)/ (5;2)
RFU	This is reserved for future use and may not be the same data as that sent in the request. It should not be sent unless otherwise agreed between PSP and APP Provider.	O	AN (2500)

- Let us assume that the Transaction Reference Number received from the PSP for successful payment of ₹300 by the user is [ABCDRC202006126782341](#).
- Date and time of transaction is 12/06/2020@18:47:57

## QR Ticketing System – Interface Specification

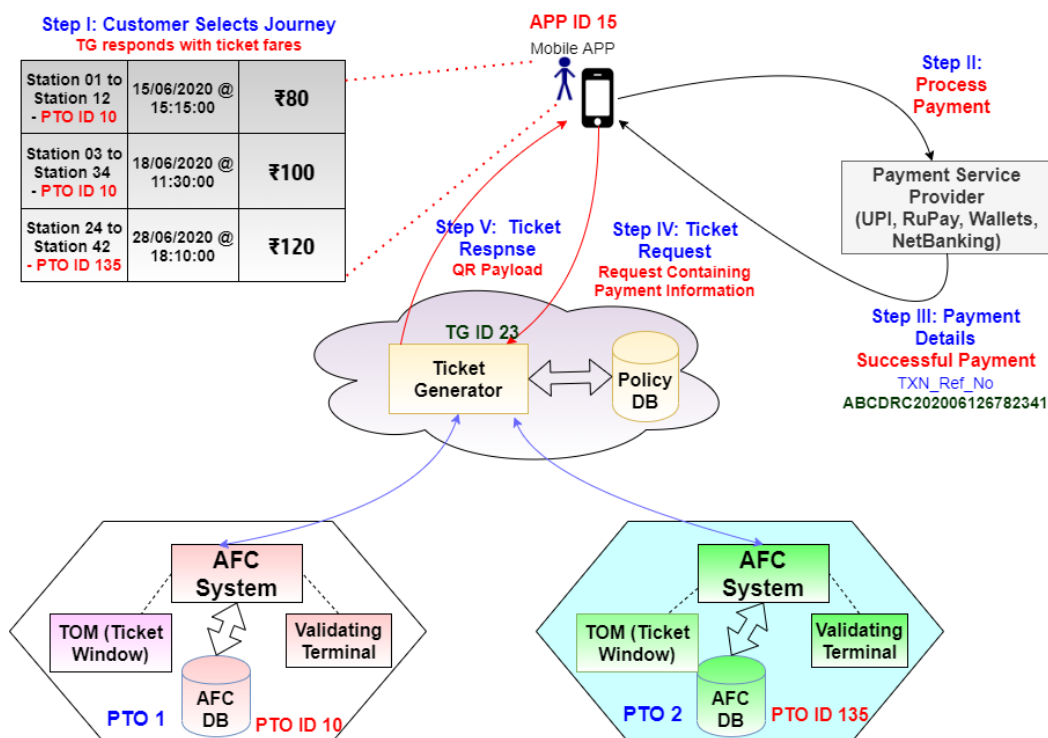
- Let us assume that the Challan Identification Number (for Govt. Orgs) received from the PSP is [MERCHANT\\_15\\_20200612\\_00312](#).
- We shall assume the [Total Fee for the PSP as 0](#) for simplicity.

Consistent with the rest of the document, we now show the response in JSON format here.

```
"PSP_Transaction_Response": {
  "Merchant_Id": "Merchant15",
  "Merchant_Order_No": "MO-1",
  "Merchant_Country": "INDIA",
  "Merchant_Currency": "INR",
  "TXN_Amount": "300.00",
  "Payment_Mode": "IMPS",
  "TXN_Ref_Num": "ABCDRC202006126782341",
  "TXN_Date": "12/06/2020@18-47-57",
  "Gateway_Code": "BNK_GC_12",
  "BNK_TRN": "5467908",
  "Total_Fee": "0",
  "CIN": "MERCHANT_15_20200612_00312",
  "Status": "SUCCESS",
  "Error_Text": "",
  "Other_Details": "78;15;23;9201345678;10;135"
}
```

**Figure 16: PSP sends Transaction Response to APP Provider**

The entire payment process workflow is shown pictorially in the [Figure 17](#) below. More details about PTO Payments and MIS reports are given in [Section 5.2](#).



**Figure 17: Customer Selects Journey and Makes Payment**

## 4.3.1.4. Ticket Request to TG

Following a successful payment, the APP automatically sends the Ticket Request to the Ticket Generator. In this section, we now show the Ticket Request format pertaining to the example we are following here.

*The Ticket Request from APP to the TG, and the Response from the TG back to APP is over the HTTPS protocol. Data exchanges may be implemented with the JSON/REST API model.*

- The APP Provider must digitally sign the ticket information by its private key before sending it to the TG.
- Sending the customer's mobile number is mandatory for APP bookings.
- An optional step is also provided here. The booking location coordinates of the Mobile phone is also sent here → "booking\_latitude": "28605", "booking\_longitude": "77299". The values are exponent  $10^3$ , i.e. 28605 is 28.605 and 77299 is 77.299.

We shall now define the data elements in a QR Ticket Request.



**Table 4.11: Data elements of QR Ticket Request**

Field Name	Field Description	Field Presence	Field Data type / Length
Requester ID	Same as in <a href="#">Table 4.3</a>	M	NS/ 5
Language	As defined in Appendix III of QR Code Dataset Specifications – Part I.	M	NS / 3
TXN_Ref_No	Same as in <a href="#">Table 4.10</a> .	M	AN / 22
Transaction Date	Date and time at which the transaction took place. Same as in <a href="#">Table 4.10</a> .	M	DS
PSP Specific Data	‘;’ separated values as defined in <a href="#">Table 4.10</a> → <i>PSP_ID;Merchant_ID;MerchantOrderNo;TransactionAmt;PaymentMode;BNK_TRN;CIN</i>	M	AN / 100
Booking Latitude	As defined in Table 5.4 of QR Code Dataset Specifications – Part I.	O	NS / 5
Booking Longitude	As defined in Table 5.4 of QR Code Dataset Specifications – Part I.	O	NS / 5
Mobile No	Same as in <a href="#">Table 4.3</a> . Only needed if booking is done online using the Webclient or the Mobile App.	C	NS / 10
Ticket Block	This may contain from 1 up to a maximum of 10 tickets as defined in Table 5.12 of QR Specifications – Part I.	M	Variable
Ticket Signature	The APP Provider takes a SHA-256 hash of the route request details. The APP Provider generates the Digital Signature of the QR Ticket Request with its Private Key.  It then creates the Base64 representation of the signature (produces 172 bytes).	M	B64S / 64

In the form of a JSON, the request is as follows.

```

"App_Ticket_Request": {
  "Ticket_Signature": {
    "Signature": "TkhsXgFafTiM3dFIJ+YMMCq7MjJuVDiFGJw47ZkvfvjEkmmw6wvVRNz1Bn0BYSpKoi8KrARwCLgk80vDdi8z2UOoMFpdeDusjSqt07kM73O6z2vY9iD7avHd3izA6vj8E11kNXK0S1"
  }
  "QR_Ticket_Request": {
    "Requester_ID": "15",
    "Language": "0",
    "TXN_Ref_No": "ABCDRC202006126782341",
    "TXN_Date": "12/06/2020@18-47-57",
    "PSP_Specific_Data": "78;Merchant15;MO-1;300;IMPS;5467908;Merchant15_20200612_00312",
    "Booking_Lat": "28605",
    "Booking_Lon": "77299",
    "Mobile": "9201345678",
    "TicketBlock": {
      "DynamicBlock": {
        "OperatorID1": {
          "OpID": "10",
          "NoOfTickets": "2",
          "TicketInfo": {
            "TicketInfo1": {
              "Grp_Size": "1",
              "Src_Stn": "01",
              "Dest_Stn": "12",
              "Activation_Date": "15/06/2020@15-15-00",
              "Ticket_Fare": "80",
              "Product_Id": "01",
              "Service_Id": "01",
              "Validity": "480",
              "Duration": "180"
            },
            "TicketInfo2": {
              "Grp_Size": "1",
              "Src_Stn": "03",
              "Dest_Stn": "34",
              "Activation_Date": "18/06/2020@11-30-00",
              "Ticket_Fare": "100",
              "Product_Id": "01",
              "Service_Id": "02",
              "Validity": "480",
              "Duration": "180"
            }
          }
        },
        "OperatorID2": {
          "OpID": "135",
          "NoOfTickets": "1",
          "TicketInfo": {
            "TicketInfo1": {
              "Grp_Size": "1",
              "Src_Stn": "24",
              "Dest_Stn": "42",
              "Activation_Date": "28/06/2020@18-10-00",
              "Ticket_Fare": "120",
              "Product_Id": "01",
              "Service_Id": "04",
              "Validity": "480",
              "Duration": "180",
              "Route_Id": "8011"
            }
          }
        }
      }
    }
  }
}

```

Figure 18: Ticket Request JSON from APP

The process of generating the Ticket Signature should be described here. In a formula notation, the Ticket Signature may be denoted as follows,

*Signature = Base64(Digital Sign<sub>Requester\_Private\_Key</sub>{SHA256<sub>HASH</sub>[QR\_Ticket\_Request]})*

## Step 1. Generate a SHA256 Hash of the Ticket request →

```
SHA256HASH["QR_Ticket_Request":{"Requester_ID":"15","Language":"0","TXN_Ref_No":"ABCDRC202006126782341","TXN_Date":"12/06/2020@18-47-57","PSP_Specific_Data":"78;Merchant15;MO-1;300;IMPS;5467908;Merchant15_20200612_00312","Booking_Lat":"28605","Booking_Lon":"77299","Mobile":"9201345678","TicketBlock":{"DynamicBlock":{"OperatorID1":{"OpID":"10","NoOfTickets":"2","TicketInfo":{"TicketInfo1":{"Grp_Size":"1","Src_Stn":"01","Dest_Stn":"12","Activation_Date":"15/06/2020@15-15-00","Validity":"480","Ticket_Fare":"80","Product_Id":"01","Service_Id":"01","Duration":"180"},"TicketInfo2":{"Grp_Size":"1","Src_Stn":"03","Dest_Stn":"34","Activation_Date":"18/06/2020@11-30-00","Validity":"480","Ticket_Fare":"100","Product_Id":"01","Service_Id":"02","Duration":"180"}}},"OperatorID2":{"OpID":"135","NoOfTickets":"1","TicketInfo":{"TicketInfo1":{"Grp_Size":"1","Src_Stn":"24","Dest_Stn":"42","Activation_Date":"28/06/2020@18-10-00","Validity":"480","Ticket_Fare":"120","Product_Id":"01","Service_Id":"04","Duration":"180","Route_Id":"8011"}}}}}]} → f9e9df0d65c19003d8d3aebf4854169f589093de1517ed26707823d5ee72dfc3
```

## Step 2. Digital Signing with Requester's (App Provider's) Private Key →

This creates gibberish: !±",gy\$P&O<i3Ů%|;/Ů#1à%.ÄÄ^q̄=1>DĚ{-ŽRhP\$íeīδ[ŮE+C°HKQ|×ÄŮERİĖ'ó-□~†BÅ4Xú34öG^ÿ/°¹±ðe...âIbb1BİÖt&u2t 𐄂𐄂M^f-Ÿ...H

## Step 3. Apply Base64 encoding to Digital Signature →

Sane data, Signature:

TkhsXgFafTiM3dFIJ+YMMCq7MjJuVDiFGjw47ZkvfjEkmw6wvVRNzIBn0BYSpKoi8KrARwCLGk80vDdi8z2UOoMFpdeDusjSqt07kM73O6z2vY9iD7avHd3izA6vj8EIlnXK0SI

### 4.3.1.5. QR Ticket Response from TG

After the TG receives the QR Ticket Request it has to generate the QR Ticket Payload with the elements of the QR Code Dataset as defined in QR Specifications – Part I. In other words, the payload is created with the following:

- QR Security, QR Dataset Version & QR Common Data together termed as *QR\_SVC*.
- QR Ticket Block termed as *QR\_Ticket\_Block*.
- Digital Signature of the above elements termed as *QR\_Signature*.

The TG must create the *QR\_SVC* and *QR\_Ticket\_Block* elements in JSON for it to be sent to the AFC of the PTOs. Then the TG encodes the *QR\_SVC* and *QR\_Ticket\_Block* in the SQDSR format and responds to the QR Ticket Request.

#### TG creates QR\_SVC:

We shall now proceed to create the JSON formats of *QR\_SVC*. For this, let us assume the ticket generation date and time is 12/06/2020@18:48:11. Also suppose the current running sequence number at the TG is 32446.

Notice that the TG fills up some elements in the QR Common Data – QR\_Gen\_Datetime, TXN\_Type (=65 for Purchase QR), Tkt\_SI\_No (as outlaid in the Terms and Definitions section of QR Specifications – Part I), Total Fare (sum total of all the ticket fares) and includes the TG\_ID as well. The JSON file representation of the QR\_SVC is as follows. The Security Scheme implemented is equal to 4 (SecureQR level-4 security – Refer Table 4.2 of QR Specifications Part II). The Dataset version is 1.0 which is actually equal to the value 4 (Refer QR Dataset Version Table 5.3 of QR Specifications Part I).

```
{
  "Security": {
    "Security_Scheme": "4"
  },
  "QR_Dataset_Version": {
    "Version_No": "1.0"
  },
  "QR_Common_Data": {
    "Requester_ID": "15",
    "Language": "0",
    "TG_ID": "23",
    "TXN_Type": "65",
    "TXN_Ref_No": "ABCDRC202006126782341",
    "QR_Gen_Datetime": "12/06/2020@18-48-11",
    "Tkt_SI_No": "12062020184811M0000032446",
    "Total_Fare": "300",
    "Booking_Lat": "28605",
    "Booking_Lon": "77299",
    "Mobile": "9201345678"
  }
}
```

**Figure 19: QR\_SVC JSON format**

## TG creates QR\_Ticket\_Block:

We shall now proceed to create the JSON QR\_Ticket\_Block. This is exactly the same Ticket Block from [Figure 18](#). The only difference is the addition of a new field Validator\_Info. This 1-byte field is dual purpose – a) the MSB 4-bit nibble tells the scanning capability of PTO and b) the remaining nibble tells whether a PTO uses encryption or not. Please refer to [Section 4.3.3.1](#) for a detailed explanation of this field. In this example, we just assume some hypothetical scenarios for Validator\_Info:

- Validator\_Info = 49 (decimal) or 0x31 Hex for Operator ID 10 implying it supports Camera & NFC scanning and uses encryption.
- Validator\_Info = 16(decimal) or 0x10 Hex for Operator ID 135 implying it only supports Camera scanning and does not use encryption.

```

"QR_TicketBlock": {
  "Dynamic_Block": {
    "OperatorID1": {
      "OpID": "10",
      "NoOfTickets": "2",
      "Validator_Info": "49",
      "TicketInfo": {
        "TicketInfo1": {
          "Grp_Size": "1",
          "Src_Stn": "01",
          "Dest_Stn": "12",
          "Activation_Date": "15/06/2020@15-15-00",
          "Ticket_Fare": "80",
          "Product_Id": "01",
          "Service_Id": "01",
          "Validity": "480",
          "Duration": "180"
        },
        "TicketInfo2": {
          "Grp_Size": "1",
          "Src_Stn": "03",
          "Dest_Stn": "34",
          "Activation_Date": "18/06/2020@11-30-00",
          "Ticket_Fare": "100",
          "Product_Id": "01",
          "Service_Id": "02",
          "Validity": "480",
          "Duration": "180"
        }
      }
    },
    "OperatorID2": {
      "OpID": "135",
      "NoOfTickets": "1",
      "Validator_Info": "16",
      "TicketInfo": {
        "TicketInfo1": {
          "Grp_Size": "1",
          "Src_Stn": "24",
          "Dest_Stn": "42",
          "Activation_Date": "28/06/2020@18-10-00",
          "Ticket_Fare": "120",
          "Product_Id": "01",
          "Service_Id": "04",
          "Validity": "480",
          "Duration": "180",
          "Route_Id": "8011"
        }
      }
    }
  }
}

```

Figure 20: QR Ticket Block JSON

## QR Ticketing System – Interface Specification

From this point forward, the processes of how the TG constructs the response and sends it back to the APP is described in details in the QR Specifications – Part II, Sections 5.1.2 and 5.1.3. Finally, the JSON format of the response is as follows.

### TG Sends QR Payload to Application:

The table below represents the data structure elements that the TG sends to the App. This data structure is also referred to as ‘QR Ciphertext’ and is defined in Table 5.13 of Part II – QR Specifications.

**Table 4.12: Data elements of QR Ciphertext**

QR Ciphertext				
Payload Length	Element		Size (chars)	Description
459 to 1946	QR Payload	QR Signature	172	Content signed by Ticket generator
		QR Data	198 – 1685	Full QR information – security, dataset version, common data and ticket block
	SHA256 or higher hash of QR Payload		64	This is for data integrity check. The SHA256 (or higher) hash of “Requester ID#TG ID# QR_Signature#QR_SVC#QR_Tkt_Block #QR_Tkt_SI_No” is sent along. Here we are assuming a SHA256 Hash.
	QR Serial No		25	QR Ticket Serial No in representational form or display form.

In the JSON format the QR Ciphertext will look as follows:

```
{
  "QR_Payload": {
    "QR_Signature": "1+jQB0tiqSA3mcAuic0nW6R17/o7wOwAmUqszcm57lzTzyK7FN3ajqlxJVE5QpRTdIaKWJmra3ZF+pBYxybgfuEcqEZT2I3/0ixDPCD/h+iNmj83i+/sFfigIGUvwqHMboBwWUjDrrC27AigpK18qSKUNNo1a0MmeqnxUpp/Op0=",
    "QR_SVC": "{4}{4}{0|17|41|5EE3809380007EBE|5EE38093|ABCDRC202006126782341|F|7530|6FBD|12DF3|9201345678}",
    "QR_Ticket_Block": "{(<A|2|31|U2FsdGVkX18CTq23LFaQH4Oeca8aardPFHSioqpWbnPseksRXJKtvNKSYjX0ud1ALZm9iM0dOScw9bGLbGp0PV+d+gedSQpr0+RJDGHBZEs=><87|1|10|[%1|18|2A|5EF88FA0|1E0|78|1|9|4650|8011%]>)}"
  },
  "QR_SHA256": "f774b0628e0f3cad72dfb1ef8f2b835d6ca9804daee3abb9df675cd405de263e",
  "QR_Tkt_SI_No": "12062020184811M0000032446"
}
```

Figure 21: Ticket Response JSON from TG

#### 4.3.1.6. APP Renders QR Ticket image

In order to render the QR image, the Mobile APP first checks the SHA256 Hash for integrity and shall then append the Dynamic Data to the payload received from the TG. The process of rendering the QR Image can be described in 3 broad steps as shown below.

##### I. Create the Dynamic Data Update Datetime element

- The local time at the mobile is 12/06/2020 18:48:36 which is 5EE36CCB in Hexadecimal.
- Latitude and Longitude are not implemented and is given as 0. So the plain Dynamic Data is {5EE36CCB|0|0|0} in SQDSR format.

##### II. Create the QR Payload String to encode

- Using the values of the elements – QR\_Signature, QR\_SVC and QR\_Tkt\_Block from the Ticket\_Response in Figure 21 and appending the Dynamic Data, the APP renders the QR image. Each element is separated by the ‘#’ character as follows.

```
1+jQB0tiqSA3mcAuic0nW6R17/o7wOwAmUqszcm57lzTzyK7FN3ajqlxJVE5QpRTdIaKWJmra3ZF+pBYxybgfuEcqEZT2I3/0ixDPCD/h+iNmj83i+/sFfigIGUvwqHMboBwWUjDrrC27AigpK18qSKUNNo1a0MmeqnxUpp/Op0=#{4}{4}{0|17|41|5EE3809380007EBE|5EE38093|ABCDRC202006126782341|F|7530|6FBD|12DF3|9201345678}#{(<A|2|31|U2FsdGVkX18CTq23LFaQH4Oeca8aardPFHSioqpWbnPseksRXJKtvNKSYjX0ud1ALZm9iM0dOScw9bGLbGp0PV+d+gedSQpr0+RJDGHBZEs=><87|1|10|[%1|18|2A|5EF88FA0|1E0|78|1|1|4650|8011%]>)}#{5EE36CCB|0|0|0}
```

##### III. Render the QR Image using specific properties

- In this example, for encoding the QR, the APP uses parameters – ‘dots per module’ = 4, ‘mode’ = binary and ‘ECC’ = M. The QR rendered is Version 16 or a ‘81 x 81 QR’ module.
- The APP also retrieves the QR\_Tkt\_SI\_No from the response – 12062020184811M0000032446
- The QR serial number along with name and mobile number of person can be displayed on the mobile screen along with the QR image.

The three broad steps are now shown in the diagram below, in Figure 22.

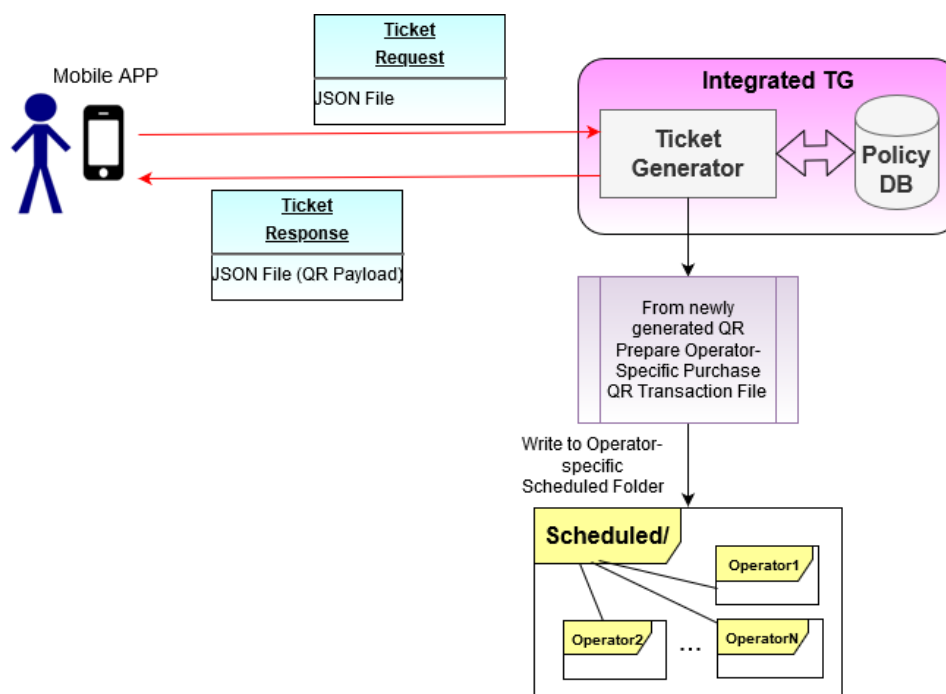


The AFC is a PTO-specific system so this interface is essentially also an interface between the AFC and PTO. In this section we describe the interface related to the ‘Purchase QR’ transactions only. The interface for exchanging financial data is described in [Section 5.2](#).

After the Ticket generator generates QR ticket response and sends it back to App / TOM, it sends the ticket information to the respective AFC System of all the operators that are involved in the QR ticket. The well-known IP addresses of the AFC System are present in the Policy DB of the operator.

- TG segregates the QR Ticket into operator-specific information and writes it to separate files
- TG sends files to respective PTOs at an interval of at least 2 minutes





**Figure 23: TG prepares Purchased QR for Scheduler**

The table below shows the Message format that is required to be followed in order to exchange information between the TG and AFC.

**Table 4.13: Structure for Message Exchange between TG & AFC**

Element	Size (bytes)	Data Type	Size (bytes)
Hash Token	64	AN	64-byte Hash of the Message Payload. This is for integrity checking. The Hashing algorithm to be used shall be provided by the Operator as a Policy DB parameter. Refer <a href="#">Table 5.1</a> . SHA256 Hash produces a 64-byte Hex string and is considered to be a good hashing algorithm, but operators are free to choose their own.
Message Payload	Variable*	AN	The user data that contains the main QR transaction information and shall differ from message to message.

The Message Payload again consists of 2 parts – the ‘Message Key’ and the ‘Payload Data’. The ‘Payload Data’ may be encrypted as well. Naturally if the contents are encrypted, it must be converted to the Base64 format. In every subsequent section when this structure is used, we shall provide the specific definitions of every payload. The structure shown in [Table 4.13](#) can then be put together as a JSON file

and sent using any communication mechanism like TCP, FTP, HTTP, etc. It may also be sent over some secure tunnel like VPN if agreed between the participating entities.

*\*In general, protocols like HTTP and TCP does not limit the size of the payload that is sent or received and the application can take care of the details. But for the technically inclined, there is a limit imposed by protocols as shown in the table below.*

**Table 4.14: Length of Payload**

Protocol	Size Limit	Description
HTTP or HTTPS	4 GB	Most Http(s) clients specify the limit of payload length as 4 GB for a single POST and GET request/response.
TCP or TLS	No limit	It is implementation specific. It usually allows up to 64 Kilobytes in one go, but the responsibility of breaking up large messages into segments lies with the application.

We shall now describe the process of creating the operator-specific transaction files that are sent to the respective AFC systems of PTO. The transaction file shall be a JSON object file as shown in the diagram below. The file naming convention used here is **qr\_DDMMYYYYhhmmss\_TGID\_PTID.JSON**.

```
{
  "Hash_Token": {
    "Hash_String": "Default is SHA256 hash string"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "Defined by the QR Ticketing System"
    },
    "Payload_Data": {
      "Filename": "qr_DDMMYYYYhhmmss_TGID_OpID",
      "No_Of_Records": "N",
      "Txn_Records": {
        "Record-1": "<QR_Dataset_Record>",
        "Record-2": "<QR_Dataset_Record>",
        ...
        "Record-N": "<QR_Dataset_Record>"
      }
    }
  }
}
```

**Figure 24: Operator-specific Transaction File Format (Generic)**

Each record of the transaction file is the entire QR data present but having only the operator's own ticket information. This is not the same as the ticket block that was generated and sent to App. The TG does not send the transaction information as soon as it generates the QR. It runs a scheduler that does the job in a configurable interval.

### 4.3.2.1. Purchase QR Transaction Request from TG to AFC

After the TG has sent the QR Ciphertext to the APP, the next step is to send the QR Purchase information for Operator ID = 10 and Operator ID = 135 to their respective AFCs. This is required so that the Validating terminals at the PTO premises are prepared beforehand to scan the purchased QRs.

Let us assume that the date and time portion of the transaction file names is **12062020185001**. The file name for Operator 10 is **qr\_12062020185001\_23\_10.JSON** and for Operator 135 is **qr\_12062020185001\_23\_135.JSON**. From the data shown in Figure 19 and Figure 20, the TG constructs each QR Purchase File as follows.

The JSON file shown in Figure 25 corresponds to the records for Operator ID 10 and the JSON file shown in Figure 26 corresponds to the records for Operator ID 135. The QR\_SVC part needs to be present in all the records. The total fare in Common Data has to be adjusted to reflect amount for the operator only.

Also note that during the process of creating the QR Payload for Ticket Response, the transaction data is already there at the TG – in Figure 19 and Figure 20. There is no duplication of effort here. The only additional step that needs to be taken is segregation of operator-specific data.

## QR Ticketing System – Interface Specification

Purchase QR file for Operator ID 10:

```
{
  "Filename": "qr_12062020185001_23_10",
  "No_Of_Records": "1",
  "Record_1": {
    "Security": {
      "Security_Scheme": "4"
    },
    "QR_Dataset_Version": {
      "Version_No": "1.0"
    },
    "QR_Common_Data": {
      "Requester_ID": "15",
      "Language": "0",
      "TG_ID": "23",
      "TXN_Type": "65",
      "TXN_Ref_No": "ABCDRC202006126782341",
      "QR_Gen_Datetime": "12/06/2020@18-48-11",
      "Tkt_Sl_No": "12062020184811M00000032446",
      "Total_Fare": "180",
      "Booking_Lat": "28605",
      "Booking_Lon": "77299",
      "Mobile": "9201345678"
    },
    "QR_TicketBlock": {
      "OpID": "10",
      "NoOfTickets": "2",
      "Validator_Info": "49"
    },
    "TicketInfo1": {
      "Grp_Size": "1",
      "Src_Stn": "01",
      "Dest_Stn": "12",
      "Activation_Date": "15/06/2020@15-15-00",
      "Ticket_Fare": "80",
      "Product_Id": "01",
      "Service_Id": "01",
      "Validity": "480",
      "Duration": "180"
    },
    "TicketInfo2": {
      "Grp_Size": "1",
      "Src_Stn": "03",
      "Dest_Stn": "34",
      "Activation_Date": "18/06/2020@11-30-00",
      "Ticket_Fare": "100",
      "Product_Id": "01",
      "Service_Id": "02",
      "Validity": "480",
      "Duration": "180"
    }
  }
}
```

Figure 25: Purchase QR Transaction File qr\_12062020185001\_23\_10.JSON

Purchase QR file for Operator ID 135:

```
{
  "Filename": "qr_12062020185001_23_135",
  "No_Of_Records": "1",
  "Record_1": {
    "Security": {
      "Security_Scheme": "4"
    },
    "QR_Dataset_Version": {
      "Version_No": "1.0"
    },
    "QR_Common_Data": {
      "Requester_ID": "15",
      "Language": "0",
      "TG_ID": "23",
      "TXN_Type": "65",
      "TXN_Ref_No": "ABCDRC202006126782341",
      "QR_Gen_Datetime": "12/06/2020@18-48-11",
      "Tkt_Sl_No": "12062020184811M00000032446",
      "Total_Fare": "120",
      "Booking_Lat": "28605",
      "Booking_Lon": "77299",
      "Mobile": "9201345678"
    },
    "QR_Ticket_Block": {
      "OpID": "135",
      "NoOfTickets": "1",
      "Validator_Info": "16",
      "TicketInfo1": {
        "Grp_Size": "1",
        "Src_Stn": "24",
        "Dest_Stn": "42",
        "Activation_Date": "28/06/2020@18-10-00",
        "Ticket_Fare": "120",
        "Product_Id": "01",
        "Service_Id": "04",
        "Validity": "480",
        "Duration": "180",
        "Route_Id": "8011"
      }
    }
  }
}
```

**Figure 26: Purchase QR Transaction File qr\_12062020171512\_23\_135.JSON**

Observe that in the figures – Figure 25 and Figure 26 – the field ‘Total\_Fare’ in ‘QR\_Common\_Data’ now has the updated values of the total amount of all the individual journeys i.e. sum of ‘Ticket\_Fare’.

