



iVeri Client Developers Guide

(Version 4.0.2)

SPECIFIC TO:

iVeri Client .NET 4.0.2

Table of Contents

1	Revision History	7
2	Introduction	9
2.1	iVeri Enterprise	9
2.2	File Transfer	9
2.3	Security	9
2.4	Merchant Requirements	9
2.5	Terminology	10
2.6	iVeri Client Requirements	10
3	iVeri Client Configuration	11
3.1	Make contact with an iVeri Distributor and obtain a User Group (Billing Details ID)	11
3.2	Install the relevant runtime	11
3.3	Run the iVeri Client Configuration	11
3.4	Ensure valid internet connection (with sufficient bandwidth) to the iVeri Gateway	12
3.5	Default Location (iVeriClient_Home)	12
3.6	Configure Proxy Server (if necessary)	12
3.7	Capture Default CertificateID	13
3.8	Set ClientAuthentication	13
3.8.1	CertificateSource (Java)	14
3.8.2	CertificateSource (Dotnet)	14
3.9	Perform an ApplicationID Ping (Enterprise specific)	15
3.10	Develop and Test the Application	15
3.11	Perform 1 cent Transaction (Enterprise specific)	15
3.12	Switch to "Active"	15
3.13	Move to Production machine	15
4	Important Security Overview	16
4.1	Security in iVeri Client .NET	16
4.2	Security in iVeri Client .Java	16
5	Payment mechanisms : PAN vs Track2 vs PIN	17
5.1	PAN (also known as "Card Not Present" or "Keyed")	17
5.2	Track2 (also known as "Card Present" or "Swiped")	17
5.3	PIN (also known as "Card Present" with PIN or "Swiped" with PIN)	17
6	Commands and Actions	18
6.1	Commands	18
6.1.1	Transaction commands (may cause transfer of funds)	18
	Authorisation (also known as "Preauthorisation")	18
	Authorisation Reversal	18
	Debit (also known as "Sale" or "Purchase")	18
	Credit (also known as "Refund" or "Pay")	18
	Void	18
6.1.2	Enquiry commands (do not cause transfer of funds)	18
6.1.3	FileTransfer commands	18
6.2	Actions	19
6.2.1	Authorisation with PAN	19
6.2.2	Authorisation with Track2	19
6.2.3	Additional Authorisation with TransactionIndex	19
6.2.4	Authorisation Reversal with TransactionIndex	19
6.2.5	Debit with PAN	19
6.2.6	Debit with Track2	19

6.2.7 Debit with PIN.....	20
6.2.8 Debit with TransactionIndex.....	20
6.2.9 Credit with PAN.....	20
6.2.10 Credit with Track2.....	20
6.2.11 Credit with TransactionIndex.....	20
6.2.12 Void.....	20
6.2.13 Balance Enquiry with PIN.....	20
6.2.14 PAN Enquiry with PAN.....	20
6.2.15 PAN Enquiry with Track2.....	20
6.2.16 PANToken Enquiry.....	20
6.2.17 Ping.....	20
6.2.18 Get Device PIN Key.....	20
6.2.19 ThreeDSecureCheckEnrollment.....	21
6.2.20 ThreeDSecureValidateAuthentication.....	21
7 Enterprise development.....	22
7.1 Overview.....	22
7.2 Enterprise class.....	22
7.3 Enterprise Gateway Parameters.....	22
7.4 Parameter description.....	24
7.5 Parameters per action.....	34
7.5.1 Input Parameters per action.....	34
7.5.2 Output Parameters per action.....	36
8 Transaction sequences and terminology.....	39
8.1 Overview.....	39
8.2 Unique Identifiers.....	39
8.3 TransactionIndex and Follow up transactions.....	39
8.4 Reversal transaction (Negative transaction).....	40
8.5 Sequences.....	41
9 Ensuring end to end transaction integrity.....	44
9.1 Overview.....	44
9.2 Individual transactions.....	44
9.2.1 Void.....	44
9.2.2 Retry.....	45
9.2.2.1 Retry with client recorded Merchant Trace.....	45
9.2.2.2 Retry without Merchant Trace.....	46
9.2.2.3 Retry with irrelevant Merchant Trace (or irrelevant Merchant Reference).....	46
9.2.1 Enquiry.....	46
9.2.2 Conclusion.....	46
9.3 Duplicate transactions.....	46
9.3.1 Specify a unique Merchant Trace for each step in a Transaction Sequence.....	47
9.3.2 Merchant Reference validity period.....	47
9.3.3 Recurring transaction checking.....	47
10 Common parameters.....	48
10.1 Result Codes.....	48
10.2 Mode: Test vs Live.....	49
10.3 Track2.....	49
10.3.1 Track2 Service Code values.....	49
10.4 Budget Period.....	50
11 Specialised techniques.....	51
11.1 Ping.....	51
11.2 Mod-10 checking.....	52
11.3 Card Number Checking.....	53
12 ThreeDSecure (also known as “VerifiedByVISA” or “MasterCard SecureCode”).....	54

12.1 Centinel MAPS.....	54
12.1.1 Test Environment.....	54
12.1.2 Live Environment.....	54
12.2 The 3D Secure Transaction Process Flow.....	54
12.3 Using 3D Secure with iVeri.....	56
12.4 ThreeD Secure Process Flow Diagram.....	58
12.5 3D Secure.....	59
12.5.1 Checking Enrollment.....	59
12.5.2 Validate Authentication.....	60
13 NPay.....	61
13.1 NPay packages.....	61
13.1.1 NPay Development package.....	61
13.1.2 NPay Production package.....	61
13.2 NPay Architecture.....	61
13.2.1 nPay Process Flow.....	62
13.3 Integration with the iVeri Gateway.....	65
13.3.1 Get Track2 Session Key.....	65
13.3.2 Perform a Debit or Balance Enquiry.....	65
13.3.2.1 Debit.....	65
13.3.2.2 Balance Enquiry.....	65
14 payD.....	66
14.1 payD Process.....	66
14.2 Non Registered Users.....	66
14.3 Registered Users.....	66
14.4 Voiding payD transactions.....	66
15 Web Service.....	67
15.0.1 Message Formats used by the Web Service.....	67
15.0.1.1 Example of SOAP Request.....	67
15.0.1.2 Example of SOAP Response.....	67
15.0.2 Web Service URL's.....	67
15.1 Web Service implementation.....	68
15.2 WebService Methods.....	68
15.2.1 string Execute(bool validateRequest, string protocol, string protocolVersion, string request).....	68
15.2.1.1 SOAP Request.....	69
15.2.1.2 SOAP Response.....	70
15.2.2 PinBlock GetPinBlock (string mode, string pan, string pin).....	71
15.2.2.1 SOAP Request.....	71
15.2.2.2 SOAP Resonse.....	72
15.2.3 string GenerateCertificateID (string billingDetailsID).....	73
15.2.3.1 SOAP Request.....	73
15.2.3.2 SOAP Response.....	73
15.2.4 string SubmitCertificateRequest (string certificateID, string certificateSigningRequest)...	75
15.2.4.1 SOAP Request.....	75
15.2.4.2 SOAP Response.....	76
15.2.5 string GetCertificate(string certificateID).....	77
15.2.5.1 SOAP Request.....	77
15.2.5.2 SOAP Response.....	77
15.2.6 string RenewCertificateID (string certificateID).....	79
15.2.6.1 SOAP Request.....	79
15.2.6.2 SOAP Response.....	79
16 POS Device Integration.....	81

16.1 PINBlock encryption via Triple DES DUKPT encryption.....	81
16.1.1 Key Injection for DUKPT Mode Test.....	81
16.2 PINBlock encryption via Master/Session encryption.....	81
16.2.1 Key Injection for Master/Session Mode Test.....	82
16.2.2 Get Device PIN Key.....	82
16.3 Track2 encryption via Master/Session encryption.....	82
16.3.1 Track2 Key Injection for Master/Session Mode Test.....	83
16.4 Track2 encryption via Dukpt encryption.....	83
16.4.1 Track2 IPEK Injection for Dukpt Mode Test.....	83
16.5 PAN encryption via Dukpt encryption.....	83
16.6 “Debit with PIN” and “Balance Enquiry”.....	83
16.6.1 Debit with PIN.....	84
16.6.1 Balance Enquiry.....	84
16.6.2 Test environment for PIN cards.....	84
16.6.3 Determining if a card is PIN based.....	84
16.7 “EMV Transactions”.....	85
16.8 Coding for EMV data.....	85
17 Procurement transactions.....	86
17.1 Coding for Procurement.....	86
17.2 Advanced Fraud Screening.....	87
17.3 Tax Calculation.....	87
17.3.1 Calculation when TransactionDiscount is zero.....	88
17.3.2 Calculation when TransactionDiscount is NOT zero.....	88
18 Fleetcard transactions.....	89
18.1 Coding for Fleetcards.....	89
19 Airline addendum data.....	90
19.1 Coding for Airline addendum data.....	90
20 CyberSource Fraud Management.....	91
20.1 Device Fingerprinting.....	91
20.2 Coding for CyberSource data.....	91
21 Advanced settings.....	92
21.1 Merchant Reference validity period.....	92
21.2 Recurring transaction checking.....	92
21.3 ReconReference extraction.....	92
21.4 Transaction Type repetition within transaction sequence.....	92
22 File Transfer Development.....	94
22.1 Overview.....	94
22.2 FileTransfer class.....	94
22.3 File Transfer Data Types.....	94
22.4 File Transfer Parameters.....	95
22.5 File Transfer Commands.....	97
22.6 File Transfer Parameters per action.....	98
22.7 Automating the File Transfer process.....	99
23 Domain Knowledge.....	100
23.1 Card Present vs. Card Not Present.....	100
23.2 Online vs. Offline Transactions.....	100
23.2.1 Online Transactions.....	100
23.2.2 Offline Transactions.....	100
24 Frequently Asked Questions (common to .NET and Java).....	102
24.1 Problems connecting to the iVeri Gateway.....	102
24.2 Request timedout when network pinging the gateway.....	102
24.3 execute() or executeAsync() must be called before this method is allowed.....	103
25 Contact Information.....	104

25.1 Distributors.....	104
25.1.1 Nedbank South Africa.....	104
25.1.2 Nedbank Namibia.....	104
25.1.3 CBZ Zimbabwe.....	104
25.1.4 I&M Bank Kenya.....	104
25.2 Websites.....	105
25.2.1 Nedbank.....	105
25.2.2 CBZ Bank.....	105
25.2.3 I&M Bank.....	105
25.2.4 DNS configuration.....	105
<i>Appendix A: V_XML Message Examples</i>	106
Appendix B: ACS Redirect Example Page.....	111
Authenticating Enrolled Cards.....	111
POST Form.....	111
Authentication Frame.....	111
PARes Message.....	113
Response Messages.....	113

1 Revision History

Version	Author	Date	Description
4.0.2	Eugene Kriek	2016-02-19	When no CertificatePath is set the CertificatePath would point to the Client Home directory
4.0.1	Eugene Kriek	2015-11-03	Added a method to both the Enterprise and the FileTransfer classes to retrieve the Client Home directory
4.0.0	Eugene Kriek	2015-06-22	<p>Removed Requesting Certificates. Removed Checking for Available Certificates Java Support JDK 7 upwards DotNet Support 4.5.x upwards Added Capture Default CertificateId Added ClientAuthentication Added properties to override the default values</p> <ul style="list-style-type: none"> • CertificatePath • CertificateFile • CertificatePassword

2 Introduction

The iVeri range of payment products, developed by iVeri Payment Technologies (Pty) Ltd (www.iveri.com) provide proven credit card payment solutions for businesses on and off the internet.

The iVeri Client software facilitates the secure communications between the Merchant and the iVeri Gateway via client side software, primarily enabling iVeri Merchants to initiate transactions on cards from within their systems, be they websites, call centres, etc

iVeri Client comes in the following forms:

- Client .NET – Built on Microsoft's .NET Framework is compatible with all .NET compatible programming languages. Currently dependant on Microsoft operating systems.
- Client Java – Built on Oracle JDK / JRE 7. Platform independant.

iVeri Client has two classes within it that provide two different abilities:

- Enterprise: For the sending and receiving messages to / from the iVeri Gateway.
- FileTransfer: For uploading/downloading of files to / from the iVeri Gateway.

This document is written to show:

- The usage of iVeri Client Configuration utility for setting up certificate requirements for communicating with the iVeri Gateway
- the usage of the Enterprise class
- the usage of the FileTransfer class

Whenever the iVeri Gateway is enhanced with changes that effects the Gateway Protocol, this document should be updated and the latest version posted on the iVeri website.

See the "Protocol History" section for information on changes within the Gateway protocol, including its enhancements and depreciated (backward compatible) functionality.

The full functionality of the current protocol requires iVeriClient version 4.0.0 or later.

Merchants that are using an earlier version can either upgrade or refer the earlier version documentation.

Database storage, formatting and integration of iVeriClient into a merchant system is specified by developer(s) appointed by the merchant.

2.1 iVeri Enterprise

Enterprise is aimed at Merchants that have the following characteristics:

- Access to development resources.
- Medium to large websites.
- Call centres and physical store environments.
- Own Merchant Number and a relationship with an authorised Acquiring Institution

2.2 File Transfer

The File Transfer component of iVeri Client enables merchants to upload/download files to/from the iVeri Gateway. There may be files available for upload/download within iVeri Client that are not available via the iVeri BackOffice. Similarly, there may be files available for upload/download within iVeri BackOffice that are not available via the iVeri Client. The following other sections within this document relevant to File Transfer are: 3, 10.1 and 22.

2.3 Security

iVeri Client utilizes the worldwide-accepted SSL standard for encryption in order to protect the integrity of transaction information. All communication between the iVeri Client software and the iVeri Gateway is encrypted with 256 bit SSL. Client certificates are issued by the iVeri CA. Whereas the Gateway certificate is a public certificate.

Note for iVeri Enterprise Users: It is the merchants responsibility to secure the link between the cardholder and the merchant server on which the iVeri Client software operates.

2.4 Merchant Requirements

Merchants are required to enter into a "Agreement" with an authorised Acquiring Institution. See your iVeri Distributor (see section [25](#)) for more information on entering into a Merchant Agreement, and obtaining a Merchant Number.

2.5 Terminology

This document uses certain iVeri specific terminology (like `ApplicationID`). See sections 7.4 and 22.4 for descriptions of many of these terms.

2.6 iVeri Client Requirements

Merchants using iVeri Client require:

- User Group
- Test ApplicationID and Live ApplicationID

Please contact your [iVeri Distributor](#) (see section 25) for more information on how to obtain this information.

The following are the technical requirements for installing, configuring and using iVeri Client:

- **For iVeri Client.NET:** The Microsoft .NET Framework 4.5 SDK / runtime must be installed before using iVeri Client.NET 4.5.x (previous iVeri Client .NET supported the Microsoft .NET Framework 3.5 or later).
- **For iVeri Client Java:** Java 7 JDK / JRE (previous iVeri Client supported JDK 6 or later) must be installed before using the iVeri Client.Java software
- **Network Connection** from the Server to the iVeri Gateway. This may be via any network over which TCP/IP can be run as the protocol e.g. The Internet, Diginet, Leased Line, ISDN etc.

3 iVeri Client Configuration

3.1 Make contact with an iVeri Distributor and obtain a User Group (Billing Details ID)

Contact your iVeri Distributor (see section [25](#)) and request to be captured on the system. Discuss with the operator which iVeri Product you are suited for:

- **iVeri Enterprise:** You need to use the iVeri Client software.
- **iVeri Batch:** You may upload, and download batches either via the iVeri BackOffice website, or via the software within iVeri Client.
- **Certain other products:** You may download files (for example Reconciliation file) via the iVeri BackOffice website, or via the software within iVeri Client.

You will be then be asked to supply certain information. Once the iVeri Distributor has processed this information, you will be emailed a User Group and Password. Keep these details readily available, as this will be required for requesting Certificate IDs, and for logging in to the iVeri BackOffice website. This process will result in an ApplicationID being emailed to the designated technical contact.

3.2 Install the relevant runtime:

- **iVeri Client. Java:** Install Java 7 JDK / JRE or later. This can be downloaded free of charge from <http://www.oracle.com/technetwork/java/index.html>
- **iVeri Client. NET:** Install the Microsoft .NET Framework 4.5. This can be downloaded free of charge from <http://www.microsoft.com/net>

3.3 Default Location (iVeriClient_Home)

Select option 5 (Show Config file location).

The Config file will reside in one of the following places

1. Environment Variable configure: **iVeriClient_Home**
2. System Property configured **iVeriClient_Home**
3. Application / User Working directory

You may obtain the working directory by running Option 6

The configuration file location is:

[C:\](#) [Some path]\iVeriClient.config

Note: When changing the **iVeriClient_Home** environment variable and running a website in IIS, you have to reset IIS after setting the environment variable

3.4 Run the iVeri Client Configuration

- **iVeri Client. Java:**
 1. If you drop the iVeriClient.jar into the ~\jre\lib\ext directory you can just type java ClientConfig.
 2. If you want to run the **Configuration Utility** from your applications working directory drop iVeriClient.jar in your application directory. Now type java -jar iVeriClient.jar

The following should appear:

```
iVeri Client Configuration Utility v4.0.0
Copyright (c) iVeri Payment Technologies 2003-2015
Java Version 1.8.0_45
```

```
Initialising...
```

Menu

1. Capture Default CertificateID
2. Set Client Authentication
3. ApplicationID Ping iVeri Gateway
4. Perform 1c Transaction
5. Show iVeri Client Home Location
- A. Get Track2 Session Key
- X. Exit

Choice [1] :

- **iVeri Client. NET:** Run the **Configuration Utility** from your applications working directory by running the “iVeri.ClientConfig.exe” application.

The following should appear:

```
iVeri Client Configuration Utility v 4.0.0
Copyright (c) iVeri Payment Technologies 2003-2015
Dotnet Version 4.0.30319.34209
```

Initialising...

Menu

1. Capture Default CertificateID
2. Set Client Authentication
3. ApplicationID Ping iVeri Gateway
4. Perform 1c Transaction
5. Show iveri Client Home Location
- A. Get Track2 Session Key
- X. Exit

Choice [1] :

NOTE: The Client Configuration utility uses the convention that a default entry is indicated by square braces (eg [...]) before the required input. Therefore above default choice is 1.

3.5 Ensure valid internet connection (with sufficient bandwidth) to the iVeri Gateway

Select option 1, and follow the instructions.

Note: This functionality used to perform an “iVeri Gateways network pings” (ie icmp packets). However due to the fact that many iVeri Gateways have icmp disabled, testing connectivity to the iVeri Gateway is done in other ways (from version 2.3.1)

3.6 Configure Proxy Server (if necessary)

Open the configuration file in your favorite text editor and add the following XML sub elements to the `DEFAULTS` section of the configuration file.

- `PROXY_HOST`
This element identifies the proxy server by name or IP address.
- `PROXY_PORT`
Specifies the port on the proxy server to connect to.
- `PROXY_USERNAME`
The user name to use when proxy authentication is required.
- `PROXY_PASSWORD`
The proxy server's password when proxy authentication is required.

Example:

```
<XML>
  <GATEWAYS>
    <DEFAULTS>
      <VERSION>2.0</VERSION>
      <TIMEOUT>60</TIMEOUT>
      <ERRORFILE>iVeriClientErrors.txt</ERRORFILE>
      <DEFAULTGATEWAY />
      <PROXY_HOST>10.0.0.2</PROXY_HOST>
      <PROXY_PORT>3128</PROXY_PORT>
      <PROXY_USERNAME></PROXY_USERNAME>
      <PROXY_PASSWORD></PROXY_PASSWORD>
    </DEFAULTS>
    <GATEWAY Name="nedbank">
      <ADDRESS>portal.nedsecure.co.za</ADDRESS>
      <PATH>/iVeriGateway/Gateway.aspx</PATH>
    </GATEWAY>
    <GATEWAY Name="host">
      <ADDRESS>portal.host.iveri.com</ADDRESS>
      <PATH>/iVeriGateway/Gateway.aspx</PATH>
    </GATEWAY>
  </GATEWAYS>
</XML>
```

In the above example iVeriClient would connect to proxy server 10.0.0.2 listening on port 3128 without a user name and password.

3.7 Capture Default CertificateID

Select option 1. This will allow you to set the default CertificateID to be used by iVeriClient. Copy the CertificateID without the '{, }' braces and paste it

Verify the information entered.

3.8 Set ClientAuthentication

Select option 2. This will allow you to configure whether you are going to use Client Certificates or not.

Enable Client Authentication

- 1. Yes
- 2. No
- X. Exit

By selecting option 2, **No**, you will disable Client Authentication and no client certificates will be used when connecting to the gateway.

By selecting option 1, **Yes**, you will have to enable Client Authentication by completing the steps following.

3.8.1 CertificateSource (Java)

When you want to use Client Certificates there are two **CertificateSources** whereby the certificate can be stored by.

1. File - Certificate is stored as a **PKCS#12** file with a .p12 extension.
2. Keystore - Certificate is stored in a **Java Keystore (JKS)** keystore.

Certificate Source

1. File
2. KeyStore
X. Exit

Selecting option 1 will perform the following functionality:

- Set Certificate Path. Validate the path exists.
- Set Certificate File Name. Validate that the certificate exists and that it is a **PKC#12** certificate.
- Set Certificate Password. Validate the certificate password.
- Verify information and confirm.
- Enable Client Authentication

Selecting option 2 will perform the following functionality:

- Set Certificate Path. Validate the path exists.
- Set Keystore File Name. Validate that the keystore exists and that it is a **JKS** keystore
- Set Keystore Password. Validate the keystore password.
- Set Certificate Password. Validate the certificate password.
- Verify information and confirm.
- Enable Client Authentication

3.8.2 CertificateSource (Dotnet)

When you want to use Client Certificates there are two **CertificateSources** whereby the certificate can be stored by.

1. File - Certificate is stored as a **PKCS#12** file with a .p12 extension.
2. Certificate Store - Certificate is stored in a the **Local Machine Store**.

Certificate Source

1. File
2. Certificate Store
X. Exit

Selecting option 1 will perform the following functionality:

- Set Certificate Path. Validate the path exists.
- Set Certificate File Name. Validate that the certificate exists and that it is a **PKC#12** certificate.
- Set Certificate Password. Validate the certificate password.
- Verify information and confirm.
- Enable Client Authentication

Selecting option 2 will perform the following functionality:

- Validate that certificate with **DefaultCertificateId** as CommonName exists inside Local Machine Store.
- Enable Client Authentication

3.9 Perform an ApplicationID Ping (Enterprise specific)

Select option 3. This will perform the following functionalities:

- Validate your Certificate
- Check that your ApplicationID is ready for use
- Check your ApplicationID-CertificateID combination is valid
- Check that the iVeri Gateway is up
- Check that the link between the iVeri Gateway and the acquirer(s) is up

3.10 Develop and Test the Application

Develop and Test your application referencing the sections below and samples as appropriate. Your development should include the logging of transaction information in database (preferable) or file for future references and queries. Should there be a dispute over what data was sent between the Merchant and the iVeri Gateway, the Merchant would be required to produce these logs.

In order to ensure that iVeri Enterprise merchants keep client side logs, we require that a sample of these logs (in any format displayable in a text editor) be emailed when development is completed. See "Contact information" (section 25) for an email address to submit logs to.

Logs should include any actions that may be performed by the merchant.

The `getLoggableRequest()` and `getLoggableResponse()` methods can be used for logging the input and output from the transaction. These two methods retrieve the xml sent to or received from the gateway and contain all the necessary transaction information to be logged. The `getLoggableRequest()` method is called the line before execution, while the `getLoggableResponse()` method is called the line after execution.

These methods allow logging required sensitive data (e.g. Credit Card Numbers) in a secure manner.

3.11 Perform 1 cent Transaction (Enterprise specific)

The "1c Live Transaction" functionality is available via option 5 in the iVeri ClientConfig. This helps the merchant check that iVeri Gateway and the acquirer are correctly configured for the Merchant to receive a 1c deposit into their account. This functionality is also known as a 1c "Trial" transaction because it allows a "Live" transaction, which is otherwise not allowed until the Merchants Live ApplicationID Status is "Active" (see 3.12 for more information).

3.12 Switch to "Active"

Once you:

- have emailed the required logs as per Step 3.10 AND
- (Enterprise specific) have performed a live (trial) transaction and seen the deposit into your bank account as per Step Error: Reference source not found

Contact the iVeri Distributor help desk and request that your ApplicationID Status be made "Active". Only once your logs have been checked by the iVeri Distributor will your ApplicationID will switched to "Active".

3.13 Move to Production machine

If your development machine and production machines are two different machines, you will need to install your developed code to the production machine. To do this, you need to repeat Steps 3.2 through to Error: Reference source not found for FileTransfer and Steps 3.2 through to 3.9 for iVeri Enterprise to ensure everything is correctly configured on the Production machine.

4 Important Security Overview

4.1 Security in iVeri Client .NET

When iVeriClient is run with **ClientAuthentication** enabled and using the Local Machine Store as the **CertificateSource** make sure that the user that the application is run under has permissions to the specific certificate in the store. Please make sure that the user that the application is run under has read, write permissions to the [iVeriClient_Home](#) directory.

4.2 Security in iVeri Client .Java

IveriClient is required to be able to retrieve client certificates from disk in an OS and Application Framework independent way.

Please ensure that your application has read permissions to the directory where your keystore / certificate is located. Please make sure that your iVeriClient has read, write permissions to the [iVeriClient_Home](#) directory.

5 Payment mechanisms : PAN vs Track2 vs PIN

iVeri Gateway separates accounts into 3 different payment mechanisms:

- PAN
- Track2
- PIN

In order to process a transaction with an account you need to know which payment mechanism you are going to use.

Note that independent of the payment mechanism is used, the iVeri Gateway returns a dotted out "PAN" which can be used for display purposes.

