

**A SECURE CLIENT / SERVER INTERFACE PROTOCOL FOR THE
ELECTRICITY PREPAYMENT VENDING INDUSTRY**

by

Kennedy Pregarsen Subramoney

Submitted in partial fulfilment of the requirements for the degree

Master of Science (Electronics)

in the

Faculty of Engineering, Department of Electrical, Electronic and Computer Engineering

UNIVERSITY OF PRETORIA

August 2009

Summary

Keywords: Electricity, Prepayment, Vending, Standard Transfer Specification (STS), Protocol, Specification, Web Services, Unified Model Language (UML).

Electricity prepayment systems have been successfully implemented by South Africa's national electricity utility (Eskom) and local municipalities for more than 17 years. The prepayment vending sub-system is a critical component of prepayment systems. It provides convenient locations for customers to purchase electricity. It predominantly operates in an "offline" mode, however, electricity utilities are now opting for systems that operate in an "online" mode.

"Online" mode of operation or online vending is when a prepayment token is requested from a centralised server that is remote from the client at the actual point of sale (POS). The token is only generated by the server and transferred to the POS client, once the transaction, the POS client and the payment mechanism has been authenticated and authorised. The connection between the POS client and the server is a standard computer network channel (like Internet, direct dial-up link, X.25, GPRS, etc)

The lack of online vending system standardisation was a concern and significant risk for utilities, as they faced the problem of being locked into proprietary online vending systems. Thus the South African prepayment industry, lead by Eskom, initiated a project to develop an industry specification for online vending systems.

The first critical project task was a current state analysis of the South African prepayment industry, technology and specifications. The prepayment industry is built around the Standard Transfer Specification (STS). STS has become the de-facto industry standard to securely transfer electricity credit from a Point of Sale (POS) to the prepaid meter. STS is supported by several "offline" vending system specifications.

The current state analysis was followed by the requirements analysis phase. The requirements analysis confirmed the need for a standard interface protocol specification rather than a full systems specification. The interface specification focuses on the protocol between a vending client and vending server and does not specify the client and server

application layer functionality and performance requirements. This approach encourages innovation and competitiveness amongst client and server suppliers while ensuring interoperability between these systems.

The online vending protocol design was implemented using the web services framework and therefore appropriately named, XMLVend. The protocol development phase was an iterative process with two major releases, XMLVend 1.22 and XMLVend 2.1. XMLVend 2.1 is the current version of the protocol. XMLVend 2.1 addressed the shortcomings identified in XMLVend 1.22, updated the existing use cases and added several new use cases. It was also modelled as a unified modelling language (UML) interface or contract for prepayment vending services. Therefore, clients using the XMLVend interface are able to request services from any service provider (server) that implements the XMLVend interface. The UML modelled interface and use case message pairs were mapped to Web Service Definition Language (WSDL) and schema (XSD) definitions respectively.

XMLVend 2.1 is a secure and open web service based protocol that facilitates prepayment vending functionality between a single logical vending server and 'n' number of clients. It has become a key enabler for utilities to implement standardised, secure, interoperable and flexible online vending systems.

Opsomming

Sleutelwoorde: Elektrisiteit, Voorafbetaalde, Verkoopstele, Standaard Oordrag Spesifikasie (STS), Protokol, Spesifikasie, Webdienste, Verenigde Modellingtaal (UML)

Voorafbetaalde elektrisiteitstelsels is suksesvol deur Suid-Afrika se nasionale elektrisiteitsverskaffer (Eskom) en plaaslike munisipaliteite geïmplementeer vir meer as 17 jaar. Die Voorafbetaal verkoop-subsisteem is 'n esensiële komponent van voorafbetaal elektrisiteitstelsels. Dit laat gebruikers toe om elektrisiteit te koop by 'n verskeidenheid van verkooppunte. In die verlede het hierdie stelsels meestal bestaan as alleenstaande verkooppunte maar elektrisiteitsverskaffers is besig om hulle stelsels te verander om in 'n aanlyn modus te werk.

Aanlyn verkoop is wanneer 'n voorafbetaalkoepon versoek word vanaf 'n sentrale bediener wat vêr verwydered is van die kliënt se verkooppunt. Die koepon word slegs gegenereer deur die bediener en gestuur aan die kliënt nadat die transaksie, die kliënt self, en die betaling meganisme, gemagtig is. Die koppeling tussen verkooppuntkliënt en die bediener is 'n standaard kommunikasie kanaal, (byvoorbeeld; Internettoegang, direkte inbel skakel, X.25 en "GPRS")

Die gebrek aan 'n standaard vir aanlynverkoopstelsels was 'n bekommernis en beduidende risiko vir elektrisiteitsverskaffers, aangesien hulle 'n probleem ondervind dat hulle ingeperk sal word tot 'n eksklusiewe ontwerp vir so 'n aanlynverkoopstelsel. Dus het die Suid Afrikaanse voorafbetaal industrie, gelei deur Eskom, 'n projek begin om 'n industriespesifikasie te ontwikkel vir aanlyn verkoopstelsels.

Die eerste kritiese projek taak was 'n analise van die huidige stand van die Suid-Afrikaanse vooruitbetaling industrie, die tegnologie en spesifikasies. Die voorafbetaal sektor is gebou rondom die Standaard Oordrag Spesifikasie, bekend as "Standard Transfer Specification"

(STS). STS word algemeen aanvaar as die industrie standaard vir die oordrag van elektrisiteit krediet vanaf 'n Verkoop punt na die voorafbetaalmeter. STS word ondersteun deur verskeie alleenstaande verkoopstelsel spesifikasies.

Die analise vir die huidige status was opgevolg deur 'n studie van die vereistes vir so 'n stelsel. Die vereistes analise het die behoefte bevestig vir 'n standaard koppelvlak protokol spesifikasie, eerder as 'n nuwe spesifikasie vir 'n volledige voorafbetaalstelsel. Dit bepaal alleenlik die protokol koppelvlak tussen 'n voorafbetaalkliënt en die bediener. Dit spesifiseer nie die program vlak funksionaliteit of prestasie vereistes, vir die kliënt en bediener nie. Hierdie benadering bevorder innovasie en mededingendheid onder kliënt- en bediener-verskaffers, terwyl dit nog steeds verseker dat die stelsels wedersyds aanpasbaar bly.

Die aanlyn verkoopprotokol ontwerp is geïmplementeer met die webdienste raamwerk en staan bekend as XMLVend. Die protokol vir die ontwikkeling fase was 'n iteratiewe proses met die twee groot weergawes, XMLVend 1.22 en XMLVend 2.1. Die huidige weergawe van die protokol - XMLVend 2.1, adresseer die tekortkominge wat geïdentifiseer is met XMLVend 1.22, terwyl dit ook die bestaande gebruiksevalle opdatteer en verskeie nuwe gebruiksevalle byvoeg. Dit was ook geskoei as 'n verenigde modelleringtaal (UML) koppelvlak, of 'n kontrak, vir die voorafbetaal verkoopsdienste. Kliënte is daarom in staat om, met behulp van die XMLVend koppelvlak, dienste te versoek van enige diensverskaffer wat die XMLVend koppelvlak ondersteun. Die UML gemodelleerde koppelvlak- en gebruiksevalle- boodskappare was gemodelleer in die Web Dienste Definisie Taal (WSDL) en skema (XSD) definisies onderskeidelik.

XMLVend 2.1 is 'n sekure en oop webdienste-gebaseerde protokol wat dit moontlik maak om voorafbetaalfunksies te fasiliteer tussen 'n enkele logiese verkoopbediener en 'x' aantal kliënte. Dit het 'n sleutelrol aangeneem vir verskaffers om 'n gestandaardiseerde, veilige, wedersyds-aanpasbare en buigsame aanlyn verkoopstelsels moontlik te maak.

Acknowledgments

I would like to take this opportunity to thank the following people and organisations (in no specific order) for their support and participation in the successful completion of this project and my dissertation.

- My sincere thanks and wishes are extended to members of the XMLVend working group for their contributions and commitment to the XMLVend specification.
- I would like to thank my family, wife (Nelliandrie), twins (Dasendhran and Divashen) and baby Kanthum for your understanding, encouragement and support through the project and the dissertation compilation. You guys rock!
- I would like to thank my parents, Mr and Mrs Subramoney, whom have been my motivation and support, to always to “do and be good”. My sincere thanks are also extended to my mother in-law for all her support. Thanks and love you Mum, Dad and Umma.
- I would like to thank Eskom, all my colleagues and the Eskom Online Vending project team for their support and contributions. A special word of thanks to Mr. Jimmy ‘O Kennedy for his assistance with the Afrikaans translations and his on-going mentorship. Thanks guys!
- I would like to thank my supervisors at the University of Pretoria for their support, guidance and invaluable input.
- Finally, I would like to thank the prepaid customers and vendors that I have met while developing and implementing this specification. Thanks for your support, humility and reminding me why I love my work so much. Aluta continua!

List of Abbreviation

ACB	Automatic Clearing Bureau
AMEU	Association of Municipal Electrical Undertakens
AT	Algorithm Type
ATM	Automatic Teller Machine
BP	Basic Profile
CA	Certificate Authority
CDU	Credit Dispensing Unit
CORBA	Component Object Request Broker Architecture
CPU	Central Processing Unit
CTMS	Credit and Tariff Management System
CVS	Common Vending System
DCOM	Distributed Component Object Model
DES	Data Encryption Standard
Dk	Meter key
DOS	Denial of Service
EBSST	Electricity Basic Support Service Tariff
ECT	Electronic Communications and Transactions
EDI	Electricity Distributor Industry
EFT	Electronic Funds Transfer
EKE	Encrypted Key Exchange
ERP	Enterprise Resource Planning
ESLC	Electricity Supply Liaison Committee
FBE	Free Basic Electricity i.e. EBSST
GPRS	General Packet Radio Service

HTTP	Hyper-text Markup Protocol
HTTPS	Hyper-text Markup Protocol Secured
IEC	International Electrotechnical Commission
IEP	Integrated Electrification Programme
IETF	Internet Engineering Task Force
IT	Information Technology
IVR	Interactive Voice Response
Kek	Key Exchange Key
KLF	Key Load File
KMC	Key Management Centre
KRN	Key Revision Number
kWh	kilo watt hour
LDAP	Lightweight Directory Access Protocol
MAC	Message Authentication Code
MIG	Manufacturers Interest Group
MIS	Management Information System
MSF	Microsoft Solution Framework
MSNO	Meter Serial Number
NERSA	National Electricity Regulator of South Africa
NRS	National Rationalised Specification
OSI	Open Systems Interconnect
OVC	Online Vending Client
OVS	Online Vending Server
PAS	Publically Available Standard
PC	Personal Computer



PKC	Public key Certificate
PKI	Public Key Infrastructure
POTS	Plain old telephone service
POS	Point of Sale
RFC	Request for Comments
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SA	South Africa
SABS	South African Bureau of Standards
SET	Secure Electronic Transactions
SGC	Supply Group Code
SM	Security Module
SMS	Short Message Service
SOA	Service Orientated Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
STS	Standard Transfer Specification
STSA	Standard Transfer Specification Association
STT	Standard Token Translator
TC	Technical Committee
TCP/IP	Transmission Control Protocol over Internet Protocol
TI	Tariff Index
TID	Token Identifier
TLS	Transport Layer Security
TT	Token Technology (Token Type)
UDDI	Universal Description Discovery Interface

UML	Unified Modelling Language
URDAD	Use-case Requirements Driven Application Development
Vk	Vending key
VPN	Virtual Private Network.
WWW	World Wide Web
W3C	WWW Consortium
WSDL	Web Service Definition Language
XMLVend	Common name for the NRS Standard that defines secure communication between the Vending Server and Vending Clients
XML	eXtensible Markup Language
XSD	XML Schema Document
XSL	eXtensible Style Sheet Language



TABLE OF CONTENTS

CHAPTER 1 : INTRODUCTION.....	2
1.1 BACKGROUND.....	2
1.2 PROBLEM STATEMENT.....	3
1.3 OBJECTIVES.....	4
1.4 SCOPE OF WORK.....	4
1.5 METHODOLOGY.....	4
1.6 OUTLINE.....	5
1.7 CONTRIBUTION.....	5
CHAPTER 2 : CURRENT STATE ANALYSIS.....	7
2.1 INTRODUCTION.....	7
2.2 BASIC DESCRIPTION OF ELECTRICITY PREPAYMENT.....	7
2.3 DRIVERS FOR PREPAYMENT IN SOUTH AFRICA.....	9
2.4 KEY PREPAYMENT STAKEHOLDERS.....	11
2.4.1 Electricity Supply Liaison Committee (ESLC).....	11
2.4.2 Standard Transfer Specification Association.....	12
2.5 PREPAYMENT TECHNOLOGY AND STANDARDISATION.....	13
2.5.1 Overview of Standard Transfer Specification (STS).....	14
2.5.1.1 Background.....	14
2.5.1.2 How does STS work?.....	15
2.5.2 Overview of Offline Vending Systems.....	22
2.5.2.1 Background.....	22
2.5.2.2 Credit Dispensing Units (CDUs).....	23
2.5.2.3 System Master Stations (SMSs).....	25
2.5.2.4 Vending Information Management.....	25
2.5.2.5 Offline Vendor Management.....	27
2.6 SUMMARY.....	27
CHAPTER 3 : REQUIREMENTS ANALYSIS – THE FUTURE STATE.....	29
3.1 INTRODUCTION.....	29
3.2 THE PROJECT WORKING GROUP.....	29
3.3 UTILITY REQUIREMENTS.....	31
3.3.1 Drivers for online vending.....	33
3.3.2 Online vending challenges.....	33
3.3.3 Online vending models.....	34
3.3.4 Online vending system requirements.....	35
3.3.4.1 Functional requirements.....	36
3.3.4.2 Technical requirements.....	37
3.4 SUPPLIER REQUIREMENTS.....	39
3.4.1 Online vending system requirements.....	39
3.4.2 Online vending specification requirements.....	40
3.5 ANALYSIS OF ONLINE VENDING SPECIFICATION REQUIREMENTS..	41
3.5.1 Offline vending.....	41
3.5.2 Online vending - or is it?.....	42
3.5.3 Online vending.....	42
3.5.4 Online vending specification scope.....	44
3.5.5 Supplier proposals.....	46
3.6 SUMMARY.....	47
CHAPTER 4 : DEVELOPMENT OF XMLVEND VERSION 1.....	48



4.1	INTRODUCTION	48
4.2	WORKING GROUP DESIGN REVIEW SESSIONS.....	48
4.3	THE PROTOCOL DESIGN PROCESS	49
4.3.1	Use case definitions	50
4.3.2	Use case message exchange pattern	54
4.3.3	Use case request and response message definitions	55
4.3.4	Fault condition support.....	57
4.3.5	Message delivery reliability	58
4.3.6	Protocol security	59
4.3.6.1	XMLVend security requirements	59
4.3.6.2	XMLVend security options	62
4.3.6.3	SSL / TLS overview	63
4.3.6.4	XMLVend SSL / TLS recommendations	68
4.4	XMLVEND AND WEB SERVICES.....	69
4.4.1	Web services overview	70
4.4.1.1	eXtensible Markup Language (XML)	71
4.4.1.2	XML Schema.....	72
4.4.1.3	Simple Object Access Protocol (SOAP)	73
4.4.1.4	Web Services Description Language (WSDL).....	75
4.4.1.5	Hyper-text Transport Protocol (HTTP)	77
4.4.1.6	Web service – Interoperability (WS-I) compliance.....	82
4.4.1.7	Putting it all together	84
4.4.2	Defining XMLVend as a web service	86
4.4.2.1	The XMLVend web service protocol stack.....	86
4.4.2.2	XMLVend web services process flow	87
4.4.2.3	XMLVend fault messages	88
4.4.2.4	Defining the XMLVend Specification	89
4.4.2.5	Message compression	90
4.4.2.6	XMLVend 1 release and piloting	93
4.5	SUMMARY	96
CHAPTER 5 : DEVELOPMENT OF XMLVEND VERSION 2.....		98
5.1	INTRODUCTION	98
5.2	XMLVEND 2 REQUIREMENTS	98
5.2.1	Use cases	98
5.2.1.1	New use cases	98
5.2.1.2	Enhancement to existing use cases.....	100
5.2.2	Design requirements	100
5.3	DESIGNING THE XMLVEND 2 PROTOCOL	102
5.3.1	Modelling the XMLVend protocol.....	103
5.3.1.1	Unified Modelling Language (UML) Interfaces	103
5.3.1.2	Modelling the XMLVend interface	104
5.3.1.3	Modelling the generic input and output parameters	105
5.3.1.4	Modelling the concrete XMLVend operations.....	109
5.3.1.5	Modelling the XMLVend use case message pairs	114
5.3.1.6	Interpreting optional message parameters	117
5.3.1.7	Message delivery failure scenarios.....	118
5.3.1.8	Fault condition support.....	120
5.4	XMLVEND 2 AS A WEB SERVICE.....	121
5.4.1	Use case domain mapping to WSDL and Schema	122
5.4.2	Mapping the UML designs to Schema and WSDL	124
5.4.2.1	The contract-first development approach	124



5.4.2.2	Mapping the Message Design to XMLVend Schema	126
5.4.2.3	Mapping interface models to WSDLs	129
5.4.3	Utility specific extensions	132
5.4.4	XMLVend 2 Stabilisation.....	134
5.5	SUMMARY	134
CHAPTER 6 : XMLVEND COMPLIANCE TESTING		136
6.1	INTRODUCTION	136
6.2	THE XMLVEND TEST SUITE.....	136
6.3	XMLVEND TEST SUITE – WALK THROUGH.....	138
6.4	CUSTOMISING AND ENHANCING THE TEST SUITE.....	142
6.5	SUMMARY	142
CHAPTER 7 : CONCLUSIONS AND RECOMMENDATIONS		143
7.1	CONCLUSIONS	143
7.2	RECOMMENDATIONS	145
REFERENCES		146
ADDENDUM A. : THE ONLINE VENDING SPECIFICATION (NRS009-6-10).....		150
ADDENDUM B. : WORKING GROUP PARTICIPANT LIST.....		151

LIST OF FIGURES

Figure 2.1 – Generic Electricity Prepayment System.....	8
Figure 2.2 – Sample prepayment electricity receipt	9
Figure 2.3 – Typical Prepayment Customers	11
Figure 2.4 – STS Entity Relationships [16].....	18
Figure 2.5 – TSM210 – Security Module.....	20
Figure 2.6 – TSM410 – High-Speed Security Module (HSM)	21
Figure 2.7 –Offline Electricity Sales System (NRS 009) Context Diagram	23
Figure 2.8 – Specialised CDU	24
Figure 2.9 –Typical PC based CDU Installation	24
Figure 3.1 – Normal Vendor Model	34
Figure 3.2 – Client Server / Gateway Model.....	35
Figure 3.3 – Offline Vending	41
Figure 3.4 – Database Vending	42
Figure 3.5 – Online Vending	43
Figure 3.6 – Online vending context diagram	44
Figure 3.7 – Online Specification Context Diagram	45
Figure 4.1 – XMLVend 1.0 Use Cases.....	52
Figure 4.2 – Synchronous Request/Response Sequence Diagram	54
Figure 4.3 – Example spreadsheet format XMLVend 1 message definition	56
Figure 4.4 – Online Vending Security Context	60
Figure 4.5 – SSL / TLS over TCP /IP	65
Figure 4.6 – SSL Protocol Stack [57].....	65
Figure 4.7 – SSL Handshake Messages	67
Figure 4.8 – SSL record operation protocol	68
Figure 4.9 – XMLVend SSL certificate profile.....	69
Figure 4.10 – HTTP Traffic - XMLVend Request / Response	78
Figure 4.11 – HTTP request message format	79
Figure 4.12 – HTTP response message format	81
Figure 4.13 – Web services stack	85
Figure 4.14 – XMLVend Web services Protocol Stack	87
Figure 4.15 – Synchronous Request/Response Sequence Diagram	87
Figure 4.16 – Compressed versus Uncompressed XMLVend 1.22 Messages.....	92
Figure 4.17 – Vend Message Overhead 1.22 Protocols	92
Figure 4.18 – Eskom Western region pilot site	94
Figure 4.19 – Connectivity problems cause long queues	96
Figure 5.1 – Generic XMLVend UML Interface Diagram	104
Figure 5.2 – Class diagram template	106
Figure 5.3 – BaseReq Class Diagram	107
Figure 5.4 – BaseResp Class Diagram.....	107
Figure 5.5 – XMLVend 2 Use Cases.....	109
Figure 5.6 – XMLVend Revenue Interface	112
Figure 5.7 – XMLVend Meter Interface	112
Figure 5.8 – Purchase Credit Token Sequence Diagram.....	114
Figure 5.9 – Purchase Credit Token Request Message Model.....	115
Figure 5.10 – Purchase Credit Token Response Message Model	116
Figure 5.11 – Issue Advice Request Message	119
Figure 5.12 – Issue Advice Response Message.....	119

Figure 5.13 – XMLVend Fault Response Message.....	121
Figure 5.14 – Schema and WSDL for each Domain	122
Figure 5.15 – Schema and WSDL Development Approach.....	125
Figure 5.16 – Eskom specialisation of BusinessRuleEx	133
Figure 6.1 – XMLVend Test Suite Components.....	137
Figure 6.2 – XMLVend reference Client - Default User Interface(UI).....	138
Figure 6.3 – Purchase Credit Token Use Case Client UI	139
Figure 6.4 – Reference Client Response UI	139
Figure 6.5 – Message Interceptor	140
Figure 6.6 – XMLVend Compliance Report	141
Figure 6.7 – XMLVend Compliance Report with failures	141
Figure 6.8 – Eskom Customised XMLVend Client	142
Figure 7.1 – Specification roadmap.....	143
Figure 7.2 – 2006 African Utility Week – Eskom XMLVend stand.....	144

LIST OF TABLES

Table 2-1 – STS System Entities [16]	17
Table 2-2 – STS Identifiers [16].....	17
Table 3-1 – Visited Suppliers	39
Table 4-1 – Use Case Actors and Responsibilities.....	51
Table 4-2 – XMLVend 1 use case descriptions.....	54
Table 4-3 – Example XMLVend 1 Faults	58
Table 4-4 – Activities grouped by Web services stack layer	85
Table 4-5 – XMLVend Version 1 Updates	94
Table 5-1 – Purchase Credit Token Use Case Definition.....	114

CODE LISTINGS

Listing 4.1 – XMLVend 1 data model notation.....	56
Listing 4.2 – XMLVend 1 data model - Login Request / Response messages	57
Listing 4.3 – XMLVend 1 data model - Fault response message.....	58
Listing 4.4 – An XML representation of an FBE token issue	72
Listing 4.5 – FBE token issue - XSD	73
Listing 4.6 – FBE token issue - SOAP	75
Listing 4.7 - HTTP Request - XMLVend Request.....	80
Listing 4.8 – HTTP Response - XMLVend Response	81
Listing 4.9 – XMLVend fault message	89
Listing 5.1 – CreditVend Req complex type	126
Listing 5.2 – CreditVendReq instance	126
Listing 5.3 – AbstractCreditVendReq is a specialisation of BaseVendReq	127
Listing 5.4 – PurchaseValue complex type.....	128
Listing 5.5 – PurchaseValueCurrency is a specialisation of PurchaseValue..	128
Listing 5.6 – Currency complex type.....	129
Listing 5.7 – CurrencySymbol simple type with restriction	129
Listing 5.8 – portType Mapping	130
Listing 5.9 – operation name mapping.....	130
Listing 5.10 – message part mapping	131
Listing 5.11 – The type element specifies XSD location	131
Listing 5.12 – WSDL concrete elements.....	132
Listing 5.13 – Utility specific exceptions.....	133

CHAPTER 1 : INTRODUCTION

1.1 BACKGROUND

South African electricity utilities, led by Eskom¹, have successfully implemented electricity prepayment systems for the past 17 years. Prepayment has proven to be a sustainable option for the distribution of electricity to domestic customers. South Africa's current install base of prepayment meters is approximately four million and growing by approximately 100 000 per annum. South Africa is therefore acknowledged as a world leader in the development and deployment electricity prepayment technology.

Prepayment systems consist of three main components:

- The vending system that dispenses the prepayment tokens;
- The token transfer technology that securely transfers credit dispensed from the vending system to the prepaid meter; and
- The prepaid meter that securely connects and disconnects the prepaid meter to the electricity network based on the token value entered.

The prepayment vending system is a critical component of electricity prepayment infrastructure. It provides convenient point of sales (POSs) for customers to purchase electricity tokens. Most point of sale (POS) devices operated in an "offline" mode, referred to as offline vending. Offline POS devices generate the prepaid token using locally hosted security modules (SMs), without the need for communication with a higher-level management system [3]. SM's are the secure cryptographic devices used to generate the credit transfer tokens.

In 2001, South African electricity distribution utilities started acknowledging the benefits of online vending over offline vending. Online vending centralises the token generation and vending business logic on a central vending server. The online POS device communicates with the vending server over a network link to request a token. Some benefits of online vending are [6][7][8][9][10]:

¹ Eskom is South Africa's state-owned national electricity utility that provides electricity generation, transmission and distribution services to South Africa and the southern African region.



- It improves customer service through the introduction of new vending channels, for example, Internet Web sites, Interactive Voice Response (IVR) and Short Message System (SMS), chain store outlets and many other potential channels.
- It provides access to real-time vending data that assists utilities with credit control, customer and system data management.
- It improves data integrity and data management since data is centrally located.
- It improves system security, especially since the security modules (SM) are hosted centrally in a secure server environment. Securing the SMs provides improved protection against system fraud and a major benefit of online vending.

1.2 PROBLEM STATEMENT

Online vending is a significant improvement in the provision and management of prepayment vending services. However, the lack of an industry specification exposed utilities to significant risk. Some of the risks were:

- Being locked into a single vending system supplier for both the vending server and the POS devices.
- Being locked into supplier controlled proprietary technology.
- Vending system supplier controlling the utilities vending POS channels and more concerning the expansion of these channels.
- Suppliers controlling the cryptographic security of the system, which could result in inconsistent and possibly insecure security across implementations.
- Integration with utility backend systems could be restricted and costly.
- Finally, the proliferation of proprietary online vending systems could have a detrimental impact on the already standardised prepayment industry.

Under the auspices of the Electricity Supply Liaison Committee (ESLC), the online vending specification project was initiated. The project was mandated with the challenge of developing an industry specification for online vending [1]. The project was led by Eskom and local municipalities, supported by vending equipment suppliers and a group of technical specialists.



1.3 OBJECTIVES

The ultimate project objective was to develop an industry specification for online vending.

1.4 SCOPE OF WORK

The key project activities were:

- Analyse the current state of prepayment metering and vending technology.
- Elicit and consolidate stakeholder requirements for the online vending specification.
- Design, develop, test and publish an industry acceptable specification.

1.5 METHODOLOGY

The specification development methodology was based on the Microsoft Solutions Framework (MSF) iterative approach to application development. The MSF approach was chosen for the following reasons:

- The framework is not prescriptive but adaptable to plan, develop and deploy any information technology (IT) related project.
- The framework provides guidance on both the technology development and people related challenges. It recognises the fact that experience has shown that a successful project outcome is related more to the stakeholders and processes rather than the technology itself.
- It is based on industry best practice and incorporates Microsoft's experiences and lessons learnt in the high-tech IT industry.
- The author has had significant experience and success with the methodology on previous projects.

The project adopted MSF recommendation of a version release strategy. Core functionality is built first and more features are added in subsequent releases. This strategy has been shown to improve the team's relationship with the customer and ensures that the best ideas are reflected in the solution [4]. Using this strategy customers also become more receptive



to deferring features until later releases once they trust the team's ability to deliver. The first release is usually a pilot version, which allows it to be field tested. It is then enhanced and stabilised based on pilot feedback and then released as a production release.

1.6 OUTLINE

- CHAPTER 2 discusses the current state of prepayment in South Africa. It focuses on the credit transfer specification and offline vending systems.
- CHAPTER 3 discusses the requirements of online vending systems and how these translated to the online vending specification requirements and scope.
- CHAPTER 4 discusses the development of version 1 of the online vending protocol (XMLVend). It also analyses the design processes and the realisation of the protocol as a web service.
- CHAPTER 5 discusses the development of version 2 of the online vending protocol. It analyses the new functionality, the object orientated design process and the realisation of the protocol as a web service, using the contract first methodology.
- CHAPTER 6 discusses the development and use of the XMLVend test suite.
- CHAPTER 7 is the final chapter which provides conclusions and recommendations.

1.7 CONTRIBUTION

The following are the author's contributions to the design, development and publication of the Online Vending protocol and associated specification.

- As project leader and lead technical architect, the author developed and fostered a favourable environment of trust and mutual respect amongst all project stakeholders, with the common goal of producing an industry acceptable specification.
- Undertook a detailed survey of the current prepayment landscape including the stakeholders, technology and current challenges. This key activity was essential for the author to develop domain knowledge and an understanding of the problem domain. This survey represents a consolidation of more than 15 years of technology development and field experience in the prepayment domain.



- Undertook a detailed study into the future state requirements of the all the stakeholders. Analysed the requirements and motivated that specification be limited to an interface protocol between online client and server rather than a complete system functional specification.
- Reviewed several approaches and technologies to implement the online vending interface protocol. Using current industry trends and support from industry experts motivated that the protocol be implemented using the web services framework.
- Reviewed several approaches and technologies to secure the online vending interface protocol. Using current industry trends and support from industry experts motivated that the protocol be secured using the industry standard secure socket layer (SSL) protocol.
- Analysed and defined all the protocols use cases and use case scenarios with support from the project working group.
- Developed and modelled all the use case message pairs using unified modelling language (UML) interfaces an associated class diagrams. This was done with the guidance of UML modelling experts.
- Manually mapped the UML modelled interface and use case message pairs to Web Service Definition Language (WSDL) and schema (XSD) definitions respectively using the “contract first” approach. Ensured that the WSDL and XSD were created to be fully WS-I basic profile compliant.
- Specified the requirements of the protocol’s reference implementations and test suite and assisted with its development. The author has been responsible for continued maintenance and upgrades of the reference implementations and test suite since it initial development.
- All diagrams have been illustrated by the author, unless otherwise indicated. All photographs were taken by the author unless otherwise indicated.
- Authored the protocol’s specification document. The specification is currently in the process of being published as a national specification of the ESLC, specification document number, NRS009-6-10.

CHAPTER 2 : CURRENT STATE ANALYSIS

2.1 INTRODUCTION

This chapter provides a current state analysis of prepayment vending systems. This analysis provides an understanding on the current prepayment environment, its stakeholders, technologies and specifications. It also provides the background and foundation from which the online vending specification could be designed and developed.

The current state analysis was undertaken by the author using extensive literature reviews, formal and informal discussions with industry and technology experts.

2.2 BASIC DESCRIPTION OF ELECTRICITY PREPAYMENT

Electricity prepayment operates conceptually, in a similar manner to popular prepaid cellular services. It essentially means that customers must pay upfront for electricity before it can be consumed. Electricity consumption is controlled at the customer's home through a prepaid meter. The meter must have credit loaded before electricity can be consumed and interrupts the electricity consumption once the credit is depleted.

A generic prepayment system consists of the following components (Figure 2.1):

- Credit Dispensing Unit (CDU);
- The encrypted credit transfer token;
- The prepaid meter; and
- An optional management information system (MIS)

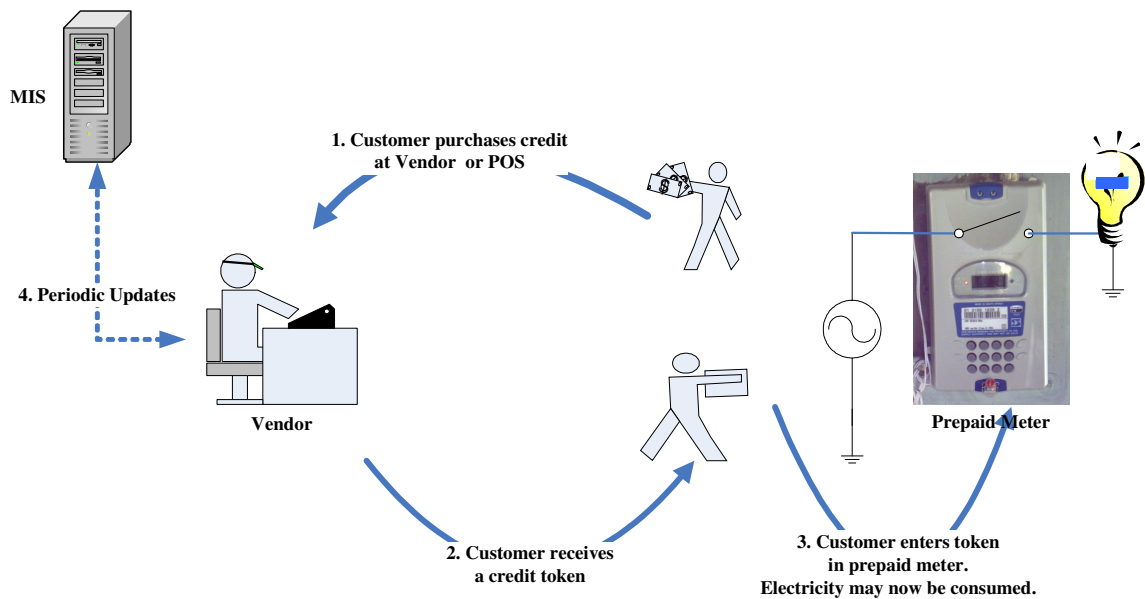


Figure 2.1 – Generic Electricity Prepayment System

A customer purchases electricity credit at the nearest electricity vendor. The vendor uses an electricity POS device, referred to as a credit dispensing unit (CDU), to generate the credit transfer token for the sale. The information carried between the CDU and the meter is usually encrypted into a fixed length value commonly referred to as a token. The token is usually printed on a receipt or encoded on a magnetic card for input into the meter. Figure 2.2, illustrates a typical receipt, the 20 digit token to be entered into the meter is indicated by the red block. The meter's credit is updated once the token is accepted. Electricity credit is usually transferred as a kilowatt hour (kWh) value to the meter. The vendor's transactions are optionally uploaded to the utilities (MIS).

CREDIT VEND - TAX INVOICE		
<u>Distributor</u>	<u>VAT Number</u>	
Eskom Online	4740101508	
<u>Date</u>		
2008/12/10 04:21:20 PM		
<u>Address</u>		
Megawatt Park, Contact Centre tel 086-003-7566		
<u>Receipt No</u>		
CIGIC2A02051146		
<u>Client ID</u>	<u>Terminal ID</u>	
6004708001493	0000000078056	
<u>Meter No</u>	<u>Tok Tech</u>	<u>Alg</u>
07029104267	01	07
<u>SGC</u>	<u>KRN</u>	<u>TI</u>
100209	1	07
<u>Energy *kWh</u>	<u>Token Amt</u>	
17	R 10.00	
<u>Description</u>		
Normal Sale		
<div style="border: 2px solid red; padding: 5px; display: inline-block;"> 3342 9613 6447 9659 3040 </div>		
Meter not registered for Free Basic Electricity. Please apply at your local office.		
VAT included at 14%		

Figure 2.2 – Sample prepayment electricity receipt

2.3 DRIVERS FOR PREPAYMENT IN SOUTH AFRICA

The “Electricity for All” programme started in 1989 and was driven by social, political and economic pressures [3]. It required mass electrification of previously underdeveloped and rural areas of South Africa. The lack of telecommunications and postal infrastructure, combined with the informal nature of many of the areas to be electrified, meant that the delivery of conventional bills was impractical. Prepayment metering systems did not rely on such prerequisites and became a viable alternative for electrification.



Prepayment metering systems addressed the following electrification issues [1][2]:

- Many customers had to be supported by the smallest amount of Eskom personnel. The system therefore had to operate with a low level of management and maintenance. The standard billed system required a lot of day-to-day management to process accounts and maintain connections and disconnections.
- Many of the areas where typical electrification customers reside had almost no infrastructure. There were no formal addresses for customers, many did not have permanent jobs or bank accounts and there were no or limited postal services in those areas (Figure 2.3). These were significant limitations that prevented a typical utility billed system to operate effectively.
- Many customers were illiterate and did not understand (or had the budget) to pay for the fixed charges or bills that arrive only after the electricity had been consumed.
- Difficulties experienced with customers withholding payment for electricity.
- Difficult or very remote access to meters for meter reading.
- Deposit management problems.
- Customers do not understand, trust or could always afford the fixed monthly portion of a conventional account.
- The need to charge large up-front connection fees.

South African utilities have learnt that prepayment systems must be installed only after consultation with and agreement by the community to be electrified. Experience has also shown that prepayment enjoys better customer acceptance if it is not promoted as a solution for theft or to punish customers. Rather customer benefits of prepayment systems must be promoted. Some customer benefits are:

- There are no fixed monthly charges or reconnection fees.
- There is a continuous display of the available credit which allows the customer to budget and it eliminates surprises like a large account at the end of the month.
- Usually money tendered for the electricity purchases is not used to also pay for other services like refuse removal.
- The customer has control over electricity expenses and is therefore able to budget and control consumption.



Figure 2.3 – Typical Prepayment Customers

As customers become more sophisticated in their electricity use, their focus turns toward its availability. Therefore, the availability and convenient access to prepayment electricity POS's become very important to the customer.

The current government's vision is to provide all South Africans access to electricity by 2012. This is being achieved through the Integrated Electrification Programme (IEP) under the auspices of the Department of Minerals and Energy. By mid-2007, since the inception of the electrification programme in 1991, 3 469 650 homes have been electrified [58].

Prepayment was initially perceived as a low maintenance technical solution in an era of widespread non-payment for services. However, life-cycle costing studies have proven that prepayment is a more cost effective option of system operation than billed systems [2]. It is now seen as a sustainable technology option to achieve the 2012 vision.

