ISBN 978-0-626-22828-6

SANS 1524-6-10:2010
Edition 1

NRS 009-6-10:2010
Edition 1

SOUTH AFRICAN NATIONAL STANDARD

Electricity payment systems

Part 6-10: Interface standards — Online vending server — Vending clients

This national standard is the identical implementation of NRS 009-6-10:2010 and is adopted in terms of a Memorandum of Agreement between the Electricity Suppliers Liaison Committee and the SABS Standards Division.

Published by SABS Standards Division
1 Dr Lategan Road Groenkloof ✉ Private Bag X191 Pretoria 0001
Tel: +27 12 428 7911 Fax: +27 12 344 1568
www.sabs.co.za

SANS 1524-6-10:2010
Edition 1
NRS 009-6-10:2010
Edition 1

Table of changes

Change No.	Date	Scope

Foreword

This South African standard was prepared by a working group of the Electricity Suppliers Liaison Committee and adopted by National Committee SABS TC 62, *Electrical measurements,* in accordance with procedures of the SABS Standards Division, in compliance with annex 3 of the WTO/TBT agreement.

The adoption has been done in terms of a Memorandum of Agreement between the Electricity Suppliers Liaison Committee and the SABS Standards Division.

This document was published in xxxxx 2010.

NRS 009-6-10:2010

Edition 1

ELECTRICITY PAYMENT SYSTEMS

Part 6-10: Interface standards — Online vending server — Vending clients

This specification is issued by
the Standardization Section, Eskom,
on behalf of the
User Group given in the foreword

Table of changes

Change No.	Date	Text affected

Correspondence to be directed to

The NRS Projects Manager
The Standardization Section
Eskom
Private Bag X13
Halfway House 1685

Telephone : (011) 651 6832
Fax : (011) 651 6827
E-mail : nrs@eskom.co.za

Website : http://www.nrs.eskom.co.za

Printed or electronic copies are obtainable from

SABS Standards Division
Private Bag X191
Pretoria 0001

Telephone : (012) 428-7911
Fax : (012) 344-1568
E-mail : sales@sabs.co.za

Website : http://www.sabs.co.za

Printed in the Republic of South Africa
by the SABS Standards Division
1 Dr Lategan Road, Groenkloof, Pretoria

Notice

NRS 009-6-10 (XMLVend specification) was compiled with active participation and sharing of information by several utilities, vending service providers, vending equipment suppliers and other interested industry stakeholders. XMLVend is an industry specification with the key objective to provide a secure, standards-based, open, non-proprietary platform for providing online vending services.

The compilers acknowledge that the vending environment and user requirements are continuously evolving and that the current version of the specification may not always support these changing requirements. Therefore, utility-driven extensions to the specification are allowed and encouraged. However, such extensions should only be done in a manner that acknowledges the open and sharing culture that has been adopted in the development of XMLVend.

Therefore, it is expected that any extensions are made available freely and without limitations to the XMLVend working group and users of the specification. This ensures that openness and interoperability between XMLVend implementations are promoted and maintained. It also provides a mechanism for updates and new requirements to be added to future specification versions. Please refer to the inside cover for the contact details of the NRS Project Manager to whom such updates should be referred.

Further information concerning XMLVend can be obtained from its website:
http://www.nrs.eskom.co.za/xmlvend

Foreword

This part of NRS 009 was prepared on behalf of the Electricity Suppliers Liaison Committee (ESLC) and approved by it for use by supply authorities in South Africa.

This part of NRS 009 was prepared by a working group which, at the time of publication, comprised the following members:

S J van den Berg (Chairman)	Mangaung Municipality (Centlec)
N Ballantyne	City of Cape Town
R Devparsad	eThekwini Electricity
P A Johnson	IARC (Resources & Strategy)
W L Mathers	Ekurhuleni Metropolitan Municipality
J O'Kennedy	IARC (Resources & Strategy)
T Sibiya	IARC (Resources & Strategy)
M Singh	eThekwini Electricity
K Subramoney	IARC (Resources & Strategy) [Compiler]
D van Rooi	IARC (Resources & Strategy)
J van Vuuren	Nelson Mandela Metropolitan Municipality
K Venketiah	IARC (Resources & Strategy)
P Watkins	eThekwini Electricity
H Wouda	City Power Johannesburg

In addition to the NRS 009 working group members listed above, an "XMLVend working group" consisting of utilities, suppliers and other interested parties was extensively consulted during the compilation and preparation of this part of NRS 009. Their contribution to its development is acknowledged. The list of the XMLVend working group members is available at http://www.nrs.eskom.co.za/xmlvend.

NRS 009 was initially based on Eskom specification MC114, *Requirements specification for a common vending system for electricity dispensing systems,* and consists of the following parts, under the general title *Electricity sales systems:*

Part 0: Standard transfer specification – Synopsis. (Under consideration.)

Part 1: Glossary and system overview. (Withdrawn)

Part 2: Functional and performance requirements.
- *Section 1: System master stations.* (obsolete)
- *Section 2: Credit dispensing units.* (obsolete)
- *Section 3: Security module.* (obsolete)
- *Section 4: Standard token translators.* (obsolete)
- *Section 5: Error handling.* (obsolete)

Part 3: Database format. (obsolete)

Part 4: National prepayment electricity meter cards.

Part 5: Testing of subsystems. (obsolete)

Part 6: Interface standards.
- *Section 1: Credit dispensing unit – Standard token translator interface.* (obsolete)
- *Section 2:* Not used.
- *Section 3: System master station – Credit dispensing unit.* (obsolete)
- *Section 4: Data transfer by physical media – System master station – Credit dispensing unit.* (obsolete)
- *Section 5: Not allocated*

Foreword *(concluded)*

Section 6[1)]*: Standard transfer specification/Credit dispensing unit – Electricity dispenser – Categories of token and transaction data fields.*
Section 7[1)]*: Standard transfer specification/Credit dispensing unit – Electricity dispenser – Token encoding and data encryption and decryption.*
Section 8[1)]*: Standard transfer specification/Disposable magnetic token technology – Token encoding format and physical token definition.*
Section 9[1)]*: Standard transfer specification/Numeric token technology – Token encoding format and physical token definition.*

Part 7[1)]*: Standard transfer specification/Management of cryptographic keys.*

Part 8: The management of secure modules.

The NRS 009 series is being adopted and published as dual-numbered standards under the designation SANS 1524/NRS 009. The following documents will be available under the general title *Electricity payment systems*:

Part 1: Payment meters.

Part 1-1: Mounting and terminal requirements for payment meters.

Part 1-2: Surge protective devices for the protection of payment meters.

Part 4: National prepayment electricity meter cards.

Part 6-10: Interface standards – Online vending server – Vending clients.

Part 7: Standard transfer specification/Management of cryptographic keys.

Part 8: The management of secure modules.

Part 9: Implementing vending systems.

Annexes A to D for information only.

1) Parts and sections of NRS 009 which specify the Standard Transfer specification have been published by the IEC as IEC 62055-41 and IEC 62055-51, *Electricity metering – Payment metering system – Standard transfer specification (STS).*

Introduction

Prepayment vending systems play a critical role in supporting electricity prepayment-metering infrastructure. Current standardized offline vending systems enable convenient access to point of sale (POS) or credit dispensing units (CDUs) for customers to purchase electricity.

However, several electricity distributors wished to use online vending systems, and in the absence of an acceptable industry standard, chose different proprietary systems. Although these systems represent a technological step forward for prepayment vending, they could have a detrimental effect on an already standardized prepayment industry. This prompted the standard transfer specification user group, under the auspices of the Electricity Supply Liaison Committee (ESLC), to initiate the online vending standardization project.

Online vending systems require the exchange of messages relating to prepayment electricity vending transactions between a vending server and vending client. Therefore a secure interface protocol is required to facilitate the message exchanges between servers and clients that employ specifications at different application levels.

This part of NRS 009, NRS 009-6-10, also called the XMLVend specification, was prepared by the XMLVend working group to establish, promote and implement the specified interface protocol between an XMLVend server and XMLVend clients. The interface protocol is hereafter referred to as the XMLVend protocol.

This part of NRS 009, which is the first edition of NRS 009-6-10 to be formally released and published by the SABS Standards Division on behalf of the ESLC, is technically equivalent to what was released by the NRS 009 working group as “version 2.1” during the development phase. Version 2.1 was released as an interim NRS specification in February 2006 and superseded the initial version 1 (February 2004) and version 2 (November 2005).

Keywords

electricity dispenser, electricity sales systems, online vending, payment systems, prepayment, standard transfer specifications, token web services, XML, XMLVend.

Contents

This page intentionally left blank

ELECTRICITY PAYMENT SYSTEMS

Part 6-10: Interface standards – Online vending server – Vending clients

1 Scope

1.1 Purpose

This part of NRS 009 provides the necessary information to implement the XMLVend protocol as part of an online prepayment electricity vending system. This part of NRS 009 consists of a collection of material from several different sources and is intended to create a single source of information that will provide the necessary basics to understand and implement XMLVend protocol.

This part of NRS 009-6 is divided into two parts:

a) part 1, *User requirements*, which is aimed at both users (utilities) and suppliers. This part outlines the implementation models, business application domains and the content and structure of the use case message pairs; and

b) part 2, *Web services implementation*, which is aimed mainly at suppliers. This part describes the technical detail of how the use case message pairs should be securely exchanged using web services. The XMLVend reference model is illustrated in figure 1.

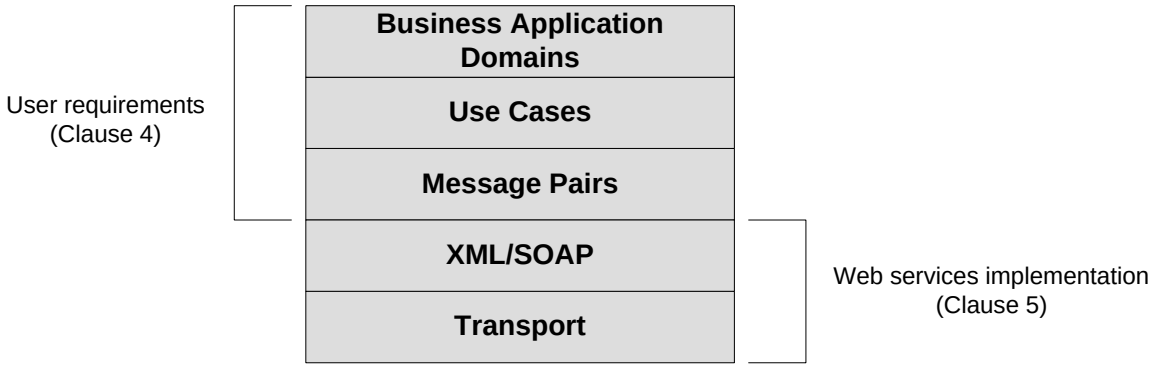

Figure 1 — XMLVend reference model

This approach is to deliberately reach both the utilities and suppliers of XMLVend compliant systems. Based upon the reader's experience and intentions, specific clauses of this part of NRS 009 might be more valuable than others.

This part of NRS 009 is meant to be used in conjunction with the latest versions of the XMLVend WSDL and schema documents, which are available on the XMLVend website.

1.2 Items outside the scope

Payment transactions will be processed outside XMLVend, but the tender type (for example, cash, credit cards) is covered in this part of NRS 009.

XMLVend does not directly support clients contracted with more than one utility and would therefore have to communicate with servers that are managed by different utilities.

The protocol specifically excludes the case of an end customer who transacts directly with the XMLVend server. If a customer would like to purchase a prepayment token online through for example, the internet, the customer would have to connect to an intermediate entity, in this case, a web server. The web server would then handle the web browser presentation layer details and in the end, it would communicate with the XMLVend server as described in the protocol. The web server in this instance would therefore act as an XMLVend client to handle the communications with the XMLVend server.

2 Normative references

The following documents contain provisions which, through reference in this text, constitute provisions of this part of NRS 009. All documents are subject to revision and, since any reference to a document is deemed to be a reference to the latest edition of that document, parties to agreements based on this specification are encouraged to take steps to ensure the use of the most recent editions of the documents listed below. Information on currently valid national and international standards can be obtained from the SABS Standards Division.

2.1 Standards and specifications

Hypertext Transfer Transport Protocol (HTTP) protocol specification, version 1.1. Available at <http://www.w3c.org>.

NRS 009, *Electricity sales systems – Addendum 07/2001 – STS EBSST implementation for initial pilot sites.*

NRS 009-2-2, *Electricity sales systems – Part 2: Functional and performance requirements – Section 2: Credit dispensing units.*

NRS 009-4, *Electricity sales systems – Part 4: National prepayment electricity meter cards.*

NRS 009-6-7, *Electricity sales systems – Part 6: Interface standards – Section 7: Standard transfer specification/Credit dispensing unit – Electricity dispenser – Token encoding and data encryption and decryption.*

SANS 62051/IEC 62051, *Electricity metering – Glossary of items.*

WSDL, SOAP and XML schema specifications. Available at <http://www.w3c.org>.

WS-I Usage Scenarios, version 1.01. Available at <http://www.ws-i.org>.

WS-I Basic Profile, version 1.0a. Available at <http://www.ws-i.org>.

GZIP specifications. Available at http://www.gzip.org.

2.2 Other publications

O'KENNEDY JC, *Generic online vending glossary and definition of terms*, version 1.12. Available at <http://www.nrs.eskom.co.za/xmlvend>.

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purposes of this document, the terms, definitions and abbreviations given in SANS 62051, the terms, definitions and abbreviations contained in the *Generic online vending glossary and definition of terms* (see 2.2) and the following apply.

3.2 Abbreviations

API: application program interface

AT: algorithm type

B2B: business to business

CDU: credit dispensing unit

CHRG: charge

ebxML: electronic business extensible markup language

FBE: free basic electricity

HTTP: hypertext transport protocol

HTTPS: secure hypertext transport protocol

IP: internet protocol

JAX_RPC: java API for XML remote procedure call

KCT: key change token

KRN: key revision number

MCT: meter credit transformer

MSNO: meter serial number

POS: point of sale

SOAP: simple object access protocol

SGC: supply group code

SM: security model

SSL: secure socket layer

SMTP: simple mail transfer protocol

STS: standard transfer specification

TCP: transport control protocol

TI: tariff index

TT: token technology

TLS: transport layer security

UDDI: universal description discovery and integration

UML: unified modelling language

URI: universal resource identifier

WSDL: web services description language

WSI: web service interoperability

XML: extensible markup language

4 User requirements

NOTE Clause 4 is aimed at both utilities that use XMLVend compliant equipment and suppliers that develop XMLVend protocol equipment (client and server). Clause 4 outlines the implementation models, the business application domains, the use cases and the content and structure of the use case message pairs.

4.1 Implementation models

4.1.1 Introduction

The implementation models illustrate the two models that may be used to communicate vending requests to an XMLVend server.

4.1.2 Implementation model 1

Implementation model 1, sometimes referred to as the “normal vendor” model, illustrates that the customer interacts with an XMLVend client (through an operator) that communicates directly with the XMLVend server. This model may be applied to (but is not limited to) implementations, where vendors who currently use offline credit dispensing units are upgraded to online vending clients. Implementation model 1 is illustrated in figure 2.

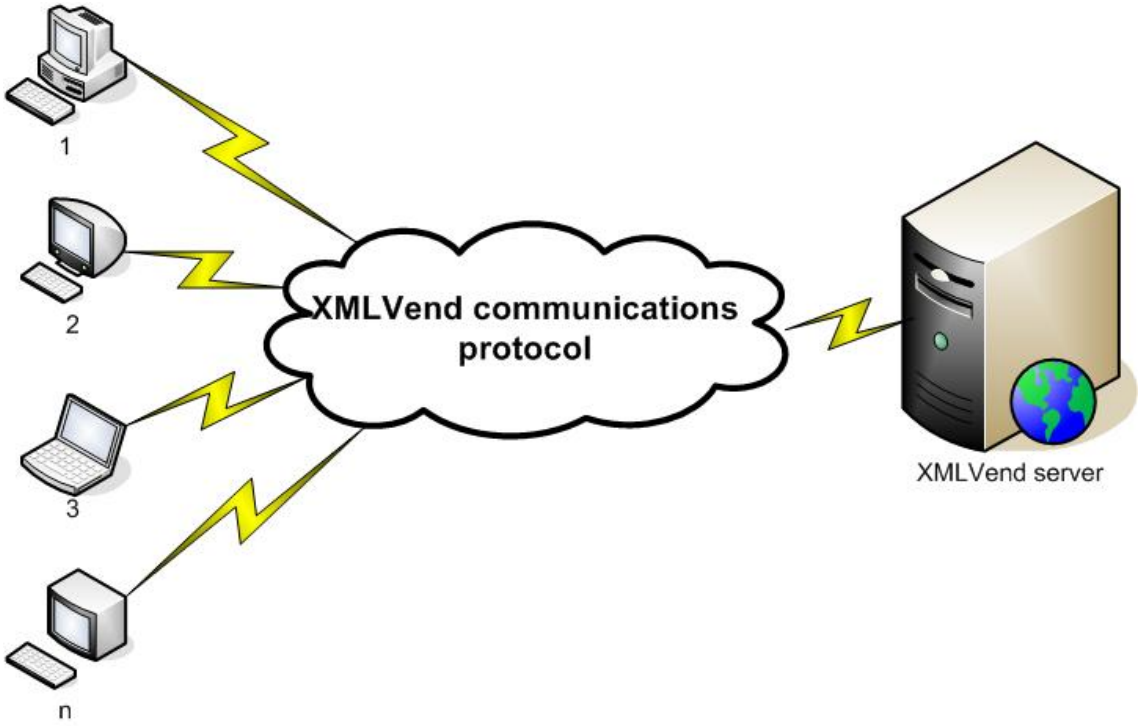

Figure 2 — Implementation model 1

4.1.3 Implementation model 2

Implementation model 2, sometimes referred to as the “client server” model, illustrates that the customer interacts with terminals, that communicate using a proprietary protocol communicate to a XMLVend client which then formats these requests as XMLVend requests and submits these to the XMLVend server.

This model may be applied to (but is not limited to) vendors who currently have a footprint of terminals and would like to add prepayment electricity vending capabilities to the terminals. Implementation model 2 is illustrated in figure 3.