5.1 PAN (also known as "Card Not Present" or "Keyed")

The Primary Account Number (PAN) is given by the card holder to the merchant.

Either the card is not present with the Merchant, or the Merchant does not have a card reader to "swipe" the card.

The account can be any card that has the PAN embossed or printed on the card but does not require a PIN.

Mandatory Input Parameters: PAN, ExpiryDate.

Optional Input Parameters: StartDate, CardSecurityCode.

5.2 Track2 (also known as "Card Present" or "Swiped")

A card reader is available to "swipe" the cardholders card, and read the Track2 from it.

The account can be any card that has a Track2 on the magnetic strip, but does not require a PIN.

Mandatory Input Parameter: Track2

5.3 PIN (also known as "Card Present" with PIN or "Swiped" with PIN)

The account is a card that requires a PIN (eg debit card) and a card reader is available to "swipe" the cardholders card, and read the Track2 from it.

A PED (PIN entry device) is available to securely capture the cardholders PIN and encrypt it.

See "PIN based transactions" (section 16) for more details.

6 Commands and Actions

6.1 Commands

6.1.1 Transaction commands (may cause transfer of funds)

- **Authorisation (also known as “Preauthorisation”)**

Causes a reservation of funds on the cardholders account.

- **Authorisation Reversal**

Unreserve the funds previously reserved on the cardholders account.

- **Debit (also known as “Sale” or “Purchase”)**

Causes a transfer of funds from the cardholder to the merchant.

- **Credit (also known as “Refund” or “Pay”)**

Causes a transfer of funds from the merchant to the cardholder.

- **Void**

Cancel one of the above commands within a short time after the command was initiated.

6.1.2 Enquiry commands (do not cause transfer of funds)

See section on “Actions” below for a description of these commands

- Balance Enquiry
- PAN Enquiry
- PANToken Enquiry
- Ping
- GetDevicePINKey
- ThreeDSecureCheckEnrollment
- ThreeDSecureValidateAuthentication

6.1.3 FileTransfer commands

See section 22 on “FileTransfer” for a description of these commands

- Batch Upload
- Batch Download
- HotCard Download
- BINLookup Download
- BINManagement Download
- BlackCard Download
- TransactionHistory Download
- Recon Download
- AcquirerRecon Download
- Inventory Download

6.2 Actions

An Action corresponds to the combination of a Payment Mechanism and a Command. The concept of an Action is used within the documentation and examples as a means of describing functionality. Note that the iVeriClient software and the iVeriGateway use the concepts Payment Mechanism and Command instead of Action.

The iVeri Gateway allows for the following actions:

- Authorisation with PAN
- Authorisation with Track2
- Authorisation with VisaCheckoutCallID
- Additional Authorisation with TransactionIndex
- Authorisation Reversal with TransactionIndex
- Debit with PAN
- Debit with Track2
- Debit with PIN
- Debit with TransactionIndex
- Debit with VisaCheckoutCallID
- Credit with PAN
- Credit with Track2
- Credit with TransactionIndex
- Credit with VisaCheckoutCallID
- Void
- Balance Enquiry with PIN
- PAN Enquiry with PAN
- PAN Enquiry with Track2
- PANToken Enquiry
- Ping
- Get Device PIN Key
- ThreeDSecureCheckEnrollment
- ThreeDSecureValidateAuthentication

6.2.1 Authorisation with PAN

Reserve funds when a card not present.

6.2.2 Authorisation with Track2

Reserve funds when a card is present. Funds reservation is not applicable for cards requiring a PIN.

6.2.3 Authorisation with VisaCheckoutCallID

Reserve funds when a card is present. Funds reservation is not applicable for cards requiring a PIN.

6.2.4 Additional Authorisation with TransactionIndex

Increase the amount previously reserved via iVeri by an additional amount. The addition of funds previously reserved outside the iVeri Gateway is not supported.

6.2.5 Authorisation Reversal with TransactionIndex

Release the funds previously reserved on the cardholders account via the iVeri Gateway. The release of funds reserved outside the iVeri Gateway is not supported.

6.2.6 Debit with PAN

Transfer of funds from cardholder to merchant when a card not present. The use of an AuthorisationCode previously obtained outside the iVeri Gateway is supported.

6.2.7 Debit with Track2

Transfer of funds from cardholder to merchant when a card is present. The use of an AuthorisationCode previously obtained outside the iVeri Gateway is supported.

6.2.8 Debit with PIN

Transfer of funds from cardholder to merchant when a card requiring a PIN is present.

6.2.9 Debit with TransactionIndex

Transfer of funds from cardholder to merchant. Follow up of an action previously sent to the iVeri Gateway. Not supported for cards requiring a PIN.

6.2.10 Debit with VisaCheckoutCallID

Transfer of funds from cardholder to merchant. Follow up of an action previously sent to the iVeri Gateway. Not supported for cards requiring a PIN.

6.2.11 Credit with PAN

Transfer of funds from merchant to cardholder when a card not present.

6.2.12 Credit with Track2

Transfer of funds from merchant to cardholder when a card is present. Credit is not currently supported for cards requiring a PIN.

6.2.13 Credit with TransactionIndex

Transfer of funds from merchant to cardholder. Follow up of an action previously sent to the iVeri Gateway. Not supported for cards requiring a PIN.

6.2.14 Credit with VisaCheckoutCallID

Transfer of funds from merchant to cardholder. Follow up of an action previously sent to the iVeri Gateway. Not supported for cards requiring a PIN.

6.2.15 Void

Cancel a transaction command within a short time after the command was initiated.

6.2.16 Balance Enquiry with PIN

Obtain the balance of the PIN based account in the currency of the account. Note that this currency may be different to the currency of the merchant.

6.2.17 PAN Enquiry with PAN

Obtain information about a card (which is not present) without performing a transaction, for example to check if the card is a hot card, or blacklisted by the merchant.

6.2.18 PAN Enquiry with Track2

Obtain information about a card (which is present) without performing a transaction, for example to check if the card is a hot card, or blacklisted by the merchant. PAN Enquiry is not currently

supported for cards requiring a PIN.

6.2.19 PANToken Enquiry

Obtain a TransactionIndex for the card number and expiry date without performing a transaction. The TransactionIndex is to be used in a Tokenized transaction.

6.2.20 Ping

The Ping command is primarily used to determine if the connection status between the Merchant and the Acquirer. If the connection is down, then the Ping command can also be used to check when the status is back up.

See section 11.1 for more information.

6.2.21 Get Device PIN Key

Get the current Triple DES session key for a device. See PIN based transactions (section 16).

6.2.22 ThreeDSecureCheckEnrollment

Check the CardHolders 3DSecure Enrollment. (section 12.5).

6.2.23 ThreeDSecureValidateAuthentication

Validate the CardHolders 3DSecure Authentication process. (section 12.5).

7 Enterprise development

7.1 Overview

The following is available to assist Enterprise development:

- iVeri Client Developers Guide (this document)
- The iVeri Client API (which includes the Enterprise class)
- iVeri Enterprise code samples
- Distributor website (see section 25) for updates to the documentation mentioned

7.2 Enterprise class

The Enterprise class has the following basic flow:

- Instantiate Enterprise class
- Set the Gateway and the CertificateID.
These can be set within the Enterprise constructor, or as separate methods (properties in iVeriClient.net)
- You may choose to override the (properties in iVeriClient.java) CertificatePath, KeyStoreFile, CertificateFile, KeyStorePassword and CertificatePassword
- You may choose to override the (properties in iVeriClient.net) CertificatePath, CertificateFile, and CertificatePassword
- Prepare the command to be sent to the gateway
Call one of the following methods: prepare / authorisation / authorisationReversal / debit / credit / ping
- set the various input parameters: generally via setTag, or setAttribute
- submit the request to the iVeri Gateway: via execute or executeAsync
- check the results returned, particularly ResultStatus and ResultCode
- check other output parameters: generally via getTag, or getAttribute

7.3 Enterprise Gateway Parameters

Below we describe the various gateway parameters.

This is followed by a table of what input and output parameters are relevant for the each action.

Parameters are shown grouped according to their usage.

The following key is used for data types:

Data Type	Description
A	Alpha only (A-Za-z)
AN	Alphanumeric (a-zA-Z0-9)
Base64	Base64 encryption of binary data
Boolean	True or False
Guid	Globally Unique Identifier: {[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}}
Hex	Hex (0-9A-Fa-f)
N	Numeric (Positive integer)
N.	Digits and dots (.) (e.g's 4242.....4242, 1.0)
N:	Digits and colons
String	ANPS Free format string containing: Alpha, numeric, special and padding (printable ASCII)

Data Type	Description
Z	Positive or negative integer

The Node Type column corresponds to how the Enterprise class should be used for the parameter:

Node Type	Set input parameter value	Get output parameter value
attribute	enterprise.setAttribute(..)	enterprise.getAttribute(..)
tag	enterprise.setTag(..)	enterprise.getTag(..)
parameter	enterprise.prepare(...)	N/A
attribute parameter	enterprise.prepare(...)	enterprise.getAttribute(..)
property	Use method or property of enterprise corresponding to the parameter	N/A
subtag	call enterprise.setTag(..) within enterprise.openElement(...) ... enterprise.closeElement()	N/A

7.4 Parameter description

Core Parameters					
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description
ApplicationID	attribute parameter	Guid	38	38	Identification of the merchants configuration within the iVeri Gateway
CertificateID	property	Guid	38	38	The iVeri CertificateID installed on the server communicating with the iVeri Gateway
CertificatePath	property	A			Valid Path to directory where the KeyStoreFile or CertificateFile exists on the filesystem
CertificateFile	property	A			Existing certificate file
CertificatePassword	property	A			Password of the certificate
KeyStoreFile	property	A			Existing keystore file (Only iVeriClient.java)
KeyStorePassword	property	A			Keystore password (Only iVeriClient.java)
Gateway	property	A		10	The name of the gateway connecting to. If not explicitly set, the default gateway is used.
Command	attribute parameter	A		50	The command specifying what should be done by the iVeri Gateway
					GetDevicePINKey
					Get DevicePINKey (use Category='Security')
					Ping
					Ping (use Category='System')
					Authorisation
					Authorisation (use Category='Transaction')
					AuthorisationReversal
					Authorisation Reversal (use Category='Transaction')
					Credit
					Credit (use Category='Transaction')
					Debit
					Debit (use Category='Transaction')
					Void
					Void (use Category='Transaction')
					Balance
					Balance Enquiry (use Category='Enquiry')
					PAN
					PAN Enquiry (use Category='Enquiry')
					PANToken
					PANToken Enquiry (use Category='Enquiry')
					ThreeDSecureCheckEnrollment
					ThreeDSecureCheckEnrollment Enquiry (use Category='Enquiry')
					ThreeDSecureValidateAuthentication
					ThreeDSecureValidateAuthentication Enquiry (use Category='Enquiry')
Mode	attribute parameter	A	4	4	The mode of the corresponding ApplicationID. See section 10.2
					Test
					Live
RequestID	attribute	Guid	38	38	A unique identifier generated by the iVeri Gateway for this request
Category	parameter	A		50	A categorisation of the request. Only required in conjunction with a prepare method.
					Transaction
					use if Command = 'Debit' or 'Credit' or 'Authorisation' or 'AuthorisationReversal' or 'Void'
					Enquiry
					use if Command = 'Balance', 'PAN' or 'PANToken'
					System
					use if Command = 'Ping'

					Security	use if Command = 'GetDevicePINKey'
Common Parameters						
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description	
Acquirer	tag	A	3	32	The Acquiring system to which this transaction was routed by iVeri	
					Nedcor	
					NedbankBICISO	
					IMNarada	
					DashenACI	
					ChamsACI	
					MSCCTranzWare	
					CTLNigeria	
AcquirerDate	tag	N	8	8	The date that the Acquirer allocated to this transaction	
					YYYYMMDD	YearMonthDay
AcquirerReference	tag	N:		64	A reference allocated by the Acquirer to this transaction. It is a composite element whose format is dependent on the particular acquirer to which this transaction was routed.	
					CCCC:TTTTTTTT	Cycle:Trace (for provider: Nedcor)
					yyyyMMdd:tttttt	SettlementEndDate:ProviderTrace (for providers: TMS)
					yyyyMMdd:ydddhhTTTT	SettlementEndDate:RetrievalReferenceNumber (for providers: CSC, CardTech)
AcquirerTime	tag	N		6	The time that the Acquirer allocated to this transaction	
					HHMMSS	HourMinuteSecond
Amount	tag	N	0	12	The total value of the transaction in the smallest unit of the currency specified (eg in cents)	
AuthorisationCode	tag	AN	0	6	The Authorisation Code issued by the Issuer to the Merchant either telephonically or electronically	
BudgetPeriod	tag	N	0	2	The number of months over which the cardholder would like to pay the transaction off. See section 10.4	
					0	default
					3	
					6	
					9	
					12	
					18	
					24	
					36	
CardSecurityCode	tag	N	3	4	The 3 or 4 digits printed on the card which are not contained on the magnetic strip. Usually printed after the CCNumber on the signature strip. Corresponds to American Express CIV, MasterCard CVC2 and VISA CVV2. Does not exist within Associations	
Currency	tag	A		3	The ISO 4217 currency code of the value of the transaction. e.g. USD or ZAR or GBP	
DisplayAmount	tag	String		30	The Amount returned in a currency aware printable format	
ExpiryDate	tag	N	4	6	The last month of the validity period of the card	
					MMYY	MonthYear
					MMYYYY	MonthYear
CardholderName	tag	String		50	The name of the cardholder as embossed on the card.	

MerchantName	tag	String		50	The merchant name associated with the ApplicationID																																										
MerchantAddress	tag	String		50	The merchant address associated with the ApplicationID																																										
MerchantCity	tag	String		50	The merchant city associated with the ApplicationID																																										
MerchantCountryCode	tag	String		2	The merchant country code associated with the ApplicationID																																										
MerchantCountry	tag	String		50	The merchant country name associated with the ApplicationID																																										
MerchantReference	tag	String	0	64	A merchant generated identifier that is unique within a specified time period that identifies a transaction sequence																																										
OriginalMerchantReference	tag	String	0	64	A merchant generated identifier of a previous transaction used in a Tokenized PAN transaction																																										
MerchantTrace	tag	String	0	64	Unique merchant identification for the request																																										
MerchantUSN	tag	String	0	15	The merchant USN associated with the ApplicationID																																										
OriginalMerchantTrace	tag	String		64	A reference to the original MerchantTrace previously sent to the iVeri Gateway																																										
OriginalRequestID	tag	Guid	38	38	The RequestID that was returned as part of the original transaction																																										
PAN	tag	N.		20	Primary Account Number (eg Credit card number), may have been extracted from Track2. When this is an output parameter, then a section of it is dotted out, and safe to display or print																																										
PANMode	tag	String	0	128	The mechanism by which the PAN was determined from the card. The value is a comma separated list of the following:																																										
					<table border="1"> <thead> <tr> <th>Name</th> <th>Direction</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Keyed</td> <td>Input/Output</td> <td>Card number was keyed</td> </tr> <tr> <td>Tokenized</td> <td>Output</td> <td>PANFormat 'TransactionIndex', 'OriginalMerchantReference' or 'MSISDN' (non-registration) was specified in the request</td> </tr> <tr> <td>Swiped</td> <td>Input/Output</td> <td>Card was swiped. This means that a Track2 must be specified in the request.</td> </tr> <tr> <td>Dipped</td> <td>Input/Output</td> <td>Card was processed while in the card reader slot of the device. This means that a Track2 must be specified in the request.</td> </tr> <tr> <td>Tapped</td> <td>Input/Output</td> <td>Card was processed by a contactless device. This means that a24 Track2 must be specified in the request.</td> </tr> <tr> <td>ConstructedTrack2</td> <td>Output</td> <td>For when a partial Track2 is received in the request usually accompanied by a PINBlock as one of the legs of a PANFormat "MSISDN" transaction.</td> </tr> <tr> <td>CVV</td> <td>Output</td> <td>The CardSecurityCode tag had a value in the request</td> </tr> <tr> <td>PIN</td> <td>Output</td> <td>The PINBlock tag had a value in the request</td> </tr> <tr> <td>PINCapable</td> <td>Input/Output</td> <td>No online PIN or EMV data sent to the gateway, but the device is capable to process PIN.</td> </tr> <tr> <td>EMV</td> <td>Output</td> <td>EMV data were specified in the request18</td> </tr> <tr> <td>EMVFallback</td> <td>Output</td> <td>No EMV data specified in the request, but the service code of the request Track2 indicate a chip card.</td> </tr> <tr> <td>EncryptedPAN</td> <td>Output</td> <td>The value of the PAN tag in the request specified encrypted data</td> </tr> <tr> <td>EncryptedTrack2</td> <td>Output</td> <td>The value of the Track2 tag in the request specified encrypted data</td> </tr> </tbody> </table>	Name	Direction	Description	Keyed	Input/Output	Card number was keyed	Tokenized	Output	PANFormat 'TransactionIndex', 'OriginalMerchantReference' or 'MSISDN' (non-registration) was specified in the request	Swiped	Input/Output	Card was swiped. This means that a Track2 must be specified in the request.	Dipped	Input/Output	Card was processed while in the card reader slot of the device. This means that a Track2 must be specified in the request.	Tapped	Input/Output	Card was processed by a contactless device. This means that a24 Track2 must be specified in the request.	ConstructedTrack2	Output	For when a partial Track2 is received in the request usually accompanied by a PINBlock as one of the legs of a PANFormat "MSISDN" transaction.	CVV	Output	The CardSecurityCode tag had a value in the request	PIN	Output	The PINBlock tag had a value in the request	PINCapable	Input/Output	No online PIN or EMV data sent to the gateway, but the device is capable to process PIN.	EMV	Output	EMV data were specified in the request18	EMVFallback	Output	No EMV data specified in the request, but the service code of the request Track2 indicate a chip card.	EncryptedPAN	Output	The value of the PAN tag in the request specified encrypted data	EncryptedTrack2	Output	The value of the Track2 tag in the request specified encrypted data
Name	Direction	Description																																													
Keyed	Input/Output	Card number was keyed																																													
Tokenized	Output	PANFormat 'TransactionIndex', 'OriginalMerchantReference' or 'MSISDN' (non-registration) was specified in the request																																													
Swiped	Input/Output	Card was swiped. This means that a Track2 must be specified in the request.																																													
Dipped	Input/Output	Card was processed while in the card reader slot of the device. This means that a Track2 must be specified in the request.																																													
Tapped	Input/Output	Card was processed by a contactless device. This means that a24 Track2 must be specified in the request.																																													
ConstructedTrack2	Output	For when a partial Track2 is received in the request usually accompanied by a PINBlock as one of the legs of a PANFormat "MSISDN" transaction.																																													
CVV	Output	The CardSecurityCode tag had a value in the request																																													
PIN	Output	The PINBlock tag had a value in the request																																													
PINCapable	Input/Output	No online PIN or EMV data sent to the gateway, but the device is capable to process PIN.																																													
EMV	Output	EMV data were specified in the request18																																													
EMVFallback	Output	No EMV data specified in the request, but the service code of the request Track2 indicate a chip card.																																													
EncryptedPAN	Output	The value of the PAN tag in the request specified encrypted data																																													
EncryptedTrack2	Output	The value of the Track2 tag in the request specified encrypted data																																													
PurchaseDate	tag		4	4	The date on which the purchase was made. Defaults to current Date																																										
					MMDD MonthDay																																										
PurchaseTime	tag		4	4	The time at which the purchase was made. Defaults to current Time																																										
					HHMM HourMinute																																										

ReconReference	tag	N	0	8	Identifier that the Merchant is returned during a transaction and that also appears in the reconciliation information supplied by the Acquirer. Either assigned by the iVeri Gateway or derived from the specified MerchantReference.
StartDate	tag	A		6	The date of the start of the validity period of the card number. The start date is not available on many cards, and will remain an optional parameter until the start date becomes more common
					MMYY MonthYear
					MMYYYY MonthYear
Terminal	tag	N	0	8	Identifier optionally allocated by the merchant which is intended to group transactions together for reporting purposes in reconciliation statements.
Track2	tag	String		39	Track2 after being read by the swipe device from the magnetic stripe on the card (for card present transactions). It is inclusive of the beginning and end markers being ; and ? respectively. See section 10.3
TransactionIndex	tag	Guid	38	38	Unique identifier allocated by iVeri for a series of related transactions. If PANFormat is 'TransactionIndex', TransactionIndex is used to locate a previous transaction for the PAN to be resolved.
MobileMoneyID	tag	N	1	6	An additional transaction identifier returned for a PANFormat "MSISDN" transaction. This tag should be passed to the Gateway along with OriginalRequestID or OriginalMerchantTrace when the transaction needs to be Voided.
MSISDN	tag	String			A mobile number required by PANFormat "MSISDN" transactions
VisaCheckoutCallID	tag	String	0	48	Visa Checkout transaction ID associated with a payment request.
PANFormat	tag	String			An enumeration specifying how to obtain the PAN details
					PAN Required tag: PAN. If PANFormat is not specified, this value is used if the PAN tag is set and the Track2 tag not set.
					Track2 Required tag: Track2. If PANFormat is not specified, this value is used if the Track2 tag is set.
					TransactionIndex/ OriginalMerchantReference/ Required tags: PAN and one of TransactionIndex or OriginalMerchantReference. The PAN is the dotted out number of the card used with the original transaction. The full PAN is looked up using the given identifier for the original transaction.
					VisaCheckoutCallID Required tags: VisaCheckoutCallID The VisaCheckoutCallID is used to retrieve the Payment Data from the Visa Checkout Process.
					MSISDN Required tag: MSISDN. If in addition, AccountType, PAN (full) and ExpiryDate are provided, the transaction also serves as a registration of the MSISDN, thereby enabling future transactions with MSISDN only.
CardHolderPresence	tag	String	0	128	Specify how the cardholder is involved when a virtual transaction is performed. This tag supersedes the ElectronicCommerceIndicator tag. The value is a comma separated list of the following:
					CardPresent Gets set in the output when a full Track2 was specified in the request. This value may also be set in the request when the card number was keyed on a POS device.
					CardNotPresent The physical plastic of the card was not present when the request was sent to the gateway.
					eCommerce Gets set in the output when one of the ElectronicCommerceIndicator values was specified in the request
					ThreeDSecure Encryption (SSL) used between CardHolder and Merchant. ThreeDSecure was successful
					ThreeDSecureAttempted Encryption (SSL) used between CardHolder and Merchant. ThreeDSecure was attempted unsuccessfully
					SecureChannel Encryption (SSL) used between CardHolder and Merchant. ThreeDSecure not

					used	
					ClearChannel	No encryption used between CardHolder and Merchant
					MOTO	Telephonic or mail order
					Recurring	Transactions submitted by a merchant automatically
Card Detail Parameters						
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description	
Association	tag	String	0	50	The association to which the card number belongs e.g. VISA or Master card or American Express	
BIN	tag	N	1	9	The Bank Identification Number into which PANs are grouped. Generally this is the first two to eight digits	
CardType	tag	String	0	50	A description of the type of the card used e.g. Credit Card, Maestro, Electron etc	
Issuer	tag	String	0	50	If the card is local this gives a description of the institution that issued this particular card	
Jurisdiction	tag	String	0	50	Description of whether the card is a local or international card	
Result Parameters						
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description	
Code	attribute	N		3	The numeric Result Code of the completed execution. See section 10.1	
Description	attribute	String		1024	A description of the results of the completed execution. Only relevant where ResultStatus is 'Unsuccessful' [-1] or 'Successful with warning' [1]	
Source	attribute	String		128	The source of the result	
Previous	attribute	Boolean			In most cases returns false, indicating that the other result values refer to the immediately completed execution. For advanced usage within a Reprint, to indicate whether the result returned refers to the current execution, or a previous execution.	
Status	attribute	Z		2	The status of the completed execution	
					-1	Unsuccessful
					0	Successful
					1	Successful, with warning
3DSecure Parameters (see section 12)						
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description	
ElectronicCommerceIndicator	tag	A	0		More commonly known as the ECI, it describes for virtual transactions what steps were taken to secure and authenticate the transaction. Values are the following subset of CardHolderPresence values: ThreeDSecure, ThreeDSecureAttempted, SecureChannel and ClearChannel.	
CardHolderAuthenticationData	tag	Base64	28	28	For usage with Verified by Visa / MasterCard SecureCode. The CAVV or UCAF field depending on whether the card is VISA or Master card. Note: Original binary format length = 20 bytes. Mandatory when ElectronicCommerceIndicator = ThreeDSecure	
CardHolderAuthenticationID	tag	Base64	28	28	For usage with Verified by Visa / MasterCard SecureCode. The Transaction ID. Note: Original binary format length = 20 bytes. Mandatory when ElectronicCommerceIndicator = ThreeDSecure	
ThreeDSecure_RequestID	tag	Guid	38		The RequestID that was returned as part of the Enrollment Check that will be used to retrieve data for the Authentication Validation	
ThreeDSecure_ACS_URL	tag	URL		No Limit	URL of the Access Control Server of the card-issuing bank where you need to send the ThreeDSecure_PAREq so that the customer can be authenticated.. The field length can be very large.	
ThreeDSecure_PAREq	tag	Base64		No Limit	Digitally signed payer authentication request message that contains a unique transaction ID.The field length can be very large.	