2.4 KEY PREPAYMENT STAKEHOLDERS

2.4.1 Electricity Supply Liaison Committee (ESLC)

The ESLC comprises South African supply authorities and related stakeholders. It comprises members from the Association of Electrical Municipal Undertakings (AMEU), Eskom divisions (Distribution, Transmission and Resources and Strategy), larger municipal electricity distribution utilities and Standards SA. The National Electricity



Regulator of South Africa (NERSA) and the Electricity Distribution Industry (EDI) Holdings Company observe the activities of the ESLC [18].

The Electricity Supply Liaison Committee (ESLC) has been directing the National Rationalised Specifications (NRS) programme for over 16 years. The NRS programme produces rationalised user specifications that allow member organisations to define common performance, interface, security and quality requirements for electricity supply related equipment and systems. That is, NRS specifications are a set of industry specifications² for use by the South African electricity supply authorities.

The NRS009 series is a set of specifications for South African electricity sales systems. They specify standardised offline vending devices and interfaces. They are prepared on behalf of the Electricity Supply Liaison Committee (ESLC) and managed by Eskom.

2.4.2 Standard Transfer Specification Association

Standard Transfer Specification (STS) defines the credit transfer specification between the vending system and the prepaid meter. STS is managed and maintained by the Standard Transfer Specification Association (STSA). The STSA was formed in 1997 to maintain the necessary infrastructure, promote the technology and further develop the standard to meet emerging international demands for additional functionality. It is a non-profit association, comprising members from meter and vending system manufacturers, as well as utilities. STSA objectives include:

- STS standardisation and enhancement;
- Develop technical guides;
- Provide accreditation testing;
- Maintain an approved accreditors register;
- Maintain an approved list of STS-compliant equipment; and
- Provide key management and key security and certification.

² NRS specifications are however not a standard as contemplated in the Standards Act, 1993 (Act 29 of 1993).



2.5 PREPAYMENT TECHNOLOGY AND STANDARDISATION

The initial prepayment systems utilised proprietary technology. However, by 1991 utilities started experiencing significant problems with proprietary prepayments systems, such as [3]:

- Complex logistics and management of holding spares for several proprietary systems;
- Locked into single supplier's equipment;
- Difficulties with the management of points-of-sale and customer databases;
- Manufacturers adopted different operating philosophies, making the training of staff difficult and complex; and
- The key realisation was that the systems security was in the hands of suppliers. In other words the suppliers literally controlled the cash registers of the utilities. Security audits also identified that the degree of security provided by each system varied.

The first step was to develop an industry specification for the prepaid meter that addressed functional, performance and reliability requirements. These were initially published as NRS009 part 1, 2 and 3 documents and later replaced by a national standard, SABS 1524-1. The meter standard, SABS 1524-1:1993, does not impact the online vending specification and will not be discussed any further.

Thereafter, utilities turned their priority to developing a viable, secure standard token transfer interface from CDUs to meters. The main requirement was to standardise the data format of the instructions (token) to the meter and the encryption algorithm. This would enable a token produced by a standard CDU to work in any standard meter. The standardisation of the token between the CDU and meter heralded a new era for prepayment. The additional requirement for "proprietary compatibility" option was included to allow continued support for already installed proprietary meters.

In 1993, Conlog, a major prepayment system supplier, was contracted by Eskom to develop such an industry specification. This contract resulted in the first version of the Standard Transfer Specification (STS). The STS protocol was released into the public domain as part of the NRS 009 series of specifications.



In conjunction with the STS development, Eskom further defined and developed the Common Vending System (CVS) to provide a total electricity system capable of supporting a widespread deployment of vending systems and meters sourced from a number of different manufacturers. The CVS was also later adopted, with minimal changes, as part of NRS 009 series.

2.5.1 Overview of Standard Transfer Specification (STS)

2.5.1.1 Background

STS is an open system for one-way prepayment meters. Although several proprietary transfer technologies exist, STS has become the de-facto industry specification for one-way prepayment electricity token transfer technology. To enhance its credibility internationally, STS has been offered, through IEC TC13 for publication as a publicly available specification (PAS). Publication as an IEC PAS is a mechanism for standards that already have a level of acceptance in a significant segment of industry to be endorsed by the IEC.

To date there are more than 4 million STS meters installed in the field. The specification has been stable for 10 years and has adopted by approximately 400 utilities in 25 countries.

Some of the key security requirements of STS were to prevent:

- Fraudulent generation of tokens from ‘hit and miss’ attempts at entering the correct number;
- Fraudulent generation of tokens from legitimate vending stations outside of the utility's area;
- Fraudulent use of tokens which have already been used; and
- Tampering of legitimate tokens, for example, to change the encoded value.

In order to achieve the above, the standard defines the following mechanisms:

- The use of advanced encryption techniques, which are at all times hidden from the customer;

- The use of very secure key management procedures, including the manner in which keys are generated and transported; and
- The required functionality at both the vending station and the meter is tested and verified by a rigorous compliance process.

It was critical that the key management be provided by a trusted, neutral third party. Eskom, supported by the broader industry, has built and still manages the first STS prepayment key management centre (KMC) on behalf of the STSA. This KMC is currently the only approved STS KMC and services all the current users and suppliers of STS compliant systems both local and international.

2.5.1.2 How does STS work?

This section provides a high level analysis of key STS concepts and processes. This analysis assists in understanding the relationship between STS and the online vending specification.

STS defines a secure message protocol that allows information to be carried between the CDU and meter. The information carried between the CDU and the meter is encrypted by the protocol into a fixed length value commonly referred to as a token. More than one token may be created depending on the amount of information being carried between the CDU and meter. STS caters for several message types such as credit, configuration, display and test instructions. It further specifies devices and codes of practice that allows for the secure management (generation, storage, retrieval and transportation) of cryptographic keys used within the system.

STS compliant prepayment systems consist of the following devices, entities and concepts (Table 2-1).

Entity	Description
Key management centre (KMC)	A trusted and physically secure domain that generates and manages the storage and distribution of cryptographic keys.
Secure module (SM)	A device which is physically secure from tampering, into which cryptographic keys emanating from the KMC are stored. The SM is installed in CDUs and used to generate tokens.

Meter	The prepayment meter consists of the following basic components, token reader, accounting register, measurement element and load switch.
Credit dispensing unit (CDU)	The CDU is a point of sale device that is used in conjunction with the SM to generate and dispense STS tokens. It also stores transactions, maintains customer data and tariff information.
System master station (SMS)	The equipment where all relevant system information is maintained. It is a higher level management information system (MIS) that consolidates all transactions from CDUs that it manages.
Meter token or token	An encrypted fixed length value used to transport messages from the CDU to the meter. Currently STS defines two token technologies as carriers of the token, that is, a magnetic card and a printed numeric receipt [21][22].
Meter card	A standard format magnetic card that carries all relevant customer and meter information for a CDU to generate a token, namely meter and tariff data. The customer uses the meter card to identify his meter details to a CDU. The meter card specification is defined in [27].
Meter communications port	A serial port accessible on the rear of the meter. It uses the IEC1107 protocol and supports all token transfer functions.
Standard Token Translator (STT)	A device that is able to accept an STS token via an RS232 port and translate it into a proprietary format. It is used where backward compatibility with legacy systems is required. The STT is defined in [30].
Utility	The utility supply company that contracts with the customer to supply a utility like electricity.
Supply groups	Sub groups within the domain of the utility supply company's supply or distribution networks. Grouping may be based on geography, tariff, load, etc. Generally used to confine consumers to purchase tokens within the group or to confine vendors to vend only within a certain group. In a scenario, where a vendor may vend to multiple utility customers, the supply group can be used to determine which utility should receive the revenue for sales.
Key Load File (KLF)	This is file used to transport vending keys from the KMC to a SM.
Key exchange key (KEK)	A secret 112 bit Data Encryption Standard (DES) cipher key shared between the KMC and the SM. It is used to encrypt vending keys for transportation via the KLF and used to decrypt the vending keys when loading them into the SM.
Vending key (Vk)	A secret 56 bit single DES cipher key that is linked to a supply group of a utility. It is generated in the KMC. The vending key is stored in SM's that are authorised to vend to the SGC. It is also used to generate meter keys.

Meter key (Dk)	The meter key is generated from the appropriate vending key. It is a secret 64 bit DES based cipher key shared between the SM and the meter. It is used to encrypt and decrypt messages exchanged between the SM and meter. The meter key is used to generate the fixed length transfer tokens.
Customer	The legal entity that contracts with the utility supply company for the supply and delivery of a utility like electricity.
Vendor	An agent or employee of the utility supply company that sells prepaid tokens to customers on its behalf.

Table 2-1 – STS System Entities [16]

The following identifiers are used to identify the entities within an STS prepayment system (Table 2-2).

Identifier	Description
Supply Group Code (SGC)	A number allocated to a utilities supply group and registered with the KMC. It identifies a sub group within the supply or distribution domain of the utility supply company. Generally a large utility may have several SGCs. It is managed by the KMC.
Meter number (MSNO)	A unique number comprising a manufacturer number and a meter serial number that identifies a particular device within the entire STS system internationally. It is managed by the meter manufacturer.
Manufacturer number	A unique number allocated to a manufacturer of meters or secure modules. It is managed by the STSA.
Token Identifier (TID)	A 24 bit number that represents the number of minutes that have elapsed since midnight 1 st January 1993 to the actual time of token generation. It is thus time sequenced and is coded onto the token. It is used by the meter to prevent the re-use of previously used tokens.
Tariff index (TI)	A number associated with a particular tariff allocated to a particular customer. The maintenance and content of the tariff tables is loaded into the CDUs. They are the responsibility of the utility and not covered in the STS.
Key revision number (KRN)	A number associated with a particular revision of vending key (Vk) and therefore meter key (Dk).
Key type	A number associated with a vending key and meter key that depicts the type of key. STS defines four types: <ul style="list-style-type: none"> • manufacturer unique • manufacturer default • supplier unique • supplier common

Table 2-2 – STS Identifiers [16]

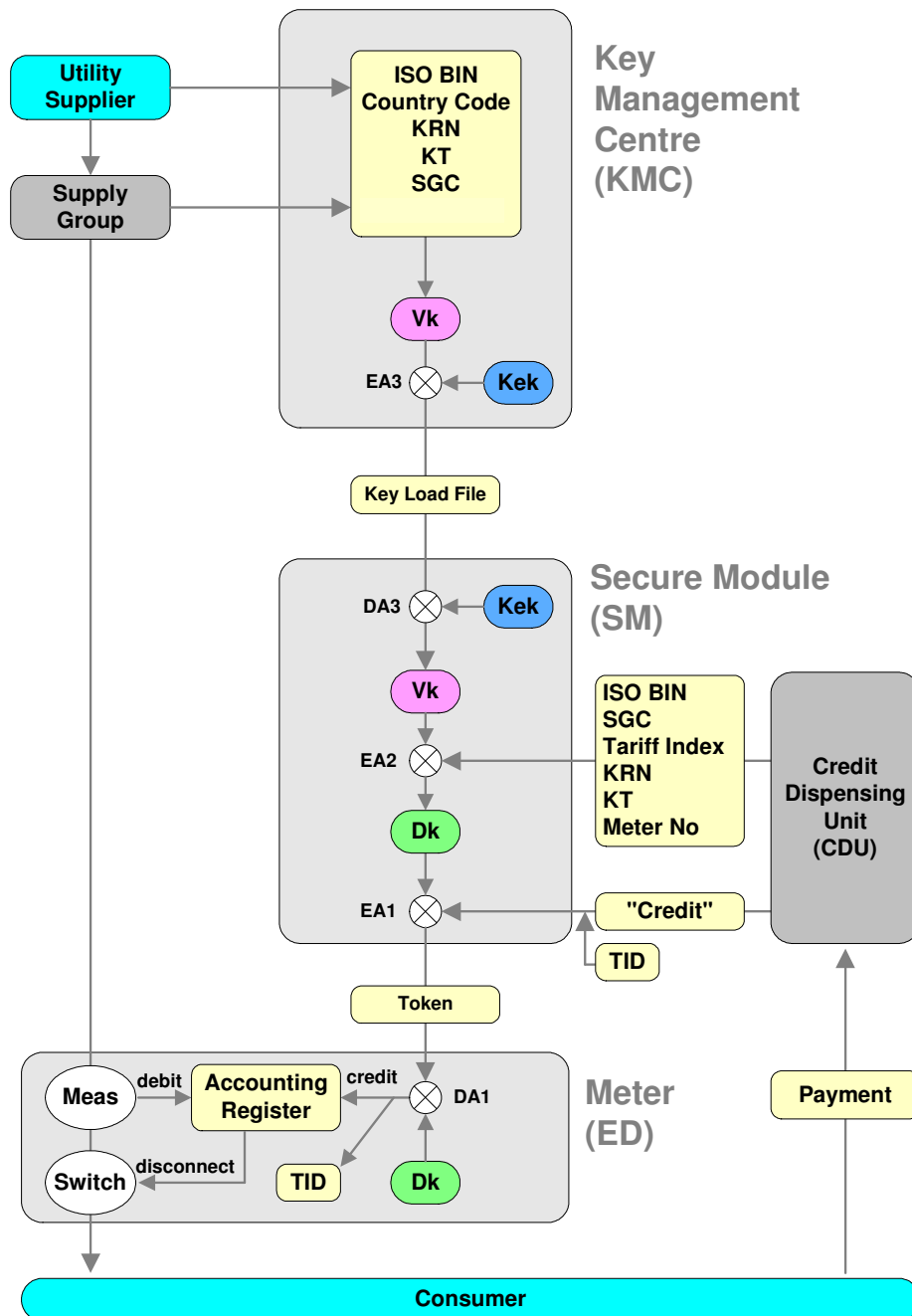


Figure 2.4 – STS Entity Relationships [16]

Figure 2.4 illustrates the relationship between the devices, entities and identifiers in an STS prepayment system.

- A utility may divide its distribution area into one or more sub areas, commonly referred to as supply groups. The supply groups may be geographically or commercially based. In each supply group prepayment meters are installed.
- The utility requests the KMC to allocate a supply group code (SGC) for each of its supply groups. The KMC also generates a vending key (Vk) for each of its supply



groups. The vending keys are generated and stored within the secure environment of the KMC.

- The utility supplies its SM's to the KMC for registration and initialisation. All SM's must be registered and initialised in the KMC. The SM provides secure storage for utilities vending keys and the key exchange key (KEK). The SM also provides a secure environment in which all encryption and decryption functions are executed during the process of key loading and token generation.
- The KMC generates key load files (KLF's) for each SM. The KLF securely transports the utilities vending keys (Vk) from the KMC to a specific SM. The contents of the KLF is first encrypted using encryption algorithm (EA3) with the secret key exchange key (KEK) shared by the KMC and the SM. The KLF is decrypted using decryption algorithm (DA3). The vending keys are then loaded into the SM.
- The SM and the prepayment meter share a secret meter key (Dk), which is stored in the meter during a special initialisation process during manufacture and subsequently in the field. Dk is generated within the SM using encryption algorithm (EA2) with the utilities Vk and the following meter configuration identifiers, SGC, KRN, TI, MSNO and some other data parameters. A change to any of the meter configuration identifiers will require a new Dk to be transferred to the meter. This is achieved by means of a special token pair, called key change tokens. Key change tokens are generated by CDUs using the meter's current configuration identifiers and the meter's new identifiers.
- Meter manufacturers supply meters to the utility in two configurations based on the utilities requirements. In the first configuration, the meters are initialised to the default SGC. Default supply group is allocated to meters whose eventual supply code is unknown at the time of manufacture. The utility would then have key change the meter from the default SGC to the utilities SGC before installing the meter in the field. In the second configuration, the meters are shipped pre-initialised with the utilities specified SGC's. These meters can be installed directly in the field.
- SM's are now able to send encrypted messages to the meter using encryption algorithm (EA1) with Dk and the meter is able to receive and decrypt these messages using decryption algorithm (DA1) with Dk.
- The information required for a customer to purchase tokens at a CDU are obtained from one of three sources, namely from the customer's meter card, from an old token and finally from the customer / meter database maintained on the CDU. The tendered amount is translated to a kWh value using the tariff function and used to create a

“credit” message. The “credit” message is encrypted using EA1 with Dk and encoded onto a magnetic card or printed on a receipt depending on the meter’s token technology. At the meter, the customer inserts the magnetic token into the reader or enters the numeric number by means of a key pad. The meter decrypts the token using DA1 with Dk. If the token is valid, the meter reads the “credit” message and adds the message’s kWh value to the meter’s accounting register.

- To prevent a token from being used more than once on the same meter, a token identifier (TID) code is added to the “credit” message on the token prior to encryption in the SM. STS defines this TID to be a number representing the number of minutes that have elapsed since midnight 1st January 1993 based on the actual time of token generation at the CDU. In this way TID codes are sequenced in time. The meter stores the TIDs of the “youngest” 50 valid tokens that it had accepted and compares these to any token being presented to the meter. The meter is therefore able to discern whether the token had already been used or not. In addition the magnetic card tokens are also magnetically erased.

Figure 2.5 is a SM implementation named TSM210. The TSM210 has undergone stringent security and vulnerability testing and is currently only STS approved SM implementation. The TSM210 stores a maximum of 79 vending keys. The TSM210 is a low speed device and can achieve a maximum of 2 transactions a second. It is suited for a low transaction volume environment such as offline CDU.



Figure 2.5 – TSM210 – Security Module

Figure 2.6 is the next generation SM implementation named TSM410. The TSM410 has undergone stringent security and vulnerability testing and is in the process of being

approved by the STSA. The TSM410 is a high speed device capable of achieving 50 transactions per second. It is more suited to a high transaction volume environment such as an online vending server. The TSM410 is also able to store a maximum of 999 vending keys.



Figure 2.6 – TSM410 – High-Speed Security Module (HSM)

The security of a utilities prepayment system is largely determined by the secure generation, storage and distribution of its vending keys. These services are provided and maintained by the KMC. The SMs ultimately securely store the vending keys, which are used for token generation in CDUs. Therefore, the system's security is also determined by the utilities capability to manage and control its SM's since [17]:

- Vk authorises credit transfer to customer;
- Anyone in possession of Vk can transfer credit;
- A loaded SM is a credit transfer machine; and
- A “lost” or “unused” SM is a money printer.

The absence of strict SM controls by utilities, specifically for SMs that have been loaded with vending keys, places the utilities prepayments system and therefore its revenue at significant risk of abuse. The risks associated with uncontrolled SMs loaded with vending keys are similar to risks associated with an automated teller machine (ATM) bank card with its personal identification number (PIN) written on the card. That is, anyone with access to your card can make withdrawals from your account without your knowledge. Further, by the time you discover it, there is nothing you can do to get your money back. Similarly, once there is unauthorised access to loaded SMs, criminals with access to CDU



software can generate and sell tokens without the knowledge of the utility. The abuse can only be stopped once the abused SMs are brought back under the utilities control.

2.5.2 Overview of Offline Vending Systems

2.5.2.1 Background

Eskom defined and developed the Common Vending System (CVS) to provide a total prepayment electricity system capable of supporting a widespread deployment of vending systems and meters. The main aim of the standardised system was to source system components from different manufacturers while ensuring interoperability.

The system functionality provides utilities with capabilities of controlling electricity sales to its customers. The system enables customers to purchase electricity from places and times convenient to both the customer and the utility. The system also provides for accounting, data collection and processing to aid administration and provides safeguards against fraud.

A typical electricity sales system (Figure 2.7) consists of CDUs linked to one or more System Master Stations (SMS) and an optional link to higher level computer system. In most utilities such a higher computer system is usually the utilities customer and billing system.

The system operates in an offline mode since tokens are vended and transaction concluded at the CDU. There is no need for the CDU to communicate with any higher level management system at the time of the transaction.

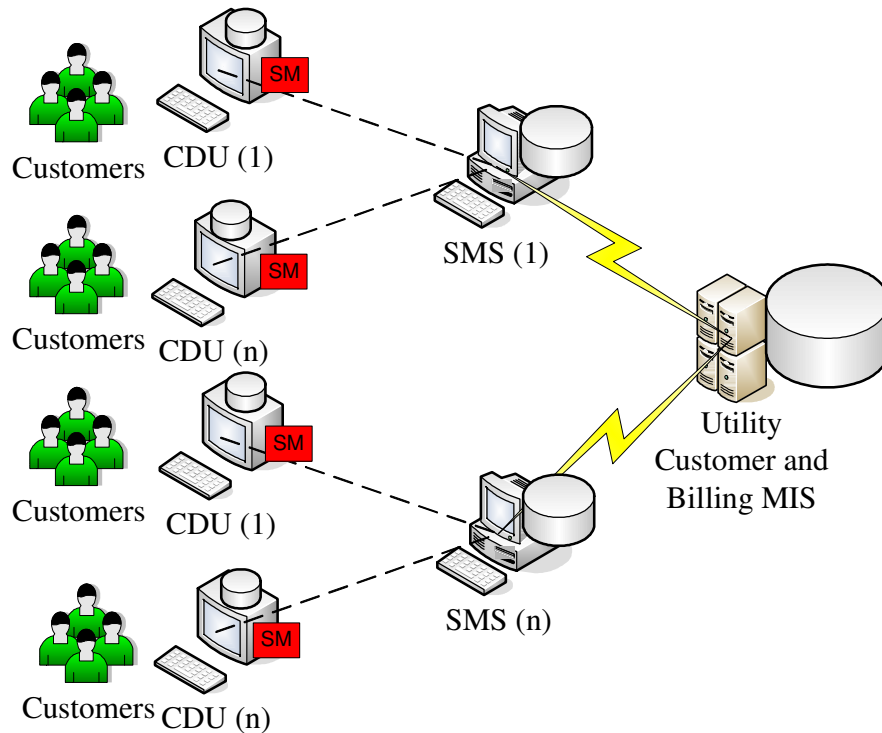


Figure 2.7 –Offline Electricity Sales System (NRS 009) Context Diagram

Although most supply authorities have standardised on STS meters, some authorities still have a significant historic install base of proprietary meters. Therefore, most CDU's are also configured to vend both STS and proprietary tokens.

2.5.2.2 Credit Dispensing Units (CDUs)

Figure 2.8 shows a typical first generation CDU, which was developed to the functional and performance requirements defined in [25]. These devices were built as specialised, self-contained devices for prepayment vending that could be securely mounted on a wall or counter. They include a card reader for reading customer meter cards and capabilities to vend magnetic and numeric tokens. Their design also assumed a low operator skill and education level; therefore a specialised touch screen user interface was developed. This very user friendly interface supported repetitive tasks in a high transaction volume environment.

The operating environments of these devices were sometimes very harsh such as, the living-room floor in a vendor's hut in a remote area. Therefore, it was designed to

withstand high levels of dust, static electricity, power surges, vibration, insects and operator abuse.



Figure 2.8 – Specialised CDU³

The newer second generation CDUs (Figure 2.9) are no longer built as specialised devices. They make use of a standard personal computer (PC) form factors and peripherals like, PC mice and keyboards. This has resulted in price reductions of CDUs. Operators are however required to have some minimal PC skills training. The benefit of such training is that operators are now empowered to apply their PC skills to more general computer use.



Figure 2.9 –Typical PC based CDU Installation

³ Photograph courtesy of Eskom's photo archive.



2.5.2.3 System Master Stations (SMSs)

SMS functional and performance requirements are specified in [24]. The SMS application is installed on a standard personal computer (PC) and operated at the utilities offices. Typical SMS functions include managing vending transactions, customer and meter management, tariff management, vendor credit and CDU configuration management. The SMS acts like a data concentrator for transaction data, which consolidates transaction information from all linked CDUs. The consolidated transaction data is then optionally uploaded to a higher-level management information system. Data transfer between the SMS and CDUs is accomplished in two ways:

- Modem transfer as specified in [32].
- Disk transfer as specified in [33].

Although the interface between the CDU and SMS has been specified in [32], most suppliers have implemented their own customisations to this interface in an attempt to provide additional functionality. This has resulted in incompatibilities between CDUs and SMSs supplied by different manufacturers.

Most SMS's connect to a higher-level information management system which is responsible for managing the customer database and the database for transactions. Annexure A of [24] provides a file based interface to transfer customer and transaction data between the SMS and information management system. An enhanced SMS can also function independently from a higher-level information management system and therefore operates in a standalone mode. In this mode the SMS is responsible for the management of the customer and transaction databases.

2.5.2.4 Vending Information Management

Offline vending enables customers to purchase electricity using information from three sources:

- A customer meter card;
- A previous token; and
- A customer / meter database record maintained at the CDU.



This means that offline vending can provide close to 100% system availability and is therefore well suited to rural applications. However, the distributed nature of CDUs results in utilities sometimes allowing updates to CDU customer / meter records independent of their customer management systems in favour of customer service.

As the number of CDU's and SMS's increase, maintenance becomes more costly and complex to manage. The following key issues are encountered:

- Unable to perform customer and vendor reconciliations based on transaction information from CDUs;
- Unable to timeously update tariff information on CDUs, which ensures customers purchase on the latest tariffs; and
- Unable to timeously update a customer's meter information should a customer move to another tariff, move to another location with a new meter, or have his existing meter replaced.

This results in CDUs controlling and managing customer and tariff information rather than the utility's management system. Therefore, utilities find it difficult to maintain and control customer information and tariffs. In some instances customers purchase electricity but the utility had no record of these customers. These customers are referred to as "unallocated" customers since the utility is unable allocate such transactions to customer accounts in their customer database. Unallocated transactions pose a major management and financial control issue for utilities.

Therefore, more offline vending systems are providing at least periodic communications between the utility management systems, the SMS and CDUs. Such communication links ensure that CDU transactions are uploaded to the SMS and customer, meter and tariff information is downloaded to the CDU. Where such communication links are reliable and cost-effective enough, CDU's are even "locked down" to only vend from the customer information downloaded from the SMS. Although such functionality addresses the unallocated transaction issues it can have a negative impact on a customer's ability to purchase electricity. That is, the customer must be registered on the utility's management system, downloaded to the SMS and then to the CDU before being allowed to purchase electricity.



2.5.2.5 Offline Vendor Management

The utility typically manages vendors using one of two business models [11]:

- **Credit business model:** The credit business model requires vendors to pay a deposit with the utility before being enabled to vend electricity. The vendor periodically deposits the sales into the utility's bank account. The vendor is paid commission on the deposited sales. This model requires trust since there is no guarantee that the vendor will deposit the sales into the utilities account. Experience has shown that this model requires close vendor management.
- **Upfront business model:** The upfront business model requires vendors to make a payment upfront before being allowed to vend electricity. The upfront payment becomes the vendor's credit with the utility to sell electricity. The vendor's credit is reduced after every prepayment sale. Once the vendor's credit reaches zero, vending is disabled. The vendor's credit is updated when the vendor deposits money into the utility's bank account. This option requires a secure mechanism to deduct and update vendor credit at the CDU.

In both models the utility most often also supplies the CDU and required stationery due to the specialised nature of the CDUs. Further, utilities also like to maintain control over the CDU functionality for fraud prevention and investigation purposes. Therefore offline CDUs require close monitoring and control incurring significant costs for utilities.

2.6 SUMMARY

In this chapter, the author performed a detailed current state analysis of the electricity prepayment environment. The history, technology and drivers for prepayment were reviewed.

Prepayment technology has matured into a standardised industry in all three key components of the prepayment systems, namely, the meter (SAB1524), the credit transfer (STS) and the vending systems (NRS009 series). STS has been proven as stable and secure transfer technology for prepayment and enjoys significant support and install base. The standardisation efforts also enjoy significant participation from users and suppliers.



Prepayment technology has emerged as a sustainable option for electrification with benefits for both utility and customer.

Although offline vending systems have to been successfully implemented the following shortcomings were identified:

- Limited control over the CDU which could lead to abuse;
- Limited control over the CDU / SMS synchronisation which could lead to data management problems;
- Limited control over the SMs can lead to abuse and fraud if the CDU is stolen [17];
- Maintenance technicians to have access to the SM's for the purposes of spares. Such a situation has the potential to be abused and is open to fraud, if not properly controlled and managed [17];
- In some cases an inability to effectively support and manage new tariffs, such as free basic electricity (FBE); and
- There is not a standardised mechanism for vendor credit management.

CHAPTER 3 : REQUIREMENTS ANALYSIS – THE FUTURE STATE

3.1 INTRODUCTION

The previous chapter provided an understanding of the current prepayment environment, the key technologies, specifications and challengers. This chapter looks at next generation prepaid electricity vending systems, that is, online vending systems. It also documents the online vending specification requirements.

The requirements analysis was undertaken by author using formal working group meetings, interviews with individual stakeholders, focus group discussions and informal discussions with industry and technology experts.

3.2 THE PROJECT WORKING GROUP

The online vending specification project was initiated under the auspices of the ESLC and led by Eskom, NRS 009 working group, industry technical specialists, equipment suppliers and manufacturers – in association with the STSA. The author was seconded from Eskom and appointed as the project leader. The project leader’s ultimate responsibility was to lead the specification development and release an industry specification for online vending.

The project success depended on the support, commitment and more importantly participation of all stakeholders. This meant that stakeholders needed to be identified, their role, influence and level of involvement defined. The key stakeholders identified were the utilities. It was critical that the differing functional requirements of the various utilities be adequately addressed. This approach supported the assertion that “Standards are a consensual activity” [13].

The commitment of the utilities as the users of the specification was essential as described [13]:

- Commitment to the problem: The user community must recognise the problem and must recognise the need for a standard as the solution.



- Commitment to the solution: The user community must seek to incorporate divergent viewpoints and seek consensus. The “rules of engagement” must be defined upfront and maintained. The community must identify contributors and commit resources to the problem.
- Commitment to implementation: The user community must commit to the implementation of the standard.
- Commitment to support: The user community must commit to supporting the improvement of the specification between versions. They must also continuously promote interest, de-mystification, training and best practices of the standard and its implementation.

Further, utilities acknowledged that input and technical support from the competing proprietary online vending system and offline vending equipment suppliers would be essential. Supplier support and contribution was essential to ensure that lessons learnt from proprietary protocols and other online vending best practises could be incorporated into the specification development process. This would be an extremely difficult task since it required participation and sharing amongst competing vending equipment suppliers that had vested interests and possibly driven by “hidden” agendas.

An industry working group was constituted to monitor and guide the protocol development process. The working group was mandated to review, moderate and approve all specification proposals. Members of NRS 009 Manufacturers Interest Group (MIG) and the NRS 009 working group were the first participants invited to join the working group. The working group was also opened to any other interested parties. ADDENDUM B provides a full list of working group participants.

The author chaired and managed the working group. It was imperative that an appropriate working environment was fostered to facilitate the development of the specification amongst competing organisations. Therefore, a participatory approach was employed, which fostered an open, inclusive and transparent environment. The environment enabled participants to develop mutual “trust” and trust in the process and its objectives. The participatory approach also promoted collective ownership and a vision for the common good.



Working group meetings were kept to a minimum since participation was voluntary and all participation costs were borne by the participant or participating organisation. Therefore, working group meetings were only organised when key deliverables needed ratification or urgent issues needed resolution. Between workgroup meetings participation was encouraged through an open mailing list, which proved to be a cost-effective mechanism to discuss and reach consensus. Several issues were discussed and resolved using the mailing list.

At the requirements analysis phase working group meeting the participating utilities communicated their intention to develop an industry specification for online vending. The utilities also requested the vending system supplier's support and assistance in specification development process. The meeting concluded with the following guidelines for the specification development:

- Utilise or specify existing internationally accepted standards where appropriate;
- The protocol must focus on defining the application layer of the Open System Interconnect (OSI) model. The rest of the layers should be based on existing standards;
- The message exchange formats would form the core of the protocol, but their exact definition would be defined in the detailed design. eXtensible Markup Language (XML) was recommended as the preferred message format;
- The protocol should cater for cross distributor vending; and
- The protocol should define an open interface specification between the Point of Sale (POS) and Security Module (SM).

3.3 UTILITY REQUIREMENTS

The key beneficiaries and end users of the online vending specification are electricity distribution utilities. In South Africa the licensed utilities are Eskom and some local municipalities. It was therefore critical to understand the varied utility drivers, challengers, functional and technical requirements of online vending systems. This would enable the project to capture these requirements in the specification design.

All utilities approached were supportive and committed to an industry specification for online vending. The utilities also acknowledged that while open standards can sometimes



suffer from an overly bureaucratic process they can carry tremendous weight as stated by [12]:

- The user community feels comfortable knowing that one vendor is not trying to lock them out of another vendor's products;
- If a new standard is developed to supersede an existing one, there is a strong likelihood that the controlling body will propose a migration path wherever possible;
- Costs can remain competitive because competing vendors comply with the same standard;
- The standardisation process adopted by the controlling body is normally (more) transparent and open to scrutiny; and
- The user community gets some level of comfort knowing that the technical integrity of the standard has not been subjugated by commercial expediencies.

However, utilities did differ considerably on the scope of the specification. Some utilities requested that the specification address a complete online vending system, which included specifications for the online vending server, the online vending client devices, the client / server messaging and communications protocol. Other utilities however requested that the specification focus only on the client / server messaging protocol and leave the server and client development to supplier innovation.

A few municipalities could not wait for an industry specification before deploying their online vending systems. Although these municipalities implemented proprietary systems, their feedback, experience and contribution to the development of the online vending specification proved invaluable. Their participation in the project demonstrated that they were fully committed to an industry specification. In some instances these municipalities contractually bound their online vending system supplier to upgrade to the industry specification once released. Municipalities that pioneered Online Vending systems in the absence of an industry specification were:

- Manguang Municipality;
- Nelson Mandela Municipality; and
- Buffalo City Municipality.

Eskom has the largest footprint of prepaid metering and vending systems, with more than 3.5 million installed meters and 1500 offline vending points of sale. It also committed to



only implementing an online vending system that complied with the industry specification. Eskom maintained that a standardised online vending system was a prerequisite to the long term sustainability of such a solution. This view was based on Eskom's successful deployment of prepayment metering and offline systems, which was largely as a result of their commitment and support of prepayment standards.

3.3.1 Drivers for online vending

Online vending systems promise several benefits to utilities. The following key utility drivers for deploying online vending systems were identified [6]:

- Current offline vending systems reaching the end of their life cycle;
- To streamline and improve the management of prepayment meters, vendors, customers and transactions;
- To integrate vending systems with other business systems, such as the utility Enterprise Resource and Planning (ERP) systems;
- To improve customer service by increasing vending footprints, i.e. providing the customer with convenient access to POSs;
- To efficiently offer “Free Basic Electricity” within the prepayment domain; and
- Reducing prepayment vending system fraud.

3.3.2 Online vending challenges

Online vending systems also introduce new challenges and risks that utilities had to acknowledge and implement appropriate mitigation strategies. The following challenges and risks were identified [9][10].

- The relatively high cost start-up costs of online vending systems;
- Vending server downtime means that no vending can take place. Therefore, sufficient backup and business continuity systems are required to minimise server downtime;
- Online vending requires real-time network connectivity between the server and the POS devices. This would be a serious challenge in some rural and outlying areas where network connectivity was limited or non-existent;

- Network connectivity (especially a public network like the Internet) opens the vending systems to attack from malicious sources;
- No industry standards exist for online vending. Therefore current systems are proprietary and could lock the distributor into a single supplier and technology;
- Online vending systems are dependent on reliable communications links; and
- Online vending systems are far more complex than current offline systems. They implement multi-disciplinary technologies such as vending, prepayment metering, systems engineering (information technology), computer networks, business continuity and computer / network security. Therefore, highly skilled personnel are required to implement, maintain and administer such systems.

3.3.3 Online vending models

Utilities identified the following online vending models, Normal Vendor or Gateway Vendor.

Figure 3.1 illustrates the “Normal Vendor” model. This model operates very similarly to current offline vendors. The vendor operates an online POS client device where customers purchase electricity tokens. The online POS client communicates with the online vending server to generate and return the requested tokens.

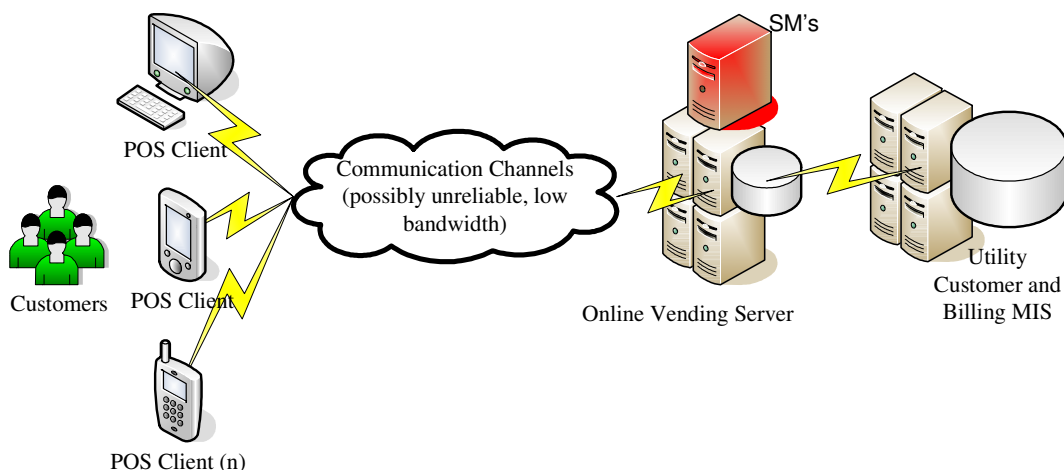


Figure 3.1 – Normal Vendor Model

Figure 3.2 illustrates the gateway vendor model. In this model, customers purchase electricity at vendor terminals. The terminals communicate customer requests to a vendor

gateway server. The gateway server communicates with the online vending server to generate and return the requested tokens. The gateway model applies to vendors who currently have a footprint of terminals and would like to add prepayment electricity vending capabilities to their footprint.