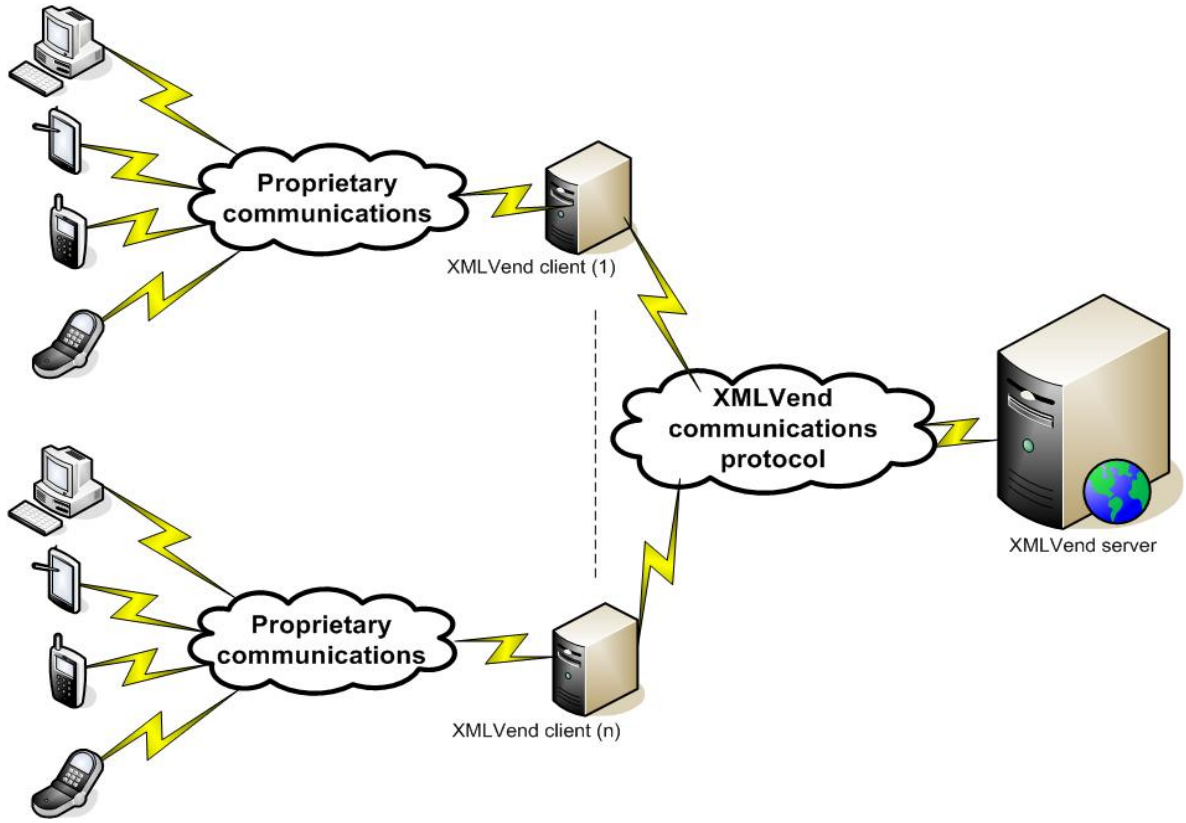

Figure 3 — Implementation model 2

4.2 Use cases[2)]

XMLVend use cases are used to describe the functionality exposed by the XMLVend protocol.

Each use case is aimed at complying with specific requirements of the following use case actors:

a) customers;
b) vending operators;
c) technical operators;
d) technicians; and
e) XMLVend clients.

XMLVend utilities do not have to implement all the XMLVend use cases but should select those that most suit their requirements.

4.3 Use case actors, responsibilities and collaborators

Table 1 describes use case actors, their responsibilities and collaborators.

2) A use case is defined as a single task, performed by the end user of a system, which has some useful outcome to the end user.

Table 1 — Roles, responsibilities and collaborators

1	2	3
Use case actor	**Responsibilities**	**Collaborators**
Customer	• Initiates customer use cases. • Provides identification information. • Tenders payment. • Receives requested token receipt.	• Vending operator.
Vending operator	• Verifies customer identification. • Submits vending requests on behalf of customer. • Performs operator specific tasks such as open batches and close batches. • Hands generated tokens to customers. • Handles customer queries.	• XMLVend client
Technical operator	• Performs meter management requests.	• XMLVend client
XMLVend client	• Authenticates the XMLVend Server. • Compiles and sends XMLVend request messages to XMLVend server. • Receives and formats XMLVend response messages from the XMLVend server. • Automatically initiates the “Issue Advice” use case for use cases that might require this service.	• XMLVend server
XMLVend server	• Authenticates the XMLVend clients • Complies with an appropriate XMLVend response message based on its application business logic. • Responds with a fault response message, if required.	• XMLVend clients

4.4 Use case domains

4.4.1 Grouping of use cases

The use cases have been divided into the following two business application domains:

a) revenue management; and

b) meter management.

A further grouping of use cases, referred to as base use cases have also been defined. These use cases do not belong to any specific business application domain but are used in both the revenue and meter management domains.

The grouping of the use cases into domains eases the implementation and maintenance of XMLVend. It is also envisaged that some XMLVend implementations may only implement the revenue management use cases and some only meter management use cases.

The XMLVend web service description language and schema documents have also been separated into these two domains with a common base schema. See 5.2.

4.4.2 Revenue management domain

The revenue management domain is concerned with credit vending, revenue collection and management. The 17 use cases that belong to this domain are illustrated in figure 4.

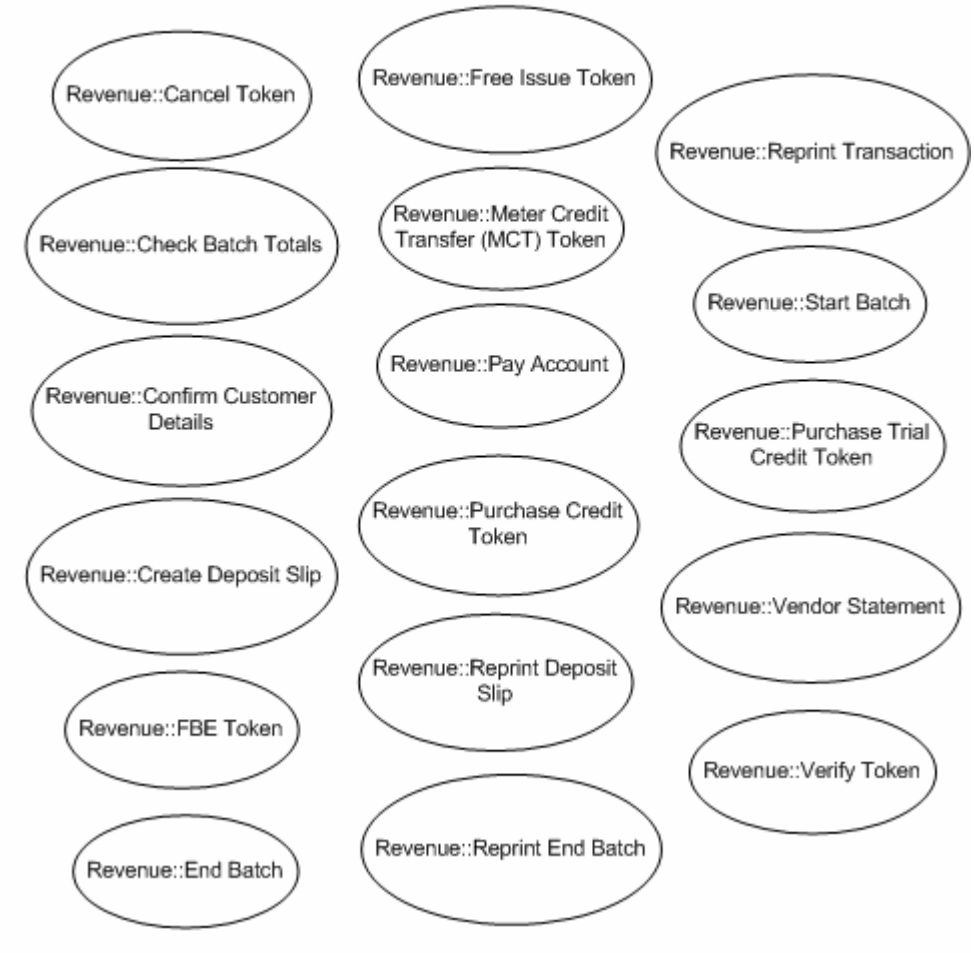

Figure 4 — Revenue domain use cases

4.4.3 Meter management domain

The meter management domain is concerned with meter management functions. The five use cases that belong to this domain are illustrated in figure 5.

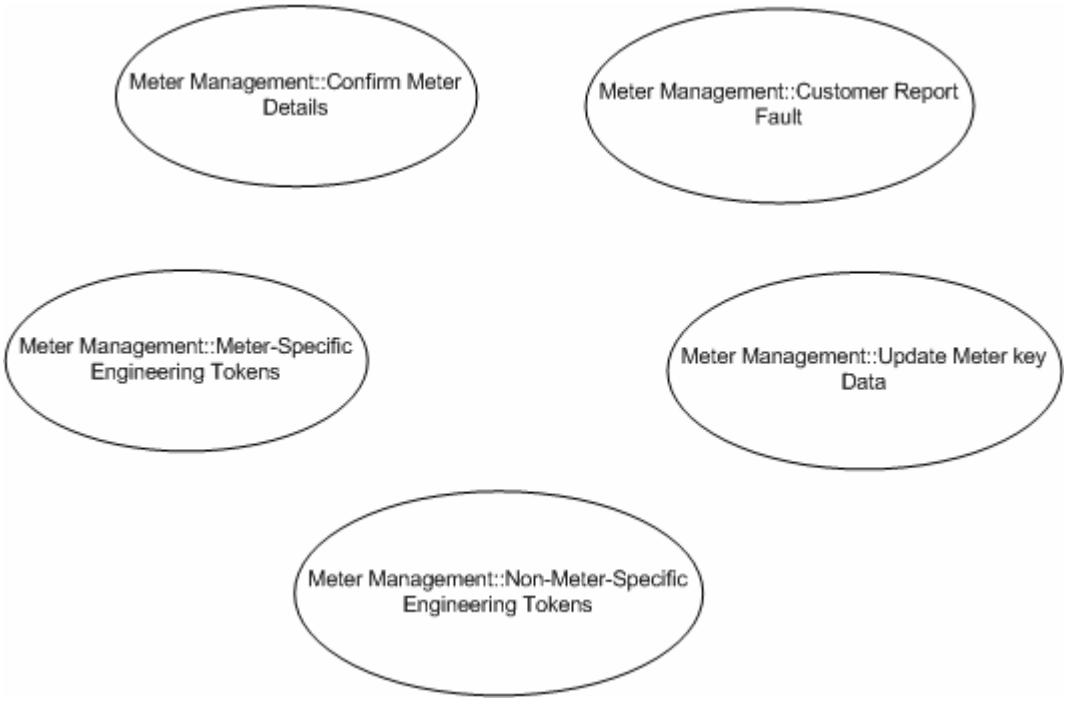

Figure 5 — Meter domain use cases

4.4.4 Base domain

The base domain use cases that do not belong to any specific business application domain but are used by both the revenue and meter management domains. The only use case currently defined for this domain is the Issue Advice Use Case.

4.5 Use case descriptions

4.5.1 General

The use case description provides the business rationale and detailed description of each XMLVend use case. The use cases have also been specified as generic as possible to ensure that they comply with the requirements of all potential utilities.

Utilities will tailor the use case descriptions to comply with their specific requirements and compile these into their own implementation-specific use case description document. An example of how this can be approached is provided as supporting documentation, on <http://www.nrs.eskom.co.za/xmlvend>.

4.5.2 Revenue domain use cases

4.5.2.1 Cancel token

Description
The cancel token use case is used to cancel a previous credit vend transaction. This use case only applies to disposable magnetic card tokens.

NOTE It is the responsibility of the client device to destroy the token information stored on the magnetic card. This use case can potentially be abused for fraudulent purposes. If it has to be supported, it should be carefully controlled and monitored.

This use case corresponds to the “cancel token” transaction number 006 described in NRS 009, addendum 07/2001.

Desired outcome
The encoded token data is erased.

Dependencies
Follows: purchase credit token
Includes: issue advice, issue fault (if required)

Preconditions
The following apply:
a) a customer wants to cancel an unused magnetic card encoded token;
b) the reprint token use case is disabled;
c) the XMLVend server supports this use case;
d) the operator is authorized to initiate this use case; and
e) the same amount of credit as originally issued is contained on magnetic card data.

Postconditions
The following apply:
a) a “cancel token” transaction is recorded on the server;
b) the encoded token data is erased; and
c) the customer receives the applicable refund.

Participants
The following apply:
a) customer;
b) vending operator;
c) XMLVend client; and
d) XMLVend server.

Happy path
The cancel token happy path is illustrated in figure 6.

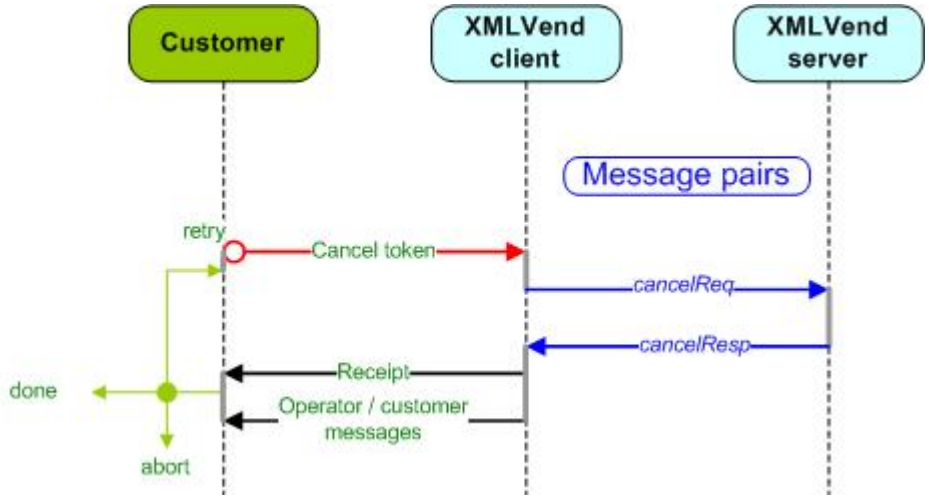

Figure 6 — Cancel token happy path

Implementation
The “cancel token” and “reprint token” use cases are mutually exclusive. If cancellations of tokens are allowed, then reprints shall not be allowed.

In the event of an exemption occurring, it is highly recommended that this use case use the Issue Advice Use Case to ensure that client and sever have the same understanding of the use case’s outcome on both server and client. See 4.5.4.2.

4.5.2.2 Check batch totals

Description
The check batch totals is similar to the End Batch Use Case (see 4.5.2.6), except that it does not close the batches but just returns the request batch summaries.

Desired outcome
The requested batch summary is returned.

Preconditions
The requested batch shall be open.

Postconditions
The batch summary is returned to the client.

Participants
The following apply:
a) vending operator;
b) XMLVend client; and
c) XMLVend server.

Happy path
The happy path is illustrated in figure 7.

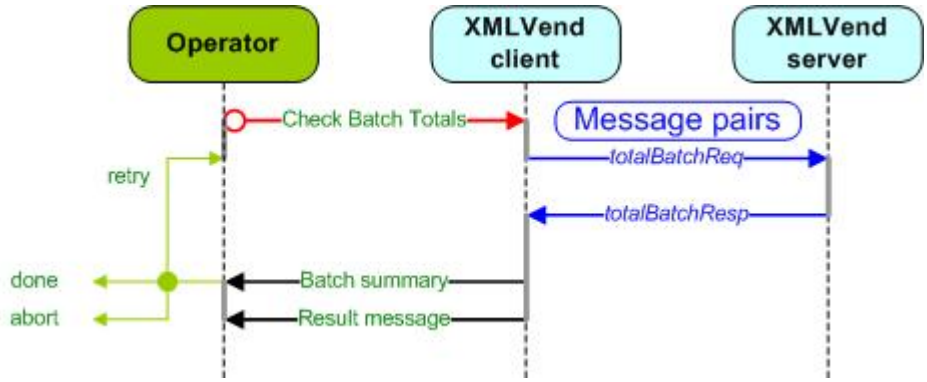

Figure 7 — Check batch totals happy path

Implementation
None.

4.5.2.3 Confirm customer details

Description
The confirm customer use case is used to confirm the customer details associated with a particular meter.

Desired outcome
The customer details that match the search criteria entered are returned.

Preconditions
The XMLVend server supports this use case.

The vending operator is authorized to initiate this use case.

A customer identification parameter or meter identification parameter that is supported by the server shall be provided, such as

a) customer name;
b) customer address;
c) customer account number;
d) customer ID number;
e) meter card (track 2 data);
f) meter serial number; and
g) meter configuration data, usually from an old token.

Postconditions
The customer details that match the identification parameter entered are returned to the client.

Participants
The following apply:
a) customer;
b) vending operator;
c) XMLVend client; and
d) XMLVend server.

Happy path
The use case sequence diagram is illustrated in figure 8.

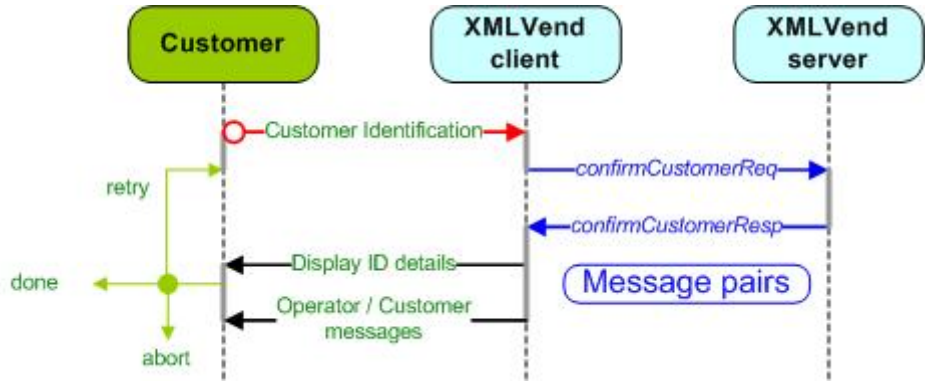

Figure 8 — Confirm customer details happy path

Implementation
This use case should not normally be used in conjunction with normal vending use cases since there is a performance and cost penalty.

4.5.2.4 Create deposit slip

Description
The create deposit slip use case is used to assist the vendor to bank his sales. It is usually used by implementations that do not use batches and operate on an “upfront” business model.

NOTE The original requirement to tie a banking batch and sales batches to a deposit reference does not apply in an “upfront” model. In a “non-upfront” model, the banking deposits are normally linked to the banking batch to reconcile deposits and sales. For the upfront model such reconciliation is not required since available credit is validated for every individual transaction.

Desired outcome
A deposit slip is generated that can be used by the vendor to bank his sales.