ThreeDSecure_SignedPAREs	tag	Base64		No Limit	Digitally signed PAREs message that contains the authentication result.The field length can be very large.
POS Parameters (see section 16)					
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description
DeviceFirmware	tag	String		64	The firmware (software) currently loaded on the device. This may be set as an integer
DeviceFirmwareVersion	tag	String		64	The version of the firmware (software) currently loaded on the device.
DevicePINKey	tag	Hex	32	32	Master/Session encryption specific: The device PIN Key encrypted under the device master key i.e. DMK(DPK)
MACDevicePINKey	tag	Hex	8	8	Master/Session encryption specific: The MAC of the device PIN Key i.e. MAC(DPK)
AccountType	tag	A	0	32	The type of account that this transaction relates to (known to the cardholder)
					Savings Credit
					Cheque NotSpecified
AvailableBalance	tag	N		12	The available amount in the cardholders account given in the smallest unit of currency of the cardholder (not necessarily the currency of the merchant)
DisplayAvailableBalance	tag	String		25	The AvailableBalance returned in a currency aware printable format
CashAmount	tag	N		12	The Cash (back) portion of the Amount. Subject to the constraint: 0 <= CashAmount <= Amount
CurrentBalance	tag	Z		12	The current balance in the cardholders account given in the smallest unit of currency of the cardholder (not necessarily the currency of the merchant)
DisplayCurrentBalance	tag	String		25	The CurrentBalance returned in a currency aware printable format
DeviceMake	tag	String		64	The manufacturer of the device (terminal). e.g. Dione / Sagem
DeviceSerialNumber	tag	String		64	The serial number of the device (terminal)
DeviceCycle	tag	String		20	The current batch id of the device (terminal)
PINBlock	tag	Hex	16	16	The cardholders PIN encrypted using the current device pin key
KeySerialNumber	tag	Hex	20	20	DUKPT encryption specific: Mandatory input parameter needed in decryption of a DUKPT PINBlock.
Track2KeySerialNumber	tag	Hex	20	20	DUKPT Track2 encryption specific: Mandatory input parameter needed in decryption of a DUKPT Track2.
PANKeySerialNumber	tag	Hex	20	20	DUKPT PAN encryption specific: Mandatory input parameter needed in decryption of a DUKPT PAN.
EMV Parameters (see section 15)					
EMV_AuthorisationRequestCryptogram	tag	Hex		16	EMV tag 9F26
EMV_ApplicationIdentifier	tag	Hex		32	EMV tag 9F06
EMV_ApplicationInterchangeProfile	tag	Hex		4	EMV tag 82
EMV_CardSequenceNumber	tag	Hex		2	EMV tag 5F34
EMV_ApplicationTransactionCounter	tag	Hex		4	EMV tag 9F36
EMV_ApplicationVersion	tag	Hex		4	EMV tag 9F08
EMV_CardHolderVerificationMethodResult	tag	Hex		6	EMV tag 9F34
EMV_CryptogramInformationData	tag	Hex		2	EMV tag 9F27
EMV_IssuerApplicationData	tag	Hex		64	EMV tag 9F10
EMV_TerminalCapabilities	tag	Hex		6 or 8	EMV tag 9F33
EMV_TerminalType	tag	N		2	EMV tag 9F35
EMV_TransactionType	tag	N		2	EMV tag 9C
EMV_TerminalVerificationResult	tag	Hex		10	EMV tag 95

EMV_UnpredictableNumber	tag	Hex		8	EMV tag 9F37
EMV_TransactionStatusInformation	tag	Hex		4	EMV tag 9B
EMV_IssuerAuthenticationData	tag	Hex		16 or 32	EMV tag 91
EMV_IssuerScriptTemplate1	tag	Hex		999	EMV tag 71
EMV_IssuerScriptTemplate2	tag	Hex		999	EMV tag 72
EMV_ResponseCode	tag	AN		2	EMV tag 8A
Procurement Parameters (see section 17)					
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description
CustomerReferenceIdentifier	tag	AN		17	Description allocated by the merchant to be printed on the Cardholders statement
CustomerVATRegistrationNumber	tag	AN		13	The VAT registration number of the Customer if available
DestinationCountry	tag	A	2	2	The ISO code for the country of residence of the Merchant. e.g. ZA, US, GB
DestinationZIPCode	tag	AN	0	10	The Customers Destination ZIP code
NationalTax	tag	N	0	12	The VAT amount of the transaction in the currency's smallest denomination. Depending on the NationalTaxIndicator this may be included or in addition to the Amount field
NationalTaxIndicator	tag	N	0	1	Indicates whether VAT is applicable to this transaction.
				0	NationalTax is NOT included
				1	NationalTax is included
OrderDate	tag	A	0	6	The date that the order was placed. This is not necessarily the same as the PurchaseDate. Defaults to current Date
				YYMMDD	YearMonthDay
PurchaseIdentifier	tag	AN	0	25	Optional description allocated by the Merchant to be displayed on the Merchants statement. Defaults to MerchantReference.
ShipFromZIPCode	tag	AN	0	10	The ZIP code of the Merchant (Shipping source ZIP code)
ShippingAmount	tag	N	0	12	The amount charged to the customer for shipping charges in the currency's smallest denomination.
ShippingTaxRate	tag		0	4	The VAT rate that applies to the ShippingAmount. It is a percentage and not the actual value. The value is the rate multiplied with 10000.
TransactionDiscount	tag	N	0	12	Discount value for the transaction as a whole, it is the result of individual line item percentage discounts applied to line items. In the currencies smallest denomination
UniqueVATInvoiceReferenceNumber	tag	AN	0	15	The invoice number allocated by the merchant and appears on the merchants invoice given to the customer. This is an optional field is independent of the Merchant Reference.
Procurement LineItem Parameters (see section 17)					
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description
Discount	subtag	N	0	12	The total discount for the line item in the currency's smallest denomination.
ItemCommodityCode	subtag	N	0	4	The ISO commodity code for the line item unit.
ItemDescriptor	subtag	N	0	26	A description for the line item unit.
ProductCode	subtag	AN	0	12	The merchant's code for the line item unit.
Quantity	subtag	N	0	8	The total number of line item units.
TaxRate	subtag	N	0	4	The VAT rate applicable to the line item. The value is the rate multiplied with 10000.
Total	subtag	N	0	12	The total amount for the line item in the currency's smallest denomination. The value equals Quantity x UnitCost - Discount.

UnitCost	subtag	N	0	10	The cost of a line item unit in the currency's smallest denomination.
UnitOfMeasure	subtag	AN	0	12	The measurement unit description.
PassengerFirstName	subtag	AN	0	60	Passenger's first name.
PassengerLastName	subtag	AN	0	60	Passenger's last name
PassengerID	subtag	AN	0	32	ID of the passenger to whom the ticket was issued. For example, you can use this field for the frequent flyer number
PassengerStatus	subtag	AN	0	32	Your company's passenger classification, such as with a frequent flyer program. In this case, you might use values such as standard, gold, or platinum.
PassengerType	subtag	AN	0	32	<p>Passenger classification associated with the price of the ticket. You can use one of the following values:</p> <ul style="list-style-type: none"> • ADT: Adult • CNN: Child • INF: Infant • YTH: Youth • STU: Student • SCR: Senior Citizen • MIL: Military

Note: The fields in **Blue** will only be used when doing CyberSource Advanced Fraud Screening.

Fleet Parameters (see section 18)

Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description				
CustomerReferenceIdentifier	tag	AN		8	Contains two subfields, and is in the format: "Type" + "Odometer Reading"				
					<table border="1"> <tr> <td>Type</td> <td>Private (P) or Business (B). Max Length = 1. Data Type = A</td> </tr> <tr> <td>Odometer reading</td> <td>Odometer reading. Max Length = 7. Data Type = N</td> </tr> </table>	Type	Private (P) or Business (B). Max Length = 1. Data Type = A	Odometer reading	Odometer reading. Max Length = 7. Data Type = N
Type	Private (P) or Business (B). Max Length = 1. Data Type = A								
Odometer reading	Odometer reading. Max Length = 7. Data Type = N								

Fleet Lineltem Parameters (see section 18)

Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description																								
ProductCode	subtag	AN	0	64	The iVeri Product Code for the Fleet Lineltem. The latest set of iVeri Fleet Product Codes can be obtained via the Inventory download command (Only use the ProductCodes with Type='Fleet'). The set at time of writing is provided here.																								
					<table border="1"> <tr> <td>FLEET OIL</td> <td>Oil</td> </tr> <tr> <td>FLEET SERVICE</td> <td>Service</td> </tr> <tr> <td>FLEET TYRES</td> <td>Tyres</td> </tr> <tr> <td>FLEET ACCIDENTS</td> <td>Accidents</td> </tr> <tr> <td>FLEET ACCESSORIES</td> <td>Accessories</td> </tr> <tr> <td>FLEET BRAKES</td> <td>Brakes</td> </tr> <tr> <td>FLEET EXHAUSTS</td> <td>Exhausts</td> </tr> <tr> <td>FLEET SHOCK ABSORBERS</td> <td>Shock absorbers</td> </tr> <tr> <td>FLEET BATTERIES</td> <td>Batteries</td> </tr> <tr> <td>FLEET GLASS</td> <td>Glass</td> </tr> <tr> <td>FLEET TRUCK STOP</td> <td>Truck Stop</td> </tr> <tr> <td>FLEET FORECOURT SHOP</td> <td>Forecourt Shop</td> </tr> </table>	FLEET OIL	Oil	FLEET SERVICE	Service	FLEET TYRES	Tyres	FLEET ACCIDENTS	Accidents	FLEET ACCESSORIES	Accessories	FLEET BRAKES	Brakes	FLEET EXHAUSTS	Exhausts	FLEET SHOCK ABSORBERS	Shock absorbers	FLEET BATTERIES	Batteries	FLEET GLASS	Glass	FLEET TRUCK STOP	Truck Stop	FLEET FORECOURT SHOP	Forecourt Shop
FLEET OIL	Oil																												
FLEET SERVICE	Service																												
FLEET TYRES	Tyres																												
FLEET ACCIDENTS	Accidents																												
FLEET ACCESSORIES	Accessories																												
FLEET BRAKES	Brakes																												
FLEET EXHAUSTS	Exhausts																												
FLEET SHOCK ABSORBERS	Shock absorbers																												
FLEET BATTERIES	Batteries																												
FLEET GLASS	Glass																												
FLEET TRUCK STOP	Truck Stop																												
FLEET FORECOURT SHOP	Forecourt Shop																												

					FLEET HOTEL EXPENSES	Hotel Expenses
					FLEET TOLL GATE	Toll Gate
					FLEET MISCELLANEOUS	Miscellaneous
					FLEET PREMIUM	Premium Fuel
					FLEET REGULAR	Regular Fuel
					FLEET SASOL	Sasol Fuel
					FLEET DIESEL	Diesel Fuel
					FLEET UNLEADED SUPER	Unleaded Super Fuel
					FLEET UNLEADED PREMIUM	Unleaded Premium Fuel
					FLEET AVIATION	Aviation Fuel
Quantity	subtag	N	0	8	The total number of line item units	
QuantityDecimalPlaces	subtag	N	0	4	The number of implied decimals in the Quantity	
UnitCost	subtag	N	0	10	The cost of a line item unit in the currency's smallest denomination	

AirlineData Parameters (see section 19)					
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description
PassengerName	subtag	AN	0	20	Passenger name as printed on ticket.
PrimaryTicketNumber	subtag	N	0	15	The ticket number.
FirstDepartureLocationCode	subtag	AN	0	3	Code for departure airport location, eg. JNB for Johannesburg
FirstArrivalLocationCode	subtag	AN	0	3	Code for destination airport location, eg. JNB for Johannesburg
PNRNumber	subtag	AN	0	6	The PNR number
OfficeIATANumber	subtag	N	0	8	The office IATA number
OrderNumber	subtag	AN	0	8	The order number
PlaceOfIssue	subtag	AN	0	20	The ticket office location
DepartureDate	subtag	N	0	8	Date of departure in yyyymmdd format
CompleteRoute	subtag	AN	0	255	Concatenation of individual travel legs in the format ORIG1-DEST1[:ORIG2-DEST2...:ORIGn-DESTn], f or example:CPT-JNB :JNB-:NBO. For airport codes
DepartureTime	subtag	AN	0	15	<p>Departure time of the first leg of the trip. Use one of the following formats:</p> <ul style="list-style-type: none"> • HH:mm \"GMT\"zzz • HH = two digit hour in 24-hour format • mm = two digit minutes • zzz = time zone of the departing flight, for example: If the airline is based in city A, but the flight departs from city B, z is the time zone of city B at the time of departure. <p>Important For travel information, use GMT instead of UTC, or use the local time zone.</p> <p>Examples</p> <p>19:55 GMT+02:00 19:55 GMT+0200 11:25 GMT-03:00</p>

					11:25 GMT-0300 Note When specifying an offset from GMT, the format must be exactly as specified in the example. Insert no spaces between the time zone and the offset.
JourneyType	subtag	AN	0	32	Type of travel, for example: one way or round trip.
Note: The fields in Blue will only be used when doing CyberSource Advanced Fraud Screening and will not be recorded on the iVeri system.					

CyberSource Parameters (see section 20)					
Parameter	Node Type	Data Type	Required	Maximum Length	Description
DeliveryMethod	subtag	AN	O	255	Physical or Virtual
DeviceFingerprintID	subtag	AN	O	88	Session ID for the fingerprint.
BillTo_FirstName	subtag	AN	R	60	First name of cardholder
BillTo_LastName	subtag	AN	R	60	Last name of cardholder
BillTo_Street	subtag	AN	R	60	Street address of cardholder
BillTo_City	subtag	AN	R	50	Billing city for cardholder
BillTo_State	subtag	AN	O	2	State or province of the cardholder. Required for U.S. and Canada. Use the two-character state, province, or territory codes.
BillTo_Country	subtag	AN	R	2	Billing country for cardholder. Use the two-character country codes.
BillTo_PostalCode	subtag	AN	O	10	Postal code of the billing address. Required only if the BillTo_Country field is US or CA
BillTo_Email	subtag	AN	R	100	Email address of the cardholder
BillTo_IPAddress	subtag	AN	O	15	IP address of the cardholder as reported by your Web server via socket information.
Note: If any of the fields marked in Red is unavailable all the fields in Red will be replaced with defaults specified by CyberSource					

7.5 Parameters per action

7.5.1 Input Parameters per action

The following table uses the following key for input parameters per action:

M	Mandatory
O	Optional
C	Conditional
blank	not relevant

Context	Parameter	Authorisation with PAN	Authorisation with Track2	Authorisation with VisaCheckoutCallID	Additional Authorisation with TransactionIndex	AuthorisationReversal with TransactionIndex	Debit with PAN	Debit with Track2	Debit with PIN	Debit with VisaCheckoutCallID	Debit with TransactionIndex	Credit with PAN	Credit with Track2	Credit with TransactionIndex	Void	Balance Enquiry with PIN	PAN Enquiry with PAN	PANToken Enquiry	PAN Enquiry with Track2	Ping	Get Device PIN Key	ThreeDSecureCheckEnrollment	ThreeDSecureValidateAuthentication
Core	ApplicationID	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Core	Category	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Core	CertificateID	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Core	Gateway	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
Core	Command	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Core	Mode	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Common	Amount	M	M	M	M	O	M	M	M	M	O	M	M	O								M	M
Common	AuthorisationCode						O	O		O													
Common	BudgetPeriod	O	O	O	O		O	O		O													
Common	CardSecurityCode	O	O	O	O	O	O	O		O			O										
Common	PAN	M					M					M					M	M				M	M
Common	Currency	O	O	O			O	O	O	O	O	O			O							M	M
Common	VisaCheckoutCallID			M																			
Common	ExpiryDate	M			O	O	O				O	O	O				O	M				M	M
Common	CardholderName																						
Common	MerchantReference	M	M	M	M	M	M	M	M	M	M	M	M									M	M
Common	OriginalMerchantReference	C					C					C											
Common	MerchantTrace	O	O	O	O	O	O	O	O	O	O	O	O	O									
Common	OriginalMerchantTrace															C							
Common	OriginalRequestID														C								
Common	PurchaseDate	O	O	O	O	O	O	O	O	O	O	O	O	O									
Common	PurchaseTime	O	O	O	O	O	O	O	O	O	O	O	O	O									
Common	StartDate	O			O	O	O				O	O					O						
Common	Terminal	O	O	O	O	O	O	O	O	O	O	O	O	O									
Common	Track2		M		O	O		M	M		O		M	O		M			M				
Common	TransactionIndex				M	M					M			M									

Context	Parameter	Authorisation with PAN	Authorisation with Track2	Authorisation with VisaCheckoutCallID	Additional Authorisation with TransactionIndex	Authorisation Reversal with TransactionIndex	Debit with PAN	Debit with Track2	Debit with PIN	Debit with VisaCheckoutCallID	Debit with TransactionIndex	Credit with PAN	Credit with Track2	Credit with TransactionIndex	Void	Balance Enquiry with PIN	PAN Enquiry with PAN	PAN Token Enquiry	PAN Enquiry with Track2	Ping	Get Device PIN Key	ThreeDSecureCheckEnrollment	ThreeDSecureValidateAuthentication
Common	MobileMoneyID														O								
Common	MSISDN						C																
Common	PANFormat	O		M			O			M		O											
Common	PANMode	O	O	O	O	O	O	O	O	O	O	O	O	O		O	O		O				
Common	CardHolderPresence	O	O		O		O	O	O														
3DSecure	ElectronicCommerceIndicator	O	O		O		O	O															
3DSecure	CardHolderAuthenticationData	O	O		O		O	O															
3DSecure	CardHolderAuthenticationID	O	O		O		O	O															
3DSecure	ThreeDSecure_RequestID																						M
3DSecure	ThreeDSecure_SignedPAREs																						M
POS	AccountType								O							O							
POS	CashAmount								O														
POS	DeviceFirmware								O							O						O	
POS	DeviceFirmwareVersion								O							O						O	
POS	DeviceMake								M							M						M	
POS	DeviceSerialNumber								M							M						M	
POS	DeviceCycle						O	O	O		O		O	O									
POS	KeySerialNumber								C							C							
POS	Track2KeySerialNumber		C						C	C			C			C			C				
POS	PANKeySerialNumber		C						C	C			C			C			C				
Procurement	CustomerReferenceIdentifier						O	O			O	O	O	O									
Procurement	CustomerVATRegistrationNumber						O	O			O	O	O	O									
Procurement	DestinationCountry						O	O			O	O	O	O									
Procurement	DestinationZIPCode						O	O			O	O	O	O									
Procurement	NationalTax						O	O			O	O	O	O									
Procurement	NationalTaxIndicator						O	O			O	O	O	O									
Procurement	OrderDate						O	O			O	O	O	O									
Procurement	PurchaseIdentifier						O	O			O	O	O	O									
Procurement	ShipFromZIPCode						O	O			O	O	O	O									
Procurement	ShippingAmount						O	O			O	O	O	O									
Procurement	ShippingTaxRate						O	O			O	O	O	O									
Procurement	TransactionDiscount						O	O			O	O	O	O									
Procurement	UniqueVATInvoiceReferenceNumber						O	O			O	O	O	O									
Procurement: Lineltem	Discount						O	O			O	O	O	O									

Context	Parameter	Authorisation with PAN	Authorisation with Track2	Authorisation with VisaCheckoutCallID	Additional Authorisation with TransactionIndex	Authorisation Reversal with TransactionIndex	Debit with PAN	Debit with Track2	Debit with PIN	Debit with VisaCheckoutCallID	Debit with TransactionIndex	Credit with PAN	Credit with Track2	Credit with TransactionIndex	Void	Balance Enquiry with PIN	PAN Enquiry with PAN	PAN Token Enquiry	PAN Enquiry with Track2	Ping	Get Device PIN Key	ThreeDSecureCheckEnrollment	ThreeDSecureValidateAuthentication	
Procurement: Lineltem	ItemCommodityCode						O	O			O	O	O	O										
Procurement: Lineltem	ItemDescriptor						O	O			O	O	O	O										
Procurement: Lineltem	ProductCode						O	O			O	O	O	O										
Procurement: Lineltem	Quantity						O	O			O	O	O	O										
Procurement: Lineltem	TaxRate						O	O			O	O	O	O										
Procurement: Lineltem	Total						O	O			O	O	O	O										
Procurement: Lineltem	UnitCost						O	O			O	O	O	O										
Procurement: Lineltem	UnitOfMeasure						O	O			O	O	O	O										
Fleet	CustomerReferenceIdentifier						O	O			O	O	O	O										
Fleet: Lineltem	ProductCode						O	O			O	O	O	O										
Fleet: Lineltem	Quantity						O	O			O	O	O	O										
Fleet: Lineltem	UnitCost						O	O			O	O	O	O										

7.5.2 Output Parameters per action

The following table uses the following key for output parameters per action:

Y	relevant (may be populated)
C	returned if supported by Acquirer
blank	not relevant

Context	Parameter	Authorisation with PAN	Authorisation with Track2	Additional Authorisation with TransactionIndex	AuthorisationReversal with TransactionIndex	Debit with PAN	Debit with Track2	Debit with PIN	Debit with TransactionIndex	Credit with PAN	Credit with Track2	Credit with TransactionIndex	Void	Balance Enquiry with PIN	PAN Enquiry with PAN	PANToken Enquiry	PAN Enquiry with Track2	Ping	Get Device PIN Key	ThreeDSecureCheckEnrollment	ThreeDSecureValidateAuthentication
Card Detail	Issuer	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y		Y					
Card Detail	Jurisdiction	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y		Y					
Result	Code	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y		
Result	Description	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y		
Result	Source	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y		
Result	Status	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y		
POS	AccountType							C						C							
POS	AvailableBalance							C						Y							
POS	CurrentBalance							C						Y							
POS	DisplayAvailableBalance							C						C							
POS	DisplayCashAmount							Y						Y							
POS	DisplayCurrentBalance							C						C							
POS	DevicePINKey																		Y		
POS	MACDevicePINKey																		Y		

8 Transaction sequences and terminology

8.1 Overview

An iVeri transaction is the combination of a transaction request (command: `Authorisation`, `AuthorisationReversal`, `Debit`, `Credit`, `Void`), and its corresponding response.

A transaction sequence relates to the complete set of movement of goods and services, and can include many related transactions.

An iVeri transaction involves communication between the following players:

Card Holder ↔ Merchant ↔ iVeri Gateway ↔ Acquirer ↔ Association ↔ Issuer

8.2 Unique Identifiers

The players in an iVeri transaction generate the following fields to identify an individual transaction:

<i>Player</i>	<i>Individual transaction identifier</i>
Merchant	Merchant Trace
iVeri Gateway	RequestID
Acquirer	Acquirer Reference

The players in an iVeri transaction generate the following fields to identify a transaction sequence:

<i>Player</i>	<i>Transaction sequence identifier</i>
Merchant	Merchant Reference
iVeri Gateway	Transaction Index

A `MerchantReference` is a unique identifier defined by the Merchant for a transaction sequence within a limited time period. A `MerchantReference` is mandatory (for initial transaction requests and optional for follow-up requests). It typically corresponds to an invoice or ticket number (a transaction sequence).

A `MerchantTrace` is a unique identifier for each request sent to the gateway, and is an optional parameter. The `MerchantTrace` corresponds to a database index that was generated before a request was sent to the gateway. In short the `MerchantTrace` refers to a particular step in the transaction sequence.

[Example]:

For a `Debit` followed by an immediate `Void` (cancellation of ticket etc): The `MerchantReference` remains the same for both steps of the transaction sequence, while the `MerchantTrace` is different for the `Debit` and `Void`.

A single `MerchantReference` can be associated with many `MerchantTrace`'s. Similarly a single `TransactionIndex` can be associated with many `RequestID`'s.

8.3 TransactionIndex and Follow up transactions

`TransactionIndex` refers to the unique identifier given by the iVeri Gateway to a set of related transactions. When a `TransactionIndex` is an input parameter, then the command is referred to as a "follow up". Therefore the actions with the suffix "with `TransactionIndex`" mean "as a Follow up transaction".

Follow up transactions by default use the same card information that was set in the initial

transaction, however by default the follow up is considered a Card not present (Keyed) transaction. Specifying one of the following mutually exclusive optional follow up input parameters can change the default behaviour:

- a) `Track2` : the follow up transaction is considered a “Swiped” transaction
- b) `ExpiryDate`: applicable to change when the original expiry date is in the past.

A follow up transaction can be done within 6 months of the original transaction. It can be used within a valid transaction sequence (eg a credit after a debit), but not for an invalid sequence (eg a debit following a credit). It cannot be used for a PIN based transaction.

8.4 Reversal transaction (Negative transaction)

A reversal transaction is an equal but opposite transaction to a previously successful transaction. This is typically a refund (i.e. a credit following a debit). A reversal typically results in both legs of the transaction being shown on the merchants and cardholders statements.