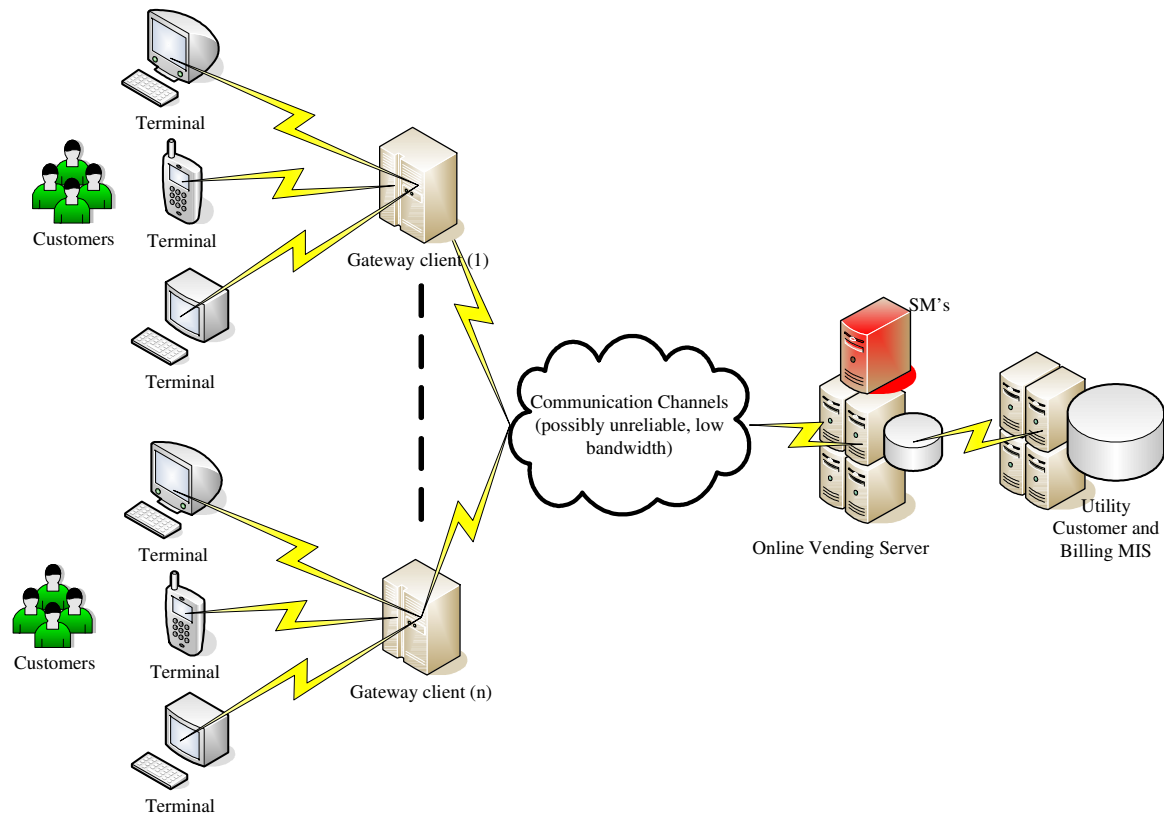


Figure 3.2 – Gateway Vendor Model

Utilities also indicated that online vending had to support both upfront and credit vending business models (see section 2.5.2.5). Utilities also indicated that they would prefer a model where the vendors provide the vending client and required stationary. A well defined online vending interface specification between client and server would be required to support this requirement.

3.3.4 Online vending system requirements

The discussions and interviews elicited the following generic online vending system functional and technical requirements.



3.3.4.1 Functional requirements

- **Free Basic Electricity (FBE):** The system must be able to support the issue of FBE tokens. The FBE token is a special type of electricity credit token, designed to issue a pre-programmed credit amount to each meter once per month, free of payment. FBE has been successfully implemented in South Africa as a poverty alleviation intervention [10]. Support for FBE was one of the main business drivers for utilities considering the implementation of online vending systems. Especially municipalities which still had large install bases of proprietary meters.
- **Debt Recovery and Account Payments:** The system must be able to support the following debt recovery and account payment functionality:
 - Support payment of other municipal accounts such as municipal services, rates, etc.
 - Implement blocking of electricity purchases until the debt is settled or arrangements have been made to settle the debt.
 - Recovering a fixed amount to settle part of a debt from a customer before an electricity purchase is allowed.
 - Recovering a percentage of each electricity purchase towards settlement of the debt.
 - Recover a fixed amount per time period, such as a fixed monthly charge on the first purchase of the month.
 - The system must allow for the recording and allocation of individual customer agreements.
- **Multiple Supply Group Vending:** The system must be able to support vending to more than one supply group. The addition and removal of supply groups must be easily configurable. There should be no limit to the number of configurable supply groups.
- **Algorithm Technology:** The server must provide for generations of STS and proprietary tokens.
- **Vending Database:** All data capturing, e.g. purchase transactions, customer information, etc. must be stored in a secure central database. The system should be able to archive selectable record sets. The system must also allow for archived records to be restored and queried without affecting the normal operation of the system.
- **Offline Vending Support:** Offline vending must be possible at a specially selected subset of vending stations under special circumstances. For example, when the



communication link with the vending server is not available. A requirement for a specially developed online / offline CDU was expressed. Such a CDU should operate by default in an online mode, however when communications with the server is unavailable it should automatically continue operating in an offline mode. It should automatically switched back to the online mode when communications with the server is restored. At this point the all offline processed transactions must be replicated to the server.

- **Vendor Management:** The system should support the following vendor management functions:
 - Operator shift management and security.
 - Banking, sales and shift batches.
 - Setting up credit limits for individual vendors.
 - Up-front vending.
 - Electronic banking reconciliation and credit replenishment.
 - Blocking of a vendor or terminal.
- **Reporting:** The system should provide the following generic reports and provide for user defined reports:
 - Electricity purchased by cash, cheque, credit card, debit card and electronic funds transfer (EFT).
 - Recovery of arrears.
 - Customer purchase history.
 - Vendor purchase history.
- **Prepayment specification support:** The system must support existing prepayment specifications were applicable.

3.3.4.2 Technical requirements

- **Transactional Support:** The server requests must be encapsulated within a transaction. That is, if part of transaction fails then entire transaction should fail. This improves the reliability of token requests and database integrity.
- **Capacity and Performance:** The system should be scalable to process 60 or more standard vending requests per second. A standard token request from start to end



should not exceed 5 seconds. The system should be able to handle 120 million transaction records.

- **System Integration:** The system will have to provide integration capabilities with existing and future systems. Examples of such systems are Financial Systems, Web Servers, SMS gateways, Billing Systems, Customer Relationship Management Systems and other systems.
- **Online Vending channels:** The system should be able to process token requests through a variety of channels. Examples of such channels are:
 - The world wide web (WWW), using the Internet.
 - An interactive voice response (IVR) system, using the telephone network.
 - Short Message System (SMS) using the cell phone network.
 - Using a private network, such as an organisation's Intranet.
- **Vending POS:** The system should support the following list of possible POS devices:
 - Fat or thin client application.
 - A CDU that that has been updated to operate in an online mode.
 - A POS banking terminal.
 - Automatic Teller Machines (ATMs).
 - Cellular telephone.
 - Transaction switching servers.
- **Communication Protocols:** Utilise standard network communication protocols. Further, such communication should be kept to minimum to reduce system communication costs.
- **Disaster recovery and business continuity:** Backup and business continuity needs to be integrated into designed of the system. This applies to both the system hardware, databases and communications network. The disaster recovery plan must consider all possible scenarios so that the downtime is kept to an absolute minimum.
- **System administration:** The system should provide the facilities for remote administration.
- **Auditing:** An audit log of all transactions processed by the system must be kept so that transaction records can be traced in the case of disputes.
- **System Security:** The areas where system security needs to be addressed are:
 - **The Vending Server:** The SMs used to generate the tokens would be installed in the vending server therefore it would need to be hosted in a secure environment with strict physical and remote access control.



- **The POS or CDU:** The removal of the SMs from the POS device significantly reduces utility risks. However, the POS should implement security measures limit vendor risk if stolen.
- **Communications between POS and Server:** The communication must ensure data confidentiality, data integrity and authentication of the communicating parties.

3.4 SUPPLIER REQUIREMENTS

The following vending equipment suppliers were approached to elicit their input and participation in the specification development process.

No	Supplier name
1.	Landis and Gyr
2.	Conlog
3.	Contour
4.	CBI
5.	PN Energy
6.	Actaris
7.	Prism (EasyPay)
8.	Syntell

Table 3-1 – Visited Suppliers

All the suppliers were frank and open about their views on developing a specification for online vending. All the manufacturers supported the need for an open specification. They also expressed similar views as the utilities with regard to the specification scope. All suppliers indicated their willingness to participate in the specification development process.

Suppliers of proprietary “online” vending systems also revealed that their current “online” vending product offerings were seen as their competitive edge and were not willing to disclose much in this regard. However, most of the suppliers provided useful insights on issues that would have to be addressed by an industry specification.

3.4.1 Online vending system requirements

Suppliers expressed the following online vending system requirements:



- Network and database redundancy.
- Support for both offline and online vending infrastructure, with at least offline credit token vending at limited sites.
- Support for both STS and proprietary tokens. The vending of proprietary tokens by all suppliers is problematic since suppliers are not always amicable to sharing or licensing proprietary token transfer technologies.
- The token generation times, client device response times and transaction processing capacity should be specified.
- Support for vendor credit management on the server.
- Support for cross utility vending.
- Support for utility specific business and system management processes.
- Compliance with the electronic communications and transaction (ECT) bill.

3.4.2 Online vending specification requirements

Suppliers expressed the following comments with respect to the online vending specification development:

- The specification should be an interface specification and not a system specification. The application level specification should be private to each supplier but comply with the interface specification.
- The interface specification should specify a basic set of processes and leave room for supplier specific additions and enhancements. It should not define protocols / formats/ etc, but merely define the high level processes that must be supported.
- The specification should be written for both international and South African users.
- The specification should support various methods of payments, e.g. cash, credit card, cheque, debit card.
- Transaction support must be an integral part of the specification.
- The testing methodology for conformance and adherence to the specification must be specified and centrally managed by a third party.
- The specification would need to specify whether server or client formatted receipts are supported.

3.5 ANALYSIS OF ONLINE VENDING SPECIFICATION REQUIREMENTS

This section summarises and analyses the requirements documented in the previous sections. It aims to provide a common understanding of online vending and the scope of the online vending specification.

3.5.1 Offline vending

Currently most electricity prepayment vending is done in an offline mode. In this mode the client application provides the following services locally (Figure 3.3):

- STS token generation.
- Business logic processing.
- Local customer and transaction database.

The local database is periodically synchronised with a master station. The periodic synchronisation is achieved through floppy disk or modem communication.

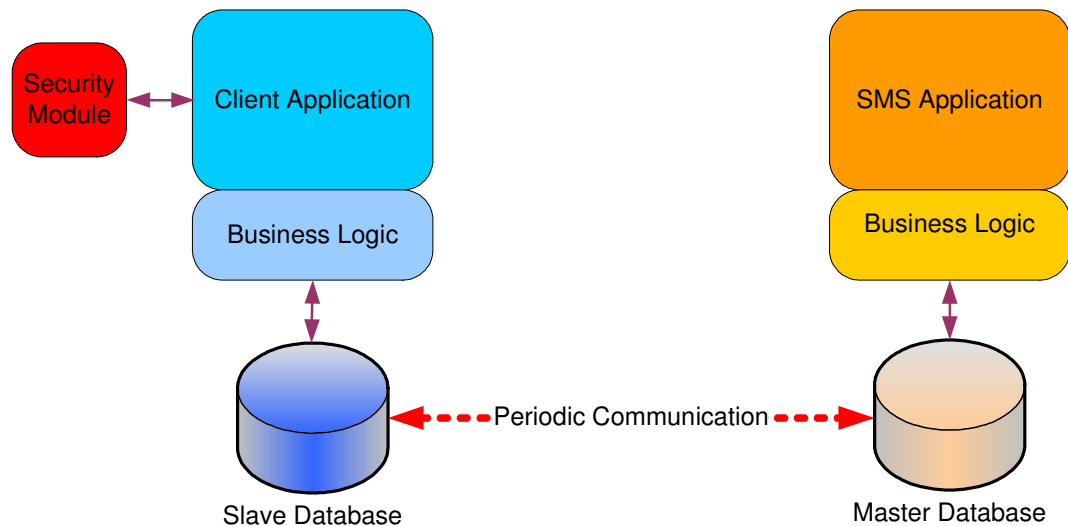


Figure 3.3 – Offline Vending

In offline vending systems, the POS device is defined as a credit dispensing unit (CDU). A CDU has a client application, business logic and slave database. The CDU hosts SMs that generate prepaid tokens. The higher-level system is represented by a SMS. The SMS has a SMS application, business logic and master database. The SMS manages several slave CDUs.

3.5.2 Online vending - or is it?

Figure 3.4 illustrates the database services being accessed from a central location. However, the following key services are still provided locally:

- STS token generation; and
- Business logic processing.

The difference between this configuration and that in Figure 3.3 is that now the database transactions are executed on a single master database through a permanent communications link. Figure 3.4 is referred to as a database vending.

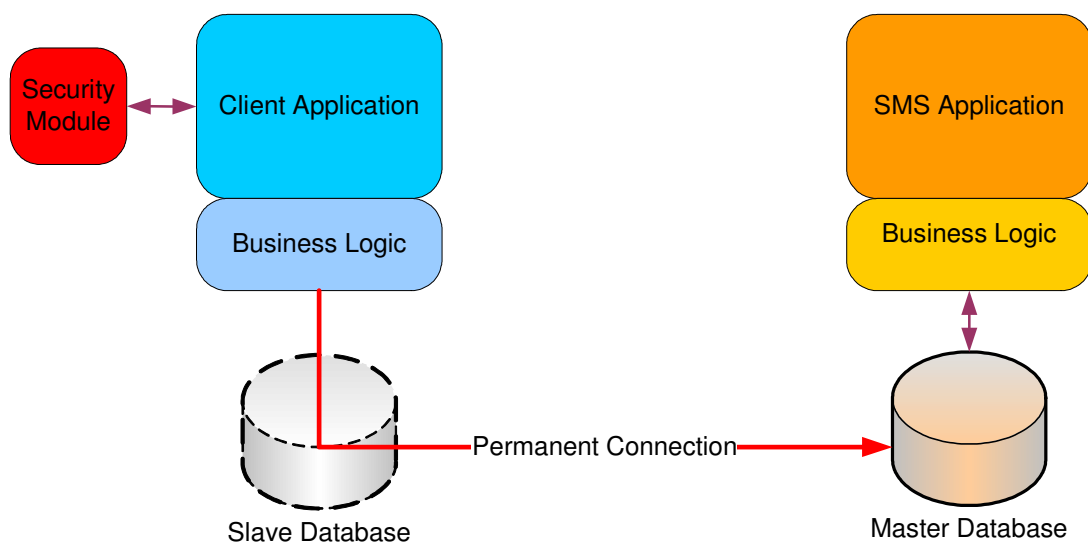


Figure 3.4 – Database Vending

In this configuration a permanent network connection is setup to facilitate the use of a multi-user master database. This configuration is not considered a “fully-fledged” online vending system since the token generation and encryption services are still provided by the CDU. Such a configuration may however be useful in a “trusted” environment and could be used to support both database and offline vending.

3.5.3 Online vending

Online vending is achieved once all of the services are accessed from a centralised “server” (Figure 3.5). These services are:

- Prepaid token generation and encryption;
- Key business logic processing; and
- Single multi-user master database.

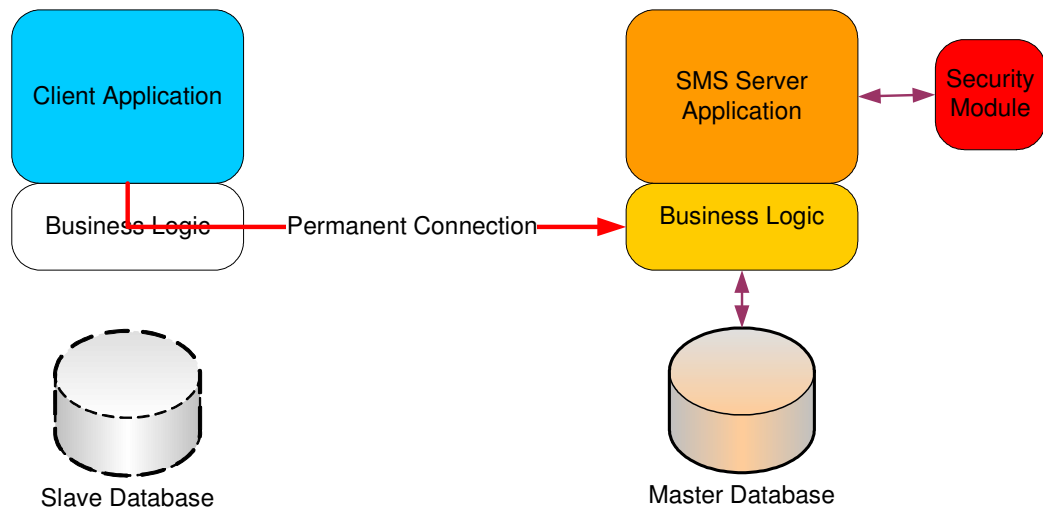


Figure 3.5 – Online Vending

This configuration is referred to as a client / server architecture. Further, if an appropriate interface protocol is implemented between the client and server, the presentation layer details can be separated from the vending business logic. For example, the client application could be hosted on a Web Server, an IVR Server or a Short Message System Server, each supporting various presentation layer formats, while the backend communications with the vending server uses a standardised interface protocol.

Based on the analysis in section 3.5, a common and easily understandable definition for online vending was compiled. Online vending is defined as [5]:

A prepayment token is requested from a Server that is remote from the actual point of sale client device. The token is only generated on the Server and transferred to the POS client, once the transaction, the POS client and the payment mechanism has been authenticated and authorised. The connection between the POS client and the Server is a standard computer network channel, such as telephone, Internet, direct dial-up link and general packet radio service (GPRS).

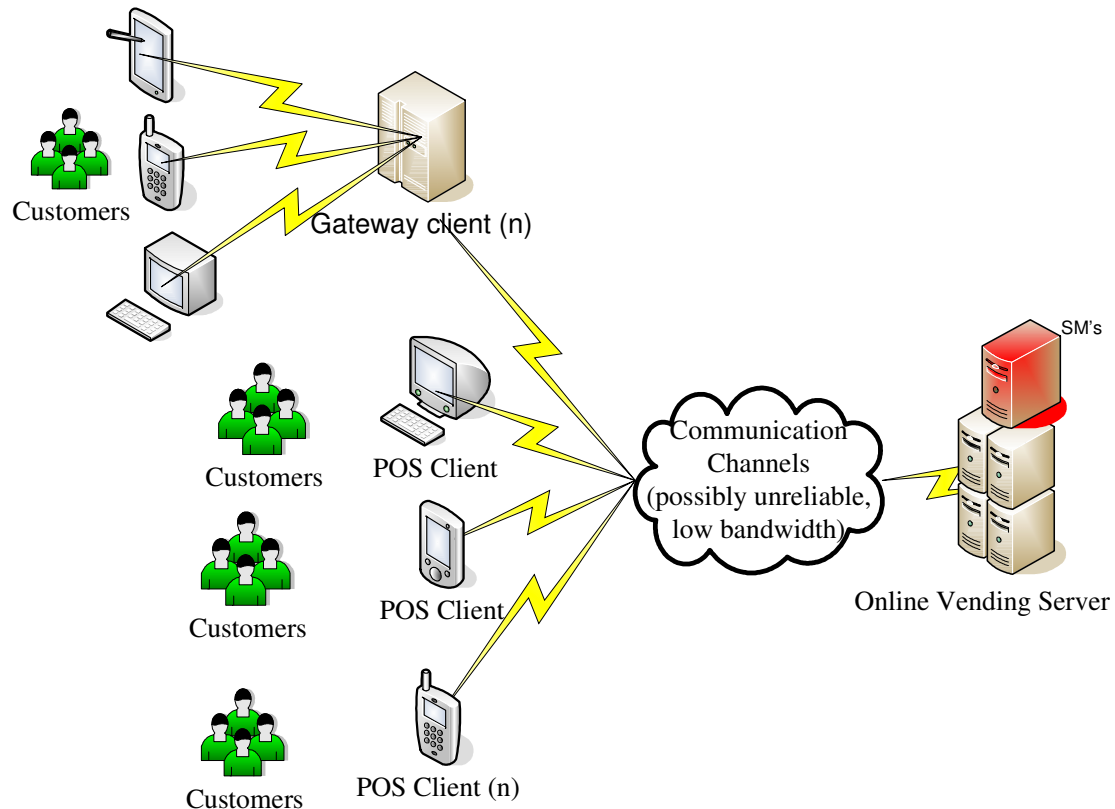


Figure 3.6 – Online vending context diagram

In other words, a POS client must communicate with a higher level management system (Online Vending Server) to complete a prepaid vending transaction (Figure 3.6). Without such a communication link no prepaid transactions are possible. The most important difference between offline and online POS devices is that online POS client devices have no built-in token generating capabilities. This service is now provided by the vending server.

3.5.4 Online vending specification scope

As discussed earlier the scope of the online vending specification was not initially very clear and stakeholders had varying views in this regard. However, the working group finally agreed on the following:

- The online vending specification should specify the interface protocol between the vendor server and the vending client;
- Web service technology should be used as the basis for messaging specification; and

- The specification should allow for protocol adapters / bridges as a gateway to support of legacy and proprietary protocols.

The scope of the online vending specification was therefore defined and illustrated (Figure 3.7) as follows:

The online vending specification defines a secure and open web service based interface protocol that facilitates prepayment vending functionality between a single logical vending server and n number of clients.

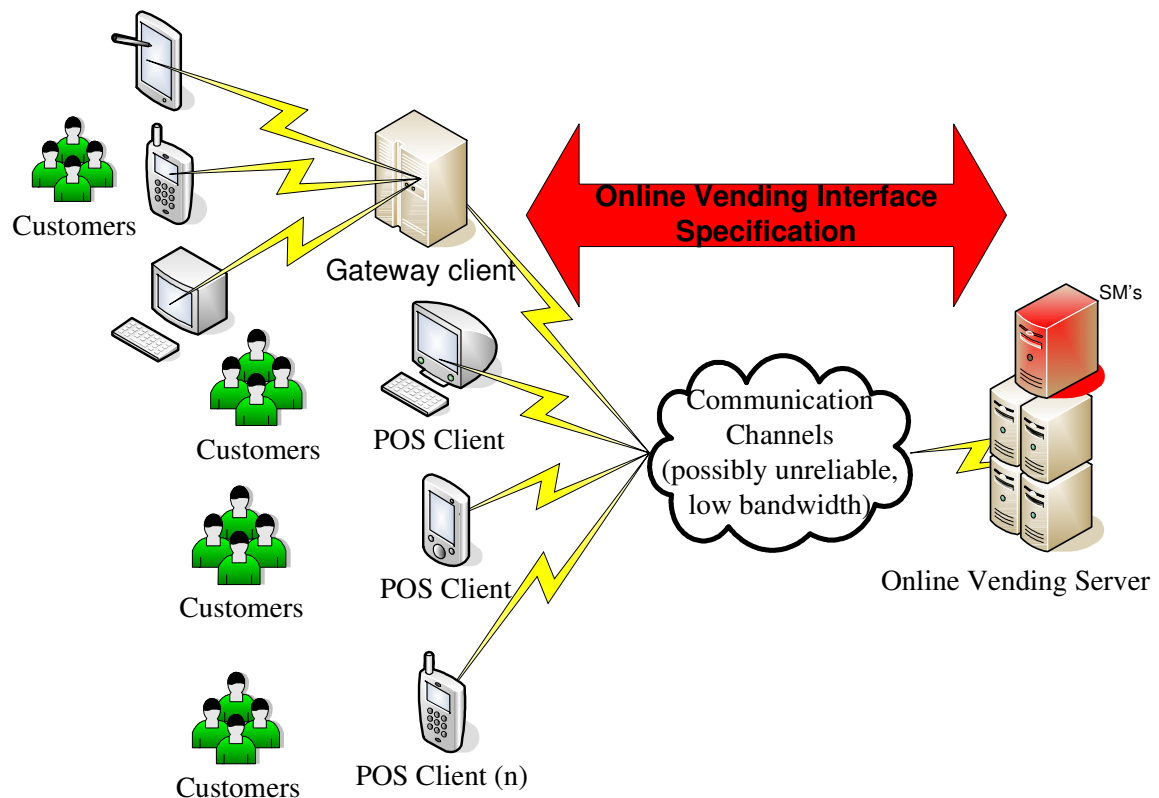


Figure 3.7 – Online Specification Context Diagram

The online vending interface protocol empowers utilities to source online POS client devices and servers from different suppliers. It protects utilities from being locked into a single supplier. Utilities also have greater freedom to substantially expand their vending footprint through various established channels using a single interface protocol. It also encourages innovation and competitiveness amongst client and server suppliers while ensuring interoperability between their systems. It also results in the widest possible choice of price and performance to the user.



An interface specification therefore assists to prevent supplier and technology lock-in. It provides for supplier choice and competition, while ensuring all implementations meet an acceptable level of performance, integration and security standards.

The online vending interface protocol and specification was appropriately named, XMLVend. XMLVend is essentially a vertical industry vocabulary, aimed at the prepayment vending industry. XMLVend is just one of many vertical industry vocabularies that have been developed, such as XMLPay [46]. The decision to implement the message design using the web services framework was based on current industry trends. Web services and related technologies have been accepted as the preferred framework to integrate systems across boundaries such as operating systems, languages and platforms [14].

3.5.5 Supplier proposals

Suppliers on the working group were requested to make their protocols available to the industry for review and consideration as an industry specification. Suppliers were initially nervous and suspicious of this process. They also viewed their protocols as a product differentiator and competitive advantage. However, the following suppliers did make their XML based protocols available review:

- Actaris,
- Syntell; and
- Landis and Gyr

Actaris was the first company to make the protocol available for review. The client / server based protocol made use of direct TCP/IP socket connections to transfer their XML messages between client and server. The protocol was a full production version and was used in their online vending system. The protocol was therefore well tested and stable.

The Syntell XML protocol was a full web services implementation. Therefore it exchanged SOAP messages over Hyper Text Transport Protocol (HTTP) connections between the client and server. This protocol was however conceptual and not field tested.



The Landis and Gyr protocol was a simple XML protocol and made use of HTTP to communicate messages between client and server. This protocol was a production version and therefore field tested and stable. The protocol also used SSL with client and server certificates for mutual authentication.

The protocol proposals were reviewed and it was decided to use all the proposals as building blocks for the new XMLVend protocol. XMLVend would also include additional functionality and be fully web service enabled. This decision was supported by the working group.

3.6 SUMMARY

Utility and supplier requirements were analysed and the author succeeded in obtaining a common understanding of online vending among all stakeholders. Further the scope of the online vending specification was restricted to the interface protocol between the client and server. The working group also supported the use of web services framework as the basis of the interface protocol.

The server business logic, online vending system infrastructure requirements and client hardware and presentation requirements were excluded from the specification. This crucial decision enabled the protocol development to focus on the message design and definition. This drastically simplified a potentially very complex specification design.

Restricting the online specification to an interface protocol also encourages innovation and competitiveness amongst client and server suppliers while ensuring interoperability between their systems. This results in the widest possible choice of functionality, reliability, performance and price to the user, which meets the user's requirement for an industry specification.

It was decided to develop the industry protocol for online vending, XMLVend, based on the three supplier proposals. XMLVend would also be enhanced to provide additional functionality and be fully web services enabled.

CHAPTER 4 : DEVELOPMENT OF XMLVEND VERSION 1

4.1 INTRODUCTION

This chapter focuses on XMLVend version 1 development. To facilitate a faster development process, it was decided to develop two versions of the specification, that is, a pilot version (XMLVend 1) and a subsequent “production” version (XMLVend 2). The pilot version would support a limited number of use cases to speed-up the specification development process and making it available to utilities for pilot use. This approach suited the iterative design methodology and the pilots would be able to test, verify and improve the core aspects of the specification. XMLVend 2 would be an enhancement of XMLVend 1 based on lessons learnt and also support additional use cases.

The author managed the entire XMLVend version 1 development process and was the main contributor to the development. A key goal of the author was to develop a solid technical base for XMLVend version 1. This would support the protocols upgrade to XMLVend version 2 without fundamentally impacting the underlying design. The author had to also ensure that version 1 met the critical business and technical requirements defined in CHAPTER 3

Although the current version of protocol is XMLVend version 2.1, this important chapter documents the foundation on which the XMLVend protocol was developed. It also highlights XMLVend 1 shortcomings.

4.2 WORKING GROUP DESIGN REVIEW SESSIONS

Three working group meetings were convened during the development phase. The meetings were convened to discuss, review and approve protocol proposals and monitor progress. It was essential that working group consensus was reached on each critical issue raised before it could be accepted.



The meetings were attended by highly technically competent participants from competing companies. Chairing and facilitating the meetings were not an easy task. However, the following basic elements contributed to their success:

- Proposals, meeting discussion documents and the meeting agenda were circulated well in advance on the meeting.
- Critical issues were discussed and lobbied well ahead of the meeting.
- The meeting always started with:
 - A summary of key decisions taken the previous meeting.
 - Feedback on outstanding issues.
- Open and frank debates were always encouraged.
- Participants were frequently reminded of the main objective of the working group meeting and the project's desired outcome since discussions frequently went off the topic and did not add significant value to the issue at hand.
- Smaller task teams were constituted and mandated to resolve particular issues that could not be resolved, or consensus could not be reached at the meeting.
- Accurate meeting minutes were kept and circulated shortly after the meeting.

4.3 THE PROTOCOL DESIGN PROCESS

The protocol design was undertaken considering the following design criteria and guidelines:

- Utilise existing industry accepted protocols where possible.
- Specify minimum security, communication, and interoperability standards.
- It should be platform independent, extensible and adaptable.
- It should support existing prepayment standards, STS and CVS (NRS009 series).
- It should utilise existing open industry technology standards and protocols. New or proprietary protocols should be avoided.
- Ensure that widely available development tool sets support the chosen technology and protocols.
- Ensure that the chosen technology and protocols make use of existing developer skills.

The working group also supported the author's proposals that the protocol should not address the following:



- It would not address the scenario where a vendor uses his client to communicate with more than one utility server.
- It would not specify the transaction reconciliation process between the utility and vendor.
- It would not specify the business rules applied to a particular use case. Utilities would have to define their own use case business rules.
- It would not support electronics payments directly. The protocol could however capture the payment information (credit card, debit card, cash, etc) as part of the token request. Electronic payments for electricity tokens could use existing banking payment terminals and then process the token request using the online vending client.

Using the above design criteria and constraints the author approached the protocol design using the following high-level design steps:

- Step 1: Verify the business requirements.
- Step 2: Compile and define the use cases and use case scenarios.
- Step 3: Define the request and response message pairs. This included defining the each message element, its data type and constraint.
- Step 4: Define a fault handling mechanism.
- Step 5: Define a message delivery reliability mechanism.
- Step 6: Ensure that the protocol design was secure.

4.3.1 Use case definitions

Use cases define services offered by a system as perceived by external objects that interact with the system. Use cases were used to define the “services” or outcomes that could be achieved using the XMLVend protocol.

The use case actors and their responsibilities (Table 4-1) were defined as follows:

Actor	Responsibilities	Collaborators
Customer	<ul style="list-style-type: none"> • Initiate customer use cases. • Provide identification information. • Tender payment. 	Operator



	<ul style="list-style-type: none"> • Receive requested token / receipt. 	
Operator	<ul style="list-style-type: none"> • Open and close vending batches. • Verify customer identification. • Submit appropriate customer requests. • Present tokens / receipts to customers. • Handle customer queries. 	XMLVend Client or Terminal.
XMLVend Client	<ul style="list-style-type: none"> • Send request messages to the online vending server. • Receive and appropriately process responses messages. • Initiate the “Issue Advice” use case for use cases that may require this service. 	Terminal and Vending Server
XMLVend Server	<ul style="list-style-type: none"> • Authenticate vending clients / operators. • Produce appropriate response messages. 	Vending Client

Table 4-1 – Use Case Actors and Responsibilities

After reviewing an extensive list of use cases, the use cases were short listed to those illustrated in Figure 4.1.

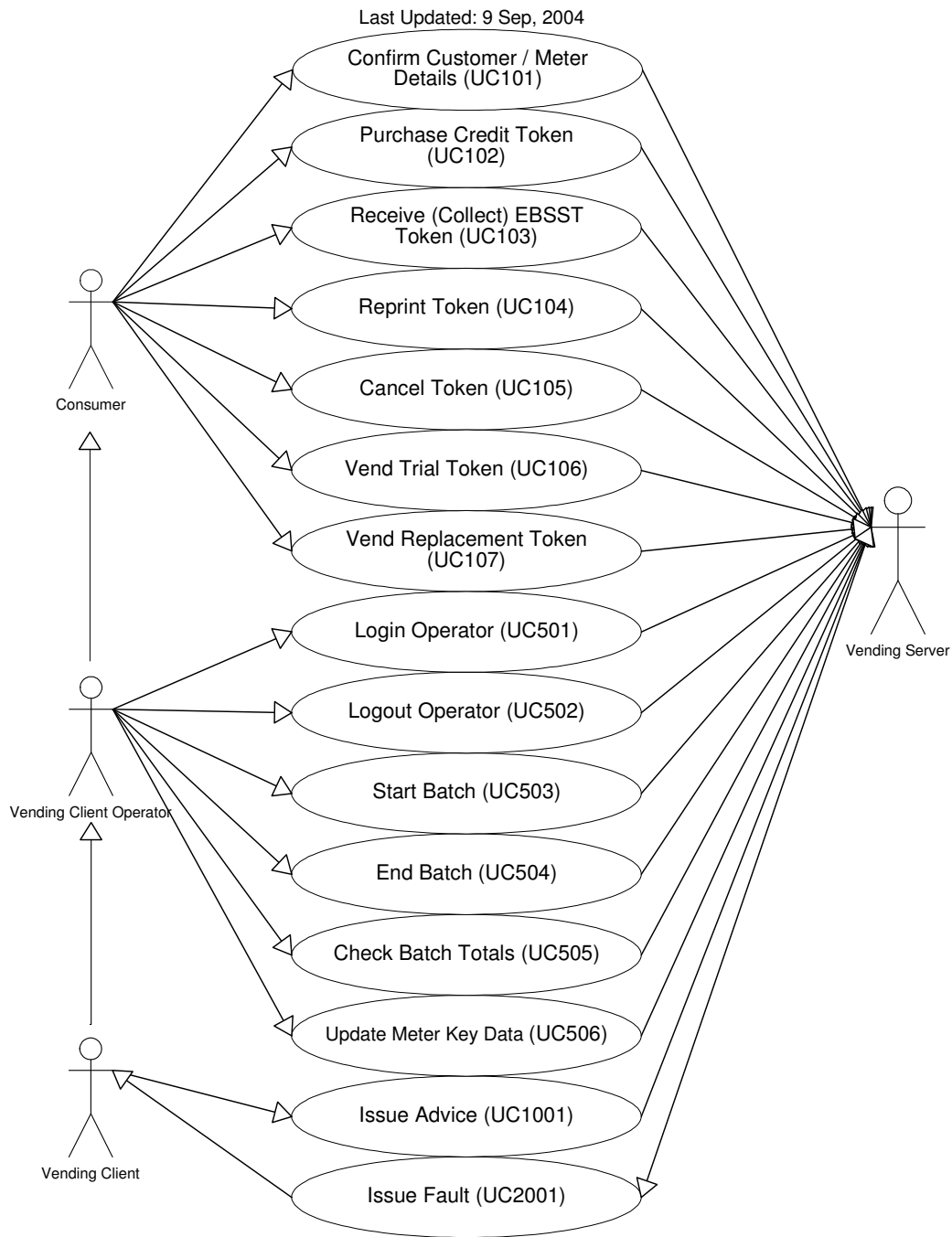


Figure 4.1 – XMLVend 1.0 Use Cases

Table 4-2 provides a description of each use case.

No	Use case	Description
1.	Issue Advice	Issue Advice is used to advise receipt of another use case’s server response. Two use case scenarios were defined: <ul style="list-style-type: none"> advice(reversal): The advice(reversal) message indicates to the server that the client has not received



		<p>an expected response message and the server should roll-back the transaction if completed.</p> <ul style="list-style-type: none"> • advice(confirmation): The advice(confirmation) indicates to the server that client has received the response message and the transaction can be committed.
2.	Confirm Customer / Meter Details	This use case returns the customer and meter information stored on the vending server. It is usually used to confirm supplied customer identification details before proceeding with use cases that create tokens such as Purchase Credit Token. This reduces the risk of generating incorrect tokens for the customer's meter.
3.	Purchase Credit Token	This use case is used to purchase credit tokens for a customer's meter. The value purchased may be expressed as currency or in kilowatt-hours. The server may also supply an EBSST ⁴ token (if applicable) with the receipt.
4.	Receive EBSST Token	This use case is used to obtain a customer's monthly allocation of "free" electricity through an EBSST token.
5.	Reprint token	This use case reprints a customers last purchased credit token.
6.	Cancel token	This use case cancels a customer's purchased token. It only applies tokens encoded on magnetic cards.
7.	Vend trial token	This use case simulates a purchase credit token use case but does not generate a token. It is used to check if a customer can afford the requested token when the token is requested as a kWh value.
8.	Vend replacement token	This use case is used to issue a replacement token for credit left in a replaced or faulty customer's meter.
9.	Log in	It is used to start an operator session with the server.
10.	Log out	It is used to end an operator session with the server.
11.	Check batch totals	It is used to check the status of current batch totals.
12.	Update Meter Key Data	It is used to request key change tokens for a customer's

⁴ The term EBSST has been replaced by term FBE in XMLVend 2.1.



		meter, which when entered into meter updates the meter key.
13.	Start batch	It is used to start a banking, sales or shift batch for an operator.
14.	End batch	It is used to end a banking, sales or shift batch for an operator.

Table 4-2 – XMLVend 1 use case descriptions

The fault response is not strictly a use case. It is used to return a fault message to the client when any of the use cases can not be successfully completed on the server.