#### **Scheduling tasks to send transactions to AFC System:**

Now when the scheduler kicks-off we have to send the transactions.

In this section, the exchange of information between the TG and AFC System is described. QR Specifications – Part I Table 4.1 describes 6 different types of QR Payloads that may be exchanged between AFC and the QR Ticketing System viz. – *Purchase QR, Transaction QR, Verification QR, Error QR, Duplicate QR and Update QR* with values ranging from 65 through 70.

Now let us examine the Message Payload for Purchase QR in details.

QR Message Payload Element	Size (bytes)	Data Type	Description	
Message Key	2	NS	01 – QR_TICKET_PURCHASE_REQUEST Key ID = 01	
Payload Data	Variable	AN	Filename	
			Number of Records	
			Record 1	Contents like in <a href="#">Figure 25</a> and <a href="#">Figure 26</a> .
			Record 2	Contents like in <a href="#">Figure 25</a> and <a href="#">Figure 26</a> .
			...	...
			Record N	Contents like in <a href="#">Figure 25</a> and <a href="#">Figure 26</a> .

Following the message payload construction as shown in the table above, we can now exchange the Purchase QR information to the respective AFCs as follows.

### Purchase QR Request Message:

Table below describes the Purchase QR Request Message structure from TG to PTO ID 10 and PTO ID 135.

**Table 4.16: Purchase QR Request Message – TG to AFC**

Operator ID	QR Message Format – TG & AFC communication		
	Hash Token – 64 bytes SHA-256(Message Payload)	Message Payload	
		Message Key – 2 bytes	Payload Data – Variable size
10	5a1393795b59f5c30b17df2f1b8849378aa54d44ee500a89a658c24fa552abd1	01	Contents of JSON file <a href="#">Figure 25</a>
135	5eff7ad24fde0b97b6e64a5ed5f8479936d45b0a770d12a987cbe0ebaf6799e8	01	Contents of JSON file <a href="#">Figure 26</a>

The whole message is then packed in JSON files in the structure described in [Figure 24](#) and sent to the AFC. Requests are shown below in [Figure 27](#) and Figure 28.

```

"QR_Purchase_Request": {
  "Hash_Token": {
    "Hash_Value":
      "5a1393795b59f5c30b17df2f1b8849378aa54d44ee500a89a658c24fa552abd1"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_ID": "01"
    },
    "Payload_Data": {
      "Filename": "qr_12062020185001_23_10",
      "No_Of_Records": "1",
      "Record_1": {
        "Security": {
          "Security_Scheme": "4"
        },
        "QR_Dataset_Version": {
          "Version_No": "1.0"
        },
        "QR_Common_Data": {
          "Requester_ID": "15",
          "Language": "0",
          "TG_ID": "23",
          "TXN_Type": "65",
          "TXN_Ref_No": "ABCDRC202006126782341",
          "QR_Gen_Datetime": "12/06/2020@18-48-11",
          "Tkt_Sl_No": "12062020184811M00000032446",
          "Total_Fare": "180",
          "Booking_Lat": "28605",
          "Booking_Lon": "77299",
          "Mobile": "9201345678"
        },
        "QR_TicketBlock": {
          "OpID": "10",
          "NoOfTickets": "2",
          "Validator_Info": "49"
          "TicketInfo1": {
            "Grp_Size": "1",
            "Src_Stn": "01",
            "Dest_Stn": "12",
            "Activation_Date": "15/06/2020@15-15-00",
            "Ticket_Fare": "80",
            "Product_Id": "01",
            "Service_Id": "01",
            "Validity": "480",
            "Duration": "180"
          },
          "TicketInfo2": {
            "Grp_Size": "1",
            "Src_Stn": "03",
            "Dest_Stn": "34",
            "Activation_Date": "18/06/2020@11-30-00",
            "Ticket_Fare": "100",
            "Product_Id": "01",
            "Service_Id": "02",
            "Validity": "480",
            "Duration": "180"
          }
        }
      }
    }
  }
}

```

Figure 27: QR Purchase Transaction Request Operator ID 10



```

"QR_Purchase_Request": {
  "Hash_Token": {
    "Hash_Value":
      "5eff7ad24fde0b97b6e64a5ed5f8479936d45b0a770d12a987cbe0ebaf6799e8"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "01"
    },
    "Payload_Data": {
      "Filename": "qr_12062020185001_23_135",
      "No_Of_Records": "1",
      "Record_1": {
        "Security": {
          "Security_Scheme": "4"
        },
        "QR_Dataset_Version": {
          "Version_No": "1.0"
        },
        "QR_Common_Data": {
          "Requester_ID": "15",
          "Language": "0",
          "TG_ID": "23",
          "TXN_Type": "65",
          "TXN_Ref_No": "ABCDRC202006126782341",
          "QR_Gen_Datetime": "12/06/2020@18-48-11",
          "Tkt_Sl_No": "12062020184811M0000032446",
          "Total_Fare": "120",
          "Booking_Lat": "28605",
          "Booking_Lon": "77299",
          "Mobile": "9201345678"
        },
        "QR_Ticket_Block": {
          "OpID": "135",
          "NoOfTickets": "1",
          "Validator_Info": "16",
          "TicketInfo1": {
            "Grp_Size": "1",
            "Src_Stn": "24",
            "Dest_Stn": "42",
            "Activation_Date": "28/06/2020@18-10-00",
            "Ticket_Fare": "120",
            "Product_Id": "01",
            "Service_Id": "04",
            "Validity": "480",
            "Duration": "180",
            "Route_Id": "8011"
          }
        }
      }
    }
  }
}

```

**Figure 28: QR Purchase Transaction Request Operator ID 135**

When the scheduler program at the TG sends the transaction files to the respective AFC Systems, the AFC system will acknowledge every request that it receives so the sending program at the TG must wait for these responses or ACKs.

#### 4.3.2.2. Purchase QR Transaction Response from AFC

##### Purchase QR Response Message Format:

The format of the response message is exactly the same as the request as shown in [Table 4.13](#). However, the response payload differs in that it is an acknowledgement of the request. The payload structure is described in the table below.

**Table 4.17: Structure for Purchase QR Response Payload – AFC to TG**

QR Message Payload Element	Size (bytes)	Data Type	Description
Message Key	2	NS	02 – QR_TKT_PURCHASE_RESPONSE Key ID = 02
Payload Data	Variable	AN	<p><b>Filename</b> = Name of the file sent as a field inside the Purchase QR Request.</p> <p><b>Ack</b> = String “<b>ACK-N</b>”, where N is value of the No_Of_Records field in the request.</p> <p><b>TG_ID</b> = ID of the TG (in all examples in this document we have used the value 23)</p> <p><b>Operator ID</b> = ID of the Operator (in all examples in this document we have used two values for Operator/PTO ID, 10 and 135)</p> <p><b>ErrorMsg</b> = Can be either the string “SUCCESS” by default for no error or some other elaborate message chosen by the operator. When the TG receives an ACK that has an ErrorMsg other than SUCCESS, it should try to resend the message again.</p>

### Purchase QR Response (General AFC Systems):

Table below describes the Purchase QR Response Message structure from PTO ID 10 and PTO ID 135.

**Table 4.18: Purchase QR Response Message AFC to TG**

Operator ID	QR Message Format – TG & AFC communication		
	Hash Token – 64 bytes SHA-256 of Message Payload	Message Payload	
		Message Key – 2 bytes	Payload Data – Variable size
10	a114a3bce1596e271201db47 4613b3eb11bd2a51d2a9f88d5 1dac160e1a19087	02	Element 'Payload Data' in <a href="#">Figure 29</a> .
135	08f150b0ba9d5bce50225e241 0bde4383e489ac6ae15de9f30 b7e86096b7e0cf	02	Element 'Payload Data' in <a href="#">Figure 30</a> .

In terms of JSON files the acknowledgements are as follows.

```

"QR_Purchase_Response": {
  "Hash_Token": {
    "Hash_Value":
      "a114a3bce1596e271201db474613b3eb11bd2a51d2a9f88d51dac160e1
      a19087"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "02"
    },
    "Payload_Data": {
      "Filename": "qr_12062020185001_23_10",
      "Ack": "ACK-1",
      "TG_ID": "23",
      "OpID": "10",
      "ErrorMsg": "SUCCESS"
    }
  }
}

```

**Figure 29: QR Purchase Transaction Response Operator ID 10**

```

"QR_Purchase_Response": {
  "Hash_Token": {
    "Hash_Value":
      "08f150b0ba9d5bce50225e2410bde4383e489ac6ae15de9f30b7e86096
      b7e0cf"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "02"
    },
    "Payload_Data": {
      "Filename": "qr_12062020185001_23_135",
      "Ack": "ACK-1",
      "TG_ID": "23",
      "OpID": "135",
      "ErrorMsg": "SUCCESS"
    }
  }
}

```

**Figure 30: QR Purchase Transaction Response Operator ID 135**

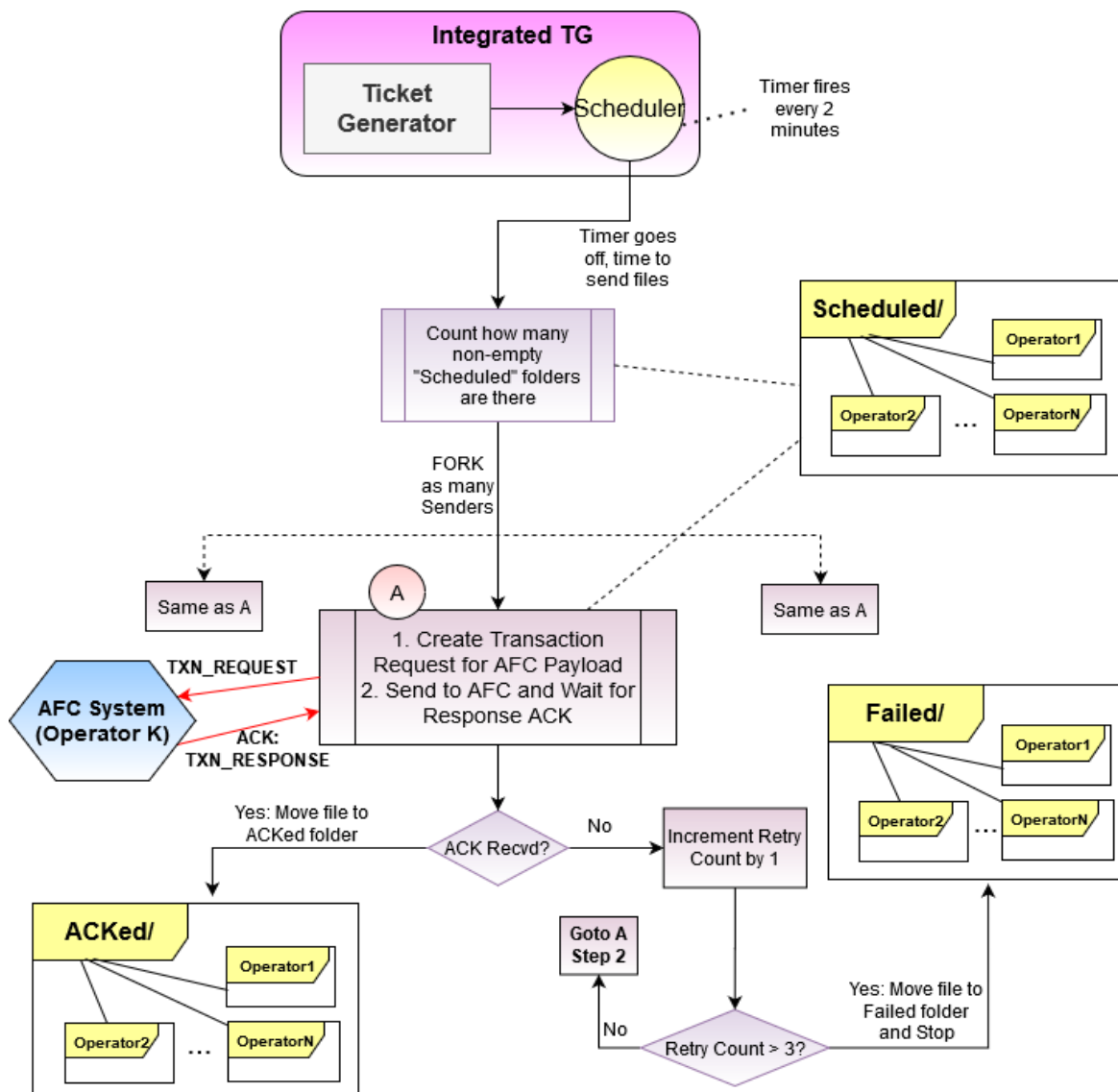
When the TG receives the ACK Responses as shown in the figures above, it must take the following steps

- Compares the hashes to ensure that the file was received intact with full integrity
- Examine the ErrorMsg element to check if the string “SUCCESS” has been received.
- Assuming the previous steps were satisfied, it removes the respective JSON files from the ‘Scheduled’ folder to the ‘Acknowledged’ folder.

Note: - It is also desirable to have the TG to wait for the response from AFC for a specified amount of time. In the event of the TG not able to send a file to the AFC, or not receiving an acknowledgement or the AFC sending an ErrorMsg other than “SUCCESS”; the sender program must retry sending it again. The number of retries should be equal to 3. After 3 retries, the TG should remove the JSON files from the ‘Scheduled’ folder to the ‘Failed’ folder.

## QR Ticketing System – Interface Specification

Here below in the diagram we depict the above mentioned steps in the form of a flow chart. This flow chart is a generic form of a sender/receiver algorithm that every TG must implement in some way in to exchange information with trusted entities like the PTO or Operator. This flow chart is also referred to describe the process of exchanging financial information between the TG and PTO in [Section 5.2](#).



**Figure 31: Send QR Info to AFC System & Receive Response**

*Note: - Files in the 'Acknowledged' folder can serve as the legitimate history data of all QRs generated by the TG. The data is also easily traceable back to the user that purchased the QR. In this example we have shown an implementation wherein the history is maintained in the form of a flat file system. Individual TGs and Operators are of course free to choose their own mechanism to store this history information. Some may choose to store the history in database tables instead of a flat file system.*

**Reconciliation of "Failed" transactions (offline process)** – After the TG sends the transaction files to the respective PTO's AFC system, it waits for an acknowledgement. If it does not receive an ACK within the configured time period it retries again another 2 times, or 3 times in total. If it still does not receive the ACK after that, it writes the transaction into the "Failed" folder and stops. It is recommended that the TG should also employ some monitoring programs like a "watchdog" on the 'Failed' folder. Whenever the 'watchdog' notices a file on this folder, it can notify the System Administrator who can then take action to resend those transactions to the AFC through some other mechanism like Email.

#### **4.3.2.3. Exchange of QR Information with Typical AFC Systems**

Information generated in the QR Ticketing System viz. Tickets purchased or Policy related information, etc. have to be exchanged with the AFC Systems. For the AFC System, the main entity of the QR System – the TG – is actually a third-party system. There may be many existing AFC systems which are already following a separate format to exchange information with other third-party systems. Since the QR Ticketing System is a relatively newer system, it is imperative for this system to be flexible so that it can be easily adapted into existing AFC Systems with minimal effort. In such cases, a simple approach can be used to map the QR Ticketing System Message Payload format described in this section, into the target AFC System's expected format. Since information exchanged between any two remote systems is usually freeform text piggybacked over a protocol like HTTP or TCP/IP, performing such mappings is actually not a big issue. Illustrations of adapting the QR Ticketing System to exchange messages with such systems (specifically NCMC-compliant AFC Systems) are given in Section 4.1.4 of the QR Ticketing System Implementation Guidelines.

### 4.3.3. QR Payload Media and Validator Interface

This is the direct physical interface of the media containing the QR image – a piece of Paper or some application on a smartphone. We assume that the scanner program and the validator programs shall be hosted on the same machine i.e. the validating terminal. However, there may be situations that after a QR is scanned the program calls the Operator’s online validator somewhere on the Cloud. Here we assume the scanner and validators are local.

#### 4.3.3.1. QR Payload Transfer to Validator

In most operator premises, the QR scanners would be present along with other media scanning hardware like NFC for tokens, etc. Ideally, the QR scanner application must be capable of displaying the QR image on the monitor so that the user can direct it within the specified boundary.

However, displaying QR images is a cpu-intensive process and it may not be possible in transit environments. Alternate arrangement like emitting focused light, marking area (rectangular box) to place QR etc. may be recompensed. In this section we shall consider scanning the QR image using a camera scanner. For a detailed analysis of using other scanning devices please refer to [Appendix – II: Usage of Different Scanning Media](#).

#### 4.3.3.2. QR Watchdog in Validating Terminal

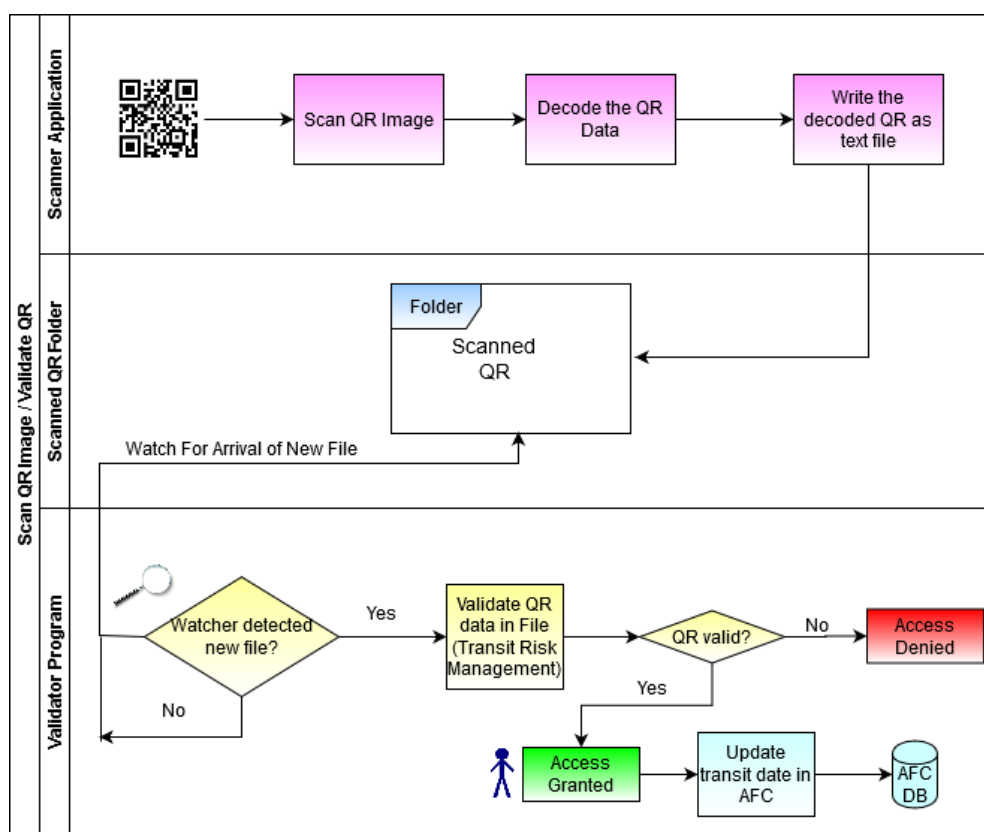
A QR image appearing before a camera scanner can be considered as an event detected in real-time. Since these events will take place randomly in the span of time, there needs to be a scheme put in place that can detect the event in real time. “Watchdogs” are a very simple and effective mechanism to detect events. In our case, the appearance of a new scanned QR file can signal the detection of such events. The “watchdog” program, as shown in [Figure 32](#), constantly monitors the ‘Scanned QR’ folder and informs the Validator program to take action whenever a new file appears.

On present day Linux/Unix/Embedded RTOS systems, there are several utilities like **watch**, **iNotify**, etc. that can be used to implement event-based applications.

The watchdog interface between the validator and the scanner is an indirect one such that they do not need to exchange messages with each other. There may be a common point of interest is the ‘**Scanned QR**’ folder – where the scanner program writes the scanned image data as plaintext files. From there the validation program picks it up to perform the business rules validation.

The process of scanning and validating using a watchdog is shown in the figure below.

### APP – Validator Interface (Scanner and Validating Terminal indirect communication mode)



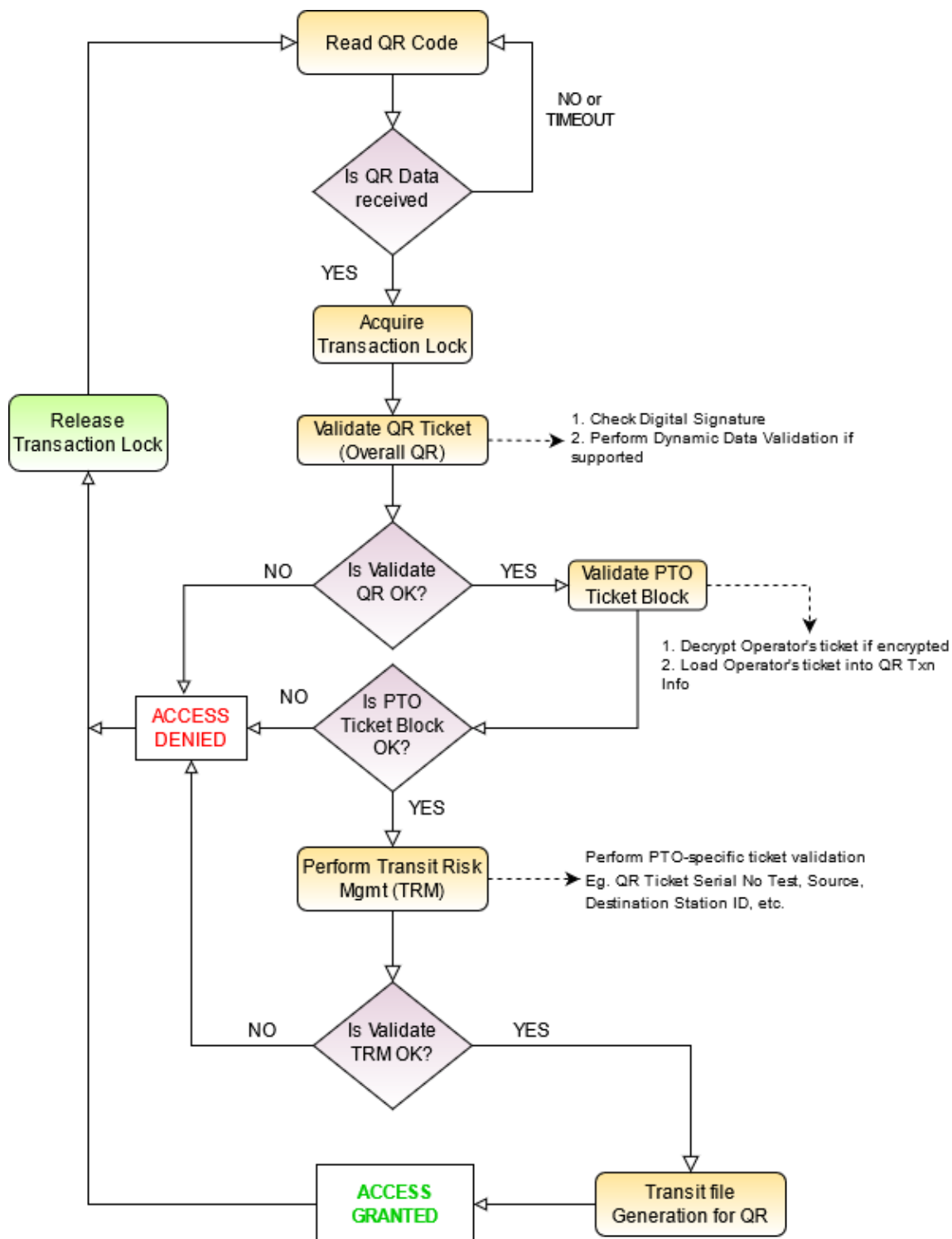
**Figure 32: Scanner and Validator Integration (Watchdog)**

#### 4.3.3.3. Direct API Mode QR Validation in Terminal

In the direct mode, the Scanning application directly sends the scanned data to the validator via an API call. In this case the validator always waits on a “function” of the Scanner as opposed to a physical folder location like in the watchdog mode.



**APP – Validator Interface (Scanner and Validating Terminal direct communication mode)**



**Figure 33: Scanner and Validator Integration (Direct mode)**

#### 4.3.3.4. QR Verification and Validation in Validating Terminal

The process of validation is discussed in detail in the Section 5 of QR Specifications – Part II. However, it is worth mentioning here, that the process of validation is a multi-level process involving validations of business rules and not just digital security validation. Some common business logic validations are given in [Appendix – I: QR Validation & Customer Care](#). These business rules may be applicable to some transit operators like Metro and Bus in much the same way, but business rules are always operator-specific.

#### Offline Validation and Transaction Updates in AFC:

Although the process of Validation in the QR Ticketing System ideally needs to be a hybrid of both online as well as offline components, it should still be capable of performing most of functionality offline so as to facilitate quick throughput of customers through operators' premises, especially during rush hours. Once a QR is successfully verified as having an authentic **digital signature**, it can be safely termed as a valid transit ticket. The transaction information generated at terminals must be updated into the AFC systems so that the PTO can tally it up with the purchased ticket information it received from the TG. Updating the status into the local AFC DB may not necessarily be at real-time and can be scheduled to take place at specified time intervals like every 5 minutes or so.

Transit information generated at terminals is not specific to only QR Tickets but encompasses all kinds of tickets like tokens, Cards, etc. and are more or less common for all kinds of fare media including QR. However, some important transit information must always be present in the ticket or generated during validation. These important parameters include among other information, the following –

- **Transaction Amount** – This parameter holds the amount charged to that transaction. In the case of QR, for an entry transaction, the amount will be zero; for an exit or ticket transaction the amount will be the number of people allowed in the ticket (as indicated in Group Size in Ticket Info part of QR Ticket Block) multiplied by the cost of one ticket (as indicated by the Ticket Fare parameter in the Ticket Info part of the QR Ticket Block)
- **Transaction Type** – This field is defined in Table 4.1 of Part I – QR Code Dataset specifications. The parameter is included in all transit-related information shared between various entities in the QR ecosystem to qualify the intended nature and use of that information. A brief explanation of each type is given below –
  - **QR Purchase** – Indicates that the transaction is related to the purchase of the QR ticket. It is also the default state of any QR ticket issued. This indicator is used in the sending of purchase information from TG to the Operator/PTO's AFC.
  - **QR Transaction** – This indicator is used by the QR validator to send QR ticket information to the backend AFC when a transaction is made that completes the validity of that particular ticket.
  - **QR Verification** – This indicator is used by the QR validator to send QR ticket information to the backend AFC when a transaction is made that indicates that a particular checkpoint in the journey allowed in the ticket has been reached. For example, consider a hypothetical scenario where a QR ticket allows journey from station A to C via B.

Transactions at A and B will be verification transactions while the transaction at C will be a finalizing QR transaction.

- **QR Error** – This indicator is used by the validator to indicate to the backend that a particular QR ticket is being misused. This will allow the PTO or the TG to denylist certain patrons found to abuse the system. Examples of ticket misuse include ticket reuse and exceeding the limits of the ticket to name a few.
- **QR Duplicate** – This indicator is used in any transaction when the Customer Care operator has to request a duplicate copy of a valid QR Ticket from the TG that for some unknown reason did not work at some terminal.
- **QR Update** – This indicator is used in any transaction used by the TG to update the QR ticket. There are various channels via which this can happen- namely via communication with TG, mobile interfaces such as Bluetooth and NFC etc. Examples of update transactions include updating a checkpoint reached or a journey completed.

**Table 4.19: QR Transaction Type Values**

Txn Type	Payment Mode QR – Bits 7 to 5	QR Type – Bits 4 to 0	Txn Type Value	Usage
QR Purchase	b010	b00001	0x41 (65)	TOM, App or TG
QR Transaction	b010	b00010	0x42 (66)	Ticket or Exit
QR Verification	b010	b00011	0x43 (67)	Entry
QR Error	b010	b00100	0x44 (68)	Inspector or exit
QR Duplicate	b010	b00101	0x45(69)	Request for a duplicate of an existing valid Purchased QR ticket.
QR Update	b010	b00110	0x46(70)	Any

- **Transaction Place\*** – This parameter is used to indicate the validator type used to perform the QR transaction. These may include fixed metro gates, mobile ETIMs etc.
- **Transaction Date and Time\*** – This parameter is used to log the date and time when the transaction was performed i.e. it is generated at run-time. The operator is free to use a date format of its own choice.

*\*Individual AFC Systems define these values and they are not required to be part of ticket information. Interested readers may refer to the NCMC Interface Specifications – Part V to check how these are defined and used in NCMC-compliant AFC systems.*

In this section we shall now describe some typical QR ticket validation for different scenarios – metro, bus and passes.

### **Validation of a QR Ticket in Metro/Bus Station Terminal (Double Tap)**

*At entry the commuter makes the first tap.*

1. The QR image is scanned, decoded and validated in accordance with the security level of the QR ticket. This process may involve decryption and authentication via digital signature as well as PTO specific security measures. (Please refer to Part II – QR Specifications for more information)
2. The QR dataset is checked for any tickets meant for use in the current PTO's transit environment. If no tickets are present, then the commuter is indicated to approach the customer care for resolution.
3. If a ticket for the PTO is present, and if the ticket contains QR Dynamic Data (such as a mobile ticket), then validation needs to be done on the basis of that data. The PTO's business rules may choose to do the check at entry or at exit or at both places
4. Next check is to see if the ticket is still active. The output is generated based on the following parameters – Activation (journey) Date, Ticket Validity, current date-time and PTO's business rules. The business rules may include whether Validity is meant to limit entry or exit time.
5. The next check is to see if the ticket serial number (TSN) of the ticket is present in the list of TSNs obtained at the terminal from the TG. Since the TG sends purchased QR information to the AFC, all valid QRs must be present. The output is generated based on the following parameters – the Tkt\_SI\_No present in Common Data of QR and the TSN status list at terminal and PTO's business rules. The business rules may include whether this check is blocking here or at output.
6. If present, a check should be made to see if the ticket has been already used. The output is generated based on the following parameters - the Tkt\_SI\_No present in Common Data of QR and the TSN status list at terminal. Again Business Rules may choose to bypass this check. (The difference between this and the previous step is that the terminal may choose to maintain a separate used ticket list communicated within the terminals of that station)
7. Next verification is to check if the current station is the same or after the source station in the direction of travel. The output is generated based on the following parameters – Src\_Stn & Current Station.