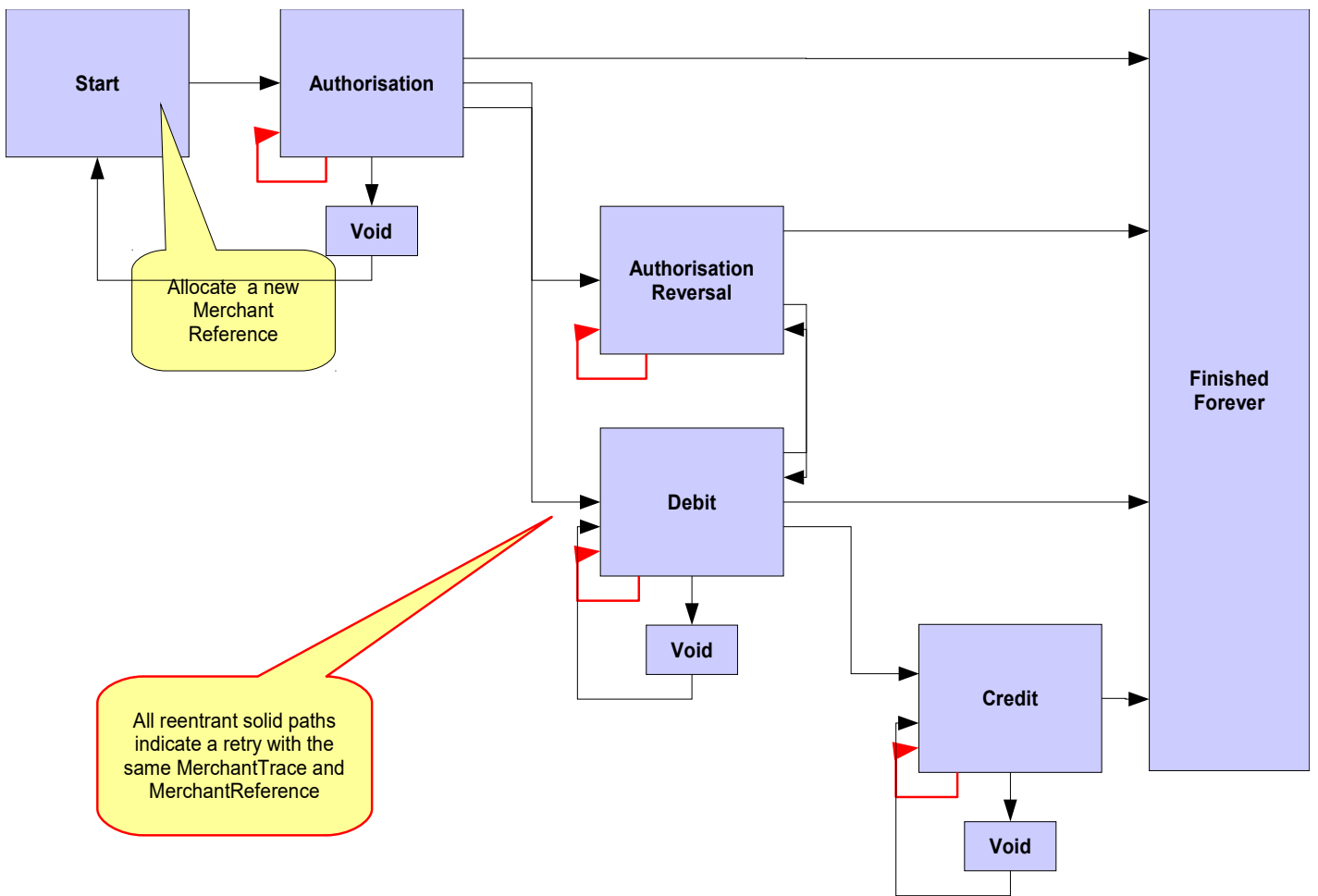
A merchant may initiate a reversal any time after a transaction was processed. A merchant can perform the reversal either by:

- performing a follow up transaction within 6 months of the original iVeri transaction, or
- initiating a new transaction request with the card holders details.

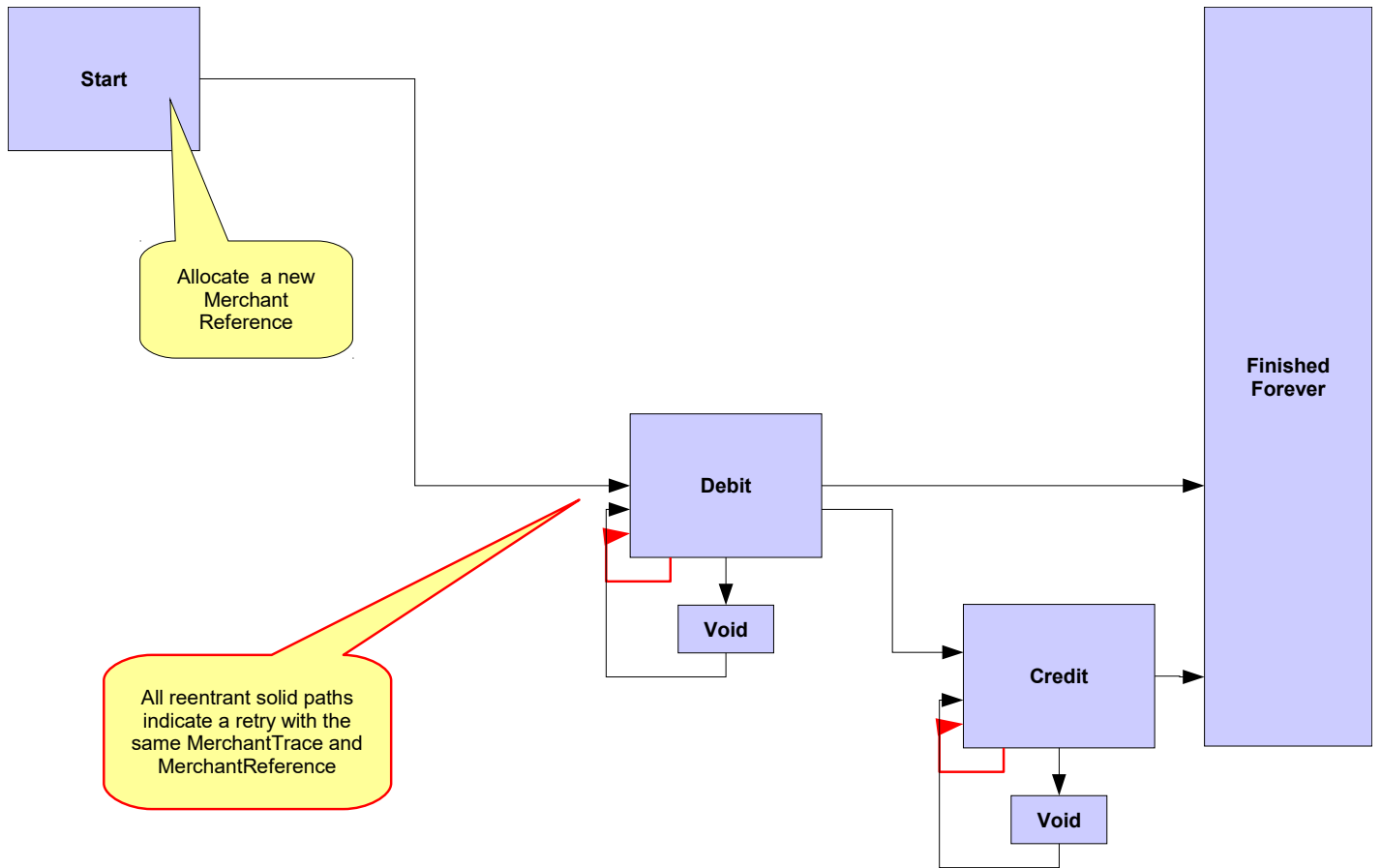
A reversal should not be confused with a Void (section 9.2.1)

8.5 Sequences

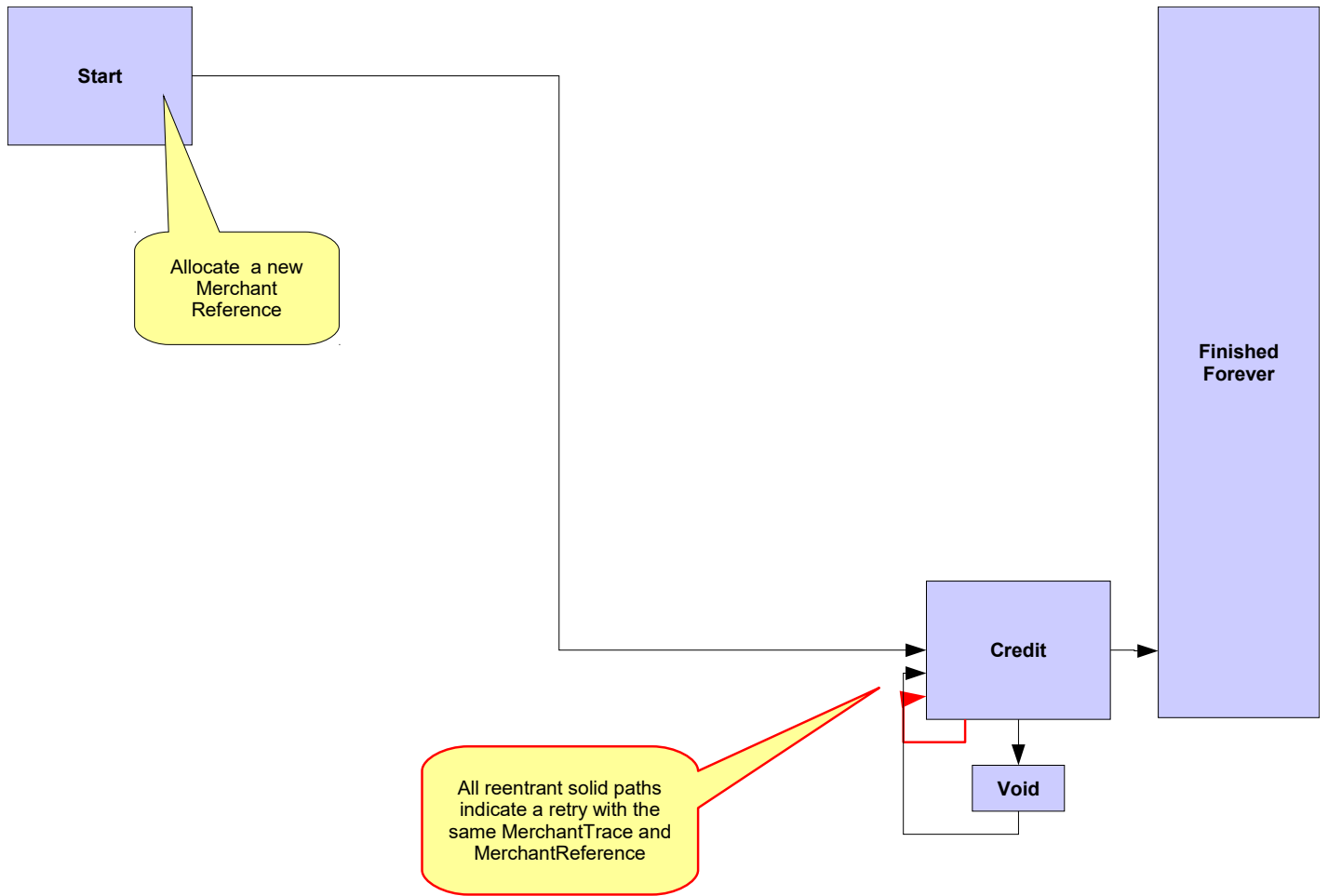
The possible transaction sequence flow with an initial Authorisation is the following:



The possible transaction sequence flow with an initial Debit is the following:



The possible transaction sequence flow with an initial Credit is the following:



9 Ensuring end to end transaction integrity

9.1 Overview

Transaction integrity can be seen as ensuring that all players within an individual iVeri transaction agree on its outcome. When a Card Holder gets his statement from his Issuer, it must correspond to his instruction to the merchant for a corresponding transaction.

When transaction integrity is compromised either willfully (fraud) or unintentionally, it can result in a disputed transaction with legal impacts.

9.2 Individual transactions

9.2.1 Void

A “Void” of a previous transaction request is a command to ignore (i.e. cancel the effects of) a previous (recently submitted) transaction request.

When a merchant receives a successful response for a transaction request, and thereafter “something goes wrong”, then the Merchant has the option to “void” the transaction.

Examples of “something goes wrong” are:

- communications error between merchant and cardholder
- printer could not print the invoice
- problem accessing merchant database

A merchant can also call “void” when s/he does not know whether a response was received from the iVeri Gateway (for example a power failure).

A successful void (cancellation) typically results in the transaction not being shown on the merchant nor the cardholders statements.

Transaction sets are settled as a group at predefined periods (cycle cut off time).

A “Void” can only be successful if the transaction being cancelled has not passed its cut off time. Therefore, when a void is required, it should be done as soon as possible (particularly since cycle cut off time is out of the control of a merchant). If the void request is submitted too late, then the void request will fail, and a separate reversal transaction would be required.

Enterprise users call Void with the syntax similar to:

```
enterprise.prepare("Transaction", "Void", applicationID, mode);
```

A Void request must set one of the following parameters: OriginalMerchantTrace or OriginalRequestID, for example:

```
enterprise.setTag("OriginalRequestID", requestID);
```

A Void has one of the following responses:

Result Code	Result Description	Action
0	Void successful	Transaction successfully voided.
13	Void unsuccessful	Too late to void the transaction.
1	Timeout	Void can be automatically retried a minute later
9	Unable to process	Void can be automatically retried a minute later
255	General Error (Exception)	Void can be retried a minute later after the problem given in the ResultDescription is fixed (which may require manual intervention)

If the iVeri Gateway receives a `Void` request referring to a `MerchantTrace` or `RequestID` that it has no knowledge of, then “Void successful” is replied.

A merchant implementing “Void” command requires a void repeat mechanism which only ends upon receiving a “Void successful” or “Void unsuccessful” response.

If you intend to resubmit the transaction, then you can either:

- Submit it immediately using a new `MerchantReference`, and have a separate process or thread to manage the sending of Voids (which may only be sent a few minutes later)
- Implement a mechanism where the transaction is only sent after the void process for the transaction is completed.
- Implement a retry mechanism discussed below without using a Void.

9.2.2 Retry

A merchant may want to retry a transaction after receiving an unsuccessful response.

If a merchant is not sure what the iVeri Gateway replied previously, then they might even want to retry after a receiving a successful response.

Typically a retry is relevant after receiving one of the following results on a transaction request:

<i>Result Code</i>	<i>Result Description</i>	<i>Action</i>
1	Timeout	Transaction can be automatically retried a minute later
9	Unable to process	Transaction can be automatically retried a minute later
255	General Error (Exception)	Transaction can be retried a minute later after the problem given in the <code>ResultDescription</code> is fixed (which may require manual intervention)

A retry is relevant after the above responses because then one would expect a different result code. Nevertheless a retry is supported after other result codes (eg `Successful` / `Denied` / `Hot card` / `Black card`) for when there is some change that could cause the transaction to be approved, or when the original reply is unknown to the merchant.

The following shows three different approaches to retries. A retry is only recommended in using the first approach, however all these approaches are supported.

9.2.2.1 Retry with client recorded Merchant Trace

A merchant records transaction information in their database before they send a transaction request to the iVeri Gateway. The database index of this request is given as the `MerchantTrace` (to uniquely identify an individual transaction request).

The merchant retries a previous transaction with the same `MerchantTrace`, `MerchantReference`, `Command`, `Amount`, `PAN` parameters as previously.

The possible responses are:

- If the previous attempt was successful, a successful result is returned.
- If the previous attempt was unsuccessful, the transaction request is retried.
- If the previous attempt is in process, then “Transaction in process” (`ResultCode 9 : Unable to process transaction`) is returned. The retry can then be reattempted a minute later.

If the merchant is only performing debits, and there is a database table where the unique identifier corresponds to the reference number given to a card holder, then it may make sense for the `MerchantReference` and `MerchantTrace` to be the same.

NB. The NedbankBICISO provider does not allow a follow-up debit to be Voided. If a transaction cannot be Voided and no response is received from the iVeri Gateway, the transaction has to be retried instead. For this, all transactions done with the `ApplicationID` has to be with `MerchantTrace`. Once a response is received for the follow-up debit, a follow-up credit has to be performed to undo the debit if it was successful.

9.2.2.2 *Retry without Merchant Trace*

It is not recommended to retry a transaction without a `MerchantTrace`.

Nevertheless, if the same `Command`, `Amount`, `PAN`, and `MerchantReference` as a previous transaction are specified (with no `Merchant Trace` given), then the possible responses are:

- If the previous attempt was successful, then "A transaction with the same `MerchantReference` was successful" (Result Code 255 : Duplicate `MerchantReference`).
- If the previous attempt was unsuccessful, the transaction request is retried.
- If the previous attempt is in process, then "Transaction in process" (Result Code 9 : Unable to process transaction) is returned. The retry can then be reattempted a minute later.

9.2.2.3 *Retry with irrelevant Merchant Trace (or irrelevant Merchant Reference)*

A merchant may be tempted to bypass the various checking mechanisms within the iVeri Gateway and just send the current date time stamp as the value for the `MerchantTrace` or `MerchantReference`. In this manner there is more chance that a transaction request will be successful. However there is also more chance that the card holder will be double debited!

For merchants who cannot store an individual request identifier before sending the request to the iVeri Gateway, see section 9.3.3 on "Recurring transaction checking".

9.2.1 *Enquiry*

A merchant with a doubt of the current status of a transaction can determine its status by:

- Performing a "Transaction lookup" within iVeri BackOffice
- Download a "Reconciliation file" (can be automated with iVeri FileTransfer) and compare records.
- Check an immediate notification mechanism (such as email for iVeriBatch clients, and SMS for iVeri Voice clients)
- Look in the logs for an exception message containing details of a misconfiguration or a coding error.

9.2.2 *Conclusion*

The Void, Retry and Enquiry mechanisms discussed above help to either fix an individual transaction integrity, or resolve the doubt around what the current state is.

A merchant should automate at least one of these mechanisms.

If a Void is unsuccessful, or the current state does not correctly correspond to the corresponding transfer of goods or services, then a follow up reversal transaction may be required.

9.3 *Duplicate transactions*

A problem of a duplicate transaction can occur if a merchant submits a previously successful transaction in a new request. A duplicate transaction of this nature is typically due to an users unintentional mistake, e.g. pressing the "Submit" button twice, or submitting the same batch twice. It is responsibility of the merchant to ensure that a single transaction request is not submitted successfully more than once.

Nevertheless the iVeri Gateway provides three mechanisms to protect against duplicate transactions. Specifying a unique `MerchantTrace` is a client side configuration, while the latter to require contacting your local distributor.

9.3.1 Specify a unique Merchant Trace for each step in a Transaction Sequence

As mentioned in section 8.2, a `MerchantTrace` is a unique identifier for each request sent to the gateway, and is an optional parameter. The `MerchantTrace` corresponds to a database index that was generated by the merchant before a request was sent to the gateway. In short the `MerchantTrace` refers to a particular step in the transaction sequence.

The recommended way to control duplicate transactions is to always specify a `MerchantTrace`. This has two benefits

- If a merchant does not receive a reply to a request then they have the choice of either voiding (9.2.1) or retrying (9.2.2) the transaction by using the same `MerchantTrace`.
- A merchant can re-use a `MerchantReference` with different `MerchantTraces` for the same `TransactionIndex`.

9.3.2 Merchant Reference validity period

A `MerchantReference` is a unique identifier allocated by the merchant for a transaction sequence.

The `MerchantReference` validity period is a mechanism to protect merchants against undesired double debiting.

When performing an initial transaction request (i.e. no `TransactionIndex` provided), then if the last use of that `MerchantReference` (within a time period) was a successful, then a new Transaction is not performed.

When performing a follow up transaction (i.e. `TransactionIndex` provided), then if the last use of the `MerchantReference` (within a time period) of the same transaction type was successful, then a new Transaction is not performed. [Assuming the Transaction Type Repetition option has not been activated].

The default time period (Merchant Reference validity period) discussed above is 6 months. This can be changed to a minimum of 3 days.

9.3.3 Recurring transaction checking

The Recurring transaction checking period is an additional mechanism to protect merchants against undesired double debiting.

By default, recurring transaction checking is disabled.

When enabled, if a transaction is attempted with the same `PAN`, `Amount`, `Command` combination, but a different `MerchantReference` as a previously successful transaction done within a period, then the subsequent transaction request will fail.

If a Merchant explicitly requests this to be enabled, a time period (in seconds / minutes / hours) should be specified. This is typically 5 minutes.

If a Merchant uses `MerchantTrace` to uniquely assign each individual transaction before submission to the iVeri Gateway, then this option should not be needed.

This option is recommended for merchants who are forced to have poor state handling - for example those that generate a merchants reference in memory and only write to the database after sending a request to the iVeri Gateway.

Note that even when this option is disabled, an acquirer or issuer "downstream" may have their equivalent option enabled.

10 Common parameters

10.1 Result Codes

The following is the list of Result Codes:

Result Status	Result Code	Result Description	Only relevant when
0 (OK)	0	N/A (Approved)	
-1 (Not OK)	1	Timeout waiting for response	
-1 (Not OK)	3	Hot card	
-1 (Not OK)	4	Denied	
-1 (Not OK)	5	Please call	
-1 (Not OK)	6	Card Address failure	
1 (Warning)	6	Warning: Approved but Card Address failure	
-1 (Not OK)	7	Card Security Code failure	
1 (Warning)	7	Warning: Approved but Card Security Code failure	
-1 (Not OK)	8	Card Type not accepted	
-1 (Not OK)	9	Unable to process the transaction	
-1 (Not OK)	10	Card blocked	
-1 (Not OK)	11	Invalid amount	
-1 (Not OK)	12	Invalid budget period	
-1 (Not OK)	13	Void unsuccessful	Command Void
-1 (Not OK)	14	Invalid card number	
-1 (Not OK)	15	Invalid track2	
-1 (Not OK)	16	Invalid card date (Expiry Date or Card period)	
-1 (Not OK)	18	Invalid authorisation code	
-1 (Not OK)	19	Incorrect PIN	PINBlock submitted
-1 (Not OK)	20	Device PIN Key has expired	PINBlock submitted
1 (Warning)	21	Warning: Approved but Cash Denied	PINBlock submitted and CashAmount submitted
-1 (Not OK)	22	EMV not supported	
-1 (Not OK)	23	Card information not present	
-1 (not OK)	24	Invalid Recurring Account	CardHolderPresence 'Recurring' specified
1 (Warning)	25	Warning: Approved but Identification Required	
-1 (Not OK)	100	The requested file is not available for download	FileTransfer
-1 (Not OK)	101	The specified file has already been uploaded	FileTransfer
-1 (Not OK)	255	General Error (Exception)	

The following Result Codes require special mention:

Result Status	Result Code	Result Description	Examples	Action
-1 (Not OK)	1	Timeout waiting for response	<ul style="list-style-type: none"> - There was no reply from the card holders issuer - Request was not processed within an expected time period 	- Request can be automatically retried (with a wait in between)
-1 (Not OK)	9	Unable to process the transaction	<ul style="list-style-type: none"> - The connection between iVeri and the acquirer is down - The iVeri Gateway is offline for maintenance 	- Request can be automatically retried (with a wait in between)
-1 (Not OK)	255	General Error (Exception)	<ul style="list-style-type: none"> - The ApplicationID has been suspended - Could not establish a connection to the iVeri Gateway due to no network connection from the client. - The merchant code is calling the iVeri Gateway incorrectly. 	- Request can be retried only after someone has read the error description and done an action to fix it

10.2 Mode: Test vs Live

The iVeri Gateway provides a mechanism where a merchant can perform test transactions that are routed to an iVeri Gateway issuer simulator. This enables a merchant to complete testing within a test environment. When the merchant is ready s/he can perform live transactions which are routed to the genuine card issuer.

A merchant specifies the mode of a transaction by setting Mode as “Test” or “Live”, and sends their corresponding Test or Live ApplicationID.

Sending the following card numbers to the Test environment has the following results:

PAN	Result
4242424242424242 or 5959595959595959	returns "Authorised"
2121212121212121	randomly returns "Hot Card", "Please Call" or "Denied"
5454545454545454	randomly returns "Unable to process" or times out
All other card numbers (eg "1111222233334444")	returns "Invalid card number"

For information on testing PIN based accounts, see “PIN based transactions”.

Merchants are requested not to abuse the test environment more than their realistic requirements.

A merchant abusing the Test environment may have their use of it suspended.

10.3 Track2

The Track2 is used for card present transactions and is sent to the iVeri Gateway after it is read by the swipe device from the magnetic stripe on the card. It is inclusive of the beginning and end markers being ; and ? respectively.

The Track2 sent to the iVeri Gateway must be in the following format:

Field	Description	Length (bytes)	Format
SS	Start Sentinel	1	;
PAN	Primary Account Number	19 max	Digits
FS	Field Separator	1	=
DATE	Expiration Date (YYMM)	4	Digits
SVC CD	Service Code	3	Digits
Discretionary Data	Optional Issuer Data	variable	Digits
ES	End Sentinel	1	?
	Total cannot exceed 39 bytes	39	

The Track2 read from by the card reader typically contains the following fields (at most 40 bytes):

SS	PAN	FS	DATE	SVC CD	Discretionary Data	ES	LRC
----	-----	----	------	--------	--------------------	----	------------

The 1 byte Longitudinal Redundancy Check (LRC) should not be sent to the iVeriGateway.

The End Sentinel (ES) may need to be converted to ? (0x3F)

10.3.1 Track2 Service Code values

The following tables can be used to determine the meaning of the 3 digit service code.

See section 16.6.3 for algorithm to determine if a card is PIN based or not.

Note that a transaction may not be declined or rejected solely because of the value of the service code.

Value Position #1	Description	MasterCard	Visa
1	International Card	Yes	Yes
2	International Card – Integrated Circuit Card	Yes	Yes
5	National Use Only	Yes	Yes
6	National Use Only – Integrated Circuit Card	Yes	Yes
7	Private Label or Proprietary Card	Yes	No

Value Position #2	Description	MasterCard	Visa
0	Normal Authorisation	Yes	Yes
2	Positive Online Authorisation Required	Yes	Yes

Value Position #3	Description	MasterCard	Visa
0	PIN Required	Yes	Yes
1	Normal Cardholder Verification, No Restrictions	Yes	Yes
2	Normal Cardholder Verification – Goods and services only at point of interaction (no cash back)	Yes	No
3	ATM Only, PIN Required	Yes	Yes
5	PIN Required – Goods and services only at point of interaction (no cash back)	Yes	No
6	Prompt for PIN if PIN Pad Present	Yes	Yes
7	Prompt for PIN if PIN Pad Present – Goods and services only at point of service (no cash back)	Yes	No

10.4 Budget Period

The Budget Period facility is only available within certain localities (eg South Africa), and therefore only currently available with distributor Nedbank.

Since a transaction sequence may entail the Budget Period being sent to the acquirer multiple times, the merchants should ensure that this value is consistent.

If an authorisation was obtained outside the iVeri Gateway, and then the AuthorisationCode is sent within “Debit with PAN” or “Debit with Track2”, then the BudgetPeriod should be set to the same value as when authorised. If the BudgetPeriod is not sent to the iVeri Gateway, then iVeri assumes to be “straight” (0 months). However if the external authorisation was “over budget”, then it would then be up to the acquirer/issuer to decide between the conflicting budget values.

