Smartload Webservice V2

Technical Interface Specification

Version 1.2.3

Document Change History

Issue	Revision	Date	Author	Reason for Change
1	0.0	2015-05-07	R Grasmuck	This document is based on the existing version 1 webservice since 90% of the content is the same. Additional features and changes have been added.
1	1.1	2015-06-10	R Grasmuck	Modified the context to SmartcallServices2
1	1.2	2016-12-19	R Grasmuck	Added Prevend Functionality
1	1.3	2017-01-30	R Grasmuck	Added CancelRecharge Functionality as well as cleaning up formatting and numbering issues

Contents

Abbreviation List

FTP	File Transfer Protocol
LBS	Location Based Services
LES	Location Enhanced Services
MSISDN	Mobile Subscriber Integrated Services Digital Network Number (or cell phone number)
WSDL	Webservices Description Language
RPC	Remote Procedure call
SOAP	Simple Object Access Protocol
Jboss AS	Java Application server
USSD	Unstructured Supplementary Service Data
WS	Webservice
HTTPS	Hypertext Transfer Protocol Secure
SSL	Secure Socket layer
XML	Extensible Mark-up Language
GSM	Global System for Mobile Communications
WAP	Wireless Application Protocol
SIMs	Subscriber Identity Cards
WSIT	Webservice Interoperability Technologies
ICCID	Integrated Circuit Card ID 19 or 20-digit serial number of the SIM card.

1. Introduction

This document describes Smartcall's Smartload Webservice which provides the opportunity for dealers to directly access Smartload by creating their own client interface.

The purpose of this document is to provide a clear technical guideline on how to connect to the webservice and how to call the individual functions as provided by the webservice. This would cover amongst others, calls to perform and query individual airtime recharge requests.

Smartload specific details and the associated rules are covered in the complementary Business Specification document. The Business Specification document should be read in conjunction with this document when implementing your own client solution.

2. Audience

This document is for developers wishing to create a secure client to connect to Smartcall's Smartload Webservice and submit Smartload recharge and operation requests.

3. Version 2 upgrade

Smartcall has released this upgrade to its version 1 webservice thereby solving a few problems experienced with its predecessor as well as some new features. The new webservice runs in conjunction with the old service and existing users need not do anything. It's however recommended that existing users of the old service upgrade to the newer service in due time since the long term plan is remove version one entirely.

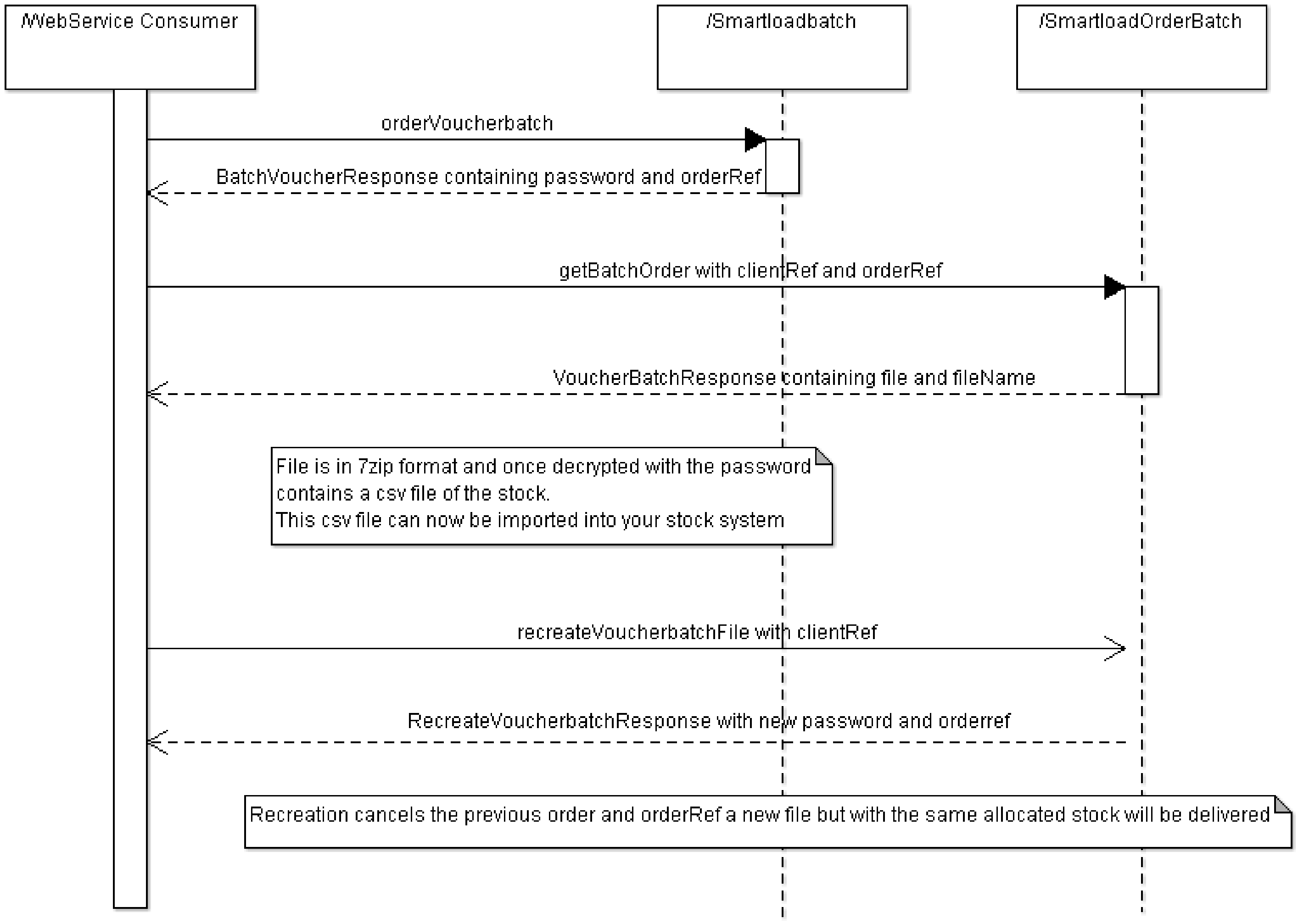
New users will only be set up on the new service and all new documentation provided will be for the version 2 service