8. Other PTO specific checks as per its business rules.
9. If all checks are passed the commuter is allowed into the paid area and makes the journey.

*The commuter then taps the QR at the exit gate. The following minimum checks are made –*

1. The QR image is scanned, decoded and validated in accordance with the security level of the QR ticket. This process may involve decryption and authentication via digital signature as well as PTO specific security measures. (Please refer to Part II – QR Specifications for more information)
2. The QR dataset is checked for any tickets meant for use in the current PTO's transit environment. If no tickets are present, then the commuter is indicated to approach the customer care for resolution.
3. If a ticket for the PTO is present, and if the ticket contains QR Dynamic Data (such as a mobile ticket), then validation needs to be done on the basis of that data. The PTO's business rules may choose to do the check at entry or at exit or at both places
4. Next step is to check if the ticket is still active. The output is generated based on the following parameters – activation / journey date, Validity, Duration, current date-time and PTO's business rules. The business rules may include whether Validity is meant to limit entry or exit time.
5. The next check is to see if the ticket serial number (TSN) is present. Since the TG sends purchased QR information to the AFC, all valid QRs must be present. The output is generated based on the following parameters – the Tkt\_SI\_No present in Common Data of QR and the TSN status list at terminal and PTO's business rules. The business rules may skip this if the check is made at entry.
6. If present, a check should be made to check if the ticket has been already used. The output is generated based on the following parameters - the Tkt\_SI\_No present in Common Data of QR and the TSN status list at terminal. Again Business Rules may choose to bypass this check. (The difference between this and the previous step is that the terminal may choose to maintain a separate used ticket list communicated within the terminals of that station)
7. Finally, a check is made if the exit validator is in a station that is within the journey parameters allowed by the QR ticket.

#### **Validation of a QR Ticket in Bus Terminal (Single Tap scenario)**

In case of bus transit, irrespective of the terminal type, only single tap verification is needed for QR tickets. When the commuter taps the QR at the validator, the following minimum checks are performed –

## QR Ticketing System – Interface Specification

1. The QR image is scanned, decoded and validated in accordance with the security level of the QR ticket. This process may involve decryption and authentication via digital signature as well as PTO specific security measures. (Please refer to Part II – QR Specifications for details)
2. The QR dataset is checked for any tickets meant for use in the current PTO's transit environment. If no tickets are present, then the commuter is indicated to approach the conductor or driver for resolution.
3. If a ticket for the PTO is present, and if the ticket contains QR Dynamic Data (such as a mobile ticket), then validation needs to be done on the basis of that data.
4. Next step is to check if the ticket is still active. The output is generated based on the following parameters – activation / journey date, Validity, Duration, current date-time and PTO's business rules.
5. Next step is to check if the ticket serial number (TSN) is present. Since the TG sends purchased QR information to the AFC, all valid QRs must be present. The output is generated based on the following parameters – the Tkt\_SI\_No present in Common Data of QR and the TSN status list at terminal and PTO's business rules.
6. If present, a check should be made to check if the ticket has already been used by checking the terminal's local used TSN list
7. Next check should ensure that the QR ticket journey parameters are within the scope of the bus's current trip parameters. That is, checks should be performed to see if the entry station is the same or after the source station AND if the exit station is the same or before the destination station in the direction of travel. The output is generated based on the following parameters - Src\_Stn, current entry station and current exit station.
8. Other PTO specific checks.

### **QR Verification and Validation of Trips/Passes in Validating Terminal**

The process of validating passes is described in this section.

Assumption: For passes, 'Total Fare' in QR Common Data is the 'Pass Value' and 'Ticket Fare' in QR Ticket Block is the 'Available Pass Balance'.

#### **Verification Steps:**

1. When the user purchases a QR Pass Ticket, just like any other QR Ticket, the TG pushes the "Purchase QR" information to the AFC. The AFC must push the Ticket Serial Number – TSN and the "Available Balance" to all the terminals.

2. On the Activation Date (day of journey), when the user presents the QR Pass, the Validator must check if the “Available Balance” is equal to the “Available Pass Balance” on the QR Payload. The first time the user presents a pass these values will be equal. Let us assume that these amounts are 500. The terminal will validate the QR and send the <TSN, StationID> (Source) and other transit information back to the AFC.
3. On exit, the terminal will validate the QR and this time send the <TSN, StationID> (Destination) and other transit information back to the AFC.
4. The AFC must match up this pair, calculate the fare that needs to be deducted from the “Available Pass Balance”. The AFC must then again push the <TSN, Available Balance> to both the TG and the terminals. Let us assume that the new “Available Balance” is 450.
5. The App Provider must provide the option to the user to update the balance on his App like “Update Pass”. When the user uses this option, the TG sends the new QR Ciphertext with updated Ticket Fare (new Available Pass Balance) to the App. Naturally, if the Digital Signature is also sent, then this signature will also change. The App must then render a new QR image with this new payload. The “Ticket Fare” filed in the QR Ticket Block will have the new “Available Pass Balance” which is 450 in this example.
6. The next time if the user presents the QR with the old QR (i.e. without updating the balance from TG), the terminal will flag it as an invalid QR. Recall from Step 4 this balance is 450 now and the value on the payload is still the old balance 500. The user must then use the “Update Pass” of Step 5 to get the updated QR image and present again.
7. When the presented QR code goes below the minimum threshold and the user tries and use it, the terminal should “DENY” entry. This minimum threshold should be the “nBaseFare” from the “nFareRules” Policy. Please refer [Figure 41](#).
8. In this way the cycle shall continue. Once the available balance reaches a minimum threshold, the App should be capable of informing the user with a warning.

#### 4.3.3.5. Validating Terminal and AFC System Interface

After a QR Ticket is validated at a terminal the process of updating transit information in the AFC Systems – its protocol, communication mode, architecture, etc. – is actually not within the scope of the QR Ticketing System specifications. But for the sake of completeness, we have included a section in the QR Ticketing System – Implementation Guidelines; to show how information generated from using a QR Ticket, could actually flow back into the AFC. *Please refer to Section 4.1.5 of the QR Ticketing System – Implementation Guidelines for details.*

#### 4.3.4. Optional Futuristic Features

QR codes are generally static in nature, i.e. once a code is rendered it is not changed. But the APP is actually online with the TG, so periodic updates can be sent to the APP by the TG. The inclusion of the QR Dynamic Data in the QR Code Dataset was introduced for that very purpose. The TG can send the QR Status updates to the APP and the APP can render fresh QR images from time. Even if the Status field is not sent from the TG, the QR Update Datetime field can be used by the APP to refresh the QR at a certain interval. This allows the PTO to thwart any attempts by rogues to outsmart the system with stale or copied QRs. For some elegant uses of Dynamic Data please refer to [Appendix – I: QR Validation & Customer Care](#) and [Appendix – III: Futuristic QR Ticketing System](#).

Easier said than done though, updating QR Status in real-time on the smartphone APP however involves high complexity because mobile phones roam and to ‘find’ a phone and update status at real-time is a complicated technical problem. Even for updating status with NFC or Bluetooth and WiFi requires the user of the APP to enable the feature on his/her mobile.

These features have been termed as ‘futuristic’ features in this specification and some methods to incorporate them are described in [Appendix – III: Futuristic QR Ticketing System](#) and also in [Section 4.2 of the Implementation Guidelines](#) document.



## 5. Integrated TG Interface with External Entities

Policy DB is tightly coupled with the TG and is together called the Integrated TG. The Policy DB may need access from other external entities like PTO in order to query/update its own policies. So clearly the Policy DB must be logically segregated into Operator-specific views. When a registered user of an authentic APP Provider queries the fare rules the request takes place through the TG. Policy DB also contains the information about the PTO and APP Provider that the TG provides services for. The TG – APP Provider Interface is detailed out in [Section 5.3](#). [Sections 5.1 & 5.2](#) describe the Application Program Interface (API) between the TG and PTO for the purposes of querying/updating its policies.

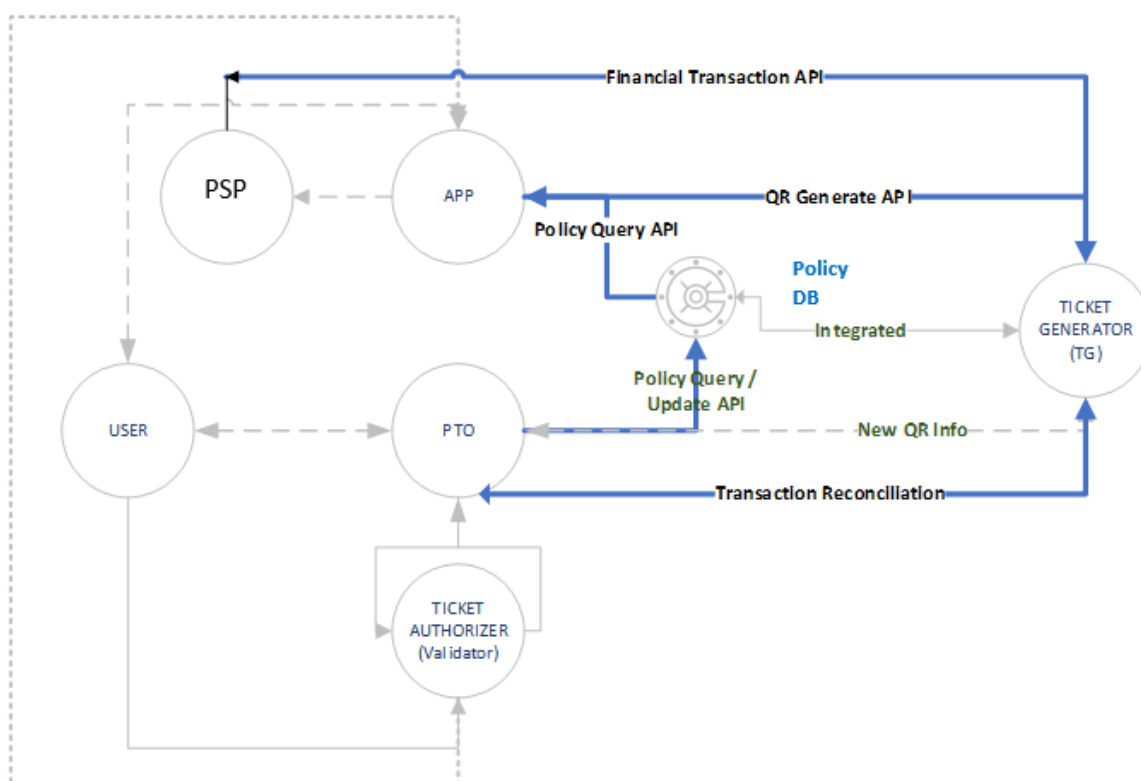


Figure 34: Integrated TG (TG + Policy DB) Interfaces

## 5.1. PTO Updates Policy DB

Policies like Fare rules, Route Station Information are PTO-specific and any changes or updates on these policies must be solely left to the discretion of the PTO. The interface shall be very much similar to the ones specified in the NCMC Interface Specifications – Part V.

### Policy update Query structure

The following section describes the data structure and elements of the Policy Rules that represents the contents of update policy query. Please note that each element is a logical unit and may actually have to be implemented entirely in one or more Database Tables.

**Table 5.1: Update Policy Query Structure**

Tag<Value>	Nested level-1 (Policy Group)	Nested level-2	Description	Data Presence	Data Type / Length
PTO <nOperatorID, OperatorName, OperationType>*	PolicyID#1 (Security-related group)	nEncryption	Whether the PTO supports encryption of its Tickets or not. By default, it is 0 – ‘No Encryption’.	O	NS / 1 Range: 1-5
	PolicyID#2 (Service-related group)	nVer	QR layout Version Capability. The PTO may have limitation for maximum QR Version that it can scan. It may also be the range of the QR versions it supports.	O	(NS.NS) / (2.2) Range: 17-40
		nService	Services provided by the PTO. A PTO may support multiple services. In such a case the structure of Policy exchange between AFC and	M	NS / 3 Range: 0-255

# QR Ticketing System – Interface Specification

			TG must be pre-agreed.		
		nProduct	Product of the PTO. A PTO may support multiple products. In such a case the structure of Policy exchange between AFC and TG must be pre-agreed.	M	NS / 5 Range: 0-65535
		nProductMinAmt	Minimum price of the Product. A PTO may support multiple products. In such a case the structure of Policy exchange between AFC and TG must be pre-agreed.	M	NS / 5 Range: 0-65535
		nValidity	The Validity Time in minutes for the amount of time a ticket shall remain valid starting from the date and time of journey.	M	NS / 5 Range: 0-65535
		nDuration	The Duration time in minutes for the amount of time a ticket shall remain valid starting from the scanned time on the date of journey.	M	NS / 5 Range: 0-65535
		nGrpMinSize	A PTO may specify the minimum size of a group ticket. So for that	O	NS/2 Range: 1-63

## QR Ticketing System – Interface Specification

			operator, the group size field cannot have a value lesser than this value.		
		nGrpMaxSize	A PTO may specify the maximum size of a group ticket. So for that operator, the group size field cannot have a value greater than this value.	O	NS/2 Range: 1-63
		nHashAlgo	The preferred Hashing algorithm of the PTO that must be used to check the integrity of messages exchanged between the PTO's AFC System and the TG. If not provided then SHA256 shall be used.	O	AN / 20  Like SHA256, SHA512, MD5, etc.
		nScan	List of scanning options available with the PTO. E.g. Camera, BLE, Wifi, etc. Default is Camera scanning. A PTO may support multiple scanners. In such a case the structure of Policy exchange between AFC and TG must be pre-agreed.	O	AN / 100  Possible values = Camera, NFC, Bluetooth or BLE, WiFi, Others

## QR Ticketing System – Interface Specification

	PolicyID#3 (Network-related group)	nTOMID	The value contains data object for the Ticket Office terminal ID or Customer Care terminal ID that will send request to TG either for Paper QR ticket or for customer care validation purposes.	M	NS / 5  Range: 0-65535
		nAFC_IP	The value contains the well-known IP Address and Port number combination of the AFC System of the PTO. This may also be the URL depending upon the agreed SLA between the TG and PTO.	M	Standard IP v4 Address:Port  Format:  IP – xxx.xxx.xxx.xxx  Port – nnnnn  Where x and n are of type NS.
	PolicyID#4 (Finance-related group)	nOpBank	This is the Bank Name and Branch code.	M	AN / 40
		nOpIfsc	Contains the IFSC code of the PTO.	M	AN / 40
		nOpAccID	Contains the Bank Account ID. This is extremely essential for the TG to make payments to the operators at end of day every day. (Meaningful for TG)	M	AN / 40

	Policy ID#5 (Route / Station - related group)	nRoutes & nStations	All the Routes and related Stations for this PTO. If Routes are not present, this shall contain only the station information.	M	Variable
	Policy ID#6 (Fare-related group)	nFareRules	The value contains data object as per PTO business rules. PTO must define the fare extraction rule for TG to be able to calculate fare. <i>The fare table may contain multiple fare-rules based on requirements.</i>	O	Variable

**Operator ID** is defined in [Table 4.3](#).

**Operator Name** is of type 'AS' and can be of length 256 chars. It may also have the address of the Operator.

**Operation Type** is only relevant only for Transit operators. Some operators may describe a double-fold request. That is first all routes are sent to user when the user wants to buy tickets for that Operators, and only thereafter, based on the selection of a route, all the information about stations or stops for that route are sent. Other operators may simply have a single-fold stations request. We describe then as Operation type: **0=Non-route based**, **1=Route-based**. Values 2 through 9 are used. They may be defined later if required.

Fare calculation is a complicated process and needs to be explained in some detail. So first we will show the policy update procedure for Policy #1 through Policy #5 and only after that we will show the update procedure for fare rules.

#### 5.1.1. Update Policy Message Exchange

As the data structure is in place, we shall now describe the procedure to update the Policy Database.

To begin with, first we only show the message exchanges for Policy #1 through Policy #5. The procedure to be followed is the same as TG-AFC interface for Purchase QR ([Section 4.3.2](#)); only the contents of the 'Message Payload' changes.

### Policy Update Request Message from Generic AFC to TG

**Table 5.2: Policy Update Request Message from AFC**

QR Message Element	Size (bytes)	Data Type	Description
Hash Token	64	AN	SHA256 Hash of the Message Payload i.e. the Message Key and Payload Data
Message Key	2	NS	11 – QR_UPD_POLICY_REQUEST Key ID = 11
Payload Data	Variable	AN	The Policy update configuration JSON files – any one of <a href="#">Figure 35</a> through <a href="#">Figure 39</a> or <a href="#">Figure 41</a> .

- For this we establish 2 new Message Keys – “[QR\\_UPD\\_POLICY\\_REQUEST](#)” and “[QR\\_UPD\\_POLICY\\_RESPONSE](#)”, for Policy Update Request and Policy Update Response respectively. The Policy ID number ‘n’, where  $1 \leq n \leq 6$ , must be sent in separate records as shown in [Figure 35](#) through [Figure 39](#) or [Figure 41](#).
- The file naming convention followed for policies related information shall be **qrpu\_DDMMYYYYHHMMSS\_PToid\_TGID.JSON**.
- Let the date and time used for Policy update be 06/04/2020@14:20:35 hours through 06/04/2020@14:20:39 hours for Policies 1 through 5.
- We shall only show the example using PTO ID = 10

The PTO may send one, more than one or all the policies in one Message inside the ‘Policy\_Details’ element. Here we show them by segregating them into single policies. The Message Payload for Policies 1 through 5 using JSON structures are shown in the following diagrams.

### Policy Update Request Message from AFC to TG

From Operator ID 10 – Policy# 1:

```
"QR_Update_Policy_Request": {
  "Hash_Token": {
    "Hash_Value":
      "f26612da644e72b42a99040d3e71ce7fea92888dd144e97e468867ff4cc
      aedab"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "11"
    },
    "Payload_Data": {
      "Operator_Details": {
        "OperatorId": "10",
        "OperatorName": "PTOName10",
        "OperationType": "0",
      },
      "Filename": "qrpu_06042020142035_10_23",
      "No_Of_Records": "1",
      "Record_1": {
        "Policy_Id": "1",
        "Security-related": {
          "nEncryption": "1"
        }
      }
    }
  }
}
```

Figure 35: Update Policy ID#1 File qrpu\_06042020142035\_10\_23.JSON



From Operator ID 10 – Policy# 2:

```
"QR_Update_Policy_Request": {
  "Hash_Token": {
    "Hash_Value":
      "2129ef13a1208b993b2e5c1c11318af52237f8a5579db41392a03646fb137271"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "11"
    },
    "Payload_Data": {
      "Operator_Details": {
        "OperatorId": "10",
        "OperatorName": "PTOName10",
        "OperationType": "0",
      },
      "Filename": "qrpu_06042020142036_10_23",
      "No_Of_Records": "1",
      "Record_1": {
        "Policy_Id": "2",
        "Service-related": {
          "nVer": "1-32",
          "nService": "1-Regular;2-Express;
            21-Normal Feeder",
          "nProduct": "1-Standard;2-SeniorCitizenPass;
            3-StudentPass;4-DailyPass;5-MonthlyPass;
            6-WeekendPass;9-SpecialPass",
          "nProductMinAmt": "1-NA;2-200;3-200;4-200;
            5-200;6-200;9-200",
          "nValidity": "480",
          "nDuration": "180",
          "nGrpSizMin": "2",
          "nGrpSizMax": "20",
          "nHashAlgo": "SHA256",
          "nScan": "Camera;NFC"
        }
      }
    }
  }
}
```

Figure 36: Update Policy ID#2 File qrpu\_06042020142036\_10\_23.JSON

From Operator ID 10 – Policy# 3:

```
"QR_Update_Policy_Request": {
  "Hash_Token": {
    "Hash_Value":
      "f2053535f54327d04095818bd75953022d9bd5d64dedc97548c432fd5138b364"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "11"
    },
    "Payload_Data": {
      "Operator_Details": {
        "OperatorId": "10",
        "OperatorName": "PTOName10",
        "OperationType": "0",
      },
      "Filename": "qrpu_06042020142037_10_23",
      "No_Of_Records": "1",
      "Record_1": {
        "Policy_Id": "3",
        "Network-related": {
          "nTOM_ID": "6000-20999",
          "nAFC_IP_Port": "114.14.115.56:4500"
        }
      }
    }
  }
}
```

**Figure 37: Update Policy ID#3 File qrpu\_06042020142037\_10\_23.JSON**

From Operator ID 10 – Policy# 4:

```
"QR_Update_Policy_Request": {
  "Hash_Token": {
    "Hash_Value":
      "1893e91ed84f9c52d20ae15453d29b2f428d1c02c542d8a7a9a38a569d6f0c29"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "11"
    },
    "Payload_Data": {
      "Operator_Details": {
        "OperatorId": "10",
        "OperatorName": "PTOName10",
        "OperationType": "0",
      },
      "Filename": "qrpu_06042020142038_10_23",
      "No_Of_Records": "1",
      "Record_1": {
        "Policy_Id": "4",
        "Finance-related": {
          "nOpBank": "GokulBankofIndia;5698",
          "nOpIfsc": "GKBI7777",
          "nOpAccID": "77777777-6528",
        }
      }
    }
  }
}
```

Figure 38: Update Policy ID#4 File qrpu\_06042020142038\_10\_23.JSON

From Operator ID 10 – Policy# 5:

```

"QR_Update_Policy_Request": {
  "Hash_Token": {
    "Hash_Value":
      "fcf6d0aa45aa716817f7d2389fe2eb7c11c4ca1e2ad9b4a15cddd47d8035ba14"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "11"
    },
    "Payload_Data": {
      "Operator_Details": {
        "OperatorId": "10",
        "OperatorName": "PTOName10",
        "OperationType": "0",
      },
      "Filename": "qrpu_06042020142039_10_23",
      "No_Of_Records": "1",
      "Record_1": {
        "Policy_Id": "5",
        "Routes_Stations-related": {
          "nRoutes": {},
          "nStations": {
            "Station_1": {
              "Station_Id": "1",
              "Station_Name": "StationName1",
              "Station_Location": "PPppp;QQqqq"
            },
            "Station_2": {
              "Station_Id": "2",
              "Station_Name": "StationName2",
              "Station_Location": "PPppp;QQqqq"
            },
            "...
            "Station_N": {
              "Station_Id": "N",
              "Station_Name": "StationNameN",
              "Station_Location": "PPppp;QQqqq"
            }
          }
        }
      }
    }
  }
}

```

**Figure 39: Update Policy ID#5 File qrpu\_06042020142039\_10\_23.JSON**

*\*Station location may also be included. The format NNnnn implies the mantissa and the exponent part – e.g. 28205 is geolocation 28.205. Here PPppp implies the latitude and QQqqq is the longitude.*

The mode of communication between AFC Systems and other entities is assumed to be TCP IP socket communication. The TG must run a service at a well-known IP Address and Port Number to accept policy update request from PTO's AFC. When this service receives a request, like the files shown in [Figure 35](#)

through Figure 39, the TG must respond with the acknowledgement. We shall now proceed to describe the Generic Policy Update responses from TG.

### **Policy Update Response Message from TG to Generic AFC**

Following the same convention as has been used so far, we first construct the generic response that should be sent from the TG to the AFC as an acknowledgement of for the update policy request.

**Table 5.3: Update Policy Response Structure**

QR Message Payload Element	Size (bytes)	Data Type	Description
Hash Token	64	AN	SHA256 Hash of the Message Payload i.e. the Message Key and Payload Data
Message Key	2	NS	12 – QR_UPD_POLICY_RESPONSE Key ID = 12
Payload Data	Variable	AN	Filename = Name of the file sent as a field inside the Purchase QR Request.
			Ack = String “ACK-N”, where N is value of the No_Of_Records field in the request.
			TG_ID = ID of the TG (in all examples in this document we have used the value 23)
			Operator ID = ID of the Operator (in all examples in this document we have used two values for Operator/PTO ID, 10 and 135)
			Policy No = Policy Numbers Updated – separated by ‘;’ if more than one.
			ErrorMsg = Can be either the string “SUCCESS” by default for no error or some other elaborate message chosen by the TG as shown in Table 5.4. When the TG receives an ACK that has an ErrorMsg other than SUCCESS, it should try to resend the message again.

Some error messages have been defined for Policy Update Request messages at the TG.

**Table 5.4: Update Policy Response Error Codes and Messages**

Error Code	Response String
0	OK-Policy Updated
1	ERROR-Policy Key Mismatch
2	ERROR-Data format Error
3	ERROR-Mandatory Element missing
4-20	ERROR-Policy Database Error
21-2000	ERROR: <21-2000> (RFU)
2001-3000	ERROR: 2001-3000 (Proprietary use)

**Purchase QR Response for PTO ID 10:**

We shall show the response for only one policy update request – Policy#2 – qrupu\_06042020142036\_10\_23 from [Figure 36](#).

**Table 5.5: Update Policy Response Message – TG to AFC**

Operator ID	QR Message Format – TG & AFC communication		
	Hash Token – 64 bytes SHA-256 of Message Payload	Message Payload	
		Message Key – 2 bytes	Payload Data – Variable size
10	eab8002fdddddadafa06bc33b01d88eb7b990e73af7e683ce7881f4c5d9b15246	12	Element ‘Payload Data’ in <a href="#">Figure 40</a> .

The JSON file representation of the acknowledgement is as follows.

ACK from TG to Operator ID 10 – Policy# 2:

```
"QR_Update_Policy_Response": {
  "Hash_Token": {
    "Hash_Value":
      "eab8002fdddddadafa06bc33b01d88eb7b990e73af7e683ce7881f4c5d9
      b15246"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "12"
    },
    "Payload_Data": {
      "Filename": "qrpu_06042020142036_10_23",
      "Ack": "ACK-1",
      "TG_ID": "23",
      "OpID": "10",
      "Policy_Id": "2",
      "ErrorMsg": "OK-Policy Updated"
    }
  }
}
```

**Figure 40: Update Policy Response for Policy ID#2 – TG to AFC**

Once the TG has sent the success message back to AFC, it updates the policy in the Policy Database

Updates for Policy# 6 – Fares-related – is also exactly the same as the ones shown in this section. But before we actually show the actual message exchanges, we need to first describe a generic data structure to describe the Fares-related policy group.

### 5.1.2. Generic Fare Calculation Scheme

The TG may choose to keep a generic fare calculation scheme (algorithm) which the PTOs can then use to calculate the Fare between two stations.

*This is only an illustration and the PTOs are not mandated to use it. All major PTOs usually have their well-established Fare-Rules Tables and should continue to use them. Those established Fare Rules can directly be imported and loaded into the Policy DB. But for smaller PTOs that do not have a proper fare rules calculator system, a scheme like this one may be the first starting point.*

The table below describes the elements that are essential to construct a general fare scheme for a transit operator.

Table 5.6: Data elements for Generic Fare Calculation Structure

Top Level	Scheme Parameter Sl. No	Nested level-1	Description	Unit / Data Type / Length	Data Presence
nFareRules <i>Policy ID#6 (Fare-related group)</i>	1	nBaseDistance	The first 'n' distance in meters to be used to calculate. Default is 2000 meters (2 Km)	NS / 10	O
	2	nBaseFare	The most basic fare for the nBaseDistance. The default is ₹10 or 1000 paisa	Paisa / NS / 10	O
	3	nIncrementalDist	Every additional 'n' meters specified here will generate a constant incremental fare. Default is 100 meters.	NS / 10	O
	4	nIncrementalFare	Fare of every incremental distance. Default is ₹3 or 300 paisa.	Paisa / NS / 10	O
	5	nTG_MDR	The fare may include the MDR that the TG will charge for providing the Ticket Generation and other services to the PTOs. This is an addition	A % / (NS.NS) / (2.2)	O



## QR Ticketing System – Interface Specification

			operation on the Fare.		
	6	nAPP_MDR	The fare may include the MDR that the APP Providers (providers of Webclients and Mobile APPs) will charge for providing the Ticket Purchasing and other services to the PTOs. This is an addition operation on the Fare.	A % / (NS.NS) / (2.2)	O
	7	nPSP_MDR	The fare may include the MDR that the Payment Service Providers will charge for providing the Ticket Payment and other services to the PTOs. This is an addition operation on the Fare.	A % / (NS.NS) / (2.2)	O
	8	nTaxes	Any applicable tax that may be levied by governmental policies. This is an addition operation on the Fare.	A % / (NS.NS) / (2.2)	O

	9	nAdhocDiscount		The Operator may choose to give discounts to its customers on special days like Independence day or some national festivals. This is a subtraction operation on the Fare. The default value is 0.	Paisa / NS / 10	O
	10	nRouteID		For a PTO having operation type = 1 i.e. route-based, the distance or fare matrix must be based on a particular route	AN/8	O
	11	Repeated for each route for a PTO having operation type = 1 i.e. route-based	nDistanceMatrix	A NxN matrix structure that contains the distance between 2 stations. Distances are in meters. If there is no connection between 2 stations the distance must be given as 0. Semicolon separated values to represent columns.	NS / Variable	O
	12	Repeated for each route for a PTO having	nFaresMatrix	A NxN matrix structure that contains the fares between 2 stations. Fares	NS / Variable	O

		operation type = 1 i.e. route-based. For non-route based operation this may be sent as a large single matrix.		are in meters. If there is no connection between 2 stations the fare is given as 0. Semi-colon separated values to represent columns.		
--	--	---	--	---	--	--

### Calculating the Fare between 2 stations:

The objective is to create 2 Policy DB tables – one for Distance and the other for Fares. Some PTOs may already have existing distances between every two stations. Distance can also be calculated using location coordinates. But that is a very complicated procedure. Similarly, PTOs may also have existing fares between two stations. In such cases there is no need to apply the Fare-calculation algorithm. The distance and fare tables can be directly imported into the DB.

For our example we assume that the PTO (PTO ID = 10) has the distance matrix available, as shown in [Table 5.7](#), but needs to calculate the fares for the same. Distances are always between 2 stations. Please recall that PTO ID = 10 has been designated as having Operation Type = 0 i.e. Non-route based. So if the PTO is designated as having Operation Type = 1 or Route-based, then the distance matrix shall have to be sent separately for each route.

For simplicity, let us assume that the distance matrix is a simple 6x6 matrix as shown here.