4.3.2 Use case message exchange pattern

A synchronous request / response message exchange pattern between XMLVend client and server was required to realise the use case outcomes. The XMLVend client invokes an XMLVend use case by sending an appropriate XMLVend request message to the XMLVend Server. The XMLVend server executes the required service, and responds with the appropriate XMLVend response message. Figure 4.2 illustrates the synchronous request/response message exchange pattern.

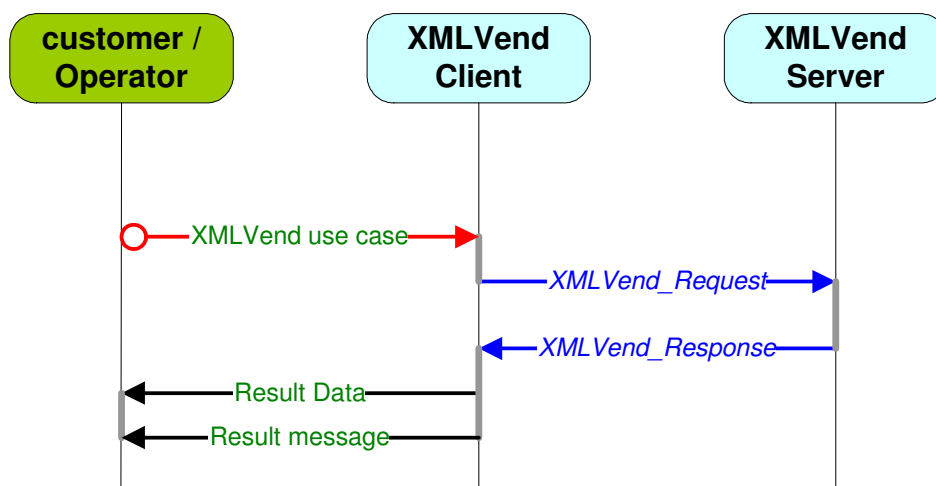


Figure 4.2 – Synchronous Request/Response Sequence Diagram



4.3.3 Use case request and response message definitions

The message pair definitions were guided by the following design considerations:

- The messaging would be stateless, that is, the server should not maintain client state at the messaging layer.
- Each request message should contain a unique message identifier. It was recommended that message identifier be a combination of the client timestamp and a unique number.
- It should specify the minimum message parameters to achieve the desired outcome of use cases.
- Use case messages pairs should comply with the basic data and functional constraints of STS and related NRS 009 specifications.
- It should provide a unique client identifier in all request messages. Servers should use this identifier to authenticate and authorised clients. The globally unique EAN numbering system was recommended for this purpose. The utility would usually be responsible for issuing the client identifier as part of the client registration process on the vending server.
- It should also provide for a terminal identifier to identify the terminals that initiated a request in gateway vendor scenario.

The initial request / response message pair parameter definitions were documented in a spreadsheet format (Figure 4.3) by the author. The spreadsheet defined the name, description and source of each message parameter on a per use case basis. The author circulated the spreadsheet to the working group where it was extensively reviewed, discussed and amended.



Name	Description	Source
Cancel_Request	SOAP message client to server	client
Input Parameter	Description	Source
MAC	Message Authentication Code (string)	client
ClientID	Client ID number (string) see Definitions	client
TerminalID	3 digits – allocated by the utility or vendor, however if the client also acts as the terminal then the terminalID defaults to "1".	Terminal/Client
MsgID	Sequential that allows each client request to be identified (numeric), see definitions	client
ID_string	Contains data value described by the ID_Type	Keyboard or Meter Card
Vend_ID_Type	Indicates the what the ID_String parameter represents, e.g. Track2Data, MSNO, consume name, etc.	Keyboard or Meter Card
Token	Token (string), used to confirm that this is a valid token.	Keyboard or Meter Card
Resource	Type of resource to be vended (1 = electricity, 2 = water, etc) default =1	keyboard
OpName	Optional Operator Name (string)	keyboard or ID device reader
Password	Password (string) (optional)	keyboard or ID device reader
Custom	Optional Custom Data (variable length string)	client
Cancel_Response	SOAP message server to client	server
Output Parameter	Description	Source
Header		
MAC	Message Authentication Code (string)	server
TerminalID	3 digits – allocated by the utility or vendor, however if the client also acts as the terminal then the terminalID defaults to "1".	Terminal/Client
MsgID	The Message ID of the corresponding request	client(input)
ResultMsg	Description of Result Message (string)(Optional)	server
DateTime	Date and Time (string DD/MM/YYYY HH:MM:SS)	server
MSNO	Meter Serial Number (string)	server database
AT	Algorithm Type (string as specified by STS) In conjunction with TT tells client which device to use for encoding or printing the token being served.	server database
KRN	Key Revision Number (string as specified by STS)	server database
SG	Supply Group (string as specified by STS)	server database
TI	Tariff Index (string as specified by STS)	server database
TT	Token Technology (string as specified by STS) In conjunction with TT tells client which device to use for encoding or printing the token being served.	server database

Figure 4.3 – Example spreadsheet format XMLVend 1 message definition

The message definitions were then converted from the spreadsheet format to the data model format used by the XMPay specification [46][6]. The XMLPay format was used to define the XMLVend 1 message data models. The XMLPay format was more intuitive and easier to understand, using the following conventions (Listing 4.1):

```
<Example>
(element)
(optional element)?
(alternate element1 | alternate element2)
(element)+
(element)*
</Example>
```

Element	Indicates the occurrence of a (possibly complex) XML element (for example, <element>.....</element>
?	Indicates an optional element
	Separates alternative elements, any of which is allowed.
+	Indicates that one or more occurrence of an element is allowed.
*	Indicates that zero or more occurrences of an element is allowed.

Listing 4.1 – XMLVend 1 data model notation



Listing 4.2 illustrates the XMLVend Login Request / Response message data models.

```
<Login_Request>
(ClientID)
(TerminalID)
(MsgID)
(OpName) ?
>Password) ?
(Custom) ?
</Login_Request>

<Login_Response>
(TerminalID)
(MsgID)
(CurrentVendorCredit)
(OperatorMsg) ?
(Custom) ?
</Login_Response>
```

Listing 4.2 – XMLVend 1 data model - Login Request / Response messages

Although the XMLPay format was an improvement on the spreadsheet format it did suffer some shortcomings, such as:

- It was difficult to maintain;
- It did not support data constraint definitions;
- It was not widely understood by the industry; and
- It was not supported by data modelling software tools.

4.3.4 Fault condition support

The protocol design needed a standard mechanism to communicate application faults to the client. Application faults are raised by the server while processing a request and the desired outcome can not be achieved.

Therefore, an XMLVend message called the “XMLVend fault response message” was defined to communicate fault information to the client. The XMLVend fault response message provides specific operator and customer fault information that could be used to resolve the fault condition.



The XMLVend fault response message was defined as follows (Listing 4.3):

```
<NRSFault_Response>
(TerminalID)
(MsgID)
(NRSFaultGroup)
(NRSFaultCode)
(FaultDescription)
(MessageVendor)
(MessageCustomer)
(Custom) ?
</NRSFault_Response>
```

Listing 4.3 – XMLVend 1 data model - Fault response message

A default list of fault scenarios was compiled as part of XMLVend 1. The list defined fault scenario codes, fault descriptions and scenario appropriate vendor and customer messages. Examples of these faults are given in Table 4-3. The complete list is given in appendix A of [51].

NRSFault Group	NRSFault Code	FaultDescription	Operator Message	Consumer Message
000 - General Fault Scenarios – Applies to all Use Cases				
000	001	Operator (x) Session has timed-out. X = Operator Name	Re-login.	To be defined in next version
000	002	A server error occurred processing request (x). X=MsgID and error message describing specific error.	Re-try. If error persists contact service provider.	To be defined in next version
UC102 – Purchase Credit Token – Fault Scenarios				
102	009	Unable to process 'Purchase Credit Token' due to business rule: (x). X= the business rule being violated + ref number.	Retry request without breaking the described business rule or contact service provider.	To be defined in next version

Table 4-3 – Example XMLVend 1 Faults

4.3.5 Message delivery reliability

In some scenarios, a client sends a request message but does not receive a response message. The client is now an uncertain of the outcome of the requested use case.



Therefore, the protocol design had to provide a mechanism to ensure message delivery or a mechanism to inform the client or server that a message was not successfully delivered.

The working group supported the inclusion of an additional use case, “Issue Advice” use case to enable message delivery reliability. The “Issue Advice” use case placed the responsibility and control of the resolving message delivery fault scenarios with the client application. XMLVend 1 defined two types of advice messages: reversals and confirmations. An implementation could implement reversals only or a combination of reversals and confirmations.

- **Negative Confirmation:** When a client issues a request and receives a valid response, then the client continues operating as normal. However, if the client does not receive a valid response, it can send a system reversal message to the server. The reversal advice message indicates to the server that the client has not received the response message and that the associated transaction must be rolled-back. In this implementation, confirmations are implied.
- **Positive Confirmation:** For all valid responses received by the client, a confirmation advice message to the server. However, if the client does not receive a valid response, it can send a system reversal message to the server. The reversal advice message indicates to the server that the client has not received the response message and that the associated transaction must be rolled-back. In this implementation confirmations are explicit.

4.3.6 Protocol security

This section discusses the XMLVend security framework requirements and the selection process.

4.3.6.1 XMLVend security requirements

Figure 4.4 illustrates the security context of Online Vending. It was assumed that the protocol would be implemented over an un-trusted and insecure network.

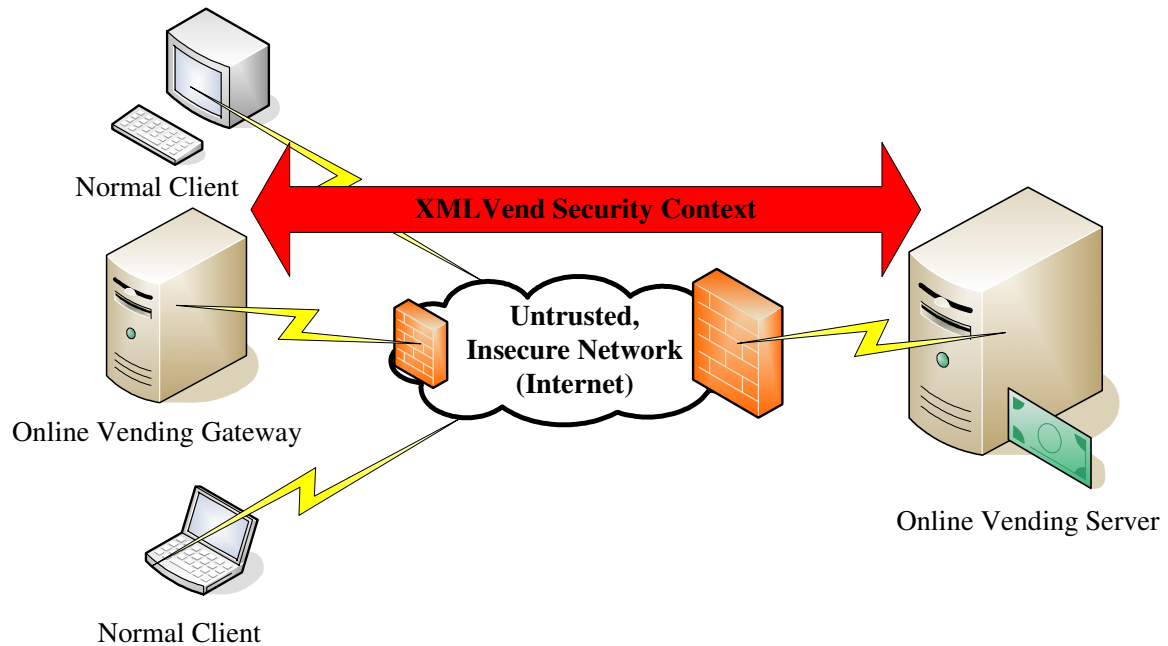


Figure 4.4 – Online Vending Security Context

The security requirements of the online vending protocol were analysed and presented in a working group discussion document [37]. It was essential that the protocol's security services protected the transacting parties from each other and attacks from malicious third parties. The online vending security requirements were:

- **Authentication:** Authentication is the process of confirming the claimed identity of an entity. Client authentication is necessary to ensure that only legitimate vending clients use the service. Once a vending client has been authenticated, the service can give the client access to authorised vending services. While the server would be reluctant to allow any client to connect and make requests, vending clients also need to be sure they are communicating with legitimate server. Therefore mutual-way authentication was essential.
- **Authorisation:** This ensures that vending client has the required rights for the services being requested. For example, a client may not be authorised to perform a Cancel transaction function, but may be authorised to perform a Purchase Credit token transaction function.
- **Confidentiality:** The token information need not be encrypted since it was already encrypted using the STS protocol. However, customer information should be kept confidential.
- **Integrity:** Modification of request and response messages in its path from client to server and back can significantly impact the reliability and trust worthiness of the



system. Therefore such modifications should be prevented and if it should occur it should be detectable so that the appropriate action can be taken.

- **Non-repudiation:** The transacting parties should be protected from either party repudiating a transaction. Non-repudiation guarantees that both the server and client are protected against claims from either party that the transaction did or did not occur at a later point in time.
- **Accessibility / Availability:** The success of an online vending system depends mainly on its high availability. Therefore, mechanisms need to be provided to prevent attacks, like denial-of-service (DoS), from outside or inside of the system hosting the service.
- **Audit:** A record of successful and unsuccessful service requests needs to be kept in order to:
 - Detect unauthorised access attempts;
 - To identify individuals attempting to access resources without proper authorisation; and
 - As evidence in cases of disputes.

The analysis however revealed that the following minimum security services would be acceptable:

- Authenticate the origin of messages (the source of a message must be proven);
- Guarantee the integrity of messages (contents of messages cannot be changed without being detected by the receiver); and
- Securing the message transmission was not essential.

In terms of communication and performance requirements, the XMLVend security protocol had to also consider the following:

- Communication bandwidth required must be kept as low as possible;
- Number of messages passed to achieve a part of communication must be minimized in order to maximize the responsiveness of the system; and
- The number of resources required to communicate to achieve a secure communications must be minimized.



4.3.6.2 XMLVend security options

The author, supported by the working group, commissioned Professor Walter Penzhorn, formerly of the University of Pretoria, to review the following security systems and protocols:

- Kerberos;
- Secure European System for Applications in a Multi-vendor Environment (SESAME);
- Secure Electronic Transaction (SET);
- Password Encryption and Authentication Routine Library (PEARL); and
- Secure Sockets Layer (SSL).

The conclusions and recommendations of the review were [42]:

- Kerberos would be suitable option for securing XMLVend, since it provides all the necessary security services. Furthermore, since the source code is freely available, it can be easily adapted to the requirements of the proposed system.
- SESAME is based on the Kerberos protocol, and in a certain sense it may be regarded as a derivative of Kerberos. SESAME also makes use of a trusted third party, though the security architecture is more complex than that of Kerberos and was therefore not a recommended option.
- The secure electronic transaction (SET) standard is prohibitively complex, and extremely difficult to implement and maintain. Consequently, to date no vendor has implemented SET, and it seems highly unlikely that SET will ever become operational. Therefore, SET would not be feasible option for securing XMLVend.
- The PEARL protocol may be too time-consuming for clients with limited computational capability, such as a smart card or similar device, and may pose a serious bottleneck in the execution of the protocol in such environments. Furthermore, although PEARL was based open Encrypted Key Exchange (A-EKE) protocol, it was a PRISM⁵ proprietary implementation.
- SSL provides all the necessary security services required by XMLVend. SSL is also well established, widely used, easy to deploy and a trusted protocol to secure Internet connections. It does however have some disadvantages, such as being tightly coupled to transport layer protocol (TCP), does not support message level non-repudiation and

⁵ PRISM is a South African company and major player in the prepayment electricity vending industry. They provide outsourced vending solutions to several utilities, including Eskom.



has significant central processing unit (CPU) resource requirements. Despite these shortcomings, SSL would be a recommended option for securing XMLVend.

The author motivated and recommended that SSL / TLS would be the most appropriate security protocol for XMLVend to initially implement. This decision was motivated by:

- The recommendation of industry expert, Professor Walter Penzhorn and
- Recommendations of the WS-I Basic Profile [40]. See also section 4.4.1.6.

The working group supported the author's recommendation and SSL / TLS was adopted as the security protocol for XMLVend. The author then undertook an analysis of the SSL / TLS and its application to web services. This analysis was used to determine how best to specify the implementation of SSL / TLS to meet the requirements of XMLVend.

4.3.6.3 SSL / TLS overview

SSL is specified in an expired Internet Draft working document of the Internet Engineering Task Force (IETF). Transport Layer Security Version 1 (TLS 1.0) specification is an Internet Request for Comments specification (RFC 2246). TLS 1.0 is based on SSL 3.0 and the differences are not major. The differences are however significant enough that TLS 1.0 and SSL 3.0 do not interoperate. XMLVend supports both protocols, therefore the terms SSL and TLS are used interchangeably in this document.

SSL provides point-to-point security between a client and a server, connected over an insecure network. The security services offered by SSL include authentication of servers (always) and clients (optional) and message confidentiality and integrity (always).

SSL depends on public key cryptography to securely exchange secret keys. Public key cryptography uses two separated but mathematically related keys, a public key and a private key. Although both keys are related it is not possible by knowing one of the pair to discover or calculate the other. For example, a server would freely distribute its public key and keep the private key secret. Clients wishing to communicate securely with the server encrypt data with the server's public key and only the server will be able to decrypt the



data using its private key. This provides a secure and powerful mechanism for secret key exchange for web based applications.

Public key cryptography also provides a mechanism for authentication. A server wishing to prove its identity to a client encrypts some known data with its private key. The client decrypts the data with the server's public key and compares it to the known data. If the data match then the client is confident the data was received from the server since only the server has the private key that encrypted the known data.

The client however has one problem - how can the client be certain that public key really belongs to the server and not an entity pretending to be the server? This problem is solved using trusted digital certificates. A digital certificate is an electronic document in an X509v3 format [60] that contains the server's public key. The certificate is electronically signed by a third party. The third party guarantees that the certificate holder (server) is the owner of the corresponding private key. This mutually trusted third party is referred to as a certificate authority (CA).

Public key cryptography however has one major disadvantage. It is computationally intensive and therefore "slow" to process public key operations. SSL therefore takes a middle of the road approach. It makes use of public key cryptography for authentication and to exchange traditional share secret keys. The shared secret keys are then used to create a secure connection using symmetric cryptography.

SSL operates between the transport layer and the application layer. It provides security services to layer 5 protocols and associated applications (Figure 4.5). It is predominantly used by web based applications using hyper-text transfer protocol (HTTP). HTTPS is the acronym that identifies HTTP secured with SSL. This means that application level protocols requiring secure communications can operate as usual with the assumption the SSL layer will provide the required security services.

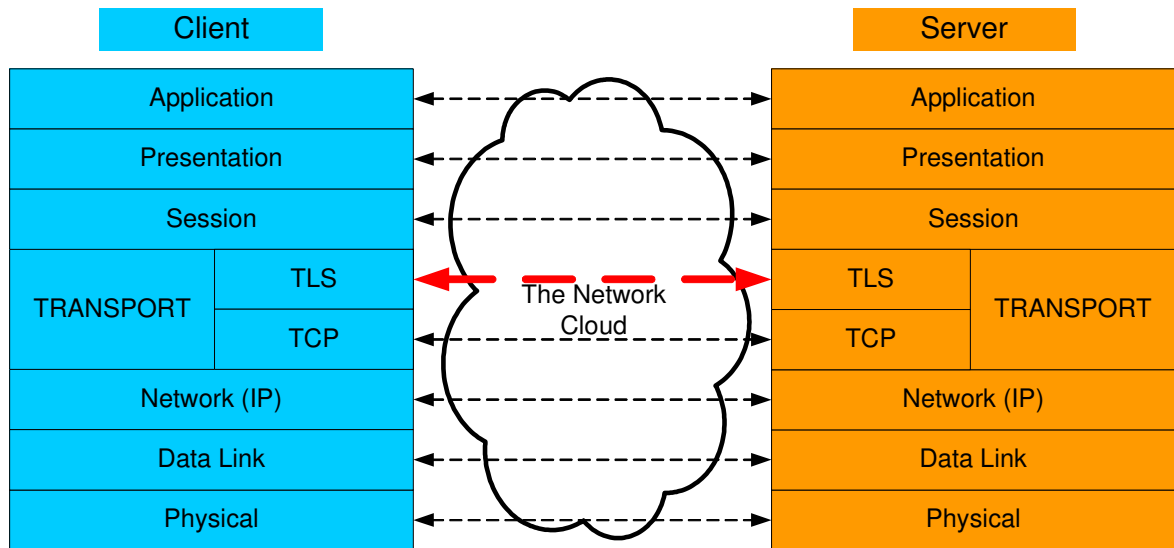


Figure 4.5 – SSL / TLS over TCP /IP

SSL makes use of a TCP connection to provide a reliable end-to-end network connection between client and server. A transport connection secured with SSL is referred to as a SSL connection. An SSL connection is characterised by security parameters like encryption keys, message authentication code (MAC) keys and cryptographic algorithms. The algorithms used are referred to as the ciphersuite. Multiple SSL connections can be part of SSL session. The SSL session is characterised by security parameters like ciphersuite and master key. All SSL connections belonging to a SSL session derive their security parameters from the session master key.

Figure 4.6 illustrates SSL consisting of two layers of protocols. The SSL record protocol provides basic security services to the three higher layer protocols. The three higher-layer protocols are:

- The handshake protocol;
- The Change Cipher Spec Protocol; and
- The Alert Protocol.

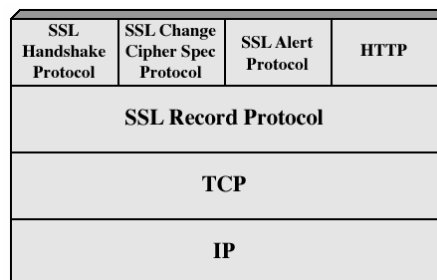


Figure 4.6 – SSL Protocol Stack [57]



Figure 4.7 shows the full exchange needed to establish a logical connection between client and server.

- **Client Hello (unencrypted):** The client sends a hello message to the server. The message defines the SSL version supported by the client, a random number used to seed the cryptographic computations, session-id, a list of ciphers and compression algorithms the client supports.
- **Server Hello (unencrypted):** The server responds with the protocol version, a random number to seed cryptographic computation, a session-id, a cipher and compression algorithm.
- **Server Certificate (unencrypted):** The server sends its certificate in a standard X509v3 format. The client uses the certificate to authenticate the server's identity.
- **Client certificate request (unencrypted) (optional):** The server sends the certificate request message to request the client present its certificate to the server.
- **Server Done (unencrypted):** The server done message indicates to the client that the server has completed its part of the initial SSL negotiation.
- **Client certificate (unencrypted) (optional):** The client then presents its certificate to the server.
- **Certificate verify (unencrypted) (optional):** This message is used by the server to authenticate the client.
- **Client key exchange (unencrypted):** This message provides the client shared secret keys to the server encrypted using the server's public key.
- **Client change cipher spec (unencrypted):** This message indicates to the server that all subsequent communication will be encrypted.
- **Client finished (encrypted):** This client message is encrypted using the agreed security parameters. The server must be able to successfully decrypt this message to continue with the communication.
- **Server change cipher spec (unencrypted):** This message indicates to the client that all subsequent communication will be encrypted.
- **Server finished (encrypted):** This server message is encrypted using the agreed security parameters. The client must be able to successfully decrypt this message to continue with the communication.

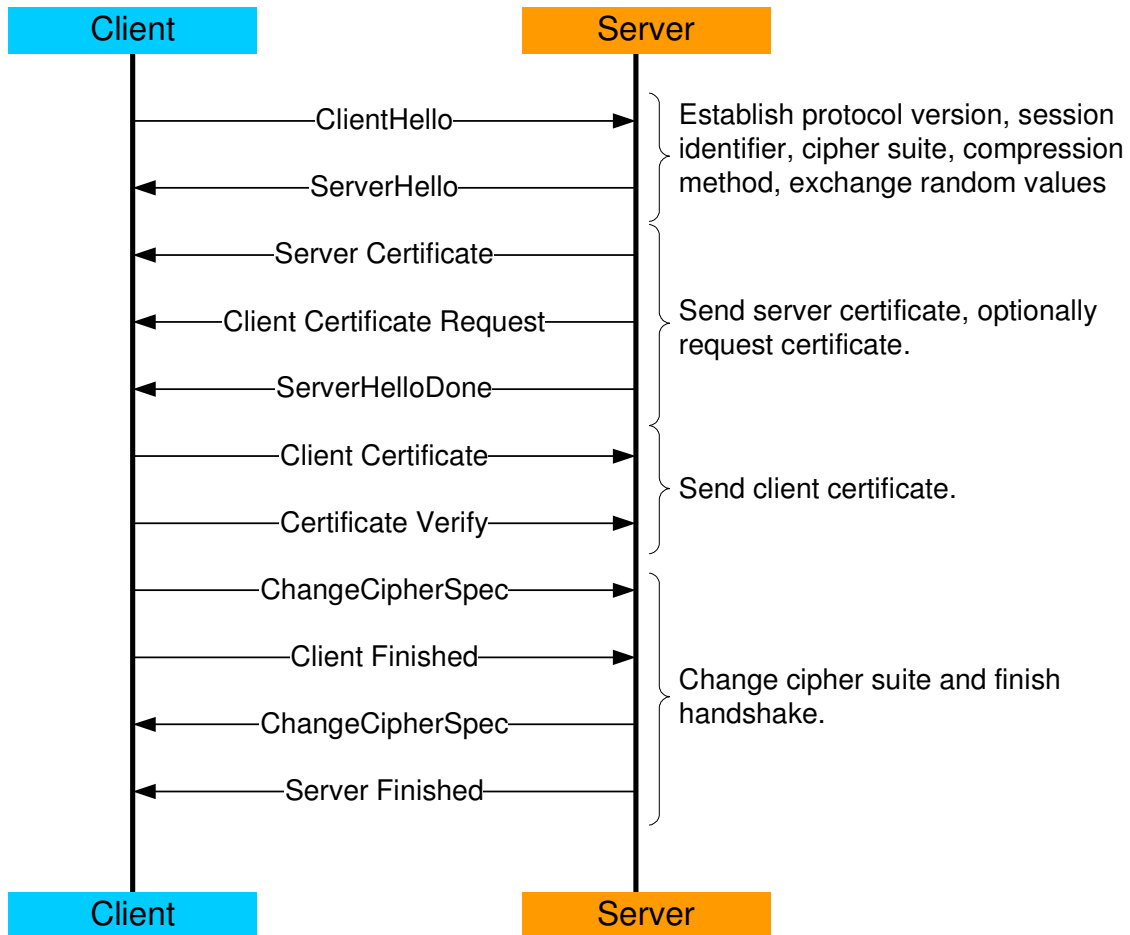


Figure 4.7 – SSL Handshake Messages

Once the parties are authenticated, the SSL secure record protocol ensures the confidentiality and integrity of application layer data being transmitted. The secure record protocol operates as follows (Figure 4.8):

- The applications layer data is fragmented into manageable blocks.
- The first fragment is optionally compressed.
- To ensure integrity of the fragment, a message authentication code (MAC) is computed using the negotiated algorithm and cryptographic keys. The MAC is then attached to the fragment.
- To ensure confidentiality of the data (fragment + MAC) it is encrypted using the negotiated encryption algorithm and cryptographic keys.
- The SSL record header is attached to encrypted data, and transmits the resulting unit in a TCP segment.
- The process is repeated for all remaining fragments.

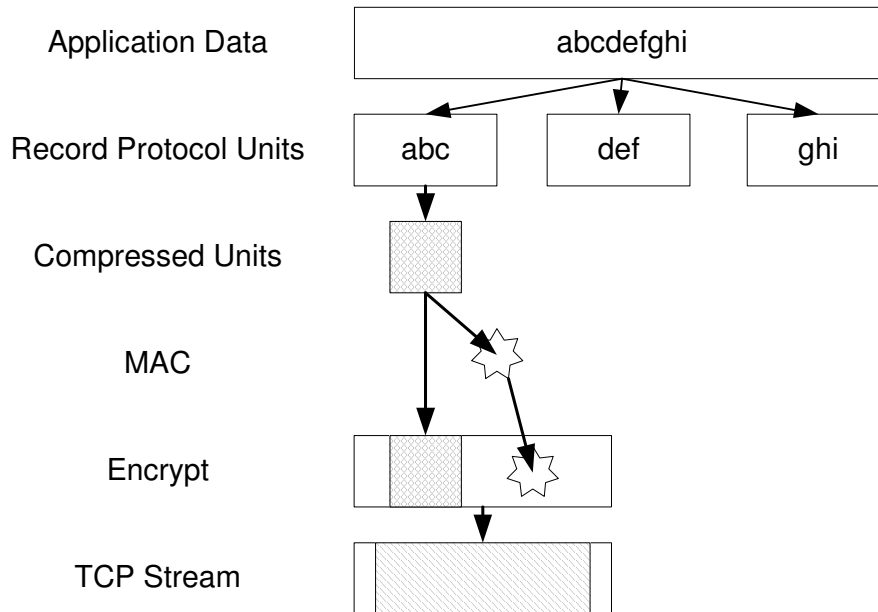


Figure 4.8 – SSL record operation protocol

4.3.6.4 XMLVend SSL / TLS recommendations

The analysis of SSL / TLS and its application to web services resulted in the following recommendations being specified in XMLVend. XMLVend would support the use of either TLS 1.0 or SSL 3.0 and adhere to the HTTPS constraints specified for the Synchronous Request / Response usage scenario in WS-I Usage Scenarios specification [40].

XMLVend would also mandate client authentication and recommends implementations utilise the certificate profile illustrated in Figure 4.9. Client applications should ensure that the client certificate common name (CN) field contains the same value as the client identifier parameter in their XMLVend request messages. If they do not contain the same value, the server should raise an exception and suspect possible malicious activity.



CN	Server or client identifier
O	Utility Commercial Name
OU	Utility Department
L	City
S	Province
C	Country (ZA)
altName	Address

Figure 4.9 – XMLVend SSL certificate profile

It is also recommended that private keys are stored in a secure location, such as a dedicated hardware storage device. Clients should make use of portable removable hardware storage devices, such as a smart card or similar device. Servers should make use of hardware security modules (HSMs) that offloads the SSL processing from the server.

The quality of SSL/TLS protection depends on an adequate SSL/TLS ciphersuite and key length being selected. Care must be taken in selection of ciphersuites and key lengths to prevent downgrade attacks. Ciphersuite options that have inadequate security should not be offered by client or server. It was recommended that XMLVend implementations only use ciphersuites recommended by [61].

4.4 XMLVEND AND WEB SERVICES

The XMLVend protocol design was realised within the web services framework. The web services framework was selected to meet the interoperability requirement between client and server implementations. Especially, since client and server implementations may use differing, platforms, operating systems, programming languages and be developed by different suppliers. However, to ensure maximum interoperability between client and server implementations it was important the XMLVend protocol define exactly how web service implementations should operate.



This section provides an overview of web services and related technologies. It then discusses XMLVend specific implementation details.

4.4.1 Web services overview

This section provides an overview of the web service framework technology suite. The literature review revealed several definitions for Web services but the definition that captures the essence of Web service is:

Web services are a standard platform for building interoperable distributed applications using open Internet based transport and messaging protocols.

A more technical definition by the World Wide Web Consortium (W3C) is [39]:

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Web services combine the best aspects of component-based development and Web technologies. Web services represent black box functionality that can be used without worrying about how the service is implemented. Unlike current component technologies, Web services are not accessed through object-model-specific protocols such as, Common Object Request Broker Architecture (CORBA), Distributed Component Object Model (DCOM) and Remote Method Invocation (RMI) [64]. Instead, Web services are accessed through ubiquitous Internet protocols and data formats, such as HTTP, SOAP and XML.

Web Services are therefore realised using the following web based technologies:

- Platform-independent format language for structured data exchange. This is achieved through the use of eXtensible mark-up language (XML).



- A way of describing the structure of the data being exchange. This is achieved through the use of XML Schema.
- A standard method of packaging the data for transmission over the communications network. This is achieved through the use of SOAP.
- A way for the Web services to describe their public interface to clients. This is achieved through the use of web services description language (WSDL) [63].
- A standard method of transporting the data across the network. This is achieved through the use of hypertext transport protocol (HTTP) and TCP/IP.

The following sections discuss each of these technologies in more detail.

4.4.1.1 eXtensible Markup Language (XML)

XML is a standard for describing data with a system of tags. XML defines the rules for constructing XML documents, defining and using tags, elements and attributes. XML is very simple yet powerful since the tags used to describe data, are user defined and not predefined as with mark-up languages like Hypertext Markup Language (HTML).

Therefore, XML empowers individuals, communities and industries to develop their own mark-up languages (vocabularies) that define their own specific domain or industry [62]. For example, MathML⁶, which is a mark-up language that is to represent mathematical formulae using XML tags.

XML is an effective mechanism to separate the data and the presentation layers, which allows data to be manipulated without the need to consider presentation layer issues. XML is platform independent which makes it easier for applications running on differing platforms to not only share data but also processes. In addition to being simple to create and parse, XML is platform and programming language independent. This characteristic of

⁶ MathML is intended to facilitate the use and re-use of mathematical and scientific content on the Web and for other applications such as computer algebra systems, print typesetting, and voice synthesis.



XML is the reason XML has been adopted as the de-facto data format for data sharing between disparate systems.

Listing 4.4 illustrates an XML representation of an FBE token issue.

```
<creditTokenIssue type="FBECredTokenIssue">
  <desc>FBE Token</desc>
  <meterDetail ti="07" krn="1" sgc="100611"
msno="06686069342">
  <meterType tt="02" at="07"/>
  <meterDetail>
  <token xsi:type="STS1Token">
    <stsCipher>12342345345645675678</stsCipher>
  </token>
  <units siUnit="kWh" value="50"/>
  <resource xsi:type="Electricity"/>
</creditTokenIssue>
```

Listing 4.4 – An XML representation of an FBE token issue

4.4.1.2 XML Schema

XML provides a simple way of representing data but it does not define the data types and how to validate the data. Web services is about data exchange between heterogeneous applications, however data exchange cannot be accomplished without a common, agree-upon type system that provides standard types and enables users to define their own types [53]. The W3C XML Schema or XML Schema Definition (XSD) is the standard that specifies built-in types and a language to define user specific types.

XSD is used to describe the structure of an XML document and is the type system of Web services. XSD defines a type of XML document in terms of constraints upon what elements and attributes appear in the XML instance document. It also describes the relationships of elements and attributes, their data types and possible constraints of the data.

XSD makes it possible for software to validate XML documents in order to ascertain whether it is a valid instance of that XSD. When services are developed in a development



language for example C#, VB.NET or Java, the data types used must be translated to XSD types to conform to Web Services standards.

Listing 4.5⁷ illustrates XSD for XML data in Listing 4.4.

```
<complexType name="MeterSpecificTokenIssue" abstract="true">
  <sequence>
    <element minOccurs="1" maxOccurs="1" name="desc"
type="Msg"/>
    <element minOccurs="1" maxOccurs="1" name="meterDetail"
type="MeterDetail"/>
    <element minOccurs="1" maxOccurs="1" name="token"
type="Token"/>
  </sequence>
</complexType>

<complexType name="CreditTokenIssue">
  <complexContent>
    <extension base="MeterSpecificTokenIssue">
      <sequence>
        <element minOccurs="1" maxOccurs="1" name="units"
type="Units"/>
        <element minOccurs="1" maxOccurs="1" name="resource"
type="Resource"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Listing 4.5 – FBE token issue - XSD

4.4.1.3 Simple Object Access Protocol (SOAP)

SOAP provides a standard mechanism to invoke services within the Web services framework. SOAP can be described as the protocol that facilitates process sharing between disparate systems. Similar to how XML facilitates data sharing between disparate systems. The SOAP specification provides defines the XML format for a SOAP message and how to communicate SOAP over HTTP.

⁷ Not all data definitions and are shown for simplicity purposes. The complete listing can be retrieved from <http://nrs.eskom.co.za/xmlvend>.



SOAP is actually a bit of misnomer as it implies that Web services representation is an object when in fact it does not have to be an object. A Web service can be written as a series of C functions and still be invoked using SOAP [53].

A SOAP message is a XML document consisting of the three main elements.

- The `<Envelope>` element is the top-level element of a SOAP message and wraps the actual contents of the message.
- The optional `<Header>` element contains information that does not form part of the message itself. Routing and security information are examples of information usually communicated in the `<Header>` element.
- The `<Body>` element contains the actual request or response message from the client or server respectively. It may also contain a SOAP `<Fault>` message.

The benefits of using SOAP are:

- It is becoming the standard for business-to-business integration.
- SOAP supports the packaging of auxiliary information with a service request. This could be auxiliary security or transactional information or additional information, which is relevant to some role players.
- It supports publication of services in registries (UDDI and ebXML registries) for lookup by clients.
- There are a number of frameworks, which automate the mapping from normal programming languages to SOAP. These include JAX-RPC from Sun, Axis from Apache, .Net from Microsoft and others.

Listing 4.6 illustrates XML data being transported in a SOAP message.