The important differences are

1) Addition of the ordering of bulk vouchers in file format. Users can now request and order of more than one file and then receive a single file with the vouchers asynchronously
2) New products and offerings will no longer require a regeneration of client code as they are no longer represented by an enum

4. Ordering of vouchers in files

The ordering of vouchers encompasses as an asynchronous process whereby the user requests a file for a certain amount and denomination. An order number and password is returned, which is then used to query for the file.
After a certain period the file is ready and by calling the orderRequestBatch request, the password protected file in 7zip file is returned. The process is displayed below.


/WebService Consumer
/Smartloadbatch
/SmartloadOrderBatch
orderVoucherbatch
BatchVoucherResponse containing password and orderRef
getBatchOrder with clientRef and orderRef
VoucherBatchResponse containing file and fileName
File is in 7zip format and once decrypted with the password contains a csv file of the stock. This csv file can now be imported into your stock system
recreateVoucherbatchFile with clientRef
RecreateVoucherbatchResponse with new password and orderref
Recreation cancels the previous order and orderRef a new file but with the same allocated stock will be delivered

Format of Vouchers

Vouchers are supplied in a CSV file with the following fields (in order)
Pin : number that needs to be part of the redemption process
Batch : used to trace stock
VoucherNumber : unique number needed when querying the voucher with the network
Expiry Date
Voucher description : simple text description

5. Sample Demo application

In order to help users of the webservice get a working prototype up quicker, as well as to provide some reference implementation, Smartcall has released a demo application, together with its source code, that calls all the possible functionality provided by the webservice.

The idea of this application is that a developer can see firsthand how integration is done and how the calls are made. In addition, an easy method of generating public/private keys is provided. The application is for demo purposes only and is not meant for production use, especially since the database is only a in memory database.

The demo application has been written in Java 1.8, using JavaFX, Apache CFX and Maven. The project is split into two parts, whereby the webservice integration with cfx forms its own project (WebServiceClient). Users can just use this part and incorporate this in their projects.

Both projects can be retrieved from github at

https://github.com/smartcall01/webservice2-sample-demo

Users can use this source code and build versions for different operating systems if they require.

6. Authentication

Encrypting a webservice can be done in two different ways. First, you can use SSL to transport messages using HTTPS. You can also use WS-Security; the contents of the message are encrypted by the JAX-WS implementation on both the client and the server. These two methods are illustrated in figure 2, below.

