RAVASVend Interface Specification

RAVASVend Web Service v1.06

S.L. Dedekind

7/15/2020

Document History

Date	Author	Comment
2020-08-01	SL Dedekind	1st Draft
2020-09-03	SL Dedekind	Renamed VoucherInfo class to VoucherInformation as conflict name with Method VoucherInfo
2021-01-15	SL Dedekind	IncompleteTransactionFault Example added
2021-09-29	SL Dedekind	Added VoucherCodeDetailExt & LiveVoucherListReq/Resp
2021-11-26	SL Dedekind	Added PinlessVoucher & PinlessVoucherProductList Methods
2022-02-18	SL Dedekind	Added AnyTimeVoucherRedeem and Anytime Samples
2022-03-11	SL Dedekind	Added AnyTimeVoucherLookup and Samples

Document Notes

Table of Contents

Contents

Introduction

The RAVASVend web service was created to allow Re-Sellers of VAS Products to purchase and vend these products and services from RA Cellular.

Prepaid Electricity is also a prepaid service but for simplicity the following is defined for the rest of the document.

Prepaid Vouchers Are Pin based vouchers that are not customer specific, i.e. any customer can purchase and use the voucher and no customer identification is required to create the pin. These are vouchers like the Vodacom, MTN, CellC, ValuePin, UniPin etc. They can be printed on a receipt and can be used by any customer.

Direct Top-ups are subscriber specific; they can be pin based or pin less transactions where the MSISDN, Smart Card Number, Account Number or other customer identifying property of the Customer is supplied to RAVASVend and the subscriber account is directly credited at the Utility or Pin is returned that can only be used by the customer.

Prepaid Electricity for example requires customer information to create the prepaid pin and can only be used by the Customer with whose information the pin was created and is thus declared a Direct TopUp and not a Prepaid Voucher.

"Customer" : is henceforth the name given to the end user that will use the Prepaid Voucher, Direct Top-up, Electricity Voucher or be the recipient of a Cash Withdrawal.

"Client" : is henceforth the name given to the Business or person that has an account on the RAVASVend system and connects to RAVASVend on behalf of the Customer to purchase Prepaid products. The Client may also be called the aggregator. The Client is normally a hosted server.

"Terminal" : is henceforth the name given to the device that is used to connect to the Client and performs the task of creating the GUI for the terminal operator and will Print/email/SMS/display the result to the Customer. This could be an ATM, Credit Card type terminal or website to name a few.

Prepaid Vouchers are products that do not require RAVASVend to have an active connection to the Product manufacturer. RAVASVend maintains a level of stock of these vouchers on the RAVASVend database and hence purchasing these vouchers should be without delay.

Direct Top-Ups and Financial Transactions are products that requires the RAVASVend system to have an active connection to Utilities such as Vodacom, MTN, CellC, DSTV, Eskom etc. servers to perform the service queries. Because of this connection the queries associated to the Direct Top-Ups may have some delay and in extreme cases may not be available at all.

It is because of this connection to Utility servers that extreme caution should be exercised when developing the integration system with specific attention given to the Advice scenario as will be explained later in this document.

RAVASVend is a SOAP service and all messages are XML formatted.

Connectivity to RAVASVend shall be over the internet from a static IP address using SSL.

Contents

1. User Requirements

1.1 Access requirements

The following information is required to create a TEST and then PRODUCTION account on RAVASVend for the integration to begin.

Company Name	
Company Registration Number	
Company VAT	
Company Physical Address	
Company Postal Address	
Technical Primary Contact Name	
Technical Primary Contact Tel	
Technical Primary Contact email	
Technical Secondary Contact Name	
Technical Secondary Contact Tel	
Technical Secondary Contact email	
Accounts Primary Contact Name	
Accounts Primary Contact Tel	
Accounts Primary Contact email	

After the information is supplied, a test account will be created, and the following will be supplied

a. Account Number
b. Server URL (WSDL can be downloaded from URL)
c. Authorization credentials (Username and Password)

1.2 Daily Recon Files

The RAVASVend system will create a daily recon file listing all billed transactions.

The client is required to provide an email or ftp or sftp location to where these files can be published daily.

Email address	
FTP URL FTP Credentials	
SFTP IP and Port SFTP Credentials SFTP Directory	

2. MsgID (what it is and its format)

The MsgID is a string that uniquely identifies each request sent to RAVASVend.

The MsgID should be a 20-digit string that is a concatenation of the current date and time and a 6 digit incrementing counter.

The format is hence yyyyMMddHHmmss + dddddd.

The 6-digit incrementing counter should start at 000000 and rollover at 999999.

Hence at 2016-03-22 11:33:25 AM with the counter at 123456 the MsgID would be "20160322113325123456"

If the next request is at 2016-03-22 15:36:27 PM the MsgID would be "20160322153627123457"

This allows 1 million unique requests every second to be generated.

If a request is received with a duplicate MsgID the response will be an error as duplicate MsgID's are not allowed.

The MsgID format is not bound to the previously described format (this is however suggested) It can for example be a UUID or incrementing integer number, but it must have the following characteristics

1. It is unique for each request
2. Length does not exceed 40 characters
3. Must consist only of numbers/letters and special characters from the ASCII code table, no control characters (for example NULL, SOH, LF etc.) are allowed.

3. VoucherCode (what it is and how it is used)

The VoucherCode is a unique identifier for each product type that is available on RAVASVend.

Each VoucherCode is configured to have a discount amount for the client, the discount values that a VoucherCode may have are:

Type	Description
Percentage	Percentage of Purchase Value.
Fixed	Fixed amount per transaction in Rands
CpkWh	Cents per Kilo Watt Hour
Cash	Percentage amount of Purchase Value if Cash is Tendered by the Customer
Credit	Percentage amount of Purchase Value if a Credit Card is Tendered by the Customer
Debit	Percentage amount of Purchase Value if a Debit Card is Tendered by the Customer
MaxComm	A Maximum Fixed value in Rands that caps the commission

Discounts for a vouchercode is the sum of the calculated Values (Percentage + Fixed + CpkWh + Cash|Credit|Debit) capped to a maximum value of MaxComm (if MaxComm is applicable)

**Attempts to circumvent limits and commissionable rates will cause the Client account to be suspended on RAVASVend. For example, if a specific vouchercode has a Fixed amount of R1 and the Client submits 10 x R100 requests for the same Meter Number instead of 1 x R1000 in order to get R10.00 commission instead of R1.00.*

4. ReceiptFormat (what it is and how it is used)

The receipt format is an enum that is specified in a request that returns a receipt. A printable receipt string is then formatted to the format specified by the ReceiptFormat requested.

The possible values are as follows

Name	Description
NONE	No Receipt will be returned, the Client can then create their own receipt using data returned in the response
EN_UNFORMATED	Receipt is returned in English with no Formatting for a 50mm thermal receipt printer
EN_FORMATED_80	Receipt is returned in English formatted for an 80mm thermal receipt printer
EN_FORMATED_50	Receipt is returned in English formatted for an 50mm thermal receipt printer

There are 2 options with the selection namely a Formatted and an Unformatted response. The formatted response will contain markers that can be used to print or format the receipt in a more readable manner, whilst the Unformatted receipt will simply contain the data.

The markers used to format the receipt are as follows

Name	**Description**	**Name**	**Description**
{US}	Underline Set	{UR}	Underline Reset
{AL}	Align Left	{AR}	Align Right
{AC}	Align Centre		
{BS}	Bold Set	{BR}	Bold Reset
{C0}	Character set 0 (normal)	{C1}	Character set 1 (double height)
{C2}	Character set 2 (double wide)	{C3}	Character set 3 (double height and wide)
{BC}	Barcode (the line contains a barcode)		

Each line will return only 1 instance of {US} or {UR} (one line will not contain both i.e the entire line is Underlined or Not)
Each line will return only 1 instance of {AL}, {AC} or {AR}. (the entire line is aligned either left/center/right)
Each line will return only 1 instance of {BS} or {BR}. (the entire line is bold / not bold)
Each line will return only 1 instance of {C0}, {C1}, {C2} or {C3}. (the entire line is either C1 / C2 / C3 / C4)

```
public enum ReceiptFormat
{
    /// <summary>
    /// No Receipt string will be returned
    /// </summary>
    NONE,
    /// <summary>
    /// English Unformated receipt
    /// 30 normal width characters in the horizontal direction
    /// </summary>
    EN_UNFORMATED,
    /// <summary>
    /// English Formated receipt for 80mm Printer
    /// 42 normal width characters in the horizontal direction
    /// </summary>
    EN_FORMATED_80,
    /// <summary>
    /// English Formated receipt for 50mm Printer
    /// 30 normal width characters in the horizontal direction
    /// </summary>
    EN_FORMATED_50,
}
```

The receipt lines are terminated by a new line character “\n” and although sometimes the carriage return character may be present “\r\n” it is not guaranteed, and hence the lines should be split on the new line character only.

Example of EN_FORMATED_50 Electricity Receipt	Example of EN_UNFORMATED Electricity Receipt

{C3}   TSHWANE
{AC}{C0}- - - - - - - - - - - - - - - -
Tax No :   400142267
2017/08/12   18:14:33
- - - - - - - - - - - - - - - -
Receipt No :   52710591
Meter No :   14225598649
    Electricity Pre-Paid
- - - - - - - - - - - - - - - -
Name :   DEDEKIND
Min. vend :   0.19
    Electricity Credit
- - - - - - - - - - - - - - - -
{C0}CREDIT TOKEN 1
{C3} 1926 0966 4374
{C3}   5135 2725
{C0}- - - - - - - - - - - - - - - -
Units (kWh) :   394.70
Arrears   :
Value Exc. :   701.75
VAT @ 14%   :   98.25
Total   :   800.00
- - - - - - - - - - - - - - - -
Operator :   R&A Cellular
ENQUIRIES: PHONE 076 387 9508
	TSHWANE
	- - - - - - - - - - - - - - - -
	Tax No : 400142267
	2017/08/12 18:14:33
	- - - - - - - - - - - - - - - -
	Receipt No : 52710591
	Meter No : 14225598649
	    Electricity Pre-Paid
	- - - - - - - - - - - - - - - -
	Name : DEDEKIND
	Min. vend : 0.19
	    Electricity Credit
	- - - - - - - - - - - - - - - -
	CREDIT TOKEN 1
	1926 0966 4374
	5135 2725
	- - - - - - - - - - - - - - - -
	Units (kWh) : 394.70
	Arrears :
	Value Exc. : 701.75
	VAT @ 14% : 98.25
	Total : 800.00
	- - - - - - - - - - - - - - - -
	Operator : R&A Cellular
	ENQUIRIES: PHONE 076 387 9508

5. Base Request and Response

All requests and responses to and from RAVASVend inherit from a BaseReq and BaseResp class. In this section we will briefly look at these two and explain their data members.

5.1 BaseReq

The BaseReq has 4 data members.

Name	Class Type	Explanation	Max Length
terminalMsgID	string	This is the terminal message ID supplied to the Client from the terminal for the current request, this terminalMsgID does not need to conform to the MsgID format required by the msgID and is simply informational and allows a more localized search criteria in the case of client discrepancies.	40 chars
terminalID	string	This is the terminal identifier to the Client (for example "ATM00235") this can be any string but should be chosen carefully as transactions are locked for a terminal ID for Revenue transactions.	40 chars
msgID	string	This is the MsgID of the request (please read about MsgID)	=20 chars
authCred	AuthCred	This is the credentials of the Client that authenticates the request on RAVASVend (Username and Password)	

As mentioned in the introduction a request for an Electricity Voucher or Direct Top-up requires RAVASVend to maintain a connection to the relevant Utility. Our experience has however shown that they do not always respond to requests and we have also experienced situations where a Utility is offline for several hours when they experience data center issues.

In the case that RAVASVend receives (for example) a CreditVend Request from a client, it will attempt to purchase the Electricity Voucher from the utility through this maintained connection. The terminalID is then locked until this request is completed and cannot perform any further transactions until this request is completed. If a terminalID is locked due to an incomplete request, the request is completed by performing an Advice request as described later in this document. (This is applicable to all Revenue type transactions)

Consequently, the following scenarios exists.

If for example the client is a website and all requests use the same terminalID then no further transactions are allowed until the blocking request is completed. It is thus advised to rather use a terminalID that identifies the Customer like their Account Number, Cellphone number or meter number.

If for example the client is a group of ATM's or terminals and they all use the same terminalID then the same situation can arise where they are all blocked from performing further transactions until the blocking transaction is completed. It is advised to rather use a unique identifier for each device.

Great care should hence be taken to choose a terminalID that does not interfere with other business functions.

The terminalID should however not be random for each request as it defeats the point of the Advice function and there will be monetary consequences for transactions not completed.

RA Cellular has several thousand machines in South Africa that each have a unique terminalID, if they perform a CreditVend request and there is no response then they are not allowed to perform further transactions until the transaction is completed.

The situation has arisen where the request was received by the Utility but before they could respond; their system went down for several hours. Hence the response was only received once their system came online and the transaction was billed. This is an unfortunate consequence of the Live Vending system and although this situation is extremely rare, RA Cellular together with the Utilities are working constantly to improve their systems reliability. The RAVASVend system will not lock a terminalID indefinitely. It will unlock a terminalID if an Advice Request is received and it determines that there is no possibility of returning a successful response in a reasonable time frame but it is the responsibility of the client to continually and repeatedly send the advice request to try and complete the request until RAVASVend decides to unblock the transaction.

```
/// <summary>
/// The Base Request for all requests
/// </summary>
public partial class BaseReq
{
    /// The MsgID Supplied by the Terminal to the Aggregator for this request
    public string terminalMsgID;
    /// The ID or Identifier of the Terminal to the Aggregator for this request
    public string terminalID;
    /// The Aggregator msgID for this request, this must be Unique for each request
    public string msgID;
    /// The Aggregator Credentials used to login to the server
    public AuthCred authCred;
}
```

```
public partial class AuthCred
{
    /// the Username of the Aggregator
    public string opName;
    /// the Password of the Aggregator
    public string password;
}
```

5.2 BaseResp

All responses from RAVASVend inherit from the BaseResp class

Name	Class Type	Explanation	Max Length
reqterminalMsgID	string	The terminalMsgID supplied in the request	40 chars
reqterminalID	string	The terminalID supplied in the request	40 chars
reqMsgID	string	The msgID supplied in the request	=20 chars
respDateTime	DateTime	The Date and time of the response	
hasFault	Boolean	A value that says if the response contains a fault	
fault	Fault	If hasFault then this class will provide all the information about the fault. If hasFault=false then this value will be null	
Server	string	Used internally by RAVASVend	40 chars

```
/// The Base Response for all responses
public partial class BaseResp
{
    /// The MsgID Supplied by the Terminal to the Aggregator for the request associated to this response
    public string reqterminalMsgID;
    /// The ID or Identifier Supplied by the Terminal to the Aggregator for the request associated to this response
    public string reqterminalID;            //the DeviceID of the Terminal from 3rdParty Vendor
    /// The Aggregator msgID for the request associated to this response
    public string reqMsgID;
    /// The Response Date and Time
    public System.DateTime respDateTime;
    /// Specifies whether the Response contains a Fault
    public Boolean hasFault;
    /// The Fault if the Response hasFault=True
    public Fault fault;
    /// Used internally by RAVASVend Server so that the response has a quick identifier which aggregation channel the response is allocated to
    public String Server;
}
```

6. Prepaid Electricity and Water

Prepaid electricity and Water (and GAS for that fact) has become a monthly necessity for many households. The ease of use, easy access and simple to use vending operation it offers has accelerated its acceptance in South Africa.

Prepaid Water and GAS use the same services and methods as Electricity and henceforth discussions and mentions of "Prepaid Electricity" shall be used to include that of Water and Gas as well.

RAVASVend offers several services for Prepaid Electricity and are enumerated as follows

Method Name	Description	Revenue Transaction
ConfirmCustomer	Search for a prepaid customer based on the meter identifier (meter Number)	No
CreditVend	Purchase Electricity Credits for a prepaid customer	Yes
TrialCreditVend	(Eskom Specific) Simulate Electricity Credit Purchase from Eskom	No
FBE	Obtain a Monthly Free Basic Electricity Credit for registered and qualifying customers	Yes
Reprint	Reprint the last Electricity Credit Purchase	No
UpdateMeterKey	Update the Meter Details based on new information from the Utility	No
ConfirmMeterDetails	(Eskom Specific) Obtain the full meter details to make a prepaid meter card	No
PayAcc	(Eskom Specific) Allows a customer to pay their account or debt at Eskom without purchasing electricity credits	Yes
CustReportFault	(Eskom Specific) Allows a customer to report a fault with their meter or supply at a terminal	No
VerifyToken	(Eskom Specific) Confirm that tokens printed are correct.	No

Almost all Electricity Transactions begin with the ConfirmCustomer method. The result of this method enumerates the capability of the utility that has ownership of the meter.