```

<?xml version="1.0"?>
<Envelope>
  <Body>
    <creditVendResp>
      .
      <creditTokenIssue type="FBECredTokenIssue">
        <desc>FBE Token</desc>
        <meterDetail ti="07" krn="1" sgc="100611"
msno="06686069342">
          <meterType tt="02" at="07"/>
          <meterDetail>
            <creditTokenIssue>
              .
            </r0:creditVendResp>
          </Body>
        </Envelope>

```

Listing 4.6 – FBE token issue - SOAP

4.4.1.4 Web Services Description Language (WSDL)

WSDL is a syntax and vocabulary for creating an XML-based interface to an application. WSDL specifies a standard format for specifying messages and data passed between a Web services client and server, and describes the semantics for these messages (that is, unidirectional or bi-directional, asynchronous or synchronous). It also includes directions for accessing a Web service using one or more transport protocols.

A WSDL document includes nine basic XML elements, which define and describe a Web service and how to implement or use it. Five elements are abstract in that they provide general information about what the Web service does and three are concrete, which means they format this information for a specific network protocol and address. The final element provides definitions relating to the service.

The `definitions` element is the first element in a WSDL document. It identifies namespaces and schemas used by Web services included in the document. A single WSDL document can define any number of Web services, although most documents cover only one service. The `definitions` element also specifies a “target” namespace to prevent name



collisions when one WSDL file imports information from other, physically separate WSDL files.

WSDL documents may include an `import` element. This optional element references information located in physically separate files. It is usually used to identify the location of an external schema.

The five abstract elements, `portType`, `operation`, `message`, `part` and `type` provide implementation independent information about what a Web service does. These elements function like abstract classes in an object diagram describe functionality but are never instantiated. The abstract elements are general descriptions of a Web service's functions. The abstract elements will be described from the largest to the smallest grained. However, the WSDL document defines the elements in the opposite direction, starting with `Type` and ending with `PortType`.

- `portType`: The `portType` element specifies a name and a set of operations or services associated with a specific Web service. As such, the port type is an interface that describes the Web service's operations or services, independently of any network protocols or physical locations. The idea is first to define what the Web service does and then provide specific access information such as protocols, bindings and addresses for locating and using the Web service.
- `operation`: Each `operation` element is an actual service offered by the `portType`. Like operations described with programming languages, port type operations have inputs, outputs and optional faults. WSDL uses four types of operations:
 - **One-way**, where the client sends a request to the server but does not receive a response.
 - **Notification**, where the server sends a request to the client and does not receive a response.
 - **Request/Response**, where the client sends a request to the server and the server returns a response to the client.
 - **Solicit/Response**, where the server sends a request to the client and the client returns a response to the server. WSDL 1.1 has no protocol binding for Solicit/Response.



- **message:** Operations use `message` elements to communicate information, such as inputs, outputs and faults. Each `message` element includes the name of the message and at least one part. Each input, output and fault has a corresponding XML element within the message.
- **part:** The `part` element separates the message content into logically separate data. Each part refers to an element of a specific data type.
- **type:** The `type` element identifies the data types such as strings, floats, structs or integers used by the operations and identify the address of the documents that define these datatypes. Datatypes can be simple or complex. By default, WSDL documents use standard W3C XML Schema (XSD) datatypes, but types defined in document type definitions (DTDs) or other schemas are also acceptable.

The three concrete elements, `service`, `port` and `binding` format information for use with network protocols and provide the address where this instance of the service exists. Concrete elements are described from the largest to the smallest but are also specified in WSDL document in the opposite order.

- **service:** The `service` element names the service and defines it as a collection of ports and bindings.
- **port:** Each port element identifies the name of the port, its address and its binding. The address format depends on the transport protocol specified by the binding; in most cases, it appears as a uniform resource location (URL).
- **binding:** The `binding` element connects, or binds, the Web service's operations to a specific transport protocol. The WSDL 1.1 specification provides pre-built bindings for SOAP 1.1, HTTP GET/POST and MIME protocols.

4.4.1.5 Hyper-text Transport Protocol (HTTP)

HTTP is an open standard that is administered by the World Wide Web Consortium (W3C) to transfer data between a client and a server. Although HTTP is widely implemented by web browsers to request hyper text mark-up language (HTML) pages from web servers, HTTP is not limited to browser applications. HTTP is also widely utilised to transport XML messages between web applications.



HTTP is a request / response application layer protocol and makes use of a transport control protocol (TCP) layer connection between client and server. The client is responsible for setting up the TCP connection. After the initial TCP exchange, the client sends it HTTP request and the server responds with the requested data. The server is then responsible for closing the TCP connection.

The first versions of HTTP required clients to establish separate TCP connections for each new HTTP request. For a client issuing high-volume HTTP requests such as an XMLVend gateway, this would create a significant performance bottle neck. That is, separate TCP connection must be created for each XMLVend request / response message pair.

HTTP version 1.1 eliminated the problem of multiple TCP connections by introducing connection persistence as a default mechanism. Connection persistence allows a client to use same TCP connection for all HTTP requests. Persistence requires the co-operation of client and server. The client must indicate that it supports persistence and the server must not close the TCP connection after the first client request. Persistence was also supported in HTTP version 1.0 but was not a default setting and was therefore not support by several implementations. Figure 4.10, illustrates the setup of a TCP connection, exchange of HTTP request and response messages and then closing of the TCP connection.

5	0.002138	172.24.41.39	172.24.41.32	TCP	1219 > 8080 [SYN] Seq=0 Ack=0 Win=64512 Len=0 MSS=146
6	0.002521	172.24.41.32	172.24.41.39	TCP	8080 > 1219 [SYN, ACK] Seq=0 Ack=1 Win=64512 Len=0 MS
7	0.002558	172.24.41.39	172.24.41.32	TCP	1219 > 8080 [ACK] Seq=1 Ack=1 Win=64512 Len=0
8	0.003411	172.24.41.39	172.24.41.32	HTTP	POST /xmlvend-server/service/ HTTP/1.1
9	0.100985	172.24.41.32	172.24.41.39	HTTP	HTTP/1.1 200 OK[Unreassembled Packet]
10	0.101721	172.24.41.32	172.24.41.39	HTTP	Continuation or non-HTTP traffic
11	0.101794	172.24.41.32	172.24.41.39	HTTP	Continuation or non-HTTP traffic
12	0.101821	172.24.41.39	172.24.41.32	TCP	1219 > 8080 [ACK] Seq=1076 Ack=2309 Win=64512 Len=0
13	0.183264	172.24.41.39	172.24.41.32	TCP	1219 > 8080 [FIN, ACK] Seq=1076 Ack=2314 Win=64507 Le
14	0.183713	172.24.41.32	172.24.41.39	TCP	8080 > 1219 [ACK] Seq=2314 Ack=1077 Win=63437 Len=0
15	0.183845	172.24.41.32	172.24.41.39	TCP	8080 > 1219 [FIN, ACK] Seq=2314 Ack=1077 Win=63437 Le
16	0.183897	172.24.41.39	172.24.41.32	TCP	1219 > 8080 [ACK] Seq=1077 Ack=2315 Win=64507 Len=0

Figure 4.10 – HTTP Traffic - XMLVend Request / Response

HTTP defines one message format for request messages and one for responses. Figure 4.11 illustrates the HTTP request message format.

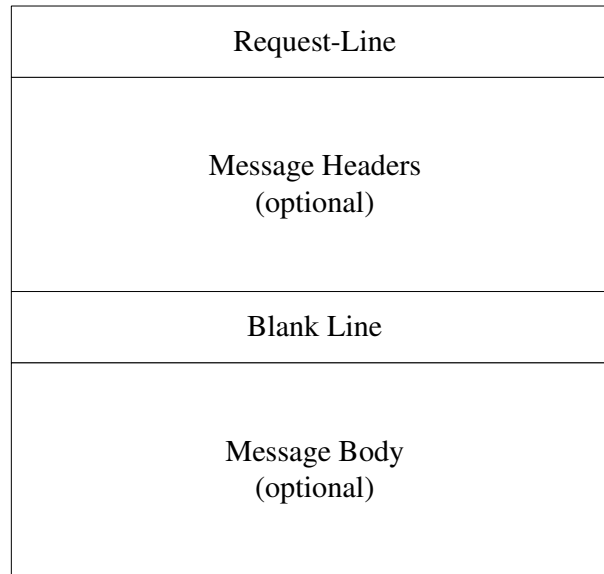


Figure 4.11 – HTTP request message format

It starts with the request line, which indicates the method, the resource being requested and the HTTP version the client supports. Each separated by one or more spaces.

HTTP defines four basic methods:

- GET
- POST
- PUT and
- DELETE.

GET is the simplest HTTP operation and used to request a resource from the server, identified by a uniform resource identifier (URI). This is usually a uniform resource location (URL) of the web page being requested. The POST method allows the client to submit information to the server. This method is used by SOAP to submit request to the server. The PUT method is used to transfer objects to the server, its most common purpose is to upload files to the server. The DELETE method is used to delete objects from the server.

After the request-line, the request may include one or more lines of message headers. Three headers types are defined:

- General headers apply to HTTP communications in general.
- Request headers apply to the specific request and



- Entity headers apply to the message body.

A blank line follows the headers. If the request contains a message body, it follows directly after the blank line. The blank line identifies the end of the request message headers.

Listing 4.7 is an example of HTTP POST request containing an XMLVend request message.

```
POST /xmlvend-server/service/ HTTP/1.1
SOAPAction:
User-Agent: Jakarta Commons-HttpClient/3.0
Host: 172.24.41.32:8080
Content-Length: 905
Content-Type: text/xml

<?xml version="1.0"?>
<Envelope>
  <Body>
    <creditVendReq>
      .
      .
      .
    </creditVendReq>
  </Body>
</Envelope>
```

Listing 4.7 - HTTP Request - XMLVend Request

HTTP responses look like HTTP requests (Figure 4.12).

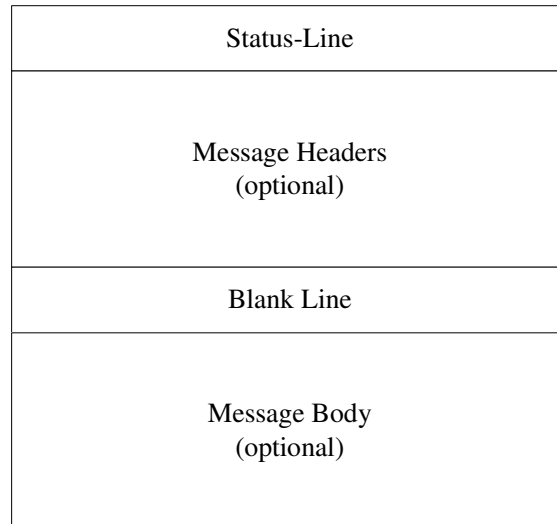


Figure 4.12 – HTTP response message format

The only difference is the request-line is replaced by a status-line. The status-line indicates the highest HTTP version supported by the server, the status code and the reason phrase. The status code is a three digit number that indicates the result of the request. The reason phrase is a human readable phrase describing the status code. The status-line is then followed by the message headers and then the message body. Listing 4.8 shows a HTTP response containing an XMLVend response message.

```
HTTP/1.1 200 OK
Content-Type: text/xml
Transfer-Encoding: chunked
Date: Thu, 02 Nov 2006 14:07:04 GMT
Server: Apache-Coyote/1.1

<?xml version="1.0"?>
<Envelope>
<Body>
  <creditVendResp>
    .
    .
    .
  </creditVendResp>
</Body></Envelope>
```

Listing 4.8 – HTTP Response - XMLVend Response

HTTP is also a very attractive protocol since most firewalls allow traffic through the HTTP port, 80. Therefore, Web services do not need special firewall configurations to make use



of Web services that are hosted on the other side of firewalls. Such firewall “support” is particularly useful in business to business applications.

4.4.1.6 Web service – Interoperability (WS-I) compliance

The primary challenge to widespread Web service adoption has been lack of interoperable web service implementations. There are three main reasons for these interoperability problems:

- Ambiguity among the interpretations of the existing web service standards;
- Differences among specifications that have yet to gain widespread adoption; and
- Insufficient understanding of the interactions among specifications.

Information Technology (IT) leaders behind the web service specifications realised that interoperability is in the best interest of all industry participants. In early 2002, key industry leaders created the Web Services Interoperability Organisation (WS-I), an industry organisation focused on promoting Web Services Interoperability across platforms, operating systems and programming languages.

WS-I is dedicated to accelerating the adoption of Web services by assisting in the selection and interpretation of Web services specifications. It also develops common best practices for web service development, deployment, and integration of business applications.

Rather than creating new specifications, WS-I aims to meet its interoperability goals through the following deliverables:

- Web Services Profiles that specify collections of specifications, along with clarifications of the ambiguities, so that they can be adopted in an interoperable fashion.
- Web Services testing and implementation guidance to accelerate deployments.
- Developments and encouragement of web services best practices, usage scenarios, use case and sample implementations that illustrate how web services profiles can be applied to solve interoperability challenges.
- Creation of self-administered and self-validating test suites for conformance testing to WS-I profiles.



The WS-I profile, called Basic Profile (BP) version 1.0, was completed in April 2004. The BP states that conforming Web services must use the following web service specifications SOAP 1.1, Web Services Description Language (WSDL) 1.1 and Universal Description Discovery and Integration (UDDI) 2.0. The BP provides valuable clarifications, refinements, interpretations and amplifications of those specifications to promote interoperability.

In conjunction with the Basic Profile guidance, the WS-I has also released the WS-I Usage Scenarios [41]. Scenarios are independent of any application domain. They describe fundamental web service design patterns that can be combined and built upon like building blocks. The Synchronous Request/Response scenario is of particular interest to XMLVend as it supports the protocol's message exchange pattern (see section 4.3.2).

A complete sample application based on a supply-chain management scenario (complete with use cases, design, and implementation files) has also been made available by WS-I. In addition, the WS-I has provided a suite of testing tools for verifying BP conformance in user applications. The tools developed monitor the interactions with a web service, record those interactions and analyze them to detect implementation errors. The web service itself is treated as a "black box". The testing tools do not interact with the web services, nor do they have any view of the supporting code or infrastructure.

The guidance provided by the BP is organized into four categories: messaging, service description, service publication and discovery, and security. However, the bulk of the guidance is focused on the proper design of WSDL definitions as most web service toolkits map between WSDL definitions and programming language methods. It is incompatibilities at this level that usually cause significant interoperability problems.

The BP states that WSDL must be used to describe web services. The WSDL specification was originally designed to accommodate a wide variety of scenarios and development styles. WSDL was intentionally designed to be open and flexible with numerous options and choices available to developers. The WS-I BP has addressed this situation by removing many of the choices. For example, it clarifies many of the ambiguities revolving around the proper use of WSDL and XML Schema "import" elements and "targetNamespace" attributes. It clarifies how to deal with such things as parameter order, one-way operations, response message wrapper elements, fault descriptions, and message validation [54].



The BP also imposes several restrictions on the WSDL specification. The BP disallows:

- The use of type systems other than XML Schema.
- Solicit-response and notification operations
- Operation name overloading.
- MIME and HTTP GET/POST bindings
- Transports other than HTTP.
- Encodings, including the SOAP encoding.
- Mixing styles within a single binding and
- Requiring custom WSDL extensions.

The clarifications and restrictions of WSDL, supports the WSDL-first development approach [35] since it gives the developer complete control over WSDL design. Starting development with the WSDL ensures the WSDL is WS-I compliant before coding the solution. This prevents expensive changes later in the development cycle to ensure interoperability.

Also the restriction to document type encoding supported the notion that most messaging systems are loosely coupled using document exchange. A messaging client thinks in terms of sending a message (documents) to the server and receiving a response message (document). Messaging is focused on the format of the documents exchanged rather than remote objects interface. Focusing only on the message formats and by designing extensible messages, the client and server are less tightly coupled than in the case with RPC [53].

The BP also adopts, but does not mandate use of web services secured with either TLS 1.0 or SSL 3.0 (HTTPS). That is, conformant Web services may use HTTPS; they may also use other countermeasure technologies or none at all.

4.4.1.7 Putting it all together

A Web service application may include several logical layers incorporating functions such as the Web service instance and application business logic. This section discusses the

interaction of each layer of the web service stack (Figure 4.13). Each layer of the Web services stack represents one of the fundamental functional areas of a Web service instance.

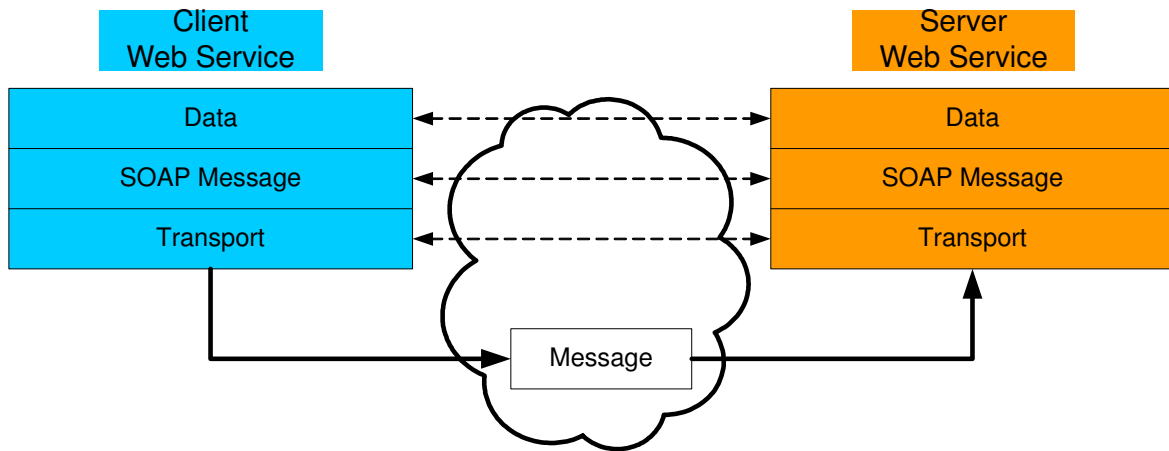


Figure 4.13 – Web services stack

A set of activities is defined for each layer of the Web service stack (Table 4-4). Activities are the fundamental operations that comprise a Web service. A single activity has several constraints applied to it from the Basic Profile [40]. For example, the “Send HTTP” activity is constrained by the guidelines specified in the SOAP 1.1 and HTTP sections of the Basic Profile.

Layer	Activity
Data Layer	Write XML Process XML
SOAP Message Layer	Write SOAP envelope Process SOAP envelope Write SOAP body Process SOAP body Write SOAP header Process SOAP header
Transport Layer	Send HTTP Receive HTTP

Table 4-4 – Activities grouped by Web services stack layer

- Data Layer:** The data layer translates the application specific data into the model chosen for the specific Web service. The data layer includes the functions necessary to support flexible data typing. This layer maps to the `wsdl:types` and `wsdl:message` definitions within a WSDL, which in turn map to the types and elements defined the schema documents. The data layer is the most important layer with respect XMLVend protocol because the data layer describes the XMLVend messages pairs.



- **SOAP Message Layer:** The SOAP message layer is the infrastructure that processes SOAP messages, dispatches them, and may optionally fulfil quality of service requirements. On the sending side the message layer writes SOAP messages, based on the data model defined in `wsdl:portTypes` and `wsdl:message` definitions with the WSDL document. On the receiving side the message layer processes the SOAP messages and dispatches requests to the correct application or method.
- **Transport Layer:** The transport layer sends and receives messages. For the BP, this includes only HTTP client and server platforms. This layer maps to the `wsdl:binding` and `wsdl:port` definitions with the WSDL document.

4.4.2 Defining XMLVend as a web service

This section discusses the key aspects required to define XMLVend within the web service framework for maximum interoperability. It discusses where and how the protocol will operate within web services framework. Support for message compression is also discussed.

4.4.2.1 The XMLVend web service protocol stack

Figure 4.14 illustrates the XMLVend Web Services Protocol Stack. XMLVend utilises all protocols as specified WS-I with the addition of GZIP (section 4.4.2.5).

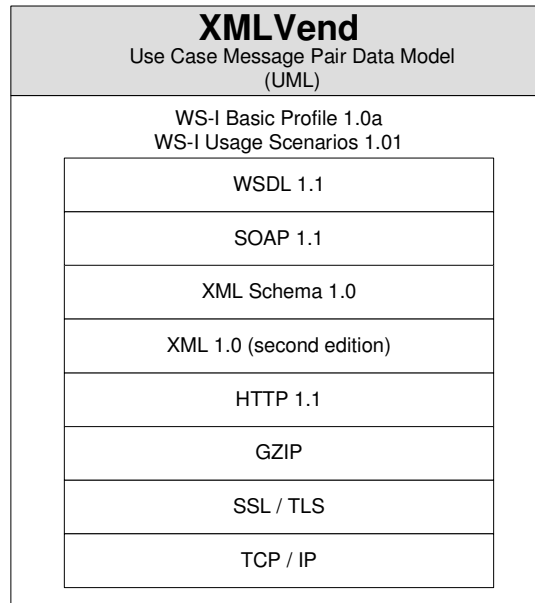


Figure 4.14 – XMLVend Web services Protocol Stack

4.4.2.2 XMLVend web services process flow

The XMLVend message pairs exchanged between XMLVend Client and Server are implemented using Synchronous Request / Response as described in WS-I Usage Scenario [41].

Figure 4.15 illustrates the high-level interactions between an XMLVend Client and Server using Synchronous Request/Response Usage Scenario:

- The XMLVend Client invokes an XMLVend use case by sending an appropriate XMLVend request SOAP message bound to an HTTP request to the XMLVend Server.
- The XMLVend Server executes the service and sends an appropriate XMLVend response SOAP message bound to an HTTP response to the XMLVend Client.

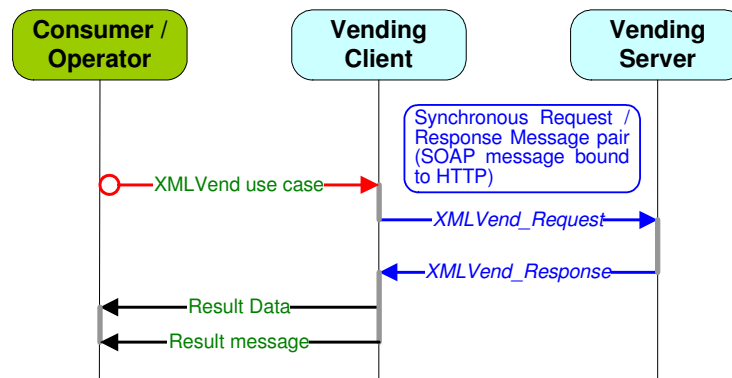


Figure 4.15 – Synchronous Request/Response Sequence Diagram



Figure 4.15 illustrates the XMLVend request flow. Each bulleted item represents the activities performed within one layer of the XMLVend Web service stack required to complete the flow. The client activities are:

- Data Layer:
 - Write XML.
- SOAP Message Layer:
 - Write SOAP envelope
 - Write SOAP body
- Transport Layer:
 - Optional Compress HTTP Body Message
 - Send HTTP

The server activities are:

- Transport Layer:
 - Receive HTTP
 - Uncompress HTTP Body Message (if required)
- SOAP Message Layer:
 - Process SOAP envelope
 - Process SOAP body
- Data Layer:
 - Process XML. The data payload is processed according to the data model and dispatched to the server application.
- Application Layer:
 - Process Security Credentials.
 - Process business rules.

4.4.2.3 XMLVend fault messages

As described in section 4.4.2.2, a standard XMLVend fault response message is defined to communicate application layer errors to the client. This section describes implementation of the XMLVend fault response message as a standard SOAP fault message. SOAP faults must adhere to constraints and behaviour as prescribed by [41].



The specific application error information will be compiled into the XMLVend Fault Response message (section 4.4.2.2) and included into the `<detail>` element of a standard SOAP `<fault>` element.

Listing 4.9 is an example of an XMLVend SOAP fault. The standard SOAP fault is shown in **bold** font and the XMLVend specific fault message is shown in *italic* font as a child element `<detail>`.