Smartcall has used WSIT from the Metro stack to implement WS-Security, which provides the service with better performance and allows for interoperability between the Java platform and Windows Communication Foundation (WCF) (aka Indigo) in .NET 3.0 and .NET 3.5. We use the Mutual Certificates Security mechanism which adds security via authentication and message protection that ensures integrity and confidentiality. When using mutual certificates, a keystore and truststore file must be configured for both the client and server sides of the application. Both the client-side and server-side use X509 certificates to authenticate to each other. The client request is signed using the Client's X509 certificate, then signed using an ephemeral key. The webservice signs the response using keys derived from the client's key.

When the client creates their private X509 certificate they must create the Distinguished Name with a CN (Common Name) value of the dealer MSISDN. Public keys/cert created from both the client and Smartcall are exchanged.

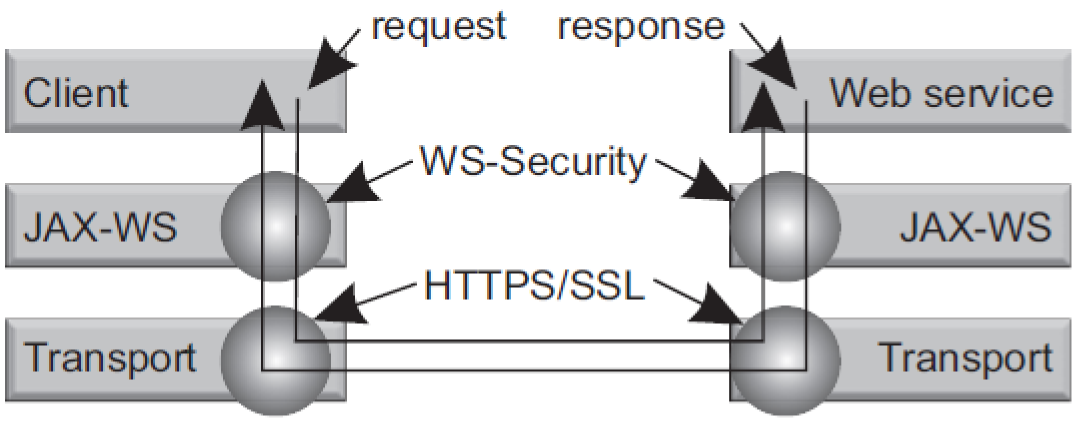


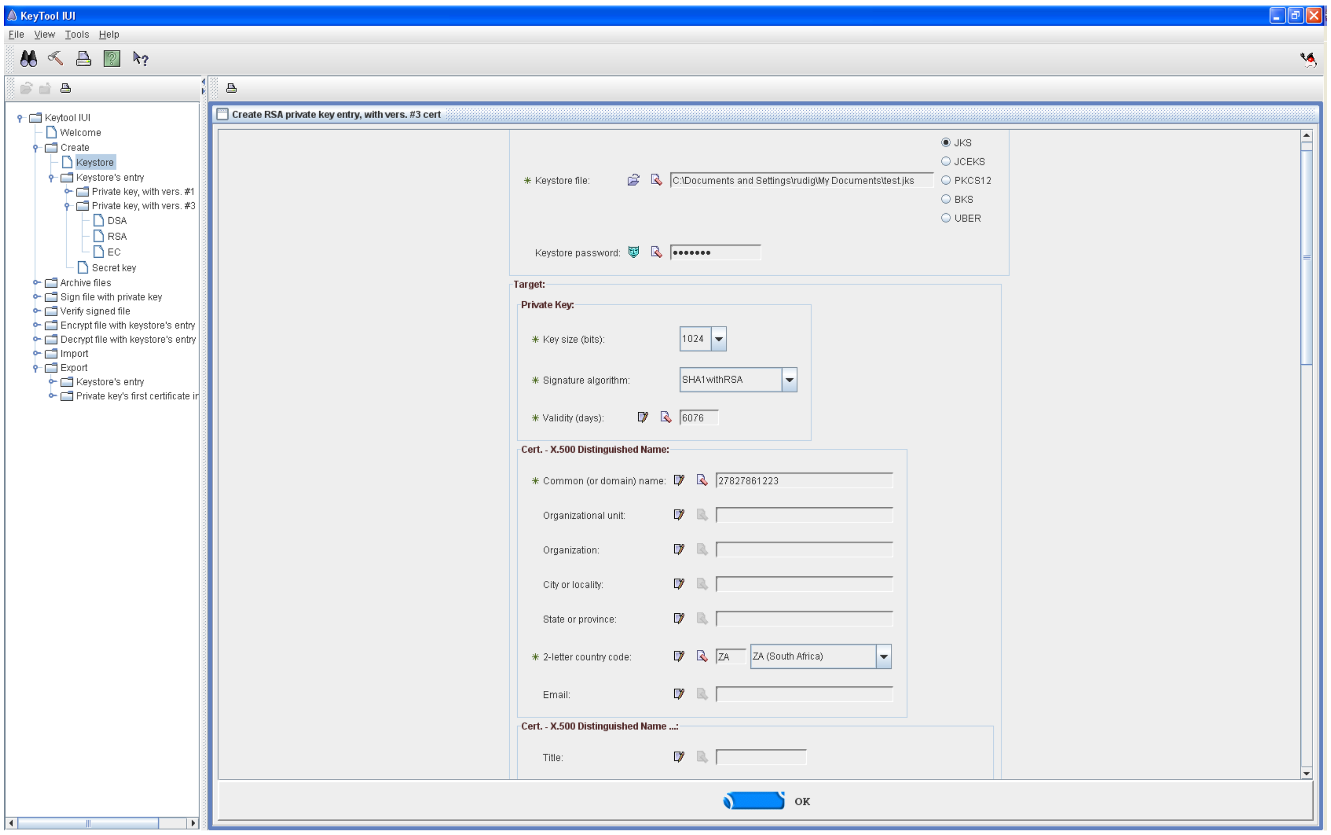
Figure 1 Basic encryption on the various levels