Many utilities (both municipal and private) use the prepaid electricity channel to recover debts and service fees so it may happen that a customer purchasing units for R100.00 may not receive any units as the amount is taken towards a debt payment.

6.1 ConfirmCustomer

Minimum Timeout (allSuppliers=False)	20 seconds
Minimum Timeout (allSuppliers =True)	65 seconds
TerminalID Locked	No

The ConfirmCustomer Method is used to confirm the identity and capability of the prepaid electricity meter. The customer will identify their meter and the response to the ConfirmCustomer Request will contain all the registered accounts that match the search criteria.

RAVASVend has in some cases multiple channels to the same utility; If for example a resident of Tshwane searches for their meter it may happen that three accounts are returned. All three account should have the same customer Name but will have different vouchercodes representing the different channels available. Of specific concern to the Client is which vouchercode to use for the subsequent CreditVend Request. RA Cellular has in cases such as Tshwane negotiated different rates with the different channels and each channel represented by the returned vouchercode has an associated discount rate (the commission earned for the request) RAVASVend will order the responses in such a way that the first entry in the returned array of accounts will have the best discount, the 2nd and 3rd vouchercodes (representing different channels) can be used to purchase credits in the case that the 1st is non-functional.

South Africa has not controlled the use of prepaid electricity meters in the sense that each utility (private, municipal or national) has created, implemented or made use of their own vending and metering system, a consequence of this is that duplicate meter numbers exists. As an example, an Eskom prepaid meter installed in the eastern cape has the same meter number as a municipal meter in Emalahleni. If this meter would be queried there will be two accounts returned (with different account holder names, Utility Names and vouchercodes) although this a very rare occurrence it has happened and causes the customer to purchase credits for a meter that does not belong to them if they are not provided the option to choose the correct Utility.

The ConfirmCustomer Method is a non-revenue transaction and as a result the requesting terminalID is not locked.

```
public partial class ConfirmCustomerReq : BaseReq
{
    /// If the amount of the subsequent purchase is known, then this is required as it assists in ordering the voucherCodes
    public decimal amount;
    /// if the vouchercode of the supplier is known then if this is supplied then only the relevant supplier will be queried
    /// otherwise this should be empty,null or 'UNKNOWN'
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// Return all suppliers regardless of previous request attempts and cache of results, Default must be false
    public bool allSuppliers;
}
```

```
public partial class MeterIdentifier
{
    /// Meter Serial Number
    public string msno;
    /// Supply Group Code  (should be 6 digits)
    public string sgc;
    /// Key Revision Number (should be 1 digit)
    public string krn;
    /// Tarriff Index (should be 2 digits)
    public string ti;
    /// algorithm Type (should be 2 digits)
    public string at;
    /// Token Technology (should be 2 digits)
    public string tt;
    /// Track2Data of the meter card excluding start and end sentinal
    public string track2Data;
}
```

```
public partial class ConfirmCustomerResp : BaseResp
{
    /// The Customer Information
    public ConfirmCustResult[] confirmCustResult;
}
```

```
public partial class ConfirmCustResult
{
    /// the vouchercode for this meter from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// extra functions that this voucherCode has access too
    /// Reprint Last Voucher, Update Meter Key, etc...
    /// this list will grow so its simply a comma sepereted list
    public string capability;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// customer information
    public CustDetail custDetail;
    /// utility information
    public UtilityDetail utilityDetail;

}
```

```
public class VoucherCodeDetail
{
    /// Unique VoucherCode for the product
    public String voucherCode;
    /// PastelCode unique to the vouchercode
    public String pastelCode;
    /// Percentage discount applicable to vouchercode
    /// if percentage = 1.5 and vend for R100 is requested then transaction cost will be R98.50
    public Decimal percentage;
    /// Fixed discount applicable to vouchercode
    /// if fixedAmt = R2.00 then regardles of request amount, tranacion cost will be requestamount  - R2.00
    public Decimal fixedAmt;
    /// Cents per Kilo Watt Hour
    /// TranasctionCost is Request amount  - (units returned * cperkwh/100)
    public Decimal cperkwh;
    /// Bank Costs recoverable for Cash Tendered by customer
    /// specified as a % of request amount
    public Decimal cash;
    /// Bank Costs recoverable for Debit Card Tendered by customer
    /// specified as a % of request amount
    public Decimal debitCard;
    /// Bank Costs recoverable for Credit Card Tendered by customer
    /// specified as a % of request amount
    public Decimal creditCard;
    /// If specified (maxCommission > 0) then transactionCost can be a minimum of requestAmount - maxCommission
    /// so regardless of percentage/cash/debitcard etc
    public Decimal maxCommission;
    /// whether VoucherCode is enabled for customer or not
    public Boolean enabled;
}
```

```
public partial class CustDetail
{
    public string name;         //customer name
    public string address;      //customer address

    public bool fbeDue;             //is FBE Due
    public bool fbeDueSpecified;    //is FBE Due specified

    public decimal maxVendAmt;      //maximum vend amount 0-infinite
    public bool maxVendAmtSpecified;

    public decimal minVendAmt;      //minimum vend amount
    public bool minVendAmtSpecified;

    public decimal debts;           //outstanding debts the customer has
    public bool debtsSpecified;
}
```

```
public partial class UtilityDetail
{
    public string name;             //The utility Name
    public string address;          //The utility Address
    public string taxRef;           //The Utility Tax Ref No.
    public string contactNo;        //The customer care contact number for the Utility
}
```

The Capability String returned for a customer enumerates the capability of the utility.

The String returned may be something like “REPRINT,FBE” meaning that both Reprint and FBE Methods are supported by the Utility. The following possibilities can be present in the capability String (comma separated list)

FBE,REPRINT,UMK,TRIAL,PAYACCOUNT,VERIFYTOKEN,FAULTREPORT,METERDETAIL

RAVASVend Keeps and maintains a database of meters and their respective Utilities. This database is used to only query the meters utility and not all Utilities when searching for a meter. This database can however be ignored by specifying (allSuppliers=true) in the request. This may however not be the default value and should only be used in extreme cases after a ConfirmCustomer response with allSuppliers=false returned incomplete or incorrect meter information.

6.2 Credit Vend

Minimum Timeout	35 Seconds
TerminalID Locked	Yes (transaction must be completed)

The Creditvend Request is used after the ConfirmCustomer Request to purchase units for a prepaid meter.

A Creditvend is revenue transaction and thus must be completed.

The terminalID supplied in the request will be locked from further requests until the transaction is completed.

```
public partial class CreditVendReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode for the Utility Required
    /// for creditvend it cannot be null or UNKNOWN as these will with other unconfigured vouchercodes be rejected
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    public decimal purchaseValue;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public enum TenderType
{
    CASH,
    CHEQUE,
    CREDITCARD,
    DEBITCARD,
    VOUCHER,
    EFT,
    OTHER,
}
public enum TenderFromAccount
{
    NONE,
    SAVINGS,
    CHEQUE,
    CREDIT,
}
public class Tender
{
    public TenderType tenderType;

    /// First6 and last 4 digits of card, other numbers should be replaced with 0
    public string tenderPAN;

    public TenderFromAccount fromAccount;

    /// additional information that can be added to the tender
    /// for CHEQUE this could be the cheque number or the voucher number etc...
    public string tenderRef;

}
```

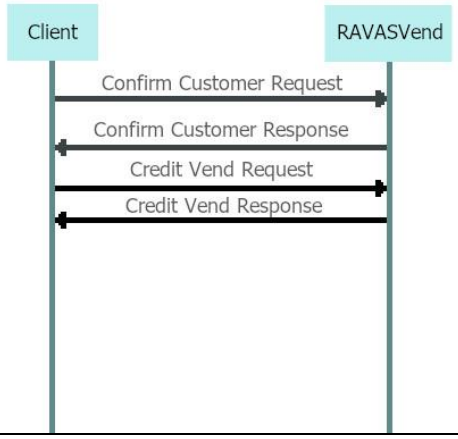
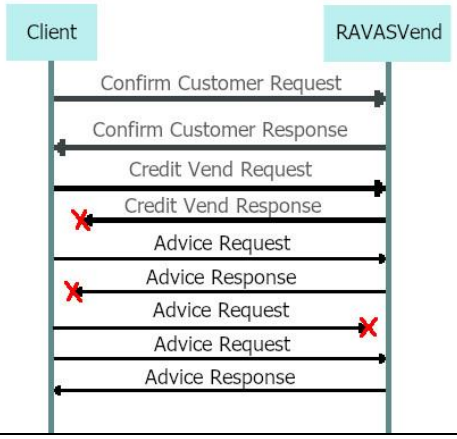
```
public partial class CreditVendResp : BaseResp
{
    /// The Customer Information
    public ConfirmCustResult confirmCustResult;

    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;

    /// Standard Token Transactions that contain the Recharge Cipher
    public StandardTokenTx[] standardTokenTx;
    /// Key Change Tokens that were returned with the vend
    public KeyChangeTokenTx[] keyChangeTokenTx;
    /// Fixed Charges transactions like Service Fee's and Connection Fees
    public FixedChargesTx[] fixedChargesTx;
    /// Debt Payments or contributions to outstanding debts at the Utility
    public DebtPaymentTx[] debtPaymentTx;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// for example a request for purchaseValue=R50.00 May be calculated at R49.00
    /// Hence R49.00 will be subtracted from the Aggregator Account at RACellular
    public Decimal TransactionCost;
}
```

```
public partial class StandardTokenTx
{
    /// The Type of Token, Normal Sale, FBE, Over Recovery etc...
    public string desc;
    /// The Quantity of Units of the Resource Purchased
    public decimal units;
    /// The type of Resource the Units represents (kWh / kL) etc.
    public string unitsISOUnit;
    /// The Value of the Token (inlcuding Tax) in Rands (0.00)
    public decimal amount;
    /// The VAT portion on the Token (VAT) in Rands (0.00)
    public decimal vat;
    /// The Receipt Number for the Token (Customer Receipt number from Utility)
    public string receiptNumber;
    /// Token to be printed on receipt
    /// this can be 20 digit STS token or proprietary token of any length
    /// this could also be "No Token","" or similar string for Smart Meters.
    public string token;
    /// The Tarriff applied for the Token (may be empty)
    /// this can be a multi-line string with lines terminated by CR and NL character '\r\n'
    /// For example
    /// Business 1000.0-2000.0kWh:692.46@ 2.1274/kWh
    /// Business 2000.0-3000.0kWh:153.39@ 2.2032/kWh
    public string tarriff;
}
```

```
public partial class KeyChangeTokenTx
{
    /// The Description of Key Change Token, Update Meter Key, etc...
    public string desc;
    /// Old Supply Group Code
    public string oldSGC;
    /// New Supply Group Code
    public string newSGC;

    /// Old Tariff Index
    public string oldTI;
    /// New Tariff Index
    public string newTI;

    /// Old Key Revision Number
    public string oldKRN;
    /// New Key Revision Number
    public string newKRN;

    /// 1st Token
    public string token1;
    /// 2nd Token
    public string token2;

    /// The Description of the power Limit Token (may not be present)
    public string pwrLimitDesc;
    /// Power Limit Token (may not be present)
    public string pwrLimit;

}
```

```
public partial class FixedChargesTx
{
    /// The Receipt Number for the Fixed Charge (Customer Receipt number from Utility)
    public string receiptNumber;
    /// The Value of the Fixed Charge (including Tax) in Rands (0.00)
    public decimal amount;
    /// The VAT portion on the Charge (VAT) in Rands (0.00)
    public decimal vat;
    /// Fixed Charge Description (e.g. "Radio Tax","Litter Tax","Service Charge","Connection Fee" etc)
    public string desc;
}
```

```
public partial class DebtPaymentTx
{
    /// The Receipt Number for the Debt Payment (Customer Receipt number from Utility)
    public string receiptNumber;
    /// The Value of the Debt Payment in Rands (0.00),
    /// A Debt payment cannot have VAT as it is only a payment to towards a previously generated invoice
    public decimal amount;
    /// The Remaining Debt Balance
    public decimal remainder;
    /// Whether the remainder is specified or not
    public bool remainderSpecified;
    /// Debt Payment Description (A description to what the debt is e.g. "Fraud"...)
    public string desc;
}
```

	
Successful Credit Vend (Happy Path)	Credit Vend Example Requiring Advice

6.3 FBE

Minimum Timeout	35 Seconds
TerminalID Locked	Yes (transaction must be completed)

The FBE Request is used after the ConfirmCustomer Request to obtain the free units for a customer who is on an indigent tariff and receives free units monthly from their utility.

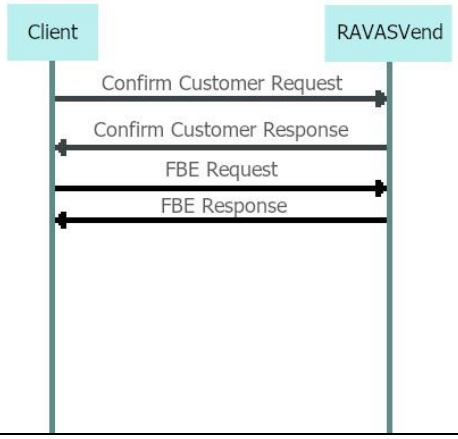
An FBE is classified as a revenue transaction and thus must be completed.

FBE Requests are repeatable on some Utilities like Eskom but on many others they are not and subsequent requests for FBE after a token has been generated will fail. The terminalID supplied in the request will be locked from further requests until the transaction is completed.

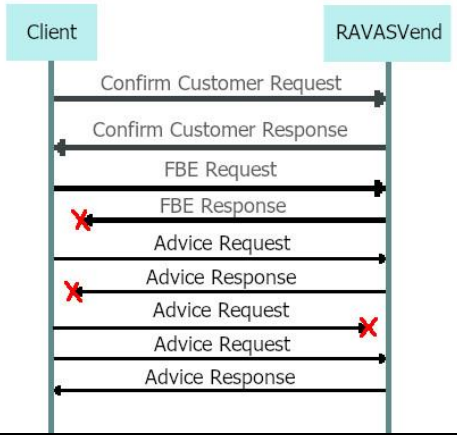
FBE is only available from certain Utilities. (capability will contain the value "FBE" if it is supported)

```
public partial class FBEReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode for the Utility Required
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class FBEResp : BaseResp
{
    /// The Customer Information
    public ConfirmCustResult confirmCustResult;

    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;

    /// Standard Token Transactions that contain the Recharge Cipher
    public StandardTokenTx[] standardTokenTx;
    /// Key Change Tokens that were returned with the vend
    public KeyChangeTokenTx[] keyChangeTokenTx;
    /// Fixed Charges transactions like Service Fee's or Connection Fee's
    public FixedChargesTx[] fixedChargesTx;
    /// Debt Payments or contributions to outstanding debts at the Utility
    public DebtPaymentTx[] debtPaymentTx;

    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;

    /// The Receipt string formatted to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
}
```



Successful FBE (Happy Path)



FBE Example Requiring Advice

6.4 Trial Credit Vend

Minimum Timeout	35 Seconds
TerminalID Locked	No

The Trial Credit Vend Request is available from some utilities to allow a customer to see how many units they would receive for a specified amount before the CreditVend is requested. The request parameters are identical to that of the CreditVend but the trial vend is not a revenue transaction.

Trial CreditVend is only available from certain Utilities. (capability will contain the value "TRIAL" if it is supported)

```
public partial class TrialCreditVendReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode for the Utility Required
    /// for trialcreditvend it cannot be null or UNKNOWN as these will with other unconfigured vouchercodes be rejected
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    public decimal purchaseValue;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class TrialCreditVendResp : BaseResp
{
    /// The Customer Information
    public ConfirmCustResult confirmCustResult;
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// Standard Token Transactions that contain the Recharge Cipher
    public StandardTokenTx[] standardTokenTx;
    /// Key Change Tokens that were returned with the vend
    public KeyChangeTokenTx[] keyChangeTokenTx;
    /// Fixed Charges transactions
    public FixedChargesTx[] fixedChargesTx;
    /// Debt Payments or contributions to outstanding debts at the Utility
    public DebtPaymentTx[] debtPaymentTx;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost (for example a request for purchaseValue=R50.00 May be calculated at R49.00
    /// Hence R49.00 will be subtracted from the Aggregator Account at RACellular
    public Decimal TransactionCost;
}
```

6.5 Reprint

Minimum Timeout	35 Seconds
TerminalID Locked	No

The reprint request is available from some utilities, it allows the client to reprint the last token for a customer that has perhaps lost their receipt, or the receipt is unreadable.

The returned token will be the last token the customer purchased from ANY location where the utility is supported.