**Table 5.7: A Simple Distance Matrix – nDistanceMatrix**

StationID	1057	1058	1071	1148	1161	1185
1057	0	1000	1500	2000	2500	3000
1058	1000	0	500	1000	1500	2000
1071	1500	500	0	500	1000	1500
1148	2000	1000	500	0	500	1000
1161	2500	1500	1000	500	0	500
1185	3000	2000	1500	1000	500	0

We can use the same Update Policy Message Exchange procedure shown in [Section 5.1.1](#) to send the fare rules as follows. Here we are only showing the JSON format. The details of packing up the message structure and other details are omitted here as it is exactly the same as in [Section 5.1.1](#).

```
"QR_Update_Policy_Request": {
  "Hash_Token": {
    "Hash_Value":
      "3bbfa414cd7ba405c14686ae7fe6b4e916dca563b489d96a287f4b2a91ce836b"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "11"
    },
    "Payload_Data": {
      "Operator_Details": {
        "OperatorId": "10",
        "OperatorName": "PTOName10",
        "OperationType": "0",
      },
      "Filename": "qrpu_06042020142040_10_23",
      "No_Of_Records": "1",
      "Record_1": {
        "Policy_Id": "6",
        "Fares-related": {
          "nFareRules": {
            "nBaseDistance": "1000",
            "nBaseFare": "1000",
            "nSubsequentFare": "300",
            "nTG_MDR": "1",
            "nAPP_MDR": "1",
            "nPSP_MDR": "0",
            "nTaxes": "0",
            "nAdhocDiscount": "0",
            "nDistanceMatrix_1": {
              "nRouteID": "",
              "nDistanceMatrix": {
                "1057": "0;1000;1500;2000;2500;3000",
                "1058": "1000;0;500;1000;1500;2000",
                "1071": "1500;500;0;500;1000;1500",
                "1148": "2000;1000;500;0;500;1000",
                "1161": "2500;1500;1000;500;0;500",
                "1185": "3000;2000;1500;1000;500;0"
              }
            }
          }
        }
      }
    }
  }
}
```

**Figure 41: Update Policy ID#6 File qrpu\_06042020142040\_10\_23.JSON**

## **Fare Calculation algorithm:**

The TG shall use the information received from the PTO and calculate the fares between two stations. The result of applying this algorithm is to generate the Simple Fare Matrix.

The input to this algorithm is 2 stations – station1 and station2. For illustration purpose let us suppose that station1 = 1058 and station2 = 1161.

Fare is calculated with the following logic.

1. Find the Distance 'D' from the nDistanceMatrix received. D = 1500 meters.
2. Calculate the Total Fare as

$$\text{TotalFare} = \text{nBaseFare} + \left[ \left\{ \frac{D - \text{nBaseDistance}}{\text{nIncrementalDist}} \right\} * \text{nIncrementalFare} \right], \text{ if } D > \text{nBaseDistance} \text{ ----- (i)}$$

Or

$$\text{TotalFare} = \text{BaseDistanceFare}, \text{ if } D \leq \text{nBaseDistance} \text{ ----- (ii)}$$

Since  $D > \text{nBaseDistance}$ , therefore,  $\text{TotalFare} = 1000 + \left[ \left\{ \frac{1500 - 1000}{100} \right\} * 300 \right] = 2500$

3. Add different MDRs, Applicable Taxes & Discount (Scheme Parameters Sl. No. 4, 5, 6, 7, 8 & 9 in [Table 5.6](#)) if applicable.

$$\text{TotalMDR} = [(\text{TG\_MDR} + \text{APP\_MDR} + \text{PSP\_MDR} + \text{TAX} - \text{Ad-hocDiscount}) * 0.01] * \text{TotalFare} \text{ (2)}$$

$$\text{TotalMDR} = [1 + 1 + 0 + 0 - 0] * .01 * 2500 = 50$$

4. Return Ticket Fare as

$$\text{TicketFare} = \text{TotalFare} + \text{TotalMDR} \text{ (3)}$$

Finally, Ticket Fare = 2500 + 50 = 2550 paisa or ₹25.50

Applying the same algorithm for each station pair provided in [Figure 41](#), the TG should produce the Fares Matrix as shown in below in [Table 5.8](#).

**Table 5.8: A Simple Fares Matrix – nFaresMatrix**

StationID	1057	1058	1071	1148	1161	1185
1057	0	1020	2550	4080	5610	7140
1058	1020	0	1020	1020	2550	4080
1071	2550	1020	0	1020	1020	2550
1148	4080	1020	1020	0	1020	1020

1161	5610	2550	1020	1020	0	1020
1185	7140	4080	2550	1020	1020	0

### 5.1.3. Query Policy Message Exchange

The Query Policy Message Exchange is exactly similar to the Update Policy Message Exchange, except the following:

1. In this case the AFC will send the “Query” request for specific policy and the TG responds by sending the updated policy configuration file retrieved from the Policy DB.
2. Payload Data in request message will have an empty ‘Filename’ and the ‘Number of records’ shall mention the number of policies requested. If the request is for more than one policy, then the Policy Id’s must be separated by a ‘;’ like “1;4;6”.

It uses the same service used by the Update Query exchanges.

For this we again establish 2 new Message Keys that must be agreed with the all the PTOs – “QR\_QRY\_POLICY\_REQUEST” and “QR\_QRY\_POLICY\_RESPONSE”. Policy ID number must be sent in the Payload Data.

#### Policy Query Request Message from an AFC to TG

This message also follows the same protocol as all the other requests described thus far viz. QR Purchase and Update Policy requests. In tabular form, the request can be summed up as follows.

**Table 5.9: Generic Query Policy Request Message Format**

QR Message Element	Size (bytes)	Data Type	Description
Hash Token	64	AN	SHA256 Hash of the Message Payload i.e. the Message Key and Payload Data
Message Key	2	NS	21 – QR_QRY_POLICY_REQUEST Key ID = 21
Payload Data	Variable	AN	The ‘Payload Data’ element of Policy query JSON file – like <a href="#">Figure 42</a> . The query may request the responses for multiple policies i.e. policies 1 through 6.

Below we show the request in JSON format.

```

"QR_Query_Policy_Request": {
  "Hash-Token": {
    "Hash_Value":
      "205579d91ca87ac37da5c19d27e8f3dc0a99db08266b7da270074c7ddba
      35166"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "21"
    },
    "Payload_Data": {
      "Operator_Details": {
        "OperatorId": "10",
        "OperatorName": "PTOName10",
        "OperationType": "0",
      },
      "Filename": "",
      "No_Of_Records": "1",
      "Policy_Id": "6"
    }
  }
}

```

**Figure 42: Query Policy ID#6 Request Data**

#### **Policy Query Response Message from TG to AFC**

In response to the above request, the TG will construct the suitable response having the policy details in the payload. As noted earlier, there may be more than policy requested in the message.

Assuming the date and time of sending the response as 07/04/2020@18:01:30 hours, the TG creates a new JSON file bearing the name **qrpq\_07042020180130\_10\_23.JSON**. The payload data shall contain the DistanceMatrix and the FaresMatrix shown in [Table 5.7](#) and [Table 5.8](#).

In tabular form, the structure of the response is as follows:

**Table 5.10: Generic Query Policy Response Message Format**

QR Message Element	Size (bytes)	Data Type	Description
Hash Token	64	AN	SHA256 Hash of the Message Payload i.e. the Message Key and Payload Data
Message Key	2	NS	<a href="#">22 – QR_QRY_POLICY_RESPONSE</a> Key ID = 22
Payload Data	Variable	AN	The 'Payload Data' element of Policy query JSON file – like <a href="#">Figure 43</a> .

Response Payload Data (Query Policy# 6) for Operator ID 10:

```
"Payload_Data": {
  "Operator_Details": {
    "OperatorId": "10",
    "OperatorName": "PTOName10",
    "OperationType": "0",
  },
  "Filename": "qrpq_07042020180130_10_23",
  "No_Of_Records": "1",
  "Record_1": {
    "Policy_Id": "6",
    "Fares-related": {
      "nFareRules": {
        "nBaseDistance": "1000",
        "nBaseFare": "1000",
        "nSubsequentFare": "300",
        "nTG_MDR": "1",
        "nAPP_MDR": "1",
        "nPSP_MDR": "0",
        "nTaxes": "0",
        "nAdhocDiscount": "0",
        "nDistanceMatrix_1": {
          "nRouteID": "",
          "nDistanceMatrix": {
            "1057": "0;1000;1500;2000;2500;3000",
            "1058": "1000;0;500;1000;1500;2000",
            "1071": "1500;500;0;500;1000;1500",
            "1148": "2000;1000;500;0;500;1000",
            "1161": "2500;1500;1000;500;0;500",
            "1185": "3000;2000;1500;1000;500;0"
          }
        }
      },
      "nFaresMatrix_1": {
        "nRouteID": "",
        "nFaresMatrix": {
          "1057": "0,1020,2550,4080,5610,7140",
          "1058": "1020,0,1020,1020,2550,4080",
          "1071": "2550,1020,0,1020,1020,2550",
          "1148": "4080,1020,1020,0,1020,1020",
          "1161": "5610,2550,1020,1020,0,1020",
          "1185": "7140,4080,2550,1020,1020,0"
        }
      }
    }
  }
}
```

Figure 43: Query Policy ID#6 Payload Data



ACK from TG to PTO ID 10 – Policy# 6:

```
"QR_Query_Policy_Response": {
  "Hash_Token": {
    "Hash_Value":
      "b680f623c48dc1bea088ea79ec67afad7c8bdafa6dbe5c344bb1b92e50b
      ceble"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "22"
    },
    "Payload_Data": "As shown in Figure 43"
  }
}
```

Figure 44: Query Policy#6 Response File qrpq\_07042020180130\_10\_23 – TG to AFC

## 5.2. PSP – APP– TG – AFC Interface

This section describes the financial related transaction that shall be exchanged among PSP, APP, TG and AFC System. The Payment Service Provider (or PSP in short) gives the electronic payment processing system interface to the APP. The role of the PSP in the entire financial settlement process is also discussed in this section.

As TG is front facing for Operators (PTO), TG should be responsible for payment to PTO. Payment reconciliation with PTO is also the responsibility of the TG.

### 5.2.1. PSP – APP Provider– TG Interface

In a practical scenario the APP provider becomes the merchant for the Payment Service Provider (or PSP in short). The PSP receives all the customer amounts on behalf of the APP Provider. There are two ways to settle the amounts due to the operator: –

- Direct method – the APP Provider provides the breakup of the PTO amounts (within the QR Ticket) to the PSP and the PSP credits those accordingly. This is the method shown here in this section.
- Indirect method – Since the TG is front-facing to the respective operators so at the end-of-day every day, the APP Provider must credit the TG with the ‘Total Amount’ of all QRs generated by the TG on the day. This method is not within the scope of this document.

Details of message exchanges in direct method are discussed in this section.

The same example used throughout [Section 4.3](#) is being followed in this section as well. We have already described the process and the dataflows in the example given in [Section 4.3.1.3](#). To reiterate it again here, in the example the customer bought 3 tickets –

- Total value of service, Total Amount → ₹300/-

- 2 tickets for Operator ID 10 worth ₹180/-
- 1 ticket for Operator ID 135 worth ₹120/-
- We assumed that both APP Provider and the TG have 1 MDR and 1 Corporate Account each as shown below.
  - *APP\_Prov\_15\_MDR\_Acc (No: 3344 last 4 digits)*
  - *TG\_23\_MDR\_Acc (No: 1729 last 4 digits)*
- We also assumed that both the operators – PTO ID 10 and PTO ID 135 also have 1 Corporate Account each as shown below.
  - *PTO\_10\_Corp\_Acc (No: 9187 last 4 digits)*
  - *PTO\_135\_Corp\_Acc (No: 6528 last 4 digits)*
- For sake for completeness we assumed that the PSP institution ID is **78**.

We shall now describe the financial process flow from the PSP to the APP Provider to the TG.

#### 5.2.1.1. PSP Settles Amount with Operators

- i. The APP Provider and TG have a trust relationship. So the APP Provider must have received directions from the TG to credit the MDR Account of the TG and also the Corporate Accounts of the Operators. This information must be further provided by the APP Provider to the PSP which is associated with it.
- ii. Sometime during the day, the amount the customer paid using the online payment facility given by the PSP will be transferred into the PSP's pooling account.
- iii. PSP credits amounts from previous day on the next business day (T+1) as per agreement between APP Provider and PSP. The amounts are credited to the Accounts of the PTOs, and the MDR accounts of the Merchant (APP Provider) and TG respectively. The breakup of these payment distributions and acknowledgement are already provided in [Section 4.3.1.3 – Figure 15](#).
- iv. After the PSP credits all the accounts on T+1, the PSP will then send the MIS Report of executing this process to the APP Provider. The MIS report shall have filename *<AggregatorName>\_<MO\_No>\_DDMMYYYYHHMM.JSON*. There shall be one report for every Merchant Order that was fulfilled that day. The Aggregator Name and the Merchant Order Number defined in [Section 4.3.1.3](#) is PSPEPAY and MO-1 respectively. Accordingly, the filename becomes PSPEPAY\_MO-1\_130620200115.JSON.

*All communication mechanism between PSP and APP Provider must use HTTPS protocol. Furthermore, the APP Provider and PSP may choose to use an encryption/decryption mechanism for enhanced security.*

Table below shows the MIS report structure. Note that the data types are not mentioned here as all the fields that appear here have already been described earlier in [Section 4.3.1.3](#).

**Table 5.11: PSP to APP Provider MIS Report Structure**

MIS Report				
Element		Field Name		Presence
Unique Filename		MIS_Rpt_Name		M
Merchant Identifier		Merchant_Id		M
APP Provider Identifier		APP_Id		M
Merchant Identifier		Merchant_Id		M
Report Direction*		PSP-2-APP (This is a constant string)		M
Settlement Transaction Details	APP Provider Settlement details	App_Acc_Num		M
		App_Txn_Ref_Num		M
		App_Payout_Amount		M
		App_Txn_Date		M
	PTO Settlement details (may be multiple PTOs)	PTO_Acc_Num		M
		PTO_Txn_Ref_Num		M
		PTO_Payout_Amount		M
		PTO_Txn_Date		M
		Net_Deducted_Amount (default 0)		M
		Refund Info records (Optional Element. It is only present if there is a refund against the PTO. There may be multiple records)	Refund_Id (default is 0)	O
			Refund_Amount (default is 0)	O
			Refund_Date	O
		TG_Acc_Num		M

	TG Settlement details	TG_Txn_Ref_Num	M
		TG_Payout_Amount	M
		TG_Txn_Date	M
Customer TXN records (may be multiple)	Merchant Order Number	Merchant_Order_Num	M
	TransactionRef Number	TXN_Ref_Num	M
	Transactiondate and time	TXN_Date	M
	Transaction Amount	TXN_Amount	M
	Payment Mode	Payment_Mode	M
	Other Details	PSP_ID;APP_ID;TG_ID; PTO_ID (may be multiple)	M

\*Since reports are moved around across multiple entities, we introduce the concept of Message Direction to show the traceability of a report from its origin to its destination. The general notation is <Origin Entity Abbreviation>-2-<Destination Entity Abbreviation>. For example –

- PSP-2-APP would mean report from the PSP to the APP Provider,
- APP-2-TG would mean report from the APP Provider to the TG,
- TG-2-PTO would mean report from the TG to the PTO, etc.

Many PSPs use the term ‘Settlement Report’ instead of ‘MIS Report’. We shall use both the terms interchangeably.

The Settlement Report for the only Transaction that has been used throughout this document is now shown in [Figure 45](#) below. Also please note that it is not a mandatory requirement to implement systems to exchange MIS Reports in JSON format.

```

"PSP_MIS_Report": {
  "APP_Id": "15",
  "PSP_Id": "78",
  "MIS_Rpt_Name": "PSPEPAY_MO-1_130620200115",
  "Report_Direction": "PSP-2-APP",
  "Merchant_Id": "Merchant15",
  "Multi_Account_Settlement_Info": {
    "APP_Prov_15_MDR_Acc": {
      "App_Acc_Num": "xxxxxxxx3345",
      "App_Txn_Num": "PSPORC20200613176790",
      "App_Payout_Amount": "3",
      "App_Txt_Date": "13/06/2020@01-15-19"
    },
    "TG_23_MDR_Acc": {
      "TG_Acc_Num": "xxxxxxxx1729",
      "TG_Txn_Num": "PSPORC20200613176794",
      "TG_Payout_Amount": "3",
      "TG_Txn_Date": "13/06/2020@01-15-20"
    },
    "PTO_Accounts_Settlement_Info": {
      "PTO_10_Corp_Acc": {
        "PTO_Acc_Num": "xxxxxxxx9187",
        "PTO_Txn_Num": "PSPORC2020061317264670",
        "PTO_Payout_Amount": "176.4",
        "PTO_Txn_Date": "13/06/2020@01-15-21",
        "Net_Deducted_Amount": "0"
      },
      "PTO_135_Corp_Acc": {
        "PTO_Acc_Num": "xxxxxxxx6528",
        "PTO_Txn_Num": "PSPORC2020061317264679",
        "PTO_Payout_Amount": "117.6",
        "PTO_Txn_Date": "13/06/2020@01-15-22",
        "Net_Deducted_Amount": "0"
      }
    }
  },
  "Customer_Txn_Info": {
    "Txn_Record_1": {
      "Merchant_Order_Num": "MO-1",
      "TXN_Ref_Num": "ABCDRC202006126782341",
      "TXN_Amount": "300.00",
      "TXN_Date": "12/06/2020@18-47-57",
      "Payment_Mode": "IMPS",
      "Other_Details": "78;15;23;9201345678;10;135"
    }
  }
}

```

Figure 45: MIS Report – PSP to APP Provider

#### 5.2.1.2. APP Provider sends MIS report to TG

- i. After the APP Provider receives the MIS report it must now send it to the TG.
- ii. The MIS Report that APP Provider sends to the TG is the same as the one that the PSP sends to the APP Provider. But since one APP Provider can be associated with multiple TGs hence the APP Provider will only send the report for that particular TG. In our example, the APP Provider is only associated with one TG – TG ID 23.
- iii. Furthermore, the PSP settles the full amounts in multiple accounts separately and sends one consolidated MIS report. The APP Provider sends the consolidated MIS report to the TG having all the transaction records in one file.
- iv. The MIS report shall have filename  
<MIS>DDMMYYYYHHMM\_<AppProviderID>\_<TGID>\_<PSP\_ID>.JSON.

**Note:** *The communication mechanism between PSP and APP Provider should be the HTTPS protocol.*

```
{
  "APP_Id": "15",
  "TG_Id": "23",
  "PSP_Id": "78",
  "MIS_Rpt_Name": "MIS130620200117_15_23_78",
  "Report_Direction": "APP-2-TG",
  "PSP_MIS_Report": {
    "MIS_Rpt_Name": "PSPEPAY MO-1_130620200115",
    "Merchant_Id": "Merchant15",
    "Multi_Account_Settlement_Info": {
      "APP_Prov_15_MDR_Acc": {
        "App_Acc_Num": "xxxxxxxx3345",
        "App_Txn_Num": "PSPORC20200613176790",
        "App_Payout_Amount": "3",
        "App_Txn_Date": "13/06/2020@01-15-19"
      },
      "TG_23_MDR_Acc": {
        "TG_Acc_Num": "xxxxxxxx1729",
        "TG_Txn_Num": "PSPORC20200613176794",
        "TG_Payout_Amount": "3",
        "TG_Txn_Date": "13/06/2020@01-15-20"
      },
      "PTO_Accounts_Settlement_Info": {
        "PTO_10_Corp_Acc": {
          "PTO_Acc_Num": "xxxxxxxx9187",
          "PTO_Txn_Num": "PSPORC2020061317264670",
          "PTO_Payout_Amount": "176.4",
          "PTO_Txn_Date": "13/06/2020@01-15-21",
          "Net_Deducted_Amount": "0"
        },
        "PTO_135_Corp_Acc": {
          "PTO_Acc_Num": "xxxxxxxx6528",
          "PTO_Txn_Num": "PSPORC2020061317264679",
          "PTO_Payout_Amount": "117.6",
          "PTO_Txn_Date": "13/06/2020@01-15-22",
          "Net_Deducted_Amount": "0"
        }
      }
    },
    "Customer_Txn_Info": {
      "No_of_Txns": "1",
      "Txn_Record_1": {
        "Merchant_Order_Num": "MO-1",
        "TXN_Ref_Num": "ABCDRC202006126782341",
        "Txn_Amount": "300.00",
        "TXN_Date": "12/06/2020@18-47-57",
        "Payment_Mode": "IMPS",
        "Other_Details": "78;15;23;9201345678;10;135"
      }
    }
  }
}
```

**Figure 46: MIS Report – APP Provider to TG**

Note: - In the above JSON file, the compound fields “PTO\_Accounts\_Settlement\_Info” and “Customer\_Txn\_Info” can be repeated multiple times. Since a QR Ticket can have one ticket each for ten different operators, so there may be up to ten PTO account credit information in the “PTO\_Accounts\_Settlement\_Info” field. There is no limit to the number of customer transactions in the “Customer\_Txn\_Info”.

### 5.2.2. TG sends Payment Details to AFC

When the TG receives the MIS report from the APP Provider it must then send individual payment information to each Operator. Shown below are two JSON files – one each for PTO ID 10 and PTO ID 135 respectively.

This is sending proof of payments to the operators. The process is similar to the process of exchanging Purchase QR information between the TG and AFC as described in [Section 4.3.2](#) or the Policy related information exchange described in the [Section 5.1](#).

The file naming convention followed for payments related information of financial QRs is [qrf\\_DDMMYYYYHHMMSS\\_TGID\\_PTOD.JSON](#). Assuming that the scheduler sends payment information to AFC every at 02:01 hours, the MIS report file name for PTO ID 10 will be [qrf\\_13062020020103\\_23\\_10.JSON](#) and for PTO ID 135 will be [qrf\\_13062020020103\\_23\\_135.JSON](#). Note that the date has rolled over to the next day (T+1).

The generic format in which the TG creates Operator-specific financial transaction JSON objects is depicted below. There may be multiple Customer transactions but only a single payment section dedicated for the PTO. Also TG must send the Operator details in the payload.



```

"Payload_Data": {
  "MIS_Rpt_Name": "qrf_DDMYYYYYhhmmss_TGID_PToid",
  "Message_Direction": "TG-2-PTO",
  "Operator_Details": {
    "OperatorId": "Operator_ID",
    "OperatorName": "OperatorName",
    "OperationType": "0",
  },
  "Customer_Txn_Info": {
    "No_Of_Txns": "N",
    "Txn_Record_1": {
      "Merchant_Order_Num": "MO-1",
      "TXN_Ref_Num": "ABCD...1",
      "TXN_Amount": "xxx",
      "TXN_Date": "DD/MM/YYYY@hh-mm-ss",
      "Payment_Mode": "XXXX",
      "Tkt_Sl_No": "12062020...444"
    },
    ...
    "Txn_Record_N": {
      "Merchant_Order_Num": "MO-N",
      "TXN_Ref_Num": "ABCD...N",
      "TXN_Amount": "xxx",
      "TXN_Date": "DD/MM/YYYY@hh-mm-ss",
      "Payment_Mode": "XXXX",
      "Tkt_Sl_No": "12062020...999"
    },
  },
  "PTO_Accounts_Settlement_Info": {
    "PTO_Acc_Num": "xxxxxxxx9187",
    "PTO_Txn_Num": "PSPORC202006...70",
    "PTO_Payout_Amount": "NNN",
    "PTO_Txn_Date": "DD/MM/YYYY@hh-mm-ss",
    "Net_Deducted_Amount": "x"
  }
}

```

Figure 47: Generic Message Format for Payment Settlement – TG to AFC

Consistent with the example used throughout this document, the records for Operator ID = 10 in JSON format is as follows:

```
"Payload_Data": {
  "Operator_Details": {
    "OperatorId": "10",
    "OperatorName": "PTOName10",
    "OperationType": "0",
  },
  "MIS_Rpt_Name": "qrf_13062020020103_23_10",
  "Message_Direction": "TG-2-PTO",
  "Customer_Txn_Info": {
    "No_Of_Txns": "1",
    "Txn_Record_1": {
      "Merchant_Order_Num": "MO-1",
      "TXN_Ref_Num": "ABCDRC202006126782341",
      "TXN_Amount": "300.00",
      "TXN_Date": "12/06/2020@18-47-57",
      "Payment_Mode": "IMPS",
      "Tkt_Sl_No": "12062020184811M0000032446"
    }
  },
  "PTO_Accounts_Settlement_Info": {
    "PTO_Acc_Num": "xxxxxxxx9187",
    "PTO_Txn_Num": "PSPORC2020061317264670",
    "PTO_Payout_Amount": "176.4",
    "PTO_Txn_Date": "13/06/2020@01-15-21",
    "Net_Deducted_Amount": "0"
  }
}
```

**Figure 48: Financial Transaction Payload Data – Operator ID 10**

Similarly, for Operator ID = 135,

```
"Payload_Data": {
  "Operator_Details": {
    "OperatorId": "135",
    "OperatorName": "PTOName135",
    "OperationType": "1",
  },
  "MIS_Rpt_Name": "qrf_13062020020103_23_135",
  "Report_Direction": "TG-2-PTO",
  "Customer_Txn_Info": {
    "No_Of_Txns": "1",
    "Txn_Record_1": {
      "Merchant_Order_Num": "MO-1",
      "TXN_Ref_Num": "ABCDRC202006126782341",
      "TXN_Amount": "300.00",
      "TXN_Date": "12/06/2020@18-47-57",
      "Payment_Mode": "IMPS",
      "Tkt_Sl_No": "12062020184811M0000032446"
    }
  },
  "PTO_Accounts_Settlement_Info": {
    "PTO_Acc_Num": "xxxxxxxx6528",
    "PTO_Txn_Num": "PSPORC2020061317264679",
    "PTO_Payout_Amount": "117.6",
    "PTO_Txn_Date": "13/06/2020@01-15-22",
    "Net_Deducted_Amount": "0"
  }
}
```

**Figure 49: Financial Transaction Payload Data – Operator ID 135**

The data exchange protocol is consistent with the Purchase QR and Policy QR messages described so far, but two new message keys are introduced for these transactions – one each for request and response.

**Table 5.12: Financial Transaction Request Message Format– TG to AFC**

QR Message Element	Size (bytes)	Data Type	Description
Hash Token	64	AS	SHA256 Hash of the Message Payload i.e. the Message Key and Payload Data
Message Key	2	NS	31 – QR_FIN_REPORT_REQUEST Key ID = 31
Payload Data	Variable	AN	Contents like the JSON files shown in <a href="#">Figure 48</a> and <a href="#">Figure 49</a> . Contents may be encrypted or sent over some secure tunnel as agreed between the participating entities.

### **Financial Transaction Request for PTO ID 10 and PTO ID 135**

Financial Report Request for PTO ID 10:

```
"QR_Fin_Report_Request": {
  "Hash-Token": {
    "Hash_Value":
      "726beed9f2592d2ba8927834947cdb6518f1ad678da0a51f034ac1e7530
     blad9"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "31"
    },
    "Payload_Data": "As shown in Figure 48"
  }
}
```

**Figure 50: MIS Report for Operator ID 10 – qrf\_13062020020103\_23\_10.JSON**

Financial Report Request for PTO ID 135:

```
"QR_Fin_Report_Request": {
  "Hash-Token": {
    "Hash_Value":
      "e4c92bcf4649721602990fdfa2dbd6fc480153a2f409780c5ce9d6b6db0
     003d6"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "31"
    },
    "Payload_Data": "As shown in Figure 49"
  }
}
```

**Figure 51: MIS Report for Operator ID 135 – qrf\_13062020020103\_23\_135.JSON**

### 5.2.3. AFC Acknowledges PTO Payment Details sent by TG

Here, we first describe the generic QR System Message structure for responses to financial report transactions.

**Table 5.13: Financial Transaction Response Message Format – AFC to TG**

QR Message Element	Size (bytes)	Data Type	Size (bytes)
Hash Token	64	AS	SHA256 Hash of the Message Payload i.e. the Message Key and Payload Data
Message Key	2	NS	32 – QR_FIN_REPORT_RESPONSE Key ID = 32
Payload Data	Variable	AN	Filename = Name of the file sent as a field inside the Purchase QR Request.
			Ack = String “ACK-N”, where N is value of the No_Of_Records field in the request. In this case, it will be the number of Customer transactions.
			TG_ID = ID of the TG (in all examples in this document we have used the value 23)
			Operator ID = ID of the Operator (in all examples in this document we have used two values for Operator/PTO ID, 10 and 135)
			ErrorMsg = Can be either the string “SUCCESS” by default for no error or some other elaborate message chosen by the TG as shown in Table 5.4: Update Policy Response Error Codes and Messages  . When the TG receives an ACK that has an ErrorMsg other than SUCCESS, it should try to resend the message again.

### Financial Transaction Response for PTO ID 10 and PTO ID 135

The AFC should compute the message response as shown in the diagrams below.

Operator ID 10:

```

"QR_Fin_Report_Response": {
  "Hash_Token": {
    "Hash_Value":
      "ffc267e1a2d0925da6e269a0f33ca737a65bea91b2de4048f71d77ebf5
      bec2d7"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "32"
    },
    "Payload_Data": {
      "Filename": "grf_13062020020103_23_10",
      "Ack": "ACK-1",
      "TG_ID": "23",
      "OpID": "10",
      "ErrorMsg": "SUCCESS"
    }
  }
}

```

Figure 52: Response from Operator ID 10 for MIS Report

Operator ID 135:

```

"QR_Fin_Report_Response": {
  "Hash_Token": {
    "Hash_Value":
      "45bcfeb2205d408fbad2670c1d09b7dbd0930919d400aaf01ca95bd17d
      2dab42"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "32"
    },
    "Payload_Data": {
      "Filename": "grf_13062020020103_23_135",
      "Ack": "ACK-1",
      "TG_ID": "23",
      "OpID": "135",
      "ErrorMsg": "SUCCESS"
    }
  }
}

```

Figure 53: Response from Operator ID 135 for MIS Report

#### 5.2.4. Reconciliation of Missed / Delayed Payments

- It may so happen that the payment one customer made, does not come back to the PSP on time or due to some problems at the customer's side.
- So in such cases, the PSP must send a new MIS report to the merchant, in our case the APP Provider, mentioning the TXN\_Ref\_No, DateTime of Txn, and the Amount missing. The APP Provider must then forward this MIS to the TG.
- The TG must similarly inform the affected PTOs in a separate MIS report about these missed payments.
- Just as missed/delayed payments are sent as separate MIS report, similarly recovered payments must also be shared and exchanged in separate MIS reports.