This certificate is then used to authenticate and authorise the user, particularly for Smartload against an LDAP server. Users are no longer required to supply Smartload credentials on each connection. The signature would be mapped to a Smartload cell phone number and pin.

Please note the following:

1) Please set the validity of the key to at least 5 years
2) The CN number must include the international format e.g. 27821234567
3) Key size should be at least 1024 bits

We have included an excerpt from the ktl tool the author used to generate test keys to give you a visual idea. This tool can be supplied if requested. In addition the sample app provides code that can be run or used to generate a key pair in the right format


KeyTool IUI
File View Tools Help
Keytool IUI
Welcome
Create
Keystore
Keystore's entry
Private key, with vers. #1
Private key, with vers. #3
DSA
RSA
EC
Secret key
Archive files
Sign file with private key
Verify signed file
Encrypt file with keystore's entry
Decrypt file with keystore's entry
Import
Export
Keystore's entry
Private key's first certificate ir
OK

Create RSA private key entry, with vers. #3 cert

Keystore file: C:\Documents and Settings\rudig\My Documents\test.jks

[✓ JKS] JCEKS PKCS12 BKS UBER

Keystore password: •••••••

Target:

Private Key:

Key size (bits): 1024

Signature algorithm: SHA1withRSA

Validity (days): 6076

Cert. - X.500 Distinguished Name:

Common (or domain) name: 27827861223

Organizational unit:

Organization:

City or locality:

State or province:

2-letter country code: ZA ZA (South Africa)

Email:

Cert. - X.500 Distinguished Name ...:

Title:

Requirements

<u>.NET Clients</u>
.NET 3.5
WCF
<u>Java Clients</u>
JDK 1.6
METRO 1.3 or higher or Apache CFX

Please refer to the following references to guide you in your environment if you need it

.Net
http://msdn.microsoft.com/en-us/library/ms733102.aspx
http://wcfsecurityguide.codeplex.com/

Java
https://wsit.dev.java.net/

7. Interfaces

Commands and responses

Below is a complete list of the webservice requests for Smartload in the version 2 interface. Function calls mapping to the WSDL calls are highlighted.

There are 3 WSDLS making up the total offering.

1) SmartloadService – all single transaction requests and basically corresponds to the version 1 offering
2) SmartloadOrderService - requests and queries requiring multiple vouchers in csv format
3) SmartBatchService – the delivery of the csv files

The left column contains the business functionality description, while the right column maps the function to the corresponding webservice method call, parameters and the return parameters.In terms of the interpretation of the return parameters please refer to the Business Specification document.