Reprint is only available from certain Utilities. (capability will contain the value "REPRINT" if it is supported)

```
public partial class ReprintReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode for the Utility Required
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class ReprintResp : BaseResp
{
    /// The date of the vend (original date and not reprint date)
    public DateTime receiptDate;
    /// The Customer Information
    public ConfirmCustResult confirmCustResult;
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// Standard Token Transactions that contain the Recharge Cipher
    public StandardTokenTx[] standardTokenTx;
    /// Key Change Tokens that were returned with the vend
    public KeyChangeTokenTx[] keyChangeTokenTx;
    /// Fixed Charges transactions like Radio Tax and Litter Tax
    public FixedChargesTx[] fixedChargesTx;
    /// Debt Payments or contributions to outstanding debts at the Utility (EDM)
    public DebtPaymentTx[] debtPaymentTx;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
}
```

6.6 Update Meter Key

Minimum Timeout	35 Seconds
TerminalID Locked	No

Update Meter Key is used to update Meter STS data when it has been updated on the Utility Server.

Update Meter Key tokens are in some cases returned together with Credit Vend Tokens but some Utilities like Eskom require the Update Meter Key transaction to be done separately to ensure the customer receives the Key Change Tokens.

If the UMK tokens are returned with a CreditVend then these should be printed 1st on the receipt as these should be entered into the meter before any other credit tokens.

Updating a Meter's STS data will synchronize it with that of the Utilities database and thus tokens generated for the meter will function as expected. If a token is generated for a meter and there is a data mismatch between the meter and the utility server then the tokens will be rejected by the meter.

UMK tokens are normally 2 sets of 20 digits. In some cases, a third power limit token will also be returned.

Update Meter Key is only available from certain Utilities. (capability will contain the value "UMK" if it is supported)

```
public partial class UpdateMeterKeyReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode
    public string voucherCode;
    /// This should be MeterDetail and have all the meter information to be updated
    public MeterIdentifier meterIdentifier;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class UpdateMeterKeyResp : BaseResp
{
    /// The Customer Information
    public ConfirmCustResult confirmCustResult;
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// Key Change Tokens that were returned with the vend
    public KeyChangeTokenTx[] keyChangeTokenTx;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
}
```

6.7 Verify Token

Minimum Timeout	35 Seconds
TerminalID Locked	No

An Eskom Specific use case that allows a customer to verify that the tokens that they have received via SMS or Printed on a receipt are in fact correct and valid.

Update Meter Key is only available from certain Utilities. (capability will contain the value "VERIFYTOKEN" if it is supported)

```
public partial class VerifyTokenReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode for the Utility Required
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;

    /// 1st Token
    public string token1;
    /// 2nd Token (should be null or empty if not a KC token)
    public string token2;

    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class VerifyTokenResp : BaseResp
{
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// token ID
    public string tokenID;
    /// transfer Amount
    public decimal transferAmt;
    /// token Type
    public string tokenType;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
}
```

6.8 PayAcc

Minimum Timeout	35 Seconds
TerminalID Locked	Yes (transaction must be completed)

The Pay account method allows a customer to pay their Utilities account without returning any Credit Units.

The Method is rarely used as most utilities recover outstanding debts and account payments using partial or full deductions from CreditVend's.

Pay Account is only available from certain Utilities. (capability will contain the value "PAYACCOUNT" if it is supported)

```
public partial class PayAccReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode for the Utility Required
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// The Information Uniquely Identifying the Customer
    public CustomerIdentifier customerIdentifier;
    /// The Account Type being Paid
    /// PayAccount,ServiceChrg,DebtRecovery,Upgrade,Connection,Tamper,Conversion
    public String payAccountType;
    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    public decimal purchaseValue;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;

    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class PayAccResp : BaseResp
{
    /// The Customer Information
    public ConfirmCustResult confirmCustResult;
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// Fixed Charges transactions like Radio Tax and Litter Tax
    public FixedChargesTx[] fixedChargesTx;
    /// Debt Payments or contributions to outstanding debts at the Utility (EDM)
    public DebtPaymentTx[] debtPaymentTx;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost (
    /// for example a request for purchaseValue=R50.00 May be calculated at R49.00
    /// Hence R49.00 will be subtracted from the Aggregator Account at RACellular
    public Decimal TransactionCost;
}
```

6.9 ConfirmMeterDetails

Minimum Timeout	35 Seconds
TerminalID Locked	No

The method allows the client to obtain full STS meter information for the requested meter. The client can then use the returned information to create a magnetic meter card for the customer if the terminal is equipped with a magnetic card writer.

Confirm Meter Details is only available from certain Utilities. (capability will contain the value "METERDETAIL" if it is supported)

```
public partial class ConfirmMeterDetailsReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode for the Utility Required
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class ConfirmMeterDetailsResp : BaseResp
{
    /// The Customer Information
    public ConfirmCustResult confirmCustResult;
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// The Receipt format Received in the ConfirmMeterDetailsReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
}
```

6.10 CustReportFault

Minimum Timeout	35 Seconds
TerminalID Locked	No

The Eskom specific CustReportFault use case allows an Eskom Customer to report a fault of their Meter, Supply or other error at a terminal as opposed to calling or visiting an Eskom Customer Support Centre.

CustReportFault is only available from certain Utilities. (capability will contain the value "FAULTREPORT" if it is supported)

```
public partial class CustReportFaultReq : BaseReq
{
    /// VoucherCode is required, It is required to call ConfirmCustomer First to obtain the voucherCode for the Utility Required
    public string voucherCode;
    /// The Information Uniquely Identifying the Meter
    public MeterIdentifier meterIdentifier;
    /// The Information Uniquely Identifying the Customer
    public CustomerIdentifier customerIdentifier;
    /// Type of Fault experienced by client
    public string faultType;
    /// a Description of the fault
    public string desc;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class CustReportFaultResp : BaseResp
{
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// Reference Number for the fault report
    public string refNo;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
}
```

Acceptable faultTypes are

NetworkFaultReport	DisplayLightsButtons	FireWaterDamage
KeepsTripping	MeterDead	NoTrip
SeriousBoxDamage	TokensNotWorking	IncorrectSGC
IncorrectTI	ConvertedFrmConventional	MeterChangedOut
NewInstallation		

Please refer to Eskom description of errors that are available for reporting at

http://www.prepayment.eskom.co.za/xmlvend/EskomXMLvendExtensions_1-8.doc

7. DSTV

Customers who have Multichoice services can pay for their subscription services in a prepaid manner.

This includes normal DSTV (all packages), GoTV as well as BoxOffice Credits.

DSTV payments is a 2-step process where firstly the identity of the customer and amount is confirmed by obtaining a quote from DSTV using DSTVGetSubscriberInfo. After this a payment can be made using the DSTVVendorPaymentTransaction.

7.1 DSTVGetSubscriberInfo

Minimum Timeout	40 seconds
TerminalID Locked	No

A DSTV account holder can identify themselves using either their DSTV Customer Number, their SmartCard Number or their personal ID number that is registered as the subscriber on the DSTV database.

```
public partial class DSTVGetSubscriberInfoReq : BaseReq
{
    /// SUBS or TVOD (SUBS = DSTV or GOTV, TVOD = BOXOFFICE)
    public string paymentType;

    /// Customer Identifier can be ID Number,SmartCardNumber or Customer Number)
    public string customerIdentifier;
}
```

```
public partial class DSTVGetSubscriberInfoResp : BaseResp
{
    /// the vouchercode for this product
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// Customer Last invoiced account number for TVOD or SUBS
    public string accountNumber;
    /// Customer Last invoiced account status for TVOD or SUBS
    /// Normally one of the following (Cancelled,Closed,Open,Overdue,Suspended)
    public string customerAccountStatus;
    public string customerCellNo;
    public string customerInitials;
    /// Customer account number
    public string customerNumber;
    public string customerSurname;

    /// for SUBS : maxLimitAmount = Max Amount field of VPS quote
    /// for TVOD : maxLimitAmount = Max BO payable amount field of VPS quote
    public decimal maxLimitAmount;

    /// for SUBS : paymentAmount = Retail Quote Total of VPS quote
    /// for TVOD : paymentAmount = BO Account Balance of VPS quote
    public decimal paymentAmount;
}
```

7.2 DSTVVendorPaymentTransaction

Minimum Timeout	40 seconds
TerminalID Locked	Yes (transaction must be completed)

After the DSTVGetSubscriberInfo Response is received then the quote amount should be used to ask the customer how much they would like to pay (Partial payments are allowed)

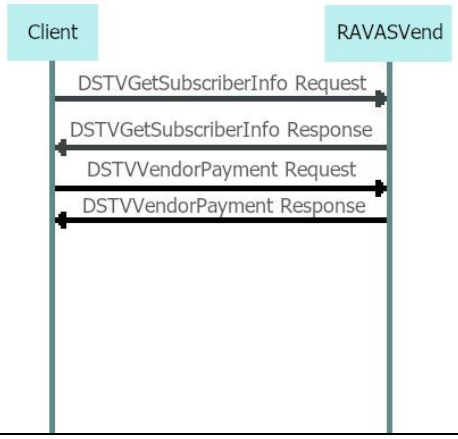
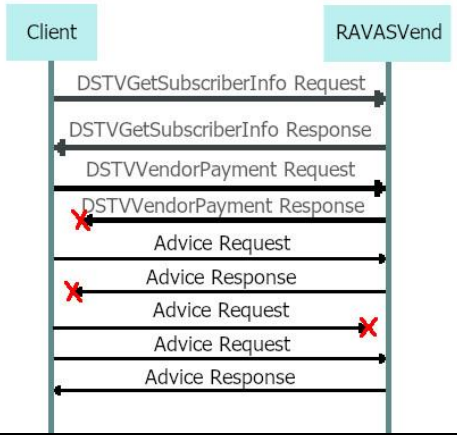
For SUBS the paymentAmount is the amount that is required to be paid for continued use of the services.

For TVOD the paymentAmount is the amount that has been used to rent movies. (if the amount is not paid then it would be added to the monthly debit order (for example))

Having confirmed the customer's identity and amount the DSTVVendorPaymentTransaction request should be used to pay for the required service.

```
public partial class DSTVVendorPaymentTransactionReq : BaseReq
{
    /// VoucherCode is required, It is required to call DSTVGetSubscriberInfoReq First to obtain the voucherCode for the Utility Required
    public string voucherCode;

    /// the Amount that is being purchased (This does not include the fee)
    /// This is the amount that is being sent to DSTV for payment
    public decimal purchaseValue;

    /// the fee being charged by the terminal for the payment
    /// So if purchaseValue = R90 and paymentFee = R5 then the receipt returned will have a total of R95
    /// but DSTV will only be paid R90 and the account at RAVASVEnd will only be debited for the R90
    /// the R5 service fee is simply part of the request to correctly create the receipt to be printed
    /// If RAVASVend is not being used to generate receipts then this can be defaulted to 0.
    public decimal paymentFee;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;

    /// Customer Last invoiced account number for TVOD or SUBS
    /// returned from DSTVGetSubscriberInfoResp
    public string accountNumber;
    /// Customer Last invoiced account status for TVOD or SUBS
    /// returned from DSTVGetSubscriberInfoResp
    public string customerAccountStatus;
    /// returned from DSTVGetSubscriberInfoResp
    public string customerInitials;
    /// returned from DSTVGetSubscriberInfoResp
    public string customerSurname;
    /// returned from DSTVGetSubscriberInfoResp
    public string customerCellNo;
    /// Customer account number
    /// returned from DSTVGetSubscriberInfoResp
    public string customerNumber;
    /// for SUBS : paymentAmount = Retail Quote Total of VPS quote
    /// for TVOD : paymentAmount = BO Account Balance of VPS quote
    /// returned from DSTVGetSubscriberInfoResp
    public decimal paymentAmount;
    /// SUBS or TVOD
    public string paymentType;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class DSTVVendorPaymentTransactionResp : BaseResp
{
    /// Customer account number
    public string customerNumber;
    /// Customer Last invoiced account number for TVOD or SUBS
    public string accountNumber;
    /// Receipt number for this payment
    public string receiptNumber;
    /// the vouchercode for this customer from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost (
    /// for example a request for purchaseValue=R50.00 May be calculated at R49.00
    /// Hence R49.00 will be subtracted from the Aggregator Account at RACellular
    public Decimal TransactionCost;
}
```

	
DSTV Payment example (Happy Path)	DSTV Payment example requiring Advice

8. Prepaid Vouchers

8.1 Introduction

Several prepaid vouchers for several suppliers are available from RAVASVed, these include the cellular providers like Vodacom, MTN, CellC, Telkom Mobile but also Prepaid Electricity vouchers that require the customer to obtain the units themselves via USSD code like the ValuePin vouchers.

What all the vouchers have in common is they have a Value, Serial Number and Pin Number. The Customer is then able to redeem the value of the voucher by following the recharge instruction (this could be by dialing a USSD code, entering the Pin into their Television Set Top Box or logging into their account at the supplier's website and redeeming the voucher)

There are four methods that are available to enable the Client to sell prepaid vouchers from their system, and they are discussed next.

Method Name	Description	Revenue Transaction
VoucherInformation	Obtain voucher information for a single voucher type specified in the request by the VoucherCode	No
VoucherList	Return a List of all vouchers available	No
PurchaseVoucher	Purchase a single voucher specified in the request by the VoucherCode	Yes
PurchaseMultiVoucher	Purchase multiple vouchers specified in the request by an Order of VoucherCodes.	Yes

8.2 VoucherInfo

Minimum Timeout	10 seconds
TerminalID Locked	No

VoucherInfo Method is used to obtain all the voucher detail about a single vouchercode specified in the request

```
public partial class VoucherInfoReq : BaseReq
{
    /// VoucherCode of the specific voucher to which voucherinfo is required
    public string voucherCode;
}
```

```
public partial class VoucherInfoResp : BaseResp
{
    /// VoucherInformation for the specified voucherCode in the request
    public VoucherInformation voucherInformation;
}
```

```
public partial class VoucherInformation
{
    /// Unique voucher identifier for the voucher type
    /// 10 chars
    public string voucherCode;
    /// Long Voucher Name for the voucher type ("Telkom Mobile R29 Data")
    public string voucherName;
    /// Short Voucher name for the voucher type (max=20 characters) ("Vodacom R5")
    public string voucherNameShort;
    /// Face Value of the voucher
    public decimal voucherValue;
    /// Barcode registered for the voucher type
    public string barcode;
    /// Readable recharge instruction for the voucher type that the customer should follow to redeem it
    public string rechargeInstruction;
    /// USSD or URL code to redeem the pin number
    /// rechargeCode will always contain the String "PIN" that should be substituted with the PinNumber of the voucher
    public string rechargeCode;
    /// If applicable the promotion line simply communicates promotional data from the voucher creator (Vodacom for e.g.)
    /// to the customer.
    /// This is normally "Thank You" but can be a multi line string
```

```
    public string promotion;
    /// whether VAT is applicable to the voucher type
    public bool vat;
    /// VAT inclusive cost price of the voucher type for the aggregator
    public decimal price;
    /// whether VoucherCode is enabled for customer or not
    public Boolean enabled;
}
```

8.3 VoucherList

Minimum Timeout	10 seconds
TerminalID Locked	No

The method will return voucher information for all vouchers that are available on the RAVASVend Server.