Preconditions
None.

Postconditions
The client prints a deposit slip.

Participants
The following apply:
a) vending operator;
b) XMLVend client; and
c) XMLVend server.

Happy path
The happy path is illustrated in figure 9.

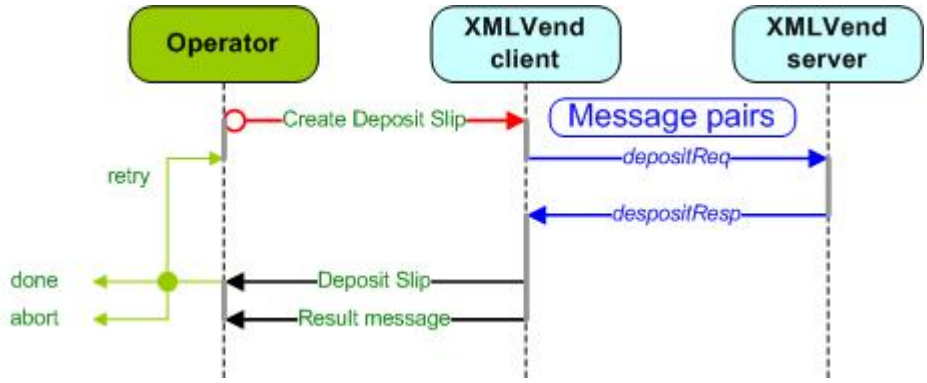

Figure 9 — Create deposit slip happy path

Implementation
None.

4.5.2.5 Collect FBE token

Description
The collect FBE token use case is used to request the free basic electricity (FBE) token. This is a particular type of electricity credit token designed to issue a pre-programmed credit amount to each meter once per month, free of payment.

This use case corresponds to the electricity basic support service token (EBSST) transaction (type 010) described in NRS 009, addendum 07/2001.

NOTE The term “EBSST” (Electricity basic support service token) has been replaced by “FBE” (Free basic electricity) token which is used throughout this part of NRS 009.

Desired outcome
The customer collects the FBE token.

Preconditions
The following apply:
a) the XMLVend server supports this use case;
b) the vending operator is authorized to initiate this use case; and
c) an identification parameter supported by the server for this use case shall be supplied, such as
   1) a meter card (track 2 data) (see NRS 009-4);
   2) a meter serial number; or
   3) meter configuration data, usually from an old token.

Postconditions
The following apply:
a) the server security module (SM) generates the requested FBE token;
b) an “FBE token” transaction is recorded on the server; and
c) the customer receives the requested FBE token.

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The happy path scenario is illustrated in figure 10.

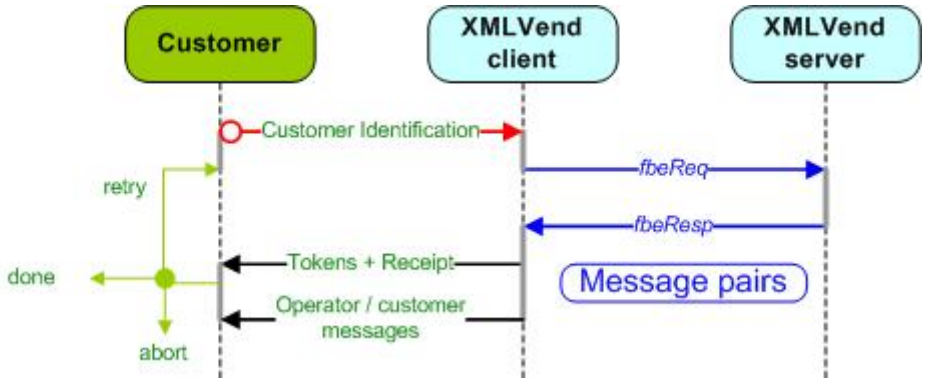

Figure 10 — Free basic electricity happy path

Implementation
This use case is specifically silent on the method of identifying whether a customer qualifies for FBE or what the amounts are that would apply. These and other parameters are not defined by XMLVend but are instead managed by the server.

4.5.2.6 End batch

Description
The end batch use case closes an open banking, sales or shift batch as defined in NRS 009-2-2. It may also be used to open a banking, sales and shift simultaneously.

The “end banking batch” response message may also be used to compile a deposit slip to assist the vendor.

Desired outcome
The requested batch type is closed and a banking deposit slip is optionally printed.

Preconditions
The requested batch shall be open.

Postconditions
The following apply:
a) the request batch type is closed;
b) the batch summary is returned to the client; and
c) banking batch closures may optionally allow the vending operator to print a deposit slip.

Participants
The following apply:
a) the vending operator;
b) the XMLVend client; and
c) the XMLVend server.

Happy path
The happy path is illustrated in figure 11.

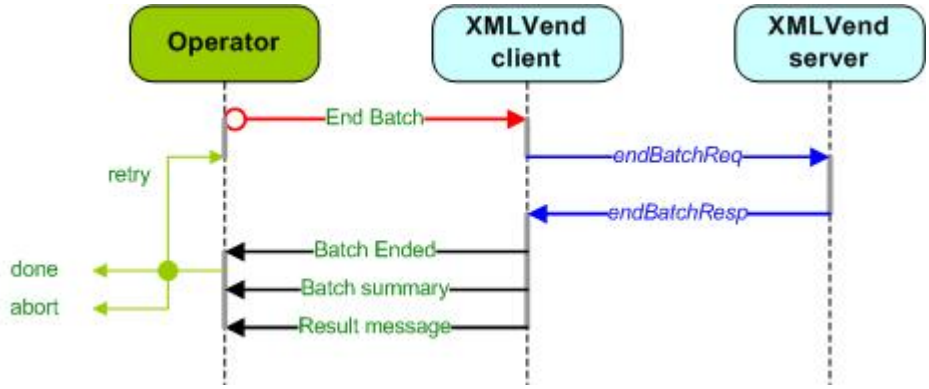

Figure 11 — End batch happy path

Implementation
In the event that an exception occurs, it is highly recommended that this use case use the Issue Advice Use Case, to ensure that client and server have the same understanding of the use case's outcome on both server and client. See 4.5.4.2.

4.5.2.7 Free issue token

Description
The free issue token is similar to the "purchase credit token" use case except that it is recorded as a "Free Issue" transaction.

This use case may be used by the utility to issue promotional or marketing credit tokens to customers at no cost to the customer.

Desired outcome
The customer receives a free issue token.

Preconditions
The following apply:
a) a customer requests a free issue token;
b) the XMLVend server supports this use case;
c) the operator is authorized to initiate this use case; and
d) an identification parameter supported by the server for this use case shall be supplied, such as
   1) a meter card (track 2 data) (see NRS 009-4);
   2) a meter serial number; or
   3) meter configuration data, usually from an old token.

Postconditions
The following apply:
a) the server security module generates a credit token;
b) a “free issue token” transaction is recorded on the server; and
c) the customer obtains the requested free issue token.

Participants
The following apply:
a) the customer;
b) the XMLVend client operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The happy path is illustrated in figure 12.

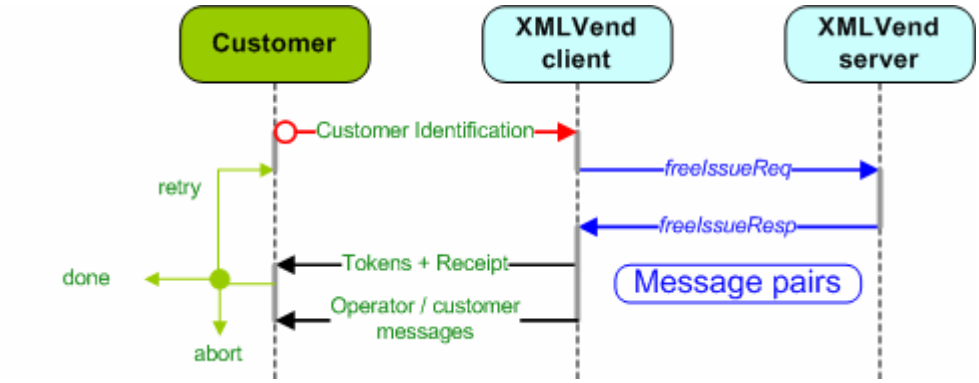

Figure 12 — Free issue token happy path

Implementation
In the event that an exception occurs, it is highly recommended that this use case use the Issue Advice Use Case, to ensure that client and server have the same understanding of the use case’s outcome on both server and client. See 4.5.4.2.

4.5.2.8 Meter credit transfer

Description
The meter credit transfer use case is used to reimburse a customer for remaining credit in a changed out meter. The customer is usually requested to provide an official “meter change out” form to the operator.

This use case corresponds to the “replacement token” transaction number 003 described in NRS 009, addendum 07/2001.

Desired outcome
The customer receives a credit token to the value stated on the “meter change out” form.

Preconditions
The following apply:
a) the customer has a “meter change out” form;
b) the XMLVend server supports this use case; and
c) the operator is authorized to initiate this use case.

Postconditions
The following apply:
a) the server security module generates a credit token;
b) a “meter credit transfer” transaction is recorded on the server; and
c) the customer obtains the requested meter credit transfer token.

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The happy path is illustrated in figure 13.

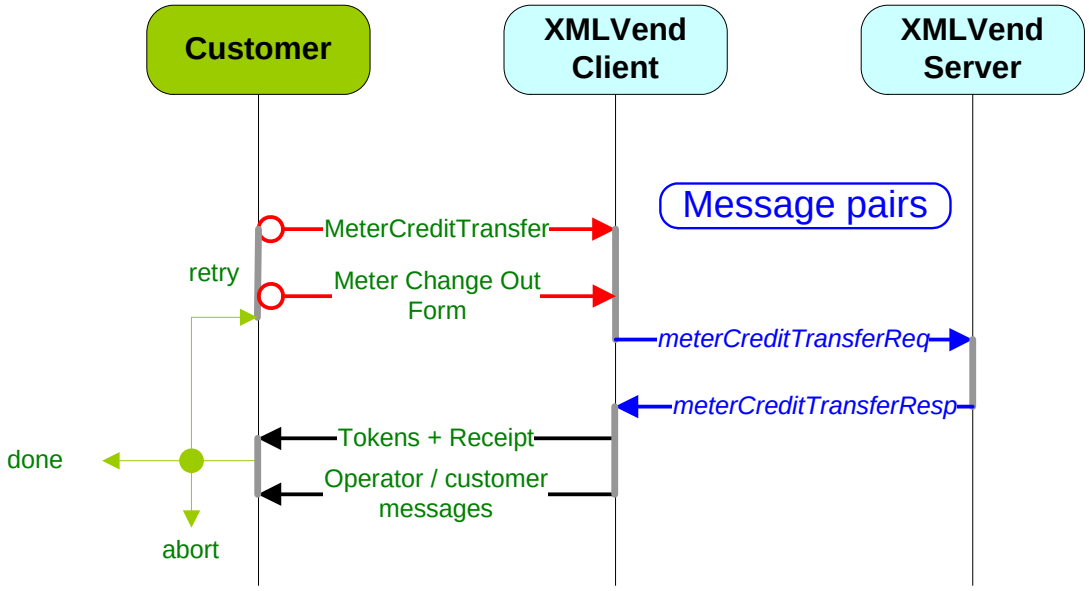

Figure 13 — Meter credit transfer token happy path

Implementation

NOTE The name has changed from “replacement token” since many people confused the name with variations of the “reprint transaction” (see 4.5.2.14), which is something totally different.

This use case is intended to issue new (i.e. replacement) credit to a customer if a faulty meter with credit has been exchanged with a new meter.

In the event that an exception occurs, it is highly recommended that this use case use the Issue Advice Use Case, to ensure that client and server have the same understanding of the use case’s outcome on both server and client. See 4.5.4.2.

4.5.2.9 Pay account

Description
The pay account use case provides a generic mechanism for a customer to pay an account that the server supports.

This use case can be used to correspond to the following transactions described in NRS 009, addendum 07/2001:

a) “fixed charge” number 004;
b) “account payment” number 007;
c) “recovery charge” number 013; and
d) “tariff surcharge” number 014.

Desired outcome
The customer pays the requested account.

Preconditions
The following apply:
a) a customer has requested to pay an account;
b) the customer has the required identification parameter/s to pay the account, such as
   1) the customer meter card (Track2Data) (see NRS 009-4),
   2) the meter serial number,
   3) the customer name,
   4) the customer address;
   5) the customer account number, and
   6) the customer ID number;
c) the server supports the requested account payment;
d) the XMLVend server supports this use case; and
e) the operator is authorized to initiate this use case.

Postconditions
The following apply:
a) the applicable transaction is recorded on the server; and
b) the customer’s account payment is accepted and a receipt is issued to the customer.

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The happy path is illustrated in figure 14.

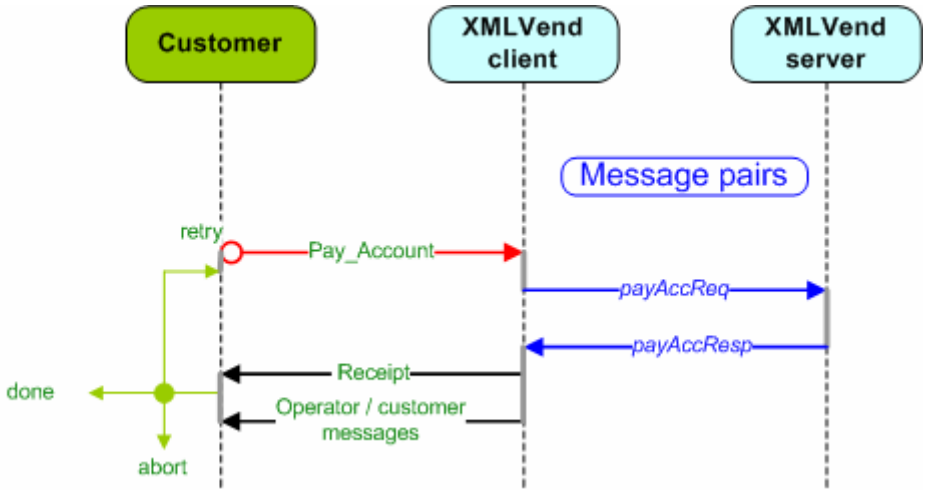

Figure 14 — Pay account happy path

Implementation
XMLVend is silent on the validation of the customer or account as this should be handled by the server functionality.

In the event that an exception occurs, it is highly recommended that this use case use the Issue Advice Use Case, to ensure that client and server have the same understanding of the use case's outcome on both server and client. See 4.5.4.2.

4.5.2.10 Purchase credit token

Description
The purchase credit token 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 in kilolitres (as applicable).

This use case corresponds to the "prepayment sale" transaction number 000 described in NRS 009, addendum 07/2001.

Desired outcome
The customer pays for and receives the purchased credit token.

Preconditions
The following apply:
a) the XMLVend server supports this use case;
b) the operator is authorized to initiate this use case;
c) the server business rules are met;
d) an identification parameter supported by the server for this use case shall be supplied, such as
    1) a meter card (track 2 data) (see NRS 009-4),
    2) a meter serial number, or
    3) meter configuration data, usually from an old token.

Postconditions
The following apply:
a) the server security module (SM) generates the requested credit token;
b) a "prepayment sale" transaction is recorded on the server; and
c) the customer obtains the requested credit token.

NOTE The server may also return an FBE token, a free issue token, or a pay account transaction together with a credit token (if so configured).

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The happy path is illustrated in figure 15.

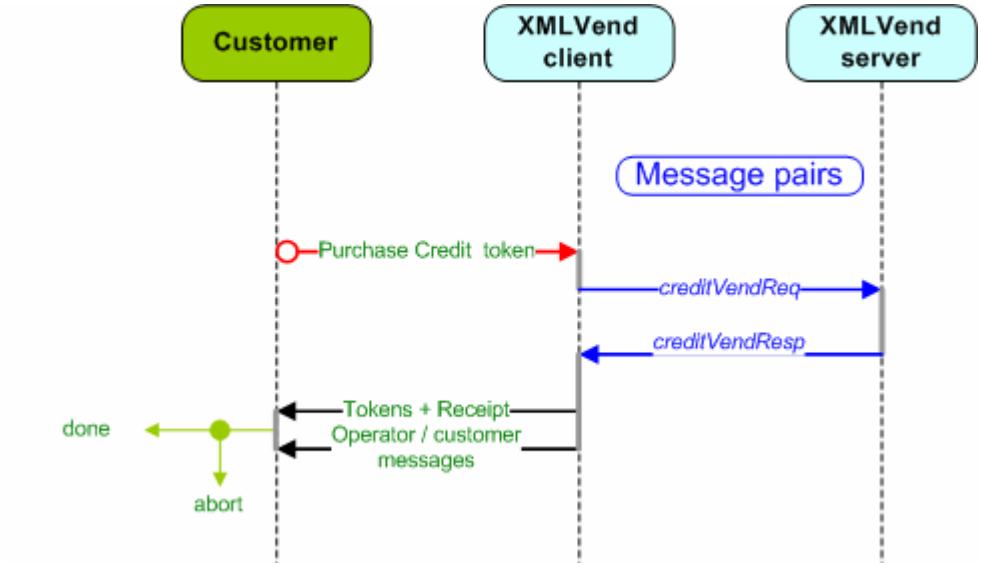

Figure 15 — Purchase credit token happy path

Implementation

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, a free issue token or a pay account transaction.

In the event that an exception occurs, it is highly recommended that this use case use the issue advice use case, to ensure that client and server have the same understanding of the use case's outcome on both server and client. See 4.5.4.2.

4.5.2.11 Purchase trial credit token

Description

The purchase trial credit token use case functions exactly like the purchase credit token use case, except that the server does not generate a "valid" credit token.

It can be used to determine the cost of a specific number of required units before committing to the transaction or it may be used to test the application layer communications between the client and server.

Desired outcome

The server processes the trial vend transaction exactly like a credit vend transaction except that the credit token returned will not be "valid".

Preconditions

The following apply:

a) the XMLVend server supports this use case;
b) the operator is authorized to initiate this use case;
c) an identification parameter supported by the server for this use case shall be supplied, such as
   1) a meter card (track 2 data) (see NRS 009-4),
   2) a meter serial number, or
   3) meter configuration data, usually from an old token.

Postconditions
The following apply:
a) a transaction may or may not be recorded on the server; and
b) a trial vend response is returned to the client.

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The happy path is illustrated in figure 16.