#### 5.2.5. Refund and Ticket Cancellation

Processing and settling Refunds is a complicated process and hence it is recommended that the TG, APP Provider (Merchant) and PSP (in this case an Aggregator) together determine a workable and comprehensive procedure to process them. Here we list down such a procedure that may be followed by concerned entities/stakeholders. Please note that here the terms APP Provider and Merchant are used interchangeably. Also the Aggregator Name for PSP is given as 'PSPEPAY' in [Table 4.8](#).

- a) Customer initiates refund on the Merchant Portal or Mobile APP. Tickets cancellation period, affirmation and number of days required for settlement is solely the discretion of the PTO and its business rules.
- b) Merchant (APP) sends to TG and TG sends Cancellation/Refund MIS to Operator/PTO.
- c) Operator(s) gives acknowledgement for Refund/Cancellation to PTO which in turn sends it to Merchant.
- d) The merchant sends a 'Refund Request' to PSP. Customer should be shown a status like 'Refund Booked – Settlement will be processed in at least 4 days or some acceptable time period.
- e) PSP internally settles amount to the Customer account.
- f) On T+1 after here i.e. the next time PSP processes all these Multi-account credits into the Operator accounts, it credits the 'Net Amount Less Refund Amount (which came in Refund Request)' into their account after all the agreed MDR amounts have been deducted. See [Figure 48](#) and [Figure 49](#). Figure 49 The MIS reporting cycle is the same as normal MIS reporting described in [Sections 5.2 and 5.3](#) i.e. PSP → APP Provider → TG → PTO.

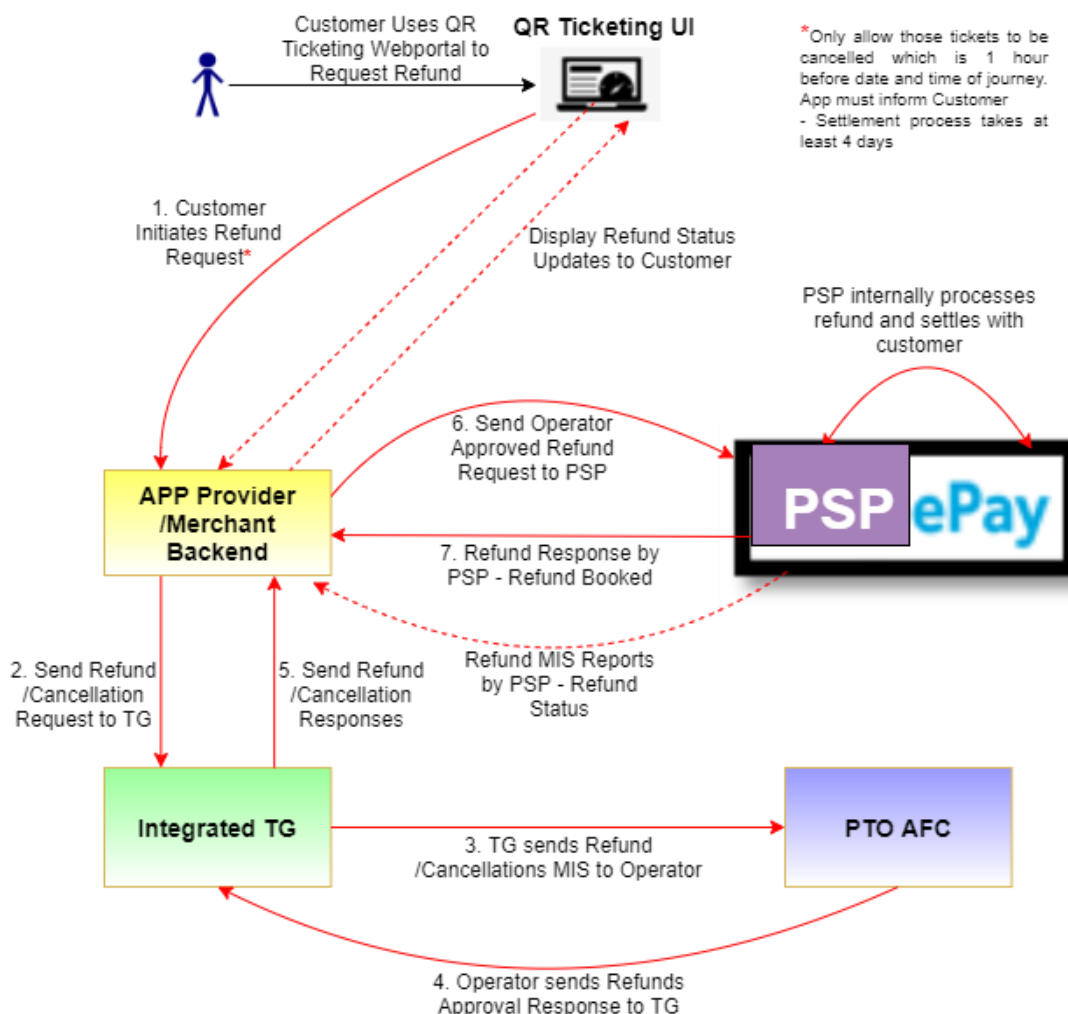
As seen here if the PTO deems the cancellation request as valid then the reverse transitive payment sequence must be followed from PTO → TG → APP Provider → PSP → Customer.

## QR Ticketing System – Interface Specification

It must also be noted that since the QR supports multiple tickets (up to 10), it may be possible to make partial cancellation of only a certain number of tickets. However, at the present moment, partial cancellation must not be allowed. Cancellation should only be allowed for the entire QR Ticket.

*All modes of communication here is HTTPS.*

The algorithm listed above is shown pictorially in the figure below,



**Figure 54: Workflow for Refunds and Cancellation**



### 5.3. TG – APP Provider Interface

The APP Provider is the developer of the Mobile or Webclient Application that the consumer can download from the Google Play Store or such other app stores (or in case of Webclient, launch a URL), and use it to buy QR tickets. As seen from the architectures depicted in [Figure 2](#) and [Figure 3](#), the Application needs to establish a trust relationship with TG. For establishing this trust, the process is the same as the 'chain of trust' between the Operator/PTO and TG described in Section 5.1 of QR Specifications – Part II. The APP Provider sends its Public Key Certificate to the TG which the TG utilizes to verify signed content during PTO ID request or Fetch Fare details, etc.

#### **APP Provider-TG is a logical 1:1 relationship:**

One APP Provider can register itself with any number of TG systems. Similarly, one TG can be associated with many APP Providers. In that sense, the cardinality of TG:APP Provider is a M:N relationship. However, when an Application (whether Mobile APP or Webclient) requests for a QR Ticket that has tickets for different operators, in that case, the single QR must always be generated on the same TG. It is not possible to have the same QR Ticket generated and signed at multiple TGs. Therefore, in that sense, the logical cardinality of TG:APP Provider concerning QR Ticket request and response is a 1:1 relationship.

**Table 5.14: APP Provider Details Structure in Policy DB**

Sl. No	Field Name	Description	Field Presence	Data Type / Length
1	APP_ID	The APP_ID is a distinct ID given to the APP Provider. This field has also been referred to as the 'Requester ID' throughout the specifications.	M	NS / 5
2	APP Provider Name & Address	Details of the establishment. It is the name and address of the APP Provider.	M	AN / 1000
3	APP_Pub_Key	Certified Public Key that the App Provider must provide to the TG to enable it to verify signed content.	M	B64S / 2048
4	PTO ID	Repeating field – containing lists of all PTO IDs. The PTO IDs are the identifiers of the PTOs that this TG is associated with.	M	NS / 5

## QR Ticketing System – Interface Specification

5	PTO Name	Repeating field – containing lists of all PTO Names. The PTO Names are the brand/company names of the PTOs that this TG is associated with.	M	AN / 100
6	PTO Logo	Repeating field – containing the official logos of all PTOs that this TG is associated with.	O	Binary Data - BLOB

The handshake between the Mobile App itself and the TG is done as follows:

- The trust relationship between the TG and App Provider ensures that the TG has provided the list of PTO ID, PTO Name and optionally the PTO Logo to the APP Provider. Therefore, when the User launches the Mobile App or visits to the Webclient URL, he/she will be shown the list of PTOs for which tickets can be purchased.
- When the application sends the Fetch Route/Station request, it must sign with the APP Provider's private key and send the request to TG.
- The TG will use the APP Provider's public key to verify the signature and ascertain its genuineness.

The information exchange between the TG and the APP Provider can take place in the same manner as the TG and AFC described in [Sections 4.3.2](#) and [5.1](#). we describe the message format briefly here.

**Table 5.15: Structure for App Registration Request Message with TG**

Element	Size (bytes)	Data Type	Size (bytes)
Hash Token	64	AS	64-byte Hash of the Message Payload (Message Key and Payload Data). This is for integrity checking. SHA-256 Hash produces a 64-byte Hex string and is considered to be a good hashing algorithm and should be used in this case.
Message Key	2	NS	<a href="#">41 – QR_APP_REG_REQUEST</a>
Payload Data	Up to 2048	B64S	The App ID, the TG ID, Name and Address of the App Provider signed by the App Provider's Private Key. The TG must be able to retrieve the details using the App Provider's pre-shared Public Key. The TG and App Provider must share their IDs and public keys beforehand either through email or some other agreed mechanism.

			A sample Registration request payload data in unsigned form is shown below in Figure 55. This should be encrypted with the APP Provider's own Private key and then encoded in Base 64 before it is sent to the TG.
--	--	--	--

```

"Payload_Data": {
  "TG_Id": "23",
  "App_Provider_Details": {
    "App_Id": "15",
    "App_Provider_Name": "AppName15",
    "App_Provider_Address": "0309 Street, Pin-110091"
  }
}

```

**Figure 55: Payload Data of App Registration Request –App Provider to TG**

The TG decrypts the payload and then send the response in a structure as follows.

**Table 5.16: Structure for App Registration Response Message with TG**

QR Message Payload Element	Size (bytes)	Data Type	Size (bytes)
Hash Token	64	AS	64-byte Hash of the Message Payload (Message Key and Payload Data). This is for integrity checking. SHA-256 Hash produces a 64-byte Hex string and is considered to be a good hashing algorithm and should be used in this case.
Message Key	2	NS	<a href="#">42 – QR_APP_REG_RESPONSE</a>
Payload Data	Variable	AN	Contents like the JSON files shown in <a href="#">Figure 56</a> .

```

"Payload_Data": {
  "App_Id": "15",
  "TG_Id": "23",
  "PTO_Records": {
    "No_Of_Records": "2",
    "Record_1": {
      "OperatorId": "10",
      "OperatorName": "PTOName10",
      "OperatorLogo": ""
    },
    "Record_1": {
      "OperatorId": "135",
      "OperatorName": "PTOName135",
      "OperatorLogo": ""
    }
  }
}

```

**Figure 56: Payload Data of App Registration Response – TG to App Provider**

*Messages, shown in the figures above, exchanged between the TG and APP Provider may be done using TCP/IP socket implementations or HTTPS. Ideally, the TG must provide a registration portal for all the App Providers to register/associate with it.*

## 5.4. TG – TOM Interface

Every PTO usually has several counters (automatic or manual) to provide service to customers enabling them purchase on-the-fly tickets for immediate use. These tickets have been termed as the Paper QR Tickets throughout the QR Specifications.

The TG – TOM interface is similar to the *Mobile/Webclient App – TG interface* except that the integration with the PSP with the TOM application is not required. This is because the PTO collects the money from the commuter directly – usually either in the form of Cash or Card purchase. The TOM Application itself should very much resemble the Webclient application that people use to buy QR Tickets online. However, the PSP Interface is not required.

### 5.4.1. TG-TOM Workflow – Ticket Request and Response

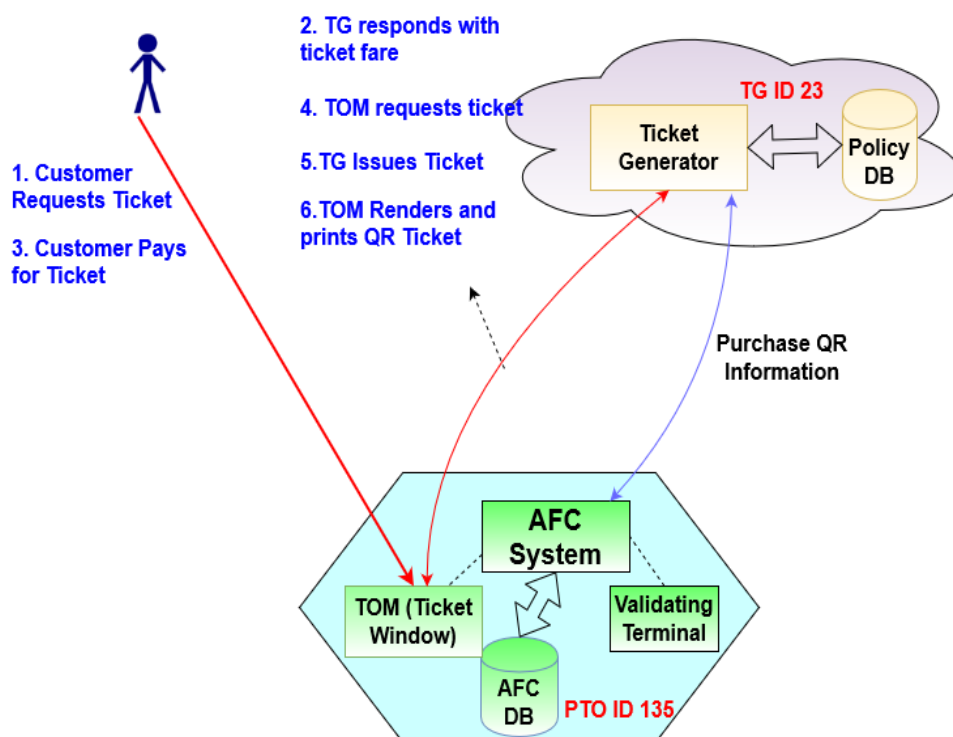


Figure 57: TOM – TG Interface

TOM tickets are always issued for one Operator/PTO only and cannot be issued for a different PTO. Furthermore, the tickets are always day tickets i.e. they are meant to be used on the same day only.

Another important point that must be noted here is that the TG can only issue tickets requested by TOMs whose IDs have already been made available on the Policy DB (Refer [Table 5.1](#)).

The PTO may also implement rules to allow some interval between issuing a Paper Ticket and the actual activation datetime of the Ticket. This interval should give enough time for the Purchase QR information to flow to the PTO's AFC systems and further trickle down to the Validating terminals, if required.

*The mode of communication between the TOM and TG is HTTPS.*

*For a detailed design of the TOM application please refer to the Implementation Guidelines V 1.0.*

### 5.4.2. Payment Reconciliation for Paper QR Tickets

As mentioned earlier, PTO collects money directly from Commuter for Paper Tickets. But it has to pay the TG for using its services. So in case of Paper Tickets, there is a reverse money flow (service charges) from PTO → TG. For this, the account information of the TG must be provided to all the PTOs associated with it. A suitable payment procedure for Paper QR Tickets is shown below:

- a. At T+1 every business day, TG should send a separate report to PTO with details on all the Paper Tickets purchased during the previous day. The interface followed for this is exactly the same as for other Application tickets shown in [Section 5.2.2](#). A sample JSON file for this request is shown below in [Figure 58](#). Please note that the Ticket Serial Numbers of all Paper QR Tickets have a 'T' which means the Requesting Source of the QR Ticket was a TOM client as specified in the Terms and Definitions section of QR Specifications – Part I.

```

"Payload_Data": {
  "MIS_Rpt_Name": "qrfp_130620200210_23_135",
  "Message_Direction": "TG-2-PTO",
  "Net_Amount": "202.0",
  "Total_TXN_Amount": "200",
  "Total_MDR_Payable": "2.0",
  "Records": {
    "No_Of_Records": "3",
    "Record_1": {
      "Tkt_Sl_No": "12062020124815T0000030644",
      "NoOfTickets": "1",
      "Tkt_Amount": "45",
      "MDR_Payable": "0.45"
    },
    "Record_2": {
      "Tkt_Sl_No": "12062020163522T0000031873",
      "NoOfTickets": "1",
      "Tkt_Amount": "75",
      "MDR_Payable": "0.75"
    },
    "Record_3": {
      "Tkt_Sl_No": "12062020180601T0000032002",
      "NoOfTickets": "1",
      "Tkt_Amount": "80",
      "MDR_Payable": "0.8"
    }
  }
}

```

**Figure 58: Payload Data for Paper QR MIS Report Request – TG to AFC**

- b. The PTO must settle the payment with the TG through a direct account-to-account transfer.
- c. When the PTO sends the acknowledgement for the MIS Report sent by the TG, it must also send the TXN\_Ref\_No and the Amount settled against the MIS\_Rpt\_Name from the previous day. This is to say that the payment for the latest report shall always be pending till the next day. A sample message is shown below in JSON format. The PTO ID 135 is assumed here for the illustration.

```

"Payload_Data": {
  "Ack": "ACK-5",
  "Filename": "qrfp_120620200210_23_135",
  "Message_Direction": "PTO-2-TG",
  "Net_Amount": "202.0",
  "Total_TXN_Amount": "200",
  "Total_MDR_Payable": "2.0",
  "No_Of_Txns": "3",
  "TXN_Ref_No": "xxxxxxxxx22456",
  "Total_MDR_Paid": "2.0",
  "TXN_Date": "13/06/2020@19-30-05",
  "ErrorMsg": "SUCCESS"
}

```

**Figure 59: Payload Data for Paper QR MIS Report Response – AFC to TG**

The protocol followed for these messages shall be the same as described in [Sections 5.2.2](#) and [5.2.3](#). For the sake of clarity, a new pair of Message Keys needs to be introduced. The structure of the message is as follows.

**Table 5.17: Message Format for Paper QR MIS Report Request & Response**

QR Message Payload Element	Size (bytes)	Data Type	Size (bytes)
Hash Token	64	AS	64-byte Hash of the Message Payload (Message Key and Payload Data). This is for integrity checking.
Message Key	2	NS	51 – QR_TOM_REPORT_REQUEST 52 – QR_TOM_REPORT_RESPONSE
Payload Data	Variable	AN	Contents like the JSON files shown in <a href="#">Figure 58</a> and <a href="#">Figure 59</a> .

**A Note about TOM & APP Provider:** - It is worth mentioning here that the TOM application does not operate under the APP Provider model – Webclient and Mobile APP. It would have to be either developed or integrated with the PTO's existing Software clients, if any, or deployed as a separate Application Software Interface integrated with the PTO's network and AFC backend.

## 6. References

S. No	References
1	“QR Specifications – Part I” – QR Code Dataset
2	“QR Specifications – Part II” –QR Security Scheme



## Appendix – I: QR Validation & Customer Care

### Various Scenarios of QR Validation and Customer Care Workflow

#### Customer Care Workflow

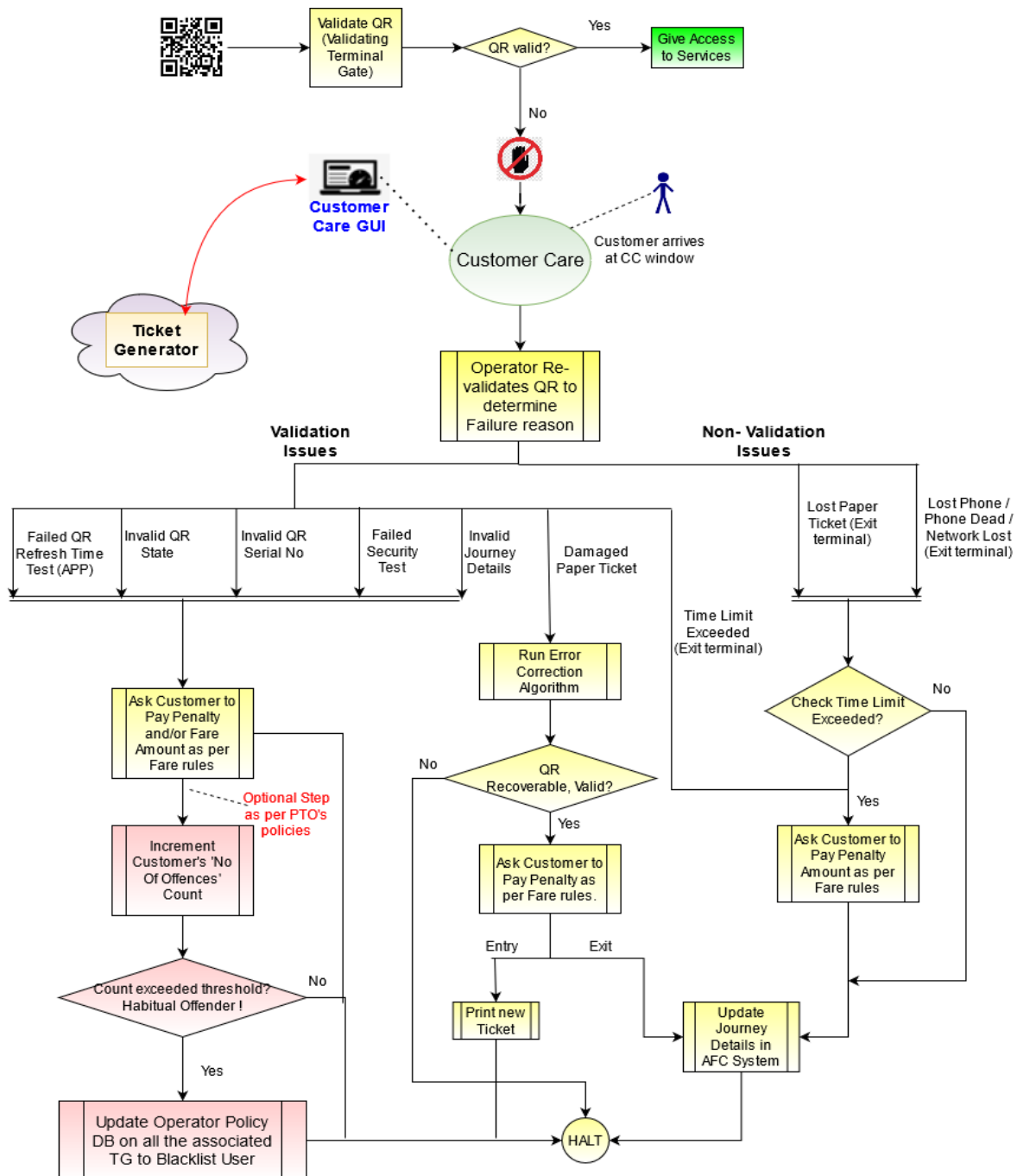


Figure 60: Customer Care Workflow Diagram

### QR Validation Rules

For a QR to be asserted as valid for entry or exit, the QR data must be put through a certain number of tests. These tests or rules are the necessary and sufficient conditions for a QR to be asserted as valid. Unless explicitly specified these rules apply to both paper and mobile QR codes. If and only if all the conditions are satisfied, the QR shall be considered valid. Failure of any of the conditions shall deem the QR as invalid. These rules are as follows:

- a. **Security Validation Test:** Check if the QR passes the security conditions – (i) signature is valid; meaning the source of the data is genuine. (ii) The QR passes the integrity test (matching hashes); meaning its content has remained intact since it was created.
- b. **Serial Number Test:** Check if the serial number of the QR is a valid supplied one or not. This test can serve as an effective measure to counter fake QR codes.
- c. **Refresh Time Test:** This only applies to QR's scanned from phones. Refer to [Appendix – III: Futuristic QR Ticketing System](#) for details related to QR Update Time in Dynamic Data element. This is an optional feature but can serve as an effective measure to counter fake QR codes.
- d. **Exit Time Test:** Maybe applied on exit terminals. Some Metro Operators implement a business rule whereby customers that have de-boarded at the destination station must leave the station premises within the next 20 minutes of de-boarding at the station. This is a hard rule to implement because the time of arrival of the Metro Car is not available within the terminals. However, with some careful planning, the duration of the journey from source to destination can be calculated and is usually a known quantity. So from the time of entry (QR Entry) to the current time, a good estimate can be made of the time spent between completion of journey and the attempt to exit the station. Furthermore, The Dynamic Data Element of the QR Code Dataset consists of the Operator-specific sub-branch of 19 bytes. Some operators may choose to use this field in order to implement the Exit Time Test. But in most other cases, this test is not applicable as a general Business Rule test.
- e. **QR State Test:** Every QR has a Status field in Dynamic Data to indicate the state of the QR viz. active, inactive, entry, exit, etc. The state of the QR changes with time so this test checks at an entry or exit terminal, if the state of the QR does or does not correspond to an expected state. This test can also serve as an effective measure to counter fake QR codes.
- f. **Booking Location Test:** As seen in the example throughout this document, the APP sends the booking location to the TG. The TG encodes this on the Common Data structure of the QR, i.e. QR\_SVC. These location coordinates maybe effectively used to tackle misuse of the QR Ticketing System. With certain modifications in the flow of Fetch Routes to Fetch Fare ([Sections 4.3.1.1 & 4.3.1.2](#)) workflows, the APP may disallow the user if he/she is attempting to buy a ticket from a restricted access area (geo-fence), like from inside the Metro Rail. But even if the APP does allow the purchase, then the Transit Risk Management validations must fail because the algorithm will detect that the ticket was bought from inside the geo-fence.

## Scenarios for Paper or Mobile Application QR (Single ticket) at Entry: -

### Scenario 1: Normal Entry (Valid QR)

When the commuter displays his QR on his mobile APP or Paper over the scanner of the entry terminal, the gate opens only after the QR passes the

- i. Security Validation Test implying it has its integrity is intact and it has a valid signature of the TG mentioned in the QR.
- ii. Journey Validation test implying the date of journey, source station and destination station are correct. Date of Journey must be checked at both Entry and Exit, Source station must be checked at Entry and Destination Station must be checked at Exit.
- iii. Serial Number Test implying the serial number is a valid supplied one.
- iv. Refresh Time Test (same as Dynamic Data → QR Update Datetime) implying the QR rendered on the screen is fresh and not a stale copy of the original. *This test is not applicable to Paper QR tickets.*
- v. QR State Test implying the state of the QR matches the expected state.

### Scenario 2: Disallow Entry (Invalid QR – Wrong Journey Details)

When the commuter displays his QR on his mobile APP (or Paper QR Ticket) over the scanner of the entry terminal, the gate does not open as the QR fails the

- Journey Validation Test either because
  - a. the date of journey does not match
 Or,
  - b. source and destination station codes do not match the validation terminal station code

In any case, the commuter is directed to go to Customer Care. The Customer Care operator makes note of the User's mobile number and increments the 'No of offences' count against it. If the count exceeds a pre-determined threshold, then he/she can take the decision to 'Denylist' the mobile number at the TG. Denylisting is an optional feature that is entirely at the discretion of the PTO's own business rules.

*Note: - The counter 'No of offences' is not implemented in the Policy DB shown in [Table 5.1](#), and has been used in this section only as a possible scenario.*

### Scenario 3: Disallow Entry (Invalid QR – Security issue)

When the commuter displays his QR on his mobile APP (or Paper QR Ticket) over the scanner of the entry terminal, the gate does not open as the QR fails the

- Security Validation Test either because
  - a. the validating terminal cannot determine if the signature it has been presented was actually created with the same private key that corresponds to the signer's public key
- Or,
- b. the data inside the QR code might have been compromised (unmatched hashes or unable to decrypt)

In any case, the commuter is directed to go to Customer Care. The Customer Care operator asks the user to pay the penalty amount and makes note of the User's mobile number and increments the 'No of offences' count against it. If the count exceeds a pre-determined threshold, then he/she can take the decision to 'Denylist' the mobile number at the TG.

### **Scenario 4: Disallow Entry (invalid QR – serial no. issue)**

When the commuter displays his QR on his mobile APP (or Paper QR Ticket) over the scanner of the entry terminal, the gate does not open as the QR fails the

- Serial Number Test implying the serial number data inside the QR code has been tampered and is not a valid supplied one.

The commuter is directed to go to Customer Care. The Customer Care operator asks the user to pay the penalty amount and makes note of the User's mobile number and increments the 'No of offences' count against it. If the count exceeds a pre-determined threshold, then he/she can take the decision to 'Denylist' the mobile number at the TG.

### **Scenario 5: Disallow Entry (Invalid QR – unexpected state)**

When the commuter displays his QR on his mobile APP over the scanner of the entry terminal, the gate does not open as the QR fails the

- QR Status Test implying the commuter is attempting to enter the station with either an expired ticket or on a day when the ticket is still 'Inactive'. This is also quite possible if the user has done 'tailgating' while entering the station and is now trying to exit the station.

The commuter is directed to go to Customer Care. If the Customer Care operator determines that the customer has not done anything wrong and the terminal was giving a wrong error, then the customer is not charged. Otherwise, the operator asks the user to pay the penalty amount and makes note of the User's mobile number and increments the 'No of offences' count against it. If the count exceeds a pre-determined threshold, then he/she can take the decision to 'Denylist' the mobile number at the TG.

**Scenario 6: Mobile phone hanged or battery Low/Dead or Network Lost at Exit**

When the commuter contacts the Customer Care and intimates about his/her phone hung or battery dead/ switched off or lost network connectivity, the operator will check the user's ticket in his system based on the customer's registered mobile number.

Assuming the search returns a positive response, the operator will be able to check the status of the ticket if it passes the– (a) Exit Time Test and (b) QR State Test

- i. If both the tests pass the operator will change the status of the QR for that particular PTO and allow the passenger to exit the station.
- ii. If either of the tests fails, then the commuter has to pay the penalty amount as per the PTO fare rules before he can exit the station.

**Scenario 7: Lost Paper QR at Exit**

In Paper QR Tickets a note is already printed at the bottom saying “Please keep it safe, until the journey ends”. So it is expected of all users to keep his/her QR safe until the completion of journey.

If at the exit terminal, the commuter intimates about a lost paper QR ticket, the commuter is directed to go to Customer Care. The operator will check the user's ticket in his system based on the customer's registered mobile number.

- The Commuter is asked to pay the penalty amount as per the PTO fare rules before he can exit the station.

*\*Please note that this scenario is only applicable at Exit terminals. The reason it should not be accommodated is because it encourages misuse. Since providing Mobile No is not mandatory for Paper QRs, it is not possible to validate a claim.*

**Scenario 8: Damaged Paper QR at Entry or Exit (Desirable feature)**

QR Codes have in-built error correction codes such that even if it is damaged, the full data can be recovered. But recovery can only happen if the damage is up to a maximum of 30%. The levels of % set up to 7, 15, 25 or 30 depending upon whether the QR has been encoded with levels L, Q, M or H respectively. The mode of QR is also only Paper QR.