The response envelope can be quite large and as such care should be taken in configuring binding settings (or similar) as having a small limit on parameters such as maxReceivedMessageSize (.net binding) will throw an exception

```
public partial class VoucherListReq : BaseReq
{
}
```

```
public partial class VoucherListResp : BaseResp
{
    /// VoucherInformation for all available vouchers on RAVASVend
    public VoucherInformation[] voucherInformation;
}
```

8.4 PurchaseVoucher

Minimum Timeout	10 seconds
TerminalID Locked	Yes

The method PurchaseVoucher should be used to purchase a single prepaid voucher. The response has the functionality to return a formatted receipt that can be printed.

```
public partial class PurchaseVoucherReq : BaseReq
{
    /// VoucherCode is required, Uniquely identifies the voucher required
    public string voucherCode;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class PurchaseVoucherResp : BaseResp
{
    /// VoucherInformation for the specified voucherCode in the request
    public VoucherInformation voucherInformation;
    /// Unique Voucher with PinNumber
    public Voucher voucher;
    /// The Receipt format Received in the PurchaseVoucherReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information about the Voucher
    public string smsreceipt;
    /// The Aggregators Transaction Cost
    public Decimal TransactionCost;
}
```

```
public partial class Voucher
{
    /// Unique voucherCode for the voucher type
    public string voucherCode;
    /// Serial Number uniquely identifying the voucher
    public string serial;
    /// Secret Pin number of the voucher
    public string pin;
    /// DateTime the voucher was purchased
```

```
    public DateTime datePurchased;
    /// the MsgID used by the aggregator in the request to purchase the specific voucher
    public string reqMsgID;
    /// The Aggregators Cost price inclusive of VAT for the specific voucher
    public decimal price;

}
```

8.5 PurchaseMultiVoucher

Minimum Timeout	10 seconds
TerminalID Locked	Yes

The method PurchaseMultiVoucher should be used to purchase 1 or more prepaid airtime vouchers. The response will only contain the voucher data and not a formatted receipt and as such the terminal should format the data for the customer.

```
public partial class PurchaseMultiVoucherReq : BaseReq
{
    /// the Order compiled of all the vouchers required.
    public VoucherOrder[] order;

    /// If the order Cannot be completed with all vouchers required by RAVASVend then the entire order should be cancelled.
    /// if partailFail=false then if some vouchers can be purchased then the order will be edited to include only those that are available.
    public bool partailFail;
}
```

```
public partial class VoucherOrder
{
    /// VoucherCode is required, Uniquely identifies the voucher required
    public string voucherCode;

    /// Positive value required specifying how many vouchers of the type are required.
    public int qty;
}
```

```
public partial class PurchaseMultiVoucherResp : BaseResp
{
    /// VoucherInformation for all the voucherCodes specified in the request
    public VoucherInformation[] voucherInformation;

    /// Unique Vouchers with PinNumber
    public Voucher[] voucher;

    /// The Aggregators Transaction Cost
    public Decimal TransactionCost;
}
```

9. LiveAirtime

Minimum Timeout	60 seconds
TerminalID Locked	Yes (transaction must be completed)

Live airtime allows the client to purchase prepaid credits directly to a supplied MSISDN. This is normally called a Pin-Less airtime purchase.
This allows a client to accept a payment from a customer without having to provide a printed receipt that contains a PIN number as the customer will receive an SMS notification from the network confirming the purchase.

It also allows a customer to purchase credits for a device that does not have a telephone function (to dial a USSD string) which is normally used to top-up credits using a pin-based voucher.

Live airtime is available for

a. Vodacom (network=”VODACOM”)
b. MTN (network=”MTN”)
c. CellC (network=”CELLC”)
d. Telkom Mobile or formerly 8ta (network=”TELKOM”)

The topupType request parameter will be used in future once all networks support the ability to top-up data instead of airtime. Until the networks offer this functionality the topupType should be set to “AIRTIME”

```
public partial class LiveAirtimeReq : BaseReq
{
    /// the MSISDN of the cell number to be topped up
    /// this should be 10 digits (no international identifier) 0721723358
    public string msisdn;
    /// this is the network of the msisdn
    /// VODACOM,MTN,CELLC,TELKOM for example are appropriate values
    public string network;
    /// This is for future use
    /// this should be AIRTIME for now
    /// in future and if available this could be DATA,SMS etc.
    public string topupType;
    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    public decimal purchaseValue;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class LiveAirtimeResp : BaseResp
{
    /// the MSISDN of the cell number that was topped up
    /// this should be 10 digits (no international identifier) 0721723358
    public string msisdn;
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// the vouchercode for this Vend from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// Supplier generated order Number for the vend
    public string orderRef;
    /// Subscriber balance before the transaction
    public Decimal subscriberBalanceBefore;
    public bool subscriberBalanceBeforeSpecified;    //is subscriberBalanceBefore specified
    /// Subscriber balance after the transaction
    public Decimal subscriberBalanceAfter;
    public bool subscriberBalanceAfterSpecified;    //is subscriberBalanceAfter specified
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost
    public Decimal TransactionCost;
}
```

10. PinlessVouchers

Like LiveAirtime, pinless vouchers are credited directly to the account balance without the need for the account holder to redeem a pin. PInlessVouchers are however static in value thus unlike LiveAirtime the purchaseValue is pre-determined by the product to be loaded.

PinlessVouchers allow the purchase and provision of data / sms / social / whatsapp etc bundle types directly to the customers account at the Network.

10.1 PinlessVoucherProductList

Minimum Timeout	70 seconds
TerminalID Locked	No

To Obtain a list of available products for either a specified network or all networks, the PinlessVoucherProductList method should be used.

Acceptable value for network are either an empty string “” meaning all networks will be returned or “VODACOM”,”MTN”,”CELLC” or “TELKOM” to only return products for a specific network.

```
public partial class PinlessVoucherProductListReq : BaseReq
{
    /// <summary>
    /// this is the network the product belongs too
    /// VODACOM,MTN,CELLC,TELKOM for example are appropriate values
    /// if left blank / null then all products for all available networks would be returned
    /// </summary>
    public string network;
}
```

```
public partial class PinlessVoucherProductListResp : BaseResp
{
    /// <summary>
    /// List of Products available
    /// </summary>
    public PinlessVoucherProduct[] products;

    /// <summary>
    /// VoucherCode detail for each vouchercode Present in the product list returned
    /// Multiple ProductCodes will use the same voucherCode
    /// </summary>
    public VoucherCodeDetailExt[] voucherCodesDetail;
}
```

```
public partial class PinlessVoucherProduct
{
    /// <summary>
    /// this is the network the product belongs too
    /// VODACOM,MTN,CELLC,TELKOM for example are appropriate values
    /// </summary>
    public string network;

    /// <summary>
    /// this is the productCode of the product that is to be topped up to the msisdn
    /// </summary>
    public string productCode;

    /// <summary>
    /// Grouping Similar products together
    /// </summary>
    public string productGroup;

    /// <summary>
    /// the voucherCode (and discount) associated to the productCode
    /// </summary>
    public string voucherCode;

    /// <summary>
    /// this is the productDescription of the product that is to be topped up to the msisdn
    /// Long and Descripting Message describing the product in detail
    /// </summary>
    public string productDescription;

    /// <summary>
    /// the Value in Rands of the product
    /// </summary>
    public decimal voucherValue;

}
```

```
public class VoucherCodeDetailExt : VoucherCodeDetail
{
    /// <summary>
    /// if the voucher is part of a group of vouchercodes that all vend to the same supplier
    /// for Example AP_ESKOM and IT_ESKOM would both have voucherGroup = "Eskom"
    /// </summary>
    public String voucherGroup;
    /// <summary>
    /// if VAT is applicable for the product
    /// </summary>
    public Boolean vatItem;
    /// <summary>
    /// if the product is commission Based
    /// </summary>
    public Boolean commissionProduct;
    /// <summary>
    /// if the product is commission Based, the PastelCode applicable
    /// </summary>
    public String commissionCode;
}
```

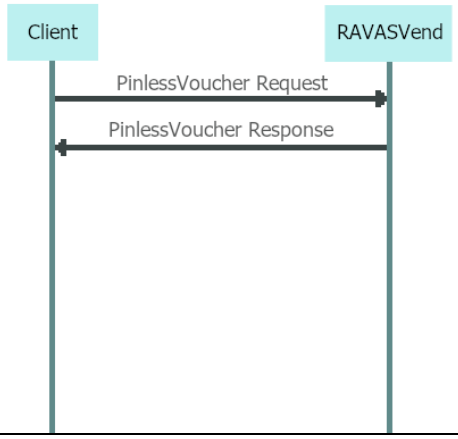
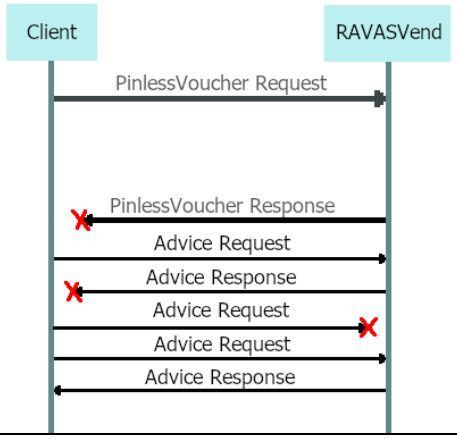
10.2 PinlessVoucher

Minimum Timeout	60 seconds
TerminalID Locked	Yes (transaction must be completed)

```
public partial class PinlessVoucherReq : BaseReq
{
    /// <summary>
    /// the MSISDN of the cell number to be topped up
    /// this should be 10 digits (no international identifier) 0721723358
    /// </summary>
    public string msisdn;

    /// <summary>
    /// this is the network of the msisdn
    /// VODACOM,MTN,CELLC,TELKOM for example are appropriate values
    /// </summary>
    public string network;

    /// <summary>
    /// this is the productCode of the product that is to be topped up to the msisdn
    /// </summary>
    public string productCode;

    /// <summary>
    /// the voucherCode (and discount) associated to the productCode
    /// </summary>
    public string voucherCode;

    /// <summary>
    /// this is the face value of the product (productCode) that should be topped up
    /// </summary>
    public decimal purchaseValue;

    /// <summary>
    /// this will specify the tender being used to make the purchase.
    /// </summary>
    public Tender tender;

    /// <summary>
    /// The Receipt Format that is required to be returned
    /// </summary>
    public ReceiptFormat receiptFormat;
    /// <summary>
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    /// </summary>
    public string terminalChannel;
    /// <summary>
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    /// </summary>
    public string terminalCompanyName;
    /// <summary>
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    /// </summary>
    public string terminalOperator;
}
```

```
public partial class PinlessVoucherResp : BaseResp
{
    /// <summary>
    /// the MSISDN of the cell number to be topped up
    /// this should be 10 digits (no international identifier) 0721723358
    /// </summary>
    public string msisdn;
```

```
    /// <summary>
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    /// </summary>
    public string customerMsg;

    /// <summary>
    /// the vouchercode for this Vend from this supplier
    /// </summary>
    public string voucherCode;

    /// <summary>
    /// this is the  product that was topped up to the msisdn
    /// </summary>
    public PinlessVoucherProduct product;

    /// <summary>
    /// Will return detail about the vouchercode if it is available
    /// </summary>
    public VoucherCodeDetailExt voucherCodeDetail;

    /// <summary>
    /// Supplier generated order Number for the vend
    /// </summary>
    public string orderRef;

    /// <summary>
    /// The Receipt format Received in the CreditVendReq
    /// </summary>
    public ReceiptFormat receiptFormat;
    /// <summary>
    /// The Receipt string formated to the requested ReceiptFormat
    /// </summary>
    public string receipt;
    /// <summary>
    /// the SMS that can be sent to the customer MSISDN containing all the information
    /// </summary>
    public string smsreceipt;
    /// <summary>
    /// The Aggregators Transaction Cost (
    /// for example a request for purchaseValue=R50.00 May be calculated at R49.00
    /// Hence R49.00 will be subtracted from the Aggregator Account at RACellular
    /// </summary>
    public Decimal TransactionCost;
}
```

	
PInlessVoucher HappyPath where PinlessVoucher Response has hasFault=False or hasFault=true with mustALR=false	PinlessVoucher Requiring advise due to no response received from RAVASVend. Advice only ends when AdviceResponse returns hasFault=False or hasFault=true and mustALR=false

11. AnyValueAirtime

Minimum Timeout	60 seconds
TerminalID Locked	Yes (transaction must be completed)

The AnyValueAirtime method requests the Cellular Utility to generate a pin number to the value of the requested amount. If for example the customer would like to purchase a Vodacom R4.35 Prepaid voucher (because the commonly available R5 voucher is too expensive) then the AnyValueAirtime Method will return the voucher for the requested amount.

At the time of writing this method is only available for the MTN network.

```
public partial class AnyValueAirtimeReq : BaseReq
{
    /// VODACOM,MTN,CELLC,TELKOM for example are appropriate values
    public string network;
    /// this should be AIRTIME for now
    /// in future and if available this could be DATA,SMS etc.
    public string topupType;
    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    public decimal purchaseValue;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class AnyValueAirtimeResp : BaseResp
{
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// the vouchercode for this Vend from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// Supplier generated order Number for the vend
    public string orderRef;
    /// Supplier generated serial Number for the vend
    public string serial;
    /// Supplier generated PIN Number for the vend
    public string pinNumber;
    /// FaceValue of the voucher
    public Decimal purchaseValue;
    /// The USSD code to recharge the PINNUMBER
    public string rechargeInstruction;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost (
    public Decimal TransactionCost;
}
```

12. OTTVoucher

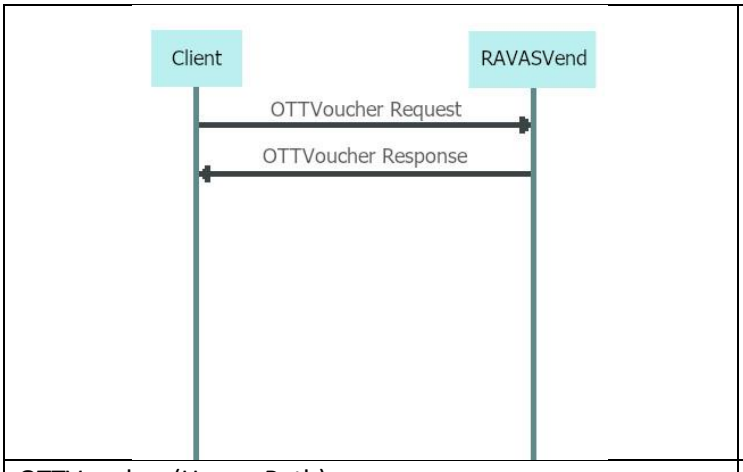
Minimum Timeout	60 seconds
TerminalID Locked	Yes (transaction must be completed)

OTT Voucher is an online payment voucher. It can be used to top-up a customer's account at many online platforms like Hollywood bets, betWay, bet.co.za to name but a few. For more information refer to www.ottvoucher.com

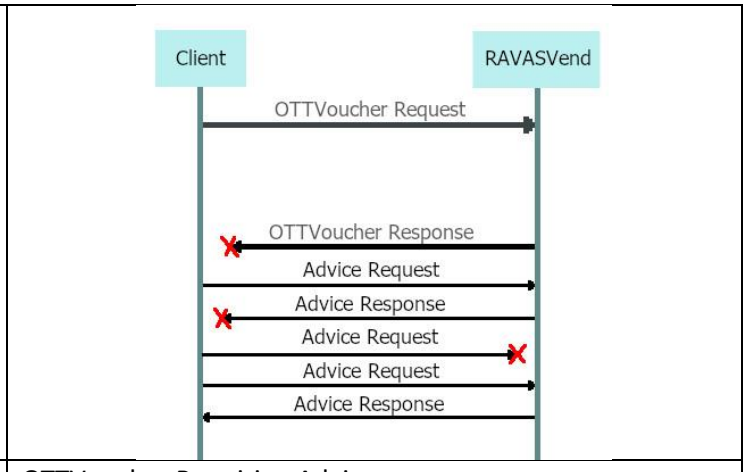
OTT Voucher works by purchasing an OTTVoucher Pin (using the OTTVoucher Method), the pinNumber can then be used at the *checkout* or *account* Menu of the online merchant where the customer would like to redeem the voucher.

```
public partial class OTTVoucherReq : BaseReq
{
    /// VoucherCode is required, Because OTT purchases are not 2 stop the voucher needs to be known before the time OTT_VOUCHER
    public string voucherCode;
    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    public decimal purchaseValue;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class OTTVoucherResp : BaseResp
{
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// the vouchercode for this Vend from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// Supplier generated order Number for the vend
    public string orderRef;
    /// Supplier generated serial Number for the vend
    public string serial;
    /// Supplier generated PIN Number for the vend
    public string pinNumber;
    /// Supplier generated Batch Number for the vend
    public string batchNumber;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost
    public Decimal TransactionCost;
}
```



OTTVoucher (Happy Path)



OTTVoucher Requiring Advice

13. Talk360Voucher

Minimum Timeout	60 seconds
TerminalID Locked	Yes (transaction must be completed)

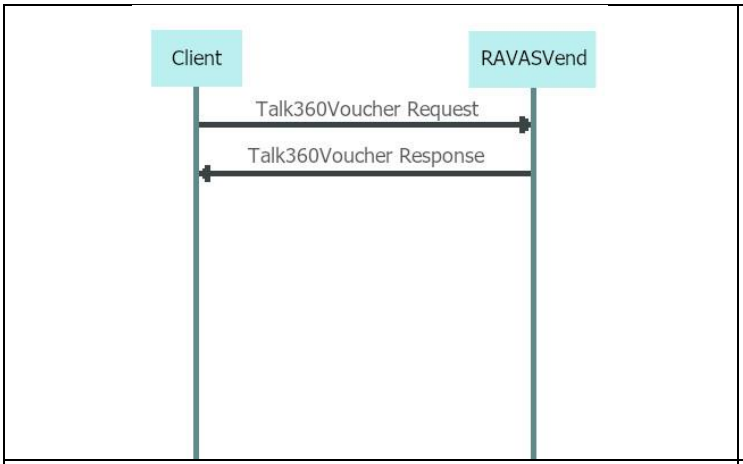
Talk360 is an internation calling application that can be installed on any SmartPhone.