```
<?xml version="1.0"?>
<Envelope>
  <Body>
    <soap:Fault>
      <faultcode xmlns="">soap:Server</faultcode>
      <faultstring xmlns="">Technical: Duplicate message ID
detected.</faultstring>
      <detail xmlns="">
        <b0:xmlvendFaultResp>
          <b0:clientID ean="6004708001" xsi:type="b0:EANDeviceID" />
          <b0:serverID ean="3210987651" xsi:type="b0:EANDeviceID" />
          <b0:terminalID ean="00000001" xsi:type="b0:EANDeviceID" />
          <b0:reqMsgID uniqueNumber="2" dateTime="20061014225101" />
          <b0:respDateTime>2006-10-15T12:08:01</b0:respDateTime>
          <b0:dispHeader>ONLINE VENDING ERROR</b0:dispHeader>
          <b0:operatorMsg>Duplicate Msg detected.</b0:operatorMsg>
          <b0:fault xsi:type="b0:BusinessRuleEx">
            <b0:desc>Duplicate message ID detected.</b0:desc>
          </b0:fault>
        </b0:xmlvendFaultResp>
      </detail>
    </soap:Fault>
  </Body>
</Envelope>
```

Listing 4.9 – XMLVend fault message

4.4.2.4 Defining the XMLVend Specification

The XMLVend specification was defined in three related documents, that is, WSDL, schema and specification document. The first draft WSDL was created with the assistance of Actaris. The process used was:



- Microsoft .Net classes were created based on the XMLVend message designs.
- The WSDL was auto-generate using a .Net framework utility.
- The WSDL was then manually edited by Syntell to be WS-I compliant.
- It was then edited by the author to include the fault response message.

From this point forward the WSDL was manually edited. The manual editing had the following advantages:

- They were not dependant of a particular toolset.
- They did not include parameters that were particular to a specific toolset or platform.
- The message formats were controlled and predictable.
- They were always WS-I compliant.

The WSDL was circulated to the working group and web service consultants, Solms Training and Consulting (STC) for review. STC made the following recommendations [44]:

- Data validation constraints should be applied to schema elements. That is, appropriate data type, data ranges and regular expression patterns could be applied at the message layer data elements to improve data integrity before being processed at an application layer.
- Ordering of the elements in WSDL file should match the message design. This would ensure consistency between the message design and messages realised in web services.
- The Cancel Token web service was not implemented.

The author then updated the WSDL, incorporating the STC comments.

4.4.2.5 Message compression

The verbose nature of XML and therefore the XMLVend messages lead some working group members to challenge the use of XML for the online vending protocol. Sighting significantly higher costs for data processing, data storage and data transmission [67]. A message compression strategy was therefore pursued to counter this challenge.



Initially the working group debated whether the message compression should be implemented at the SOAP or HTTP layers. Implementing compression at the SOAP level would mean the XMLVend would need to define the compression implementation details since SOAP did not support compression natively. HTTP on the other hand provided a built-in mechanism to support HTTP body compression. The working group agreed to utilise HTTP since it was a proven, simple and widely supported message compression technique.

The working group also defined GZIP as the supported compression algorithm. GZIP was selected since it was natively support by HTTP and it was also well supported by popular development environments such as, Java and the Microsoft Dot Net. GZIP has been proven to be an acceptable general purpose compression algorithm for XML data and web service implementations [65][66].

HTTP messages will optionally be compressed (in both directions) by making use of the standard HTTP content encoding mechanism [58]. This is achieved by the client setting the appropriate HTTP request headers e.g.:

- Content-Encoding: gzip
- Accept-Encoding: gzip

The vending server should respond with a 415 (Unsupported Media Type) message if it does not understand the encoding indicated by the **Content-Encoding** request header. The client may then retry sending the message without applying compression.

The **Accept-Encoding** request header indicates that the client will accept "gzip", encoded data; otherwise the data will not be compressed by the server. **Note:** This compression occurs at an HTTP layer, and is **not** analogous to compressing the SOAP body within the SOAP envelope.

Using the XMLVend 1.22 client and server reference implementations the impact of GZIP message compression was analysed. Figure 4.16 illustrates the comparison of compressed versus uncompressed messages for the various use cases.

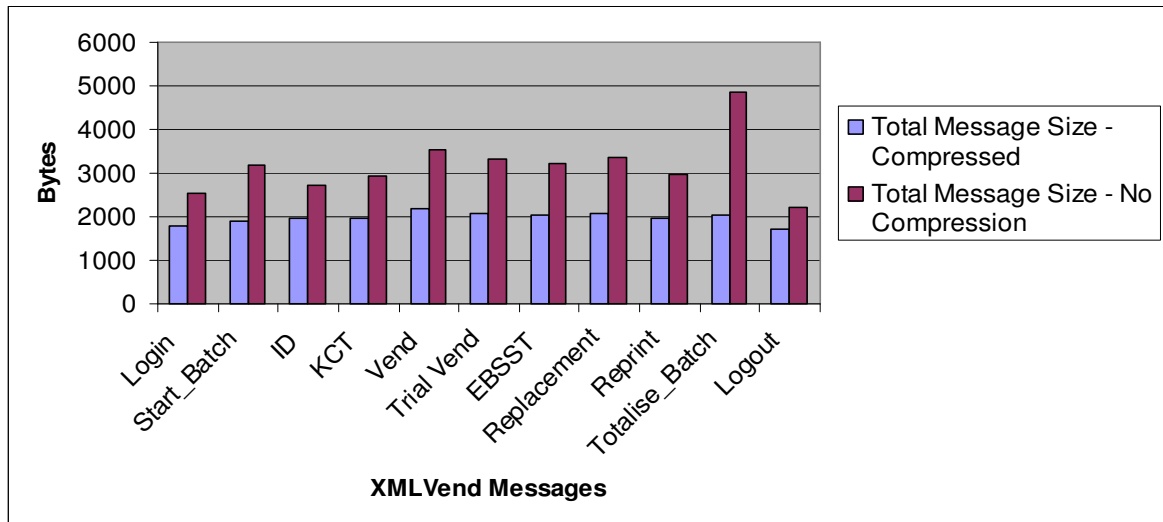


Figure 4.16 – Compressed versus Uncompressed XMLVend 1.22 Messages

The average size of uncompressed and compressed XMLVend 1.22 request / response messages was 3167 bytes and 1975 bytes respectively. GZIP produced an average saving 1192 bytes or 38%. This analysis revealed that GZIP significantly reduced the size of the request / response messages pairs.

Figure 4.17 illustrates that XMLVend messages consist of 47% overhead introduced by HTTP and SSL. The average overhead across all XMLVend request / response messages was 53%. The analysis of the protocol overheads illustrated that the HTTP headers contributed significantly to the overall size of XMLVend messages.

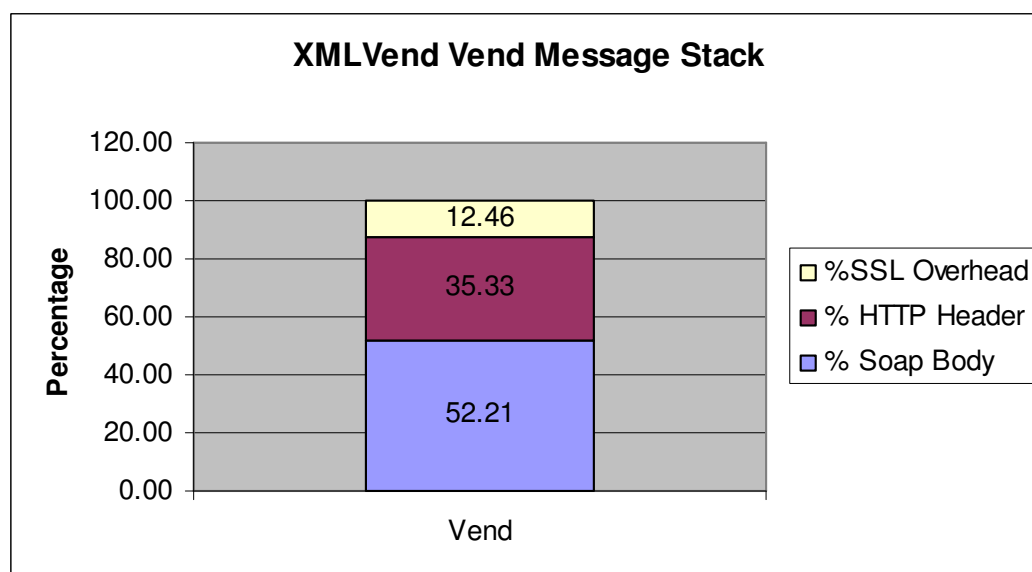


Figure 4.17 – Vend Message Overhead 1.22 Protocols



4.4.2.6 XMLVend 1 release and piloting

The XMLVend 1.0 release included the specification document (NRS 009-6-10) and XMLVend test suite. The test suite consisted of a reference client and server implementation and a test tool to validate XMLVend message compliance.

XMLVend 1.0 was updated 3 times and finally stabilised at version 1.22. Table 4-5, lists the XMLVend 1.0 updates.

Version	Updates
1.0	First release.
1.1	<ul style="list-style-type: none"> • Moved all message type definitions from the WSDL file into a separate schema file. • Corrected naming inconsistency of MsgID element. • The constraint of the NRSTrack2DataT data type was updated to be strict as specified in [29]. • Support was added for other resource types other than electricity. • A currency symbol was added to currency values. • The presence of appropriate elements were used determine the mechanism was used to identify a customer or meter. • The Trial vend and Key change token use cases were added. • All element and attribute multiplicities in schema were updated to be consistent with the message design.
1.21	<ul style="list-style-type: none"> • A customer address element was added to applicable response messages. • A specialised key change token was created to better model a key change token pair.
1.22	<ul style="list-style-type: none"> • The currency symbol attribute was removed and replaced with currency symbol element. • The “New” key change data was made optional in the key change request message.

	<ul style="list-style-type: none"> The schema “Qualification” attribute was set to "qualified" for elements and attributes to ensure correct validations.
--	--

Table 4-5 – XMLVend Version 1 Updates

XMLVend 1.1 was implemented by Actaris and Contour and tested as part of Eskom’s online vending project tender evaluation process. It was also implemented by Actaris as a live pilot in Eskom’s Western region.

XMLVend 1.22 was implemented by Actaris as part of the Eskom’s national pilot project at 6 sites across South Africa [47]. The XMLVend test tools and reference implementations proved invaluable during the Actaris system development and testing.

Figure 4.18, shows the Eskom Western region pilot site.



Figure 4.18 – Eskom Western region pilot site

The pilot installations fully tested and exercised the XMLVend compliant vending system. Although some minor XMLVend issues were identified it was acceptable for piloting purposes.



The Eskom pilot made use of the reversal advice message to “reverse” transactions that were not completed by the client. The pilot identified that advice reversals may open the utility to abuse for the following reasons:

- It is the responsibility of vending clients to issue advice(reversal) requests. It is therefore impossible for the utility to distinguish between advice(reversal) requests that are genuine and those that are fraudulent.
- The server depends on decisions made by the client to issue reverse transactions.

The pilot also emphasised the need for accurate meter configuration identifiers (SGC, KRN and TI) on the vending server database. That is, the meter data stored on the server database needs to be the same as the meters configuration in the customer’s premise. However, since the server’s meter data did not always match the meter’s actual configuration this resulted in a significant number of tokens not working in customer’s meters. The problem was further exacerbated since pilot customers only required their meter serial numbers to purchase tokens and the rest of the meter data was obtained from the server database to generate the tokens.

The only way to get the problematic token to work in the customers meter was to update the customer’s meter to match the server meter data. This required customers to be issued with key change tokens for entry into their meters before getting the problematic tokens to work in their meter. This caused major confusion and frustration to customers. Especially since customers had to be return to the vending point for the key change token to be issued.

The pilot also highlighted the need for vending clients to use reliable communications infrastructure when communicating with the vending server. If the communication infrastructure fails vending stops and this results in long queues and very unhappy customers (Figure 4.19).



Figure 4.19 – Connectivity problems cause long queues

The working group recommended the release of XMLVend 1.22 as a “trial use” specification and to continue working towards XMLVend 2. XMLVend 2 would incorporate the lessons learnt from XMLVend 1.22 and include the additional functionality.

4.5 SUMMARY

In this chapter the XMLVend version 1 analysis, design and development was discussed. The following key protocol design outcomes were achieved by the author:

- 15 use cases were defined.
- The design of the use case request and response message pairs was defined using the XMLPay notion for data models.
- A standard fault message was defined to communicate fault scenarios to the client.
- The Issue advice use case message pair was defined to address message delivery reliability.
- The synchronous request / response message exchange pattern was used to realise the use cases.
- The use of SSL / TLS with mutual client and server authentication to secure the protocol communication.



- The protocol design was realised as a web service implementation. This resulted in the development the XMLVend WSDL and schema.
- GZIP was also defined as an optional compression algorithm for XMLVend message pairs.

The protocol stabilised at XMLVend 1.22 after three iterations. It was extensively piloted and tested. It proved functionally sound and was approved for “trial use” by the working group. However, the XMLVend 1.22 design and web service implementation was not object oriented. The design did not promote reusability, flexibility, extensibility and sustainability. This key design issue would have to be addressed in XMLVend 2 to ensure its migration beyond a “trial use” protocol. The working group therefore decided to continue working towards XMLVend 2 incorporating the lessons learnt and functionality omitted from XMLVend 1.22.

CHAPTER 5 : DEVELOPMENT OF XMLVEND VERSION 2

5.1 INTRODUCTION

After the stabilisation of XMLVend 1.22 the author started work on the XMLVend 2. The focus of the XMLVend 2 development was:

- Add new use cases.
- Update and correct errors existing use cases and message pairs.
- Redesign the XMLVend interface and message structures to promote “good design”, reusability and interoperability.
- Improve understanding and maintenance of the specification.
- Realise the XMLVend web service based on the “contract first” methodology.

This chapter discusses the XMLVend 2 requirements. It is followed by the interface design analysis, message design process, exception handling and message reliability mechanism. Finally the implementation of the message design as web service is discussed.

5.2 XMLVEND 2 REQUIREMENTS

5.2.1 Use cases

XMLVend 2 would include all XMLVend 1.22 use cases with the exception of the redundant Login and Logout use cases. This section discusses the new use cases and the enhancements to existing use cases.

5.2.1.1 New use cases

The new use cases were:

- **Create Deposit Slip Use Case:** This use case generates (prints) a Vendor deposit slip with a server recommended deposit amount. It is implemented by online systems that



operate in an upfront vending mode and therefore do not require vendor credit management through batches.

- **Account Payments Use Case:** This use case enables customers to make account payments at vending points, such as debt recovery payments or service charges (but not limited to).
- **Vendor Credit Statement Use Case:** This use case displays (prints) the vendor's credit updates and current available credit. A server configuration determines the number or period of entries to be displayed. This function empowers the vendor to reconcile his deposits with those that have been processed on the online server, without the need to contact the utility.
- **Customer Fault Reporting Use Case:** This use case enables the customer to report meter faults or customer queries at an online vending point. This function empowers customer's that may not be able to travel to a utility walk-in centre or telephone the utility call centre. Further, it ensures that relevant information is captured from the customer at the vending point, such as meter details and customer details.
- **Update Meter Key Data Use Case:** This use case only allows a key change to the meter configuration stored on the online vending database. This ensures that a risk free key change token is always issued. This use case is a more controlled version of the XMLVend 1.22 "Key Change Token" use case. Further, it is enhanced to allow for a matching power limit token to be issued with the key change token if required.
- **Meter Specific Engineering Token Use Case:** This use case is aimed at the utility technician and is used to request meter maintenance tokens called , meter specific STS engineering tokens, as defined in NRS009-6-7 [20]. Generating the meter specific engineering tokens from the vending server ensures that meters are maintained and configured from a single data source, that is, the online vending database. This prevents the need for independent and possibly erroneous data sources being used to configure and setup meters.
- **Non-Meter Specific Engineering Token Use Case:** This use case is aimed at the utility technician and is used to request meter maintenance tokens called, non-meter specific STS engineering tokens, as defined in NRS009-6-7 [20]. These tokens are not meter specific and can be used in any meter since they do not change the meters configuration but displays the meters configuration parameters.



5.2.1.2 Enhancement to existing use cases

Enhancements to existing use cases were:

- **Reprint Transactions Use Case:** This use case was named “Reprint token” in XMLVend 1.22 and only allowed the reprint of prepaid token transactions. However with the introduction of account payments use case, the reprint use case would need to support prepaid token transactions and account payment transaction. Therefore in XMLVend 2.1 it has been enhanced to reprint both prepaid token transactions and account payment transactions like debt recovery.
- **Purchase Credit Token Use Case:** This use case has been enhanced to allow multiple transactions to be returned as part its response message. This enhancement was specifically aimed at supporting debt recovery as part of normal purchase credit token request.
- **Key Change Token Use Cases:** Use cases returning key change tokens have been enhanced to allow for a matching power limit token to be issued (if required).
- **Issue Advice Use Case:** XMLVend 1.22 specified the following advice message types:
 - Reversal and
 - Confirmation

In order to avoid the potential issues discussed in section 4.4.2.6, a new use case scenario has been added, “Advice Last Response”. The advice last response request is used to automatically request a reprint of a transaction, when a response message is not received by the client. The advice last response confirms if the transaction was completed on the server or not before attempting a new transaction. It also prevents clients from just reversing transactions.

5.2.2 Design requirements

The key design requirement pursued by the author for XMLVend 2 was to implement “good design” principles throughout the development process. “Good design” is defined as follows [34]:



- **Responsibility localisation:** A design with good responsibility localisation is often referred to as a design with a high level of cohesion. In such a design each component adheres to the **single responsibility principle**; each component thus has only a single responsibility at some level of granularity and all its attributes and services are narrowly aligned with its responsibility.
- **Clean layers of granularity:** This very important aspect of good design enables one to work effectively at various levels of granularity. The layers should adhere to the dependency inversion principle, that is, components in a lower level of granularity should not have any dependency on higher-level components. Also, one should be able to understand a higher-level workflow without having to understand the finer details. At any level of granularity the responsibilities should be well defined and the workflow should be self-contained and comprehensible.
- **Decoupling:** Decoupling provides a high level of flexibility and improves maintainability. If one component uses another component we effectively have a client-server relationship. Generally clients would not want to lock into a particular service provider. Instead, the client defines the requirements in a contract. The contract specifies the services which service providers need to provide (the interface), the pre- and post-conditions for those services and the non-functional requirements.
- **Simplicity:** Always keep it simple. Complexity results in increased development costs, risk and maintenance costs. A design which is understandable and conceptually intuitive is preferable above one which is difficult to explain and non-intuitive.
- **Architecture and technology neutral:** The design should remain valuable over a long period. To this end the design should be able to survive technologies, changes in access mechanisms and architectural changes. This is usually achieved by following the guidelines of Object Management Group's (OMG) Model Driven Architecture (MDA), which suggests that the core design should be technology and architecture neutral. It should be possible to map the design onto various technologies and architectures.

Adhering to the above design principles provides the following benefits [34] to the XMLVend protocol:

- **Improved Understanding:** Understanding is promoted by simplicity, good responsibility localization, intuitive naming, and the ability to view workflows at various levels of granularity.



- **Reusability:** Reusability is really a direct consequence of good responsibility localisation, together with a component based approach, where components realize well defined contracts. Classes which address a particular combination of responsibilities relevant for a particular problem are not generally re-usable. On the other hand, classes whose services address only a single domain of responsibility and whose behaviour are well defined in a contract are generally much more likely to be re-usable.
- **Testability:** Testability is facilitated through specifying a contract for each component at any level of granularity.
- **Maintainability:** Simplicity, responsibility localisation which results in localised maintenance, the ability to effectively work at different levels of granularity, decoupling, testability and reusability, all contribute to making a system maintainable.
- **Longevity:** A design which is architecture and technology neutral can survive changes in technologies and architecture. Furthermore, all the design principles which assist maintainability contribute also significantly to the longevity of the design.

5.3 DESIGNING THE XMLVEND 2 PROTOCOL

Although web service standards specify how to reveal the interfaces and the syntax of message definitions for the XML documents that they send and receive, they do not specify the conceptual design of those services and their enabling documents. Therefore, it is essential that conceptual design of the messages be complete before the messages realised in a specific technology like web services. This supports the approach of using documents as loosely coupled interfaces and hiding implementation details, underlies the idea of service-oriented architectures (SOA) as a way to create new applications as services by integrating or combining components of others.

The XMLVend 2 message design process was based on the URDAD system design methodology for service provider contracts [34]. URDAD is an iterative design methodology which designs the realisation of a use case across layers of finer and finer granularity, each layer with well defined responsibilities. URDAD requires that for each responsibility there should be a contract against which all service providers (components) which realise the responsibility can be tested.



5.3.1 Modelling the XMLVend protocol

5.3.1.1 Unified Modelling Language (UML) Interfaces

UML interfaces define:

- A mechanism for decoupling from any particular service provider implementation.
- It specifies the services or operations provided by the interface and the messages that are exchanged for each operation.

UML interfaces can be thought of as contracts developed from the client perspective. The contract may have functional and non-functional requirements. The non-functional requirements include scalability, usability, reliability, security, performance and other non-functional features. These are typically specified as quality of service requirements. The functional aspects are defined for each operation using a design-by-contract framework with:

- **Pre-Conditions:** The service provider is entitled to refuse the service without breaking the contract if any of the pre-conditions is not met. However, if all the pre-conditions are met, the service provider is obligated to provide the requested service. Otherwise it is a breach of the contract. For example, a purchase credit token service may have a pre-condition that the supplied meter serial number must be valid. If the meter number is invalid the service can refuse to process the request without breaking the contract.
- **Post-Conditions:** The post-conditions are deliverables of the service provider. These include the return value or values, but may also include service provider state information. For example, the post-condition of the purchase credit token will include the requirement to debit the vendor's available balance and return a token to the client.
- **Invariance constraints (if required):** These are symmetry rules around the service provider's state. That is, if at any stage the invariance constraints are not met, then the service provider is in an invalid state. For example, for the vendor account, the invariance constraint could be that sum of the credits minus the debits must always produce the current available balance. If this does not hold true, then the service provider is in an invalid state.

Implementations of an interface are service providers or servers. Users of an interface or components that invoke an interface are referred to as clients. Therefore a client may use any server that implements the interface, decoupling the client from implementations of the interface and hence preventing service provider locking.

5.3.1.2 Modelling the XMLVend interface

The XMLVend protocol can be naturally modelled as a UML interface or contract for prepayment vending services. Therefore XMLVend clients that use the XMLVend interface would be able to request services from all service providers (servers) that implement the XMLVend interface. It also decouples the service provider from specific client realisations as long the clients use the XMLVend interface. That is, the server will service requests from any client that uses the XMLVend interface as long as the interface constraints are met.

Figure 5.1 illustrates the XMLVend protocol at the highest-level modelled as a UML interface. The XMLVend client uses the XMLVend interface to invoke XMLVend operations from a server that implements the interface. The XMLVendOperation represents the generic XMLVend operation. In practice only specialisations of XMLVendOperation will be invoked.

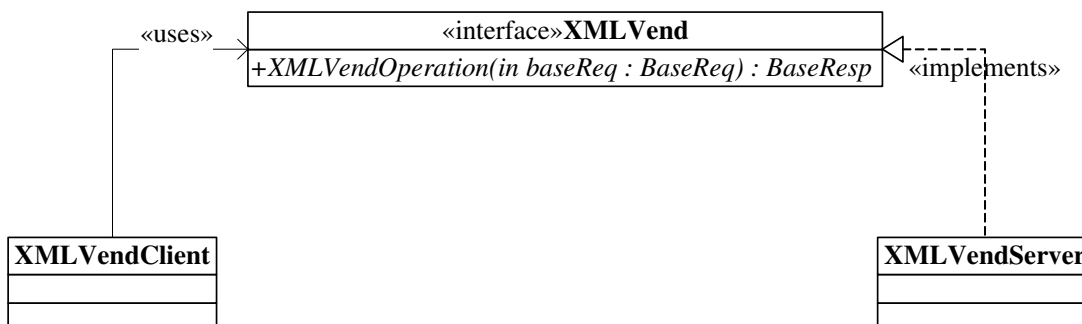


Figure 5.1 – Generic XMLVend UML Interface Diagram

At this level of abstraction the XMLVend interface is defined as follows:

- Pre-Conditions:



- The client input parameters adhere to interface specified input parameters format and data constraints.
- The client device must be uniquely identified and legitimate.
- Individual messages must be unique.
- The request message has met service provider specific business rules.
- **Post-Conditions:**
 - The operation has been processed successfully.
 - The response message adheres to the interface response message format and data constraints.
 - Or an exception message has been sent to the client, indicating the cause of the exception.
- **Invariance constraints:** None identified.

At this level of abstraction, the XMLVend client and server are only responsible for sending and returning parameters that are generic to all requests and response messages. This is also referred to as the base request and response messages. The client's responsibilities are:

- Compile the base request message.
- Send the request message to the server.
- Receive the server response message.
- Process the base server response message.

The server's responsibilities are:

- Receive and validate the base client request message.
- Process the client request message.
- Compile the base response message.
- Send the response message to the client.

5.3.1.3 Modelling the generic input and output parameters

UML class diagrams are a natural and platform neutral mechanism to model input (request) and output (response) parameters. Figure 5.2, shows a class diagram template used to model input (request) and output (response) parameters.

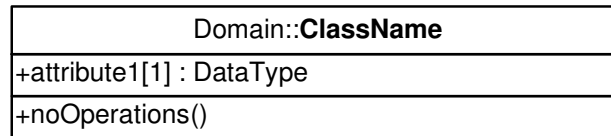


Figure 5.2 – Class diagram template

The class name is shown in the top rectangle. It is prefix with a use case domain separated by double colons (“:”). The next rectangle contains the attributes of the class. Attributes represent the parameters being communicated. The plus sign (“+”) prefix the attribute name and depicts the visibility of the attributed, that is, public (“+”) or private (“-“). All attributes are defined as public since they are communicated between client and server and therefore visible to both client and server. The attribute name appends multiplicity value in square brackets (“[]”). The multiplicity value depicts the cardinality of the attribute. The multiplicity value is post-fixed with data type of the attribute separated by a single colon (“:”). The next rectangle contains operations of the class. The class diagram models input and output messages and since messages communicate only data (attributes) no operations are defined.

The naming convention adopted by the XMLVend 2 protocol is the Camel convention. Camel convention is the standard object-oriented naming convention for UML as well as many object-oriented programming languages like Java and SmallTalk.

The rules for Camel convention are:

- Class Names start with capital letters.
- Everything else, including object and method names start with lower case.
- Word boundaries are capitalised for all scenarios.

Modelling the XMLVendOperation generic input (request) and output (response) parameters as UML class diagrams, produces the Base::BaseReq and Base::BaseResp classes illustrated in Figure 5.3 and Figure 5.4 respectively. Similar to the XMLVendOperation the BaseReq and BaseResp classes will never instantiated without being specialised. Therefore these classes are defined as abstract.

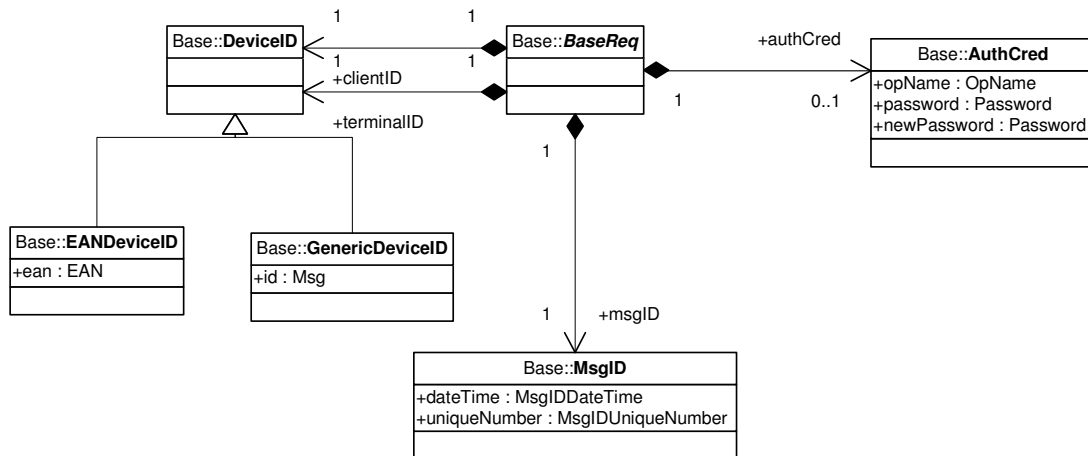


Figure 5.3 – BaseReq Class Diagram

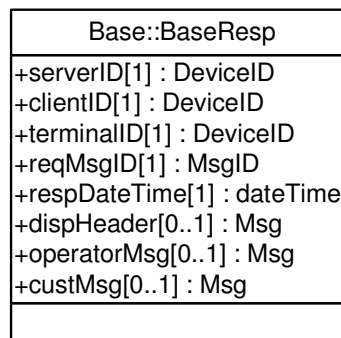


Figure 5.4 – BaseResp Class Diagram

Figure 5.3, illustrates that the `BaseReq` has `clientID`, `terminalID`, `msgID` and `authCred` (optional) as parameters. `clientID` is the client identifier that identifies the client device requesting a service from the server. The server uses the `clientID` value to authenticate the device and determine whether the client is authorised to invoke the service. The `clientID` would also map to the entity (person or organisation) that is contracted by the utility to sell electricity. Such entities are referred to as vendors.

`clientID` is specified as an abstract type, `DeviceID`. Two specialisations of `DeviceID` are specified, `EANDeviceID` or `GenericDeviceID`. The implementing utility is expected to decide on the most appropriate format. It is however recommended



that `clientID` be defined as `EANDeviceID` types as EAN numbers have several benefits, such as being globally unique.

The `terminalID` identifies the device that initiated the request in a gateway implementation model. However XMLVend only provides the `terminalID` for information purposes and is not dependant on this data to service the request. In the normal vendor implementation model the `terminalID` defaults to “1”.

`MsgID` is used to uniquely identify the request message. It is the client’s responsibility to generate a unique `MsgID` and a pre-condition of the server. The `MsgID` is composed of two attributes, a `dateTime` and a `uniqueNumber`, to ensure uniqueness.

`authCred` is an optional parameter. That is, it is not a pre-condition to be present in the message for the server to process the request. However, `authCred` supports implementations that may require individual client operators to be identified or authenticated by the server. Such a pre-condition is not handled by the interface but rather as an implementation specific business rule.

Figure 5.4 illustrates that the `BaseResp` returns the following fields: `serverID`, `clientID`, `terminalID`, `reqMsgID`, `respDateTime`, `dispHeader`, `operatorMsg` and `custMsg`. The `serverID` identifies the specific server that serviced the request. This supports a load balancing architecture where it may be useful to identify the actual server that serviced the request. The `clientID`, `terminalID`, and `reqMsgID` provide the request message context information. The `respDateTime` is the server date time stamp of when the response message was created.

The `dispHeader` is an optional field. It is usually used returned to values that must be printed at the top of a receipt or displayed at the top of a user interface. It is the responsibility of the implementing utility to specify the contents of the `dispHeader`. The `operatorMsg` and `custMsg` are also optional fields. The `operatorMsg` and `custMsg` fields are returned when the utility would like to communicate specific messages to the client operator and customer respectively. These fields are predominantly used for communicating exception information. It is the utility’s responsibility to define specific operator and customer messages for its specific business rules.

5.3.1.4 Modelling the concrete XMLVend operations

The first step in modelling the concrete XMLVend operations is to list and define the supported use cases. Figure 5.5 illustrates the complete list of XMLVend 2 use cases, categorised by the user initiating the use case.

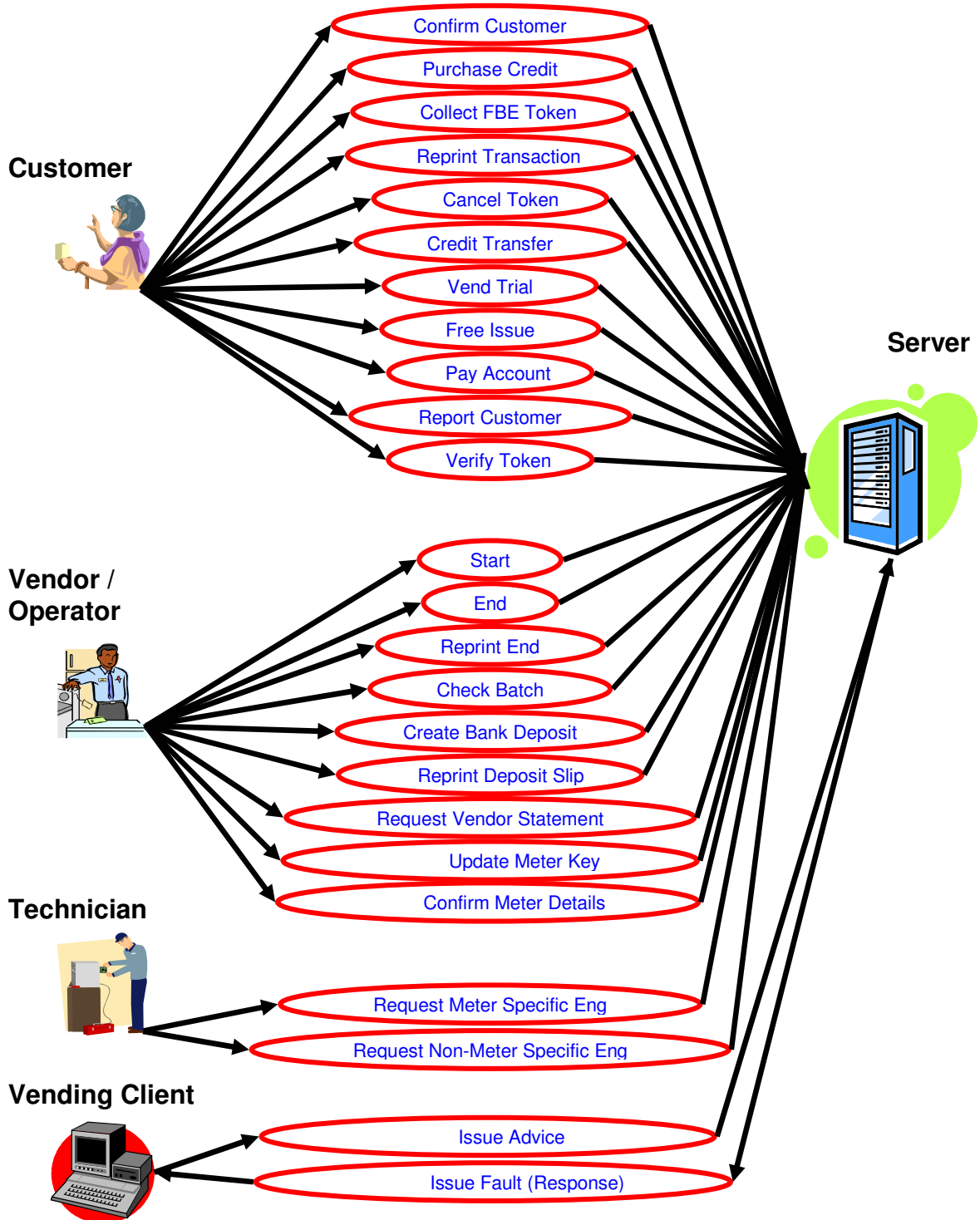


Figure 5.5 – XMLVend 2 Use Cases



It is also proved useful to categorise the use cases three implementation domains. The grouping of the use cases into implementation domains reduced complexity and improved understanding. It also eased implementation and maintenance of the protocol. The implementation domains are:

- Revenue Management;
- Meter Management; and
- Base.

The revenue management domain use cases are concerned with credit vending, revenue collection and management. The 17 use cases associated with this domain are:

- Confirm Customer Details;
- Purchase Credit Token;
- Collect FBE Token;
- Reprint Token;
- Cancel Token;
- Trial Purchase Credit Token;
- Meter Credit Transfer;
- Pay Account;
- Verify Token;
- Free Issue Token;
- Start Batch;
- End Batch;
- Reprint Batch;
- Check Batch Totals;
- Create Deposit Slip;
- Reprint Deposit Slip; and
- Issue Vendor Statement.

The meter domain use cases are concerned with meter management functions. The 5 use cases associated with this domain are:

- Confirm Meter Details;
- Report Customer Fault;
- Update Meter Key Data;



- Non-Meter Specific Engineering Tokens; and
- Meter Specific Engineering Tokens;

Use cases in the base domain are used to support use cases in the revenue and meter management domains. The only use case defined in the base domain is the Issue Advice Use Case. Further, the issue fault response is not strictly a use case. It is used to return fault information to the client when any of the use cases can not be successfully completed on the server.

The use cases map to interface operations, which are as concrete specialisations of the abstract XMLVend interface operation (see section 5.3.1.2). Therefore, 3 XMLVend interfaces are defined based on the XMLVend use case domains.

- XMLVendBaseService is an abstract interface that specifies the AdviceRequest operation only. This interface is parent interface of the revenue and meter interfaces.
- XMLVendRevenueService is the revenue interface that specifies the revenue domain use case operations (Figure 5.6) and is public interface.
- XMLMeterService is the meter interface that specifies the meter domain operations (Figure 5.7) and is a public interface.

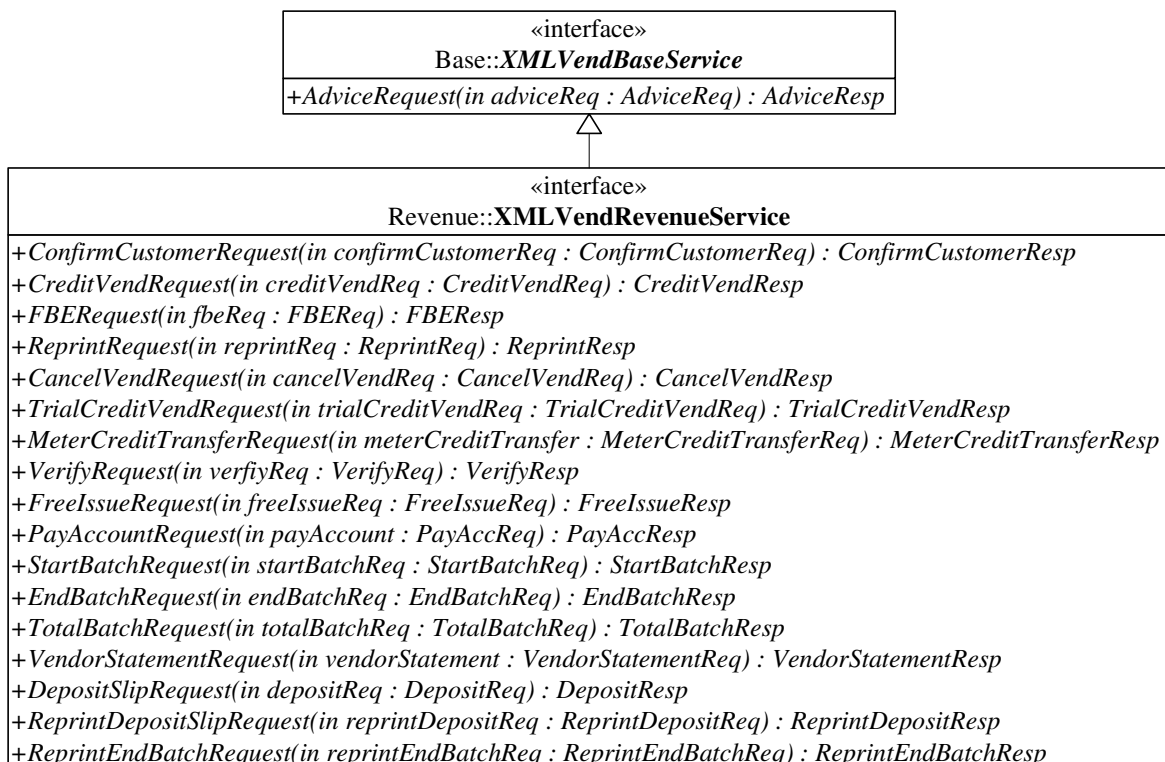
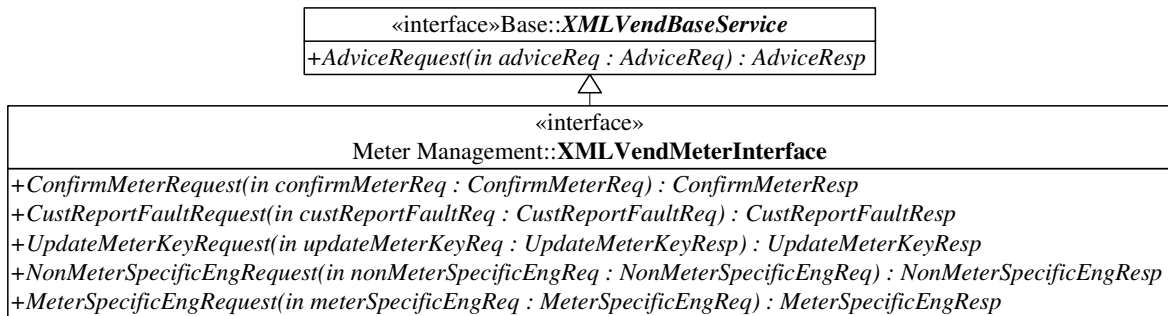


Figure 5.6 – XMLVend Revenue Interface**Figure 5.7 – XMLVend Meter Interface**

Each use case was analysed and expanded at the interface level of abstraction. That is, the pre-conditions, post-conditions and invariance constraints (if any) were defined. Table 5-1 illustrates this process for the Purchase Credit Token use case. Refer to [52] for all the use case definitions.

Description	This use case is used to purchase prepaid credit tokens. The value of the credit tokens may be expressed as currency value or in kilowatt-hours or kiloliters (as applicable). This use case corresponds to the “Prepayment sale” transaction (type 000) described in [28].
Desired Outcome	The customer pays for and receives the purchased credit token.
Dependencies	<ul style="list-style-type: none"> • Follows: - • Includes: Issue Advice, Issue Fault (if required) • Followed by: -



Pre-conditions	<ul style="list-style-type: none"> • The XMLVend server supports this use case. • The server business rules are met. • An identification parameter supported by the server for this use case must be supplied, such as: <ul style="list-style-type: none"> ○ Meter card (track 2 data), ○ Meter serial number or, ○ Meter configuration data, usually from an old token. • The request token value in currency or number units must be specified. • The resource required must be specified.
Post-conditions	<ul style="list-style-type: none"> • The server security module (SM) generates the requested credit token. • A “Prepayment sale” transaction is recorded on the server. • The customer obtains the requested credit token. • Note: The server may also return an FBE token, Free Issue and / or a Pay Account Transaction together with a credit token (if so configured). • A valid post-condition is also the return of an exception response message.
Invariance Constraints	<ul style="list-style-type: none"> • None.
Participants	<ul style="list-style-type: none"> • Customer • Vending Operator • XMLVend Client • XMLVend Server

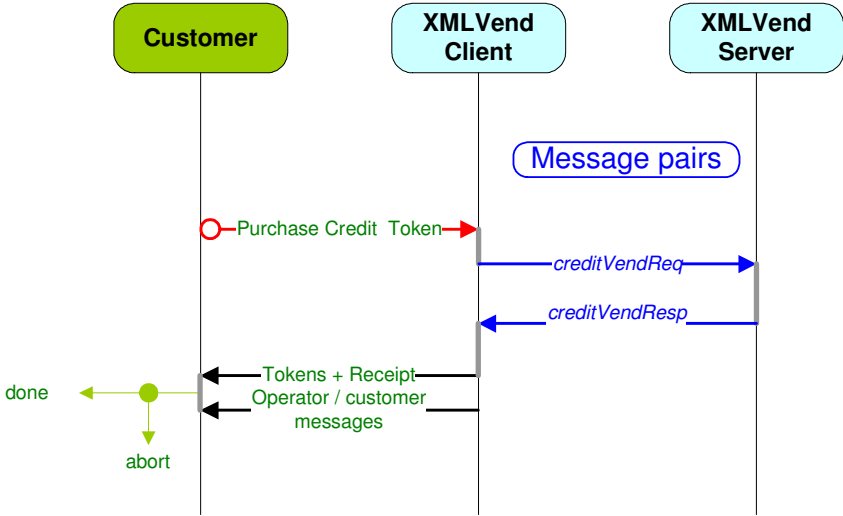
<p>Happy Path</p>	<p>The happy path scenario is illustrated below.</p>  <p style="text-align: center;">Figure 5.8 – Purchase Credit Token Sequence Diagram</p>
<p>Implementation</p>	<p>The server may also be configured to return more than one token with the requested credit token under certain circumstances, for example, an FBE token, Free Issue Token and / or Pay Account Transaction.</p> <p>It is recommended that this use case utilise the Issue Advice use case if the an exception occurs, to ensure that client and sever have the same understanding of the use case’s outcome on both server and client.</p>

Table 5-1 – Purchase Credit Token Use Case Definition

5.3.1.5 Modelling the XMLVend use case message pairs

The XMLVend 1.22 use case definitions and message pair data models were used as a starting point to model the XMLVend 2 use case message pairs as UML class diagrams. The use case definition, the use case sequence and activity diagrams were critical activities in analysing and understanding the domain and requirements of the use case message pairs. An iterative design process was used with regular reviews and verification with prepayment vending domain experts until a draft design was produced. The draft designs were reviewed and updated by object-orientated design experts, to ensure “good design”.



This process was used to analyse all the use cases and model their respective request and response message pairs.

The rest of this section illustrates the modelling process using the Purchase credit token use case message pair as an example. Figure 5.9 illustrates the Purchase credit token request message. The shaded class represents the root class of the diagram. The diagrams can be read in top-down fashion, except where the complexity of the diagram prevented this layout.

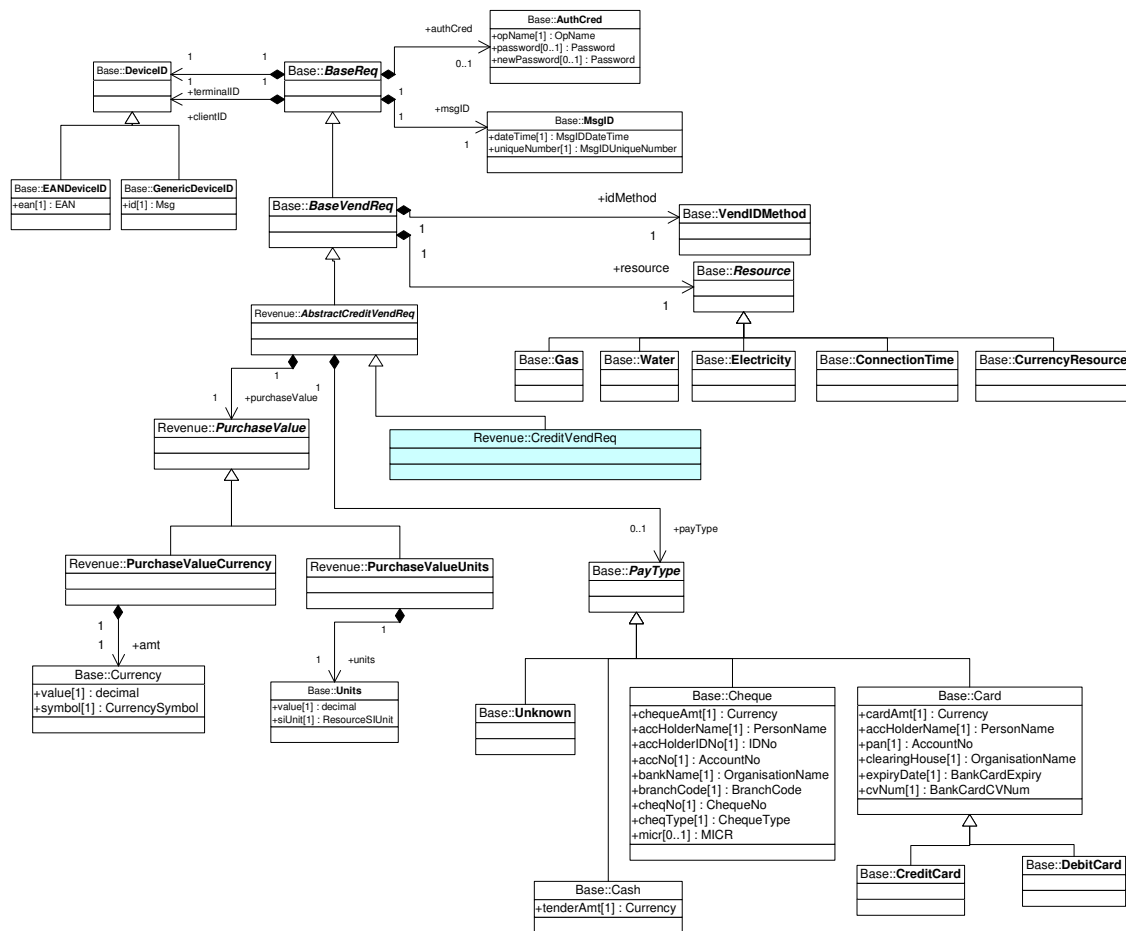


Figure 5.9 – Purchase Credit Token Request Message Model

The root class is CreditVendReq which is a specialisation of the following abstract classes AbstractCreditVend, BaseVendReq and BaseReq classes. This also means that CreditVendReq inherits all the parameters contained in these abstract classes.

idMethod, captures the mechanism used to identify the meter. resource, specifies the type of resource requested for example, electricity. The idMethod and resource are



encapsulated in the BaseVendReq abstract class. BaseVendReq is a useful class since all use case requests that vend tokens will automatically inherit the idMethod and resource parameters.

The AbstractCreditVend abstract class contains the purchaseValue (currency or units) and payType parameters. payType is optional parameter, which captures the payment mechanism used by the customer.

AbstractCreditVend is a useful class since vend requests that require a purchaseValue (currency or units) can be specialised from this class. Vend requests that do not require payment are specialised from BaseVendReq, for example the Free Basic Electricity (FBE) request message.

Figure 5.10 illustrates the Purchase Credit Token use case response message. CreditVendResp is a specialisation of BaseVendResp and BaseResp.

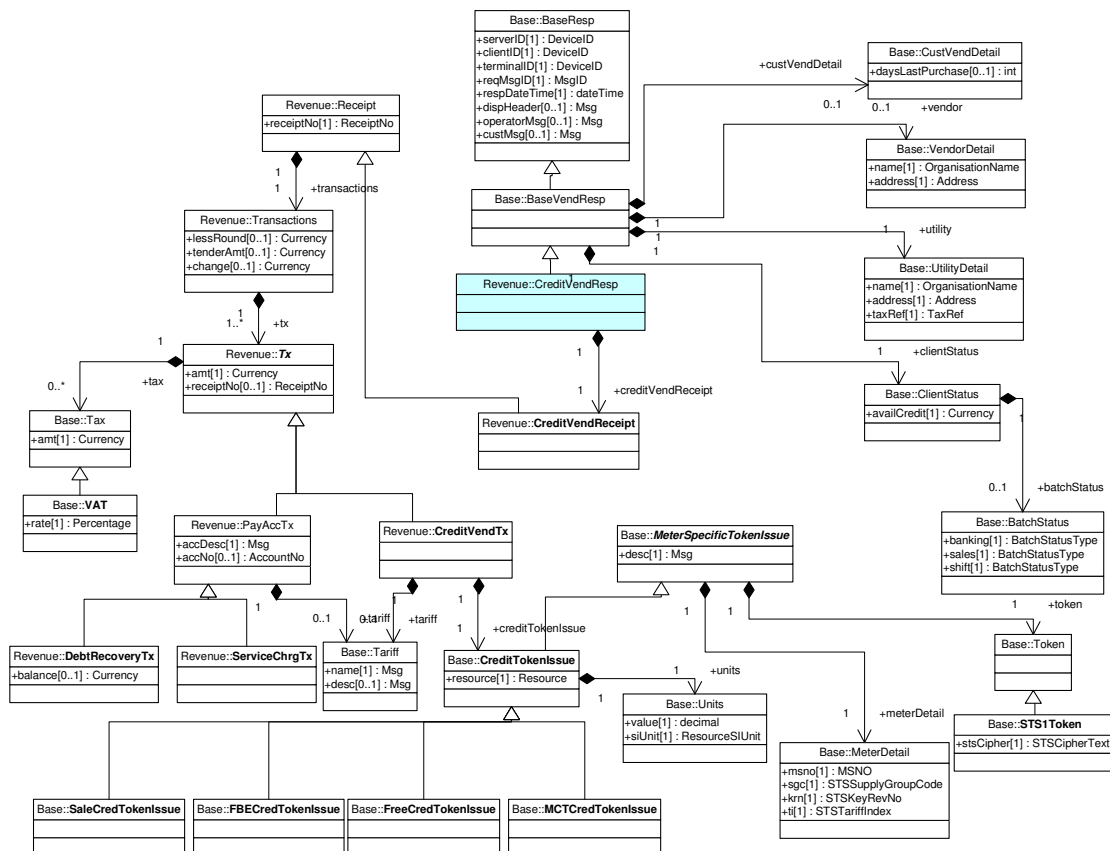


Figure 5.10 – Purchase Credit Token Response Message Model



`BaseVendResp` contains information that must be returned in all response messages that vend a token. `clientStatus` contains the `availCredit` and optional `batchStatus`. `availCredit` contains the available credit balance of the vendor. `batchStatus` indicates the current status of batches (if supported). `utility` contains the details of the utility vending the token. `vendorDetail` is an optional parameter that contains the vendor details that initiated the request. `custVendDetail` is also an optional parameter that contains the details of the customer linked to the meter contained in `meterDetail`.

`CreditVendResp` has a `creditVendReceipt` which is the ultimate outcome of the purchase credit token use case. `creditVendReceipt` is a specialisation of `Receipt`. `Receipt` contains the transactions that were generated when processing the request. This includes `CreditVendTx` and optionally `PayAccTx` transactions. There will always be a `CreditVendTx` containing a `creditTokenIssue` of type `SaleCredTokenIssue` with an associated token. The `Receipt` design also allows for additional `CreditVendTx` transactions that may be automatically generated as part of request, such as a `creditTokenIssue` of type `FBEcredTokenIssue`. `PayAccTx` is included to support payment transactions that may be automatically generated as part of request, such as a debt recovery.

The rest of the use case request / response messages pairs are modelled in a similar manner. Specialisation is used extensively in the message design to provide an extensible object oriented mechanism to choose between parameter alternatives. This approach enables easy extensibility of a type to new types, when the defined types do not meet requirements. Specialisation was also used to embed lookup lists in the protocol message design. Lookup lists specify possible values where only one value must be selected, such as `resource`. XMLVend 1.22 specified these lists in external lookup tables which had to be maintained independently from the message design making maintenance complex and erroneous.

5.3.1.6 Interpreting optional message parameters

The class diagrams specify both mandatory (“[1]”) and optional (“[0]”) attributes. Mandatory request attributes are the contract pre-conditions that must be met for the



service provider to successfully process the request at the interface layer. Mandatory response attributes represent post-conditions of the service provider that must be provided by the service provide at the outcome of a use case. They are also the minimum parameters required by the client to process the response message.

Optional request attributes provide additional data to the server, and the server can choose to process or not to process the optional fields. Optional response fields provide additional data to the client, and the client can choose to process or not to process the optional fields.

However, some specific implementation business logic may require optional request fields to be presented to the server. Such a pre-condition is not handled by the interface but rather as an implementation specific business rule. Utilities would have to review their use case business rules to identify optional fields that may need to be required.

Optional response attributes may also be required by specific implementations that require optional response attributes to be processed by the client. Utilities should review their current prepayment receipts formats to identify optional fields that may b required. The redefinition of optional fields as mandatory would be specified by each utility in their implementation specific use case descriptions document. Such a change would need to be communicated to the utility server supplier and client suppliers in order to be correctly implemented.

5.3.1.7 Message delivery failure scenarios

It is important that request / response messages are reliably delivered. However message delivery failures are inevitable and unpredictable due to possible network, system and software failures.

The “Issue Advice” use case assists with message delivery fault scenarios. For example, the client does not receive a response message and is uncertain if the request was processed by the server. Three advice use case scenarios are provided to assist in resolving this fault scenario (Figure 5.11):

- Advice (Reversal);



- Advice (Confirmation); and
- Advice (Last Response).

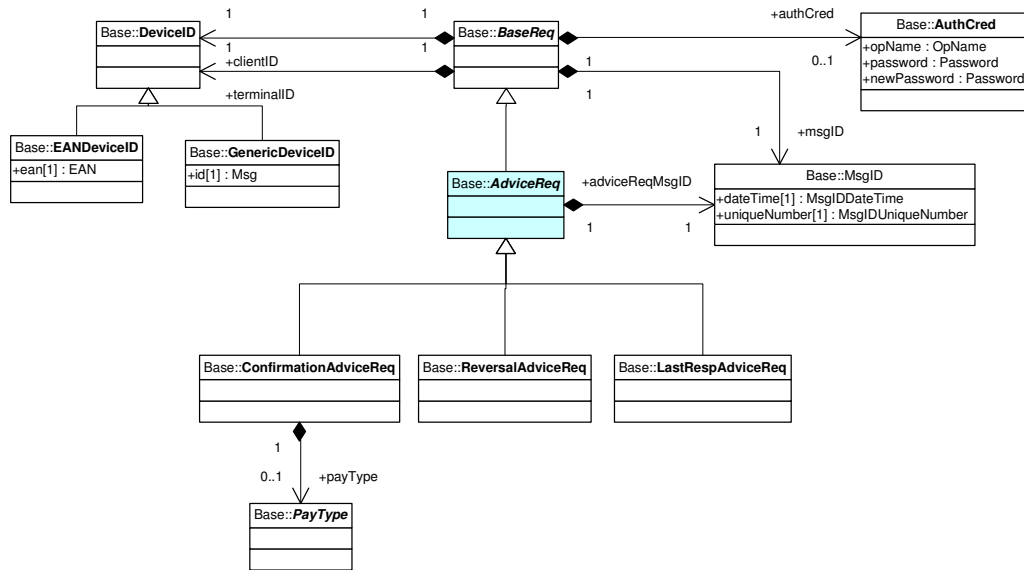


Figure 5.11 – Issue Advice Request Message

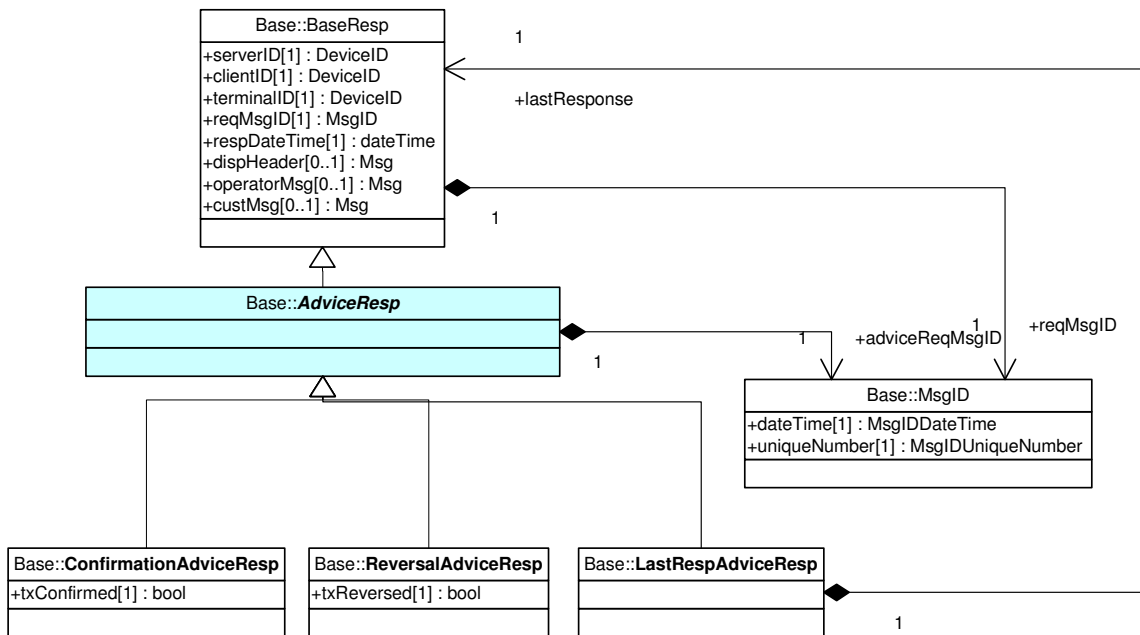


Figure 5.12 – Issue Advice Response Message

The Advice (Reversal) reverses the transaction on the server. This use case scenario however has the following risks:

- The client can issue fraudulent reversal requests; and
- The server depends on decisions made by the client.



This implies that the server would have to “trust” the client to implement the Advice (Reversal) scenario, which may not be acceptable to the utility.

The Advice (Last Response) scenario was added to XMLVend 2 in response to the risks highlighted above. Invoking the Advice (Last Response) use case scenario requests that the server resend a specific response message. The response message to be resent is identified by its message identifier specified in the `adviceReqMsgID` parameter. The resent message can then be printed and handed to the customer. If the request was not processed, an XMLVend fault response message returned of type `LastResponseEx`. The Advice (Last Response) mechanism offers the least risk to the utility since the server does not depend on client decisions.

5.3.1.8 Fault condition support

Exceptions that occur while processing a request need to be communicated to the client. XMLVend 2 implements the same fault condition design as XMLVend 1.22, except the message design updated to the message XMLVend 2 message design principles. The “XMLVend Fault Response Message” (Figure 5.13) is defined to communicate fault descriptions to the client.

The fault response message is a specialisation of `BaseReq` and therefore also enables operator and customer specific fault messages to be returned to the client.

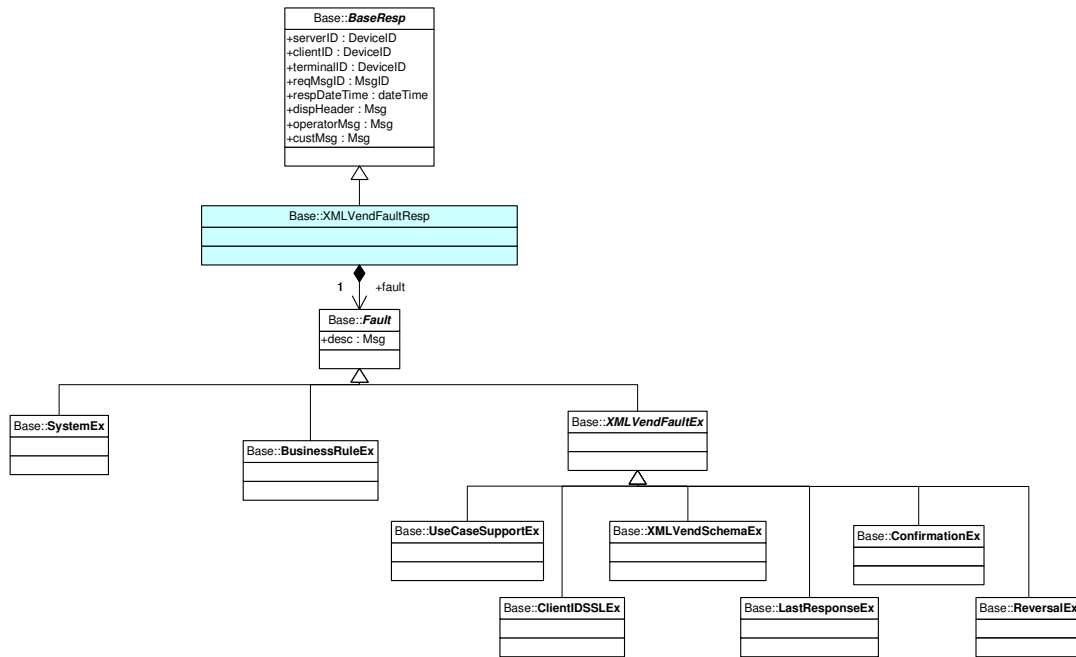


Figure 5.13 – XMLVend Fault Response Message

Figure 5.13 also illustrates the three high-level fault specialisations:

- **XMLVendFaultEx** – These are XMLVend interface related exceptions, such as “XMLVend Schema validation exception”.
- **SystemEx** – These implementation specific system failure exceptions, such as, “Security Module Server not responding”. It is the responsibility of the utility or server supplier to define these, as a specialisation of `SystemEx`.
- **BusinessRuleEx** – These are business rule exceptions, such as, `UnknownMeterEx`. The XMLVend specification document defines several generic business rule exceptions and provides suggested customer and operator messages [52]. Implementation specific business rule exceptions should be defined by the utility or server supplier (or both) as specialisations of `BusinessRuleEx`. The exceptions must be specified as part of a utility’s specific vending use case descriptions and vending processing flow.

5.4 XMLVEND 2 AS A WEB SERVICE

This section discusses the process used to map the XMLVend 2 interface and message pair designs from UML (see section 5.3) to web service implementation files (WSDL and

Schema). This mapping process was manual from the start and not like XMLVend 1.0 where the initial files were auto-generated through a toolset.

The confidence to map the Schema and WSDL files resulted from the experience gained developing the XMLVend 1.22 implementation files, the best practice support provided by the WS-I use scenarios [41]. This resulted in a simple mapping process between UML models and Schema and WSDL that will be presented in this section.

5.4.1 Use case domain mapping to WSDL and Schema

The first task was to decide whether to map the use case domains into respective web service packages or a single package. After significant investigation and analysis by working group members it was decided to create three implementation packages:

- A revenue package that maps to the revenue domain use cases;
- A meter package that maps to the meter domain use cases; and
- A full package that maps both revenue and meter domain use cases.

Figure 5.14 illustrates how the schemas and WSDL's were mapped to each domain. The diagram also shows how the core schemas can be extended to support user specific requirements. The mechanism to extend the core XMLVend schema is described in section 5.4.3.

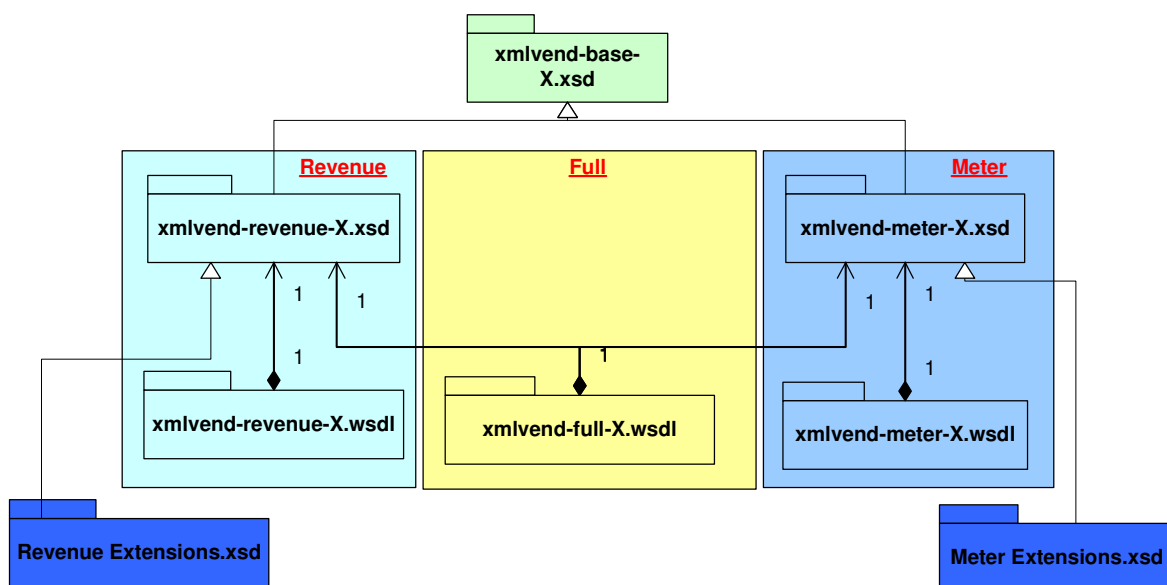


Figure 5.14 – Schema and WSDL for each Domain



Three schemas are defined:

- **Base schema** – The base schema contains common types, constraints and message pairs that are used by both the revenue and meter schemas.
- **Revenue schema** – The revenue schema contains all message pairs and types that are specific to the revenue management domain.
- **Meter schema** – The meter schema contains all message pairs and types that are specific to the meter management domain.

The naming convention used for the schema files is **xmlvend-“domain”-“version”.xsd** and the naming convention for the schema `targetNamespace` is **<http://www.nrs.eskom.co.za/xmlvend/“domain”/“version”/schema>**.

The XML Schema namespaces serve two purposes:

- Ensure uniqueness of XML element and attribute names (eliminate name collisions); and
- Provide a URI to specify the domain associated with a given XML element or attribute.

Three WSDLs are defined:

- **Full** – The full WSDL includes all operations (use cases). This WSDL would usually be used by suppliers that implement use cases from both the revenue and meter management domains.
- **Revenue** – The revenue WSDL contains the revenue management specific operations (use cases). This WSDL would be used by suppliers that implement revenue management operations only.
- **Meter** – The meter WSDL contains the meter management specific operations (use cases). This WSDL would be used by suppliers that implement meter management operations only.

The WSDL naming convention is, **xmlvend-“domain”-“version”.wsdl**. The WSDL `targetNamespace` naming convention is, **<http://www.nrs.eskom.co.za/xmlvend/service/“version”/“domain”>**

The specification version is embedded into the namespaces. Therefore, the specification version is controlled through the schema and WSDL namespaces. If a new XMLVend



version is released then the applicable namespaces will change. The namespaces of the individual schemas can change independent of each other. The “full” WSDL namespace, which includes the meter and revenue domain services, does not specify a “domain”.

5.4.2 Mapping the UML designs to Schema and WSDL

The XMLVend schema and WSDL forms the web service interface contract and therefore the deliverable to ensure interoperability. Therefore, the approach used to realise the schema and WSDL had to support interoperability. In this section the contract-first development approached is reviewed, followed by the message model to schema and interface model to WSDL mapping processes.

5.4.2.1 The contract-first development approach

The “Contract-First Development approach [35] promotes the definition and development of interoperable web services. The process is illustrated in Figure 5.15.

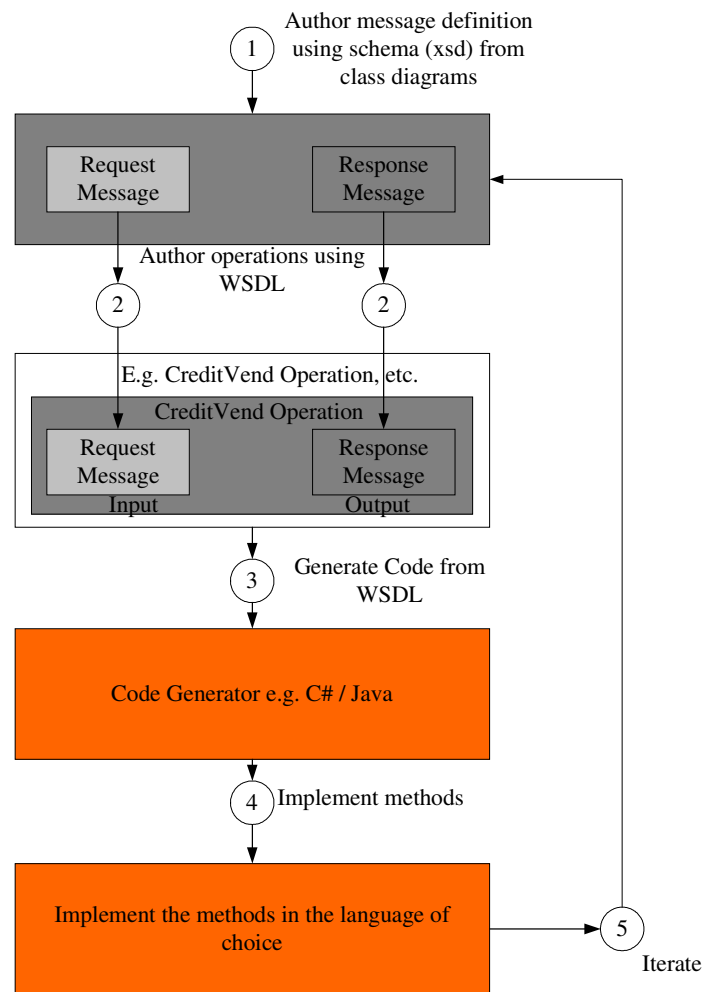


Figure 5.15 – Schema and WSDL Development Approach

Steps 1, 2 and 5 relate to the XMLVend protocol development process. Steps 3 and 4 are used by suppliers to implement an XMLVend compliant server or client.

- **Step 1** is used to author the request / response messages using schema based on the class diagrams described in section 5.3.1.5. Step 1 focuses on naming and structuring the data messages that are exchanged between client and server and will be discussed in detail in section 5.4.2.2.
- **Step 2** specifies the interface specification service operations and transport bindings using WSDL. It is discussed in detail in section 5.4.2.3.
- **Step 5** provides feedback from the implementations to update and enhance the specification.



5.4.2.2 Mapping the Message Design to XMLVend Schema

The schema defines the message contract that governs the XMLVend client and server interactions and therefore was the most critical process in realising XMLVend as a web service. The difficult task of designing the messages was completed in section 5.3.1.5 with the design of the class models for the messages. The aim of this process was to map the message pairs to schema realisations. The process of defining XML vocabularies from UML models is well documented by several industry experts [55][56].

To describe this process, consider the `CredVendReq` request message class diagram (Figure 5.9). A class in UML defines a complex data structure that maps to `complexType` in schema. Therefore, starting at the root class, `CredVendReq`, the schema definition is represented in Listing 4.9:

```
<complexType name="CreditVendReq">
  <complexContent>
    <extension base="AbstractCreditVendReq" />
  </complexContent>
</complexType>
```

Listing 5.1 – CreditVend Req complex type

The above definition illustrates how a key object orientated concept of specialisation is mapped to schema. That is `CredVendReq` is a specialisation of `AbstractCreditVendReq` and this relationship is mapped in schema using the following syntax, `<extension base="AbstractCreditVendReq" />`.

Listing 5.2 shows the instance of `CredVendReq`. The name attribute value starts with a lower case letter (`credVendReq`), which is the camel notation for instance.

```
<element name="credVendReq" type="CredVendReq"/>
```

Listing 5.2 – CreditVendReq instance



`AbstractCreditVendReq` is also a specialisation of `BaseVendReq`, as illustrated in Listing 5.3.

```
<complexType name="AbstractCreditVendReq" abstract="true">
  <complexContent>
    <extension base="BaseVendReq">
      <sequence>
        <element minOccurs="1" maxOccurs="1"
name="purchaseValue" type="PurchaseValue"/>
        <element minOccurs="0" maxOccurs="1"
name="payType" type="PayType"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Listing 5.3 – AbstractCreditVendReq is a specialisation of BaseVendReq

The `AbstractCreditVendReq`, schema definition also illustrates object orientated concept of composition. Composition defines a “has a” relationship. Composition is the relationship is designated by a diamond at root class and a navigation arrow at the target class. The composition has a role name on the relationship line and multiplicity that specifies how the target is related. The fact that the role name starts in small letters, indicates that it is an instance of the target class. Therefore, `AbstractCreditVendReq` has two element fields, `purchaseValue` and `payType`, of type `PurchaseValue` and `PayType` respectively. The multiplicity is represented using `minOccurs` and `maxOccurs` attributes. `purchaseValue` has a multiplicity of [1] (required field), therefore has `minOccurs="1"` and `maxOccurs="1"`. `payType` has a multiplicity of [0..1] (optional field), therefore has `minOccurs="0"` and `maxOccurs="1"`.

XMLVend also adds an additional constraint on the representation of optional fields. That is, XML instance documents that specify `<payType xsi:nil="true"/>` or `<payType/>` for an omitted optional element is not valid. Optional elements must either be present or totally omitted. Optional attributes should also be treated similarly. This was done keep the messages sizes as small as possible.



The `<sequence>` tag indicates that order of the fields between the `<sequence>` tags must be maintained in XML instance documents, otherwise the instance will be invalid. Further, since `BaseVendReq` is the super class of `CredVendReq`, its fields must appear before any other fields.

Listing 5.4 shows the `PurchaseValue` class represented in schema.

```
<complexType name="PurchaseValue" abstract="true"/>
```

Listing 5.4 – PurchaseValue complex type

The `abstract` attribute set to “true” indicates that it is only intended to represent specialised classes of itself. Therefore, an XML instance cannot contain a type, `PurchaseValue`, but only specialisations of `PurchaseValue`.

Listing 5.5 shows the schema representation of `PurchaseValueCurrency`, which is a specialisation of `PurchaseValue`.

```
<complexType name="PurchaseValueCurrency">
  <complexContent>
    <extension base="i0:PurchaseValue">
      <sequence>
        <element minOccurs="1" maxOccurs="1"
name="amt" type="b0:Currency"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Listing 5.5 – PurchaseValueCurrency is a specialisation of PurchaseValue

Listing 5.6 shows `PurchaseValueCurrency`, which has an `amt` field of type `Currency`.



```
<complexType name="Currency">
  <attribute use="required" name="value" type="decimal"/>
  <attribute use="required" name="symbol"
type="i0:CurrencySymbol"/>
</complexType>
```

Listing 5.6 – Currency complex type

The Currency fields have been represented as `<attribute>` tags. Schema attributes are used to define fields whose values are of simple XSD type. Therefore, using the previous example, `value` is of type `decimal`, a simple type. `symbol` is of type `CurrencySymbol`. Listing 5.7 shows the `CurrencySymbol` type definition which is based on simple type, string and restricted with a regular expression.

```
<simpleType name="CurrencySymbol">
  <restriction base="string">
    <pattern value="(\S){1,3}"/>
  </restriction>
</simpleType>
```

Listing 5.7 – CurrencySymbol simple type with restriction

The type `CurrencySymbol` is restricted to only contain values that are characters and digits whose length is limited to between 1 and 3 characters. This mechanism to restrict data values is very powerful and has been used extensively in the XMLVend schema to ensure that data is contained to domain specific constraints. For example, a supply group code can only have 6 decimal numbers.

Using the above UML to schema mapping techniques, all the request and response messages pairs were mapped to their respective schema documents [52].

5.4.2.3 Mapping interface models to WSDLs

Section 5.3.1.2, modelled the three XMLVend interfaces: `XMLVendBaseService`, `XMLVendRevenueService` and `XMLVendMeterService`. This section describes



the process used to map the interface models to XMLVend Web Service interface descriptions using WSDL.

The XMLVendMeterService interface model (Figure 5.7) will be used to illustrate the process. The interface model artefacts map to the abstract elements: `portType`, `operation`, `message`, `part` and `type`. The WSDL `portType` element is similar to the UML interface. `portType` is a collection of operations with inputs, outputs and faults. The `name` attribute of `portType` maps to the interface name (Listing 5.8)

```
<portType name="XMLVendMeterServiceSoap">  
    .....  
</portType>
```

Listing 5.8 – portType Mapping

The `name` attribute of the `operation` element maps to the interface method name (Listing 5.9).

```
<portType name="XMLVendMeterServiceSoap">  
    <operation name="ConfirmMeterRequest">  
        <input message="ConfirmMeterRequestSoapIn" />  
        <output message="ConfirmMeterRequestSoapOut" />  
        <fault name="genFault" message="NRSFaultResponseSoapOut" />  
    </operation>  
</portType>
```

Listing 5.9 – operation name mapping

The `input` and `output` elements map to the interface input and output parameters. The `fault` element maps to the standard XMLVend fault response messages. However, a direct mapping can not be done as WSDL specifies an additional message layer. Therefore, message elements are first created (Listing 5.10) that will contain a `part` element that will map to the input, output and fault parameters as per the interface model.



```

<message name="ConfirmMeterRequestSoapIn">
  <part name="messagePart" element="confirmMeterReq" />
</message>
<message name="ConfirmMeterRequestSoapOut">
  <part name="messagePart" element="confirmMeterResp" />
</message>
<message name="NRSFaultResponseSoapOut">
  <part name="messagePart" element="xmlvendFaultResp" />
</message>

```

Listing 5.10 – message part mapping

The `element` attribute of `part` maps to the request and response message pairs defined in the XMLVend schema. Specifying the `element` attribute instead of the `type` attribute also specifies that document style messaging should be used instead of RPC style messaging. The same process is repeated for each interface operation until the meter WSDL was compiled.

Listing 5.11 shows the `type` element, specifying the location where the XMLVend meter schema that needs to be imported into WSDL.

```

<types>
  <s:schema>
    <s:import schemaLocation="xmlvend-meter-2.1.xsd"
      namespace="http://www.nrs.eskom.co.za/xmlvend/meter/2.1/schem
      a" />
  </s:schema>
</types>

```

Listing 5.11 – The `type` element specifies XSD location

The concrete WSDL elements `service`, `port` and `binding` are specified (Listing 5.12) as per the WS-I Synchronous Request/Response scenario constraints guidelines [41].



```
<binding name="XMLVendMeterServiceSoap"
type="s0:XMLVendMeterServiceSoap">
  <binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
  <operation name="ConfirmMeterRequest">
    <operation soapAction="" style="document" />
    <input>
      <body use="literal" />
    </input>
    <output>
      <body use="literal" />
    </output>
    <fault name="genFault">
      <fault name="genFault" use="literal" />
    </fault>
  </operation>
</binding>
<service name="XMLVendMeterService2.1">
  <port name="XMLVendMeterServiceSoap"
binding="s0:XMLVendMeterServiceSoap">
    <soap:address location="http://manufacturer-webservice-
address" />
  </port>
</service>
```

Listing 5.12 – WSDL concrete elements

The `soapAction` attribute was also specified as an empty string. This removes the close coupling between the protocol and HTTP, allowing other transports to be utilised in the future [53]. It is also significantly reduces the HTTP header sizes.

5.4.3 Utility specific extensions

The message design provides support for utility extensions. Eskom has used this mechanism to support additional functionality not found in the core XMLVend message design. For example, Eskom has specialised the generic `BusinessRuleEx` class to create several Eskom specific business rule exceptions (Figure 5.16).

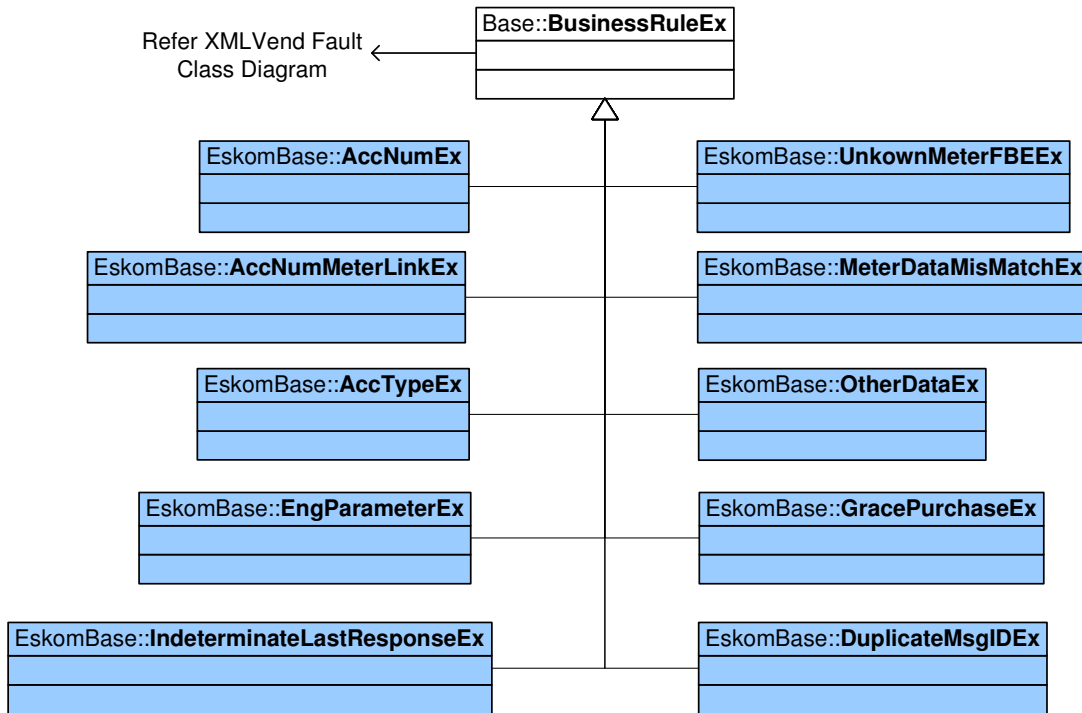


Figure 5.16 – Eskom specialisation of BusinessRuleEx

The new exceptions are then mapped to an Eskom schema (Listing 5.13). The text in **bold** font highlights how XMLVend base schema is imported and referenced by the Eskom schema. The `BusinessRuleEx` is defined in the XMLVend base schema.

