QR Parser Technical Documentation-iOS

MVisaQRParser

What's in QRParserSDK-*{x.y.z}*.zip :

Inside QRParserSDK-*{x.y.z}*.zip, there will be MVisaQRParser-*{x.y.z}*-DEBUG-SWIFT{swift-version}.zip and MVisaQRParser-*{x.y.z}*-PRODUCTION-SWIFT{swift-version}.zip zip files:

MVisaQRParser-*{x.y.z}*-DEBUG-SWIFT{swift-version}.zip

The content of package are to be used during the development phase of the issuer application. This package contain binaries for universal (debug/development) architecture supporting simulator and device, but not eligible for app store submission.

This folder will have following data :

Name	Description
MVisaQRParser.framework	Contains the public interface for integrating with MVisaQRParser.framework
SampleApp	Sample App which use MVisaQRParser.framework
CHANGE_LOG	Change log file to show all changes on MVisaQRParser.framework for each version

MVisaQRParser-*{x.y.z}*-PRODUCTION-SWIFT{swift-version}.zip

The content of package are to be used in the release version of the issuers application. The binaries in this package supports device only architectures and eligible for app store submission.

This folder will have following data :

Name	Description
MVisaQRParser.framework	Contains the public interface for integrating with MVisaQRParser.framework
CHANGE_LOG	Change log file to show all changes on MVisaQRParser.framework for each version

How to integrate MVisaQRParser.framework:

Following are steps to create Switf Project to use MVisaQRParser.framework for parsing QR code Data.

1. Xcode -> New -> Project -> Single View Application -> language Swift

2. Select Project -> Target -> General -> Embedded Binaries -> click on plus (+) -> Add Other -> Select MVisaQRParser.framework (if needed selected copy checkbox)

3. Open ViewController.swift and

```
import MVisaQRParser
```

4. In viewDidLoad method copy following code

```
let tlvString = "000201021647613688588820065204412153033565413123456789101.5802IN5912Corner Store6006Mumbai6304DFF1"

        QRCodeParser.parseQRData(qrCodeString: tlvString) { (parserResponse) in
            if ((parserResponse!.qrCodeData) != nil) {
                //read qrCodeData object
                print("QR Code Data: \n" + parserResponse!.qrCodeData!.description + "\n");
            }
            else {

                //show error
                 print("Error Code: \n" + parserResponse!.qrCodeError!.description + "\n");
            }

}
```

MVisaQRParser.framework API Detail:

This section consists of other important information related to the use and support of the SDK. As some of the information may be subjected to regular updates such as supported handset devices and IDE as well as list of software limitations, it will be advisable to refer to the information on the Visa Developers Portal for the latest update if needed.

API Class and Methods

The following describes the methods used in MVisaQRParser with their use cases, input & output parameters.

USE CASE	ENTRY POINT METHOD	INPUT PARAMETERS	OUTPUT PARAMETERS
Parse the QR code data string	QRCodeParser. parseQRData(qrCodeData)	The raw tlvString read from QR Code	QrCodeParserResponse object – Model object containing all the data present in the input qrCodeData string/list of error codes
Parse the QR code data string into a JSON String	QRCodeParser. parseQRData(qrCodeData)	The raw tlvString read from QR Code	QrCodeParserResponse object – Model object containing all the data present in qrCodeData object and all errors detail in qrCodeError object. you can get the parsed Json by calling qrCodeData.jsonOutput.

Class - QRCodeParser

The QRCodeParser class provides sync and async methods to parse QR code string, all the available methods are class level. To parse QR Code, call class method parseQRData: with required parameter.

Method - parseQRData

- Parse QR code string and provide String JSON object with parsed JSON data OR error JSON data.
- Input Parameter qrCodeString: QR code string which need to be parsed.
- Return string json object with parsed data or error json.
- Return String Json has 2 keys, qrCodeData and qrCodeError. qrCodeData key will have parse json and qrCodeError key will have error detail.
- If qrCodeError is nil, which means parsing is successfully done and qrCodeData will have parsed data.
- If qrCodeError is not nil which means QR data has some error and qrCodeError.errorCodes will provide array of error codes.

API Objects

The following describes the objects used by the methods in the SDK with their type, purposes and where they are used.

QRCodeParserResponse

ITEM	DESCRIPTION	FORMAT	METHODS

qrCodeData object of QRCodeData	Model object containing all the data present in the input tlv	CONDITIONAL Will be null in case of error in parsing	qrResponse.description will print all variable data.
qrCodeError object of QRCodeError	Model object containing error info like array of errorCodes, description.	CONDITIONAL Will be null in case of no errors found.	qrError.description will print all error codes. qrError.errorCodes will give you array of error code string.