SmartloadService

Perform Airtime recharge Please refer to the Business Specification Document for a list of products, alternatively you can call the getNetworks, GetProductTypes and then the getProducts commands to determine what you can pass as a product.	**Request name** **performProductRechargeWithClientReference** **Required parameters:** • ProductRechargeRequest (recipientMsisdn, rechargeDeviceId, productId, amount, pinless, sendSms) • clientReferenceNumber a) amount: Please refer to the Business Specification document for the valid amounts per network b) pinless: The default is pinless (true), but on some networks we do offer voucher PINs, in which case this would then be false. c) deviceId: This is the cell phone / electricity meter being recharged. d) sendSms: You may opt to send the response SMS yourself in the case of voucher PIN recharges. You would then set sendSMS to false. The default is true. e) smsRecipientMsisdn. In the case where the deviceId is not the same as the recipient of the SMS you can enter the msisdn here **Return parameter:** RechargeResponse • Recharge (balance, batchNumber,boxNumber, orderReferenceId, ticketNumber, voucherPin, expiryDate) The Smartcall generated orderReferenceId is a unique number for all Smartload transactions that can be used when querying a recharge with Smartcall's customer care. In the case of pinless airtime ticketNumber,batchNo,boxNo and CardNo and expiryDate will not be set. *BaseResponse

Querying a Smartload dealer's last transaction whether recharge or transfer Retrieves the details of the last transaction that was performed by the dealer i.e. either last transfer or last recharge.	**Request name :** getLastTransaction **Required parameters:** no parameters **Return parameter:** LastTransactionResponse with Transaction *BaseResponse

Querying a Smartload dealer's transaction by the reference number Retrieves the details of the transaction for the valid reference number. This number is returned from each perform recharge call.	**Request name :** getLastTransactionForReference **Required parameters**: • queryReference: This is the client reference number **Return parameter**: LastTransactionResponse with Transaction *BaseResponse

Querying a Smartload dealer's transaction by the client reference number Retrieves the details of the transaction for the reference number as generated by the client. This is particularily useful when a request is sent but no response was received for whatever reason.	**Request name :** getLastTransactionForClientReferenceResponse **Required parameters**: ClientReferenceNumber **Return parameter**: LastTransactionResponse with Transaction *BaseResponse

Querying a Smartload dealer's account balance Retrieves the dealer balance in Rands.	**Request name :** getDealerBalance **Required parameters:** no parameters **Return parameter:** • DealerBalanceResponse (amount/balance in Rands) *BaseResponse

Checking dealer's registration Checks if the cell phone number is registered or not with Smartload. You can check any cell phone number if it is registered for Smartload. If it is not, you can either do a transfer or deposit money for the cell phone number. Smartcall will also register users on request though the process can take longer.	**Request name :** **isDealerRegistered** **Required parameters:** • queryMsisdn **Return parameter:** DealerRegisteredResponse (isRegistered) *BaseResponse

Perform Smartload funds transfer Transfer Smartload funds between two dealers Smartload funds can be transferred between two Smartload dealers with no fee charged or discount applied. An example: Dealer A (0721234567) with a Smartload balance of R100 can do a funds transfer of R55 to dealer B (0831234567) who in turn has a Smartload balance of R15. After performing the funds transfer the new Smartload balance for dealer A= R45 and dealer B= R70	**Request name :** **performFundsTransfer** **Required parameters:** FundsTransferRequest • Amount • RecipientMsisdn • SendSms (notify recipient) **Return parameter:** FundsTransferResponse • CurrentDealerBalance • NewDealerBalance (this returns 0 for security reasons) *BaseResponse

Return networks/providers This will returned a current list of Networks and providers that we currently service. e.g. Vodacom,Eskom,MTN,Telkom Mobile etc	**Request name :** **getNetworks** **Required parameters:** • none **Return parameter:** • A list of Network Objects that should be used in queries together with the productType and product tree structure

Return product types This will returned a current list of all products types that are supported by the passed network (as obtained from above) e.g. Airtime,Data Bundles,Insurance etc	This is now returned as part of the Network Object returned above and a getProductTypes on the network object will a return a list of all product types for this network

Return products This will returned a current list of all products that are supported by the passed network and product types. This would be used in the product recharge request above. (performProductRechargeWith – ClientReference)	This is now returned as part of the ProductType Object returned above and a `getProducts` on the network object will a return a list of all product types for this network

SmartloadBatch

Order voucher batch Order vouchers by supplying a productId and the number of vouchers required. A password for the expected future file and a Recharge object is returned describing the request. (containing a orderReferenceNumber that is required later)	**Request name :** orderVoucherBatch **Required parameters:** • batchVoucherRequest • clientReference (client generated) **Return parameter:** BatchVoucherResponse object *BaseResponse

Recreate voucher batch In the advent of a possible files lost you can reorder the file with the already used clientReferenceNumber. A new file,password and orderRefNumber is generated	**Request name :** recreateVoucherBatch **Required parameters:** • clientReference (client generated as supplied in the orderVoucherRequest) **Return parameter:** BatchVoucherResponse object *BaseResponse