The app allows its users to make international calls at reduced rates from what their own cellular number.

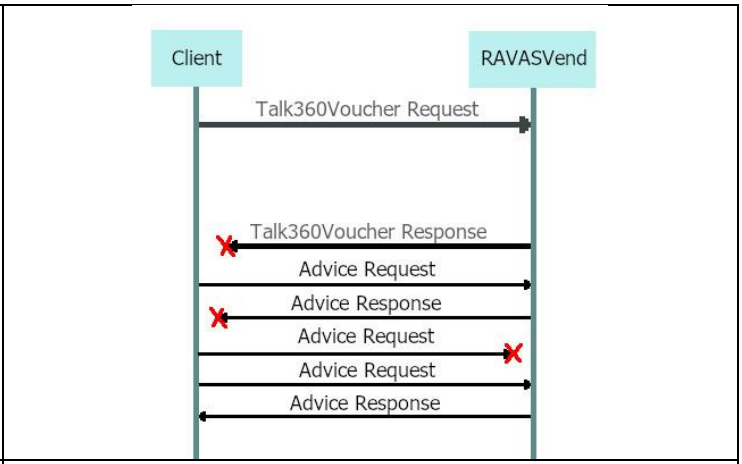
The Talk360Voucher method is used to purchase a Talk360 Voucher PinNumber which is redeemed on the Talk360 App to top-up the customers wallet balance.

```
public partial class Talk360VoucherReq : BaseReq
{
    /// VoucherCode is required, Because Talk360 purchases are not 2 stop the voucher needs to be known before the time TALK360_VOUCHER
    public string voucherCode;
    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    public decimal purchaseValue;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class Talk360VoucherResp : BaseResp
{
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// the vouchercode for this Vend from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// Supplier generated order Number for the vend
    public string orderRef;
    /// Supplier generated serial Number for the vend
    public string serial;
    /// Supplier generated PIN Number for the vend
    public string pinNumber;
    /// Supplier generated Batch Number for the vend
    public string batchNumber;
    /// The Receipt format Received in the Talk360VoucherReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost
    public Decimal TransactionCost;
}
```



Talk360Voucher (Happy Path)



Talk360Voucher Requiring Advice

14. EasyPay Bill Payments

The EasyPay Bill Payment service is an electronic solution for payment for third party accounts, such as municipal and telephone bills, traffic fines and insurance policies.

EasyPay BillPayments is a 3 Step process and each step/process is described in the following section.

14.1 EasyPayConfirm

Minimum Timeout	35 seconds
TerminalID Locked	Yes

The EasyPayConfirm method is used to Query the status of an account, traffic fine or other payment. The response contains all the pertinent information necessary to proceed to the payment portion of the EasyPay Bill Payment Process.

The confirmation can be done using either an EasyPay Number (epNo) or a traffic infringement notice number (noticeNo).

```
public partial class EasyPayConfirmReq : BaseReq
{
    //easyPay Number or Notice Number
    public String accountNumber;
    //type of accountNumber
    public EasyPayPaymentIdentifierType accountType;
}
```

```
public partial class EasyPayConfirmResp : BaseResp
{
    /// the vouchercode for this product
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// The correct amount specified by the Utility
    public decimal correctAmount;
    public Boolean correctAmountSpecified;
    /// The maximum amount that the customer can pay
    public decimal maxAmount;
    public Boolean maxAmountSpecified;
    /// The minimum amount that the customer can pay
    public decimal minAmount;
    public Boolean minAmountSpecified;
    /// Boolean value indicating whether the amount is payable (might have allready been paid)
    /// or it might be expired (traffic fines for example)
    public Boolean payable;
    /// name of the receiver
    public string receiver;
    /// A Message to the Customer from the Utility (Additional Information about the Bill)
    public string customerMsg;

}
```

```
public enum EasyPayPaymentIdentifierType
{
    /// EasyPay numbers start with a 9 and are usually prefixed with a number of >>>> characters
    epNo,
    /// Payment of a traffic fine (Enter all characters even -'s and /'s
    noticeNo,
}
```

14.2 EasyPayBillPayment

Minimum Timeout	35 seconds
TerminalID Locked	Yes (transaction must be completed)

Once the EasyPayConfirm Response is received and validated the Operator or the terminal can continue to pay the bill. The only additional parameter required by EasyPayBillPayment is the purchaseValue (The amount that needs to be paid)

```
public partial class EasyPayBillPaymentReq : BaseReq
{
    /// returned from EasyPayConfirmResp
    public string voucherCode;

    //easyPay Number or Notice Number
    public String accountNumber;
    //type of accountNumber
```

```
    public EasyPayPaymentIdentifierType accountType;

    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    public decimal purchaseValue;
}
```

```
public partial class EasyPayBillPaymentResp : BaseResp
{
    /// the vouchercode for this product
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// the returned easypay number for the request (this is used for confirmation)
    public string easypayNumber;
    /// the returned transaction id for the request (this is use for confirmation)
    public string tranID;
}
```

14.3 EasyPayBillPaymentConfirm

Minimum Timeout	35 seconds
TerminalID Locked	Yes (transaction must be completed)

The third and final step of the Easypay Bill Payment is to either confirm or cancel the payment.

EasyPay developed the Bill Payment System as a 3 step process where the final step acknowledges or cancels the payment and this gives the cashier/operator of the terminal the opportunity to process payments from the customer and in the event that insufficient payment is available then the transaction can be cancelled.

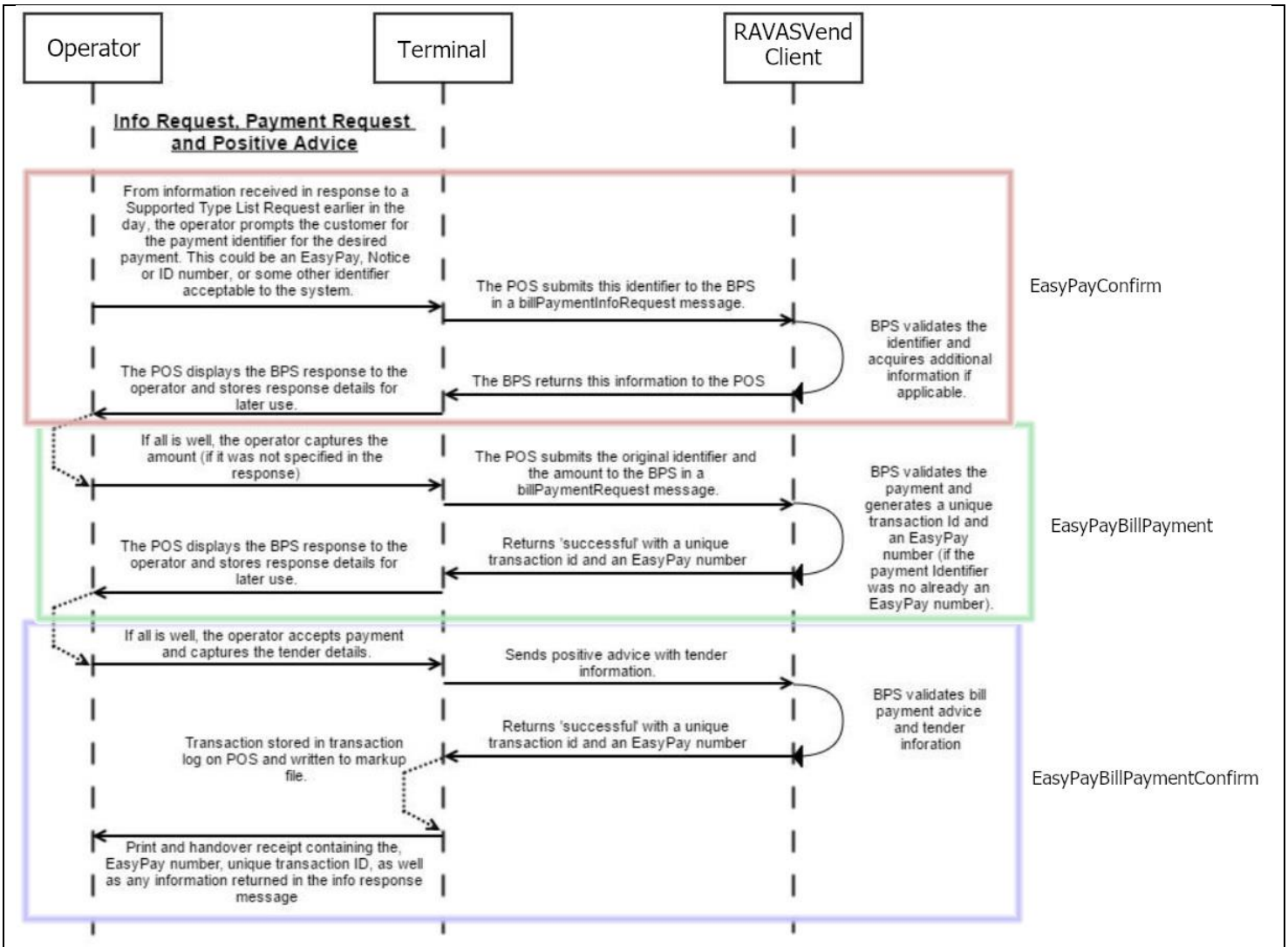
The EasyPayBillPaymentConfirm must follow the EasyPayBillPayment (either positive or negative acknowledgement) and cannot simply be left if the Operator of the terminals decides not to proceed with acknowledgement after payment.

If the EasyPayBillPaymentConfirm is negative (i.e. cancel=true) then tender and purchaseValue request parameters do not need to be populated and only the cancelReason needs to be selected from the enumerated list of available options.

```
public partial class EasyPayBillPaymentConfirmReq : BaseReq
{
    /// the vouchercode for this product
    public string voucherCode;

    public Boolean cancel;          //cancellation advice if true
    public EasyPayCancelReason cancelReason;     //reason for cancellation

    /// returned from EasyPayBillPaymentResp
    public string tranID;

    /// This must match the amount of EasyPayBillPaymentReq
    public decimal purchaseValue;

    /// this will specify the tender being used to make the purchase.
    public Tender tender;

    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;

}
```

```
public partial class EasyPayBillPaymentConfirmResp : BaseResp
{
    /// the vouchercode for this Vend from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// <summary>
    /// The Aggregators Transaction Cost
    public Decimal TransactionCost;
}
```

```
public enum EasyPayCancelReason
{
    /// the POS timed out waiting for request message.
    timeout,
    /// The user aborted transaction before completion.
    user_request,
    /// a pos error prevented the transaction from being completed. Printer error, Database error etc.
    system_error,
}
```



EasyPay Example of interaction of operator with terminal for Positive confirmation

15. Moolapin

The Moolapin voucher is a voucher developed by RA Cellular initially as a way for buyprepaid.co.za customer to top-up their buyprepaid.co.za account wallet. These vouchers are available at all RA Cellular outlets and vendors.

The voucher was however extended to allow any account within the RA Cellular network to either purchase or consume the voucher.

The purchase of a voucher reduces the balance of the purchaser at RA Cellular where the consumer does the opposite and increases the balance of the consumer at RA Cellular.

15.1 MoolaPinVoucher

Minimum Timeout	35 seconds
TerminalID Locked	Yes (transaction must be completed)

The MoolaPinVoucher Method purchases a moolapin voucher of the requested denomination. If RAVASVend is not required to populate the receipt, then the paymentFee value can be defaulted to 0. However, if a fee is being levied by the operator and the RAVASVend receipt is printed for the customer then the paymentFee value must be populated.

The vouchercode='MOOLAPIN' should be used when purchasing the MoolaPin Voucher.

```
public partial class MoolaPinVoucherReq : BaseReq
{
    /// VoucherCode is required, Because MoolaPin purchases are not 2 step the voucher needs to be known before the time
    /// MOOLAPIN variable value voucher
    public string voucherCode;

    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    /// this does not include the paymentFee,
    /// customer will be billed purchaseValue, but customer will sell at purchaseValue + paymentFee
    /// No Cents are allowed (only full Rand Values)
    public decimal purchaseValue;

    /// the fee being charged by the terminal for the payment
    /// So if purchaseValue = R90 and paymentFee = R5 then the receipt returned will have a total of R95
    /// but the MoolaPin Voucher will only have a value of R90 and the account at RAVASVEnd will only be debited for the R90
    /// the R5 service fee is simply part of the request to correctly create the receipt to be printed
    /// No Cents are allowed (only full Rand Values)
    public decimal paymentFee;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class MoolaPinVoucherResp : BaseResp
{
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// the vouchercode for this Vend from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// Supplier generated serial Number for the vend
    public string serial;
    /// Supplier generated PIN Number for the vend
    public string pinNumber;
    /// The Receipt format Received in the MoolaPinVoucherReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost
    public Decimal TransactionCost;
}
```

15.2 MoolaPinRedeem

Minimum Timeout	35 seconds
TerminalID Locked	Yes (transaction must be completed)

MoolaPinRedeem redeems the value of the moolapin voucher and credits the Clients account on RAVASVend.

```
public partial class MoolaPinRedeemReq : BaseReq
{
    /// PIN Number of the voucher
    public string pinNumber;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class MoolaPinRedeemResp : BaseResp
{
    /// The original purchaseValue when the voucher was created,
    /// this is the amount that will be credited to the account
    public decimal purchaseValue;
    /// the original paymentFee selected when the voucher was created
    /// this is just informational.
    public decimal paymentFee;
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// the vouchercode for this Vend from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// Supplier generated serial Number for the vend
    public string serial;
    /// Supplier generated PIN Number for the vend
    public string pinNumber;
    /// The Receipt format Received in the MoolaPinVoucherReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// This will be negative as it is a credit transaction
    public Decimal TransactionCost;
}
```

16. SFX

SFX is a money transfer company that together with Ria is able to send money to over 350 000 destinations all over the world. Customers can register with SFX (either online or with the mobile app) and using their account they can create beneficiaries and obtain quotes to remit money.

Once the customer has accepted a quote, they are provided by SFX a payment reference that can then be used to pay the amount required to remit the funds.

16.1 SFXLookup

Minimum Timeout	35 seconds
TerminalID Locked	Yes

SFXLookup queries the Payment Reference given by the customer to the terminal operator and will inform the operator of the amount that needs to be received from the customer to submit the SFXPayment.

```
public partial class SFXLookupReq : BaseReq
{
    /// The reference of the payment for lookup.
    public string payRef;
}
```

```
public partial class SFXLookupResp : BaseResp
{
    /// the vouchercode for this product (always DSTV_PAYMENT)
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// The reference of the payment for lookup.
    public string payRef;
    /// The amount required to settle the payment.
    public decimal paymentAmount;
    /// Customer First Name
    public string customerName;
    /// Customer Surname
    public string customerSurname;
}
```

16.2 SFXPayment

Minimum Timeout	35 seconds
TerminalID Locked	Yes (transaction must be completed)

Once the SFXLookup has been confirmed and money is received by the operator of the terminal then the operator may submit the SFXPayment that will remit the money to be collected by the beneficiary of the customer.

```
public partial class SFXPaymentReq : BaseReq
{
    /// The reference of the payment
    public string payRef;

    /// VoucherCode is required, It is required to call SFXLookupReq First to obtain the voucherCode for the Utility Required
    public string voucherCode;
    /// The amount required to settle the payment.
    /// Returned from SFXLookupReq
    public decimal paymentAmount;
    /// the fee being charged by the terminal for the payment
    /// So if paymentAmount = R90 and paymentFee = R5 then the receipt returned will have a total of R95
    /// but SFX will only be paid R90 and the account at RAVASVEnd will only be debited for the R90
    /// the R5 service fee is simply part of the request to correctly create the receipt to be printed
    public decimal paymentFee;
    /// Customer First Name
    /// returned in SFXLookupReq
    public string customerName;
    /// Customer Surname
    /// returned in SFXLookupReq
    public string customerSurname;
    /// this will specify the tender being used to make the purchase.
    public Tender tender;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc    public string terminalOperator;
}
```

```
public partial class SFXPaymentResp : BaseResp
{
    /// The reference of the payment
    public string payRef;
    /// Receipt number for this payment
    public string receiptNumber;
    /// the vouchercode for this customer from this supplier
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public VoucherCodeDetail voucherCodeDetail;
    /// The Receipt format Received in the CreditVendReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
    /// The Aggregators Transaction Cost
    public Decimal TransactionCost;
}
```

17. SBInstantMoney

Minimum Timeout	60 seconds
TerminalID Locked	Yes (transaction must be completed)

Standard Bank Instant Money sends money quickly and safely to anyone in South Africa with a cellphone number, with a PIN-controlled cardless collection a SB Instant Money recipient can collect the money due to them from a terminal exposing this service from RAVASVend.