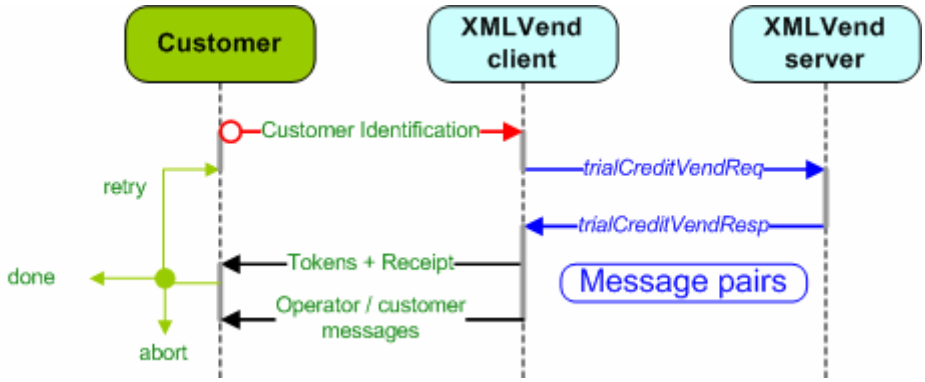

Figure 16 — Purchase trial credit token happy path

Implementation
The token cipher being returned shall not be a “valid” credit token but it shall still conform to requested algorithm type (AT) format. Returned STS tokens will be set to “00000 00000 00000 00000”. The receipt number shall be set to “0”.

4.5.2.12 Reprint deposit slip

Description
The reprint deposit slip use case reprints a “deposit slip” that might have been mislaid or soiled.

Desired outcome
A requested deposit slip is reprinted.

Preconditions
None.

Postconditions
The client reprints the last deposit slip.

Participants
The following apply:
a) the vending operator;
b) the XMLVend client; and
c) the XMLVend server.

Happy path
The happy path is illustrated in figure 17.

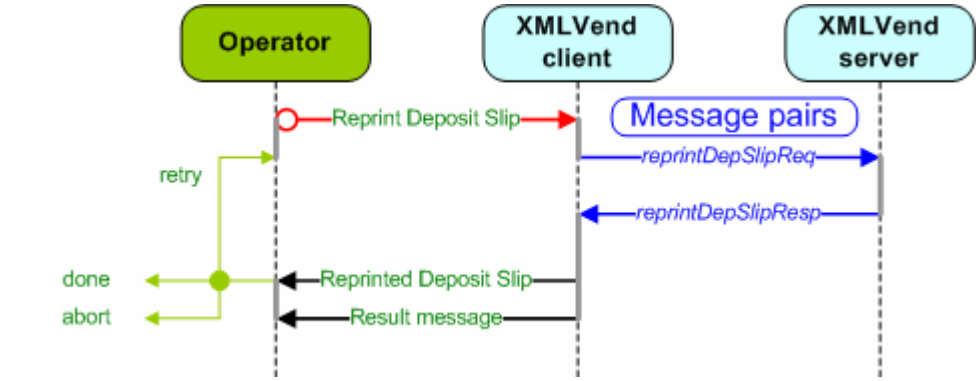

Figure 17 — Reprint deposit slip happy path

Implementation
None.

4.5.2.13 Reprint end batch

Description
The reprint end batch use case reprints the original end batch summary details that might have been mislaid or soiled. A specific batch or a batch and its “children” may be requested for reprint. If a specific batch is not requested, the last batch is reprinted.

Desired outcome
The requested end batch summary details are returned to the client. The response may include the associated deposit slip.

Preconditions
None.

Postconditions
The following apply:
a) the client obtains a reprint of the requested end batch summary details; and
b) the response may include the associated deposit slip.

Participants
The following apply:
a) the vending operator;
b) the XMLVend client; and
c) the XMLVend server.

Happy path
The happy path is illustrated in figure 18.

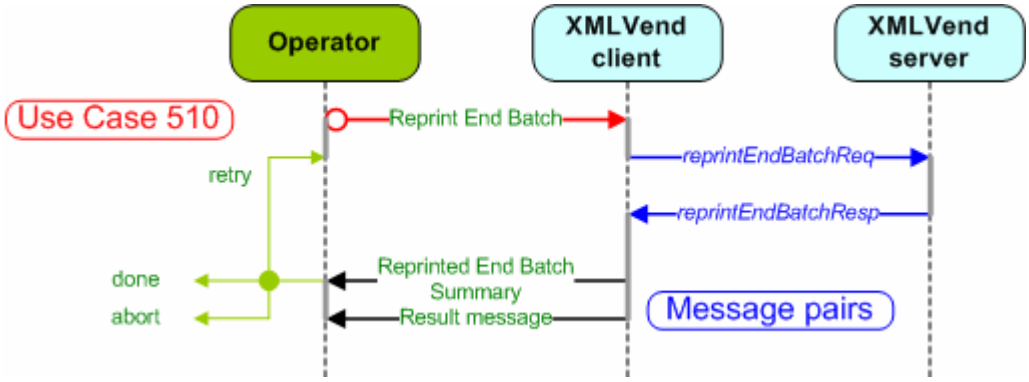

Figure 18 — Reprint end batch happy path

Implementation
None.

4.5.2.14 Reprint transaction

Description
The reprint transaction use case is used to reprint previous server transactions. The number of returned transactions is determined by the server.

This use case corresponds to the “Reprint” transaction (type 002) described in NRS 009-2-2.

Desired outcome
The customer collects the reprinted transaction receipt and possible token.

Preconditions
The following apply:
a) the XMLVend server supports this use case;
b) the operator is authorized to initiate this use case;
c) if the reprint transaction includes a token, the cancel token use case shall be disabled; and
d) an identification parameter supported by the server for this use case shall be supplied, such as
   1) a meter card (track 2 data) (see NRS 009-4),
   2) a meter serial number, or
   3) meter configuration data, usually from an old token.

Postconditions
The following apply:
a) a “Reprint” transaction is recorded on the server; and
b) the customer receives a reprinted transaction receipt and possibly a token.

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The happy path is illustrated in figure 19.

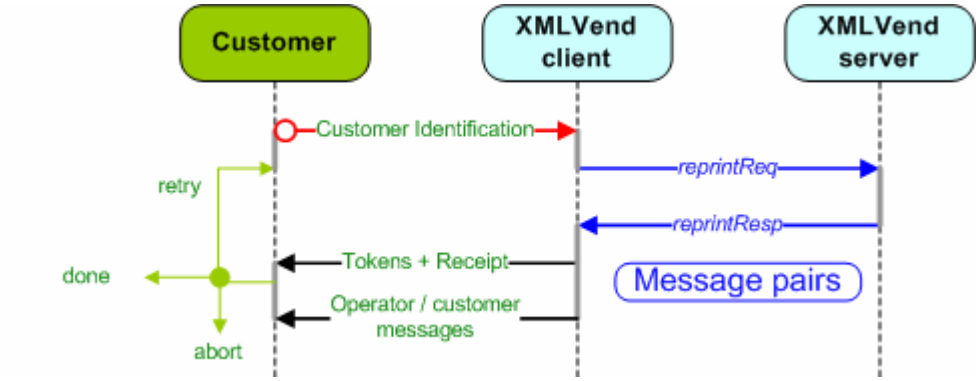

Figure 19 — Reprint transactions happy path

Implementation
If reprints of tokens are allowed, then cancellation of the tokens shall not be allowed.

4.5.2.15 Start batch

Description
The start batch use case is used to open banking, sales or a shift batch as defined in NRS 009-2-2.

Desired outcome
The request batch type is started.

Preconditions
The parent batch shall have been started.

Postconditions
The following apply:
a) the request batch type is started; and
b) the XMLVend server allocates a batch reference number to the batch as defined in NRS 009-2-2.

Participants
The following apply:
a) the vending operator;
b) the XMLVend client; and
c) the XMLVend server.

Happy path
The happy path is illustrated in figure 20.

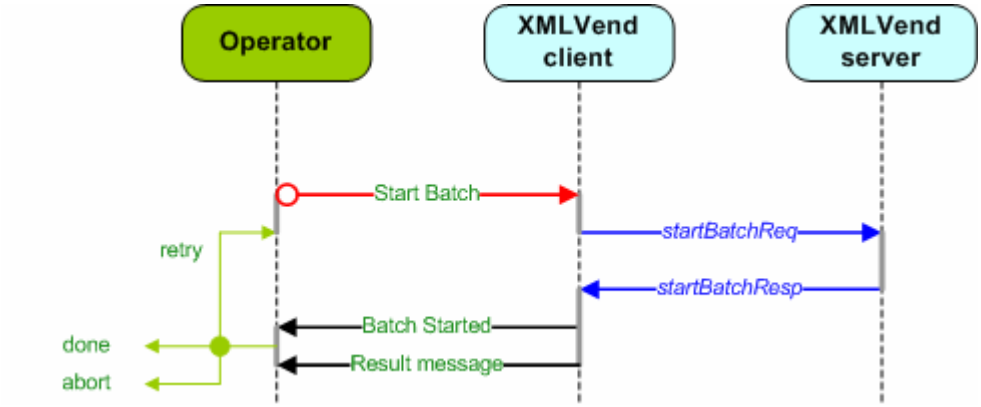

Figure 20 — Start batch happy path

Implementation
It is recommended that this use case make use of the Issue Advice Use Case to ensure that client and server have the same understanding of the use case's outcome on both server and client.

4.5.2.16 Vendor statement

Description
The vendor statement use case assists the vendor to reconcile deposits and sales with the account balance and available credit by providing the vendor with the latest deposits that have been added to his credit balance and his remaining credit at a specific time.

Desired outcome
The vendor's account balance, available credit and vendor statement transactions are returned.

Preconditions
None.

Postconditions
The server returns the account balance, the available credit and the last X number of vendor statement transactions. X is a server configured value.

Participants
The following apply:
a) the vending operator;
b) the XMLVend client; and
c) the XMLVend server.

Happy path
The vendor statement happy path is illustrated in figure 21.

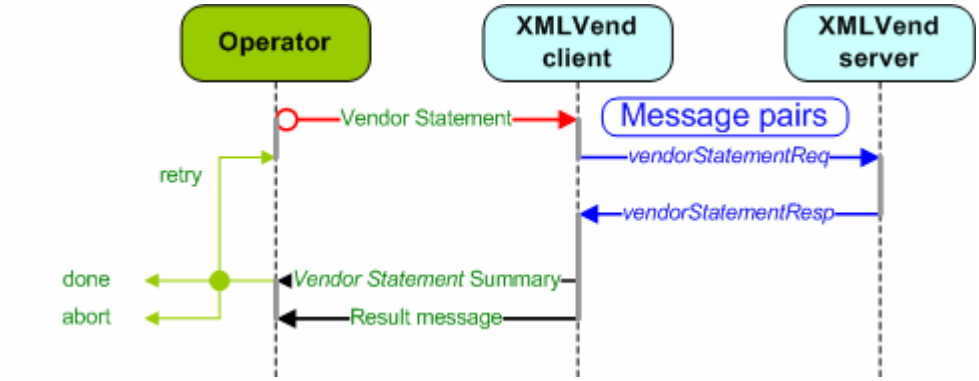

Figure 21 — Vendor statement happy path

Implementation
None.

4.5.2.17 Verify token

Description
The verify token use case allows a customer or vending operator to verify the information encoded on a token. The customer or vending operator will be required to provide the original numeric token or magnetic token.

Desired outcome
The token is verified.

Preconditions
The following apply:
a) the XMLVend server supports this use case; and
b) the operator is authorized to initiate this use case.

Postconditions
The token is verified and its associated data is returned to the client.

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The verify token happy path is illustrated in figure 22.

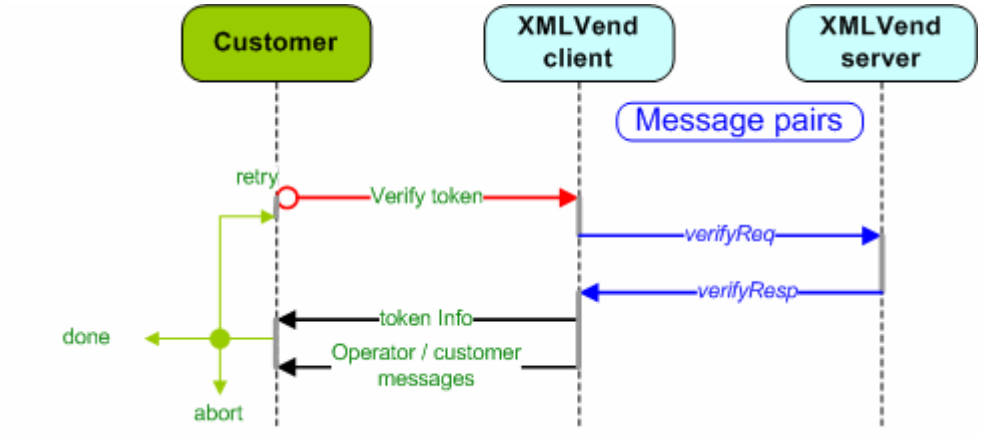

Figure 22 — Verify token happy path

Implementation
None.

4.5.3 Meter management use cases

4.5.3.1 Confirm meter details

The confirm meter details use case is used to confirm the meter details and may be used to encode a meter card.

Desired outcome
The meter details that match the search criteria entered are returned.

Preconditions
The following apply:
a) the XMLVend server supports this use case;
b) the vending operator is authorized to initiate this use case;
c) an identification parameter supported by the server for this use case shall be supplied, such as
   1) a meter card (track 2 data) (see NRS 009-4);
   2) a meter serial number; or
   3) meter configuration data, usually from an old token.
d) the server business rules are met.

Postconditions
The meter details that match the identification parameter entered are returned to the client.

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The confirm meter details happy path is illustrated in figure 23.

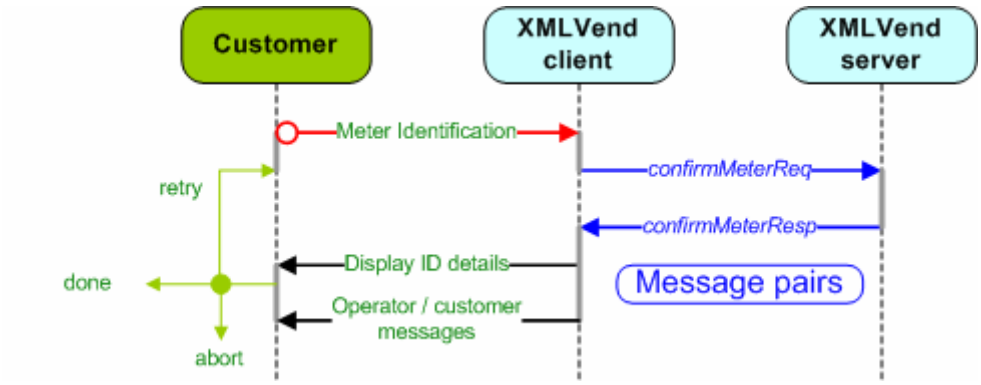

Figure 23 — Confirm meter details happy path

Implementation
None.

4.5.3.2 Customer report fault

Description
The customer report fault use case is used to report a fault to the utility that cannot be solved by the vending operator, such as a meter malfunction.

Depending on the fault being logged, meter configuration data, as well as customer contact information may be required.

Desired outcome
The customer logs a fault with the utility and receives a fault reference number.

Preconditions
The following apply:
a) the XMLVend server supports this use case; and
b) the operator is authorized to initiate this use case.

Postconditions
The following apply:
a) the server logs the customer fault; and
b) the customer obtains a reference number for the fault reported.

Participants
The following apply:
a) the customer;
b) the vending operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The customer report fault happy path is illustrated in figure 24.

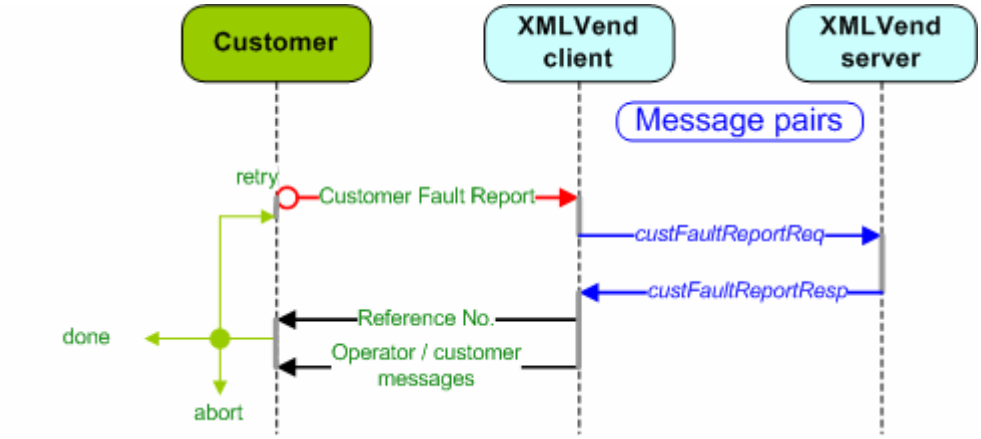

Figure 24 — Customer report fault happy path

Implementation
In the event that an exception occurs, it is highly recommended that this use case use the Issue Advice Use Case, to ensure that client and sever have the same understanding of the use case's outcome on both server and client. See 4.5.4.2.

4.5.3.3 Meter-specific engineering token

Description
This use case is used to request meter-specific engineering tokens, which are defined in NRS 009 6-7 and are usually requested by a technical operator. These tokens are used to update the meter configuration information.

The meter-specific engineering tokens in table 2 shall be supported.

Table 2 — Meter-specific engineering token

1	2	3
No.	**Token name**	**Token-specific parameters**
1	Set Power Limit	Power Limit [nnn.n kW]
2	Set Phase Unbalance	Power Limit [nnn.n kW]
3	Add Default Credit	None
4	Clear Credit	None
5	Clear Tamper	None
6	Engineering Key Change	FROM SGC [nnnnnn] KRN [n] TI [nn] TO SGC [nnnnnn] KRN [n] TI [nn]
7	Set Water Factor	Water Factor [nnnnnn]
NOTE 1 "n" =integer NOTE 2 See 3.2 for an explanation of abbreviations used in the table.		

The “Engineering Key Change” function provides more flexibility than the “Update Meter Key Use Case” (see 4.5.3.5) as it will allow the technical operator to provide the “TO” information. The “TO” information is only processed if the meter does not exist on the server. If the meter exists on the server, then the “TO” information will not be used and server information will be used.

A “set power limit” token may also be generated with a key change, if the power limit is tied to the specific tariff index. This will assist in maintaining the correct power limit for a specific tariff.

The “add default credit” is a new function. It will create a standard STS credit token but only for a set pre-configured amount (e.g. 5 kWh). Strict controls should, however, be employed for this function, as it can be abused.

Desired outcome
The requested meter-specific engineering token is returned.