*This is only a desirable feature as it involves a lot of overhead of decoding the QR with an additional error recovery algorithm. If at all it is used, it must not be employed at Entry/Exit validation terminals but only at Customer Care.*

If at an Entry or Exit terminal, the commuter presents a damaged QR Code the Validation should fail with the “Unknown or Damaged Error” message. The customer is asked to proceed to Customer Care. The customer care operator determines if the QR Ticket is a damaged QR and proceeds to scan the QR with the recovery option. The Customer Care validating terminal will attempt the following:

## QR Ticketing System – Interface Specification

- i. Up to 30% damage: If the terminal is able to decipher the code and the QR passes all the validation tests, then the customer care operator may proceed to print a new duplicate QR with the user's consent. If the terminal is not able to decipher the code, then the case will be treated in the same category as the next case below.
- ii. More than 30% damage: The terminal will not be able to validate the QR and thus disallow the customer from using it. The commuter must proceed to the Customer Care, pay a penalty and request for a duplicate ticket.

For both the above cases, when the customer agrees to print a new duplicate QR he/she must pay a penalty amount based on the PTO's fare rules.

### **Copied or Fake QR Tickets**

While on one hand QR codes have become ubiquitous around the world and find its application in almost each and every aspect of modern life, yet on the other hand copied or counterfeit QRs presents service providers with a humongous challenge on their hands. Because of the ease with which QR images can be copied, duplicated, reproduced and printed; service providers that provide QR code solutions to their customers are now trying to find more and more innovative methods to keep crooks at bay.

Paper based QR tickets pose the main problem for copied QR. Hence it highly recommended that Paper QRs are issued only –

- For the present day
- The Security Scheme is never left as disabled
- The QR Ticketing System is in sync with the AFC DB to tackle the problem effectively

### **Fake Paper QRs**

#### **Scenario 1: Fake Paper QR (Stop entry by Serial Number Test)**

Consider a scenario when a Customer has bought a single ticket; made multiple copies of it and distributed the fakes among his friends. The Serial Number Test can be an effective measure to deal with fake or copied QRs when used in conjunction with the AFC DB. The first commuter will pass the validation test and can proceed. But subsequently when a fake QR is presented at the terminal

- The terminal determines its serial number is already updated in the local AFC DB. It asserts the ticket as invalid and disallows entry.

#### **Scenario 2: Fake Paper QR(Stop exit by QR State Test)**

The scenario is only applicable for operators that support double-tap terminals i.e. ticket has to be scanned once at the entry and then again upon exit. If somehow a scammer manages to pass (or bypass) the serial number test by entering from a different terminal or before the serial number gets updated on the AFC DB (delayed syncing); then the QR State Test can come into action. When the AFC DB determines

that multiple entries of the same serial number are being updated then that serial number must be marked 'Invalid'. At the time of exit:

- The exit terminal must stop all the crooks from leaving the station without paying the penalty amount. The rogue customers are directed to proceed to Customer Care and pay the fare and penalty as per the PTO's fare rules.

### **Fake Mobile QR**

#### **Scenario 3: Fake Mobile Application QR (Refresh Time Test)**

It is quite unlikely that more than one user will be able to travel on the same ticket on their mobile phones. The QR mobile application should disable the screen shot feature. This is one feature that most present-day QR Apps tend to have to tackle the problem of counterfeits. But there is a possibility that the scammer can take a photograph of the genuine QR. In such a case, an effective measure to tackle fake mobile QRs is the Dynamic Data feature in mobile QRs. Consider the scenario when the scammer actually indulges in such acts as taking photos of QR and passing it to his friends. The terminal can still detect the fraud because:

- The fake QR will fail the Refresh Time Test. The 'Update Time' in the copied QR will be the time when the scammer took the photograph. As such, the terminal will determine that the QR is stale and assert it as Invalid.

#### **Scenario 4: Fake Mobile Application QR (Booking Location Test)**

The booking location coordinates in the QR Common Data may be used for the very purpose to tackle misuse of the QR Ticketing System. It is particularly applicable in Metro type PTO's tickets only and may not be applicable for Bus operators. When the APP requests a QR Ticket from the TG, a particular PTO may restrict that its ticket cannot be issued if the booking location of the roaming mobile is within some radius of the PTO's own premises.

Further, even if the TG allows the purchase and issues the ticket, a defaulter can still be caught. The terminal can still detect the fraud because:

- i. The User boards the Metro by tailgating.
- ii. Just before exiting user purchases a small amount ticket (like only 2 stops as opposed to the long journey travelled). However, the booking locations are already encoded in the QR.
- iii. The exit terminal will detect that the ticket cannot be valid as the booking location coordinates will indicate it is within the operator's premises. As such, the terminal will determine that the QR is low-value and assert it as Invalid.

## Appendix – II: Usage of Different Scanning Media

### Scanning QRs using media like NFC, Bluetooth, WiFi, etc.

Scanned data can be sent to Validating Terminal with:

- Camera / Optical
- NFC
- Bluetooth
- WiFi

All the above communication methods are wireless communication technologies and operate only on short distances i.e. they require somewhat close proximity between communicating devices although technology modernization and advancement has gotten rid of this limitation nowadays.

*The difference between all the above communication methods is only for establishing the communication channel between the communicating devices. After that, the transfer of payload from the APP to the Validating Terminal does not differ.*

***Furthermore, one major difference between the Camera Module (optical transfer) and the other methods (RF Communication) is that Optical transfer is uni-directional (from APP to Terminal) but NFC, WiFi and BLE can be bi-directional once the channel of communication is established.***

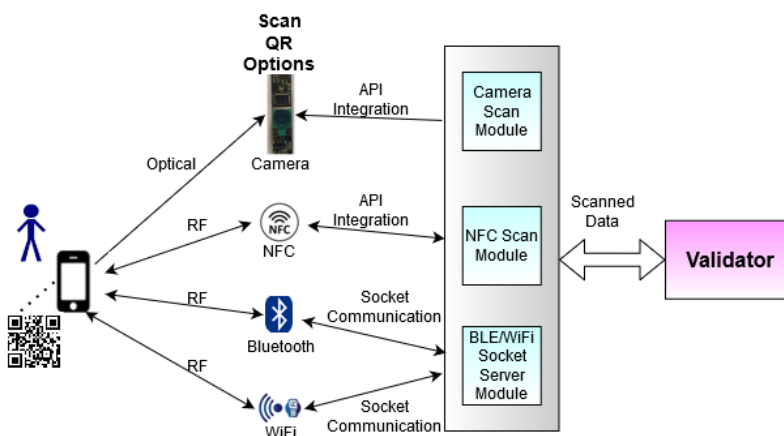


Figure 61: QR Scanning Options

### Communication using NFC, BLE and WiFi:

For completeness, here we describe some technical specifications for the other three protocols also.



- **NFC** – The data can be transferred in the form of radio wave (13.6 MHz) by contacting the two devices together and the single beam (radio waveform) transfers the data between the two devices within the range of  $\leq 4$  cm
- **BLE (Bluetooth Low Energy)** – The data can be transferred in the form of radio wave (2.45 GHz) that acts as carrier to transfer the data between the two devices within the range of up to 30-35 feet for Bluetooth 4.0 or less. For Bluetooth 5.0 the range can extend well over 300 feet. However, signal strength and throughput degrades considerably with distance.
- **WiFi** – The data can be transferred in the form of radio wave (2.4 GHz & 5 GHz) that acts as carrier to transfer the data between the two devices within the range well over 150 feet.

The process of transferring QR Data from the Mobile APP to the Validating Terminal program is described here.

1. Pairing the devices – By default the Camera option will always be there. But if the User chooses to he/she may use any other option, if available. Here we are only describing how to establish communication channel in case the user chooses NFC, Bluetooth/BLE & Wifi. NFC, BLE & WiFi all work on Radio Frequencies and the communication happens through RF Communication (RFCOMM). RFCOMM needs to be established on the MAC Sublayer, i.e. on the device MAC address. L2CAP sockets are also used sometimes but it is beyond the scope of this specification. Pairing is an essential part whose main purpose is to get the MAC address identifying the communicating devices uniquely. The physical layer of communication is invariably always UART (Serial Communication).
2. The APP shall display the Scanner capabilities of the Operator. This is defined in the Validator Info field of the ticket.

Validator Info							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1 = WiFi	1 = BLE	1 = NFC	1 = Camera	X	X	X	0 – Unencrypted, 1 – Encrypted

3. Once the devices are paired and user chooses BLE or WiFi, in order to identify external devices, APP uses the MAC address collected beforehand. It then proceeds to create a client socket which will be connected to this device (channel setup).

### Example: -

As an example for Bluetooth, all objects are identified by UUID (Universal Unique Identifier). These include services, characteristics and many other things. *For proprietary communication the Operator may choose to have its own UUID (a paid service assigned by Bluetooth).* Otherwise, the standard available UUID may be used. The following functions are available in the Android Bluetooth Library for integration with the App.

**createRfcommSocketToServiceRecord(DEFAULT\_UUID\_OF\_PTO)**

This returns a normal Application Layer Socket that the APP can then use as any other sockets viz.

**connect(Server Socket);**

**send(QR\_DATA);**

**recv(STATUS);**

Here it must be noted that it is assumed the Validating Terminal is already running a Bluetooth Socket Server.

**Server\_Socket = bluetooth.BluetoothSocket (bluetooth.RFCOMM)**

**port = 1**

**Server\_Socket.bind(("",port))**

**Server\_Socket.listen(1)**

Communication over WiFi can also follow RFCOMM sockets as shown above in the Bluetooth example.

*Camera or optical scanning is the most common and preferred form of reading QR data. This specification assumes that for all practical purposes, the scanned data sent to Validating terminal is from an optical / camera scanner. The scanning capabilities of the Optical Module (Camera) are provided in the QR Code Dataset – Part I QR Specifications.*

## Appendix – III: Futuristic QR Ticketing System

### Examples of implementing real-time status updates in Mobile APP – Desirable futuristic features

#### I. QR Validation using Dynamic Data (Geo-fencing)

As seen in [Figure 22](#), the Dynamic Data element is added in the QR Payload by the APP. Dynamic Data consists of 4 components – QR Update Time, Status, Latitude and Longitude. All these fields are numeric and so they must be represented as HEX chars, encoded in SQDSR format and then finally transformed into Base64 before using it in the QR image.

There is this idea to render a Mobile APP-based QR multiple times instead of the static render-it-once-and-forget-about-it approach. That can happen only with very tight co-operation between the APP, the AFC System and the TG. This concept is sought to tackle the potential problem of QR duplicates. Dynamic rendering of QR will be as follows.

- On the date of the journey, the APP must render a new QR image by changing the QR Update Time to the current date and time of day.
- The image can actually be rendered or ‘refreshed’ regularly based on some time interval logic
- The validator would expect the scanned QR to have a very recent QR Update time or else reject the QR ticket as invalid. This can actually effectively curtail the copying of QRs to a large extent because the copied ones would invariably always turn out to have stale images.
- If the most recent status of the QR ticket (e.g. Entry, Exit, Tap, etc.) can be communicated back to the APP in real-time, and then the APP can even update the Status field of the Dynamic Data and render a new image.
- Geographical location co-ordinates are one way of finding the mobile phone if status information needs to be sent back to the APP. If before displaying the QR to the scanner, the APP can also update the location info (by calling a GPS API – geo-fencing) and render a new QR image, then the validator can check if it has the same co-ordinates as the QR and thus determine if the QR is valid or not. The validator coordinates are fixed and can be hard-coded into the design.

#### II. Validation of QR using BLE, Wi-Fi and NFC

This is a desired feature, wherein the QRs of the user can be verified and real-time status change updates of the journey can be sent to the mobile APP, if the user or commuter joins the PTO network either through WiFi or Bluetooth. A detailed description of establishing communication with Validating Terminal using NFC, Bluetooth and Wifi is given in [Appendix – II: Usage of Different Scanning Media](#).

However, it has to be mentioned here that communicating QR Information using Bluetooth, NFC and WiFi is not without its own problems.

- In the busy rush of the day, when commuters are going about their business, they cannot be mandated to perform – BLE-pairing or join the operators Wi-Fi network as it is not in their interest to do so.
- The BLE pairing or joining Wi-Fi has to be performed at multiple places, as the source and destination stations are random, which may be an inconvenience to the commuter.
- The range of Bluetooth and WiFi are large, in access of 100 meters. So there is a possibility of some wisecracks playing pranks with the terminal gates from far –off distances and creating nuisance.

There is another method of validating QR and updating journey status using NFC (Near field communication). Even in this method, there is a requirement for the user to start the NFC communicator on his/her mobile. This method also has more cons than pros in its favour.

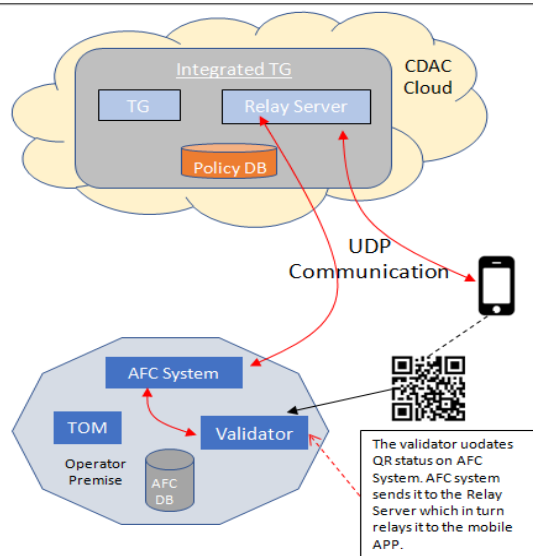
- The technology is not sound and fully mature. There have been a number of reports world-wide of people's mobile phones getting hacked and sensitive information stolen. Naturally, the public is going to very wary of using this method.
- It is a technology that is supported most in expensive smart phones, so effectively it rules out a sizeable chunk of the population. Any technology that is not seamless and addressable across the whole spectrum of the customer-base should be avoided.

### **III. Send Real-time QR Status to APP with UDP**

Given on the Implementation Guidelines V1.0 is a sophisticated scheme to send real-time journey updates to customer mobiles using some open-source Cloud Messaging and Notification Service. In this section, we'll show a bare-bones in-house implementation to send such notifications at real-time using the UDP/IP protocol suite.

Firstly, we establish the premise.

On the activation date of the QR Ticket i.e. the date of journey in case of transit applications, the QR may transition through various states – e.g. Active, Entry, Exit, Complete, etc. at different times of the day. There is a desirable requirement that the mobile APP too should reflect these state changes in real-time. In order for that to happen, these state changes must be relayed from the operator network back into the mobile APP.



**Figure 62: Real-time Status Update on Mobile with Relay Server**

Essentially the entity that actually detects the status of a QR is the Validator. When the validator updates the status of a valid QR on the AFC System, the AFC System can then relay it to the Integrated TG (relay server), which in turn can relay the status back to the APP. The APP can then reflect this change either by displaying it to the user and/or rendering a new QR image with the updated status.

The reason we choose UDP over TCP is because, it is lightweight – only 8 bytes for the header – and it is often the preferred transport mechanism if two or more parties wants to exchange information quickly. Moreover, UDP is connection-less meaning it does not have to perform handshakes or maintain internal states.

Since UDP sockets work on IP addresses, the question is how the Relay server will find the mobile APP.

The process is elucidated below:

- On the day of journey, when the user opens the APP to display the QR to the scanner, the APP sends a UDP message containing the QR\_Tkt\_Serial\_No to the Relay server and then waits for a response. This is a background activity and does not consume many system resources. The UDP payload consists of a very small size as follows:

**M:QR\_Tkt\_Serial\_No:Status:Time; where M stands for Mobile**

As an example,

Let Operator ID = 0xA. Tkt# = 1, and State = 1 (active). Therefore, status = 0xA11. Please refer to Table 5.6 of QR Specifications – Part I for details about the QR Status field.

**M:24052020M00004567:A11:113045**

## QR Ticketing System – Interface Specification

- Socket communications by default contains the IP addresses and port numbers in the header. The Relay server will store message received in a Hash Map indexed by the serial no. But it sends the real-time status as soon as it is received from AFC as mobile IP addresses are dynamic and can change frequently while roaming.

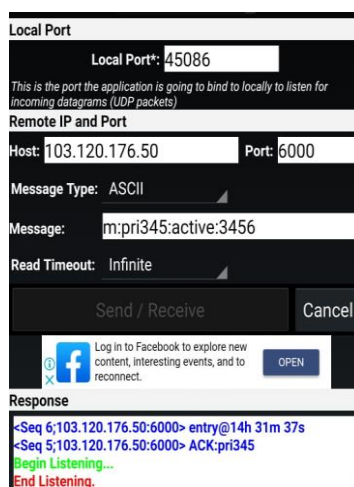
Serial#	Type	(IP Address, Port No)	Status	Time
2405202000004567	M	(100.93.34.203, 43806)	A11	113045

- Assuming the Validator receives a valid QR and the state is now Entry (=2), it sends the status to the AFC System. The AFC system relays the information back to the Relay Server as follows:

**T:24052020M00004567:A12:113147; where T stands for Terminal**

- When the Relay server sees that a message has arrived with a 'T'; it performs a lookup on the Hash Table with the serial number. If the lookup succeeds then it updates the Status, fetches the <IP Address, Port> tuple and sends the updated status to the same address as a UDP message.
- Meanwhile the APP which had been waiting for a response, receives the status update and can then display it to the user and also render a new QR image.

While doing R&D on the project, CDAC actually implemented a very bare-bones model of this design and successfully sent updates over UDP from a dummy TG/Relay server to different mobiles on different 3G/4G networks. A snapshot from one such mobile is shown here below.



**Figure 63: App Receives Status Update from Relay Server**

It must be noted here that there are many challenges associated with this approach – like packet loss, strict synchronous sequence of messages, irresponsive mobile phone when waiting for status updates, etc.

## Appendix – IV: File Naming Conventions

### Naming conventions of various JSON files used throughout this document

1. Purchase QR MIS Report from TG → AFC: qr\_DDMMYYYYhhmmss\_<TG\_ID>\_<PTO\_ID>. **E.g. qr\_12062020185001\_23\_135**
2. MIS Report PSP → APP Provider: <Aggregator Name>\_<Merchant Order No>\_DDMMYYYYhhmm. **E.g. PSPEPAY\_MO-1\_130620200115**
3. MIS Report App Provider → TG: MIS\_DDMMYYYYhhmm\_<App Provider ID>\_<TG ID>\_PSP\_ID. **E.g. MIS\_130620200115\_15\_23\_78**
4. Financial Settlement MIS Report for QRs bought online through Mobile APP or Webclient from TG → AFC: qrf\_DDMMYYYYhhmmss\_<TG\_ID>\_<PTO\_ID>. **E.g. qrf\_13062020031015\_23\_135**
5. Financial Settlement MIS Report for Paper-based QRs bought at TOM from TG → AFC: – qrfp\_DDMMYYYYhhmmss\_<TG\_ID>\_<PTO\_ID>. **E.g. qrfp\_14062020113045\_23\_135**
6. Policy update request from AFC → TG: qrpq\_DDMMYYYYHHMMSS\_PTOD\_TGID. **E.g. qrpq\_06042020142015\_10\_23**
7. Policy query request from AFC → TG: qrpq\_DDMMYYYYHHMMSS\_PTOD\_TGID. **E.g. qrpq\_06042020142015\_10\_23**

\*File extensions for the above file names are not shown and is assumed to be JSON.

\*\*\*\*\* End of Part III \*\*\*\*\*

## Conclusion

Over the last 4-5 years, QR codes have become the most popular medium of choice for instant payments in our country and across the globe. With the ever-increasing demand for deployment of QR Codes as fare media, it is important that the Public Transport ecosystems provide a consistent and seamless experience to their customers. The system should enable people to remotely book tickets not just for urban transport operations like Metro and Buses, but also other varied services like hotel reservations, entertainment events, tourist spot reservations, etc. The popularity of QR codes also means that there is a good availability of online technical support. This allows service providers to quickly develop and implement solutions compatible with most existing payment gateway systems.

The specifications described in the preceding pages above are only a first step towards achieving a world-class QR Ticketing Ecosystem that strives to provide customers a hassle-free travel experience on one hand, and to service providers a fully automated state-of-the-art ticketing solution, on the other. However, this is only just the beginning – the first innings, so to speak. The road ahead is long and winding, but as the saying goes – *‘Well begun is half done.’*

In its second innings, this ecosystem must evolve towards route and operator agnostic ticketing solutions and encompass other non-transit services under its umbrella. The not-so-distant future should also address some of the most pertinent needs of the modern world like vehicle capacity management, real-time traffic updates, attractive pricing, etc. through the use of Industry 4.0 technologies viz. artificial intelligence, machine learning, big data and IoT.



# **Implementation Guidelines & FAQ – QR Ticketing System for Transit Applications**

**Version No: V 1.0**

**Release Date: 20/03/2021**

**Centre for Development of Advanced Computing**

(A Scientific Society of the Ministry of Electronics and Information Technology, Govt. of India)

**NOIDA Unit**

**C-56/1, Institutional Area, Sector-62, Noida-201307**

## Contents

1. Introduction .....	5
2. Acronyms and Abbreviations .....	5
3. Terms and definitions .....	5
4. Implementation of the QR Ticketing System .....	7
4.1. Basic Implementation .....	7
4.1.1. Ticket Purchase Mobile and Webclient Application .....	7
4.1.2. Ticket Office Machine (TOM) Application .....	8
4.1.3. Ticket Validation in QR Ticketing System .....	10
4.1.4. Validating Terminal and AFC System Updates .....	12
4.1.5. Simple Customer Care Application Integrated with TG .....	15
4.1.6. Implementing Products and Passes in QR Ticketing System .....	17
4.2. Futuristic Implementation .....	22
4.2.1. Advanced Features in QR Ticketing System .....	22
4.2.1.1. Sending Real-time Transit Notifications to Customer's Mobile Route Request .....	22
4.2.1.2. Route-agnostic Intra-PTO QR Code Ticket .....	24
4.2.1.3. Paid-2-Paid Inter-PTO QR Code Tickets .....	25
5. References .....	29
Annexure: Frequently Asked Questions .....	30

## List of Figures

Figure 1: Flow Diagram – TOM Paper Ticket Issuing Interface .....	9
Figure 2: Entry Station Validation in Metro Gates .....	10
Figure 3: Exit Station Validation in Metro Gates .....	11
Figure 4: Sample QR Transit File .....	13
Figure 5: Flow Diagram – Customer Care Application .....	16
Figure 6: Policy DB Update for Products (Passes) .....	17
Figure 7: Sample QR Monthly Pass Ticket JSON .....	19
Figure 8: Sample QR Senior Citizen Pass Ticket JSON .....	21
Figure 9: Push real-time updates to user's mobile .....	24
Figure 10: Paid-2-Paid Interchange .....	26
Figure 11: Paid-2-Paid Interchange Ticket Information JSON .....	28

## List of Tables

Table 4.1: App Access Permission List .....	7
Table 4.2: QR Transit File Data Elements .....	14
Table 4.3: Fare Matrix PTO ID 10 .....	26
Table 4.4: Fare Matrix PTO ID 135 .....	26
Table 4.5: Combined Fare Matrix PTO ID 10 and 135 .....	27

## Version History

Date	Version	Author	Comments
21/02/2021	1.0	CDAC	First version created after Workshop for Operators

## 1. Introduction

The QR Ticketing System for Transit Operators is an integrated system of using QR Codes as prepaid tickets to avail mainly transit services of all modern operators like Metro, Bus, Rental Cars, Rail, Flight, etc. This part – **QR Ticketing System Implementation Guidelines** provides some valuable tips and insights to implementing the QR Ticketing System and integrating it with AFC systems. An annexure, addressing some of the most ‘frequently asked questions’ is also provided here.

## 2. Acronyms and Abbreviations

<b>AFC</b>	Automatic Fare Collection
<b>API</b>	Application Program Interface
<b>APP</b>	Application
<b>NCMC</b>	National Common Mobility Card
<b>PSP</b>	Payment service Provider
<b>PTO</b>	Public Transport Operator
<b>QR Code</b>	Quick Response Code
<b>RAQT</b>	Route Agnostic QR Ticketing
<b>SLA</b>	Service Level Agreement
<b>TG</b>	Ticket Generator
<b>TOM</b>	Ticket Operating/Office Machine
<b>UI</b>	User Interface
<b>UPI</b>	Unified Payments Interface

## 3. Terms and definitions

The following definitions are used throughout this Standard:

- **QR Code Tickets:** Tickets in the form of QR images shall be used for single journey / Group tickets as an alternative to cash based paper ticket and token. QR code tickets shall be used in mobile phones or in Paper Media.
- **HTTPS:** HTTPS is a secured form of HTTP protocol. HTTPS, HTTP/2.0, HTTP over SSL and HTTP with TLS are all synonymous terms. It was designed to make internet communication between remote entities secure and confidential. Security is implemented in the transport layer, which is TCP for all practical purposes. All modern browsers and mobile applications support HTTPS

communication. Applications that facilitate financial transactions are mandated by the RBI to use HTTPS as the communication mode.

- **REST API:** It refers to Representational State Transfer Architecture, a form of software architecture usually used for creating Web services. In RESTful Web service, requests and responses are usually exchanged in the form of HTML/XML/JSON and the communicating channel is HTTPS.
- **APP Provider:** The smart-phone application referred to throughout the specification as ‘APP’ may be provided by the merchant or PTO itself, or any third party aggregator service. This entity is referred to in this specification as the APP Provider.
- **Payment Service Provider:** A Payment Service Provider or PSP for short may be a bank or financial institution that processes all existing payment services e.g. Credit/debit cards, Netbanking, Wallets, etc. on behalf of a merchant. The merchant in this case may also be the APP Provider. It is the role of the PSP to handle financial transactions and distribution to operators. This distribution of funds is a very important function and is discussed in detail in this document.
- **Integrated TG:** The Ticket Generator or TG is a high-end Server that may have many components tightly integrated with it – for example the Policy DB, PTO’s AFCS, the Web Service / Web Socket process, the Relay server to relay real-time status updates to the mobile, etc. The main TG process per se implements the actual business logic. All these components and modules together are referred to as the Integrated TG. The terms “TG” and “Integrated TG” are used interchangeably in this document and must not be differentiated as separate entities.
- **NCMC-compliant AFC System:** Indigenously developed, inter-operable Automatic Fare Collection (AFC) System which is vendor and bank agnostic. Using a single NCMC card (Debit/Credit/Prepaid Card, issued by any Bank), customers can make payment for any purchases and also use it as a travel card for urban transport services like Metro transit, Bus transit, Toll, Parking etc. across the nation. AFC Systems of such transit operators that support usage of NCMC cards are usually called NCMC-compliant AFC Systems.

## 4. Implementation of the QR Ticketing System

In this section, we shall describe some concepts of the QR Ticketing System from an implementation point of view. We shall follow a logical workflow of the system from ticket purchase to validation. Finally, we shall describe some advanced (futuristic) features that are desirable in today's advanced technological age.

### 4.1. Basic Implementation

This section shows the minimum barebones features that need to be supported in a basic implementation of the QR Ticketing System.

#### 4.1.1. Ticket Purchase Mobile and Webclient Application

This is the customer face of the QR Ticketing System. Customers must be able to use an App (like a Mobile or Webclient App) to buy Bus or Metro journey tickets. The Apps can also provide facility to buy tickets for Rail or Air journeys. Some of the salient features that every App must support are

- An UI that is integrated with the QR Ticket Generator (TG) which, in turn, is integrated with PTO's AFC. The App Provider may be any third-party commercial application but integration with TG is mandatory. CDAC's Travel Mozo App is one such example of a third-party system that is easily integrable with PTO's AFC via CDAC TG Services.
- The App must follow the QR Specifications to generate QR Tickets in the typical QR Ticketing System format.
- Due to constraints of security and other services like "location", App must ask for the user's permission. The minimum set of Access Permission List is shown in [Table 4.1](#).

**Table 4.1: App Access Permission List**

Service Parameter	Description
NFC (Near-field communication)	It totally depends on the PTO station center if they need it for communication with the device to scan a QR Ticket. NFC is available only on some high-end mobile devices.
Internet Connectivity (Mobile Data & WiFi)	Device must have internet connectivity for booking a ticket. Internet connectivity may be either mobile data or WiFi.
Bluetooth connectivity	It totally depends on the PTO station center if they need it for communication with the device to scan a QR Ticket. Application will ask for this.
Live Geo-Location	The QR Ticketing System can optionally maintain the actual physical latitude and longitude of the device at the time a ticket is being purchased. This is a precautionary measure that the PTO may enforce in order to prevent misuse or fraudulent tickets. Furthermore, in case of booking a cab or rickshaw or for tracking the device, the app needs the live location of device.

Contact number	For account security verification like OTP app needs phone number of user.
SMS and Phone call	In case of cab and rickshaw, app may need to call the driver for getting the information about booked rides. SMS may be required for OTP and other purposes.
Push Notification (Message Notification)	Device must allow this for getting the notifications like: <ul style="list-style-type: none"> <li>The day of journey notification</li> <li>When user uses the QR Ticket, the real-time transit update should be sent via push notification and the App must update the information in the Ticket History accordingly</li> <li>To send QR Ticket validity (Expiry time of ticket) alert</li> </ul>

#### 4.1.2. Ticket Office Machine (TOM) Application

Issuance of Paper QR Tickets intended to replace the tokens used in Metro station. So all Metro operators must provide an Application in their Ticket Office premises that can facilitate such purchases. These applications may be attended or unattended (Ticket Vending Machines). TOM applications must also have the same integration with the PTO's own AFC. [Figure 1](#) below shows a simple design of an attended TOM application that may be installed in a Metro station premises.

#### TOM – Workflow and Design:

There are mainly 4 components to the TOM Application.

1. Super Admin – This is only required for administration purposes. If the PTO requests the TOM APP Provider to check some problem, the App Providers DB and Application Administrator can intervene and look into the issue. The Super Admin console may also be used by the PTO to create and manage the Executive/Personnel handling the requests for ticket.
2. PTO-specific TOM application tailor made for PTO's needs
3. Ticket Issuing Person / Teller – Handles day to day ticket issuing work
4. Station / Depot specific TOM Application Engine
5. Ticket Generator (TG) – The interface with the TG is almost exactly the same as the Mobile APP except that there is no Payment Service Provider Interface required. The PTO collects the Ticket fare using its own mechanisms like Cash, Card, etc. The TG may be local to the PTO and also operated by the PTO itself. But it can also be provided by a third-party service provider. In that case the TG Provider should settle the service charges for all the Paper tickets issued through it. For details on the TG and TOM Payment Reconciliation, please refer to Section 5.4 – TG-TOM Interface of Part III – QR Interface Specifications.