```

<schema
xmlns:b0="http://www.nrs.eskom.co.za/xmlvend/base/2.1/schema"
<import schemaLocation="xmlvend-base-2.1.xsd"
namespace="http://www.nrs.eskom.co.za/xmlvend/base/2.1/schema"
  />
  <complexType name="AccNumEx">
    <complexContent>
      <extension base="b0:BusinessRuleEx" />
    </complexContent>
  </complexType>
  .....
</schema>
  
```

Listing 5.13 – Utility specific exceptions

Utility specifications extensions should be compiled into a separate schema file. The recommended naming convention for this file is “utility”-xmlvend-“domain”-



“**version**”.xsd. In this example the schema file will be named, eskom-xmlvend-base-2.1.xsd.

5.4.4 XMLVend 2 Stabilisation

Shortly after the release of XMLVend 2.0, several issues were discovered with mapping of the message designs from UML to schema. This necessitated the release of XMLVend 2.1, which corrected the mapping errors and added support for the reprint of payment transactions. XMLVend 2.1 has been extensively tested as part of the Eskom online vending pilot project. It is now being used by Eskom in its national rollout of online vending.

5.5 SUMMARY

This chapter discussed the development of XMLVend 2. XMLVend updated and added several use cases to the protocol. The author focused on redesigning all the use case messages using an object orientated methodology that promoted “good” design.

The web service implementation was divided into three implementation packages:

- A revenue package that maps to the revenue domain use cases;
- A meter package that maps to the meter domain use cases; and
- A full package that maps both revenue and meter domain use cases.

The schema and WSDL was mapped using a simple, easy to follow process to map the interface and message models from the UML message design models to create the three web service packages. The process was based on the industry best practice approach of Contract First Web Service development.

The process provides a generic approach to define service specifications with the following benefits:



- Simple to define and maintain;
- Removes inconsistencies and interpretation;
- Implements open, proven and widely supported technologies; and
- Maximises interoperability.

The following lessons were learnt with respect to web service realisations:

- The manual definition of schema and WSDL is very powerful as total control of the exchanged message formats is maintained. However it does have the following disadvantages:
 - Skilled resources are required to undertake such a task with significant skill and expertise in XML and schema; and
 - All updates need to be checked and verified by another resource before finalising the implementation files WSDL and Schema as typographical errors will creep in.
- All implementations files must be tested and validated for WS-I compliance. It should also be tested for compatibility with at least two widely used web services frameworks such as Microsoft .Net and Axis.
- The test suite reference implementations must be developed in conjunction with WSDL and schema development. This will ensure that message pair instances are realised as expected and any implementation issues are identified early in the process.

CHAPTER 6 : XMLVEND COMPLIANCE TESTING

6.1 INTRODUCTION

XMLVend is a client / server interface specification that is platform, operating system and programming language independent. Therefore XMLVend client and server suppliers are able to develop using platforms and technologies of their choice while ensuring that implementations easily interoperate. A simple testing mechanism was therefore essential for clients and servers to determine if their implementations are XMLVend compliant. That is, the client and server must meet the pre-conditions and post-conditions of the interface as specified in the interface design and XMLVend schema.

This section provides an overview of the XMLVend test suite and how it should be used to check XMLVend compliance.

6.2 THE XMLVEND TEST SUITE

The benefits of the XMLVend test suite are:

- Accelerates development of XMLVend compliant systems;
- Provides a common base for testing and compliance validation; and
- Ensures that all implementations meet a basic set of functional requirements.

The author specified the following design requirements of XMLVend test suite. The XMLVend test suite should consist of the following components (Figure 6.1):

- **Reference Server.** A reference server implementation of an XMLVend interface specification. The reference server simulates the actual token vending functionality but adheres strictly to the XMLVend protocol. The server also implements supports the following functional requirements:
 - Supports both HTTP and HTTPS connections with support for mutual authentication.
 - Supports GZIP.

- Implements functionality to support Issue Advice (reversal) and Issue Advice (last Response) use cases.
- Returns XMLVend compliant messages.
- **Reference Client:** A web browser based reference client implementation of the XMLVend protocol.
- **Message Interceptor:** A message interceptor to intercept and capture client and server messages.
- **Message Validator:** The message validator analyses the captured messages and validates them against the XMLVend schemas. The Message Validator is also able to validate messages against utility specific extensions.

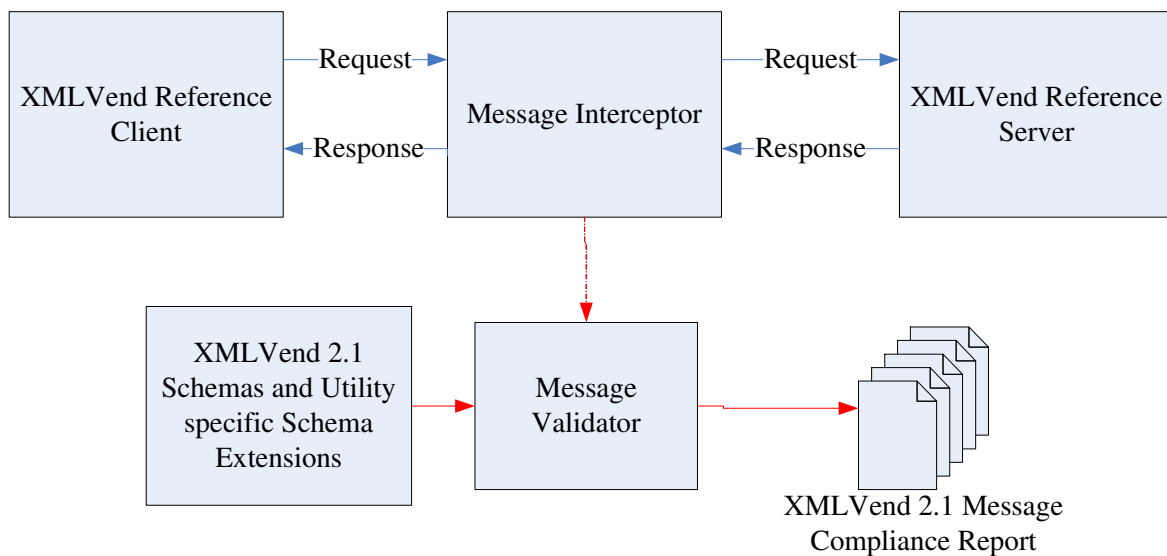


Figure 6.1 – XMLVend Test Suite Components

To test a client implementation, the reference client is replaced with the specific client implementation in Figure 6.1. Similarly, to test a server implementation, the reference server is replaced by the specific server implementation.

STC was contracted to develop the initial XMLVend test suite framework. The author was responsible for expanding the test suite framework to support all the XMLVend use cases.

6.3 XMLVEND TEST SUITE – WALK THROUGH

The reference client is a Java web application that can be accessed using any XForms, compliant Web browser. Figure 6.2, is screen shot of the default user interface (UI) of the XMLVend Client.

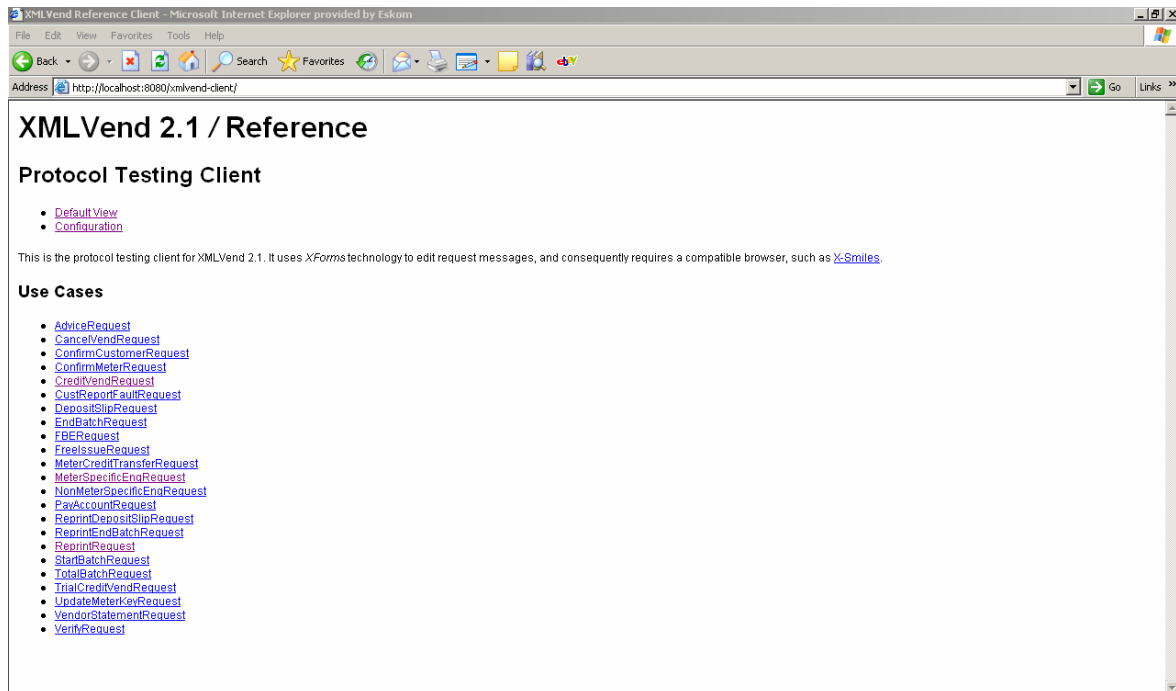


Figure 6.2 – XMLVend reference Client - Default User Interface(UI)

The Purchase Credit Token use case client user interface was developed using XForms (Figure 6.3). The user completes the form and submits the form to the client web application. The data submitted is formatted as a valid `creditVendReq` XML request message. The client web application then wraps the `creditVendReq` in SOAP tags and makes a web service request to the configured XMLVend server.



Purchase Credit Token

- [Default View](#)
- [Configuration](#)

Resource	Electricity
Meter Identification	By Meter Number
Serial Number	06686069342
Purchase Value	Currency
Currency Value	0.00
Currency Symbol	ZAR
Payment Type	Not applicable
^ - Optional	
Message ID	000001 / 20061107180043
Submit	

Figure 6.3 – Purchase Credit Token Use Case Client UI

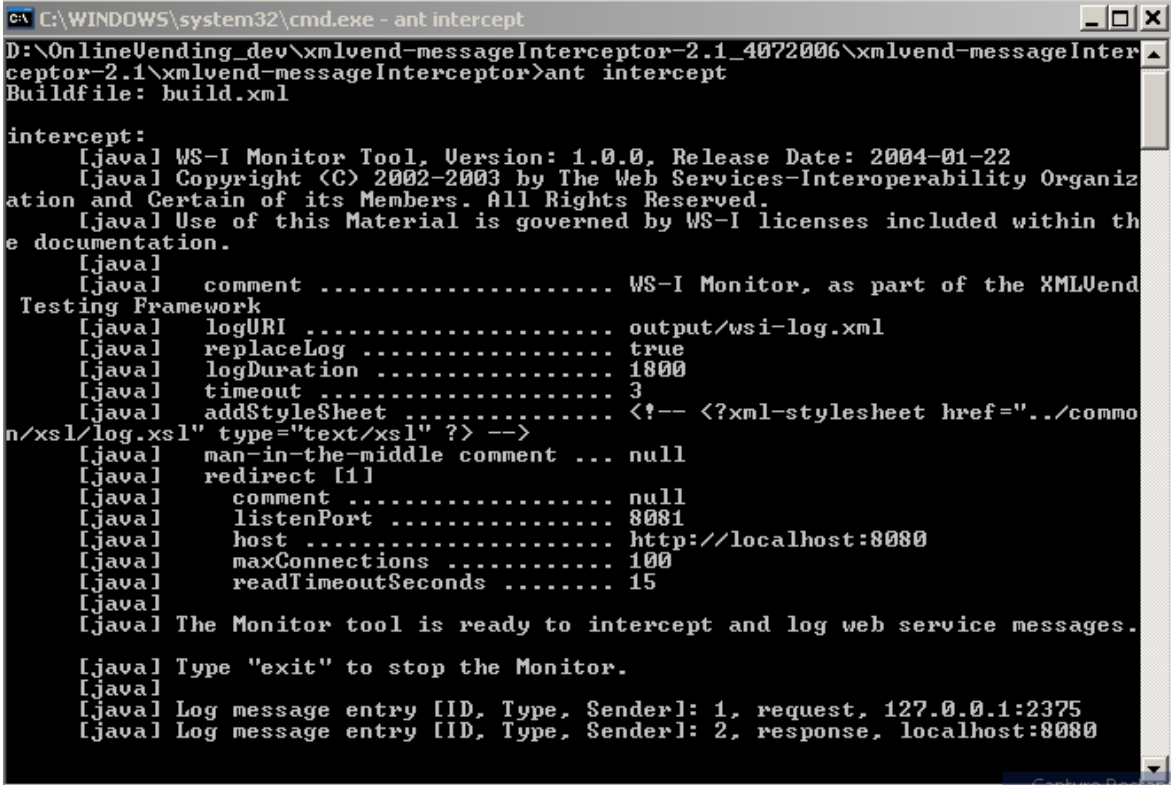
The server receives the SOAP `creditVendReq` request and using an appropriate XML style sheet (XSL) transforms the request message into a `creditVendResp` response message. The `creditVendResp` response message is then wrapped in SOAP tags and returned to the client web application. The client web application then transforms the `creditVendResp` response message to HTML and returns it to the client user interface (Figure 6.4).

The screenshot shows an XML response in a browser window. The response is structured as follows:

creditVendResp	
clientID	-
ean	0004708001622
type	b0:EANDeviceID
serverID	-
ean	0004708001998
type	b0:EANDeviceID
terminalID	-
ean	0000000000001
type	b0:EANDeviceID
reqMsgID	-
dateTime	20090611203448
uniqueNumber	000001
respDateTime	2009-06-11T08:34:54
dispHeader	CREDIT VEND - TAX INVOICE
custMsg	This is a free text message field - contains a server populated, customer directed message.
ClientStatus	
availCredit	-
symbol	ZAR
value	2000
utility	
address	Meganatt Park, Contact Centre tel 086-003-7598
name	Eskom Online
taxRef	4740101508
creditVendReceipt	
receiptNo	1234567890

Figure 6.4 – Reference Client Response UI

Figure 6.5 illustrates the Message Interceptor application capturing messages exchanged between client and server. The WS-I message interceptor is used to provide this functionality, which basically acts like a proxy between the client and server.



```

C:\WINDOWS\system32\cmd.exe - ant intercept
D:\OnlineVending_dev\xmlvend-messageInterceptor-2.1_4072006\xmlvend-messageInter
ceptor-2.1\xmlvend-messageInterceptor>ant intercept
Buildfile: build.xml

intercept:
[java] WS-I Monitor Tool, Version: 1.0.0. Release Date: 2004-01-22
[java] Copyright (C) 2002-2003 by The Web Services-Interoperability Organiz
ation and Certain of its Members. All Rights Reserved.
[java] Use of this Material is governed by WS-I licenses included within th
e documentation.
[java]
[java] comment ..... WS-I Monitor, as part of the XMLVend
Testing Framework
[java] logURI ..... output/wsi-log.xml
[java] replaceLog ..... true
[java] logDuration ..... 1800
[java] timeout ..... 3
[java] addStyleSheet ..... <!-- <?xml-stylesheet href="..//commo
n/xsl/log.xsl" type="text/xsl" ?> -->
[java] man-in-the-middle comment ... null
[java] redirect [1]
[java] comment ..... null
[java] listenPort ..... 8081
[java] host ..... http://localhost:8080
[java] maxConnections ..... 100
[java] readTimeoutSeconds ..... 15
[java]
[java] The Monitor tool is ready to intercept and log web service messages.

[java] Type "exit" to stop the Monitor.
[java]
[java] Log message entry [ID, Type, Sender]: 1, request, 127.0.0.1:2375
[java] Log message entry [ID, Type, Sender]: 2, response, localhost:8080

```

Figure 6.5 – Message Interceptor

The Message Interceptor logs the request and response messages to a log file. The log file is analysed by the Message Validator application, which produces an XMLVend compliance report (Figure 6.6).



XMLVend Test Report - Microsoft Internet Explorer provided by Eskom

Network Address: localhost8080
Client ID: ean.6004708001981

Server (Passed test)

Network Address: localhost8080

Test Session

Start: 2008-07-07T17:39:58.296
End: 2008-07-07T17:39:58.296
Total Requests: 14

r0:creditVendReq (Valid)

2008-07-07T17:39:58.296

```
<?xml version="1.0"?> <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"><Body><r0:creditVendReq xmlns:b0="http://www.nrs.eskom.co.za/xmlvend/base/2.1/schema" xmlns:r0="http://www.nrs.eskom.co.za/xmlvend/revenue/2.1/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.nrs.eskom.co.za/xmlvend/revenue/2.1/schema http://localhost:8080/eskom-xmlvend-client/contract/xmlvend-revenue-2.1.xsd"><b0:clientID ean="6004708001981" xsi:type="b0:EANDeviceID" /><b0:terminalID ean="0000000000001" xsi:type="b0:EANDeviceID" /><b0:msgID uniqueNumber="000001" dateTime="20080707173953" /><b0:resource xsi:type="b0:Electricity" /><b0:idMethod<b0:meterIdentifier msno="06686069342" xsi:type="b0:MeterNumber" /></b0:idMethod><r0:purchaseValue xsi:type="r0:PurchaseValueCurrency"><r0:amt symbol="ZAR" value="10.00" /></r0:purchaseValue></r0:creditVendReq</Body></Envelope>
```

Server Response (Valid)

Figure 6.6 – XMLVend Compliance Report

Figure 6.7 illustrates an XMLVend compliance report that has server response message validation failures.

XMLVend Test Report - Microsoft Internet Explorer provided by Eskom

Test Session

Start: 2008-07-07T18:14:16.703
End: 2008-07-07T18:36:38.687
Total Requests: 40

r0:confirmCustomerReq (Valid)

2008-07-07T18:14:16.703

```
<?xml version="1.0"?> <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"><Body><r0:confirmCustomerReq xmlns:b0="http://www.nrs.eskom.co.za/xmlvend/base/2.1/schema" xmlns:r0="http://www.nrs.eskom.co.za/xmlvend/revenue/2.1/schema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.nrs.eskom.co.za/xmlvend/revenue/2.1/schema http://localhost:8080/eskom-xmlvend-client/contract/xmlvend-revenue-2.1.xsd"><b0:clientID ean="6004708001981" xsi:type="b0:EANDeviceID" /><b0:terminalID ean="0000000000001" xsi:type="b0:EANDeviceID" /><b0:msgID uniqueNumber="000001" dateTime="20080707181412" /><r0:idMethod xsi:type="b0:VendIDMethod"><b0:meterIdentifier msno="06686069342" xsi:type="b0:MeterNumber" /></r0:idMethod></r0:confirmCustomerReq</Body></Envelope>
```

Server Response (Invalid)

2008-07-07T18:14:16.828

```
<?xml version="1.0"?> <Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"><Body><Fault><faultcode xmlns="http://www.nrs.eskom.co.za/xmlvend/base/2.1/schema" xsi:schemaLocation="http://www.nrs.eskom.co.za/xmlvend/revenue/2.1/schema http://localhost:8080/eskom-xmlvend-client/contract/xmlvend-revenue-2.1.xsd">javax.xml.transform.TransformerConfigurationException: javax.xml.transform.TransformerConfigurationException: Expected ,, but found: caused by: javax.xml.transform.TransformerConfigurationException: javax.xml.transform.TransformerConfigurationException: Expected ,, but found: </faultstring></Fault></Body></Envelope>
```

Error

XML Schema validation: UndeclaredPrefix: Cannot resolve 'soap:Server' as a QName: the prefix 'soap' is not declared.

Error

XML Schema validation: cvc-type.3.1.3: The value 'soap:Server' of element 'faultcode' is not valid.

Figure 6.7 – XMLVend Compliance Report with failures

An XMLVend compliance test template and procedure has also been developed to assist with the testing and validating of XMLVend implementations by the author.

6.4 CUSTOMISING AND ENHANCING THE TEST SUITE

The test suite is completely open source, which enables customisation of the reference client and server responses to support utility specific extensions and constraints. The author has customised the test suite to support Eskom's custom vending processes (Figure 6.8).

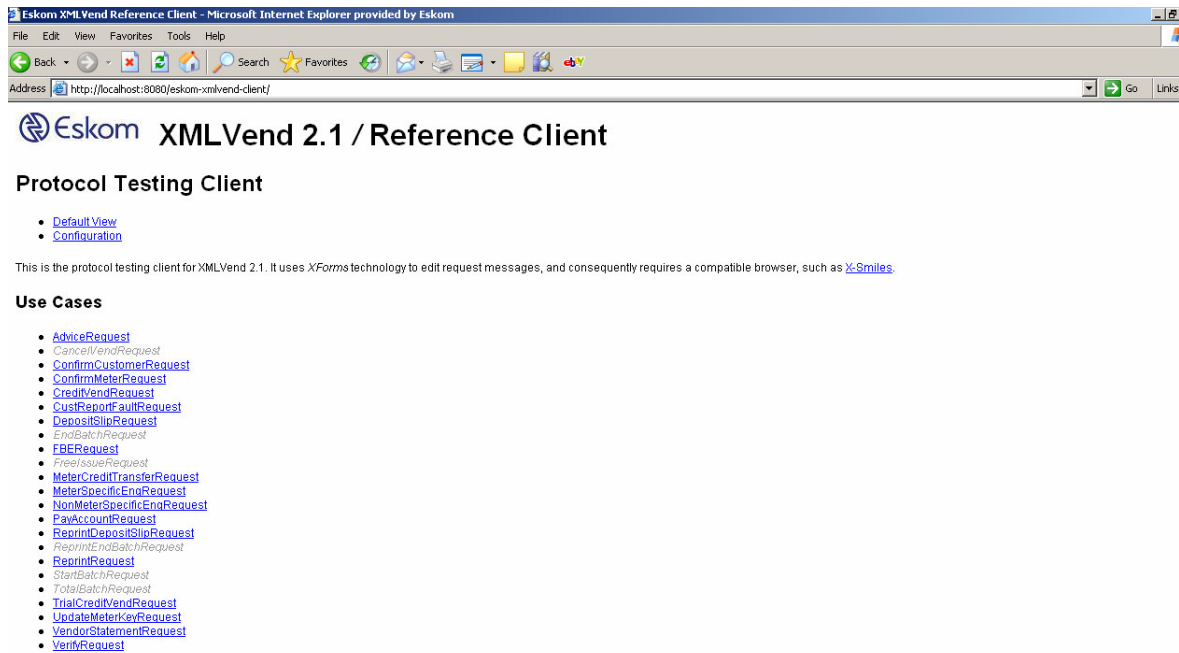


Figure 6.8 – Eskom Customised XMLVend Client

The test suite has also been enhanced by the author to support:

- Client certificate common name validation on the test server; and
- Simulate server timeout scenarios.

6.5 SUMMARY

The XMLVend test suite has proven to be a simple yet powerful tool that supports development and testing of XMLVend implementations. The test suite and reference implementations have provided invaluable support for suppliers to rapidly comply with the XMLVend protocol.

CHAPTER 7 : CONCLUSIONS AND RECOMMENDATIONS

7.1 CONCLUSIONS

XMLVend 2.1 has matured into a robust and stable open industry specification through an iterative development process (Figure 7.1).

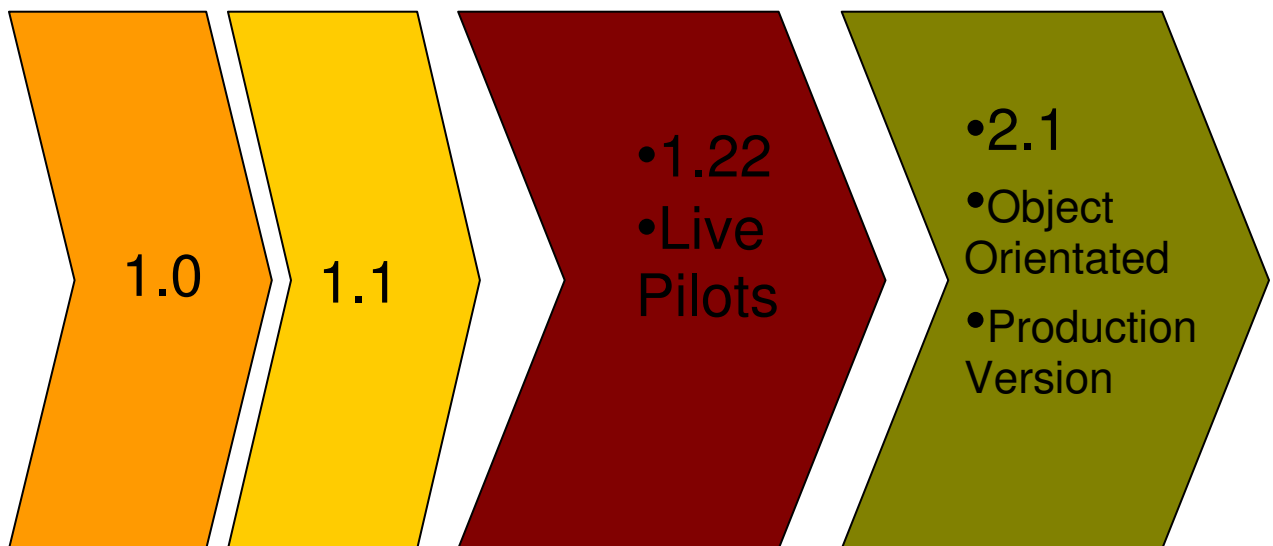


Figure 7.1 – Specification roadmap

An important task of the specification development was build awareness of the specification, to increase its adoption and implementation. XMLVend 2.1 has therefore been widely published and presented at several local and international electricity industry conferences.

A detailed paper entitled, *Development of a prepaid Online Vending Interface Specification (Another world first for the South African Prepayment Industry)* was published in the “Energise” journal in March 2004 [6]. It was very well received with positive comments from several working group members and the magazine editors. Papers were also presented at the 2004 and 2005 Prepayment Week Conference in Johannesburg and Cape Town respectively, where it was also very well received [48][49].

XMLVend posters were exhibited at the Eskom conference stand at 2005 Prepayment Week Conference, which received significant interest and questions from conference delegates.

At the 2006 African Utility Week, XMLVend 2.1 was one of the key exhibitions at the Eskom stand. The following XMLVend exhibits were developed for the stand:

- An automated electronic wall presentation.
- An XMLVend client and server test suite demonstration and
- XMLVend posters (Figure 7.2).



Figure 7.2 – 2006 African Utility Week – Eskom XMLVend stand⁸

A paper entitled, *A Secure Web Service for Prepayment Electricity Vending in South Africa (A case study and industry specification)*, was presented at Second International Conference on Internet and Web Applications and Services, in Mauritius [59]. This paper was published in an effort to get international review and recognition of the XMLVend development. The paper was very well received and several delegates commented that they were impressed to see that web services were also being used in applications that improve the quality of people's lives.

XMLVend has become a key enabler for utilities to implement standardised, secure, interoperable and flexible online vending systems. XMLVend has also succeeded in addressing the online vending risks:

- Being locked into a single vending system supplier for both the vending server and the POS devices;
- Being locked into supplier controlled proprietary technology;

⁸ Kennedy P Subramoney (author) at the Eskom XMLVend stand.



- Vending system suppliers controlling the utility's vending POS channels and more concerning the expansion of these channels;
- Suppliers controlling the cryptographic security of the system, which could result in inconsistent and possibly insecure security across implementations;
- Integration with utility backend systems could be restricted and costly; and
- Finally, the proliferation of proprietary online vending systems could have a detrimental impact on the already standardised prepayment industry.

The XMLVend specification has been an innovative development with buy-in from utilities, suppliers, industry experts and other interested parties working together for the common interest of the industry. It also confirms South Africa as prepayment industry leader in the development and implementation of innovative and sustainable prepayment solutions.

7.2 RECOMMENDATIONS

The following recommendations are made:

- XMLVend 2.1 has been stable for more than two years. However to ensure continued interest, support and implementation, it is important to continue promoting and supporting the specification.
- Studies evaluating XMLVend's performance, security and scalability on a variety of platforms and architectures will provide immense value to the protocol and its implementation.
- Studies on online vending system implementation best practices using XMLVend will also provide immense value to utilities wanting to implement an online vending system.

REFERENCES

- [1] K. van den Berg, "Development of on-line vending standards," *South African Prepayment Week, Johannesburg, South Africa*, 2003.
- [2] "Prepayment Frequently Asked Questions," Eskom, http://www.eskom.co.za/live/content.php?Item_ID=591. Last accessed on 21 September 2006.
- [3] P. Johnson, "The Standard Transfer Specification – Past, Present and Future," Eskom, Internal Document, 2002.
- [4] "MSF Process Model v.3.1," Microsoft, <http://www.microsoft.com/msf>. Last accessed on 20 September 2006.
- [5] K.P. Subramoney, Online Vending - Industry trends towards an industry specification, Eskom Resources and Strategy, Research Report, RES/RR/01/15519, 2002.
- [6] K.P. Subramoney, Development of a prepaid Online Vending Interface Specification (Another world first for the South African Prepayment Industry), *Energise (Power Journal of the South African Institute of Electrical Engineers)*, March 2004.
- [7] B. Meyer, S. Moodley, Prepayment Evolution – Key Features of a Proven Revenue Management System, *ESI Africa (The Power Journal of Africa)*, Issue 1, 2005, pp. 12-13.
- [8] Discussions with Jan Westenraad and team, at Tswane Metropolitan Municipality.
- [9] Discussions with Dave Michie (Nelson Mandela Metropolitan Municipality) and Dale Liebenberg (Ballenden & Robb Consulting Engineers).
- [10] K. van den Berg, Can current vending systems be used to implement EBSST, *Vending Options, Johannesburg, South Africa*, 2001.
- [11] M.E. Makwarela, Analysis of Prepayment System within Eskom, Management Report, University of Pretoria, 2003.
- [12] J.E. Cohen, Plant Data Systems: Facilitating integration through open interfaces, Eskom Resources and Strategy, Research Report, Eskom, RES/RR/01/15519, 2001.
- [13] A. Phillips, How standards happen, *Conference on Language Standards for Global Business, Barcelona*, May 2006.
- [14] "Cutting out the middle-tier costs", *eSecure*, August 2003, pp 14-16.
- [15] R. Kaplan, Standard Transfer Specification (STS) Guide, Eskom (Transmission Group), 1995.
- [16] D. Taylor, Standard Transfer Specification – Synopsis, STS Association, August 2002.
- [17] D. Taylor, STS, Key Management and Revenue Protection, *African Utility Week, Cape Town, South Africa*, May 2005.
- [18] F. Mabuza and M. van Rensburg, Rationalisation of Maintenance Standards, *ESI Africa (The Power Journal of Africa)*, Issue 2, 2006, pp. 66-67.

- [19] National Rationalised Specification, Interface Standards: Standard Transfer Specification /Credit dispensing unit–Electricity dispenser – Categories of tokens and transaction data fields, NRS 009-6-6.
- [20] National Rationalised Specification, Interface Standards: Standard Transfer Specification /Credit dispensing unit–Electricity dispenser – Token encoding and data encryption and encryption, NRS 009-6-7.
- [21] National Rationalised Specification, Interface Standards: Standard Transfer Specification /Disposable magnetic token technology-Token encoding format and physical token definition, NRS 009-6-8.
- [22] National Rationalised Specification, Interface Standards. Standard Transfer Specification /Numeric token technology–Token encoding format and physical token definition, NRS 009-6-9.
- [23] National Rationalised Specification, Key management: Standard transfer specification/The management of cryptographic keys, NRS 009-7.
- [24] National Rationalised Specification, Functional and performance requirements: System master stations, NRS 009-2-1.
- [25] National Rationalised Specification, Functional and performance requirements: Credit dispensing units, NRS 009-2-2.
- [26] National Rationalised Specification, Functional and performance requirements: Security modules, NRS 009-2-3.
- [27] National Rationalised Specification, Functional and performance requirements: Standard token translators, NRS 009-2-4.
- [28] National Rationalised Specification, Database Format, NRS 009-3.
- [29] National Rationalised Specification, Electricity meter cards and associated numbering standards: National electricity meter cards, NRS 009-4-1.
- [30] National Rationalised Specification, Electricity meter cards and associated numbering standards: National electricity meter numbers, NRS 009-4-2.
- [31] National Rationalised Specification, Interface Standards: Credit dispensing unit – Standard token translator interface, NRS 009-6-1.
- [32] National Rationalised Specification, Interface Standards: System Master Station - Credit dispensing unit, NRS 009-6-3.
- [33] National Rationalised Specification, Interface Standards: Data Transfer by physical media - System Master Station - Credit dispensing unit, NRS 009-6-4.
- [34] F. Solms, “Use-Case Responsibility Driven Analysis and Design (URDAD) for System Design,” Solms Training, Consulting and Development, <http://ww.solms.co.za/>, Last accessed on 24 June 2005.
- [35] A. Skonnard, “Techniques for Contract-First Development,” Microsoft, MSDN Magazine, <http://msdn.microsoft.com/msdnmag/issues/05/06/ServiceStation>, Last accessed on 20 June 2005.
- [36] U. Gopalakrishnan, K.R. Rajesh, “Securing Web Services”, IBM, <http://www-106.ibm.com>, Last accessed on 20 April 2004.
- [37] K.P. Subramoney, Online Vending Specification – Security Requirements, Eskom Resources and Strategy, Online Vending Project Discussion Document, June 2003.

- [38] Web Services Architecture Requirements, D. Austin, W. W. Grainger, A. Barbir, C. Ferris, S. Garg, W3C Working Group Note 11 February 2004, <http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211>, Last accessed on 19 October 2006.
- [39] Web Services Architecture, D. Austin, W. W. Grainger, A. Barbir, C. Ferris, S. Garg, W3C Working Group Note 11 February 2004, <http://www.w3.org/TR/ws-arch/>, Last accessed on 19 October 2006.
- [40] WS-I Basic Profile, Version 1.0a, <http://www.ws-i.org/>
- [41] WS-I Use Scenario, Version 1.01, <http://www.ws-i.org/>
- [42] W. Penzhorn, G.B. Hearn, "Investigation of security protocols for Online vending systems for STS," Eskom Resources and Strategy Online Vending Project Document, 8 October 2002.
- [43] W. Penzhorn, "Evaluation of the PEARL Protocol," Eskom Resources and Strategy, Online Vending Project Document, 16 September 2003.
- [44] F. Solms, "Verification of XML Vend 1.0 WSDL against Protocol Requirements," Eskom Resources and Strategy, Online Vending Project Document, 17 October 2003.
- [45] C. Smith, P. Garvey, R. Glushko, "Developing XML Vocabularies for Web Services," University of California (Berkeley), Center for Document Engineering, Technical Report 2003-1, 4 January 2003.
- [46] XMLPay Specification, Verisign, <http://www.verisign.com/developer/xml/xmlpay.html>.
- [47] K. Venketiah, B. Mokgele, "Eskom Online Vending Pilot – Lessons Learnt," *South African Prepayment Week, Cape Town, South Africa*, 2005.
- [48] K. P. Subramoney, "South Africa's new Online Vending Interface Specification (NRS 009-6-10 / XMLVend 1.0)," *South African Prepayment Week, Johannesburg, South Africa*, 2004.
- [49] K. P. Subramoney, "XMLVend – Towards and Industry Specification," *South African Prepayment Week, Cape Town, South Africa*, 2005.
- [50] K.P. Subramoney, Online Vending Prepayment Interface Specification (XMLVend – NRS009-6-10) - Update, *Energise (Power Journal of the South African Institute of Electrical Engineers)*, April 2006.
- [51] National Rationalised Specification, Interface Standards: On-line Vending Server – Vending Clients, NRS 009-6-10, Trial-Use release, Version 1.22, December 2004.
- [52] National Rationalised Specification, Interface Standards: On-line Vending Server – Vending Clients, NRS 009-6-10, Interim release, Version 2.1, February 2006.
- [53] Y. Shohoud, Real World XML Web Services – For VB and VB.NET Developers, 1st ed. Boston, USA: Pearson Education, 2002.
- [54] A. Skonnard, "Improving Web Service Interoperability," Microsoft, MSDN Magazine, <http://msdn.microsoft.com/webservices/webservices/building/interop/default.aspx?pull=/msdnmag/issues/04/11/servicestation/default.aspx>, Last accessed on 20 June 2006.

- [55] F. Solms, “Object Oriented Analysis and Design (OOAD),” Solms Training, Consulting and Development, Course Notes, <http://ww.solms.co.za/>, Last accessed on 24 June 2005.
- [56] D. Carlson, “Modelling XML Vocabularies with UML: Part 1-3,” <http://www.xml.com>, Last accessed on 24 August 2006.
- [57] W. Stallings, *Cryptography and Network Security: Principles and Practice* (3rd Edition), Prentice Hall 2002.
- [58] “South Africa Yearbook 2007 / 08”, Government Communication and Information System, <http://www.gcis.gov.za/docs/publications/yearbook/index.html>, Last accessed 25 June 2008.
- [59] K.P. Subramoney, G.P. Hancke, “A Secure Web Service for Electricity Prepayment Vending in South Africa: A Case Study and Industry Specification,” *iciv*, pp.66, Second International Conference on Internet and Web Applications and Services (ICIW'07), 2007.
- [60] R Housley, W. Ford, W Polk, D Solo, “Internet X.509 Public Key Infrastructure Certificate and CRL Profile,” RFC 2459, January 1999.
- [61] C.M. Chernick, C. Edington III, M.J. Fanto, R. Rosenthal, “Guidelines for Selection and Use of Transport Layer Security (TLS) Implementations,” National Institute of Standards and Technology, U.S. Department of Commerce, NIST Special Publication 800-52, June 2005.
- [62] S.J Vaughan-Nichols, “XML Raises Concerns as It Gains Prominence,” *Computer*, May 2003, pp 14 - 16.
- [63] M Milenkovic, et al., “Toward Internet Distributed Computing,” *Computer*, May 2003, pp 38 – 25.
- [64] C Shirky, “Web Services and Context Horizons,” *Computer*, September 2002, pp 98 – 99.
- [65] C.J Augeri, D.A Bulutoglu, B.E Mullins, R.O Baldwin, L.C Baird, “An Analysis of XML Compression Efficiency,” *Workshop On Experimental Computer Science*, San Diego, California, 2007.
- [66] B. Goodman, “Squeezing SOAP, GZIP enabling Apache Axis,” <http://www.ibm.com/developerworks/webservices/library/ws-sqzsoap.html>, Last accessed on 28 January 2010.
- [67] W. Ng, W. Lam, and J. Cheng, “Comparative analysis of XML compression technologies,” *World Wide Web*, 9(1):5–33, Kluwer Academic Publishers, March 2006.

ADDENDUM A. : THE ONLINE VENDING SPECIFICATION (NRS009-6-10)

The XMLVend 2.1 specification document compiled by the author has not been attached due to its large size but is accessible on the specification website: <http://www.nrs.eskom.co.za/xmlvend>. The accompanying XMLVend 2.1 schemas and WSDL documents and test suite are also downloadable from the same website address.



ADDENDUM B. : WORKING GROUP PARTICIPANT LIST

No	Name	Company
1	Sue Temple	ABSA
2	Aldo Kriel	Actaris
3	Don Taylor	Actaris
4	Johann Groenewald	Actaris
5	Hendrik vd Bijl	ADO
6	Jarred Bonert	Altivex
7	Brent Jacobs	Application Frameworks
8	Richard Stone	Application Frameworks
9	Bruce Barker	Arivia
10	Terence Venadu	Autopage
11	Gustav Vermass	BlueLable
12	Orry Melissakis	BlueLable
13	Neil Ballantyne	Cape Town Electricity
14	Roland Hill	CBI
15	Teddy Naidoo	City Power
16	Franco Pucci	Conlog
17	Lance Hawkins-Dady	Conlog
18	Poobalan Naidoo	Conlog
19	Mark Simon	Contour
20	Steve Leigh	Contour
21	Dave Semmelink	Contour Technology (Pty) Ltd
22	Christo van der Merve	Eason Electronic
23	Hassen Sheik	eCentric Switch
24	Rob de Vries	eCentric Switch
25	Robin Golden	eCentric Switch
26	Warren Mathers	Ekurleni Municipality
27	Fred Classens	Ennertec
28	Robert Wilson	Ennertec
29	Cleo Zulu	ESEC Energy
30	Deon Botes	ESEC Energy
31	Ron Pienaar	ESEC Energy
32	Sham Dhrampal	Eskom
33	Vusi Moyeni	Eskom
34	William Mype	Eskom
35	Deon Van Rooi	Eskom (Distribution)
36	Jimmy O'Kennedy	Eskom (Distribution)
37	Ken Myburgh	Eskom (Distribution)
38	Kevin Ventekiah	Eskom (Distribution)
39	Kennedy P Subramoney	Eskom (Eskom Enterprises - TSI)
40	Paul Johnson	Eskom (Standardistaion)



41	Philip Watkins	Ethekwini Electricity
42	Rajesh Devparsad	Ethekwini Electricity
43	Shadrack Palmer	FNB - Scion - Prepaid
44	James H Muller	Holistic
45	Ari Geva	Intelligent Metering Systems (IMS)
46	T Synman	Intelligent Metering Systems (IMS)
47	Henty Waker	iPay
48	Joel C	iPay
49	Mohamed Ebrahim	Itron Africa
50	Wayne Berkinshaw	Kwikpay
51	Alex Braude	Landis & Gyr
52	Jannie Vermeulen	Landis & Gyr
53	Kobus van den Berg	Mangaung Local Municipality
54	Nigel R ebdon	Meterteq
55	Kudzai Dana	Microsoft
56	Victor Ds Neves	Microsolve
57	Leon de Lange	MobilePay
58	Adrian Vermooten	MTN
59	Hitesh Morar	MTN
60	Israel Davimes	MTN
61	Clive Handley	namITech Limited
62	Johan van Vuuren	Nelson Mandela Metropolitan Municipality
63	Abrie Marais	Novitium
64	Jacques Kruger	Novitium
65	Philip Helberg	Novitium
66	Philip Lavers	OAS
67	Stephen van Der Mervwe	OAS
68	Ian Steyn	Pism TranSwitch Services
69	Colin Deirino	PN Energy
70	Domingos Dias	Post Office
71	Domingos Dias	Post Office
72	Velasen Naidoo	Post Office
73	Gareth Moore	Power Vend
74	Kevin Roberts	Power Vend
75	Richard Makhafola	Qhakaza Amps
76	Raino Botha	Qmuzik
77	Charl Mostert	Sage
78	Raghu Kalakonda	Sage
79	Busisiwe Mbuyisa	SAPO
80	Carl Du Plessis	SAPO
81	Dan Mbatha	SAPO
82	Johnny Mthoa	Savaya Telecoms
83	Martin Deufel	Smartec Technologies (Pty) Ltd



84	Martin Goldblatt	Smartec Technologies (Pty) Ltd
85	Dawid Loubser	Solms Training & Consulting
86	Dr. Fritz Solms	Solms Training & Consulting
87	CT Arul	SRV Telecom Private Limited
88	Ivor Chalton	Syntell (Tellumat)
89	Mark Linley	Syntell (Tellumat)
90	Deon Botha	Talknet
91	Johan Prinsloo	Talknet
92	Richard Schubert	Talknet
93	Werner van Ghent	The Linkafrica Network
94	Patience Mlengana	Tshahani Resources (Pty) Ltd
95	Dirk Pieterse	Tshwane Electricity
96	Jan Westenraad	Tshwane Electricity
97	Gcobane Quvile	Ukukhanya Resources CC
98	Taliesin Sisson	University Of Kwa-Zulu Natal
99	Prof W T Penzhorn	University of Pretoria
100	Kenny Setzin	V60 (Pty) Ltd (Trading as Sediba Corporation)
101	Chris Slater Jones	VCA
102	Gordon Ashby	VCA
103	Phillip Scheel	VCA