Preconditions
The following apply:
a) the XMLVend server supports this use case;
b) the technical operator is authorized to initiate this use case;
c) an identification parameter supported by the server for this use case shall be supplied, such as
   1) a meter card (track 2 data) (see NRS 009-4),
   2) a meter serial number, or
   3) meter configuration data, usually from an old token.

Postconditions
a) the server security module (SM) generates the requested engineering token; and
b) an “engineering” transaction is recorded on the server.

Participants
The following apply:
a) the technical operator;
b) the XMLVend client; and
c) the XMLVend server.

Happy path
The meter-specific engineering token happy path is illustrated in figure 25.

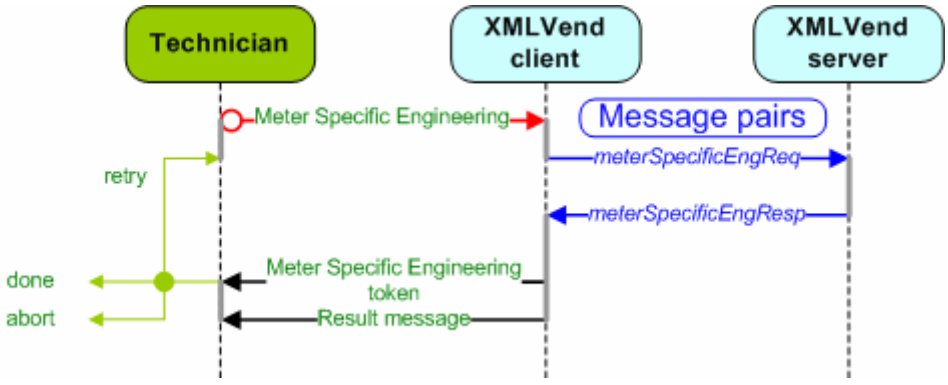

Figure 25 — Meter-specific engineering token happy path

Implementation
The server may enable or disable individual functions for certain groups of technical operators, for example a field technician may be allowed to perform engineering key changes and set maximum power but may not be allowed to clear credit or create credit tokens.

4.5.3.4 Non-meter-specific engineering token

This use case generates non-meter-specific engineering tokens and is usually requested by a technical operator. The non-meter-specific engineering tokens are risk-free tokens and may be entered into any meter. The non-meter-specific engineering tokens are illustrated in table 3.

Table 3 — Non-meter-specific engineering tokens

1	2
No.	**Token name**
0	Test All
1	Test Breaker
2	Test Display
3	Display Power Limit
4	Display Tariff Index (TI)
5	Display Key Revision (KRN)
6	Display Tamper
7	Display Instantaneous Power
8	Display Consumed Total
9	Display Phase Unbalance
10	Display Version

Desired outcome
The requested non-meter-specific engineering token is returned to the client for use in any STS meter.

Preconditions
The following apply:
a) the XMLVend server supports this use case;
b) the operator is authorized to initiate this use case; and
c) the server business rules are met.

Postconditions
The requested non-meter-specific engineering token is returned.

Participants
The following apply:
a) the technical operator;
b) the XMLVend client; and
c) the XMLVend server.

Happy path
The non-meter-specific engineering token happy path is illustrated in figure 26.

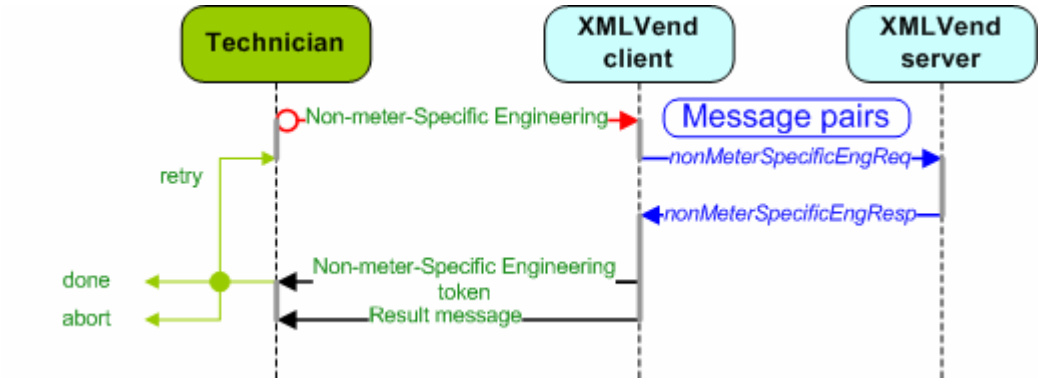

Figure 26 — Non-meter-specific engineering token happy path

Implementation
None.

4.5.3.5 Update meter key data

Update meter key data is a new function: the original key change function is now called "engineering key change" to differentiate it from "update meter key data".

The update meter key data use case is used to update one or more meter key data items for a specific meter, i.e. Supply Group Code (SGC), Key Revision Number (KRN) and tariff index (TI). It implements the standard key change process as defined by STS but the only difference being that the end state of the change is always the same as the data on the server. The "From" meter key data is obtained from a meter card or an old token. The "To" meter key data is not specified by the client as it is obtained from the server. This significantly reduces the risks associated with vendors producing key change tokens.

This use case is manually triggered from the client. It usually follows an unsuccessful purchase credit use case, with a fault response, reporting a mismatch between the server and supplied meter data items.

Desired outcome
Meter key data for a specific meter is updated to be the same as the server data.

Preconditions
The following apply:
a) the XMLVend server supports this use case;
b) the operator is authorized to initiate this use case;
c) the server business rules are met;
d) an identification parameter supported by the server for this use case shall be supplied containing the "from" meter data items, such as
   1) a meter card (track 2 data) (see NRS 009-4),
   2) meter configuration data, usually from an old token.

Postconditions
The following apply:
a) the server security module generates the required Key Change Token (KCT) set;
b) the transaction is recorded on the server;
c) the customer obtains the KCT set that will be used to update the meter key data to match the server meter key data; and
d) the customer's meter card data is also updated.

Participants
The following apply:
a) the customer;
b) the technical operator;
c) the XMLVend client; and
d) the XMLVend server.

Happy path
The update meter key data happy path is illustrated in figure 27.

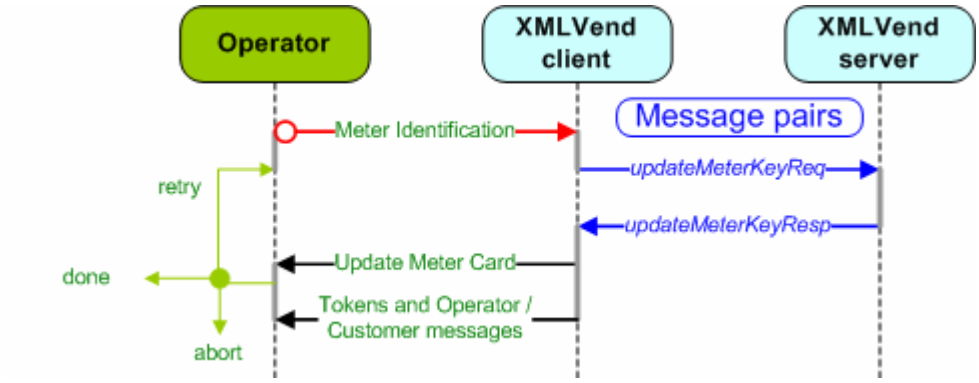

Figure 27 — Update meter key data happy path

Implementation
This use case is manually triggered from the client. It usually follows an unsuccessful Purchase Credit Use Case, with a fault response, reporting a mismatch between the server and supplied meter data items.

It can also be used when a credit token was issued from the server data, which may be different from the configuration of the meter. The update meter key use case will be used to correct the meter configuration so that the meter would accept the original credit token and also end with the correct configuration as defined by the server.

It is highly recommended that vendors who perform this function also have the capability to update the customer's meter card.

4.5.4 Base use cases

4.5.4.1 General

At the time of publication of this part of NRS 009, there was only one base use case. See 4.5.4.2.

4.5.4.2 Issue advice

Description
The Issue Advice Use Case provides three possible mechanisms to ensure that the client and server share the same understanding of the outcome of a use case. This use case is required when an exception occurs while a transaction is being processed, such as communication that fails between a client and a server after a request has been sent and the client is waiting for a response.

Three advice mechanisms have been defined as follows:

a) explicit use case confirm and reversed – a use case outcome is only finalized once it has been explicitly confirmed or reversed by the client;

NOTE Confirmation requests may include the payment method of the preceding use case.

b) implicit use case confirm and explicit reverse **–** a use case outcome is always assumed final, unless it is reversed by the client with a “Reversal Advice” message; and

c) advises last response **–** a use case outcome is always assumed final and cannot be reversed. Clients would have to determine if the use case has been processed by the server by issuing an “Advise Last Response” message.

Desired outcome
The client and server share the same understanding of the outcome of a use case.

Preconditions
The following apply:

a) explicit use case confirm and reverse;
   1) confirmation messages are issued for all response messages, and
   2) reversal messages are issued for all use cases that cannot be finalized on the client;

b) implicit use case confirm and explicit reverse; and
   1) reversal messages are issued for all use cases that cannot be finalized on the client;

c) advise last response.
   1) advice last response messages are issued when the client is uncertain if the server has processed its request message.

Postconditions
The following postconditions can exist depending on the use case scenario implemented

a) the use case is explicitly confirmed or reversed;
b) the use case is implicitly confirmed or explicitly reversed; and
c) the use case is implicitly confirmed or, if the client is in doubt, the last response of the server is requested. If the use case was successfully completed on the server, the response message will be resent. However, if the transaction was not successfully completed on the server, an XMLVend fault message will be returned and the client can safely repeat the transaction.

Participants
The following apply:
a) the XMLVend client; and
b) the XMLVend server.

Happy path
The use case scenarios are as follows:

a) **Explicit confirm and reverse:** The applicable use case is only considered successful or unsuccessful when the client explicitly sends an advice (confirm) or advice (reversal) message respectively. This scenario is referred to as Explicit Use Case Confirm and Reverse**.** In this scenario the server depends on the client.

The explicit confirm and reverse happy path is illustrated in figure 28.

Figure 28 — Explicit transaction confirm and reverse happy paths

b) **Implicit confirm and explicit reverse:** The server considers an applicable use case successful after it has sent the use case response message, unless indicated otherwise by a client with an advice (reversal) advice message. This scenario is referred to as Implicit Use Case Confirm and Explicit Reverse and is sometimes referred to as “negative confirmations”**.** In this scenario the server also depends on the client.

The implicit confirm and explicit reverse happy path is illustrated in figure 29.

NOTE The successful use case response from the server is not confirmed explicitly.

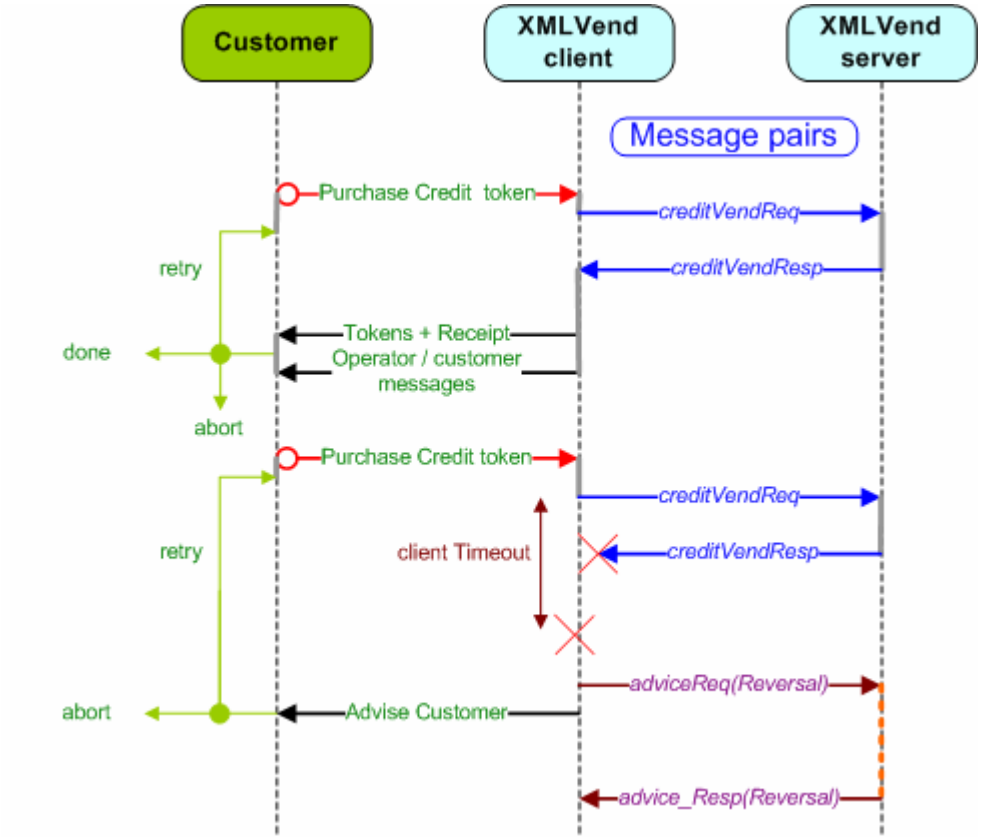

Figure 29 — Implicit confirm and explicit reverse use case happy path

c) **Advise last response:** The server always considers an applicable use case successful after it has sent the use case response message. However, if the client does not receive a response and therefore cannot be sure of the success of the use case on the server, it requests the server to resend the last response message using the advice (lastResponse) message.

If the use case was **successfully** completed on the server, the response message will be resent. However, if the transaction was **not successfully** completed on the server, an XMLVend fault message will be returned and the client can repeat the transaction.
In this scenario the server does not depend on the client. It is the client's responsibility to ensure responses are received for all requests (by verifying the message IDs). If a request has not been responded to for a use case, the customer has to request an Advise Last Response. If a response is received, the vendor can then issue it to the customer or alternatively keep it for a manual reconciliation process.

An example sequence diagram using the advise last response mechanism is illustrated below.

NOTE The successful use case response from the server is not confirmed explicitly as in (b).

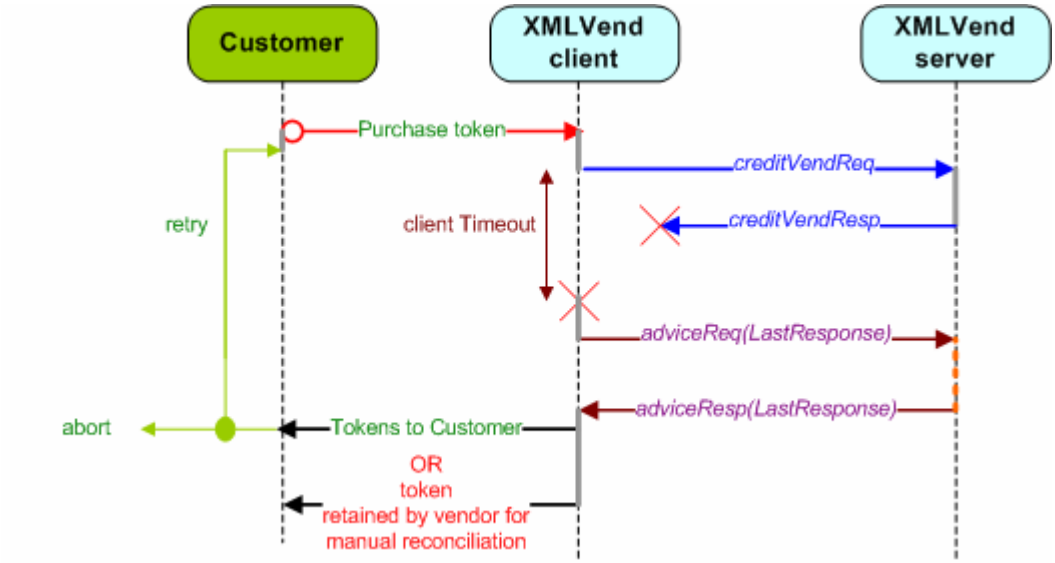

Figure 30 — Advise last response happy path

Implementation

Use case scenarios (a) and (b) can potentially be abused for the following reasons:

a) XMLVend clients can issue fraudulent advice (reversal) requests;

b) it is the responsibility of the client that once an advice (reversal) message is sent, the customer is not presented with any data in the form of a receipt or verbal communication; and

c) the server depends on decisions made by the client.

If use case scenarios (a) and (b) are implemented by utilities, the client implementations shall be strictly managed, with extensive testing and change control processes.

Use case scenario (c) offers the least risk, since the server does not depend on the client. The client is not allowed to issue advice (reversal) messages and, if they are issued, the messages shall be ignored by the server.

The advice message is system generated and not user initiated. The advice messages shall be delivered, which need not be in real-time. That is, an Issue Advice response message shall be received for the transaction to be completed. The use case can be implemented as follows:

1) locking the user interface until the exception scenario has been resolved; and
2) using a message queue to ensure guaranteed delivery of the advice messages.

Utilities will need to decide which “issue advice” scenario best suits their implementation. It is important that the decided approach is understood upfront by both client and server suppliers to ensure interoperability. Utilities may also decide to implement different “issue advice” scenarios for different vendor implementation models (see 4.1).

4.6 Fault and exception scenario handling

4.6.1 General

The XMLVend use cases covered in 4.5 specify happy path sequence diagrams for the request and response message pairs. The happy path is followed when the use case is successful, i.e. when the desired outcome is achieved. The happy path assumes the following to be true:

a) the client is able to send the request, the server able to receive the request, process it and send the response to the client. The client is able to receive the response message, and successfully process it; and

b) an application layer fault scenario does not exist while processing a request, such as a server configured business rule being violated.

This subclause describes how XMLVend addresses fault scenarios when the happy path cannot be followed.

4.6.2 Advice messages

4.6.2.1 As described in 4.5.4, the “Issue Advice” use case assists with communication-related fault scenarios. This scenario, when the client does not receive a response message, places it in an uncertain state since it is unsure if its request has been processed by the server or not. Two advice messages message pairs are provided to assist in resolving this fault scenario:

a) reversal advice; and

b) advise last response.

4.6.2.2 The “Reversal Advice” message reverses the transaction on the server. The “Reversal Advice” message has the following risks:

a) the client can issue fraudulent reversal requests; and

b) the server depends on decisions made by the client.

The “advise last response” message requests that the server resend its last response message. The resent message receipt is then handed to the customer or kept by the vending operator for manual reconciliation. If the request was not processed, then an XMLVend fault response message, “LastResponseEx”, is sent as described in 4.7.3.9.

The “advise last response” message offers the least risk, since the server does not depend on client decisions.