11 Specialised techniques

11.1 Ping

The Ping command is primarily used to determine if the connection between the Merchant and the Acquirer is down. If the connection is down, then a Ping can be used to poll the iVeri Gateway to see when the status is back up.

The Ping command checks that:

- the merchant is communicating with the iVeri Gateway, and
- the merchant configuration is active, and
- the iVeri Gateway is online, and
- the iVeri Gateway is communicating with the acquirer.

This is different to checking network connectivity to the iVeri Gateway, where a network ping should be employed. The Ping takes an ApplicationID as a mandatory input parameter and is sometimes referred to as an ApplicationID Ping.

As mentioned previously, an iVeri transaction involves communication between the following players:

Card Holder ↔ Merchant ↔ iVeri Gateway ↔ Acquirer ↔ Association ↔ Issuer

An Acquirer can route a transaction to many different Associations, and an Association can route a transaction to many different Issuers.

An individual Transaction may be replied to with a ResultCode of 1 or 9, meaning something is wrong between the Merchant and the card issuer. However in such a case the merchant does not know if the problem was between the Acquirer and an Issuer (due to the routing of the PAN), or between the Merchant and the Acquirer.

Although the individual transaction can be automatically retried, the Merchant may have some business rules for the case when the connection between the Merchant and the Acquirer is down, such as:

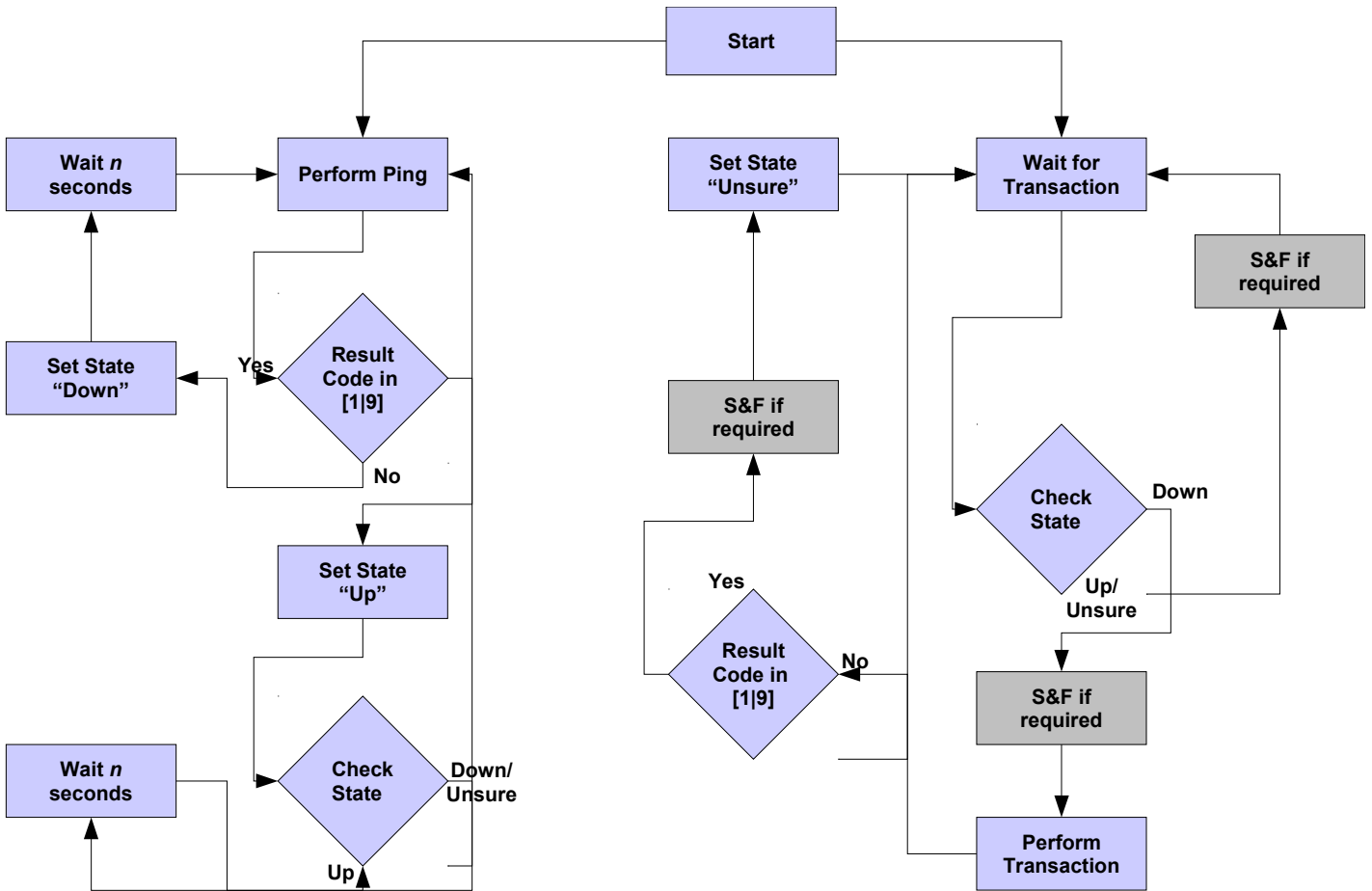
- go into Store and Forward mode
- notify a certain person
- show a different website page upon checkout

The Ping code of conduct:

- The Ping command should be used only when a merchant needs to know the connection status between the Merchant and the Acquirer after either:
 - A service startup or a Transaction result code 1 or 9 (in doubt)
 - A Ping result code 1 or 9 (connection down)
- A merchant should assume that the system is up. A Ping should not be used when the merchant thinks that the connection is up and the above conditions do not hold (i.e. it should not be used to indiscriminately repeatedly poll the iVeriGateway).
- When polling to see if the status is back up, a Ping may not be more frequent than every 20 seconds.

A merchant abusing the Ping command may have their use of it suspended.

The following is a diagram showing how a Ping can be effectively used. "S&F" refers to "Store and Forward", otherwise known as an Offline transaction.



11.2 Mod-10 checking

PANs are checked for validity using the Luhn check-digit algorithm (also known as "Mod-10 checking"). If the iVeri Gateway receives a transaction that fails Mod-10 checking, then "Invalid Card Number" (ResultCode 14) if returned.

A merchant may perform Mod-10 checking up front before sending the transaction to the iVeriGateway.

The steps of the algorithm are described below (however there are many Mod-10 algorithms available for download on the Internet):

1. Exclude the right-most digit from the calculation as this is the actual check-digit to be examined for validity.
2. Starting with the 2nd to the last digit and moving right to left, alternatively multiply each successive digit by 2 and 1 respectively.
3. Sum the integers comprising the product obtained from each of the calculations.
4. Subtract the resulting sum from the next higher multiple of ten (10). The resulting value is the desired account number check-digit.

The following example uses the PAN of: 4287 9478

Step	Description	Example
1	Example account number	4 2 8 7 9 4 7 (8)
2	Multiplier Products	$\begin{array}{cccccccc} \times 2 & \times 1 & \times 2 & \times 1 & \times 2 & \times 1 & \times 2 \\ 8 & 2 & 16 & 7 & 18 & 4 & 14 \end{array}$
3	Sum the integers	$8 + 2 + (1+6) + 7 + (1+8) + 4 + (1+4) = 42$
4	Derive the Check Digit	$50 - 42 = 8$

11.3 Card Number Checking

A merchant can perform multiple other checks on the PAN before initiating an iVeri Gateway transaction request. This can be achieved by using the contents of files downloaded with the Commands:

- HotCard
- BINLookup
- BINManagement
- BlackCard

See section 22

12 ThreeDSecure (also known as “VerifiedByVISA” or “MasterCard SecureCode”)

“Verified by VISA” (VbV) and “MasterCard SecureCode” is a new online service that uses personal passwords or identity information to help protect against unauthorised usage of VISA or MasterCard cards within the Internet environment.

This functionality within this section is currently only available for the Nedbank distributor.

12.1 Centinel MAPS

The Centinel MAPS (Merchant Authentication Processing System) performs the required interaction, message formatting and processing required by the different payment authentication initiatives such as:

- Verified by Visa
- MasterCard SecureCode
- SPA etc.

The Centinel MAPS product is a product of Cardinal Commerce and is distributed in South Africa by Bankserv. The Centinel MAPS service is hosted by Bankserve.

For Centinel integration queries and assistance, contact: cnpsupport@bankserv.co.za

12.1.1 Test Environment

The following information is needed when using the Centinel MAPS test environment for **MasterCard SecureCode** and **Verified by VISA**

- URL (MPI): <https://msgtest.bankserv.co.za/maps/txns.asp>

American Express SafeKey

- URL (MPI): <https://msgtest.bankserv.co.za/maps/amex.asp>

ProcessorID :1000

Password : DeMo123

MerchantID : 12345678

The Centinel MAPS test cards are available within the “CNP Secure Test Cards” document. These test cards are different to the iVeri test cards. Therefore when sending test cards to Centinel, use their test cards, and thereafter when sending the transaction to iVeri, set the card information according to section 10.2.

12.1.2 Live Environment

The following information is needed when using the Centinel MAPS live environment: **MasterCard SecureCode** and **Verified by VISA**

- URL (MPI): <https://msgnedcor.bankserv.co.za/maps/txns.asp>

American Express SafeKey

URL (MPI): <https://msgnedcor.bankserv.co.za/maps/amex.asp>

ProcessorID :1000

Password : **(Please request this from BankServ, send an email to**

cnpsupport@bankserv.co.za with your merchant number and Processer ID)

MerchantID : *This must be obtained from your [Distributor](#)*

12.2 The 3D Secure Transaction Process Flow

- See section 12.4 for a diagram of the 3D Secure Transaction Process Flow.
- At the point of checkout the Cardholder selects an appropriate payment method based on the initiatives supported by the Merchant. The Cardholder fills out the checkout form including the payment option and clicks buy.
- If the association of the Card is a MasterCard or Visa, and the merchant is enabled for 3D Secure, then the

Merchant passes a cmpi_lookup message to Centinel MAPS, containing the following values based on the payment information:

<i>FieldName</i>	<i>Description</i>
MsgType	cmpi_lookup
Version	Application Version identifier
ProcessorId	Application Processor identifier
MerchantId	Application Merchant identifier
OrderNumber	Order number from the Merchant Website
PurchaseAmount	Formatted total sale transaction
RawAmount	Total amount without decimalization
PurchaseCurrency	3 digit numeric ISO4217 currency code for the sale amount
PAN	Credit card number used for transaction
PANExpr	Credit card expiry date, formatted YYMM

Sample Message:

```
<CardinalMPI>
<MsgType>cmpi_lookup</MsgType>
<Version>1.3</Version>
<ProcessorId>1000</ProcessorId>
<MerchantId>12345678</MerchantId>
<OrderNumber>IVERI00001</OrderNumber>
<PurchaseAmount>ZAR11</PurchaseAmount>
<RawAmount>11</RawAmount>
<PurchaseCurrency>710</PurchaseCurrency>
<PAN>4242424242424242</PAN>
<PANExpr>0410</PANExpr>
</CardinalMPI>
```

- This message contains all the required information supplied by the cardholder in order to check the enrollment of the cardholder.
- Based on the card number range stored within Centinel MAPS, a Verify Enrollment Request (VEReq), message will be sent to the Enrollment directory server.
- The Enrollment directory server will send the VEReq to the cardholders Issuing Bank Access Control Server (ACS) where it will verify the enrollment status.
- The Verify Enrollment Response (VERes), is then passed back to the Directory server with the corresponding ACS url, if applicable.
- The information is then passed back to Centinel MAPS where it is verified and the Payment Authentication Request (PAREq) is created.
- The lookup response is then returned to the Merchant, Containing the following values:

<i>FieldName</i>	<i>Description</i>
ErrorDesc	Application error description for the associated errornumber
ErrorNo	Application error number, non zero, encountered while attempting to process the message request
TransactionId	This number is required to be passed on the cmpi_authenticate message
Payload	Contains the encoded PAREq generated by MAPS, available if Enrolled = Y
SPSHiddenFields	Contains the SPA hidden form fields, Available if Master card transaction
Enrolled	CardHolder enrolled status (Y - enrolled, N - not enrolled, U - Cardholder enrolled but authentication unavailable)
ACSUrl	Contains the fully qualified path of the Issuers ACS, this is used by the merchant to redirect the Cardholder, available if Enrolled = Y

Sample Message:

```
<CardinalMPI>
<ErrorDesc></ErrorDesc>
<ErrorNo>0</ErrorNo>
<TransactionId>jMpkHtqeHD+6GTlUhafK</TransactionId>
<Payload>*****PAREqMessage*****</Payload>
<SPASHiddenFields></SPASHiddenFields>
<Enrolled>Y</Enrolled>
<ACSUrl>http://dns/path</ACSUrl>
</CardinalMPI>
```

- Based on the existence of the ACS url in the lookup response the Merchant will redirect the cardholders browser to the corresponding ACS server. The cardholder enters their authentication data and initiates the Authentication process directly with the ACS server.
- The ACS, in conjunction with the Card Issuer, authenticates the cardholder. The Payment Authentication Response (PAREs), is send back to the Cardholder via the web browser.
- The PAREs is forwarded to the Merchant.
- The Merchant initiates a cmpi_authenticate message to the Centinel MAPS, containing the following values:

<i>FieldName</i>	<i>Description</i>
MsgType	cmpi_authenticate
Version	Application Version identifier
TransactionId	This links the cmpi_lookup with the cmpi_authenticate message together
ProcessorId	Application Processor identifier
MerchantId	Application Merchant identifier
PAResPayload	PARes generated by the Issuer ACS

Sample Message:

```
<CardinalMPI>
<msg_type>cmpi_authenticate</msg_type>
<version>1.3</version>
<processor_id>1000</processor_id>
<merchant_id>12345678</merchant_id>
<order_number>AUT000191</order_number>
<PAResPayload>*****PAResMessage*****</PAResPayload>
</CardinalMPI>
```

The authenticate response is then returned to the Merchant, Containing the following values:

<i>FieldName</i>	<i>Description</i>
ErrorDesc	Application error description for the associated errornumber
ErrorNo	Application error number, non zero, encountered while attempting to process the message request
Cavv	Transaction stain from PAREs
SignatureVerification	Status of the signature verification of the PAREs message (Y - validated successfully, N - not validated)
Xid	Transaction Xid from PAREs
EciFlag	E-Commerce indicator from PAREs
PAResStatus	Transaction status from PAREs (Y, N, U or A)

Sample Message

```
<CardinalMPI>
<ErrorDesc></ErrorDesc>
<ErrorNo>0</ErrorNo>
<Cavv>AAABAJEDFwAAAAAAMXAAAAAA=</Cavv>
<SignatureVerification>Y</SignatureVerification>
<Xid>QPqHUYfUAnlc4OJGzEM2lEzeAYU=</Xid>
<EciFlag>05</EciFlag>
<PAResStatus>Y</PAResStatus>
</CardinalMPI>
```

12.3 Using 3D Secure with iVeri

The merchant uses the accompanying flow diagram to decide whether to process or fail the transaction. When processing the transaction, the merchant determines the value of the ElectronicCommerceIndicator (ECI) flag, and sets the following parameters (tags):

- ElectronicCommerceIndicator (ECI),
- CardHolderAuthenticationData (CAVV) and
- CardHolderAuthenticationID (XID)

along with the rest of the transaction information and sends out the authorisation to the payment network.

The following are the iVeri parameters (tags) specific to 3D Secure:

- “ElectronicCommerceIndicator” which holds the enumerated ECI indicator which (if specified) is one of: [ClearChannel | SecureChannel | ThreeDSecureAttempted | ThreeDSecure]. The difference between SecureChannel or ClearChannel is whether SSL is used between the cardholder and the merchant.
- “CardHolderAuthenticationData” which holds the base64 encoded CAVV data
- “CardHolderAuthenticationID” which holds the base64 encoded XID data

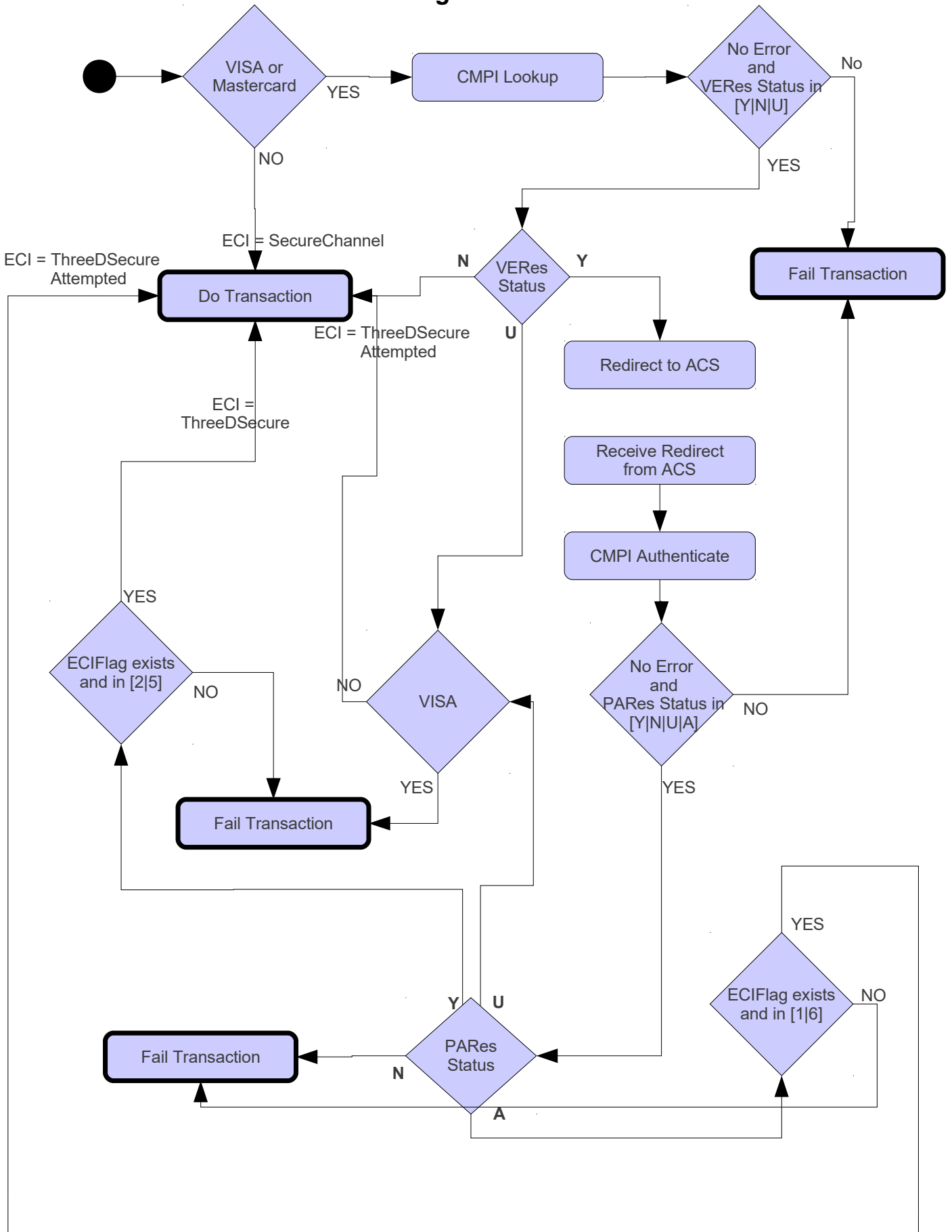
These 3 tags are set using iVeri Enterprise, for example:

```
enterprise.setTag("ElectronicCommerceIndicator", "ClearChannel");  
enterprise.setTag("CardHolderAuthenticationData", cavv);  
enterprise.setTag("CardHolderAuthenticationID", xid);
```

These tags are only needed to be set in an initial transaction, not when a TransactionIndex is set.

In other words, if a merchant performs an Authorisation and then a follow up Debit, then the above tags need only be set within the Authorisation.

12.4 ThreeD Secure Process Flow Diagram



12.5 3D Secure

Payer Authentication services enable you to quickly and easily add support for:

- Verified by Visa
- MasterCard SecureCode
- Amex SafeKey
- SPA etc.

To find out if your Visa- or MasterCard-branded card is supported by Payer Authentication, contact your payment processor.

These card authentication services deter unauthorized card use and enable you to receive protection, liability shift, from fraudulent chargeback activity.

Payer Authentication through has been made available by iVeri with the addition of the following new **Enquiry** commands.

- ThreeDSecureCheckEnrollment
- ThreeDSecureValidateAuthentication

Note: The Merchant has to be enabled for 3D Secure Checking on the iVeri Gateway.

12.5.1 Checking Enrollment

The goal is to check if the card is enrolled and to authenticate the results if it is enrolled. To do so, you would build up a **Enquiry** request with the command set as **ThreeDSecureCheckEnrollment**. The following required parameters needs to be send.

- *ApplicationID*
- *CertificateID*
- *Mode*
- *MerchantReference*
- *Amount*
- *Currency*
- *PAN*
- *ExpiryDate*

If the **Enrollment Check** was unsuccessful the merchant is not permitted to continue with the transaction. Instead, ask the customer for another form of payment.

If the **Enrollment Check** was successful the following parameters will be returned:

- ThreeDSecure_RequestID - Links the Enrollment process with the Authentication process.
- ThreeDSecure_ACS_URL - Only Present if card is Enrolled.
- ThreeDSecure_PAREq - Only Present if card is Enrolled.

If the card is not enrolled, skip the **Validate Authentication**, and proceed with the authorisation. The following would also be returned and is required to be submitted through to the gateway only if the card was not enrolled.

- *CardHolderAuthenticationID*
- *CardHolderAuthenticationData*
- *ElectronicCommerceIndicator*

If the card is enrolled. The merchants website will have to be redirected to the Access Control Server (**ACS**) URL and the **PAREq** must be posted to it. an Example of the redirect page will be provided in **Appendix B**.

12.5.2 Validate Authentication

For enrolled cards, the next step is to request the validation service to verify the authentication response message returned by the card-issuing bank. To do so, you would build up a **Enquiry** request with the command set as **ThreeDSecureValidateAuthentication**. The following required parameters needs to be send.

- *ApplicationID*
- *CertificateID*
- *Mode*
- *MerchantReference*
- *Amount*
- *Currency*
- *PAN*
- *ExpiryDate*
- *ThreeDSecure_RequestID* - *Links the Enrollment process with the Authentication process.*
- *ThreeDSecure_SignedPAREs* - *PAREs message extracted from the form. **Appendix B.***

If the **Validate Authentication** was unsuccessful the merchant is not permitted to submit the transaction for authorisation. Instead ask the customer for another form of payment.

If the **Validate Authentication** was successful the following parameters will be returned and is required to be send through to the gateway:

- *ThreeDSecure_RequestID* - *Not required tp be send to gateway with transaction.*
- *CardHolderAuthenticationID* - *Must be send to gateway with transaction.*
- *CardHolderAuthenticationData* - *Must be send to gateway with transaction.*
- *ElectronicCommerceIndicator* - *Must be send to gateway with transaction.*

13 NPay

nPay is designed to allow internet merchants to sell their products and services to card holders who have not previously been able to transact on the internet. nPay consists of the following two things:

1. A device and the nPay service that runs on a host machine. Once a machine has this package it is nPay enabled and is able to process both Signature and PIN based cards over the web from an nPay enabled website or by a local application.
2. The merchant needs to develop/modify either a website or application so that it can interact with the nPay device (through HTTP web requests).

NPay provides iVeri Enterprise merchants with a modern, clear and easy channel to interact with a device, alleviating the requirement of mastering a complex device protocol.

PIN based transactions are currently available only with distributors Nedbank.

iVeri uses the terminology *Device* where others may use the terminology *Terminal*.

13.1 NPay packages

An iVeri Enterprise merchant wishing to be an NPay merchant must contact their distributor to obtain an NPay development and/or production package.

13.1.1 NPay Development package

The NPay Development package consists of the following:

- nPay Installation instructions
- nPay exe (Windows) or nPay jar (Linux)
- Possum (POS terminal simulator)

The NPay Development package is used to enable a merchant to develop a NPay website/application within a test environment.

13.1.2 NPay Production package

The NPay Production package consists of the following:

- nPay Installation instructions
- nPay exe (Windows) or nPay jar (Linux)
- Device(s) certified for transactions of Mode Live

The NPay Production package is used to provide a device for cardholders to use to perform their NPay transactions.

13.2 NPay Architecture

The Cardholder accesses a computer that has:

- the NPay service installed
- an NPay device connected
- an Npay application or internet access to a nPay enabled merchants website

The Merchant website via the browser/application attempts to communicate with the local NPay service via JSON. If it is:

available: the website/application shows the NPay card present functionality.

unavailable: the website/application assumes it is a non NPay client, and presents the

normal card not present functionality.

The Merchant website/application uses the nPay service to control the behaviour of the device and use it to obtain card information, which is then transmitted to the iVeri Gateway using iVeri Enterprise.

The following two diagrams depicts an activity diagram showing a working relationships between the NPay host machine, Enterprise and:

- a) An nPay enabled website (figure 14.3.1).
- b) A merchant application (figure 14.3.2).