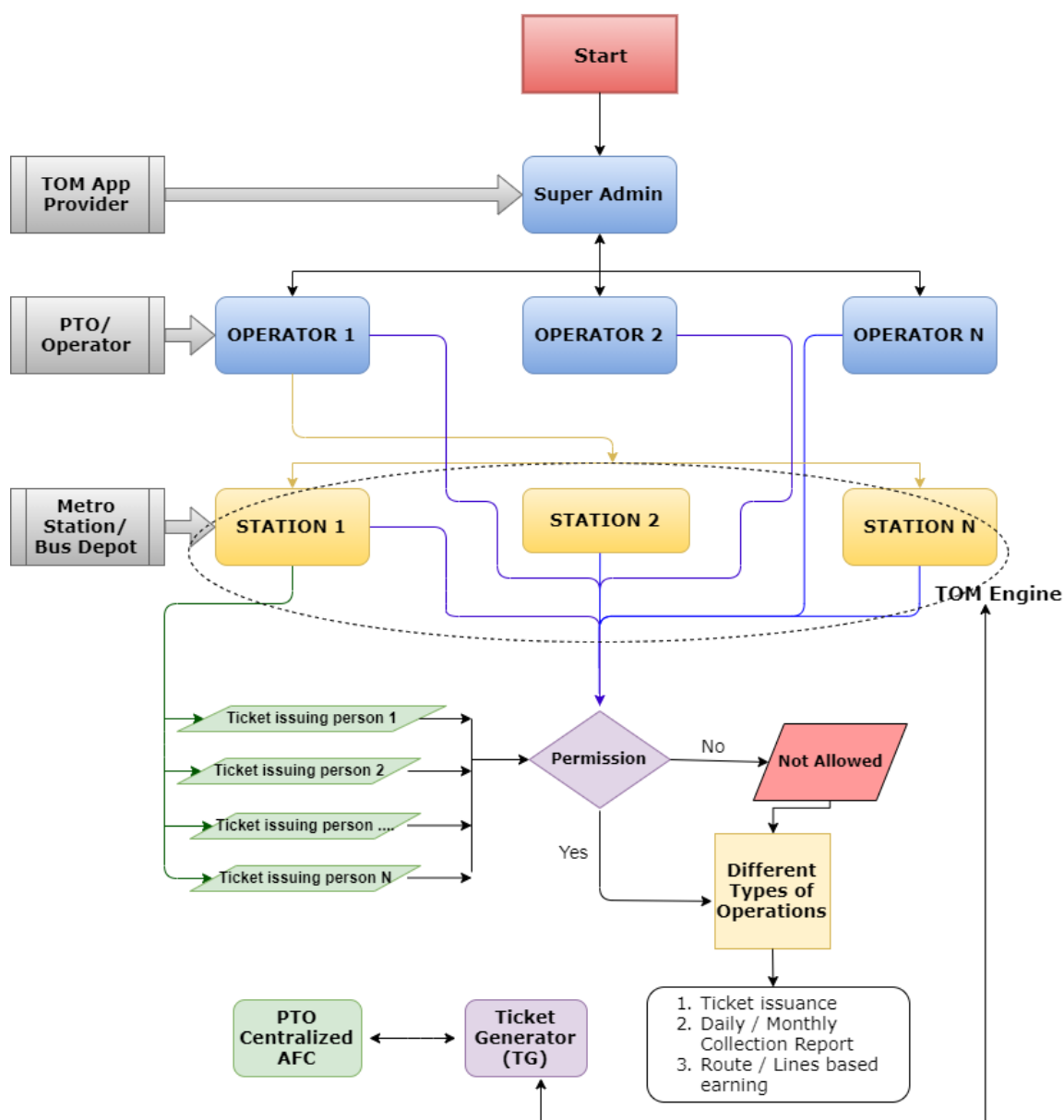


Figure 1: Flow Diagram – TOM Paper Ticket Issuing Interface

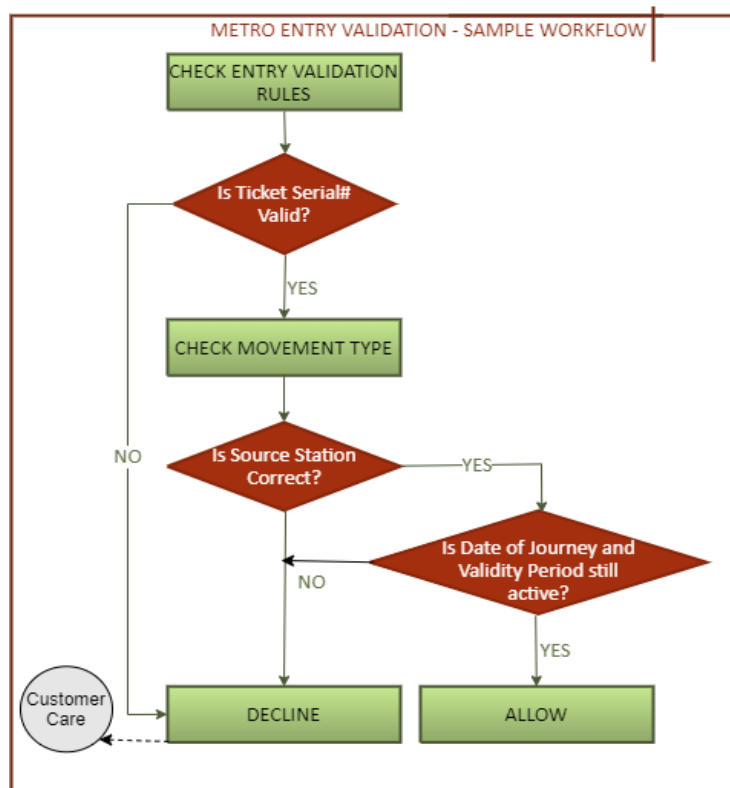
### 4.1.3. Ticket Validation in QR Ticketing System

**TRM or Transit Rules Management** is normally PTO-specific that implements the PTO's business rules pertaining to usage of tickets purchased by the customers. For QR Code Tickets, some common validations that shall be necessary to perform are as follows. Please note that these points are only a set of Recommended Rules.

- The Ticket Serial Number is valid
- Activation Date and Validity time of journey information is correct
- Entry/Exit station information is correct. Please note that some PTOs allow entry anywhere between the Source and Destination stations. Similarly, the commuter maybe allowed to de-board or exit at any station that falls before the actual Destination station specified on the QR Ticket.

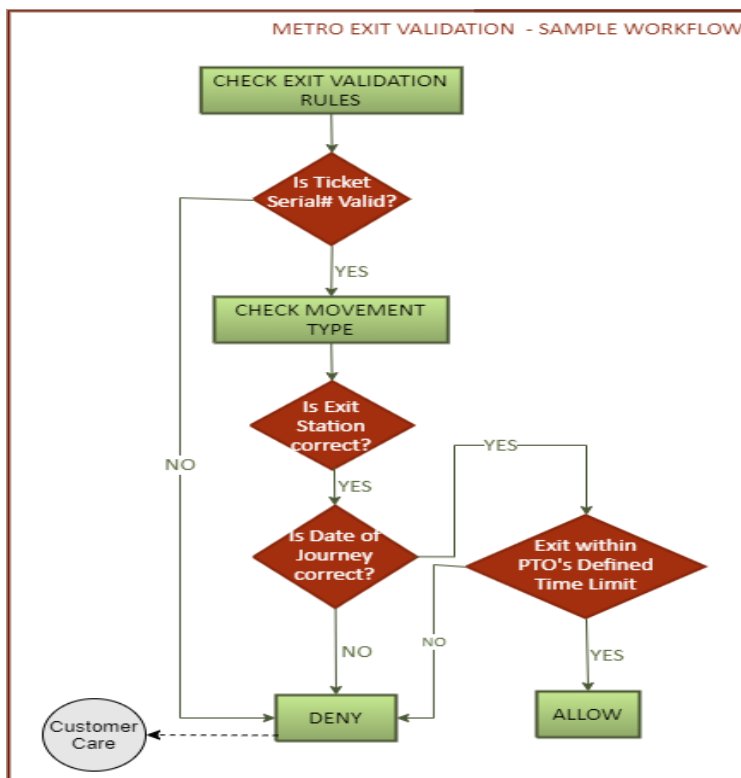
Examples of Ticket Validation for Buses and Metro are provided in Section 4.3.3.4 of Part III – Interface Specifications. [Figure 2](#) and [Figure 3](#) below show sample validation flows that take place ideally in Metro Gates. In the figure the boxes labelled “**Check Entry/Exit Validation Rules**” can be visualized as the main components of the validation engine that implements the PTO's business rules.

#### Entry Station Validation:



**Figure 2: Entry Station Validation in Metro Gates**

### Exit Station Validation:



**Figure 3: Exit Station Validation in Metro Gates**

### Online vs Offline Validation:

In order to increase the throughput, it is desirable to adopt an offline validation engine that has the capability to perform validation offline without the need to access any other system to determine the validity of a QR Ticket. The steps for syncing the used QR information with the AFC back-end can be performed at specific times of the day. However, offline validation also has some drawbacks pertaining to counterfeit tickets and can be alleviated by a hybrid approach like –

- Local LAN online validation where terminals validate with some intermediate Server on the same LAN as the terminals
- PTO may also adopt some innovative measure to counter fake tickets. For example, it can have a system that generates PTO-specific ticket serial numbers and use those serial numbers validated on the centralized Station server. These serial numbers can be encoded on the QR Tickets within the PTO-specific data element of the Dynamic Data branch of the QR Code described in Table 5.5 of Part 1 – QR Code Dataset
- Some PTO may choose to have a hybrid approach wherein, in case of Metro Gate Validation, the Entry terminals can be in an offline mode and the Exit terminals in an online mode.

These decisions to adopt a suitable validation approach lie entirely at the PTO's own discretion.

#### **4.1.4. Validating Terminal and AFC System Updates**

After a QR Ticket is validated at a terminal the process of updating transit information in the AFC Systems – its protocol, communication mode, architecture, etc. – is actually not within the scope of the QR Ticketing System specifications. For the sake of illustrating a real situation, we shall assume that the target AFC System is an NCMC-compliant one.

##### **Flow of transit information into NCMC-compliant AFC:**

1. When QR is presented at the validation terminal, terminal performs required validation checks and generates QR transit file which is then sent to the AFC.
2. Any type of transit file generated at the terminal shall be divided into two blocks – Common Transit Data Block and Variable Transit Data Block. The Common block consists of elements which are common for all media types (viz. NCMC, Cash, Token and QR) while the Variable block consists of only media-specific information. For details, please refer to the NCMC Specifications – Part V.
3. Each data block is further split into two sub-blocks. First sub-block consists of fixed set of data elements and second sub-block consists of operator defined data which may be decided by operator according to requirements.
4. Common and Variable Data Blocks which constitute the QR transit file are marked appropriately and shown in the XML file [Figure 4](#) below.
5. The source of some data elements in the Variable Data Block is the QR Payload present in the ticket. Other elements are generated by the terminal itself. In the figure below, we show the 1-to-1 correspondence of the QR Element against the actual data element of the XML transit file.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<File>
  <hdr>
    <nFlSeqno>000235</nFlSeqno>
    <nFlVERSION>10</nFlVERSION>
    <nFlDtTmLcTxn>210211161727</nFlDtTmLcTxn>
  </hdr>
  <txnblock>
    <txn RecNum="1">

      <!-- Common Transit Data Block Start !-->
      <nVERSION>10</nVERSION>
      <nORD>0000020000042104216000129BF</nORD>
      <nTrmIdOp>010002000004</nTrmIdOp>
      <nAmtTxn>000000000500</nAmtTxn>
      <nDtTmLcTxn>210211161727</nDtTmLcTxn>
      <nTxnTypQ>42</nTxnTypQ>
      <nTxnPlace>45</nTxnPlace>
      <nOpData>
        ...
      </nOpData>
      <!-- Common Transit Data Block End !-->

      <nQr> <!-- QR-related Variable Transit Data Start !-->
      <nMVERSION>10</nMVERSION>
      <nReqId>0000000e</nReqId>
      <nTGId>0007</nTGId>
      <nTSN>11022021155539M0000001384</nTSN>
      <nTktSeqNo>0</nTktSeqNo>
      <nQrType>M</nQrType>
      <nTxnRefNum>3512256587701</nTxnRefNum>
      <nQrGenDtTm>210211155539</nQrGenDtTm>
      <nQrGenLcLat>000000</nQrGenLcLat>
      <nQrGenLcLong>000000</nQrGenLcLong>
      <nUserPhNo></nUserPhNo>
      <nPTOInfo>
        ...
      </nPTOInfo>
      </nQr> <!-- QR-related Variable Transit Data End !-->
    </txn>
  </txnblock>
  <trl>
    <nNoRecords>1</nNoRecords>
    <nTotalAmt>000000000500</nTotalAmt>
  </trl>
</File>

```

Figure 4: Sample QR Transit File

**Table 4.2: QR Transit File Data Elements**

QR Payload Element Name	Data Format	Size	XML Tag	Description
Requester ID	AS (Hex String)	8	nReqId	This term represents the unique ID of Application or TOM used for availing the QR Services. This is either the AppID (Application ID) or TomID (Ticket Office Machine ID).
Ticket Generator ID	AS (Hex String)	4	nTGId	This term represents the unique ID of QR Ticket Generator (TG).
Ticket Serial No	AN	25	nTSN	This represents the unique identifier allocated for each generated QR.
Ticket Sequence No	NS	1	nTktSeqNo	This represents the sequence of ticket block in the QR ticket. Since, a QR ticket can have maximum 10 tickets; range of this field shall be 0-9.
QR Type	AS	1	nQrType	This represents a unique code of source from where request for purchasing QR ticket is generated. This shall be M for Mobile and W for Web client App or T for TOM.
TXN Ref. No	AN	Up to 22	nTxnRefNum	Transaction Reference Numbers are generated at external source, in this case the PSP.
QR Generation Date & Time	NS	12 (YYMM DDHHm mss)	nQrGenDtTm	This term shall represent the date and time of the instance at which QR code is generated
QR generation location latitude	Hex String	6	nQrGenLcLat	This represents the latitude of mobile's location used to purchase QR ticket.
QR generation location longitude	Hex String	6	nQrGenLcLong	This represents the longitude of mobile's location used to purchase QR ticket.

User Phone no.	N/AN	Up to 32	nUserPhNo	Registered phone number of user. It shall have a size of 32 in case phone number is tokenized due to security reasons.
Operator-defined Data	Several	Up to 255	nPTOInfo	Data specific to operator which may consist of information like Source Station, Destination Station, Duration, Validity, Product Type, Service type etc.
N/A	AN	32	nORD	Present in the Common Transit Data block. This is the unique ID of any transit information in NCMC-compliant AFC Systems.

#### 4.1.5. Simple Customer Care Application Integrated with TG

Customer Care is a very important feature for all transit service providers. In order to provide customers top quality service and a value for their money PTOs usually employ many Customer Care personnel. To aid the Customer Care executives in their job, a good Customer Care application is a necessary requirement. In case of the QR Ticketing System the QR ticket that the customer presents at a validation gate or terminal may run into some problem. A ticket may fail validation for a variety of reasons – genuine or due to some glitch. In all such cases, the commuter can approach the Customer Care centre. The purpose of the Customer Care application is to provide the operator with enough information about the ticket based on which the operator can take the correct decision to alleviate the problem faced by the customer. A simple Customer Care application must support at least the following minimum requirements:

1. An Application that is integrated with a QR code scanning device. These devices are readily available and are usually referred to as HID – Human Interface Device. For simplicity, it is recommended to use a USB-based HID that works as a Keyboard Emulator. That way when the operator places the cursor on a Standard Input text field and scans the QR – emulated keystrokes – which is the actual QR payload – becomes available to the Customer Care application for further validation.
2. The QR payload can then be validated and displayed on the Customer Care App Form.
3. Based on the validation, the Customer Care operator takes the decision. Decisions are entirely the PTOs own business rules and policies.
4. It is quite possible that the Customer Care App to have the option to print a duplicate QR ticket.
5. Interface of the customer care should obviously be directly interfaced to the correct Ticket Generator. Since one QR ticket can be issued by one and only one TG, the ticket validation of the issued ticket must land at the same TG.

For workflow details of several practical Customer Care scenarios please refer to Appendix I of Part III – QR Interface Specifications. Figure 5 below shows a simple Customer Care Application workflow design.

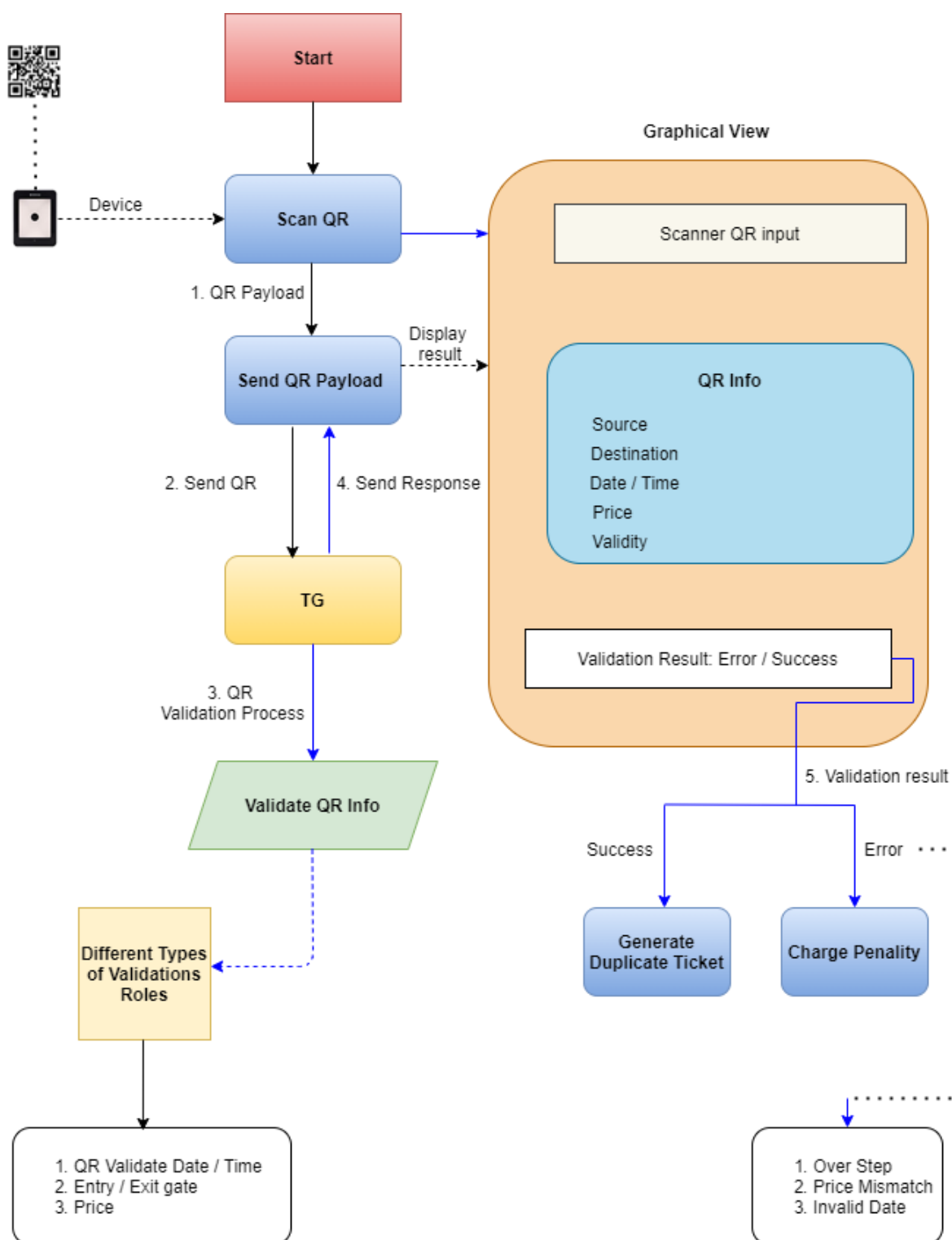


Figure 5: Flow Diagram – Customer Care Application



#### 4.1.6. Implementing Products and Passes in QR Ticketing System

Trip Passes are an integral part of any transit application. PTOs of all types – Bus or Metro operators – often issue many passes like – Senior Citizen Pass, Student Pass, Day Pass, Monthly Pass, etc.

In the QR Ticketing System, the Product ID parameter is used to populate Passes. It is sent as a Policy Update Request. The figure below is the same as **Figure 36: Part III – QR Interface Specifications**. Products must be provided by the PTO for update in the Policy DB of the TG. The PTO sends only those products that are relevant for itself. Here we have a PTO – PTO ID = 10 that implements 7 products – Standard, SeniorCitizen, Student, DailyPass, WeeklyPass, WeekendPass and SpecialPass. Please note the ID numbers of these product values must be consistent with the **Table 5.14: Part I – QR Code Dataset Specifications**.

```
"QR_Update_Policy_Request": {
  "Hash_Token": {
    "Hash_Value":
      "2129ef13a1208b993b2e5c1c11318af52237f8a5579db41392a03646fb137271"
  },
  "Message_Payload": {
    "Message_Key": {
      "Key_Id": "11"
    },
    "Payload_Data": {
      "Operator_Details": {
        "OperatorId": "10",
        "OperatorName": "PTOName10",
        "OperationType": "0",
      },
      "Filename": "qrpu_06042020142036_10_23",
      "No_Of_Records": "1",
      "Record_1": {
        "Policy_Id": "2",
        "Service-related": {
          "nVer": "1-32",
          "nService": "1-Regular;2-Express;
            21-Normal Feeder",
          "nProduct": "1-Standard;2-SeniorCitizenPass;
            3-StudentPass;4-DailyPass;5-MonthlyPass;
            6-WeekendPass;9-SpecialPass",
          "nProductMinAmt": "1-NA;2-200;3-200;4-200;
            5-200;6-200;9-200",
          "nValidity": "43200",
          "nDuration": "180",
          "nGrpSizMin": "2",
          "nGrpSizMax": "20",
          "nHashAlgo": "SHA256",
          "nScan": "Camera;NFC"
        }
      }
    }
  }
}
```

**Figure 6: Policy DB Update for Products (Passes)**

### Example of a QR Ticket using a QR Pass Ticket (Monthly Pass for Metro):

The ticket issuing Application must provide only one QR Ticket that should contain the information for a Pass. Essentially, the following criteria are of importance –

1. An App that must provide the option to User to buy Passes. Passes are PTO-specific and cannot be used for inter-operator journeys unless some operators decide to issue joint passes.
2. If a user has already bought a Pass and the validity of the pass has expired, then it is desirable for the App to disable the ticket on the Phone so that it cannot be presented. In order for the TG to determine the current status of passes, PTO's AFC shall share relevant details like transaction amount (To update remaining balance in Pass), trip count (To update remaining trips in case of trip based Pass) etc. with TG.
3. A Global "All Access" Source and Destination ID in the Policy DB that would imply a 'Pass' QR Ticket. The Station Code for "All Access" can be fixed as 255 or 'FF' Hex.
4. Validity Period = 43200 minutes. *One QR Pass Ticket is for a maximum validity of 45 days only.* Please refer to **Part I – QR Code Dataset Specifications** for details.
5. The User must pay up at least the minimum amount specified by the PTO for the Pass. This is also a Policy DB parameter as shown in the figure above. The user cannot purchase a QR Pass Ticket without paying up this minimum amount.
6. For all types of Passes, the APP must provide a facility to query the 'Query Balance'. Actually the TG must send the updated QR Payload reflecting the actual available balance on the Ticket\_Fare field. This will be clear in the following steps.
7. After the Ticket (or Pass) is purchased, like any other QR Ticket, the TG sends the "Purchase QR" information to the AFC. The AFC must notify all the terminals as a Start of Day (SOD) object the Passes information. These SOD objects must contain the 'Ticket Serial Number' of the QR the 'Available Balance Amount' and the 'Minimum Required Balance'.
8. When a Pass is presented on the terminal, the TRM Validation Rules must ensure that the Product ID is correct (= 5 in our example) and the 'Balance Amount' on the scanned QR (this is the same as the Ticket\_Fare value) and the data available on terminal must match. Then the remainder amount must also be at least equal to the 'Minimum Required Balance'. As a side note, the APP must also send notifications to user when the Minimum Balance Amount is about to exhaust so that the user can recharge the pass.
9. If the 'Minimum Balance Amount' is not enough, then – whether the commuter oversteps or not – all the Exit terminals should deny the commuter to pass through and direct him to go to Customer Care. Any additional due amount can be recovered there and then.
10. Since the "Entry-Exit" pairs of a "Pass" journey would always be available with the AFC, it must update the balance in that Pass accordingly. These updated balances (along with the Tkt\_SI\_No) must be resent to the terminals again. The updated balance information must also be sent to the TG.
11. So the next time the user sends the 'Query Balance' request to the TG, the TG updates the Ticket\_Fare with this balance to reflect the correct available balance. It must actually send the

entirely new QR Payload as the Digital Signature would have to be updated. The APP must display the “Updated Balance” to the user and also render the new QR.

12. This cycle can repeat from Step 6 until Step 9 is reached or the user actually recharges the Pass.

A sample QR Pass Ticket in JSON form may be visualized as shown in the figure below. Here we assume the value of 255 or FF (Hex) as “All Access” Source and Destination ID.

```
"QR_TicketBlock": {
  "Dynamic_Block": {
    "OperatorID1": {
      "OpID": "10",
      "NoOfTickets": "1",
      "Validator_Info": "49",
      "TicketInfo": {
        "TicketInfo1": {
          "Grp_Size": "1",
          "Src_Stn": "255",
          "Dest_Stn": "255",
          "Activation_Date": "15/02/2021@15-15-00",
          "Ticket_Fare": "200",
          "Product_Id": "05",
          "Service_Id": "01",
          "Validity": "43200",
          "Duration": "180"
        }
      }
    }
  }
}
```

**Figure 7: Sample QR Monthly Pass Ticket JSON**

### Example of a QR Ticket using a Senior Citizen / Student / Special Pass:

Senior Citizen, Student or Special (for differently-abled persons) products or passes are not as straightforward as the other passes. There is a mandatory requirement of Verification required before these can be used. The following steps provide a simple yet effective method to implement these type of Products in the QR Ticketing System.

1. In this method, the APP provides user to the option to buy a Senior Citizen or Student or Special Pass as any other Product or Pass.
2. The APP must mention that these Passes need ‘Verification’ and the process as described in these steps must be clearly mentioned.
3. For these type of Passes, it must be mandatory to fill the Personal Info, which is part of the Static Block of the QR Dataset. Please refer to Table 5.10: Part I – QR Code Dataset. In the Personal Info all information in the <Primary Member Info> → Name, Age, Gender, ID No and ID Type must be filled out.

4. The user must specify the date from which he wishes to make the Pass active. This should be filled in the Activation\_Date field of the Ticket (or Pass in this case).
5. Once all the information is filled, the user must be asked to agree with these rules and can then proceed and buy the ticket. If the payment is successful, the APP must render a QR having only the Tkt\_SI\_No.
6. On the TG, these serial numbered tickets must be marked as “Requires Verification” or something in those lines.
7. On the APP, there must be a button like “Request Ticket”. Until the ticket is actually verified, till then clicking this button the TG should simply respond with “Ticket Not Validated”.
8. On the date of Activation, the user must proceed to Customer Care with his QR. Naturally, this QR will not work on any Validation terminal as it does not have any journey information.
9. The Customer Care executive shall scan the QR and fetch the Pass details from TG. He must then proceed to verify the actual Physical Identity of the user with the information fetched from the TG.
10. Once the Customer Care executive is satisfied that the information provided when purchasing the QR Ticket (or Pass) is consistent with the actual ID, he must send the request to the TG to mark the Pass as ‘Valid’. The TG must acknowledge the Customer Care application affirmatively. The Customer Care executive must then inform the user that he can now ‘Request Ticket’.
11. This time when the TG receives the request, it must then return the QR Payload.
12. The APP can then proceed to render the actual QR Pass on the user’s APP and the user can start using it just like any other QR Pass.

A sample QR Senior Citizen Pass in JSON form may be visualized as shown in the figure below.