4.6.3 Fault response messages

Server application fault scenarios occur while processing a request message; this may prevent the server from continuing with the happy path of the use case. In this scenario the server shall respond with an appropriate fault response message to the client. The “XMLVend fault response message” is defined to communicate fault descriptions to the client.

Three main categories of fault response messages have been defined as follows:

a) XMLVendFaultEx – These fault scenarios are XMLVend protocol related;

b) SystemEx – These fault scenarios are server system failure related, such as, “Security Module Server not responding. Please try later.” As these fault scenarios are implementation specific, it is up to the user and server supplier to define these, as an extension of “SystemEx”; and

c) BusinessRuleEx – These fault scenarios are business-rule related, such as “Meter Blocked”. Some generic business rule faults have been defined in XMLVend, however, implementation of specific faults shall be defined by the utility and server supplier as an extension of “BusinessRuleEx”.

The fault response message also provides for operator- and customer-specific fault messages to be returned to the client. Utilities will define these messages, such that the correct fault description information is conveyed and some guide is given to operators on how to resolve the fault scenario. See annex A where examples of these messages are provided for the fault scenarios defined in XMLVend.

4.7 Use case class diagrams

4.7.1 General

Subclause 4.5 describes the use case happy path with the corresponding request and response messages. This subclause presents the class diagrams for the request and response messages that realise the use case. The class diagrams document the structure and content of each message pair based on the business requirements for each use case. The class diagrams are presented in a technology neutral and user accessible Unified Modelling Language (UML) notation[3)].

Each class diagram in this subclause has a shaded (red) class, which represents the root class of the diagram. Some diagrams may omit some related classes; this has been done to simplify the diagram but does not indicate that the related classes are not applicable in the message realization. In most cases the diagrams can be read in top-down fashion, except where the complexity of the diagram prevents this layout. All class and field names use the Camel naming convention. Class names are also prefixed with their domain name.

4.7.2 Interpreting optional fields

The class diagrams specify both mandatory (“[1]”) and optional (“[0]”) fields. Mandatory request fields represent the minimum fields required to process the request message on the server. Mandatory response fields represent the minimum fields required by the client to process the response message.

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

Specific implementations may, however, redefine optional request fields as mandatory, when the data contained in an optional field is required to successfully process the request on the server. Servers should reject all request messages that do not contain the required optional fields with an appropriate fault response message. Utilities can review their current CDU user interfaces to identify optional fields that need to be redefined as mandatory.

Optional response fields can also be redefined as mandatory when the specific implementation requires optional response fields to be processed by the client. Clients will be required to process the optional fields if present in the response message. Utilities can review their current prepayment receipts to identify optional fields that need to be redefined as mandatory. The redefinition of optional fields as mandatory will be specified by each utility in its implementation-specific use case descriptions document, and communicated to server and client suppliers.

4.7.3 Request and Response Message Class Diagrams

4.7.3.1 Cancel token

The cancel token request message class diagram is illustrated in figure 31.

3) The class diagrams are defined using standard UML notation. A quick reference to the UML notation used in this subclause is provided in annex B.

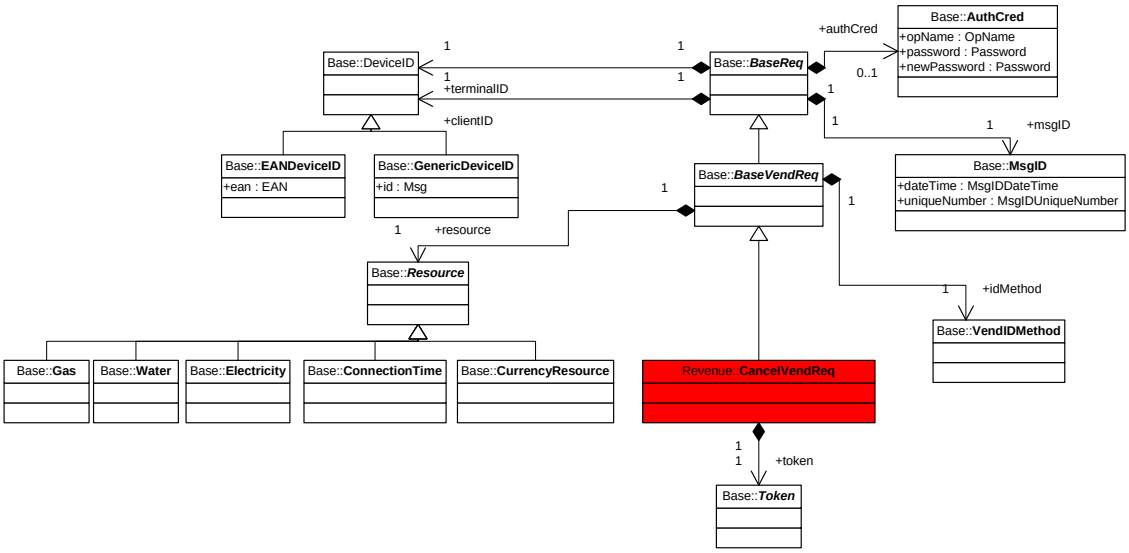

Figure 31 — Cancel token request message class diagram

The cancel token response message class diagram is illustrated in figure 32.

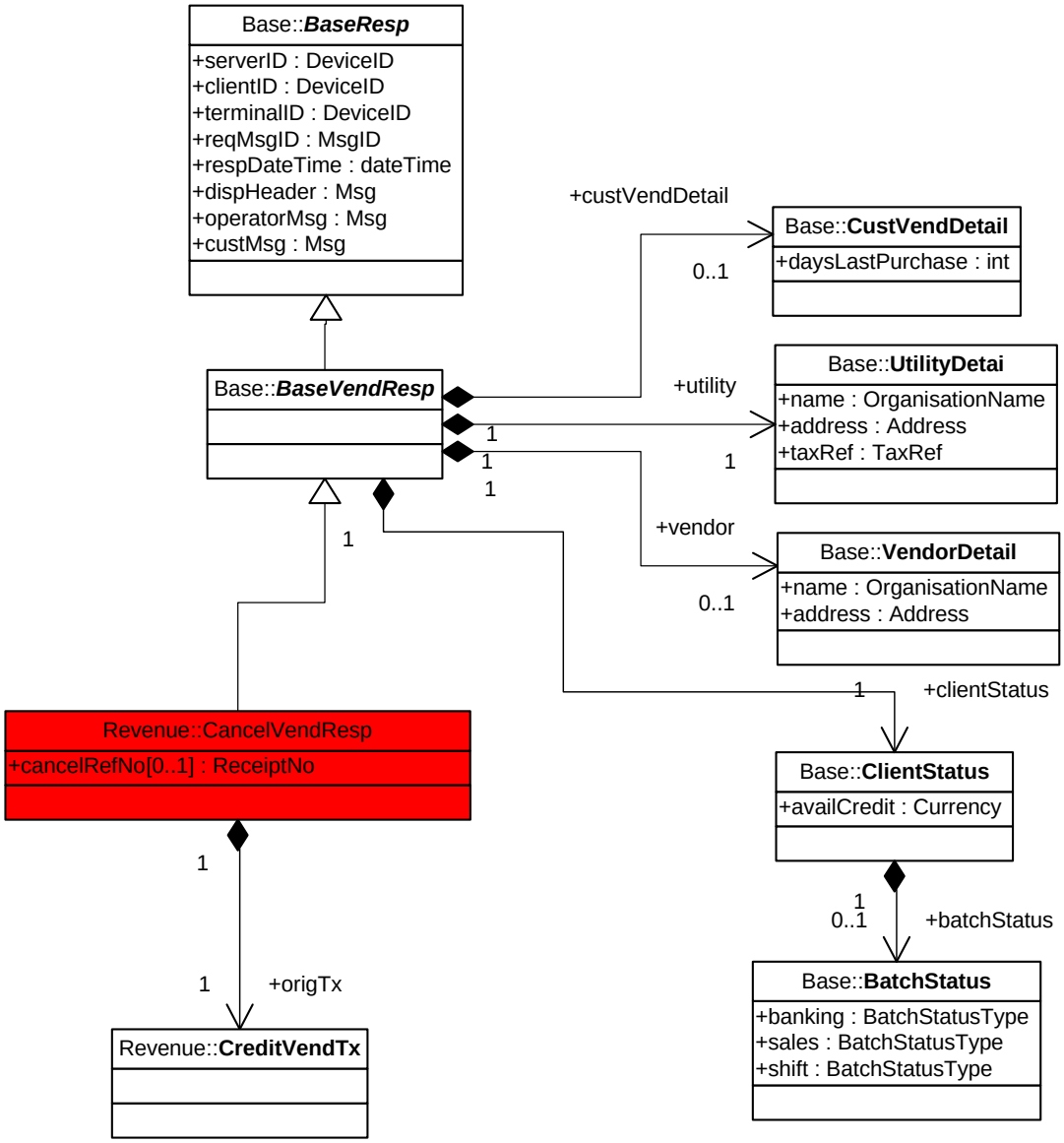

Figure 32 — Cancel token response message class diagram

4.7.3.2 Check batch totals

The check batch totals request message class diagram is illustrated in figure 33.

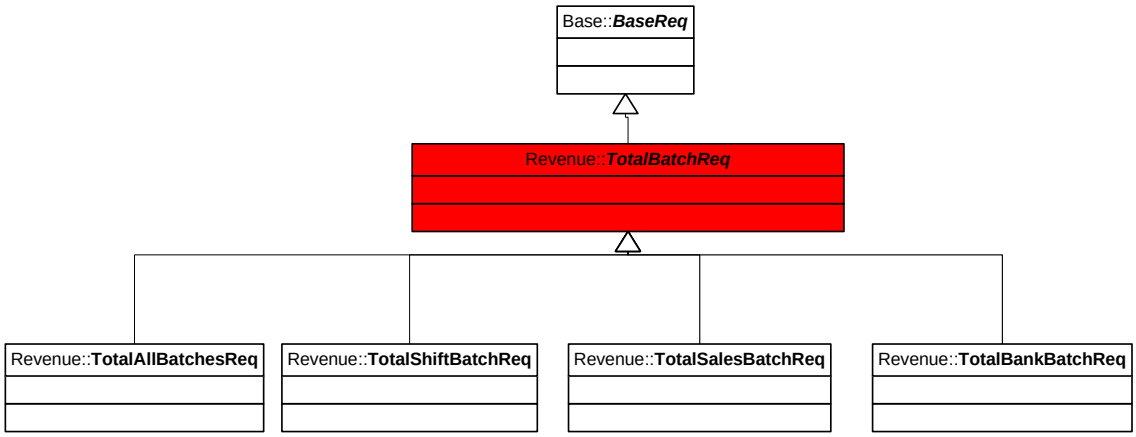

Figure 33 — Check batch totals request message class diagram

The check batch totals response message class diagram is illustrated in figure 34.

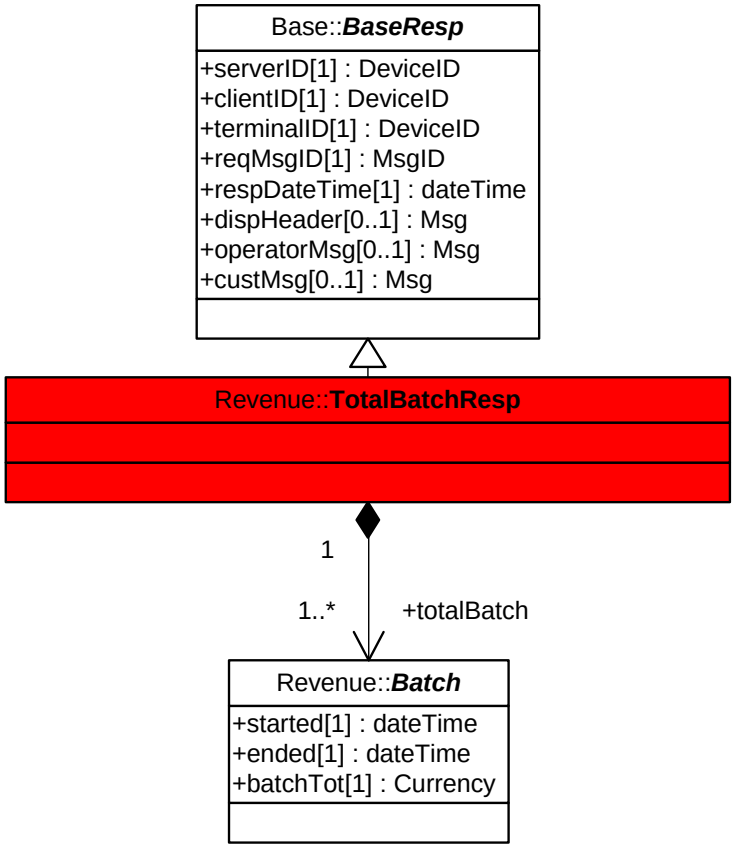

Figure 34 — Check batch totals response message class diagram

4.7.3.3 Confirm customer details check batch totals

The confirm customer details request message class diagram is illustrated in figure 35.

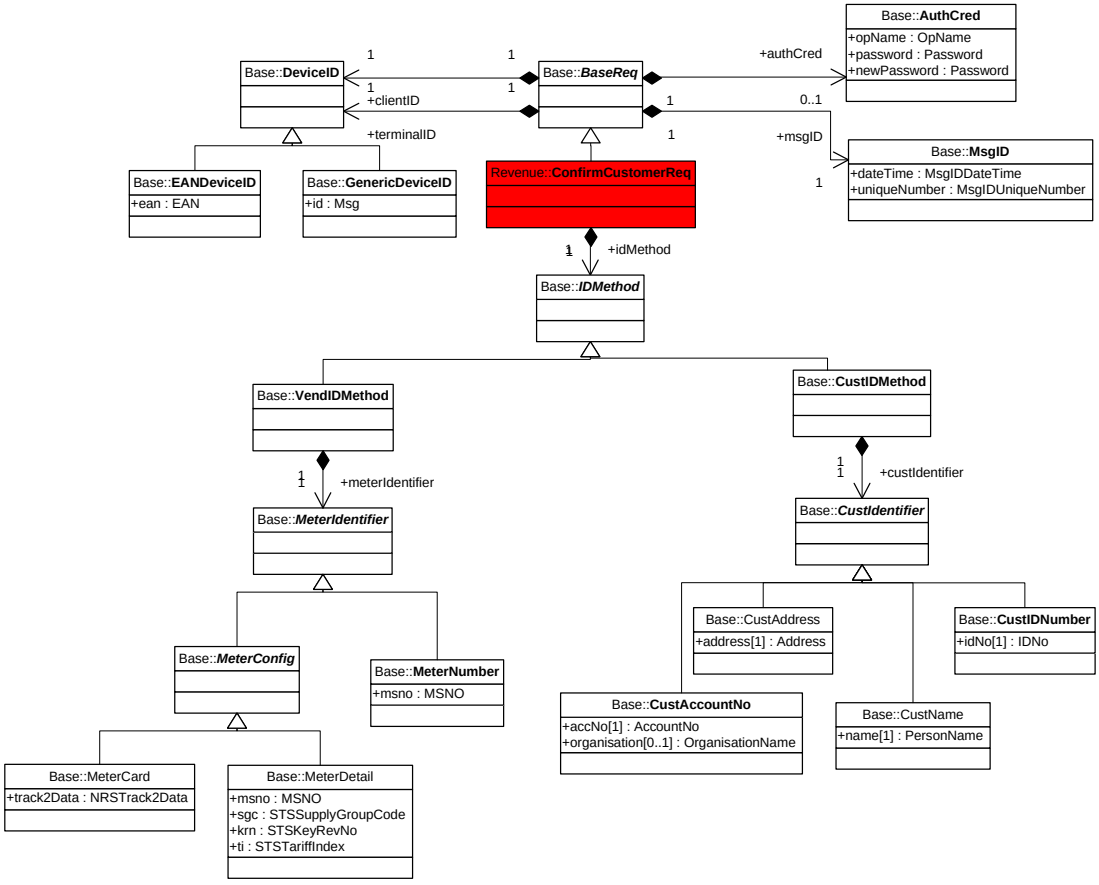

Figure 35 — Confirm customer details request message class diagram

The confirmed customer details response message class diagram is illustrated in figure 36.

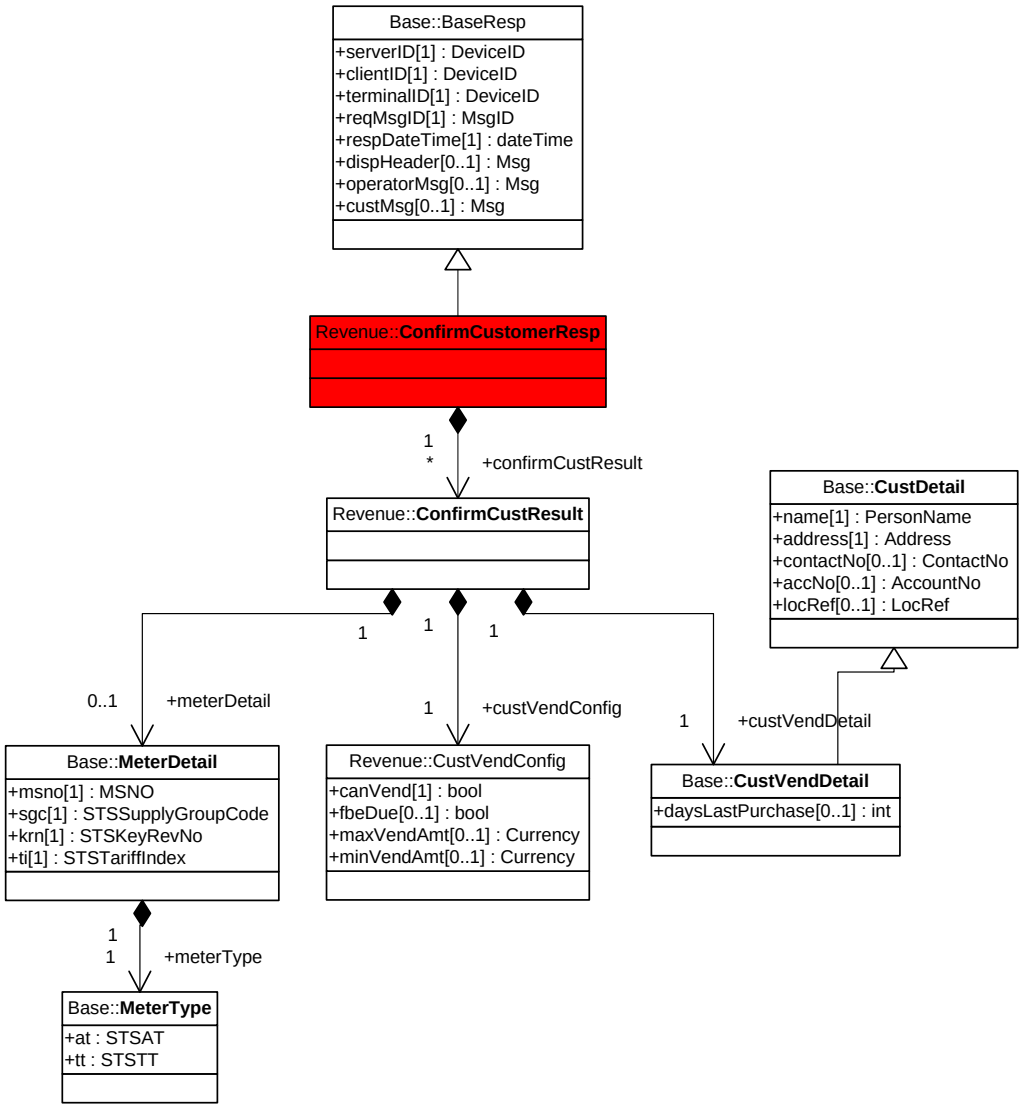

Figure 36 — Confirm customer details response message class diagram

4.7.3.4 Confirm meter details

The confirm meter details request message class diagram is illustrated in figure 37.