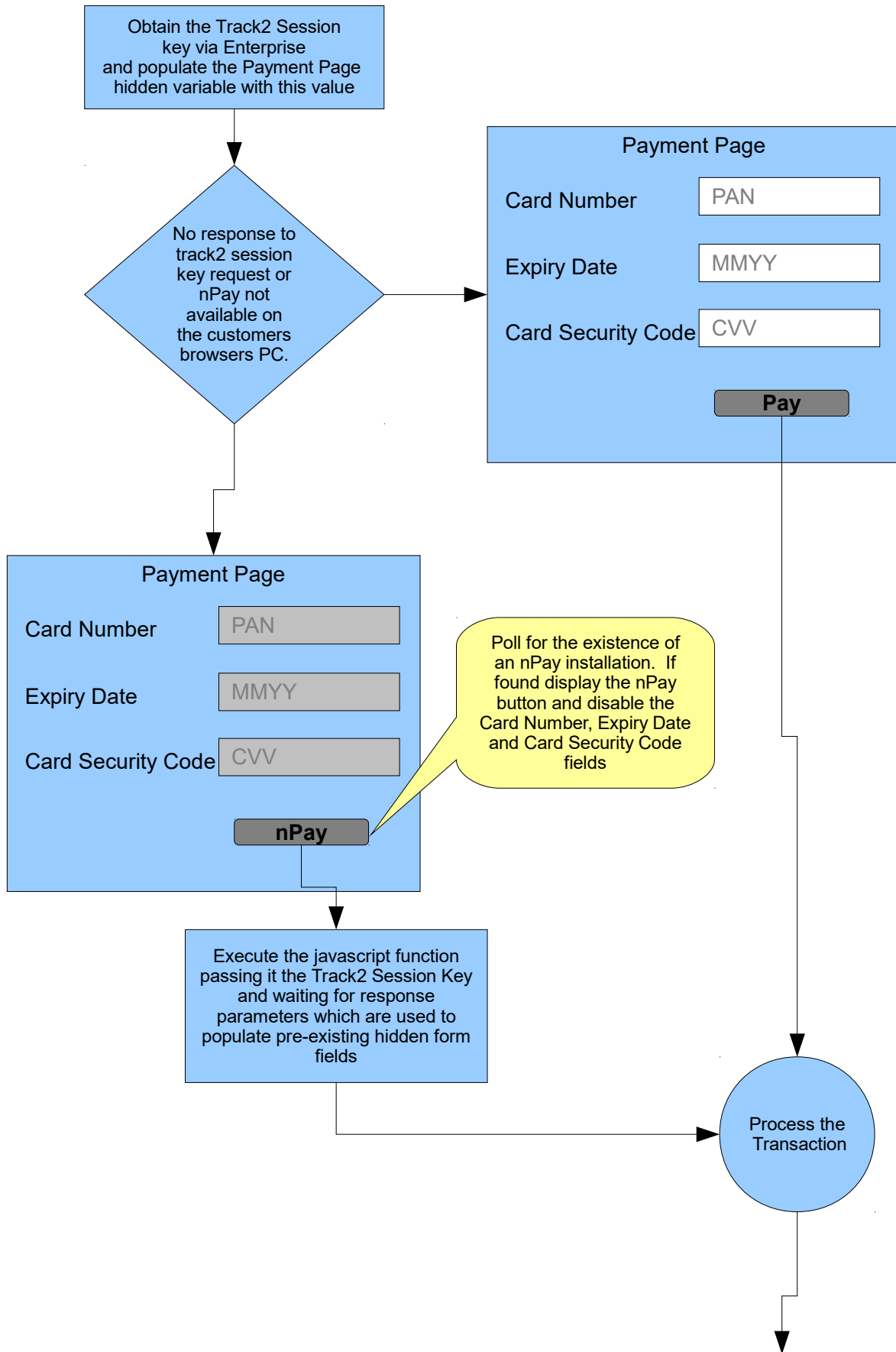
13.2.1 nPay Process Flow

The following six steps provide an explanation for the nPay process flow of a merchant website (figure 14.3.1) and an application (figure 14.3.2). Refer to each numbered:

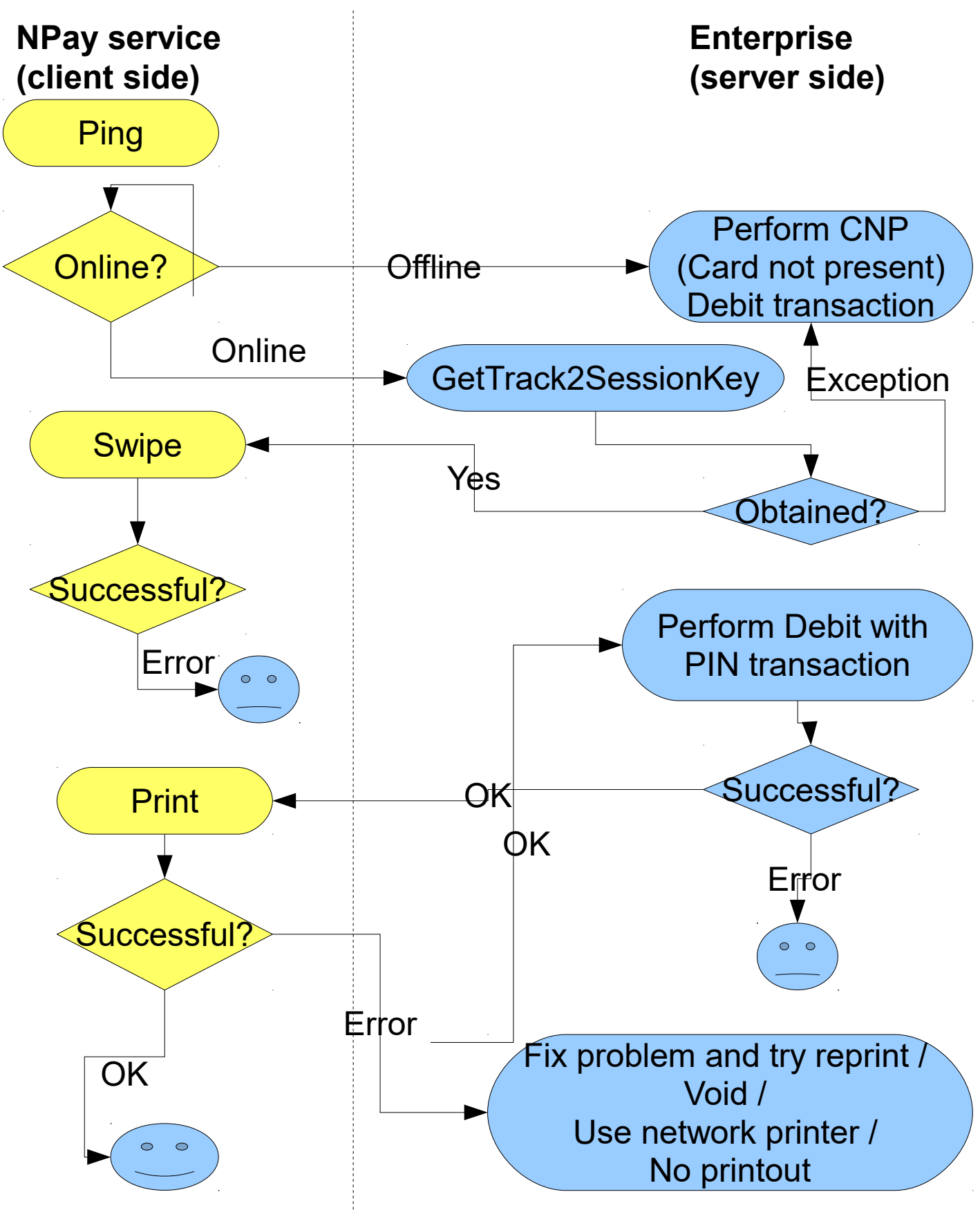
1. The merchant website obtains the Track2SessionKey using iVeri Enterprise
2. The customer interacts with the website to the point where payment is required.
3. The website/application determines if an nPay device is present by executing a 'ping' command and if present, request the customer to swipe their card by executing a 'swipe' command.
4. The nPay service returns a Json string to the merchant website, and populates form values with the card information.
5. The website/application executes an Enterprise transaction using the information populated by the nPay card swipe.
6. A transaction result is returned by the iVeri Gateway and the Merchant website displays a transaction result screen.

Npay Website Process Flow

(Figure 14.3.1)



Npay Application Process Flow
(Figure 14.3.2)



13.3 *Integration with the iVeri Gateway*

This section discusses the iVeri Enterprise commands that are relevant for NPay integration. For a complete guide to integrating nPay please refer to the nPay Developers guide shipped with nPay.

13.3.1 **Get Track2 Session Key**

In order to facilitate the mandatory Track2 encryption with the NPay Swipe command, the Track2 Session Key must be obtained via the `getTrack2SessionKey` method. This method communicates with the iVeri Gateway when necessary, and uses a caching algorithm when not necessary. Therefore this method must be called either:

- before every NPay Swipe command, or
- whenever a Swipe command has not been called within the last 24 hours

13.3.2 **Perform a Debit or Balance Enquiry**

After the NPay Swipe command is completed successfully, a list of output parameters should be available, many of which must be passed to the Enterprise `setTag` method, namely:

- PAN
- Track2
- DeviceMake
- DeviceSerialNumber

If the `PINBlock` parameter is available, it indicates that PIN based card was used, and therefore that the following Enterprise tags must be set:

- `AccountType` (optional)
- `PINBlock`
- `KeySerialNumber`

13.3.2.1 **Debit**

`CurrentBalance` and `AvailableBalance` may or may not be returned when performing a “Debit with PIN”, as it is dependent on the rules of the Acquirer.

If there is a problem after performing a successful Debit, then consider voiding the transaction (section 9.2.1). The only time the balance of a PIN based card can be increased using iVeri is by Voiding a previous transaction.

13.3.2.2 **Balance Enquiry**

Obtain the balance of the PIN based account in the currency of the account.

Using iVeri Enterprise, this can be prepared using the following syntax:

```
enterprise.prepare("Enquiry", "Balance", applicationID, mode);
```

“Balance Enquiry” returns the `CurrentBalance` and `AvailableBalance` in the currency of the cardholder, which may be different to the currency of the merchant. `CurrentBalance` and `AvailableBalance` may or may not be returned when performing a “Debit with PIN”, as it is dependent on the rules of the Acquirer.

14 payD

payD transactions allow the customer to make a payment using their debit card + pin, without a POS device present (i.e. web site purchases), by using their mobile phones.

14.1 payD Process

1. Customer selects payD as payment method for the transaction. (The payment selection process is provided by the merchant)
 1. Customer enters their mobile number instead of their card details. (Card details are not present in the V_XML)
 2. PanFormat is set to MSISDN and the MSISDN tag set to the customer's mobile number, this will identify the transaction as a payD transaction.
2. Transaction is sent to the gateway as a normal enterprise transaction.
3. The gateway identifies the transaction as a payD transaction, and forwards the transaction to payD.
4. payD determines the registration status of the user. (see respective sections below on how the process changes)

14.2 Non Registered Users

5. payD determines the user has not been registered and the transaction fails.
6. The gateway returns Error Code 23 and Description "Card information not present" to the merchant.
7. The merchant must then, in addition to the MSISDN, prompt the user for their PAN (full card number), ExpiryDate and AccountType (savings, cheque, credit)
8. PanFormat is set to MSISDN. (PanFormat must still be set to MSISDN, to identify transaction as a payD transaction)
9. Transaction is sent to the gateway as a normal enterprise transaction.
10. The gateway identifies the transaction as a payD transaction, and forwards the transaction to payD
11. payD performs the user registration
12. *...continue as a registered user...*

14.3 Registered Users

5. The customer is prompted to enter their PIN using their mobile phone.
6. payD then forwards the transaction (along with card data) to the gateway to be completed.
7. The gateway will respond to the merchant the transaction status and a new tag MobileMoneyID.

14.4 Voiding payD transactions

1. The existing void mechanism is used to void payD transactions. In addition to the existing tags used to perform a void, the merchant must also set the MobileMoneyID tag.

15 Web Service

The iVeri web service will allow cross platform integration with the various systems provided by iVeri, using a common interface.

15.0.1 Message Formats used by the Web Service

Communication between the client and web service is via SOAP protocol. The SOAP requests and responses are shown for each method provided by the web service.

15.0.1.1 Example of SOAP Request

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<Execute xmlns="http://iveri.com/">
<validateRequest>true</validateRequest>
<protocol>V_XML</protocol>
<protocolVersion>2.0</protocolVersion>
<request>.....</request>
</Execute>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

15.0.1.2 Example of SOAP Response

```
<soap:Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<ExecuteResponse>
<ExecuteResult>.....</ExecuteResult>
</ExecuteResponse>
</soap:Body>
</soap:Envelope>
```

15.0.2 Web Service URL's

iVeri WebService uses the Gateway addresses(see section 25.2) below with the following difference:

[Gateway]/iVeriWebService/Service.asmx

example:

<https://portal.nedsecure.co.za/iVeriWebService/Service.asmx>

Service Description

Nedbank: <https://portal.nedsecure.co.za/iVeriWebService/Service.asmx?wsdl>

I&M Bank: <https://portal.host.iveri.com/iVeriWebService/Service.asmx?wsdl>

CBZ Bank: <https://portal.host.iveri.com/iVeriWebService/Service.asmx?wsdl>

ECS Bank: <https://portal.host.iveri.com/iVeriWebService/Service.asmx?wsdl>

15.1 Web Service implementation

All methods provided by the web service return a V_XML formatted response. The GetPinBlock method is the only exception. The GetPinBlock method will return a PINBlock data structure.

The V_XML request and response format is defined by the V_XML.xsd schema, which can be downloaded from https://portal.nedsecure.co.za/schemas/v_xml/4.0/v_xml.xsd

Client code must check the respective Result fields to determine if the web service method was successful or not. (Result fields are defined by the V_XML specification)

In order for a client to call the execute method the client certificate must be provided with the HTTP Request.

15.2 WebService Methods

15.2.1 string Execute(bool validateRequest, string protocol, string protocolVersion, string request)

Description	
	Execute the V_XML formatted request. The action to be performed is determined by the V_XML, as defined by the V_XML specifications. ie. Transaction (Sale, Refund etc)
Returns	
	A V_XML formatted string (The return string will need to be URL decoded)
Client Certificate	
	Is required (provided with HTTP Request)
Parameters	
ValidateRequest :boolean	true: Will validate the V_XML request against the schema definition. If there are any schema errors processing of the request will terminate and the request will not be passed on to the relevant provider. or false: Will not validate the V_XML request. The V_XML request along with any schema errors will be passed on to the relevant provider for processing. If there are schema errors processing will terminate in the provider. Set this parameter to true while testing client code and false when in a production environment.
Protocol	Currently supported protocols

:string	<ul style="list-style-type: none"> • V_XML <p>If validate is set to true, an unsuccessful response will be generate for any unsupported values.</p>
ProtocolVersion :string	<p>Currently supported protocol version</p> <ul style="list-style-type: none"> • "2.0" (V_XML) <p>If validate is set to true, an unsuccessful response will be generate for any unsupported values.</p>
Request :string	<p>A V_XML format string without any namespace declarations. Providing namespace declarations will result in an unsuccessful call.</p> <p>The request must also be URL encoded</p>
Additional	
Validation Performed	<p>The execute method will also return unsuccessful if:</p> <ul style="list-style-type: none"> • No client certificate present • Certificate ID not provided in the V_XML request • Certificate ID in the Client Certificate does not match the Certificate ID in the V_XML <p><i>Validation of the Client Certificate is always performed.</i></p>

Note: URL Encode / Decode is not necessary in .NET and PHP

Method formatted as a SOAP Request and Response

15.2.1.1 SOAP Request

```
POST /iVeriWebService/Service.asmx HTTP/1.1
Host: portal.nedsecure.co.za
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://iveri.com/Execute"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Execute xmlns="http://iveri.com/">
      <validateRequest>boolean</validateRequest>
      <protocol>string</protocol>
      <protocolVersion>string</protocolVersion>
      <request>string</request>
    </Execute>
  </soap:Body>
</soap:Envelope>
```

15.2.1.2 SOAP Response

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>  
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
  <soap:Body>  
    <ExecuteResponse xmlns="http://iveri.com/">  
      <ExecuteResult>string</ExecuteResult>  
    </ExecuteResponse>  
  </soap:Body>  
</soap:Envelope>
```

15.2.2 PinBlock GetPinBlock (string mode, string pan, string pin)

Description	
	Generate a PIN Block using the HSM (Hardware Security Module)
Returns	
	<p>PinBlock data structure containing the following fields:</p> <ul style="list-style-type: none"> • ResultStatusCode • ResultDescription • DeviceSerialNumber • DeviceMake • KeySerialNumber • PinBlock <p>ResultStatusCode see Status Codes used by V_XML ResultDescription see Descriptions used by V_XML</p>
Client Certificate	
	not required
Parameters	
mode :string	"test" "live"
pan :string	Primary Account Number
pin :string	
Additional	
Validation Performed	<ul style="list-style-type: none"> • pan length is not less than 13 chracters and not more than 19 characters • pan is also validated against the LUHN Formula • pin length is not less than 4 characters and not more than 14 characters

Method formatted as a SOAP Request and Response

15.2.2.1 SOAP Request

```
POST /iVeriWebService/Service.asmx HTTP/1.1
Host: portal.nedsecure.co.za
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://iveri.com/GetPinBlock"
```

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetPinBlock xmlns="http://iveri.com/">
      <mode>string</mode>
      <pan>string</pan>
      <pin>string</pin>
    </GetPinBlock>
  </soap:Body>
</soap:Envelope>

```

15.2.2.2 SOAP Resonse

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetPinBlockResponse xmlns="http://iveri.com/">
      <GetPinBlockResult>
        <ResultStatusCode>int</ResultStatusCode>
        <ResultDescription>string</ResultDescription>
        <DeviceSerialNumber>string</DeviceSerialNumber>
        <DeviceMake>string</DeviceMake>
        <KeySerialNumber>string</KeySerialNumber>
        <PinBlock>string</PinBlock>
      </GetPinBlockResult>
    </GetPinBlockResponse>
  </soap:Body>
</soap:Envelope>

```


15.2.3 string GenerateCertificateID (string billingDetailsID)

Description	
	Creates a new unique Certificate ID to be used when generating a new certificate signing request. Certificate status will be created. This method must be used in conjunction with the SubmitCertificateRequest method to complete request of a new certificate.
Returns	
	A V_XML formatted string containing the generated certificate ID.
Client Certificate	
	not required
Parameters	
billingDetailsID :string	Billing Group ID

Method formatted as a SOAP Request and Response

15.2.3.1 SOAP Request

POST /iVeriWebService/Service.asmx HTTP/1.1
Host: portal.nedsecure.co.za
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://iveri.com/GenerateCertificateID"

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GenerateCertificateID xmlns="http://iveri.com/">
      <billingDetailsID>string</billingDetailsID>
    </GenerateCertificateID>
  </soap:Body>
</soap:Envelope>
```

15.2.3.2 SOAP Response

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GenerateCertificateIDResponse xmlns="http://iveri.com/">
      <GenerateCertificateIDResult>string</GenerateCertificateIDResult>
    </GenerateCertificateIDResponse>
  </soap:Body>
</soap:Envelope>
```

15.2.4 string SubmitCertificateRequest (string certificateID, string certificateSigningRequest)

Description	
	Submits a PKCS10 certificate signing request. Certificate status will pending.
Returns	
	A V_XML formatted string.
Client Certificate	
	not required
Parameters	
certificateID :string	Certificate ID (GUID) used to generate the certificateSigningRequest
certificateSigningRequest :string	Base 64 encoded string containing the new certificate signing request example of a certificate signing request -----BEGIN NEW CERTIFICATE REQUEST----- MIIDGjCCAoMCAQAwwbMxCzAJBgNVBAYTAipBMRAwDgYDVQQIDAdH YXV0ZW5nMRAwINT8mTAFmAe4FdIY9RUv43F9jB YqsV/B6fQxRiSehqN eS4pmK5gIXiqmfgA/NmZqRiOhLwr45SNu6NjF6Bx56T9fAw7RKTSN9dYb6 vEtOXew2kc -----END NEW CERTIFICATE REQUEST-----

Method formatted as a SOAP Request and Response

15.2.4.1 SOAP Request

POST /iVeriWebService/Service.asmx HTTP/1.1
Host: portal.nedsecure.co.za
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://iveri.com/SumitCertificateRequest"

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SumitCertificateRequest xmlns="http://iveri.com/">
      <certificateID>string</certificateID>
      <certificateSigningRequest>string</certificateSigningRequest>
    </SumitCertificateRequest>
  </soap:Body>
```

</soap:Envelope>

15.2.4.2 SOAP Response

HTTP/1.1 200 OK

Content-Type: text/xml; charset=utf-8

Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SubmitCertificateRequestResponse xmlns="http://iveri.com/">
      <SubmitCertificateRequestResult>string</SubmitCertificateRequestResult>
    </SubmitCertificateRequestResponse>
  </soap:Body>
</soap:Envelope>
```

15.2.5 string GetCertificate(string certificateID)

Description	
	Download a base 64 encoded P7B certificate that has been issued
Returns	
	A V_XML formatted string (The return string will need to be URL decoded) containing the
Client Certificate	
	not required
Parameters	
certificateID :string	Certificate ID (GUID) used to generate the certificateSigningRequest

Method formatted as a SOAP Request and Response

15.2.5.1 SOAP Request

POST /iVeriWebService/Service.asmx HTTP/1.1
Host: portal.nedsecure.co.za
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://iveri.com/GetCertificate"

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetCertificate xmlns="http://iveri.com/">
      <certificateID>string</certificateID>
    </GetCertificate>
  </soap:Body>
</soap:Envelope>
```

15.2.5.2 SOAP Response

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
```

```
<GetCertificateResponse xmlns="http://iveri.com/">  
  <GetCertificateResult>string</GetCertificateResult>  
</GetCertificateResponse>  
</soap:Body>  
</soap:Envelope>
```

15.2.6 string RenewCertificateID (string certificateID)

Description	
	Will change the certificate status of the certificate matching the certificate ID to "created". This method must be used in conjunction with the SubmitCertificateRequest method to complete the renewal of the certificate.
Returns	
	A V_XML formatted string (The return string will need to be URL decoded)
Client Certificate	
	not required
Parameters	
certificateID :string	Certificate ID (GUID) of existing certificate.

Method formatted as a SOAP Request and Response

15.2.6.1 SOAP Request

POST /iVeriWebService/Service.asmx HTTP/1.1
Host: portal.nedsecure.co.za
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://iveri.com/RenewCertificateID"

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RenewCertificateID xmlns="http://iveri.com/">
      <billingDetailsID>string</billingDetailsID>
      <certificateID>string</certificateID>
    </RenewCertificateID>
  </soap:Body>
</soap:Envelope>
```

15.2.6.2 SOAP Response

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <RenewCertificateIDResponse xmlns="http://iveri.com/">
      <RenewCertificateIDResult>string</RenewCertificateIDResult>
    </RenewCertificateIDResponse>
  </soap:Body>
</soap:Envelope>
```


16 POS Device Integration

The following section covers the details of POS Device integration (particularly PIN based transactions) that are not covered in the previous section on NPay. This section is targeted for advanced non NPay POS Device integrations, however since much of the material is covered in the section regarding NPay, the above section is recommended prerequisite reading for this section.

PIN transactions are encrypted either using Triple DES DUKPT or Triple DES Master/Session encryption architecture.

The combination of Acquirer and type of device used determine the encryption architecture chosen.

iVeri provides facilities where PIN transactions can be process in either mode Live or Test. For security reasons, a key cannot be used in both modes Live and Test. Therefore a device is either loaded for mode Test or mode Live.

A Device that is to be used for mode Live must be injected with a key within an iVeri Trusted Centre using the appropriate encryption architecture.

A Device that is to be used for mode Test may be injected by a merchant, or within the Trusted Centre.

To change a device from mode Test to Live, device has to be reinjected with a key within an iVeri Trusted Centre.

A merchant wishing to perform a Live Debit with PIN transaction requires an approved device that has been injected appropriately by the appropriate distributor.

16.1 PINBlock encryption via Triple DES DUKPT encryption

The DUKPT PINBlock encryption process flow is the following: A Device (with a `DeviceSerialNumber` and a `DeviceMake`) is injected in a Trusted Centre with an Initial PIN Encryption Key (IPEK) and Initial Key Serial Number (IKSN). Whenever a PIN block is required from the device, a counter is incremented, which indicates a PIN Encryption key. When performing a Debit with PIN, the `PINBlock` is sent encrypted using the PIN Encryption key, along with the current `KeySerialNumber` (which indicates the device and the current counter value).

16.1.1 Key Injection for DUKPT Mode Test

There is one Test IKSN and IPEK that is public knowledge. When a Device is to be injected with the Test Device Master Key, it can be done either within the iVeri Test Loading Centre, or by the merchant. When a device is loaded with a test device master key by the merchant, then the merchant must contact their iVeri Distributor with the device information: Make, Model and Serial Number.

The Test values are

IKSN: FFFF0000030000000000

IPEK: 02B50748B58B7C4452E22E39DE560CE2

16.2 PINBlock encryption via Master/Session encryption

Master/Session keys are Double Length and are sent to the iVeri Gateway in Hexadecimal format.

The Master/Session PINBlock encryption process flow is the following: A Device (with a

DeviceSerialNumber and a DeviceMake) is injected in a Trusted Centre with a Device Master Key. A merchant periodically sends a request for a session key (GetDevicePINKey) which is returned encrypted under the Device Master Key. When performing a Debit with PIN, the PINBlock is sent encrypted using the current session key.

16.2.1 Key Injection for Master/Session Mode Test

There is one Test Device Master Key that is public knowledge. When a Device is to be injected with the Test Device Master Key, it can be done within the iVeri Test Loading Centre, or by the merchant. When a device is loaded with a test device master key by the merchant, then the merchant must contact their iVeri Distributor with the device information: Make, Model and Serial Number.

The Test Device Master Key is: 375DE602546843B68089911652E951CB
(MAC: CA40C1F2)

16.2.2 Get Device PIN Key

The command "GetDevicePINKey" (which is only relevant for PINBlock encryption via Master/Session) has the following mandatory input parameters:

- DeviceMake
- DeviceSerialNumber

Unlike other commands to the iVeri Gateway, "GetDevicePINKey" does not require the input parameter ApplicationID.

Using iVeri Enterprise in iVeri Client.net, this can be prepared using the following syntax:

```
enterprise.prepare("Security", "GetDevicePINKey", new Guid(), mode);
```

or using java:

```
enterprise.prepare("Security", "GetDevicePINKey", "", mode);
```

"GetDevicePINKey" must be called if "Debit with PIN" or "Balance Enquiry" reply with the Result code: "Device PIN Key expired" [20]. It should be called upon startup and every 24 hours, or 200 transactions thereafter.

"GetDevicePINKey" returns the following output parameters, which can be stored for later usage with "Debit with PIN" and "Balance Enquiry":

- DevicePINKey
- MACDevicePINKey

16.3 Track2 encryption via Master/Session encryption

A POS Device Integration architect has the option sending the Track2 to the iVeri Gateway in an encrypted format. This optional functionality is there as an extra security measure against someone sniffing the data between the device and the server communicating with the iVeri Gateway.

The Master/Session Track2 encryption process flow is the following: A Device (with a DeviceSerialNumber and a DeviceMake) is injected in a Trusted Centre with a Track2 Master Key. A merchant periodically sends a request for a session key (getTrack2SessionKey) which is returned encrypted under the Track2 Master Key. When performing a transaction or a balance enquiry, the Track2 is sent encrypted using the current Track2 session key.

16.3.1 Track2 Key Injection for Master/Session Mode Test

There is one Test Track2 Master Key that is public knowledge. When a Device is to be injected with the Test Track2 Master Key, it can be done either within the iVeri Test Loading Centre, or by the merchant.

The Test Track2 Master Key is: BFE60D7685A24C15BC8FFBFE137C8C86
(MAC: 2F3BC80A)

16.4 Track2 encryption via Dukpt encryption

A POS Device Integration architect has the option sending the Track2 to the iVeri Gateway in an encrypted format. This optional functionality is there as an extra security measure against someone sniffing the data between the device and the server communicating with the iVeri Gateway.

The Dukpt Track2 encryption process flow is the following: A Device (with a DeviceSerialNumber and a DeviceMake) is injected in a Trusted Centre with a Track2 IPEK (initial pin encryption key). This IPEK is different from the one injected for Dukpt PIN encryption. For this encryption, the following are mandatory input parameters together with the Track2 tag:

- DeviceSerialNumber
- DeviceMake
- Track2KeySerialNumber

16.4.1 Track2 IPEK Injection for Dukpt Mode Test

BDK: 1534C275CDB5DCE015D952AB208AC1DA (MAC: B3D647B5)

IKSN: FFFF8000030000000000

IPEK: 5EC604C6F4D5EC3223F5B1BCC2AD6DD2

16.5 PAN encryption via Dukpt encryption

A POS Device Integration architect has the option sending the PAN to the iVeri Gateway in an encrypted format. This optional functionality is there as an extra security measure against someone sniffing the data between the device and the server communicating with the iVeri Gateway.

The Dukpt PAN encryption process flow is the following: A Device (with a DeviceSerialNumber and a DeviceMake) is injected in a Trusted Centre with a Track2 IPEK (initial pin encryption key). This IPEK is different from the one injected for Dukpt PIN encryption. For this encryption, the following are mandatory input parameters together with the PAN tag:

- DeviceSerialNumber
- DeviceMake
- PANKeySerialNumber

16.6 “Debit with PIN” and “Balance Enquiry”

The merchant determines if the card is PIN based typically via a local bin list (section 16.6.3). If it is PIN based, the merchant reads in the PIN from the PED, and obtains an encrypted PINBlock.

A PIN based request is identified by the existence of a PINBlock tag. The existence of the PINBlock tag implies Track2, DeviceSerialNumber, DeviceMake and DevicePINKey is mandatory. AccountType is 'Savings' or 'Cheque' or 'Credit' or 'NotSpecified'.

16.6.1 Debit with PIN

The mandatory `Amount` tag refers to the total transaction amount. The optional `CashAmount` tag is any cash portion of the transaction. Therefore always `Amount >= CashAmount`.

Normal Withdrawal: `CashAmount = Amount`

Normal Sale : `CashAmount = 0` (or not specified)

Combined Sale and Withdrawal: `0 < CashAmount < Amount`

The only time the balance of a PIN based card can be increased using iVeri is by Voiding a previous transaction.

16.6.1 Balance Enquiry

Obtain the balance of the PIN based account in the currency of the account.

Using iVeri Enterprise, this can be prepared using the following syntax:

```
enterprise.prepare("Enquiry", "Balance", applicationID, mode);
```

16.6.2 Test environment for PIN cards

iVeri Test cards are available upon request. The Result Codes responded within the PIN based issuer simulator for Mode Test are as follows:

<i>Input</i>	<i>Result returned</i>
Void of previous transaction	Successful [0]
4242424242424242 or 5959595959595959	See below
2121212121212121	randomly: HotCard [3] or Denied [4]
5454545454545454	randomly: Unable to process the transaction [9] or Timeout waiting for response [1]
All other card numbers (eg "1111222233334444")	Invalid card number [14]

"4242424242424242" returns the following:

<i>PIN</i>	<i>Amount & CashAmount</i>	<i>Result returned</i>
54321	N/A	Denied [4] (similar to PIN tries exceeded)
12345	<code>Amount <= CurrentBalance</code>	Successful [0]
12345	<code>Amount - CashAmount > CurrentBalance</code>	Denied [4] (Insufficient funds)
12345	<code>(Amount > CurrentBalance) and (Amount - CashAmount) <= CurrentBalance</code>	Approved but cash denied Warning [21] (Approve with Partial Authentication)
Other	N/A	Incorrect PIN [19]

The Current Balance for PAN 4242424242424242 for the current merchant testing is stored in memory, and is manipulated in the following manner:

- Initial amount is: 10,000.00 (max balance)
- Reset to max balance upon service restart
- Decreased after a successful debit.
- Replenished to the max balance if the CurrentBalance goes below 10.00 (min balance).
- If a Void is received then CurrentBalance is increased. Since a Void could be received after a replenishment (or service restart), CurrentBalance is forced to be less than or equal to max balance.

16.6.3 Determining if a card is PIN based

A merchant determines a the card is PIN based by using a local BIN list in conjunction with the Track2 information read from a card.

The `BINLookup` download within the File Transfer mechanism (section 22) may be used to refresh a local BIN list.

A merchant can determine if the card is PIN based in the following manner:

1. The `BINLookup` download within the File Transfer mechanism (section 22) is used to populate

- a local BIN list. It can be refreshed daily.
- 2. The Track2 of the card is read within a card reader
- 3. The PAN is determined from the Track2 (see section 10.3)
- 4. The PAN is checked against the local BIN list.
- 5. If the initial digits of the card PAN is found on the BIN list then based on the IsPINCard attribute:

0	Prompt for PIN
1	Do not prompt for PIN
2	If Position #3 of the Track2 Service Code (see section 10.3.1) is 0 or 6 then Prompt for PIN, else Do not prompt for PIN

- 6. If there is no BIN list available (or if the initial digits of the card PAN is not found on the BIN list), then:
 - If the PAN starts with 50, 56, 57, 58 or 6 then Prompt for PIN
 - Else if Position #3 of the Track2 Service Code (see section 10.3.1) is in {0,5,6,7} then Prompt for PIN
 - Else do not prompt for PIN.

16.7 “EMV Transactions”

The introduction of smart cards based on the EMV standard (EMV: Europay, MasterCard, Visa) is progressing worldwide. More and more banks rely on secure transactions for payments. With the migration from magstripe to EMV chip, banks are able to offer their customers the utmost in security. EMV chip technology prevents unauthorized access to card information and helps fight fraud.

16.8 Coding for EMV data

The following is a code snippet in C#:

EMV Request Tags:

```
enterprise.setTag("EMV_AuthorisationRequestCryptogram", authorisationRequestCryptogram);
enterprise.setTag("EMV_ApplicationIdentifier", applicationIdentifier);
enterprise.setTag("EMV_ApplicationInterchangeProfile", applicationInterchangeProfile);
enterprise.setTag("EMV_CardSequenceNumber", cardSequenceNumber);
enterprise.setTag("EMV_ApplicationTransactionCounter", applicationTransactionCounter);
enterprise.setTag("EMV_ApplicationVersion", applicationVersion);
enterprise.setTag("EMV_CardHolderVerificationMethodResult", cardHolderVerificationMethodResult);
enterprise.setTag("EMV_CryptogramInformationData", cryptogramInformationData);
enterprise.setTag("EMV_IssuerApplicationData", issuerApplicationData);
enterprise.setTag("EMV_TerminalCapabilities", terminalCapabilities);
enterprise.setTag("EMV_TerminalType", terminalType);
enterprise.setTag("EMV_TransactionType", transactionType);
enterprise.setTag("EMV_TerminalVerificationResult", terminalVerificationResult);
enterprise.setTag("EMV_UnpredictableNumber", unpredictableNumber);
enterprise.setTag("EMV_TransactionStatusInformation", transactionStatusInformation);
```

EMV Response Tags:

```
enterprise.getTag("EMV_IssuerAuthenticationData");
enterprise.getTag("EMV_IssuerScriptTemplate1");
enterprise.getTag("EMV_IssuerScriptTemplate2");
enterprise.getTag("EMV_ResponseCode");
```

17 Procurement transactions

Procurement transactions refer to transactions that contain the line item (or order basket) details. If the line item details are sent to iVeri within a transaction, then a procurement card holder get those details from their issuing bank (for example on their monthly statement). This is of assistance in tracking business to business transactions particularly in large corporations.

Procurement transactions are currently only available within distributor Nedbank.

17.1 Coding for Procurement

The following optional input parameters are Procurement specific per transaction:

- CustomerReferenceIdentifier
- CustomerVATRegistrationNumber
- DestinationCountry
- DestinationZIPCode
- NationalTax
- NationalTaxIndicator
- OrderDate
- PurchaseIdentifier
- ShipFromZIPCode
- ShippingAmount
- ShippingTaxRate
- TransactionDiscount
- UniqueVATInvoiceReferenceNumber

The following optional input parameters are Procurement specific per Line item:

- Discount
- ItemCommodityCode
- ItemDescriptor
- ProductCode
- Quantity
- TaxRate
- Total
- UnitCost
- UnitOfMeasure

Since Line items are repeated, the advanced Enterprise methods `openElement` and `closeElement` must be used. The following is a code snippet in VB.net:

```
enterprise.setTag("ShippingTaxRate", shippingTaxRate)
enterprise.setTag("TransactionDiscount", transactionDiscount)
enterprise.setTag("UniqueVATInvoiceReferenceNumber", vatReferenceNo)
```

```
Dim dr As DataRow
For Each dr In myTable.Rows
    enterprise.openElement("LineItem")
    enterprise.setTag("ItemCommodityCode", dr("Item Code"))
    enterprise.setTag("ItemDescriptor", dr("Description"))
    enterprise.setTag("ProductCode", dr("Product Code"))
    enterprise.setTag("Quantity", dr("Quantity"))
    enterprise.setTag("UnitOfMeasure", dr("Unit Of Measure"))
    enterprise.setTag("UnitCost", dr("Unit Cost"))
    enterprise.setTag("TaxRate", dr("Tax Rate"))
```

```

enterprise.setTag("Discount", dr("Discount"))
enterprise.setTag("Total", dr("Total"))
enterprise.closeElement()

```

Next

17.2 **Advanced Fraud Screening**

The following optional input parameters are CyberSource specific per Line item:

- Discount
- ItemCommodityCode
- **ItemDescriptor**
- **ProductCode**
- **Quantity**
- TaxRate
- Total
- **UnitCost**
- UnitOfMeasure
- **PassengerFirstName**
- **PassengerLastName**
- **PassengerID**
- **PassengerStatus**
- **PassangerType**

Note: The fields in **Blue** will be used when doing Advanced Fraud Screening and is not recorded on the iVeri system.

Since Line items are repeated, the advanced Enterprise methods `openElement` and `closeElement` must be used. The following is a code snippet in VB.net:

```

enterprise.setTag("ShippingTaxRate", shippingTaxRate)
enterprise.setTag("TransactionDiscount", transactionDiscount)
enterprise.setTag("UniqueVATInvoiceReferenceNumber", vatReferenceNo)

```

```

Dim dr As DataRow
For Each dr In myTable.Rows
    enterprise.openElement("LineItem")
    enterprise.setTag("UnitCost", dr("Unit Cost"))
    enterprise.setTag("Quantity", dr("Quantity"))
    enterprise.setTag("ItemDescriptor", dr("Description"))
    enterprise.setTag("ProductCode", dr("Product Code"))
    enterprise.setTag("PassengerFirstName", dr("Passenger FirstName"))
    enterprise.setTag("PassengerLastName", dr("Passenger LastName"))
    enterprise.setTag("PassengerID", dr("Passenger ID"))
    enterprise.setTag("PassengerStatus", dr("Passenger Status"))
    enterprise.setTag("PassengerType", dr("Passenger Type"))
    enterprise.closeElement()

```

Next

17.3 **Tax Calculation**

The calculations below show how the NationalTax is calculated. Note that rounding can be up or down since the resulting NationalTax value is allowed to fall in a precision tolerance interval based on the number of line items.

17.3.1 Calculation when TransactionDiscount is zero

$$\text{NationalTax} = (\text{sum of } (\text{LineItem Total} \times \text{LineItem TaxRate}) / 10000) + (\text{ShippingAmount} \times \text{ShippingTaxRate}) / 10000$$

17.3.2 Calculation when TransactionDiscount is NOT zero

The TransactionDiscount is spread across some or all of the line items resulting in an additional discount for these line items.

Lets call the LineItem Total of the affected line items, reduced LineItem Total. Then,

$$\text{NationalTax} = (\text{sum of } (\text{reduced LineItem Total} \times \text{LineItem TaxRate}) / 10000) + (\text{ShippingAmount} \times \text{ShippingTaxRate}) / 10000.$$

For the above calculation, a maximum and minimum tax amount is obtained by sorting line items in ascending and descending order respectively according to LineItem TaxRate.

18 Fleetcard transactions

Fleet cards facilitate the tracking of costs and managing statistical information of a fleet of vehicles. A Fleet transactions refer to special processing for a Fleet card.

Fleet transactions are currently only available within distributor Nedbank.

If a fleet card is received with other distributors, then the fleet specific parameters are ignored.

18.1 Coding for Fleetcards

The following optional input parameters are Fleet specific per transaction:

- CustomerReferenceIdentifier

The following optional input parameters are Fleet specific per Line item:

- ProductCode
- Quantity
- QuantityDecimalPlaces
- UnitCost

The latest available values of the ProductCode field are obtained via the "Inventory" download command (See section 22). Only use the ProductCodes with Type='Fleet'. The list at the time of writing is given in 7.4

Since Line items are repeated, the advanced Enterprise methods `openElement` and `closeElement` must be used. The following is a code snippet in C#:

```
enterprise.setTag("CustomerReferenceIdentifier",
customerReferenceIdentifier);

foreach (DataRow dr in myTable.Rows)
{
    enterprise.openElement("LineItem");
    enterprise.setTag("ProductCode", dr["Product Code"]);
    enterprise.setTag("Quantity", dr["Quantity"]);
    enterprise.setTag("QuantityDecimalPlaces",
dr["QuantityDecimalPlaces"]);
    enterprise.setTag("UnitCost", dr["Unit Cost"]);
    enterprise.closeElement();
}
```

19 Airline addendum data

Airline addendum data is additional transaction data which appear on a card holder's statement when buying a ticket from an airline merchant who include this data in a transaction request.

Airline addendum data is currently only available within distributor Nedbank. Merchants who want to make use of this need to have the NedbankBICISO provider assigned to their Application ID's on the iVeri Gateway.

In addition, for the NedbankBICISO provider, the airline addendum data is currently only supported for follow-up debit (pre-authorized debit) transactions.

19.1 Coding for Airline addendum data

The following is a code snippet in C#:

```
enterprise.openElement("AirlineData");
enterprise.setTag("PassengerName", passengerName);
enterprise.setTag("PrimaryTicketNumber", primaryTicketNumber);
enterprise.setTag("FirstDepartureLocationCode",
firstDepartureLocationCode);
enterprise.setTag("FirstArrivalLocationCode", firstArrivalLocationCode);
enterprise.setTag("PNRNumber", pnrNumber);
enterprise.setTag("OfficeIATANumber", officeIATANumber);
enterprise.setTag("OrderNumber", orderNumber);
enterprise.setTag("PlaceOfIssue", placeOfIssue);
enterprise.setTag("DepartureDate", departureDate);
enterprise.setTag("CompleteRoute", completeRoute);
enterprise.setTag("DepartureTime", departureTime);
enterprise.setTag("JourneyType", journeyType);
enterprise.closeElement();
```

Note: The fields in **Blue** will be used when doing Advanced Fraud Screening. These fields are not recorded on the iVeri system.

20 CyberSource Fraud Management

CyberSource data is additional transaction data which iVeri Payment Technologies Ltd needs to process orders within CyberSource's fraud screening solution.

20.1 Device Fingerprinting

In order to successfully implement Device Fingerprinting, a 1-pixel image file (which cannot be seen) and two scripts need to be placed in the <body> tag of the merchant's checkout page*. This will ensure a 3-5 second window in which the code segments can complete the data collection necessary to create a fingerprint for the device making the order.

Below are the code segments for implementing Device Fingerprinting:

PNG Image

```
<p style="background:url(https://h.online-metrix.net/fp/clear.png?org_id=<org ID>&session_id=<merchant id><session ID>&m=1)"></p>
```

Flash Code

```
<object type="application/x-shockwave-flash" data="https://h.online-metrix.net/fp/ fp.swf?org_id=<org ID>&session_id=<merchant id><session ID>" width="1" height="1" id="thm_fp"> <param name="movie" value="https://h.online-metrix.net/fp/fp.swf?org_id=<org ID>&session_id=<merchant id><session ID>" /> </div></div> </object>
```

JavaScript Code

```
<script src="https://h.online-metrix.net/fp/check.js?org_id=<org ID>&session_id=<merchant id><session ID>" type="text/javascript"></script>
```

Parameter	Data Type	Description
Org ID	AN	Will be supplied by iVeri Payment Technologies Ltd to the merchant.
Session ID	AN	The session ID is a string variable (letters and numbers only) that must be unique for each merchant ID. Any string that are already generating, such as an order number or Web session ID. However, do not use the same uppercase and lowercase letters to indicate different session IDs. This is the same value that must be send to iVeri in the DeviceFingerprintID field.
Merchant id	AN	Will be supplied by iVeri Payment Technologies Ltd to the merchant.

20.2 Coding for CyberSource data

The following is a code snippet in C#:

```
enterprise.openElement ("CyberSource");  
enterprise.setTag ("DeliveryMethod", Virtual);  
enterprise.setTag ("DeviceFingerprintID", sessionId);  
enterprise.setTag ("BillTo_FirstName", billTo_FirstName);  
enterprise.setTag ("BillTo_LastName", billTo_LastName);  
enterprise.setTag ("BillTo_Street", billTo_Street);  
enterprise.setTag ("BillTo_City", billTo_City);  
enterprise.setTag ("BillTo_Country", billTo_Country);  
enterprise.setTag ("BillTo_Email", billTo_Email);  
enterprise.setTag ("BillTo_IPAddress", billTo_IPAddress);  
enterprise.closeElement ();
```

21 Advanced settings

This section documents functionality within the iVeri Gateway that is not enabled by default. Contact your local distributor should you wish to activate one of these settings for your ApplicationID.

21.1 Merchant Reference validity period

See “Duplicate Transactions” (section 9.3.2) for a discussion of this option.

21.2 Recurring transaction checking

See “Duplicate Transactions” (section 9.3.3) for a discussion of this option.

21.3 ReconReference extraction

The ReconReference extraction setting is only available for the Nedbank distributor.

A 8 digit *ReconReference* is sent from iVeri to the acquirer to uniquely identify a transaction. This number is used to query transaction information with the acquirer, and it may appear on the merchant reconciliation statement.

By default, a ReconReference is generated by the iVeri Gateway.

A merchant has the option for their ReconReference to be derived from their MerchantReference. Since a MerchantReference can be in free text format of up to 64 characters, various rules determine how the 7 digit ReconReference is derived from extracting a section of the Merchant Reference.

The configuration for activating a derived ReconReference requires:

- fixed starting position
- maximum length of up to 7 digits (first digit reserved as a control digit)
- direction: left-to-right or right-to-left.

Examples:

<i>Starting position</i>	<i>Length</i>	<i>Direction</i>	<i>MerchantReference</i>	<i>Derived ReconReference (last 7 digits)</i>
12	7	left-to-right	March01Rref10	0000010
12	7	right-to-left	100Rref01March	0000100
5	4	left-to-right	Rref1000March01	1000

21.4 Transaction Type repetition within transaction sequence

Authorisations, AuthorisationReversals, Debits and Credits are by default limited to one successful of each type per transaction sequence.

An ApplicationID can be configured to not limit the successful count of any of these within a transaction sequence. A *MerchantTrace* must be supplied for transactions using an ApplicationID with this configuration activated.

AuthorisationReversals and Debits reduce the authorisation amount available (for subsequent authorisations, authorisation reversals and debits).

Similarly Credits reduce the total amount debited.

The possible Advanced transaction sequence flow with an initial Authorisation is the following:

The possible Advanced transaction sequence flow with an initial Debit is the following:

The possible Advanced transaction sequence flow with an initial Credit is unchanged from the diagram shown within the “Sequence” section (8.5).

22 File Transfer Development

The iVeri File Transfer mechanism facilitates the automation of uploading or downloading of files. There may be files available for upload/download within the iVeri Client File Transfer mechanism that are not available via the iVeri BackOffice. Similarly, there may be files available for upload/download within iVeri BackOffice that are not available via the iVeri Client File Transfer mechanism.

22.1 Overview

The following is available to assist FileTransfer development:

- iVeri Client Developers Guide (this document)
- The iVeri Client API (which includes the Enterprise class)
 - **For iVeriClient.net:** See Start - Programs - iVeriClient - "iVeri Client API"
 - **For iVeriClient.java:** `\\jre\lib\iveri\doc\index.htm`.
- iVeri FileTransfer code samples
- Distributor website (see section 25) for updates to the documentation mentioned

22.2 FileTransfer class

The FileTransfer class has the following basic flow:

- Instantiate FileTransfer class
- Set the Gateway and the CertificateID.
These can be set within the FileTransfer constructor, or as separate methods (properties in iVeriClient.net).
- You may choose to override the (properties in iVeriClient.java) `CertificatePath`, `KeyStoreFile`, `CertificateFile`, `KeyStorePassword` and `CertificatePassword`
- You may choose to override the (properties in iVeriClient.net) `CertificatePath`, `CertificateFile`, and `CertificatePassword`
- Set the mandatory properties. eg `Command`, `ApplicationID`, `Mode`
- If appropriate set various input parameters via `setTag`
- submit the request to the iVeri Gateway: via `upload` or `download`
- check the results returned, particularly `ResultStatus` and `ResultCode`
- for downloads, use the contents of the newly downloaded file.

22.3 File Transfer Data Types

File Transfer parameters have the following data types:

Data Type	Description
A	Alpha only (A-Za-z)
Guid	Globally Unique Identifier: {[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}} (eg {8E51611F-E19A-4FF0-B229-6A69F42AAA62})
N	Numeric (Positive integer)
String	Free format string containing: Alpha, numeric, special and padding (printable ASCII)

The Node Type column corresponds to how the FileTransfer class should be used for the parameter:

Node Type	Set input parameter value
property	Use method or property of FileTransfer corresponding to the parameter
tag	<code>fileTransfer.setTag(..)</code>

22.4 File Transfer Parameters

FileTransfer Parameters					
Parameter	Node Type	Data Type	Minimum Length	Maximum Length	Description
Gateway	property	A		10	The name of the gateway connecting to. If not explicitly set, the default gateway is used.
CertificateID	property	Guid	38	38	The iVeri CertificateID installed on the server communicating with the iVeri Gateway
UserGroup	property	N		10	The UserGroup used to login to the BackOffice website. Usually the same as the BillingDetailsID
UserName	property	String		16	A User name created under the specified UserGroup
Password	property	String		32	The BackOffice password of the specified user
Command	property	A		50	The command identifying what should be done by the iVeri Gateway
					Batch Upload / Download
					HotCard Download
					BINLookup Download
					BINManagement Download
					BlackCard Download
					TransactionHistory Download
					Recon Download
					AcquirerRecon Download
					Inventory Download
ApplicationID	property	Guid	38	38	Identification of the merchant profile performing the file transfer
Mode	property	A	4	4	The mode of the corresponding ApplicationID
FileName	tag	String	0	50	The identifying file name of the original batch file uploaded
FileFormat	tag	String	0	10	The format of the batch file to download
					Fixed default
					XML
StartDateTime	tag	String	0	20	The starting datetime from which reconciliation information are required
					YYYY-MM-DD HH:MM:SS YearMonthDay HourMinuteSecond
EndDateTime	tag	String	0	20	The ending datetime to which reconciliation information are required
					YYYY-MM-DD HH:MM:SS YearMonthDay HourMinuteSecond
Acquirer	tag	A	3	32	The acquirer that settled the transactions
					Nedcor
					NedbankBICISO
					IMNarada
					DashenACI
					ChamsACI
					MSCCTranzWare
					CTLNigeria
					Nips
AcquirerCycle	tag	N	5	8	The cycle from which reconciliation information are required

MerchantUsn	tag	N	5	20	Acquirer identification for a merchant account.
-------------	-----	---	---	----	---

22.5 ***File Transfer Commands***

The Upload Commands are:

- Batch

The Download Commands are:

- Batch
- HotCard
- BINLookup
- BINManagement
- BlackCard
- TransactionHistory (with a date time interval)
- Recon (with a cycle interval for an acquirer)
- AcquirerRecon (currently only available to Nips merchants)
- Inventory (currently only available to Nedbank merchants accepting Fleet cards)

22.6 File Transfer Parameters per action

The input parameters that are relevant to each action are shown in the following table using the following key:

M	Mandatory
O	Optional
blank	not relevant

	<i>Batch Upload</i>	<i>Batch Download</i>	<i>HotCard Download</i>	<i>BINLookup Download</i>	<i>BINManagement Download</i>	<i>BlackCard Download</i>	<i>TransactionHistory Download by Date</i>	<i>Recon Download by Acquirer Cycle</i>	<i>AcquirerRecon Download</i>	<i>Inventory Download</i>
Gateway	O	O	O	O	O	O	O	O	O	O
CertificateID	M	M	M	M	M	M	M	M	M	M
UserGroup	M	M	M	M	M	M	M	M	M	M
UserName	M	M	M	M	M	M	M	M	M	M
Password	M	M	M	M	M	M	M	M	M	M
Command	M	M	M	M	M	M	M	M	M	M
ApplicationID		O			M	M	M	O		
Mode		O			M	M	M	M		
FileName		O							O	
FileFormat		O								
StartDateTime							M			
EndDateTime							M			
Acquirer								M	M	
AcquirerCycle								M		
MerchantUsn								M		

The only output parameters relevant to the File Transfer mechanism are the Result related parameters (see section 10.1).

The specifications for the content of the files relevant to each of the above commands are contained, the following documents:

- iVeri Download Files XML Specification
- iVeri Batch XML Specification
- iVeri Batch Fixed Format Specification

22.7 Automating the File Transfer process

iVeriClient is released with a command line FileTransfer utility for testing and automating the FileTransfer process. This compiled utility can be used without development skills.

To obtain launch the FileTransfer utility:

iVeriClient.NET: run "iVeri.FileTransfer.exe" from the FileTransfer subdirectory of the installation directory, eg:
C:\Program Files\iVeri\iVeriClient\FileTransfer\iVeri.FileTransfer.exe *parameters*

iVeriClient.Java: type the following:

```
java FileTransfer parameters
```

Calling the utility without parameters will display the usage parameters, which is summarized here:

```
Usage: FileTransfer [[-u] | [-d]] gateway certificateID
       command userGroup userName password
       pathFileName [name1 value1] [name2 value2]...
```

The name-value pairs correspond to the relevant parameters for the chosen action as per section 22.6.

Examples:

- FileTransfer -u *iVeri Client "gateway"* {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx} **Batch**
123456 Administrator **** "C:\MyBatch.txt"
- FileTransfer -d *iVeri Client "gateway"* {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx} **Batch**
123456 Administrator **** "C:\MyBatch.txt" FileFormat XML
- FileTransfer -d *iVeri Client "gateway"* {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx} **HotCard**
123456 Administrator **** "C:\HotCard.zip"
- FileTransfer -d *iVeri Client "gateway"* {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
BinLookup 123456 Administrator **** "C:\BinLookup.zip"
- FileTransfer -d *iVeri Client "gateway"* {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
TransactionHistory 123456 Administrator **** "C:\Transactions17to20Jan.xml"
StartDateTime "2006-01-17 09:00:00" EndDateTime "2006-01-20 00:20:00" ApplicationID
{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx} Mode Live
- FileTransfer -d *iVeri Client "gateway"* {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx} **Recon**
123456 Administrator **** "C:\Recons14198.xml" Acquirer Nedbank AcquirerCycle 14198
MerchantUsn 2160000000 ApplicationID {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx} Mode Live
- FileTransfer -d *iVeri Client "gateway"* {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
AcquirerRecon 123456 Administrator **** "Recons14198-1000.zip" Acquirer Nips
- FileTransfer -d *iVeri Client "gateway"* {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
Inventory 123456 Administrator **** "C:\Inventory.zip"

Notes:

- In the above examples, the valid *gateway*, certificateID, userGroup, userName, password, ApplicationID must be inserted as appropriate.
- The pathFileName can refer to a zipped or a non zipped file.

23 Domain Knowledge

This purpose of this sections is to give some domain knowledge to people not familiar with the payment domain.

23.1 Card Present vs. Card Not Present

- A Merchant interfaces with a card holder and submits a **Card Present** Transaction request whilst the card holder is waiting and the card holder authenticates the transaction (typically via signature or PIN entry). In the event of a dispute of the transaction, the merchant must to prove that the card holder authorized it, else merchant is liable to refund the card holder.
- A Merchant submits a **Card Not Present** Transaction according to previously arranged agreement with a card holder. The card holder is not present to authenticate the transaction. In the event of a dispute on the transaction, the merchant must to prove the card holder authorized it, else the merchant is liable for refunding the card holder.
- Fully 3D Secure transactions are Card not present transactions with the protection of a Card present transaction.

23.2 Online vs. Offline Transactions

- When a Merchant submits an **Online Transaction**, s/he expects a response while the card holder is waiting. An Online Transaction can be a Card Present or a Card Not Present transaction. The result of such a transaction needs to be responded to as soon as possible.
- When a Merchant submits a **Offline Transaction** (typically with Batch of Transactions), the merchant requires a response for the transaction within the immediate future. The time frame of such transactions is they should preferably be settled before the current cycle cut off. A Batch Transaction can only be a Card Not Present transaction.
- PIN based cards can only be processed within online transactions. (It is possible for the card holder's chip card to verify the card holder's PIN locally at the card acceptance device, using the on board cryptography of the card and the PED, however chip cards are not currently supported by iVeri).

23.2.1 Online Transactions

In an Online Transaction the transfer of goods and services is ready to take place directly after the approval. No approval means no transfer of goods and services.

A Merchant interfaces with a card holder and submits an Online Transaction request whilst the card holder is waiting for the response.

- If the transaction request is approved, there is a transfer of goods or services.
- If the transaction request is denied, then either the proposed exchange of goods and services is aborted, a parameter (e.g. amount or expiry date) is changed and the transaction retried, or a different payment channel is used.
- If the transaction experiences an error, then the transaction is retried as long as the card holder is prepared to wait. Thereafter the transaction details are assessed if either it should be done offline or the proposed transfer of goods and services be aborted.

23.2.2 Offline Transactions

PIN based cards can therefore not be included in offline transactions.

An Offline Transaction is submitted when either the transfer of goods and services have already taken place or the transfer of goods and services is not a once off transfer that takes place immediately (e.g. a repetitive / continuous service).

A Merchant submits a set of offline transactions (via iVeri Enterprise or iVeri Batch) according to previously (implied) arranged agreement with a card holder.

A Merchant interfaces with a card holder. An offline transaction is initiated after one of the following occurs:

- a repetitive transaction (e.g. monthly bill) is agreed to, either in advance or in arrears
- (Store and Forward): the transaction details are below a floor limit, pass hot card and black card checking, and therefore the merchant decides to take the risk of transferring the goods and services without guaranteed payment, since either:
 - it is outweighed by the cost of going online (time or money is not worth it), or
 - an online transaction was attempted but experienced an error.
- there was a problem when the transaction was attempted online.

If the transaction request is:

- approved, the merchant gets paid.
- denied, then if the transfer of goods and services has occurred, the merchant can consider contacting the card holder, think about revising their offline business rules, or contact a debt collector.
- in error, then the transaction is retried until it is in a known state.

24 Frequently Asked Questions (common to .NET and Java)

24.1 Problems connecting to the iVeri Gateway

Ascertain if your computer can communicate directly with the iVeri Gateway, by using telnet.

Confirm the URL of your iVeri Gateway (eg `portal.nedsecure.co.za`) by consulting section 25.2.

Open a command prompt and type:

```
telnet portal.nedsecure.co.za 443
```

- If you get an error message, it means that there is a problem with the Internet connection. Please check your connection or contact your network administrator.

Windows example:

```
c:\>telnet portal.nedsecure.co.za 443
Connecting To portal.nedsecure.co.za...
Could not open a connection to host on port 443 : Connect failed
```

Linux example:

```
#telnet portal.nedsecure.co.za 443
Trying xxx.xxx.xxx.xxx...
telnet: Unable to connect to remote host: Connection refused
```

Note: If you have a proxy server refer to section 3.6

- If you are using windows and get a blank screen, it means that you have successfully connected to the iVeri Gateway.
Linux will inform you if the connection is successful.

24.2 Request timedout when network pinging the gateway

Using iVeri Client's ClientConfig utility (option 1: Network Ping iVeri Gateway), or via a command line ping, you may get *Request timed out* error messages.

Example:

```
Choice [1]: 1
Network Ping iVeri Gateway
Gateway to network ping [xxxx] :
xxxx Gateway address: gateway.xxx.xxx
Pinging gateway.xxx.xxx [xxx.xxx.xxx.xxx] with 32 bytes of data:
Request timed out.
Request timed out.
```

This is because ICMP packets have been disabled on the some iVeri Gateways, and therefore network pings will always timeout.

iVeri Client 2.3.1 and later does not employ network pings.

Check network connectivity by one of the following approaches:

- Navigate to the gateway url via a browser (see section 25.2 for URL)
- Telnet to the gateway (see section 24.1)
- Use option 5 "Validate certificate"
- Use option 6 "ApplicationID Ping iVeri Gateway"

24.3 *execute()* or *executeAsync()* must be called before this method is allowed

Applicable Version:

Any iVeri Client (.NET or .Java)

Context:

See below

Response:

The error "execute() or executeAsync() must be called before this method is allowed" is given when the following two condition hold:

1. One of the following methods or get properties are called:
 - methods: isEOF(), getAttribute(), getTag(), getXml()
 - get properties: ResultStatus, ResultNumber, ResultCode, ResultSource, ResultDescription
2. Either:
 - a) execute() or executeAsync() has not been called, or
 - b) an exception was thrown.

If you think the problem is that one of the above methods / get properties was called before execute / executeAsync was called, then make sure that you call them only after execute / executeAsync is called.

If you think the problem is that an exception was thrown, then either:

- a) Turn exception throwing off by using ResultExceptionAction.ResultExceptionThrowingOff, or
- b) catch ResultException and query its attributes (instead of the Enterprise class) for the execution results.

25 Contact Information

25.1 Distributors

An iVeri Distributor markets the services of the iVeri Gateway and products within a locality.

South Africa specific: The Nedbank Merchant agreement allows for the acceptance of Visa and MasterCard credit cards only. A Merchant that wishes to accept Diners Club and American Express cards, will need to contact these organisations separately and enter into Merchant Agreements with them. On receipt of Merchant numbers from these organisations, the merchant must contact Nedbank Card Division and ask them to update their Nedbank Merchant profile to include Diners and American Express.

25.1.1 Nedbank South Africa

Location	South Africa	
Telephone	0860 114 966	
Web sites	http://www.iveri.co.za	http://www.nedbank.co.za
Email	Submitting Logs	logs@iveri.com
	Technical Assistance	support@iveri.com
	Non technical requests/questions (e.g. costs, agreements, product information etc)	info@iveri.com

25.1.2 Nedbank Namibia

Location	Namibia	
Web sites	http://www.iveri.co.za	http://www.nedbank.co.za
Email	Submitting Logs	logs@iveri.com
	Technical Assistance	support@iveri.com
	Non technical requests/questions (e.g. costs, agreements, product information etc)	info@namibia.nedbank.iveri.com

25.1.3 CBZ Zimbabwe

Location	Zimbabwe	
Web sites	http://www.iveri.com	https://www.cbzbank.co.zw/
Email	Submitting Logs	logs@iveri.com
	Technical Assistance	support@iveri.com
	Non technical requests/questions (e.g. costs, agreements, product information etc)	info@iveri.com

25.1.4 I&M Bank Kenya

Location	Kenya	
Web sites	http://www.iveri.com	http://www.imbank.com/
Email	Submitting Logs	logs@iveri.com
	Technical Assistance	support@iveri.com
	Non technical requests/questions (e.g. costs, agreements, product information etc)	info@iveri.com

25.2 Websites

An iVeri merchant has access to 3 iVeri websites: Gateway, Certificate and BackOffice.

Website	Description
Gateway	Used whenever the Client Configuration utility performs gateway task, and whenever the execute method is called in the Enterprise / FileTransfer class
Certificate	Used when necessary by the Client Configuration utility to download the iVeri root certificate, and by Enterprise / FileTransfer class to perform crl validation
BackOffice	Manually navigated to by a merchant (or card holder). Not used by iVeri Client

iVeri Client uses the term “gateway” to refer to the institution that a merchant has an agreement with.

25.2.1 Nedbank

Website	URL	Port
Gateway	https://portal.nedsecure.co.za	443
Certificate	http://ca.iveri.com	80
BackOffice	https://backoffice.nedsecure.co.za	443

iVeri Client “gateway”	nedbank
Root Certificate	http://ca.iveri.com/PRODRootCA.crt

25.2.2 CBZ Bank

Website	URL	Port
Gateway	https://portal.host.iveri.com	443
Certificate	http://ca.iveri.com	80
BackOffice	https://backoffice.host.iveri.com	443

iVeri Client “gateway”	host
Root Certificate	http://ca.iveri.com/PRODRootCA.crt

25.2.3 I&M Bank

Website	URL	Port
Gateway	https://portal.host.iveri.com	443
Certificate	http://ca.iveri.com	80
BackOffice	https://backoffice.host.iveri.com	443

iVeri Client “gateway”	host
Root Certificate	http://ca.iveri.com/PRODRootCA.crt

25.2.4 DNS configuration

The current IP address of a website can be determined by pinging the DNS domain name.

iVeri reserves the right to change the DNS mapping to IP address at any time, however iVeri would endeavour to notify merchants timeously of such a change.

Appendix A: V_XML Message Examples

The examples provided in this apply when using the web service interface to perform various transactions.

Because of the large number of element present in the schema defintion, these examples highlight the element that are most used to perform various tansactions. It is important to note that the elements must appear in the same order they appear in the schema definition.

For a further explanation of each individual elements, see Section 7.

The examples cover the following messages:

1. Comprehensive Request and Response.

- 1. The Request message shows all of the elements that are most commonly used. The examples that follow will use the Mandatory elements from this message necessary to perform the respective transaction covered in the example.*
- 2. The response message is a typical response received from the web service to the various request messages.*

2. Credit Card Authorisation

3. Credit Card Sale

4. Credit Card Follow-up Sale (converting a previous authorisation into a sale)

5. Debit Card Sale

6. Void by Merchant Reference

7. Void by Trace

8. Void by Follow-up

9. Sale with 3D Secure Elements

1. Comprehensive Request and Response

Request Message

```
<V_XML Version="2.0" CertificateID="A8CBF8DA-92C8-4DF0-93C4-BFA285D66346" Direction="Request">
  <Transaction ApplicationID="8B2101C2-88A2-11D4-BCCF-0000E884F861" Command="Debit" Mode="Test">
    <CardHolderAuthenticationID />
    <CardHolderAuthenticationData />
    <CashAmount />
    <OriginalMerchantTrace />
    <AccountType />
    <KeySerialNumber />
    <DeviceSerialNumber />
    <DeviceMake />
    <DeviceCycle />
    <MerchantTrace>b5181eb0-b3b6-4811-a960-0000c4a0ae63</MerchantTrace>
    <PurchaseDate />
    <PurchaseTime />
    <PurchaseIdentifier />
    <Amount>123</Amount>
    <AuthorisationCode />
    <BudgetPeriod />
    <Currency />
    <ElectronicCommerceIndicator />
    <ExpiryDate>122012</ExpiryDate>
    <MerchantReference>2010-08-18 16:43:40.275</MerchantReference>
    <Terminal />
    <TransactionIndex />
    <OriginalRequestID />
    <CardSecurityCode>...</CardSecurityCode>
    <PINBlock />
    <Track2 />
    <PAN>4242.....4242</PAN>
  </Transaction>
</V_XML>
```

Response Message

```
<V_XML Version="2.0" Direction="Response">
  <Transaction ApplicationID="{8B2101C2-88A2-11D4-BCCF-0000E884F861}" Command="Debit" Mode="Test" RequestID="{59B08CCE-ACF8-48C4-9E4C-E3134640DC2D}">
    <Result Status="0" AppServer="BOROMIR" DBServer="ARWEN" Gateway="iVeri Client gateway" />
    <MerchantTrace>b5181eb0-b3b6-4811-a960-0000c4a0ae63</MerchantTrace>
    <Amount>123</Amount>
    <AuthorisationCode>615232</AuthorisationCode>
    <Currency>ZAR</Currency>
    <ExpiryDate>122012</ExpiryDate>
    <MerchantReference>2010-08-18 16:43:40.275</MerchantReference>
    <Terminal>11111111</Terminal>
    <TransactionIndex>{0505EC40-65CA-4F81-A806-FA1F2C45BA37}</TransactionIndex>
    <MerchantName>iVeri Payment Technology</MerchantName>
    <MerchantAddress>Wierda Valley</MerchantAddress>
    <MerchantCity>Sandton</MerchantCity>
    <MerchantCountryCode>ZA</MerchantCountryCode>
    <MerchantCountry>South Africa</MerchantCountry>
    <MerchantUSN>0000000000</MerchantUSN>
    <Acquirer>Nedcor</Acquirer>
    <AcquirerReference>10081:16406407</AcquirerReference>
    <AcquirerDate>20100818</AcquirerDate>
```

```

<AcquirerTime>164033</AcquirerTime>
<DisplayAmount>R 1.23</DisplayAmount>
<BIN>4</BIN>
<Association>VISA</Association>
<CardType>Unknown CardType</CardType>
<Issuer>Citibank</Issuer>
<Jurisdiction>International</Jurisdiction>
<PANMode>Keyed</PANMode>
<ReconReference>00323841</ReconReference>
<CCNumber>4242.....4242</CCNumber>
<PAN>4242.....4242</PAN>
</Transaction>
</V_XML>

```

2. Credit Card Authorisation

Request Message

```

<V_XML Version="2.0" CertificateID="{1DE2555B-5958-487C-A882-A3A10AE7C22A}" ProductType="Enterprise"
ProductVersion="iVeriWebService" Direction="Request">
  <Transaction ApplicationID="{AF8E6E69-ADC5-4D4F-B446-43D2E1E598D3}" Command="Authorisation" Mode="Test/Live">
    <CashAmount />
    <AccountType />
    <Amount>1000</Amount>
    <AuthorisationCode />
    <BudgetPeriod />
    <Currency />
    <ExpiryDate>072019</ExpiryDate>
    <MerchantReference>348004809</MerchantReference>
    <Terminal />
    <CardSecurityCode />
    <Track2 />
    <PAN>4242424242424242</PAN>
  </Transaction>
</V_XML>

```

3. Credit Card Sale

Request Message

```

<V_XML Version="2.0" CertificateID="{issued certificate id}" Direction="Request">
  <Transaction ApplicationID="{associated application id}" Command="Debit" Mode="Test">
    <Amount>1000</Amount>
    <BudgetPeriod>0</BudgetPeriod>
    <ExpiryDate>072019</ExpiryDate>
    <MerchantReference>348004809</MerchantReference>
    <PAN>4242424242424242</PAN>
  </Transaction>
</V_XML>

```

4. Credit Card Follow-up Sale (converting a previous authorisation into a sale)

Request Message

```

<V_XML Version="2.0" CertificateID="{issued certificate id}" Direction="Request">
  <Transaction ApplicationID="{associated application id}" Command="Debit" Mode="Test">
    <Amount></Amount>
    <AuthorisationCode></AuthorisationCode>
    <MerchantReference></MerchantReference>
    <Track2></Track2>
  </Transaction>
</V_XML>

```

5. Debit Card Sale

Request Message

```
<V_XML Version="2.0" CertificateID="{issued certificate id}" Direction="Request">
  <Transaction ApplicationID="{associated application id}" Command="Debit" Mode="Test">
    <AccountType>Cheque</AccountType>
    <KeySerialNumber></KeySerialNumber>
    <DeviceSerialNumber></DeviceSerialNumber>
    <DeviceMake></DeviceMake>
    <DeviceCycle></DeviceCycle>
    <Amount>1000</Amount>
    <MerchantReference></MerchantReference>
    <PINBlock></PINBlock>
    <Track2></Track2>
  </Transaction>
</V_XML>
```

6. Void by Merchant Reference

7. Void by Trace

Request Message

```
<V_XML Version="2.0" CertificateID="{1DE2555B-5958-487C-A882-A3A10AE7C22A}" ProductType="Enterprise"
ProductVersion="iVeriWebservice" Direction="Request">
  <Transaction ApplicationID="{AF8E6E69-ADC5-4D4F-B446-43D2E1E598D3}" Command="Void" Mode="Live">
    <OriginalMerchantTrace>b5181eb0-b3b6-4811-a960-0000c4a0ae63</OriginalMerchantTrace>
  </Transaction>
</V_XML>
```

Response Message

```
<V_XML Version="2.0" Direction="Response">
  <Transaction ApplicationID="{AF8E6E69-ADC5-4D4F-B446-43D2E1E598D3}" Command="Void" Mode="Live" RequestID="{03AACDCD-5EA1-4961-B7C3-14CFC26D2B6F}">
    <Result Status="0" AppServer="BOROMIR" DBServer="ARWEN" Gateway="iVeri Client gateway" />
    <OriginalMerchantTrace>b5181eb0-b3b6-4811-a960-0000c4a0ae63</OriginalMerchantTrace>
  </Transaction>
</V_XML>
```

8. Void by Follow-up

Request Message

Response Message

9. Sale with 3D Secure Elements

Request Message

```
<V_XML Version="2.0" CertificateID="" Direction="Request">
  <Transaction ApplicationID="" Command="Void" Mode="Test">
    <CardHolderAuthenitcationID></CardHolderAuthenitcationID>
    <CardHolderAuthenticationData></CardHolderAuthenticationData>
    <Amount></Amount>
```

```
<ElectronicCommerceIndicator></ElectronicCommerceIndicator>  
<MerchantReference></MerchantReference>  
<PAN></PAN>  
</Transaction>  
</N_XML>
```

Appendix B: ACS Redirect Example Page

Authenticating Enrolled Cards

You need to redirect the customer to the URL of the ACS with an HTTP form POST that contains the PAREq, TermURL, and MD. To do so, create a Web page with hidden content:

TermUrl	Termination URL on your Web site where the card-issuing bank posts the payer authentication response (PARes) message.
MD	Merchant data that you can use to match the response to the customer's order. Although iVeri recommend that you use the RequestID , you can also use an order number. This field is required, but including a value is optional. The value, which has no meaning for the bank, is returned to you as is.

POST Form

This code has two functions: a page that receives the reply fields for the enrollment check service and a form containing the required data for the card-issuing bank. The page typically includes JavaScript (an onLoad script) that automatically posts the form. In your implementation, you would replace the variables and values by your own values.

```
<body onload="document.PAEnrollForm.submit ();">
<form id="PAEnrollForm" action="acsURL value" method="post" target="paInlineFrame">
<input type="hidden" name="PaReq" value="ThreeDSecure_PAReq value" />
<input type="hidden" name="TermUrl" value="http://myPAValidationPage.ext" />
<input type="hidden" name="MD" value="<ThreeDSecure_RequestID value>" />
</form>
</body>
```

Authentication Frame


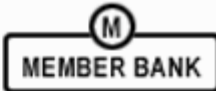
When redirected to the ACS URL, the customer's browser displays the frame that contains the card-issuing bank's password authentication dialog or the option to sign up for the program. On the page that contains the inline frame for the ACS URL, add an HTML frame large enough to accommodate either form and text to inform your customers of the process:

- HTML code:– Card issuer's authentication form: display an inline frame in a browser page that does not contain other content, such as promotional information. The frame must be large enough to show the entire 400 x 400 pixels without scrolling. You are not allowed to use a pop-up window.

```
<h2>Payer Authentication Inline Window</h2>
<iframe name="paInlineFrame" height="400px" width="400px">
</iframe>
```

Outside the frame, you must provide a brief message, for example:

Please wait while we process your request. Do not click the Back button or refresh the page. Otherwise this transaction may be interrupted.

Added Protection
Please submit your Verified by Visa password.

Merchant: Andrew's Test Account
Amount: \$42.00
Date: 04/04/2011
Card Number: *****7799
Personal Message: Password is "1234"

User Name: test1
Password:

[New User / Forgot your password?](#)

[Exit](#)

Activation form: complete Web page. Ensure that your customers can see the entire form or can scroll if necessary.




Protect your Visa Online

Your Visa has been enrolled in Verified by Visa to help protect against unauthorized use online - at no additional cost.

Whenever your card is used at participating online stores, your Issuer will ask you for your to verify that you authorize the purchase.

To activate your card, complete the form below and click Activate now. Next, you'll create your password.

Name on card:

Expiration date: / (MM/YY)

Signature Panel Code:

Last 4 digits of social security number:

Email Address:

By clicking Activate Now, you agree [Privacy & Security](#) to these [Terms of Use](#)

- **Example text:**

To increase the security of your online purchase, <business name> has partnered with <Visa, MasterCard>.

If you have signed up for Verified by <Visa, MasterCard>, please complete your bank's form to authenticate your card. The process takes about 15 seconds. If you do not currently participate in this authentication program, you can sign up now by completing your card issuer's form. If your issuing bank does not require this service, you can cancel or bypass the service. While testing your integration, verify that the frames are large enough.

PAREs Message

The card-issuing bank sends to your TermURL (<http://myPAValidationPage.ext> in this example) a POST that contains the results of the authentication in a PAREs message.

```
variable paRes = <signedPAREs replied field>
```

The base 64 string contains this information:

PaRes	Digitally signed PAREs message that contains the authentication result. Note that the field name has a lowercase a (PaRes), but the message name has an uppercase A (PAREs).
MD	Value included only if you provided one in the outgoing page.

After authentication is completed, the customer is redirected to your TermURL.

Response Messages

You need to ensure that the response messages shown to your customers are accurate and complete and that they encompass all possible scenarios for enrolled cards and for cards that are not enrolled. For example, when authentication fails, display a message such as this:

```
Authentication Failed  
Because your card issuer cannot authenticate this card, please select  
another card or form of payment to complete your purchase.
```