The SBInstantMoney transaction will credit the RAVASVend client account (i.e. money is being payed out by the operator of the terminal and as such the terminal balance will also increase)

```
public partial class SBInstantMoneyReq : BaseReq
{
    /// VoucherCode is required, Because SBInstantMoney purchases are not 2 stop the voucher needs to be known before the time SBINSTANTMONEY
    public string voucherCode;
    /// The Pin Number of the StandardBank Instant Money Voucher
    public string voucherPin;
    /// The Secret Pin Number of the Standard Bank Instant Money Voucher
    public string userPin;
    /// the Amount of the voucher
    /// this is only informative and the response amount can be different to this.
    public decimal amount;
    /// The Contact Cellphone Number of the merchant that is processing the Voucher
    public string terminalMSISDN;
    /// The Receipt Format that is required to be returned
    public ReceiptFormat receiptFormat;
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    public string terminalChannel;
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    public string terminalCompanyName;
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    public string terminalOperator;
}
```

```
public partial class SBInstantMoneyResp : BaseResp
{
    /// The Pin Number of the StandardBank Instant Money Voucher
    public string voucherPin;
    /// The Secret Pin Number of the Standard Bank Instant Money Voucher
    public string userPin;
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    public string customerMsg;
    /// voucherCode / paymentCode detail for this voucher
    public string voucherCode;
    /// Will return detail about the vouchercode if it is available
    public PaymentCodeDetail voucherCodeDetail;
    /// The amount the voucher is worth (how much cash to give the customer)
    public Decimal amount;
    /// How much the fee was for the transaction.
    /// Aggregator gets amount - paymentFee added to their account balance
    public Decimal paymentFee;
    /// Transaction Reference Number
    public string reference;
    /// The Receipt format Received in the SBInstantMoneyReq
    public ReceiptFormat receiptFormat;
    /// The Receipt string formated to the requested ReceiptFormat
    public string receipt;
    /// The Receipt string formated to the requested ReceiptFormat for the merchant.
    public string merchantReceipt;
    /// the SMS that can be sent to the customer MSISDN containing all the information
    public string smsreceipt;
}
```

```
public class PaymentCodeDetail
{
    //Unique voucherCode for the Payment Type
    public String voucherCode;
    //Pastel Code setup for the voucherCode
    public String pastelCode;
    //Pastel Code for the Fee portino of the vouchercode
    public String feeCode;

    //Percentage of value that is deducted for fee
    public Decimal percentage;
    //Fixed amount per transaction that is deducted for fee
    public Decimal fixedAmt;

    //Minimum Fee applicable to voucherCode
    public Decimal minFee;
    //Maximum fee applicable to voucherCode
    public Decimal maxFee;
    //Enabled/Disabled Indicator
    public Boolean enabled;
}
```

18. AnyTime Vouchers

AnyTime vouchers are a monetary voucher that can be redeemed for goods and services.

An Anytime voucher can be used to redeem airtime / data by dialing *130*4776*ANYTIMEPIN# from a mobile phone. It can also be used to topup online wallets such as Hollywood bets / Supa bets etc by following the recharge instructions on the relevant website.

The AnytimeVoucherRedeem method allows a RAVASVend Client to redeem the value of the voucher to their RAVASVend Balance and this functionality can thus be used to by Customers to redeem value at the client by using the anytime voucher.

A simplified version of the API is available that accepts a JSON body with Basic Authentication when only the AnyTimeVoucherRedeem and Advice methods are required and all other functionality of RAVASVend will not be utilized. The API “Simple AnyTime Redeem.pdf” specification is available on request.

18.1 AnyTimeVoucher

Minimum Timeout	35 seconds
TerminalID Locked	Yes (transaction must be completed)

The AnyTimeVoucher method is used to create / purchase a new anytime voucher to the value of the request.

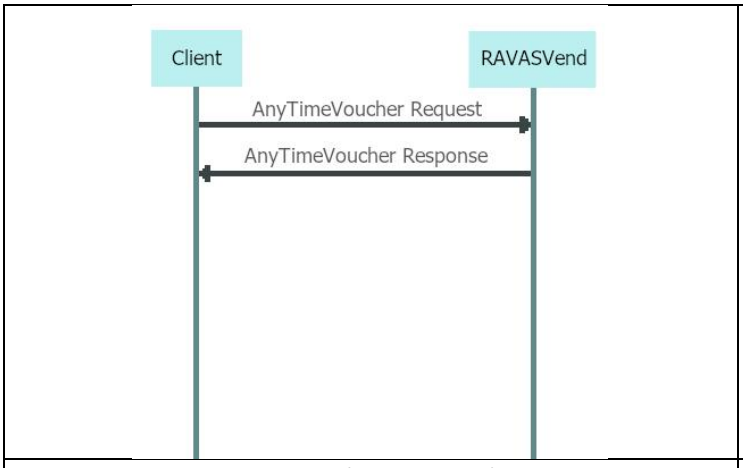
The value may be anything from 1 cent and increment by 1c to a maximum value of R999.00

The vouchercode in request must be ANYTIME_VOUCHER

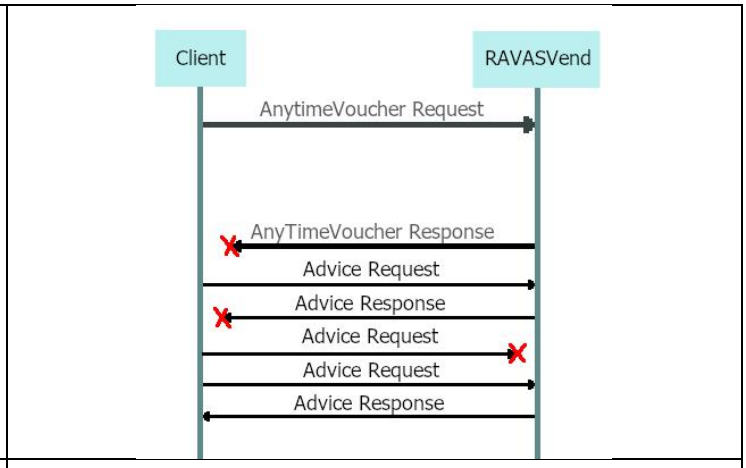
```
/// <summary>
/// Purchase an AnyTime voucher of the supplied amount
/// </summary>
public partial class AnyTimeVoucherReq : BaseReq
{
    /// <summary>
    /// VoucherCode is required, Because AnyTime purchases are not 2 stop the voucher needs to be known before the time ANYTIME_VOUCHER
    /// </summary>
    public string voucherCode;

    /// <summary>
    /// The Amount of Currency that is required in this purchase in Rands (0.00)
    /// </summary>
    public decimal purchaseValue;
    /// <summary>
    /// this will specify the tender being used to make the purchase.
    /// </summary>
    public Tender tender;

    /// <summary>
    /// The Receipt Format that is required to be returned
    /// </summary>
    public ReceiptFormat receiptFormat;
    /// <summary>
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    /// </summary>
    public string terminalChannel;
    /// <summary>
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    /// </summary>
    public string terminalCompanyName;
    /// <summary>
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    /// </summary>
    public string terminalOperator;
}
```

```
public partial class AnyTimeVoucherResp : BaseResp
{
    /// <summary>
    /// A Message to the Customer from the Utility (Normally printed on the bottom of the receipt)
    /// </summary>
    public string customerMsg;

    /// <summary>
    /// the vouchercode for this Vend from this supplier
    /// </summary>
    public string voucherCode;
```

```
    /// <summary>
    /// Will return detail about the vouchercode if it is available
    /// </summary>
    public VoucherCodeDetail voucherCodeDetail;

    /// <summary>
    /// Supplier generated order Number for the vend
    /// </summary>
    public string orderRef;

    /// <summary>
    /// Supplier generated serial Number for the vend
    /// </summary>
    public string serial;
    /// <summary>
    /// Supplier generated PIN Number for the vend
    /// </summary>
    public string pinNumber;
    /// <summary>
    /// Supplier generated Batch Number for the vend
    /// </summary>
    public string batchNumber;

    /// <summary>
    /// The Receipt format Received in the CreditVendReq
    /// </summary>
    public ReceiptFormat receiptFormat;
    /// <summary>
    /// The Receipt string formated to the requested ReceiptFormat
    /// </summary>
    public string receipt;
    /// <summary>
    /// the SMS that can be sent to the customer MSISDN containing all the information
    /// </summary>
    public string smsreceipt;
    /// <summary>
    /// The Aggregators Transaction Cost (
    /// for example a request for purchaseValue=R50.00 May be calculated at R49.00
    /// Hence R49.00 will be subtracted from the Aggregator Account at RACellular
    /// </summary>
    public Decimal TransactionCost;
}
```



AnyTime Voucher example (Happy Path)



AnyTime Voucher example requiring Advice

18.2 AnyTimeVoucherRedeem

Minimum Timeout	35 seconds
TerminalID Locked	Yes (transaction must be completed)

The AnyTimeVoucherRedeem method is used to redeem the value of an un-used AnyTime voucher and the Clients account will be credited with the Value of the voucher (less the cost of using the AnyTime voucher as a payment service)

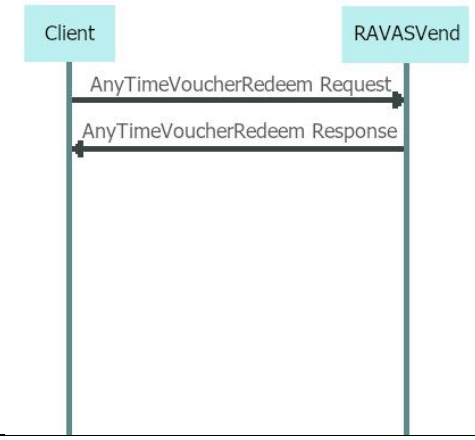
The vouchercode = ANYTIME_VOUCHER_REDEEM must be supplied in the request.

An options reason for the redemption should be supplied if available, this can be any string up to 255 characters and should briefly descript what the voucher was used for. Examples of such a reason could be “Hollywood bets redemption via website”

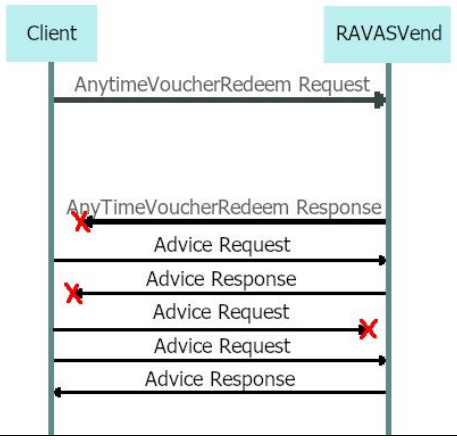
```
/// <summary>
/// Purchase an AnyTime voucher of the supplied amount
/// </summary>
public partial class AnyTimeVoucherRedeemReq : BaseReq
{
    /// <summary>
    /// voucherCode / paymentCode detail for this voucher
    /// </summary>
    public string voucherCode;

    /// <summary>
    /// Anytime Voucher Pin Number
    /// </summary>
    public string pinNumber;

    /// <summary>
    /// Used By Reason for example HOLLYWOODBETS TOPUP for selwind@gmail.com
    /// Max 200 chars
    /// </summary>
    public string reason;

    /// <summary>
    /// The Receipt Format that is required to be returned
    /// </summary>
    public ReceiptFormat receiptFormat;
    /// <summary>
    /// The Channel used for this transaction e.g. "POS","USSD","WEB","ATM" etc)
    /// </summary>
    public string terminalChannel;
    /// <summary>
    /// The Company Name that is operating the terminal e.g. "Spar Lynwood"
    /// </summary>
    public string terminalCompanyName;
    /// <summary>
    /// The Name of the Operator of the terminal that has requested the Purchase e.g. "USER1","Miguel" etc
    /// </summary>
    public string terminalOperator;
}
```

```
public partial class AnyTimeVoucherRedeemResp : BaseResp
{
    /// <summary>
    /// Anytime Voucher Pin Number from Request
    /// </summary>
    public string pinNumber;

    /// <summary>
    /// Unique RequestID
    /// </summary>
    public string requestID;

    /// <summary>
    /// voucherCode / paymentCode detail for this voucher
    /// </summary>
    public string voucherCode;

    /// <summary>
    /// The amount the voucher is worth (how much cash to give the customer)
    /// </summary>
    public Decimal amount;

    /// <summary>
    /// The Transaction ID
    /// </summary>
    public string tID;

    /// <summary>
    /// serial of the Anytime Voucher
    /// </summary>
    public string serial;

    /// <summary>
    /// Reference of the Transaction
    /// </summary>
    public string reference;

    /// <summary>
    /// The Receipt string formated to the requested ReceiptFormat for the merchant.
    /// </summary>
    public string merchantReceipt;

    /// <summary>
    /// the SMS that can be sent to the customer MSISDN containing all the information
    /// </summary>
    public string customerReceipt;

    /// <summary>
    /// the SMS that can be sent to the customer MSISDN containing all the information
    /// </summary>
    public string smsreceipt;

    /// <summary>
    /// Will return detail about the vouchercode if it is available
    /// </summary>
    public PaymentCodeDetail paymentCodeDetail;
    /// <summary>
    /// How much the fee was for the transaction.
    /// Aggregator gets amount - paymentFee added to their account balance
    /// </summary>
```

```
    public Decimal paymentFee;

    /// <summary>
    /// The Receipt format Received in the SBInstantMoneyReq
    /// </summary>
    public ReceiptFormat receiptFormat;
}
```



AnyTimeVoucherRedeem example (Happy Path)



AnyTimeVoucherRedeem example requiring Advice

18.3 AnyTimeVoucherLookup

Minimum Timeout	35 seconds
TerminalID Locked	No

The AnyTimeVoucherLooup method is used to query the status of an anytime voucher using either the serial number of the pinnumber of the voucher

```
public partial class AnyTimeVoucherLookupReq : BaseReq
{
    /// <summary>
    /// Anytime Voucher Pin Number
    /// </summary>
    public string pinNumber;

    /// <summary>
    /// Anytime Voucher Serial Number
    /// </summary>
    public string serial;
}
```

```
public partial class AnyTimeVoucherLookupResp : BaseResp
{
    /// <summary>
    /// Anytime Voucher Pin Number from Request
    /// </summary>
    public string pinNumber;

    /// <summary>
    /// The amount the voucher is worth (how much cash to give the customer)
    /// </summary>
    public Decimal amount;

    /// <summary>
    /// serial of the Anytime Voucher
    /// </summary>
    public string serial;

    /// <summary>
    /// if the voucher was reserved and when
    /// </summary>
    public DateTime dateReserved;
    public Boolean reserved;

    /// <summary>
    /// if the voucher was used and when
    /// </summary>
    public DateTime dateUsed;
    public Boolean used;

    /// <summary>
    /// string depicting for whome the voucher was used
    /// </summary>
    public string usedByReason;
}
```

19. General Methods

19.1 Balance

The Balance Method is used to obtain a RAVASVend Client account balance

Suggested Timeout	10 seconds
TerminalID Locked	No

```
public partial class AccBalanceReq : BaseReq
{

}
```

```
public partial class AccBalanceResp : BaseResp
{
    /// The Balance available to trade with
    public decimal balance;
    /// Temporary credit allocated to the aggregator.
    /// The topay credit amount is reduced to 0 with each subsequent deposit made by the aggregator
    public decimal topay;
    /// The permanent or long term credit allocated to teh aggregator
    /// deposits made by the aggregator do not affect the ToPay Amount
    public decimal rolling;
}
```

19.2 LiveVoucherList

The LiveVoucherList Method is used to obtain a list of all costing information for Prepaid Electricity and Direct TopUps (not Prepaid Vouchers)

Suggested Timeout	10 seconds
TerminalID Locked	No

```
public partial class LiveVoucherListReq : BaseReq
{
}
```

```
public partial class LiveVoucherListResp : BaseResp
{
    /// <summary>
    /// VoucherInformation for all available vouchers on RAVASVend
    /// </summary>
    public VoucherCodeDetailExt[] voucherCodeDetailExt;
}
```

```
public class VoucherCodeDetailExt : VoucherCodeDetail
{
    /// <summary>
    /// if the voucher is part of a group of vouchercodes that all vend to the same supplier
    /// for Example AP_ESKOM and IT_ESKOM would both have voucherGroup = "Eskom"
    /// </summary>
    public String voucherGroup;
    /// <summary>
    /// if VAT is applicable for the product
    /// </summary>
    public Boolean vatItem;
    /// <summary>
    /// if the product is commission Based
    /// </summary>
    public Boolean commissionProduct;
    /// <summary>
    /// if the product is commission Based, the PastelCode applicable
    /// </summary>
    public String commissionCode;
}
```

20. Fault

The fault object of the BaseResp is populated when hasFault=True.