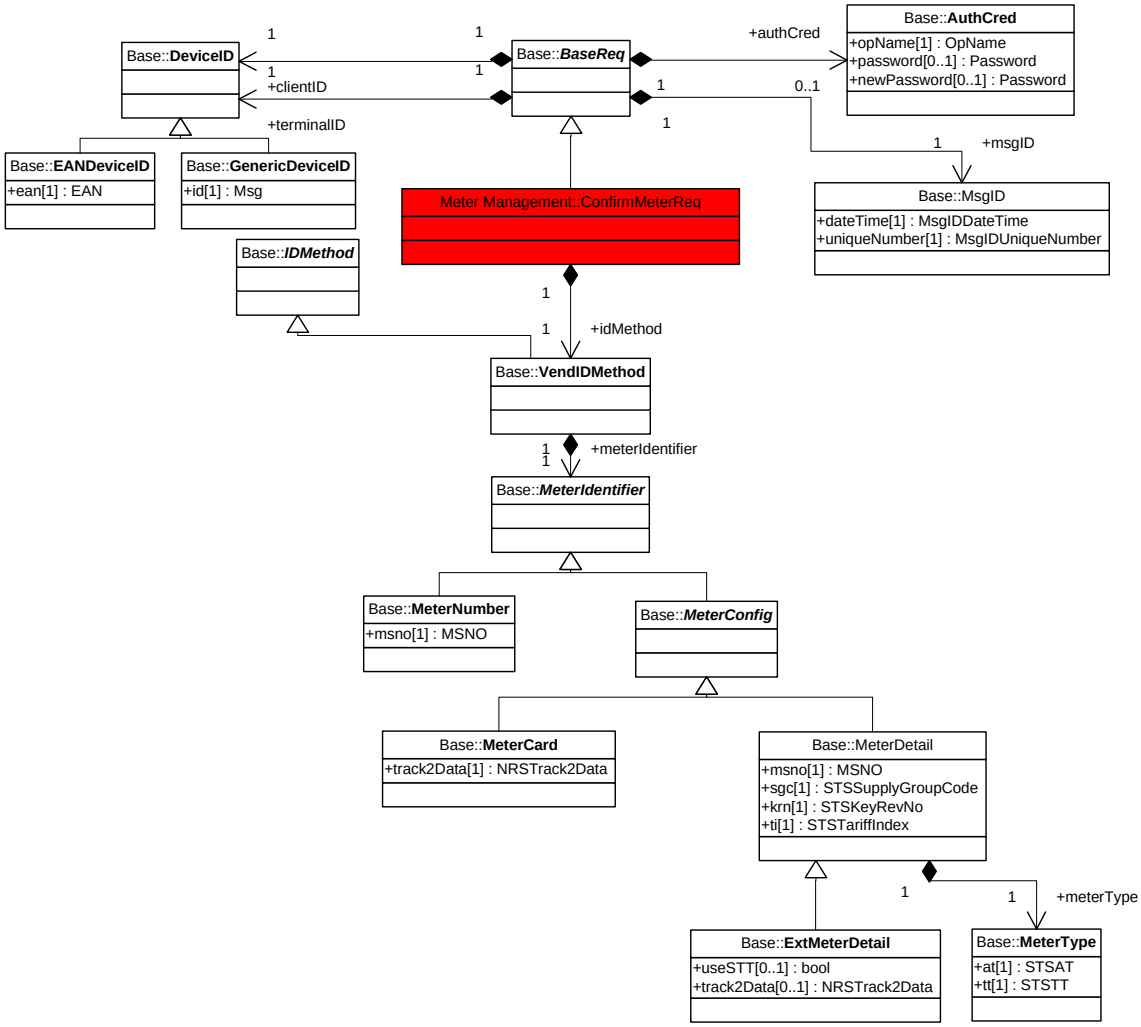

Figure 37 — Confirm meter details request message class diagram

The confirm meter details response message class diagram is illustrated in figure 38.

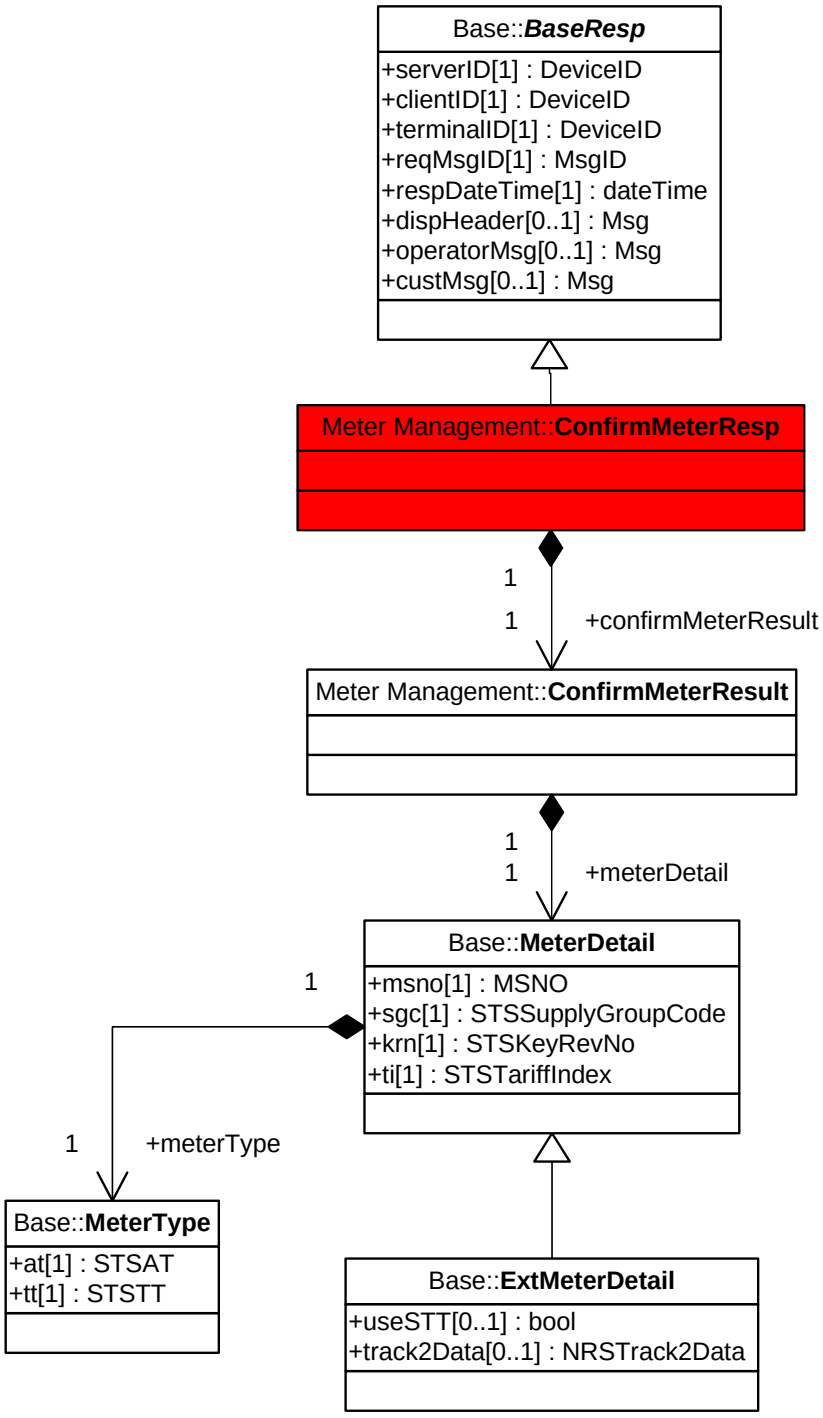

Figure 38 — Confirm meter details response message class diagram

4.7.3.5 Collect FBE token

The collect FBE token request message class diagram is illustrated in figure 39.

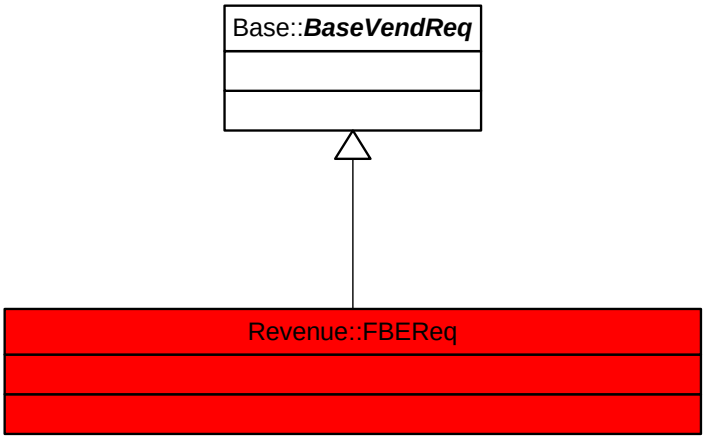

Figure 39 — Collect FBE token request message class diagram

The collect FBE token response message class diagram is illustrated in figure 40.

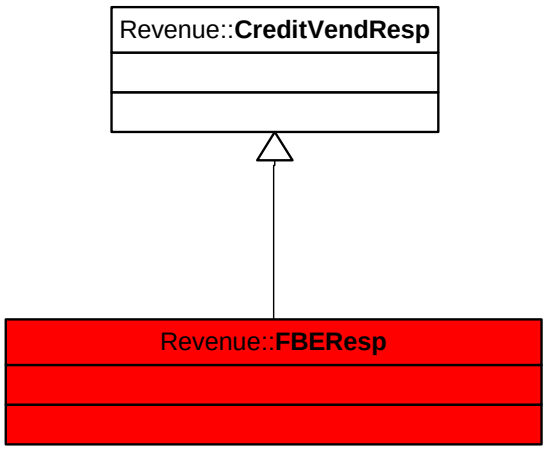

Figure 40 — Collect FBE token response message class diagram

4.7.3.6 Create deposit slip

The create deposit slip request message class diagram is illustrated in figure 41.

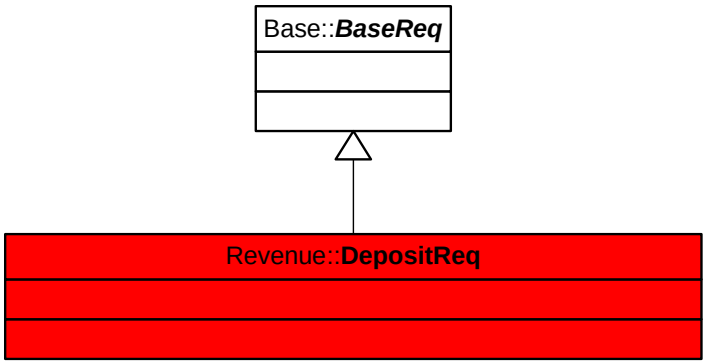

Figure 41 — Create deposit slip request message class diagram

The create deposit slip response message class diagram is illustrated in figure 42

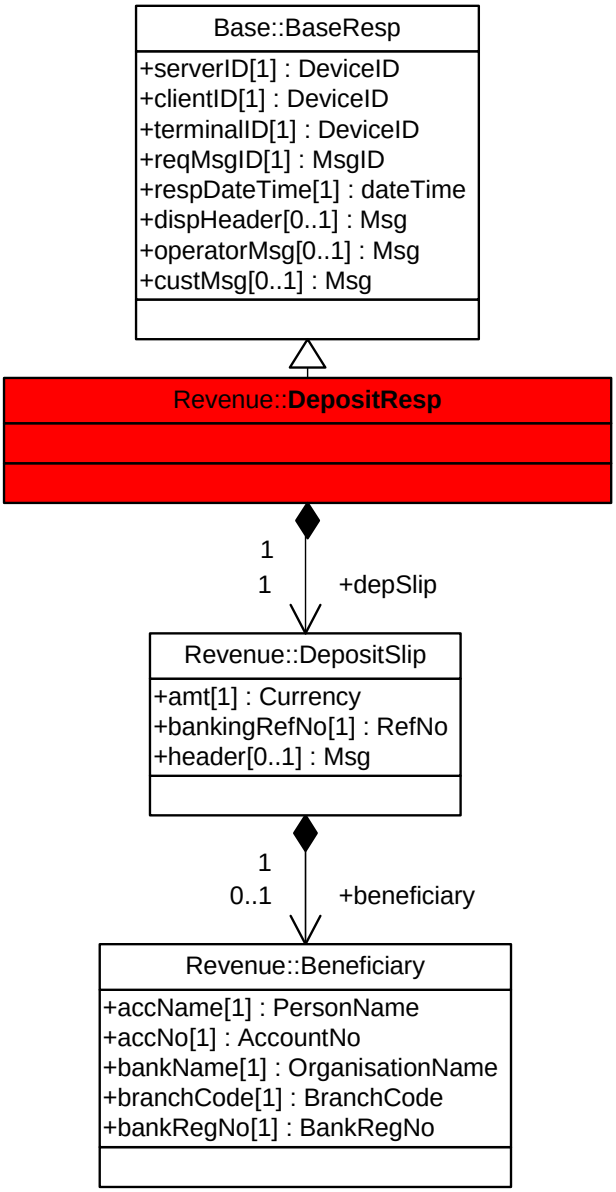

Figure 42 — Create deposit slip response message class diagram

4.7.3.7 Customer report fault

The customer report fault request message class diagram is illustrated in figure 43.

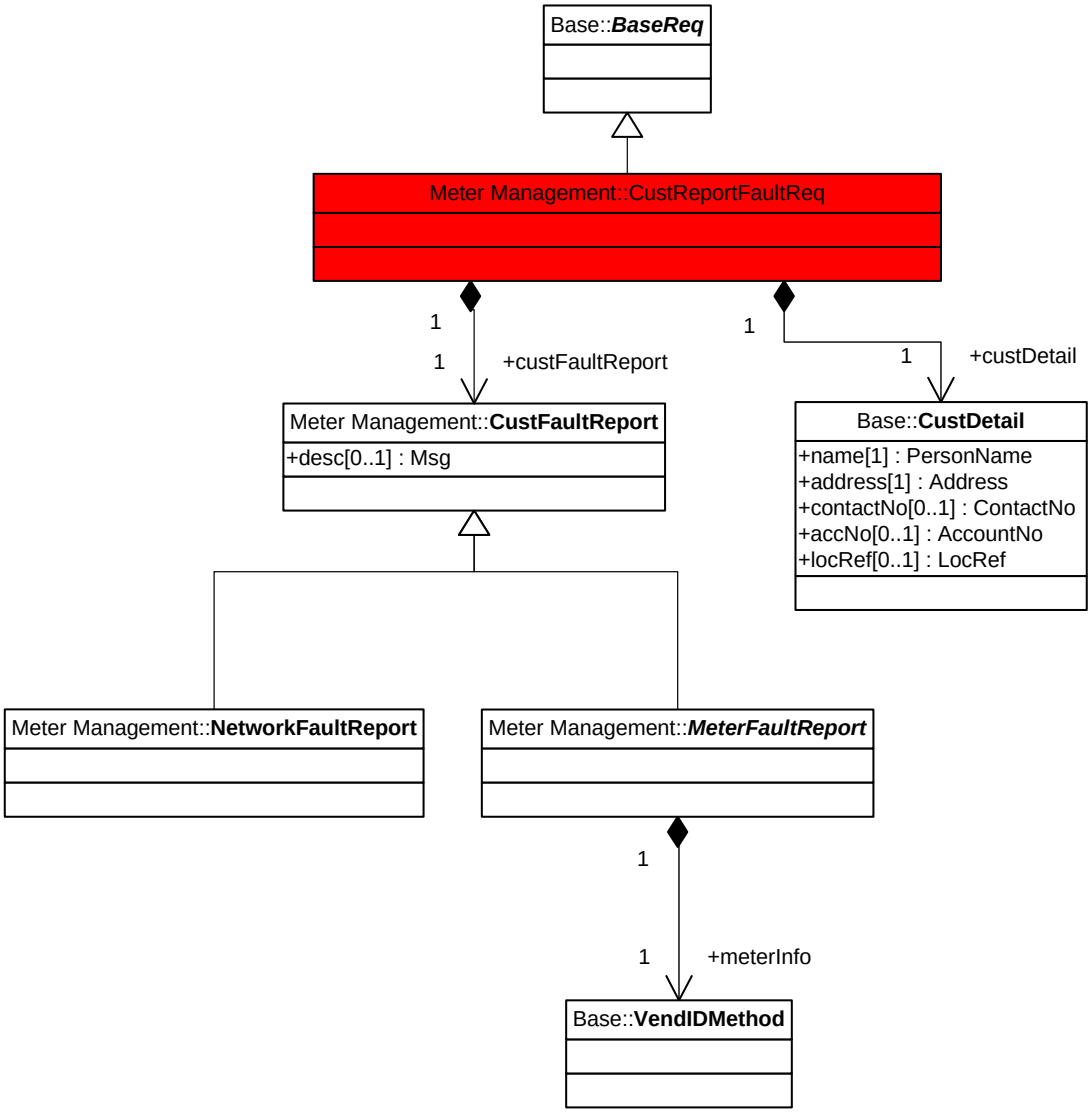

Figure 43 — Customer report fault request message class diagram

The customer report fault response message class diagram is illustrated in figure 44.