SmartloadOrderBatch

<u>Get Batch Order</u> This method is called asynchronously to receive the file. If the file is available the response will contain a SUCCESS and the file is contained within the message	**Request name :** getbatchOrder **Required parameters:** • clientReference (client generated as supplied in the orderVoucherRequest) • orderReference (as supplied in the BatchVoucherResponse) **Return parameter:** VoucherBatchResponse object (file,filename and the orderReferenceId) *BaseResponse
<u>Perform Airtime Prevend</u> Parameters and call look exactly as a perform recharge request. However the request is sent directly to the individual network and a PrevendResponse is returned.	**Request name :** performPrevendWithClientReference **Required parameters:** • ProductRechargeRequest : as above • clientReferenceNumber : as above Please use the same client reference number when performing the actual recharge request **Return parameter:** `PrevendResponse` object ResponseCode and unfiltered message from network
<u>Recharge Cancellation</u> This is aimed largely for Web Service customers who may want to cancel a recharge which remains in pending state for too long. A recharge may only be cancelled if it has not been submitted to the network. As soon as the recharge is submitted, then it must follow the normal procedures which cannot be bypassed. A typical use case would be a customer has asked for a recharge, yet the network is experiencing technical difficulties. Normally the recharge would be submitted as soon as the technical difficulty have been resolved. By cancelling it successfully the customer can be refunded, without waiting for the network to recover and no loss is incurred by the vendor	**Request name** performCancelRecharge **Required parameters:** • `CancelRechargeRequest containing the order` reference Number as returned by the original request **Return parameter:** `CancelRechargeResponse` object ResponseCode and if Error with message if the request was unsuccessful

*BaseResponse → all returns extending from this contain an Error and ResponseCode Object
The ResponseCode has returns of SUCCESS,APP_ERROR,SYS_ERROR
The Error Object will have any error messages and codes if an error has occurred

8. Prevend

This involves a direct submission to those networks that support the prevend functionality. Currently this is only Vodacom and MTN.
A response code for the submission as well the actual unfiltered message from the network is returned.
We will return a response code of 'SUCCESS' if the network indicates the recharge will go through.
If the network replies that the recharge will not go through, a 'APP_ERROR' response code is returned.
Network, provider and infrastructural errors will return 'SYS_ERROR'

Vodacom requires the actual recharge performed in conjunction with a previous prevend to have the same reference number and it is imperative that the same clientReference is used.

9. Error Codes

As an initial layer the webservice can generate errors before they are submitted to the backend Smartload system. The errors below are generated via the webservice and not Smartload as such and are therefore covered here. Smartload and recharge errors are covered in the Business Specification document.

Recharge

DUPLICATE_RECHARGE(1001, "Duplicate recharge") – Duplicate client reference number. This is verified already on the webservice side

RECHARGE_TIMEOUT (1003, "Recharge response timed out, query to validate if recharge was submitted") – An Internal timeout on Smartcall's side. Please do query the recharge to determine whether it has been received and processed successfully as the request has been submitted but with an internal timeout on the response.

RECHARGE_FAILED (1004, "Recharge Failed") – An internal Smartcall error has occurred on submitting to the backend. The recharge most probably did not succeed. Please still however query the recharge for a definitive status.

Query

CLIENT_REFERENCE_DOES_NOT_EXIST(1005, "ClientReference does not exist") – This is verified already on the webservice side, before being submitted to Smartload

System Error

DEFAULT_SYSTEM_RESPONSE_CODE = 9999 Any major unexpected error with the webservice that can be caught caused this error code to be returned. Please report this as soon as you can.

10. WSDLs

NB: The below Endpoint Addresses could change. Smartcall will provide the correct IP or domain name as they change. These addresses were valid at the time of writing

Endpoint Address test addresses
http://www.smartcallesb.co.za:8091/SmartcallServices2/SmartloadService?wsdl
http://www.smartcallesb.co.za:8091/SmartcallServices2/SmartloadOrderService?wsdl
http://www.smartcallesb.co.za:8091/SmartcallServices2/SmartBatchService?wsdl

The test account on the test address will be given a basic balance for testing. Please note that SMS's will not be delivered on this system, and recharges will not be done to the recipient cell phone numbers.

Endpoint Address live addresses
http://www.smartcallesb.co.za:8090/SmartcallServices2/SmartloadService?wsdl
http://www.smartcallesb.co.za:8090/SmartcallServices2/SmartloadOrderService?wsdl
http://www.smartcallesb.co.za:8090/SmartcallServices2/SmartBatchService?wsdl