RAVASVend does not throw SOAP faults but instead the hasFault object should be queried to determine if the request was successful or not.

The Fault object returns descriptive and readable error messages that should be shown to the operator of the terminal (and in some cases the customer) to ensure that they understand why a fault was received.

```
public partial class Fault
{
    /// Determines if an ALR (AdviceRequest) is required for the Response Received
    public bool mustALR;

    /// The fault number
    public string faultnumber;
    /// The Description of the Fault
    public string desc;

    /// ENGLISH Title of the response (used as a slip header)
    public string EN_dispHeader;
    /// ENGLISH message to be given to the Terminal Operator
    public string EN_operatorMsg;
    /// ENGLISH message to be given to the Customer
    public string EN_custMsg;
}
```

The IncompleteTransactionFault is an extension of the Fault class. It will be received by the Client when a terminalID is locked but a new request is being sent instead of the required AdviceRequest.

```
public partial class IncompleteTransactionFault : Fault
{
    /// The TerminalMsgID of the Terminal to which an AdviceRequest must be completed
    public string reqterminalMsgID;
    /// The Aggregator MsgID to which an AdviceRequest must be completed
    public string reqMsgID;
    /// The DateTime that the Incomplete Request was received by MZEVend
    public System.DateTime reqDateTime;
    /// The Request Type of the Incomplete Request e.g. "CreditVendRequest"
    public string reqType;
}
```

If an IncompleteTransactionFault is returned it will contain information about the MsgID to which an Advice must be processed.

21. Advice

21.1 Introduction

Suggested Timeout	35 seconds
TerminalID Locked	No (but this is the only way to unlock a terminalID)

The AdviceReq as explained previously is to get the response of a request sent to RAVASVend.

All Revenue Transactions must be completed.

A Revenue Transaction is any request that has an influence of the clients balance and in this document it is marked as ("Terminal ID Locked : Yes") An example of a Revenue transaction is a CreditVend or LiveAirtime Methods.

The ConfirmCustomer Method is for example not a Revenue transaction.

An advice request should be performed when either

a. No response is received from a Revenue Request (for example due to Timeout)
b. A response is received to a Revenue requests with hasFault=true and mustALR=true
c. A response is received to any request (not just Revenue transactions) where hasFault=true and the fault is of the type "`IncompleteTransactionFault`" which specifies the MsgID of the locking transaction.
d. No response is received from an Advice Request
e. A response is received to an Advice Request with hasFault=true and mustALR=true

```
public partial class AdviceReq : BaseReq
{
    /// The MsgID of the request to which the response is required (Not TerminalMsg but rather the Aggregator MsgID
    public string adviceReqMsgID;
}
```

```
public partial class AdviceResp : BaseResp
{
    /// The MsgID of the request to which the response is was required
    public string adviceReqMsgID;
    /// The Response for the requested adviceReqMsgID
    public BaseResp lastResponse;
}
```

The AdviceReq does inherit from the BaseReq so each AdviceReq must have a unique MsgID

If successful, the lastResponse will contain the original response (the outstanding response that the client requires)

If unsuccessful the AdviceResp will have fault=True and the Fault will have either mustALR=True (meaning that another AdviceReq should be sent with a new MsgID but same adviceReqMsgID) or mustALR=False (meaning it is not necessary to perform the AdviceReq again as the response received is final and further Advice Requests will not change the outcome and in this case the terminalID will also be unlocked)

22. Test Cases

The following tests have been created on the TEST server to allow the client to develop their system and test the different vending scenarios.

The test system will return static responses to allow the Client to develop and test their system based on expected outcomes.

Test case responses are changed by changing input parameters such as meter numbers, customer numbers or purchase values. The following section describes the test cases available.

22.1 Electricity

22.1.1 ConfirmCustomer

Test Case #	Meter Number	Description
1	11112222333 or PDC65431	Will receive 1 result
2	11112222341	Will receive 2 results for same utility
3	11112222358	Will receive 3 results (2 for same utility and 1 for a different one)
4	11112222366	Fault (No response from Utility)
5	11112222374	Fault (Cannot connect to Utility)
6	any meter that starts with 222	Fault (Unknown Meter)
7	11112222382	Fault (Meter Blocked)
Other	* msno contains * 21 debts=200 23 debts = 0 else debts = ? * 32 fbe=due 34 fbe=notdue else fbe = ? * 43 maxvend=1000 45 maxvend=100 else ? * 46 minvend=50 47 minvend=1 else ?	

22.1.2 CreditVend

Test Case #	Meter Number	Description
1	11112223117	StandardTokenTx
2	11112223125	StandardTokenTx & FixedChargesTx
3	11112223133	StandardTokenTx & DebtPaymentTx
4	11112223141	StandardTokenTx & FixedChargesTx & DebtPaymentTx
5	11112223158	DebtPaymentTx
6	11112223166	DebtPaymentTx & FixedChargesTx
7	11112223174	StandardTokenTx & KeyChangeTokenTx
8	11112223182	StandardTokenTx (With FBE)
9	11112223190	StandardTokenTx (With FBE) & KeyChangeTokenTx
10	11112223208	StandardTokenTx (With FBE) & KeyChangeTokenTx & FixedChargesTx
11	11112223216	Fault (Insufficient Credit) mustALR=false
12	11112223224	Fault (No Response From Utility) mustALR=false
13	11112223232	Fault (No Response From Utility) mustALR=true (advice will return successful response)
14	11112223240	Fault (No Response From Utility) mustALR=false (advice will return a request not processed)
15	11112223257	Fault (Meter Blocked)
16	11112223265	Fault (Update Meter Key)
17	11112223273	Fault (Max Vend Limit Exceeded)
18	11112223281	Fault (Min Vend Limit Exceeded)

19	11112223299	Fault (Too Much Debt)
20	11112223307	Fault (No Customer Information)
Other		StandardTokenTx

22.1.3 TrialCreditVend

Trial Vend Test cases are exactly the same as CreditVend with the exception that TrialCreditVend is a non-Revenue Transaction and as such any fault returned will always have mustALR=false.

22.1.4 FBE

Test Case #	Meter Number	Description
1	11113333014	StandardTokenTx
2	11113333022	Fault (Unable to Connect to Utility) mustALR = false
3	11113333030	Fault (No Response From Utility) mustALR=true (advice will return successful response)
4	11113333048	Fault (Meter Blocked)
5	11113333055	Fault (No Customer Information)
6	11113333063	Fault (Key Change Required)
7	11113333071	Fault (No FBE Token is due)
8	11113333089	Fault (FBE Token must vend with Sale)
Other		StandardTokenTx

22.1.5 Reprint

Test Case #	Meter Number	Description
1	11114444018	Fault (Cannot Reprint)
Other	Any Previously generated receipt for a CreditVend on the Test Server will be available for Reprint. As such any meter where a CreditVendResponse was successful will have the receipt stored for reprint testing.	

22.1.6 UpdateMeterKey

Test Case #	Meter Number	Description
1	11115555010	KeyChangeTokenTx
2	11115555028	Fault (Meter Not Found)
3	11115555036	Fault (No Response From Utility)
Other		KeyChangeTokenTx

22.2 LiveAirtime

Test Case #	MSISDN	Description
1	0110001234	Successful Response
2	0110001235	Fault (Unable to Connect to Utility) mustALR=false
3	0110001236	Fault (No Response From Utility) mustALR=true (advice will return successful response)
4	0110001237	Fault (Empty or Invalid MSISDN) mustALR=false
Other	Any MSISDN of length 10=successful response	

22.3 PinlessVoucher

Test Case #	MSISDN	Description
1	0110001234	Successful Response with valid Product in Request

2	0110001235	Fault (Unable to Connect to Utility) mustALR=false
3	0110001236	Fault (No Response From Utility) mustALR=true (advice will return successful response)
4	0110001237	Fault (Empty or Invalid MSISDN) mustALR=false
Other	Any MSISDN of length 10 and valid Product = successful response	

22.4 AnyValueAirtime

Test Case #	Amount	Description
1	12.50	Fault (Unable to Connect to Utility) mustALR=false
2	13.50	Fault (No Response From Utility) mustALR=true (advice will return fault)
3	14.50	Fault (No Response From Utility) mustALR=true (advice will return successful response)
Other	Any valid amount will return successful response	

22.5 DSTV

22.5.1 DSTVGetSubsciberInfo

Test Case #	Customer Identifier	Description
1	8701015041008 | 100515 | 111555666	Valid Customer (Account State=Open)
2	8201015041007 | 100511 | 111555777	Valid Customer (Account State=Suspended)
3	750102042006 | 100522 | 111555888	Valid Customer (Account State=OverDue)
4	740102042005 | 100612 | 111666000	Valid Customer (Account State=Open)
5	730102042005 | 100613 | 111777000	Valid Customer (Account State=Open)
6	contains("777")	Fault (Unable to Connect to Utility)
Other	Fault (Empty or Invalid Customer Identifier)	

22.5.2 DSTVVendorPaymentTransction

Test Case #	Customer Identifier	Description
1	SUBS & customerNumber=100515	Successful Payment
2	SUBS & customerNumber=100511	Successful Payment
3	SUBS & customerNumber=100522	Successful Payment
4	SUBS & customerNumber=100612	Fault (No Response From Utility) mustALR=true (advice will return fault response)
5	SUBS & customerNumber=100613	Fault (No Response From Utility) mustALR=true (advice will return successful response)
6	TVOD & customerNumber= 100515	Successful Payment
7	TVOD & customerNumber= 100511	Successful Payment
8	TVOD & customerNumber= 100612	Fault (No Response From Utility) mustALR=true (advice will return fault response)
9	TVOD & customerNumber= 100613	Fault (No Response From Utility) mustALR=true (advice will return successful response)
Other	Fault (Empty or Invalid Customer Identifier)	

22.6 OTTVoucher

Test Case #	Amount	Description
1	> R200	Fault (Max Vend Limit Exceeded) mustALR=false
2	R99	Fault (Unable to Connect to Utility) mustALR=false

3	R98	Fault (No Response From Utility) mustALR=true (advice will return fault response)
4	R97	Fault (No Response From Utility) mustALR=true (advice will return successful response)
Other	Any valid amount will return successful response	

22.7 Talk360Voucher

Test Case #	Amount	Description
1	> R200	Fault (Max Vend Limit Exceeded) mustALR=false
2	R99	Fault (Unable to Connect to Utility) mustALR=false
3	R98	Fault (No Response From Utility) mustALR=true (advice will return fault response)
4	R97	Fault (No Response From Utility) mustALR=true (advice will return successful response)
Other	Any valid amount will return successful response	

22.8 EasyPay Bill Payments

22.8.1 EasyPayConfirm

Test Case #	Account Type and account Number	Description
1	epNo 91111222233319	Successful response
2	epNo 91111222233327	Successful response with correctAmount
3	epNo 91111222233335	Successful response with maxAmount and minAmount
4	epNo 91111222233343	Successful response with payable = false
5	noticeNo G4/12345/805/025659 | 4/12345/805/025659	Successful response
6	noticeNo 18/59041/806/118906	Successful response with correctAmount
7	noteiceNo A7/22863/806/046639	Successful response with payable = false
Other	if epNo and starts with 9 and is (20 digits | 16 digits) else if noticeNo [DD]/[SSSSS]/[LLL]/[CCCCCC] else error	

22.8.2 EasyPayBillPayment

Test Case #	Account Type and account Number	Description
1	Any Successful Test case of EasyPayConfirm will return successful response if correct Amount/Max Amount/Min Amount returned values are adhered to. Else response will be a Fault	
2	epNo 92222444433345	Fault (“No Response From Utility”) mustALR=true (advice response will return fault)
3	epNo 92222444433352	Fault (“No Response From Utility”) mustALR=true (advice response will return successful response)

22.8.3 EasyPayBillPaymentConfirm

Test Case #	Account Type and account Number	Description
1	Any Successful Test case of EasyPayBillPayment will return successful response if correct Amount/Max Amount/Min Amount returned values are adhered to. Else response will be a Fault	
2	epNo 97777444433319	Fault (“No Response From Utility”) mustALR=true (advice response will return fault)
3	epNo 97777444433327	Fault (“No Response From Utility”) mustALR=true

		(advice response will return successful response)

22.9 Moolapin

22.9.1 MoolaPinVoucher

Test Case #	Req Amount	Description
1	R12.00	Fault (“No Response From Utility”) mustALR=true (advice response will return fault)
2	R13.00	Fault (“No Response From Utility”) mustALR=true (advice response will return success)
Other	Any Valid amount will return successful response	

22.9.2 MoolaPinRedeem

Test Case #	Req Amount	Description
1	Any Unused moolapin obtained from MoolaPinVoucher will return a successful response except for the subsequent test cases	
2	R16	A Moolapin generated to the value of R16 will return a Fault (mustALR=true) when being used and the advice response will be successful(

22.10 SFX

22.10.1 SFXLookup

Test Case #	Payment Reference	Description
1	Contains (“7777”)	Successful Response
2	Contains (“6666”)	Fault (Cannot connect to server)
Other	Fault (Invalid reference)	

22.10.2 SFXPayment

The following use cases are only applicable to successful responses to SFXLookup and as such Payment references in the use cases include “777” as well as the described use case

Test Case #	Payment Reference	Description
1	777755550000	Successful Response
2	777788880000	Fault (Payment allready made) mustALR=false
3	777799990000	Fault (Unable to Connect to Utility) mustALR=false
4	777744440000	Fault(No Response From Utility) mustALR=true (advice returns fault)
5	777733330000	Fault(No Response From Utility) mustALR=true (advice returns successful response)
Other	Fault (Invalid reference)	

22.11 SBInstantMoney

Test Case #	Pin & Code	Description
1	111122223333444411 - 1234	Valid response returns R30
2	111122223333444422 - 1234	Valid response returns R50
3	111122223333444433 - 1234	Fault (No Response from Utility) mustALR=true (advice returns successful response)
4	111122223333444455 - 1234	Fault (No Response from Utility) mustALR=true (advice returns fault)

5	111122223333444466 - 1234	Fault (Transaction declined) mustALR=false
Other	Fault (No/Invalid PinNumber Supplied)	

22.12 Prepaid Vouchers

22.12.1 VoucherInfo

Any Valid VoucherCode will return valid response data.

For example VOD000005 or MTN000010

22.12.2 VoucherList

Any request will return a full list of voucher information

22.12.3 PurchaseVoucher

Any Valid request will return a voucher.

Test a request for MTN000015 will return a fault for No Stock.

22.12.4 PurchaseMultiVoucher

Any Valid request will return vouchers.

Test a request with MTN000015 will return a fault if partialFail=true

22.13 Pinless Vouchers

22.13.1 PinlessVoucherList

Any valid request will return list of available product codes.

22.13.2 PinlessVoucher

Test Case #	Request Msisdn	Description
1	0110001235	Unable to connect to Utility error Response (mustALR=false)
2	0110001237	Invalid MSISDN error Response (mustALR=false)
3	0110001236	No response from Utility (mustALR=true) thus Advice is required but advice will return a successful response (Voucher was purchased)
4	Any other MSISDN	Will return valid response for valid product code

22.14 AnyTimeVouchers

22.14.1 AnyTimeVoucher

Test Case #	Request Amount	Description (Expected Response)
1	R201.00	Any Amount greater than R200 will return Max Vend Limit Exceeded
2	R99.00	Unable to connect to Utility error Response (mustALR=false)
3	R98.00	No response from Utility (mustALR=true) thus Advice is required but advice will return a response that failed
4	R97.00	No response from Utility (mustALR=true) thus Advice is required but advice will return a successful response (Voucher was created)
5	Any Other Amount < R200.00	Will return a valid Voucher

22.14.2 AnyTimeVoucherRedeem

Test Case #	Pin Number	Description (Expected Response)
1	Length !=11 or starts with 1111	Pin Not found (Does not exist)
2	Pin Starts with 2222 & (length == 11)	No response from Utility (mustALR=true) this Advice is required but advice will return a successful response (Voucher was redeemed)
3	Pin Starts with 3333 & (length=11)	No response from Utility (mustALR=true) thus Advice is required but advice will return a response that failed
4	Any Other pin that has length=11 and does not start with above test cases	Successful response

22.15 IncompleteTransactionFault

Every example provided in the preceding Test cases that is a vend transaction (i.e. would debit or credit the aggregator account) and returns a Fault with mustALR=true will have the terminalID locked.

The subsequent request for the terminalID MUST be an Advice request in order to unlock the terminalID.