*Note – Both qrCodeData and qrCodeError cannot be null at the same time.

QRCodeData

Variables	DESCRIPTION	FORMAT
String payloadFormatIndicator	Defines the format of the merchant data payload.	CONDITIONAL Will be null if Qr Code version is 00
String pointOfInitiation	Indicates the method by which the data is presented by the merchant. Indicates whether the data is static or dynamic. Refer to version 01 QR Spec	Optional
String mVisaMerchantId	Merchant Id	MANDATORY
String mVisaMerchantPan	Merchant Id converted to merchant Pan(16 digit) – should be used in mVisa transaction as merchant id	MANDATORY
String aliasID	Visa Alias Id	OPTIONAL
String masterCardPan1	Master card pan 1	OPTIONAL
String masterCardPan2	Master card pan 2	OPTIONAL
String npciid1	NPCI ID 1	OPTIONAL
String npciid2	NPCI ID 2	OPTIONAL
String merchantCategoryCode	Merchant category code as defined in As defined by ISO 8583:1993 for Card Acceptor Business Code.	MANDATORY
String currencyCode	Transaction Currency code as defined by ISO 4217	MANDATORY
String transactionAmount	Transaction Amount	Optional
String tipAndFeeIndicator	Tip and Fee Indicator "01": Indicates consumer should be prompted to enter tip "02": Indicates that merchant would mandatorily charge a flat convenience fee "03": Indicates that merchant would charge a percentage convenience fee	Optional
String convenienceFeeAmount	Convenience fee amount	CONDITIONAL
String primaryId	Primary Id value – populated only for Merchant Data version 00	OPTIONAL
String secondaryId	Secondary Id value – populated only for Merchant Data version 00	OPTIONAL

String convenienceFeePercentage	Convenience Fee Percentage	OPTIONAL
String countryCode	Country code. As defined by ISO 3166.	OPTIONAL
String merchantName	Merchant Name	OPTIONAL
String cityName	City Name	OPTIONAL
String postalCode	Postal code	OPTIONAL
String billId	Bill number (Part of Additional Data)	OPTIONAL
String mobileNumber	Mobile Number	OPTIONAL
String storeId	Store ID	OPTIONAL
String loyaltyNumber	Loyalty number1	OPTIONAL
String referenceId	Reference ID	OPTIONAL
String consumerId	Consumer ID	OPTIONAL
String terminalId	Terminal ID	OPTIONAL
String purpose	Purpose	OPTIONAL
String additionalConsumerDataRequest	Additional Consumer Data Request	OPTIONAL
String addDataMasterCard1	Additional data for Master card 1	OPTIONAL
String addDataMasterCard2	Additional data for Master card 2	OPTIONAL
String addDataNpci1	Additional data for Npci 1	OPTIONAL
String addDataNpci2	Additional data for Npci 2	OPTIONAL
String crc	Cyclic Redundant Check value	MANDATORY
boolean isPrimaryIdMandatory	Flag which defines whether primary ID is mandatory for consumer to enter or not	OPTIONAL
String primaryIdLength	Mandatory length for primary ID	OPTIONAL
boolean isSecondaryIdMandatory	Flag which defines whether Secondary ID is mandatory for consumer to enter or not	OPTIONAL
String secondaryIdLength	Mandatory length for secondary ID	OPTIONAL
boolean isBilldMandatory	Flag which defines whether Bill ID is mandatory for consumer to enter or not	OPTIONAL
boolean isMobileNumberMandatory	Flag which defines whether Mobile number is mandatory for consumer to enter or not	OPTIONAL
boolean isStoreIdMandatory	Flag which defines whether Store ID is mandatory for consumer to enter or not	OPTIONAL
boolean isLoyaltyNumberMandatory	Flag which defines whether Loyalty Number mandatory for consumer to enter or not	OPTIONAL
boolean isReferenceIdMandatory	Flag which defines whether Reference ID is mandatory for consumer to enter or not	OPTIONAL
boolean isConsumerIdMandatory	Flag which defines whether Consumer ID is mandatory for consumer to enter or not	OPTIONAL

boolean isTerminalIdMandatory	Flag which defines whether Store ID is mandatory for consumer to enter or not	OPTIONAL
boolean isPurposeMandatory	Flag which defines whether Store ID is mandatory for consumer to enter or not	OPTIONAL