```
{
  "Requester_ID": "15",
  "Txn_Ref_No": "ABCDRC2021012488888123",
  "Language": "0",
  "Txn_Type": "65",
  "Txn_Date": "24/01/2021@08-47-57",
  "PSP_Specific_Data": "Mode=IMPS;ServiceFee=1",
  "Total_Fare": "500",
  "Customer_Mobile": "9201345666",
  "TicketBlock": {
    "Static_Block": {
      "Primary_Info": {
        "Name": "Umesh Chandra",
        "ID_No": "AENPC9903L",
        "ID_Type": "02",
        "Age": "65",
        "Gender": "M"
      }
    },
    "Dynamic_Block": {
      "OperatorID1": {
        "OpID": "10",
        "NoOfTickets": "1",
        "Validator_Info": "16",
        "TicketInfo": {
          "TicketInfo1": {
            "Grp_Size": "65",
            "Src_Stn": "255",
            "Dest_Stn": "255",
            "Activation_Date": "24/01/2021@15-15-00",
            "Product_Id": "02",
            "Service_Id": "01",
            "Ticket_Fare": "500",
            "Validity": "43200",
            "Duration": "180"
          }
        }
      }
    }
  }
}
```

**Figure 8: Sample QR Senior Citizen Pass Ticket JSON**

In this example, it is seen that Grp\_Size = 65. Recall from Table 5.13 of Part I – QR Code Dataset Specifications, it implies that the Personal Info of the primary person is included in the QR Ticket seen under the “Static Block” element. Also note that Product Id = 2, which is for Senior Citizen Pass.

## 4.2. Futuristic Implementation

This section shows some advance implementation features of the QR Ticketing System.

### 4.2.1. Advanced Features in QR Ticketing System

Appendix III of QR Interface Specifications – Part III shows a number of features that have been mentioned keeping in view of the future of the QR Ticketing System. These are desirable features that should be taken up to add value to both the customer and PTO.

- Applying Machine Learning and Artificial Intelligence to the System that shall be useful not only to the public but particularly to the PTOs with valuable insights.
- Different Media scanning not just optical or camera scanning but also Bluetooth, WiFi, NFC scanning
- Designing and developing cost-efficient Android based Validation Terminals that can be quickly adopted by the transport industry and quite possibly by other industries as well for e.g. the tourist sites, parking areas etc.

The sections below describe some advanced and modern desirable features that the QR Ticketing System should strive to achieve.

#### 4.2.1.1. Sending Real-time Transit Notifications to Customer's Mobile Route Request

Consumer Mobile Apps must be informative and user-friendly. Notifications like (a) Reminders on the date of journey, (b) Transit updates – journey completion, (c) real-time schedule changes, (d) Traffic conditions (e) Availability of tickets, etc. can be sent to the user's mobile phone and shown in the App to give a clear picture to the customer. The general way to send all such updates to the user's mobile is called 'Push Notification'.

*Technology Service Providers like Google, Apple, Microsoft and Amazon provide Subscription-based Push Notification services that Application Providers can avail to send business updates to their customers in real-time.*

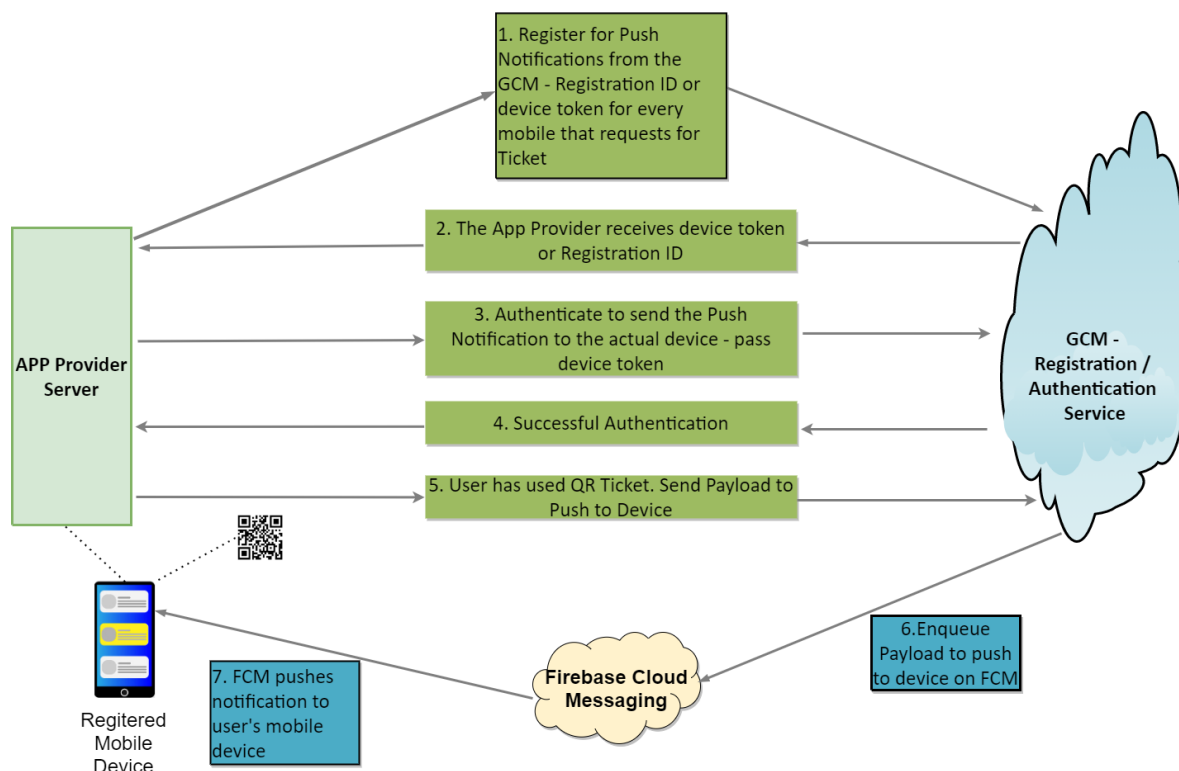
The concept of Push Notification Service is simple and can be described in a nutshell as follows:

1. Subscription to a cloud messaging service like Google Firebase Cloud Messaging (GCM – also called Firebase Cloud Messaging – FCM since 2019) or Apple Push Notification Service (APNS) – These are mobile notification services that enable third-party application developers to send notification data or information from developer-run servers to applications that either target the Google Android Phone or the Apple iOS Phones. The GCM and FCM may also be viewed as 2 separate entities operated by Google wherein the GCM does registration/authentication jobs and the FCM delivers the real-time messages to the user devices.
2. Cloud Messaging Services has the ability to send push notifications, deep-linking commands, and application data well up to 4 KB of payload data. Push Notification Services are usually a paid-

service and so in order to take advantage of this technology the APP Provider must pay up and subscribe annually with a good Messaging Service Provider like GCM.

3. All App Providers must inform the user and seek permission to send 'Push Notification' to their personal mobile. This can be done during App set and installation as a onetime pre-requisite. Here we assume that the user gives consent to the App Provider to send Transit Updates to his/her phone.
4. Upon allowing the App permission to receive and display notifications, the client APP sends a registration API request to the Cloud Messaging Service, say GCM, interface to begin the registration process.
5. The GCM Service receives and acknowledges the request and responds by giving the device a GCM Registration ID – a unique identifier that is used later to send notification to the individual mobile device. The identifier is stored onto the device, and is typically sent to the developer's Application Server (App Provider) to be stored. The Registration ID is a randomly-generated identifier that does not contain any personal or device information that could allow a developer to discover the personal identity of the user.
6. When the App Provider wishes to send a notification event to an individual device, the process begins with an API request to the GCM Authentication Service. The request must include the Registration ID among some other data like the content that is to be displayed on the device.
7. Upon successful verification of the Registration ID and other credentials, an authentication token is returned. Both these identifiers are then sent back to the GCM Service to be "Enqueued" and delivered to the device.
8. Upon receipt of the Push Notification Message, the App can then update the necessary tickets for journey updates or other such information.

The above concepts are shown diagrammatically here below in Figure 9.



**Figure 9: Push real-time updates to user's mobile**

## 4.2.1.2. Route-agnostic Intra-PTO QR Code Ticket

In ordinary day-to-day commuting in cities, it is commonplace for commuters to travel on more than one Line or Route to reach his/her destination. For example, if a person uses three different routes to reach his office or home, it essentially means that his source station (starting point) falls in the first route and his destination station ( ending point) falls in the third route. But in order to complete the entire journey, he has to learn not only about all the available routes including the connecting intermediate routes, but also the stops where he needs to descend and then ascend again in order to continue towards his destination. This is applicable for both Buses and Metro transport. Although, most of the bus transport operators have implemented route based ticketing system currently, it is desirable that the QR Ticketing System should support a Route-agnostic QR Ticketing form of implementation when issuing tickets to customers. This essentially means that the customer should only mention his Starting and Destination stations and the system must be smart enough to determine and show all the possible combinations through which these two points can be connected. The customer can then make an informed decision of which combination he or she wants to avail.



#### 4.2.1.3. Paid-2-Paid Inter-PTO QR Code Tickets

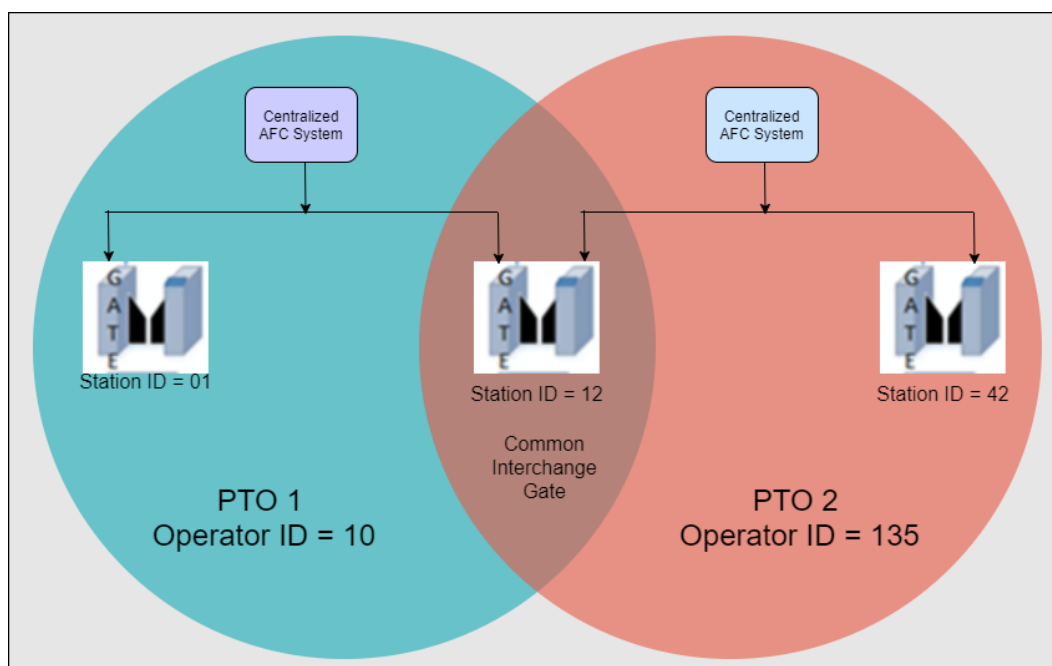
Urban transport has advanced by leaps and bounds in the last 10-15 years in our country. Not only Metro rails, but high-speed Rapid Bus Transit, Rapid Rail, Suburban Rail, Mono Rail, etc. have made their presence felt in our country providing commuters the comforts of fast travel within the city and even to long distance satellite towns and cities.

In an effort to provide further convenience to commuters, different PTOs – operating different transit medium within a city or even between cities – are now collaborating with each other to issue combined single journey tickets from each other's premises thereby sparing the customer from the burden to procuring multiple tickets when hopping from one mode of transport to another. This type of collaboration or interchange is called Paid-2-Paid interchange. One example of Paid-2-Paid interchange are Delhi Metro (operated by DMRC) and Airport Rapid Transit (operated by Reliance). In the city of Mumbai, 14 PTOs operating under MMRDA, running different modes of transport have already collaborated or are in the process of doing so to allow customers to inter-travel on any mode of transport with a single ticket. In general, the PTOs that support Paid-2-Paid travel have *common interchange points* using which the commuter can hop from the premises of one PTO and move to the premises of the other.

#### Paid-2-Paid Interchange:

The following are the main pre-requisites to provide a seamless Single Journey QR Ticket between 2 (or more) PTOs.

1. Consolidated Fare Table Matrix of both PTOs that support Paid-2-Paid interchange
2. App that support the combination or union of all stations belonging to both PTOs. The user of the App should not be bothered to have any prior knowledge of the Operators providing the service. His or her concern must only be the Source and Destination.
3. Technically, the Ticket Generator must implement the journey as a multiple-leg journey whilst the commuter still believes and feels that he/she is travelling on only one ticket. Suffice to say that in case of a single journey inter-PTO Paid-2-Paid interface for 2 PTOs, one QR Ticket would be composed of at least 2 journeys (legs), one for each PTO.
4. Proper identification of Interchange-Points – one common interchange point or station maybe visualized theoretically to belongs to 2 PTOs. It should make no difference if these Interchange Gates are Scan-and-Go or Pass-Thru gates or simply a bypass corridor. On these interchange stations, commuters would disembark from one leg of a journey and then proceed to the subsequent leg.



**Figure 10: Paid-2-Paid Interchange**

Consider the following example where there are 2 PTOs – ID = 10 and ID = 135. Assume that the PTOs have one interchange point. The Fare Tables of the PTOs are shown below.

**Table 4.3: Fare Matrix PTO ID 10**

PTO ID = 10 StationID	0001	1058	1071	0012
0001	0	14	30	40
1058	14	0	16	26
1071	30	16	0	10
0012	40	26	10	0

**Table 4.4: Fare Matrix PTO ID 135**

PTO ID = 135 StationID	0002	0012	0042	7001
0002	0	15	35	50
0012	15	0	20	35
0042	35	20	0	15
7001	50	35	15	0

When we combine both the matrices, we get a consolidated matrix as follows.

**Table 4.5: Combined Fare Matrix PTO ID 10 and 135**

PTO ID = 10 & 135 StationID	0001	1058	1071	0012	0042	7001	0002
0001	0	14	30	40	60	75	55
1058	14	0	16	26	46	61	41
1071	30	16	0	10	30	45	25
0012	40	26	10	0	20	35	15
0042	60	46	30	20	0	15	35
7001	75	61	45	35	15	0	50
0002	55	41	25	15	35	50	0

Therefore, with the above credentials as shown in the figure above – suppose the user pays for 1 QR Ticket from Source = 0001 to Destination = 0042, with Station ID = 0012 being the common Interchange point. internally the TG shall issue a 2 legged journey ticket as follows:

- Leg 1 → From Start:0001 to Destination:0012 (PTO ID = 10, Fare = 40, Activation Date and Time = 15/2/2021@15-15 hours)
- Leg 2 → From Start:0012 to Destination:0042 (PTO ID = 135, Fare = 20, Activation Date and Time = 15/2/2021@16-00 hours)

Now there may be 2 Options of purchasing the Ticket – one with the App and one from the TOM at the station premises.

- Option 1 – Mobile Purchase:** Commuter uses the Mobile App to purchase QR Ticket. It is imperative that the App must clearly show all the stations that can be reached when Paid-2-Paid interchange is available for 2 (or more) PTOs. The user should not be burdened or expected to have any knowledge about the Route/Line or PTO information in any way. This scenario is same as a <multi-operator, multi-ticket> scenario. A sample JSON file is shown below.

```

"QR_TicketBlock": {
  "Dynamic_block": {
    "OperatorID1": {
      "OpID": "10",
      "NoOfTickets": "2",
      "Validator_Info": "49",
      "TicketInfo": {
        "TicketInfo1": {
          "Grp_Size": "1",
          "Src_Stn": "01",
          "Dest_Stn": "12",
          "Activation_Date": "15/02/2021@15-15-00",
          "Ticket_Fare": "40",
          "Product_Id": "01",
          "Service_Id": "01",
          "Validity": "480",
          "Duration": "180"
        }
      }
    },
    "OperatorID2": {
      "OpID": "135",
      "NoOfTickets": "1",
      "Validator_Info": "16",
      "TicketInfo": {
        "TicketInfo1": {
          "Grp_Size": "1",
          "Src_Stn": "12",
          "Dest_Stn": "42",
          "Activation_date": "15/02/2021@16-05-00",
          "Ticket_Fare": "20",
          "Product_Id": "01",
          "Service_Id": "01",
          "Validity": "480",
          "Duration": "180"
        }
      }
    }
  }
}

```

**Figure 11: Paid-2-Paid Interchange Ticket Information JSON**

**Financial Settlement:** The QR Ticketing System is a multi-operator / multi-ticketing system. Multi-operator financial reconciliation is automatically done and these concepts already discussed in details in Section 5.2 of Part III – QR interface Specifications. Please refer to the document to understand how a single user payment is split into respective shares when multiple operators are involved.

- ii. **Option 2 – TOM (Station) Purchase:** Commuter visits PTO premise and buys a Paper QR Ticket issued using the TOM Application. As seen in [Section 4.1.2](#), there is no involvement of an integrated Payment Service Provider with the TG here. When the TOM is used, the PTO collects the money. So when a Paid-2-Paid interchange is involved, the TG issues the same kind of Ticket as shown thematically in Figure 11, except for the fact that the PTO collects the entire fare (e.g. 60/-) from the commuter.

**Financial Settlement:** The onus of settling each other PTO’s shares is now the job of the PTOs themselves. It is beyond the scope of the QR Ticketing System Specifications to provide any information on how that should be handled.

## 5. References

S. No	References
1	“QR Specifications – Part I” – QR Code Dataset
2	“QR Specifications – Part II” –QR Security Scheme
3	“QR Specifications – Part III” –QR Interface Specifications

## Annexure: Frequently Asked Questions

**Q. No. 1: For City Bus and State Transport Buses how do I represent information like State Transport Undertakings Code, Region Code, Depot Code, Route No, Vehicle No, Staff No, Waybill No, etc.?**

**Answer** – Transport Undertaking Code is actually the Operator ID. This is present in the QR Ticketing System. Any PTO-specific parameters, please make use of the ‘Operator-specific Data’ fields provided in the ‘Ticket Info’ and ‘Dynamic Data’ elements of the QR Code Dataset. Please refer Specifications – Part I.

**Q. No. 2: Is it possible to encrypt the entire QR Ticket, including the QR Common Data?**

**Answer** – Yes. The Security Scheme number 6 is just for that. Please note that in that case, if other tickets belonging to different PTOs are present, they will be affected too.

**Q. No. 3: Can one PTO have association with multiple Ticket Generators?**

**Answer** – Yes. The architecture of the QR Ticketing System allows for such association. However, please note that a single QR Ticket can be issued from one and only one TG in real-time.

**Q. No. 4: How many products and services can a single PTO have?**

**Answer** – Multiple. The Product ID field represents a Product and the Service ID field represent a Service. These fields are ticket-specific and are encoded in every individual ticket inside the QR Code – irrespective of whether that ticket belongs to the same operator or not. Technically speaking, both these fields are numeric with Product ID having a length of 2 bytes (maximum value = 65535) and the Service ID has a length of 1 byte (maximum value = 255). With QR Code Dataset V1.0, only one byte of the Product ID is used. So one operator can specify only up to 255 different products. The other byte is reserved for future use.

**Q. No. 5: Some transport operators like the Indian Railways allow up to a maximum of 6 passengers on a single ticket, but the QR Dataset supports only 5. How can I get around it?**

**Answer** – The restriction was placed after trying out several combinations. Encoding Personal Information for any more than 5 members on the QR Code Ticket overshoots the maximum QR Code size limit (binary encoding). Henceforth, the limit remains.

**Q. No. 6: Are personal details even required in QR Code Tickets for urban transport operators? Why have they been included?**

**Answer** – The Personal Details, if present, are coded in the Static Block branch of the QR Code Dataset tree. It is not a mandatory branch of the QR Dataset so operations like Metro and Buses do not usually need to use it. It was for this very reason such a segregation was done. But bear in mind, the branch should not be pruned altogether because the dataset has been made generic having the future in mind. If it has

to be implemented for Railways or Airways and other such operations, then entering personal details become a mandatory requirement for such operations. Even in case of urban transport operations it has come in handy for Personal Identification Verification for special passes like Senior Citizen Pass, Students Pass, etc. Please refer to the QR Ticketing System – Implementation Guidelines.

**Q. No. 7: Why is the person's mobile number included in the Common Data element of the QR Code Dataset? Shouldn't privacy be protected?**

**Answer** – This field has been included after much deliberation. If a robust QR Ticketing System has to be implemented, then some form of traceability is required when we want to trace counterfeit QRs back to the rogue user. Refunds and reconciliation also becomes easier when the Mobile No of the user is available. However, it is not required to encode the entire mobile number, i.e. all the 10 digits, into the QR in plain text. It can be encrypted by the TG. If the PTO needs it for some reconciliation purpose, then the TG can then decrypt it and provide it to the PTO. Furthermore, it must be added here that the App must ask the permission from the user to share its Mobile No. Please review the QR Ticketing System – Implementation Guidelines for the full 'Permissions List' that the user must agree to share before using the App.

**Q. No. 8: Can I purchase tickets for others and handover them even if I am not travelling?**

**Answer** – No. If one has to do this, then the QR Code has to be copied. This is a breach of the Security tenets laid out in the QR Specifications. Not just copying QR codes, even taking screenshots of QR Tickets should be disabled in Apps.

**Q. No. 9: Is the QR Code static or dynamic? Can an ETIM read the QR Code to validate it and / or print it?**

**Answer** – The QR Code is a Static QR Code. Yes, an ETIM can read and validate the code provided its implementation also supports parsing the SQDSR formatted QR payload as described in the Part II – Specifications. One of the main objectives that was laid out by the Ministry of Housing and Urban Affairs (MoHUA) was to have a single QR specification that shall be followed across the length and breadth of the country by all Service providers – transit or not.

**Q. No. 10: Can I do advanced booking of tickets in the Mobile App developed by CDAC?**

**Answer** – Yes. In the CDAC Mobile (and Webclient) APP – Travel MOZO – booking of tickets is presently allowed for up to 3 months in advance.

**Q. No. 11: What is the ISO standard followed for the QR Code Ticketing System?**

**Answer** – The ISO standard ISO-8877 has been referred to follow Standard QR implementation. Micro-QR, Aztec, etc. have not been used.

**Q. No. 12: What is the method for Group Ticket validation in the QR Ticketing System?**

**Answer** – This is an implementation-specific question. In group tickets, i.e. tickets where the Group Size > 1, the entry count by infra-red sensors is a possible solution in premises that support such things. But even in present day, groups are still allowed through special attended manual gates in Metro premises. For Buses and other modes of transport, the validation terminal must clearly indicate the number of persons in the ticket. For attended terminals this may be sufficient but in case of unattended ones the implementation must make use of some hardware support.

**Q. No. 13: What is the Error Correction Standard used in the QR Ticketing System?**

**Answer** – It is entirely up to the App Provider that does the QR encoding. It has been experimentally found that larger QRs (greater than 1500 bytes) take a much longer time when levels – Q and H are used. For such QRs, ‘L’ and ‘M’ must be used.

**Q. No. 14: What is the QR Version that I should use for issuing Paper QR tickets?**

**Answer** – Paper QR tickets are likely to be issued only at Station/Depot premises. So all Paper QRs can only be 1xN\_QR – where N can vary from 1 to 10. It means one PTO cannot issue a ticket that belongs to another PTO. Please refer to the Appendix II of the Part II – QR Security. The table lists scan times for different QR combos like 1x1, 1x2, 1x3 etc. Please choose a version that suits your requirements. For example, if you want to issue up to 6 tickets, you might want to choose Version 18. Conversely, if you can support up to Version 25, you can issue even 10 journey tickets in a single QR.

**Q. No. 15: What happens if I want to issue Paper QR Tickets for another operator if I have a Paid-2-Paid interchange with it?**

**Answer** – Paper QR tickets are likely to be issued only at Station/Depot premises. So all Paper QRs can only be 1xN\_QR – where N can vary from 1 to 10. It means one PTO cannot issue a ticket that belongs to another PTO. However, in case two PTOs have a Paid-2-Paid interchange arrangement, then obviously their Policy Fare rules have to be merged. The restrictions of QR Versions remain unchanged. If the PTO can support up to Version 25, it means the QR Payload can only be as big as 997 bytes.

**Q. No. 16: What happens if I want to get tickets booked for someone who is not tech-savvy?**

**Answer** – You must use the persons own mobile or her mobile, download the app and then book the ticket on that mobile itself. It is a security breach to copy a QR code and share it with someone else. The implementation of ‘Dynamic Data’ validation can easily catch this misappropriation. Feature-phone implementation of QR Codes is not supported by this Specification.

**Q. No. 17: Consider a scenario where person buys a QR Ticket some days in advance. But on the day of journey due to some unforeseeable reason, the QR fails validation What should the person do?**

**Answer** – This is a typical Customer Care scenario. Please refer to the QR Ticketing System – Implementation Guidelines to understand how it can be of help. There is also a new Transaction Type –



QR Duplicate – that has been added in the QR Specifications V1.1. One of the main purposes of including this new type is to enable issuing duplicate QR Tickets. As an aside, the validation of QR Tickets can happen in an offline mode too, provided the PTO has designed it in such a way. For offline validation, a local server at PTO level with minimum validation data may be considered to avoid misuse of Paper-based QR tickets.

**Q. No. 18: In case of advanced booking how does the dynamic update of date and time in ‘Dynamic Data’ element work?**

**Answer** – The ‘Dynamic Data’ element is relevant only for the Mobile app. Every time a person opens the App and uses the “My Bookings” tab to open a QR, the App renders a fresh QR with the updated date and time. These are system requirements that must be given to the App Provider. Dynamic Data – Update Datetime – is a field which is there to tackle counterfeit QRs. When a QR is stale – i.e. older than a few seconds (configurable) – the validator would deduce that the QR may have been copied from another phone and deny entry.

**Q. No. 19: If the PTO’s ETIMs does not support the Optical scanners (camera) what is the alternative?**

**Answer** – The QR Ticket has a parameter called Validator\_Info. It includes the scanning capabilities of the PTO that it had earlier provided to the TG in the form of Update Policy DB requests. Only 4 different modes of scanning are possible – Camera, BLE, WiFi and NFC. Some PTOs have GPRS-based scanners. One bit out of the second, third or fourth bits of Validator\_Info – which are now marked “Reserved for Future Use” – can be used to signify GPRS mode of scanning.

**Q. No. 20: How are Passes implemented in the QR Ticketing System?**

**Answer** – Passes are tagged as ‘Products’ in the QR Ticketing System. Please take a look at the various options available in the Product ID table in Part I – QR Code Dataset. There is also option for PTO to define its own Product range. For some implementation examples of Sample passes – Daily, Monthly, Senior Citizen – please refer to the QR Ticketing System Implementation Guidelines.

**Q. No. 21: If a person purchases a ticket from Station A to B station. In the middle of the journey, he decides to extend his journey to C station. Is it possible?**

**Answer** – No it is not possible. Imagine a QR Ticket as a full-fledged transit ticket which have been bought in advance. It is not an ad-hoc journey ticket. The person can get off at B and book a new ticket immediately from B to C. If he decides to proceed with over-stepping, then the Validation terminal at the Exit will catch this as an exception and deny exit.

**Q. No. 22: A person purchases a ticket from A to C, but gets down somewhere in the middle at B. Is it possible to refund fare between B and C?**

**Answer** – No it is not possible. Since it is a pre-paid ticket it should be considered as planned journeys. If the PTO has business case to provide refund, then the person may approach the Customer Care. The involvement of the TG is not possible here.

**Q. No. 23: Can any organization become the TG? Are there any certifications required to become a TG?**

**Answer** – There's no restriction as to who can become a TG and who cannot. The only requirement is possibly being that it is a technically sophisticated system and needs a fair understanding of present-day technologies (manpower requirement). Further, the organization has to establish trust relationships with many entities like the AFC, the Payment Service Provider, the App Provider, etc. Certifications are the only way to establish 'trust' in the digital world. So the TG must have them established before it can be of any service. Certificates shall be issued by a trusted organization appointed by MoHUA. Some well-known Certificate Authorities of India are shown in Figure 3 of Part II – QR Security Specifications.

**Q. No. 24: Who is responsible for issuing IDs to different entities like TG, PTO, APP in the eco-system?**

**Answer** – IDs shall be issued by some trusted organization appointed by MoHUA.

**Q. No. 25: Is it possible to generate QR code in ETM and payment by scanning the same using mobile?**

**Answer** – No payment must be done beforehand to get a QR Ticket. For Ad-hoc journey tickets, TOM can issue Paper QR Tickets which are again pre-paid tickets but PTO collects money by either Cash/Card or any other means.

**Q. No. 26: How can QR Code Tickets be validated in remote locations where there is no network?**

**Answer** – Although a hybrid model of validation – a combination of online and offline validation – is a preferred form of validation of QR Tickets, it can also happen in a completely offline mode. Please refer to the QR Ticket Generation and Validation Flowcharts in Part II – QR Security Specifications.

**Q. No. 27: How can validation of QR Tickets happen completely offline?**

**Answer** – QR Specifications – Part II actually talks about how the one time operations are done during the establishment of trust relation between the PTO (AFC System) and the TG (Ticket Issuing System). Digital Signature ensures authentication, non-repudiation and integrity of the QR Payload. The TG signs the ticket with its private key, the validator verifies it with the public key. So naturally there has to be an exchange of Certified Public Key generated at the TG. One reason of having the TG ID in the Common Data is when a QR arrives at the validator, it checks what the TG ID of this particular QR is, i.e. which TG issued this ticket. Suppose that ID is 7 then the validator will have to look up for the public key corresponding to TG ID 7. This lookup must be local to the terminal. When the signature matches it means the ticket is authentic. It also means the PTO cannot deny that ticket. Once the Signature is found to be valid, other TRM rules like valid Ticket Serial Number, Source/ Destination Station Codes, Date and Time of Journey must be verified. Serial numbers of QR Tickets need to be pushed to Validators if they can only lookup locally.

**Q. No. 28: Can the same QR Code be re-used? If not, what is the error displayed on the ETIM?**

**Answer** – No the same QR Code cannot be re-used. It is a security breach. QR Tickets have a unique Ticket Serial Number. Once a QR Ticket is used, the terminal “burns” that Serial number and marks it as “used”. So if the same ticket is presented again it will display – ‘No Valid Ticket Found’.

**Q. No. 29: Is Route selection mandatory for Bus Tickets? People who are new to city may not know the routes.**

**Answer** – No. Route selection is not mandatory for Buses or any other form of transport for that matter. The person should be able to specify just the Source and Destination and the system must be able to calculate how to connect them. If it requires multiple legs, then those options should be presented to the user by following route-agnostic ticketing approach. However, most of the bus transport operators have implemented route-based ticketing systems currently which is the reason, emphasis has been given to route-specific tickets.

**Q. No. 30: What data should be validated in Validating terminals or ETIMs?**

**Answer** – TRM or Transit Risk/Rules Management is a completely PTO-specific business rules implementation. But basic data like Digital Signature verification, Valid Ticket Serial Number, Journey Date and Time, Source and Destination Stations matching, etc. must always be performed.

**Q. No. 31: If the QR Ticket has multiple legs of journey, is there a need to perform those journeys in the same order?**

**Answer** – No there is no such limitation. Each journey has a separate set of Source and Destination. As long as the date and times are valid, there should not be any problem.

**Q. No. 32: Does including security of the QR Ticket with encryption increase the size of the QR Ticket?**

**Answer** – Yes it does to a certain extent. Encryption works on the principle of Cipher blocks. The increment usually occurs due to the padded bytes. Size calculation of QR tickets is dealt in much details in Part II – QR Security.

**Q. No. 33: Can I have a Digital Signature size that is smaller than 172 bytes?**

**Answer** – Yes. The 172-byte signature used in the examples in throughout the specifications are only for illustration.

**Q. No. 34: Is the QR Ticketing System inter-operable among different operators?**

**Answer** – Yes, so long as these Operators have followed the same specifications. One of the goals of this project was to build a common National-standard specification that can be utilized by all transit operators.

**Q. No. 35: Is there any limit to the number of tickets that the TG can issue in a single day?**

## QR Ticketing System – Implementation Guidelines

**Answer** – No. In this system, there is also no limit to how PTOs one single TG can serve. Naturally, the onus lies with the TG Provider how much load it can accommodate. It is a requirement that the PTO must specify to the TG Provider. If the PTO has a footfall of 5 lakh customer daily, then the direct business requirement on the TG is to have the capability to issue more than 5 lakh tickets daily. Furthermore, the Ticket Request-Response time should also be specified as a derived requirement.

\*\*\*\*\* **End of Document** \*\*\*\*\*