The Test server has been configured with the following rules

a. If the advice request is received within 30 seconds of the initial vend request then a “Server Busy” fault will be returned. A subsequent Advice Request should this be sent (this simulates the scenario where a utility is still busy with the transaction and more/ multiple advice requests are required to obtain the response.
b. If the advice request is received after 30 seconds of the initial vend request then the normal advice response (either successful or un-successful) will be returned.

```
<ConfirmCustomerResp xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://ravasvend.co.za/">
  <reqterminalMsgID>1</reqterminalMsgID>
  <reqterminalID>AC0001</reqterminalID>
  <reqMsgID>202101151429470000001</reqMsgID>
  <respDateTime>2021-01-15T14:29:59.1837354+02:00</respDateTime>
  <hasFault>true</hasFault>
  <fault i:type="IncompleteTransactionFault">
    <mustALR>true</mustALR>
    <faultnumber>8</faultnumber>
    <desc>Incomplete Transaction Fault</desc>
    <EN_dispHeader>Incomplete Transaction</EN_dispHeader>
    <EN_operatorMsg>There is a request that requires completion before further requests are
allowed</EN_operatorMsg>
    <EN_custMsg />
    <reqterminalMsgID>1</reqterminalMsgID>
    <reqMsgID>202101151429550020014</reqMsgID>
    <reqDateTime>2021-01-15T14:29:55</reqDateTime>
    <reqType>CreditVend</reqType>
  </fault>
  <Server>DESERTRETAIL</Server>
</ConfirmCustomerResp>
```

Table 1: Example IncompleteTranasctionFault Response

23. Test Case Validation

The following scenarios should be performed and the results thereof (request/response) must be provided to RA Cellular before production credentials will be provided.

23.1 Electricity

a) ConfirmCustomer then CreditVend for msno=11112223117
b) ConfirmCustomer then CreditVend for msno=11112223208
c) ConfirmCustomer then CreditVend and show Advice for msno=11112223232
d) ConfirmCustomer then CreditVend and show Advice for msno=11112223240
e) ConfirmCustomer then FBE for msno=11113333014
f) ConfirmCustomer then FBE and show Advice for msno=11113333030

23.2 LiveAirtime

a. Purchase for msisdn=0110001234
b. Purchase and show advice for msisdn=0110001236

23.3 AnyValueAirtime

c. Purchase for R20.00
d. Purchase and show advice for amount=13.50
e. Purchase and show advice for amount=14.50

23.4 DSTV

a. DSTVGetSubsciberInfo then DSTVVendorPaymentTransction for SUBS and IdNumber=8701015041008
b. DSTVGetSubsciberInfo then DSTVVendorPaymentTransction for TVOD and IdNumber=750102042006
c. DSTVGetSubsciberInfo then DSTVVendorPaymentTransction and show Advice for SUBS and IdNumber=740102042005
d. DSTVGetSubsciberInfo then DSTVVendorPaymentTransction and show Advice for SUBS and IdNumber=730102042005

23.5 OTTVoucher

a. Purchase for R25.00
b. Purchase then show Advice for request amount = R98.00
c. Purchase then show Advice for request amount = R97.00

23.6 Talk360Voucher

a. Purchase for R25.00
b. Purchase then show Advice for request amount = R98.00
c. Purchase then show Advice for request amount = R97.00

23.7 EasyPayBillPayments

a. EasyPayConfirm, EasyPayBillPayment then EasyPayBillPaymentConfirm for epNo=91111222233319
b. EasyPayConfirm, EasyPayBillPayment then cancel EasyPayBillPaymentConfirm for epNo=91111222233319
c. EasyPayConfirm, EasyPayBillPayment then EasyPayBillPaymentConfirm for noticeNo=4/12345/805/025659
d. EasyPayConfirm, EasyPayBillPayment then cancel EasyPayBillPaymentConfirm for noticeNo=4/12345/805/025659
e. EasyPayConfirm, EasyPayBillPayment then show advice for epNo=92222444433345

f. EasyPayConfirm, EasyPayBillPayment then EasyPayBillPaymentConfirm then show advice for epNo=97777444433319
g. EasyPayConfirm, EasyPayBillPayment then EasyPayBillPaymentConfirm then show advice for epNo=97777444433327

23.8 MoolaPin

a. MoolaPinVoucher for R16.00
b. MoolaPinVoucher and Advice for R12.00
c. MoolaPinVoucher and Advice for R13.00
d. MoolaPinRedeem R13.00 returned in (c.)
e. MoolaPinRedeem and Advice R16.00 returned in (a.)

23.9 SFX

a. SFXLookup and SFXPayment for 777755550000
b. SFXLookup and SFXPayment for 777788880000
c. SFXLookup and SFXPayment and Advice for 777744440000
d. SFXLookup and SFXPayment and Advice for 777733330000

23.10 SBInstantMoney

a. SBInstantMoney for 111122223333444411 - 1234
b. SBInstantMoney and Advice for 111122223333444433 - 1234
c. SBInstantMoney and Advice for 111122223333444455 – 1234

24. Error List

Code	Description	Operator Message
1	Login Failed	Login Failed
2	Server Off	Server is currently offline, please try again later
3	Terminal Msg ID Error	Terminal Msg ID Is null, empty or too long
4	Terminal ID Error	Terminal ID Is null,empty or too long
5	Request Msg ID Error	Request Msg ID Is null,empty or too long
6	Meter Identifier Error	Meter Identifier Is null or empty
7	Duplicate Msg ID Error	Duplicate Msg ID Exists on the Server
8	Incomplete Transaction	There is a request that requires completion before further requests are allowed
9	No Response From Utility	The Utility did not respond to the request
10	Unknown Meter	The Meter is unknown/not registered. Contact the Support services for assistance if required.
11	General Error	
12	Meter Blocked	Meter Blocked. Contact the Support services for assistance if required.
13	Duplicate Meter	Duplicate Meter. The Meter Number is supported by more than 1 supplier
14	Update Meter Key	Update meter key. Information supplied not the same as database. First Update Meter Key and encode Meter Card.
15	System Exception	
16	Unable to Connect to Utility	Connection to the Utility failed, please try again"
17	Purchase Value Error	The Purchase Value supplied is Null, Empty or Negative, Please correct and try again
18	Receipt Format Error	The Receipt Format Selected was null or empty, Please correct and try again
19	Vendor Credit Error	The Vendor does not have enough credit at the Utility to complete the request
20	Key Change Required	There are pending Key Change Tokens that need to be printed. Contact the Support services for assistance
21	Max Limit Exceeded	The Tendered Amount has exceeded its maximum value. Please decrease the request amount and try again.
23	Server Busy	The Utility server is busy, please try again later
24	Min Limit Exceeded	The Tendered Amount is less than the minimum value. Please increase the request amount and try again.
25	Too Much Debt	Too much debt. An account with a debt to be repaid is linked to this meter. This debt must be repaid before vending can take place.
26	Invalid Amount	The upper or lower limit on the amount of elec that may be purchased in a single transaction or over a period of time has been passed.
28	Cannot purchase again for 30 min	A purchase was made for the same amount less than 30 minutes ago. Please change the amount or wait 30 min. If you are missing a transaction please do a reprint
29	No FBE Token is due	No FBE Token is due
31	Invalid Tender Type	The tender type specified in the request in not recognized.
33	FBE Token must vend with Sale	A FBE token cannot be obtained unless a standard resource token is purchased.
34	No Customer Information	There is no history of a ConfirmCustomerRequest for the supplied MeterIdentifier. Please perform a ConfirmCustomerRequest first
35	Insufficient Funds	The Aggregator has insufficient funds to complete the transaction.
36	Advice Request Msg ID Null or Empty	The adviceReqMsgID required to perform the AdviceLastRequest is null or empty
37	The Request was never processed.	The Request was never processed by the server. Normal Vending can continue
38	The Request was never Received.	The Request was never received by the server. Normal Vending can continue
39	Advice Request Type Error	The adviceReqMsgID supplied is for an AdviceRequest, Please supply the MsgID of the original request to continue
40	The Last Vend cannot be found	Last Vend cannot be found in server, it may be that it happened to long ago
41	Request VoucherCode Error	Request VoucherCode Is null or empty

42	Request VoucherCode Error	Request VoucherCode Not found or Disabled
43	Cannot Vend	There was an error trying to vend to the meter, please try again
44	Cannot Reprint	There was no transactions found to reprint
45	Advice Request Type Error	The adviceReqMsgID supplied is for a type of request where there is no Advice Procedure, Please supply the MsgID of the original request to continue
46	Update Meter Key Service not Available	For Meter Management please contact the Utility Directly
47	Invalid Response received from the Server	Invalid Response received from the Server, Please try again
48	Meter Identifier Invalid	Meter Identifier Invalid, Please check and try again
49	Unknown Meter	The Meter is unknown. The Supplier may be offline. Please try all Suppliers or try again later if the problem persists.
50	Timeout Error	The Utility did not respond in time, Please try again
51	Unable to route request	Unable to find a Supplier for the vend. The Supplier may be offline. Please try all Suppliers or try again later if the problem persists.
52	Empty or Invalid Network	Empty or Invalid Network supplied, Please correct and try again
53	Empty or Invalid TopupType	Empty or Invalid TopupType supplied, Please correct and try again
54	Empty or Invalid MSISDN	Empty or Invalid MSISDN supplied, Please correct and try again
55	Unable to route request	Unable to find a Supplier for the request that is online, Please try again later.
56	Empty or Invalid Customer Identifier	Empty or Invalid Customer Identifier supplied, Please correct and try again
57	Empty or Invalid Payment Type	Empty or Invalid Payment Type supplied, Please correct and try again
58	Response VoucherCode Error	Response VoucherCode Not found or Disabled
59	Payment Fee Value Error	The Payment Fee Value supplied is Null, Empty or Negative, Please correct and try again
60	Empty or Invalid Account Number	Empty or Invalid Account Number supplied, Please correct and try again
61	Payment Amount Value Error	The Payment Amount Value supplied is Null, Empty or Negative, Please correct and try again
62	Invalid Response received from the Server	Invalid Response received from the Server, Please try again or contact support of the problem persists
63	Empty or Invalid Account Type	Empty or Invalid Account Type supplied, Please correct and try again
64	Receiver not found for accountNumber	Receiver not found for accountNumber, please do a Confirm Transaction First
65	Empty or Invalid tranID	Empty or Invalid tranID supplied, Please correct and try again
66	Invalid PurchaseValue	The PurchaseValue must match the original request, Please correct and try again
67	Transaction Cancelled	Transaction Cancelled, Please do a new request if payment should be submitted
68	No/Invalid PinNumber Supplied	Please enter a valid PinNumber and retry
69	MoolaPin Used	MoolaPin Used
70	Service Fee Invalid	Service Fee supplied Invalid. Please correct the request amount and try again.
71	Payment Reference Invalid	Payment Reference supplied Invalid. Please correct the request and try again.
72	Customer Name/Surname invalid or missing	Customer Name/Surname invalid or missing. Please correct the request and try again.
73	User Pin invalid or missing	User Pin invalid or missing. Please correct the request and try again.
74	Transaction Declined	Transaction Declined. Please correct data request and try again.
75	Invalid Fault Type	Invalid Fault Type. Please correct the request and try again.
76	Empty or Invalid Token	Invalid Token Supplied. Please correct the request and try again.
77	No Stock	No Stock of requested voucher, Please try again later.
78	No or Invalid Order	No or Invalid Order, Please correct and try again
79	Invalid Product	No or Invalid Product, Please correct and try again
80	Invalid Quotation	No or Invalid Quote Supplied or Data doesnt match stored Quote, Please correct and try again
81	Unknown MSISDN	Unable to determine network for MSISDN, please confirm MSISDN and try again
82	Empty or Invalid Voucher Number	Empty or Invalid Voucher Number supplied, Please correct and try again
83	Insufficient Voucher Funds	There are insufficient funds available on the voucher to complete the tranasction

84	Invalid Voucher Type	The voucher type is not valid to be spent at this merchant.
85	Empty or Invalid eMail	Empty or Invalid eMail supplied, Please correct and try again
86	Empty, Short or Invalid searchValue	Empty, Short or Invalid searchValue supplied, Please correct and try again
87	Cannot find Client	The Search Provided did not return valid client details.
88	No Products Available	No Products Found, Please try again
89	Pin Mining Suspected	Pin Mining Suspected, Account has been blocked/suspended
90	Voucher Allready Used	Voucher Allready Used
91	Voucher is reserved	Voucher is reserved
99	ThirdParty Exception	ThirdParty Exception

25. Appendix

25.1 EasyPay Number Validation

Every account which is payable at a Client must have a number, referred to as the EasyPay number, which will be keyed in at the POS.
An EasyPay number will have the following structure:

9[RRRR][AA...A][C]

- 9 is the Easy Pay number prefix – all Easy Pay numbers MUST begin with a 9
- RRRR = Unique 4-digit Receiver Identifier
- AA...A = 1 to 14-digit Receiver account reference
- C = Modulus 10 Luhn Check Digit

An EasyPay number may contain numeric digits only. Account numbers may range from 1 to14 digits, therefore the total length of an EasyPay number may range from 7 to 20. However, for each Receiver Identifier, the account length *must* be fixed - ie. if an account length of 10 is used for Receiver 1234, then all EasyPay numbers starting with 91234.... *must* use 10 digits for the account - padding with 0's if necessary. (all active Receiver ID"s with their respective account lengths are found in the Receiver parameter file or FFR file which is sent to POS and maintained by Easy Pay)

The check digit is calculated on the Receiver Identifier (RRRR) and the Account Number (AA...A) - the leading 9 is *not* included.

Starting from the last digit to the first, double each digit in an odd numbered position and subtract 9 if the product is larger than 9. Add up all the even-numbered digits and all the previous products. The check digit will be the number needed to make the sum a multiple of 10.

Examples

1) Receiver Identifier = **6789**, Account number = **25433678212333**

67892543367821233 3
| | | | | | | | | |

658921463377222636 == Add ==> 82 ==> Check digit will be 90-82 ==> **8**

EasyPay number: **9678925433678212333 8**

2) Receiver Identifier = **1500**, Account number = **123456789**

1500123456789
| | | | | | | |

2500226416589 == Add==> 50 ==> Check digit will be 50-50 ==> **0**

EasyPay number: **915001234567890**

25.2 Notice Number Validation

There are a number of different formats for Notice Numbers they include:

- The General Notice Number
- The 'six digit serial number' Notice Number

At time of writing EasyPay can accept the General Notice and 'six digit serial number' Notice Numbers.

The General Notice Number

A General Notice number will have the following structure:

[DD]/[SSSSS]/[LLL]/[CCCCCC]

- DD = Document Type. This field is used by the local authorities to differentiate between different infringement types. The field length is 2 and the first character *might* be an alphabetic character. If the POS is unable to process alphabetic characters this field can be omitted.
- SSSSS = Serial Number. This part of the general notice number is similar to the account number portion of the EasyPay Number. 5 numeric digits.
- LLL = Local Authority Code Uniquely identifies the Local Authority issuing the infringement notice. 3 Numeric digits.
- CCCCCC = is a checksum that is calculated on the preceding digits of the general notice number.
- Regular Expression: `^([1-9A-N][0-9A-N])/([0-9]{5})/([0-9]{3})/([0-9]{6})$`

A General Notice number may contain alphabetic and numeric digits delimited according to the above pattern by a „/" character. Either of the first 2 characters can be Alphabetic, although in practice it is expected that only the first character could be alphabetic.

The delimiting characters should be captured as it assists in the identification and validation of the General Notice Number.

The 'six digit serial number' Notice Number

An 'six digit serial number' Notice number will have the following structure:

[DD]/[SSSSSS]/[LLL]/[(C)CCCCCC]

- DD = Document Type. This field is used by the local authorities to differentiate between different infringement types. The field has a length of 2 and the characters are alpha numeric.
- SSSSSS = Serial Number. This part of the general notice number is similar to the account number portion of the EasyPay Number. 6 numeric digits.
- LLL = Local Authority Code Uniquely identifies the Local Authority issuing the infringement notice. 3 Numeric digits.
- (C)CCCCCC = is a checksum that is calculated on the preceding digits of the general notice number. it is between either 6 or 7 digits long. This check sum is padded to 6 digits to the left with 0's.
- Regular Expression: `^([1-9A-N]{2})/([0-9]{6})/([0-9]{3})/([0-9]{6,7})$`

An 'six digit serial number' Notice number may contain alphabetic and numeric digits delimited according to the above pattern by a „/" character. Any of the first group of characters can be Alphabetic.

The delimiting characters should be captured as it assists in the identification and validation of the 'six digit serial number' Notice Number.

In addition to the checks on the structure of the number (length and position of delimiting characters and the segments they delimit), The check sum should be calculated and compared to the checksum captured by the POS devise.
The checksum Algorithm uses the integer values of each portion of the 'six digit serial number' Notice number as follows:

$$C = 2*SSSSSS + DD + LLL$$