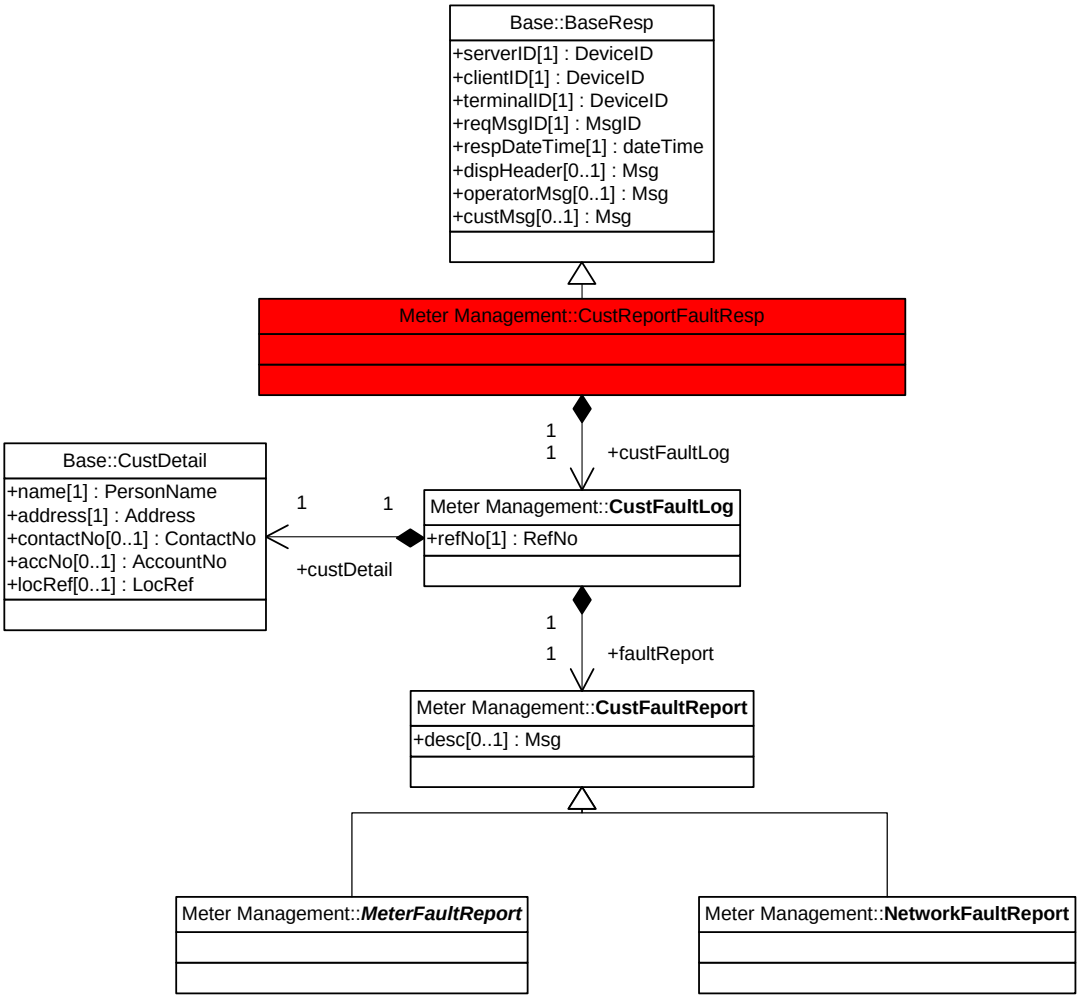

Figure 44 — Customer report fault response message class diagram

4.7.3.8 End batch

The end batch fault request message class diagram is illustrated in figure 45.

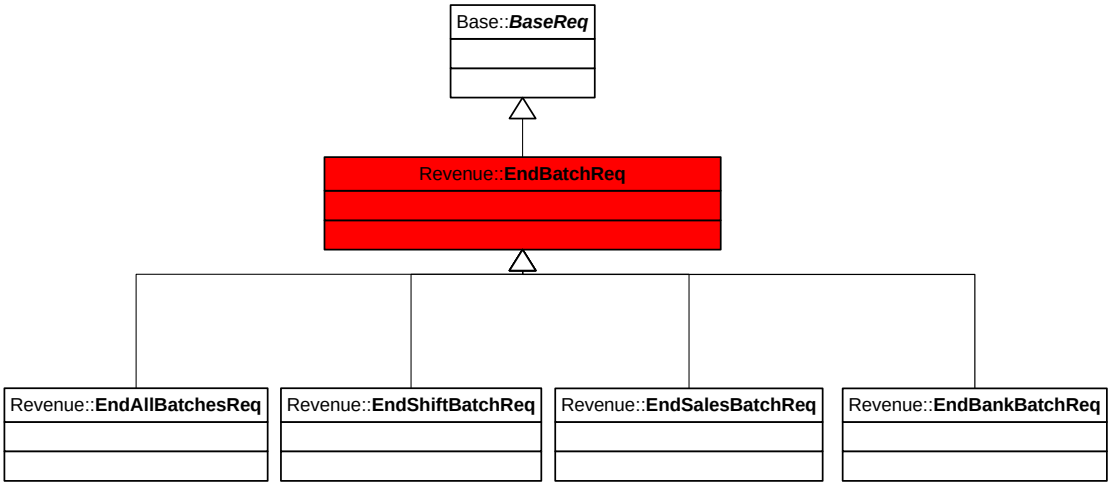

Figure 45 — End batch fault request message class diagram

The end batch fault response message class diagram is illustrated in figure 46.

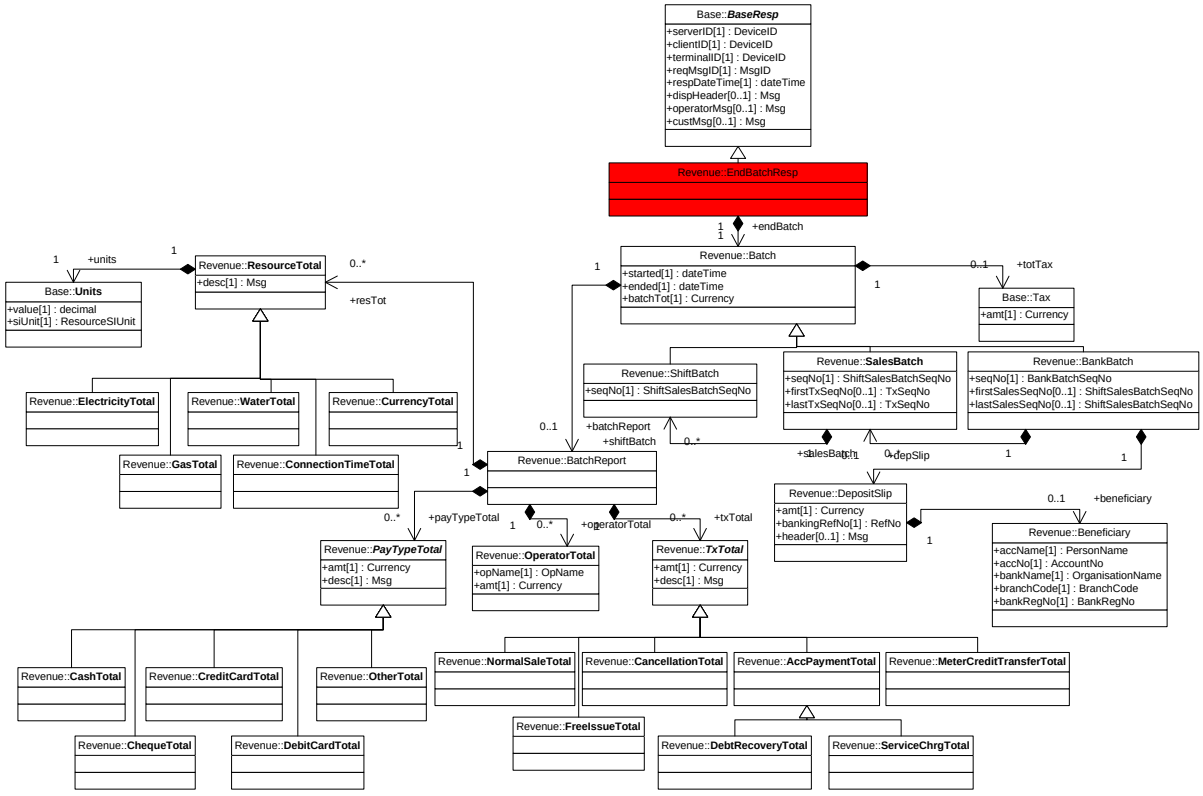

Figure 46 — End batch fault response message class diagram

4.7.3.9 Fault response message

The fault response message class diagram is illustrated in figure 47.

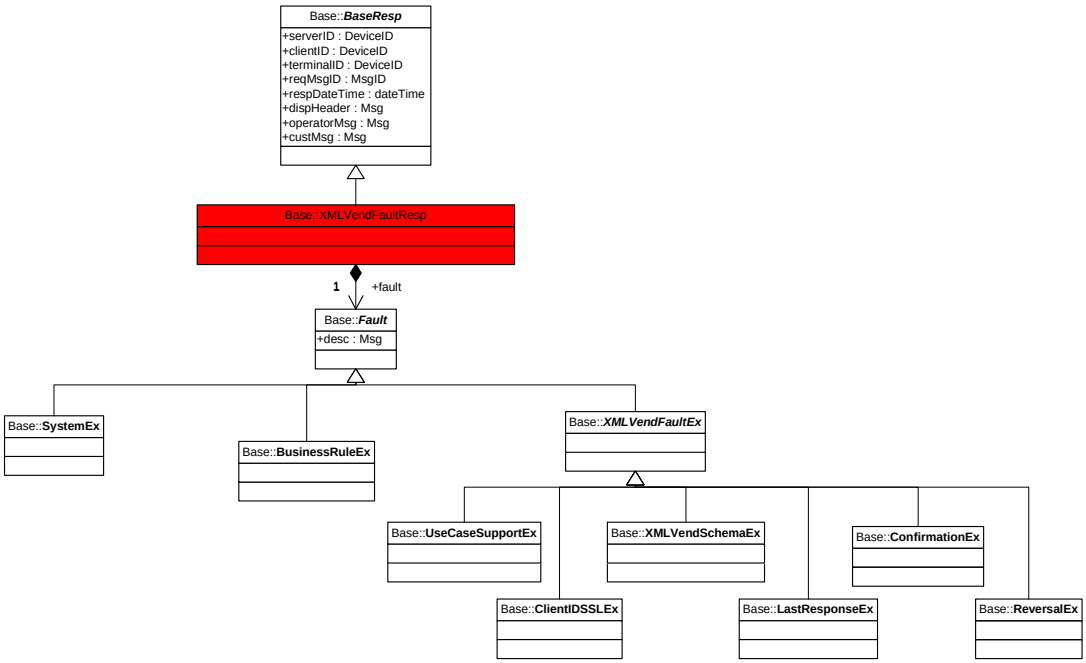

Figure 47 — Fault response message class diagram

The specializations of the BusinessRuleEx class are illustrated in figure 48.

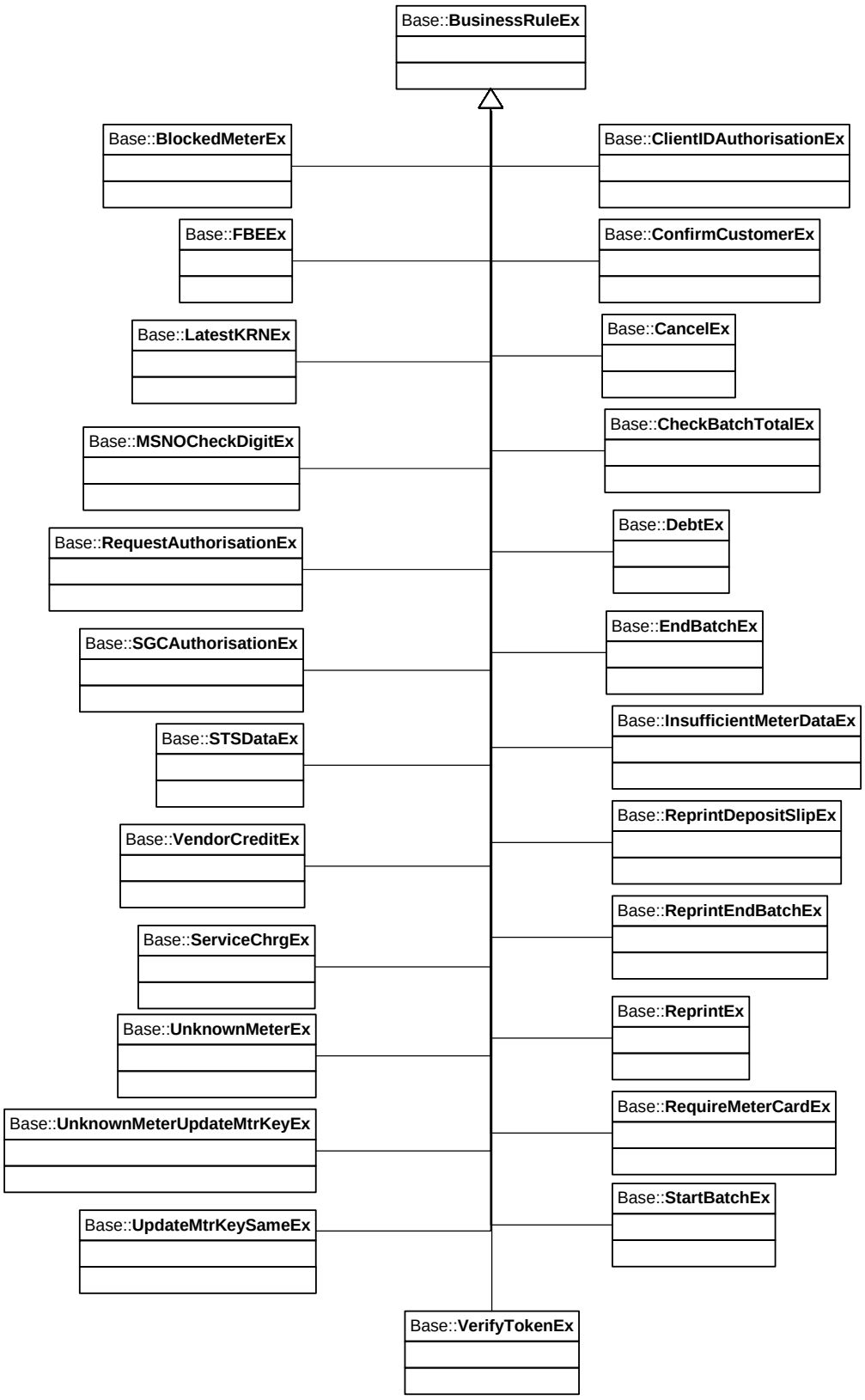

Figure 48 — Fault response message – BusinessRuleEx class diagram

4.7.3.10 Free issue token

The free issue token request message class diagram is illustrated in figure 49.

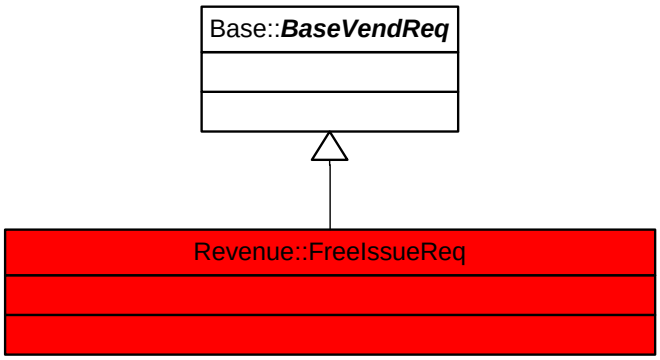

Figure 49 — Free issue token request message class diagram

The free issue token response message class diagram is illustrated in figure 50.

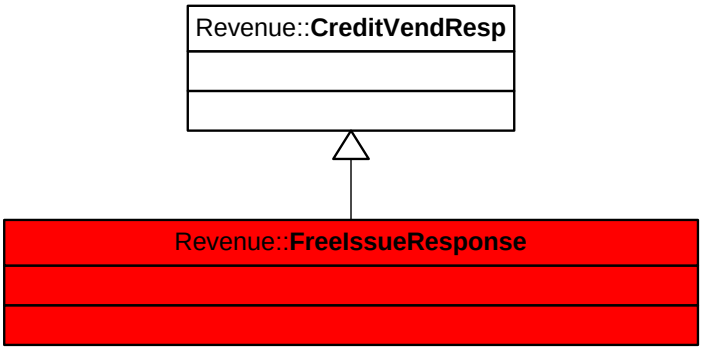

Figure 50 — Free issue token response message class diagram

4.7.3.11 Issue advice

The issue advice request message class diagram is illustrated in figure 51.

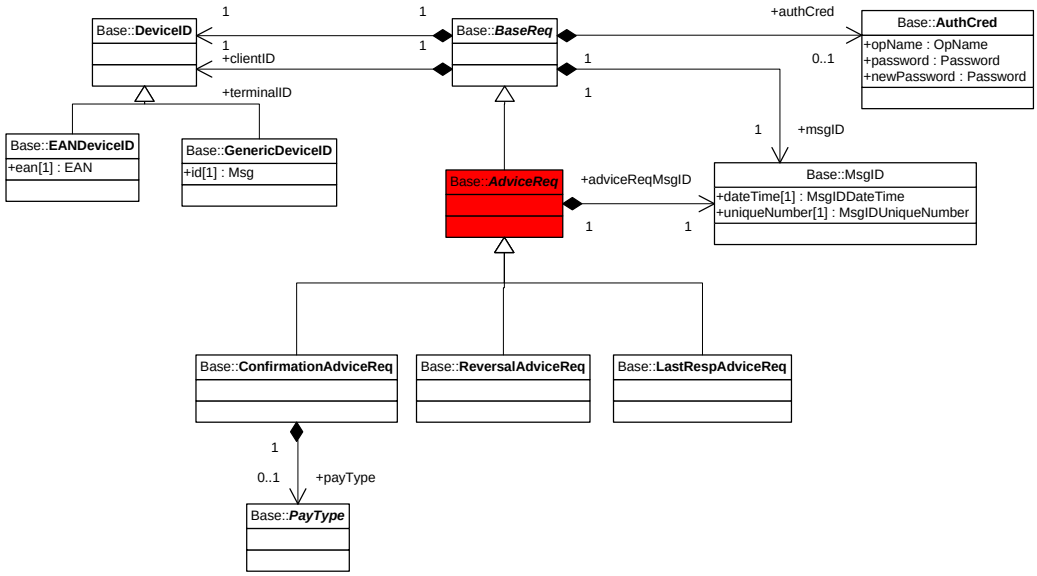

Figure 51— Issue advice request message class diagram

The issue advice response message class diagram is illustrated in figure 52.

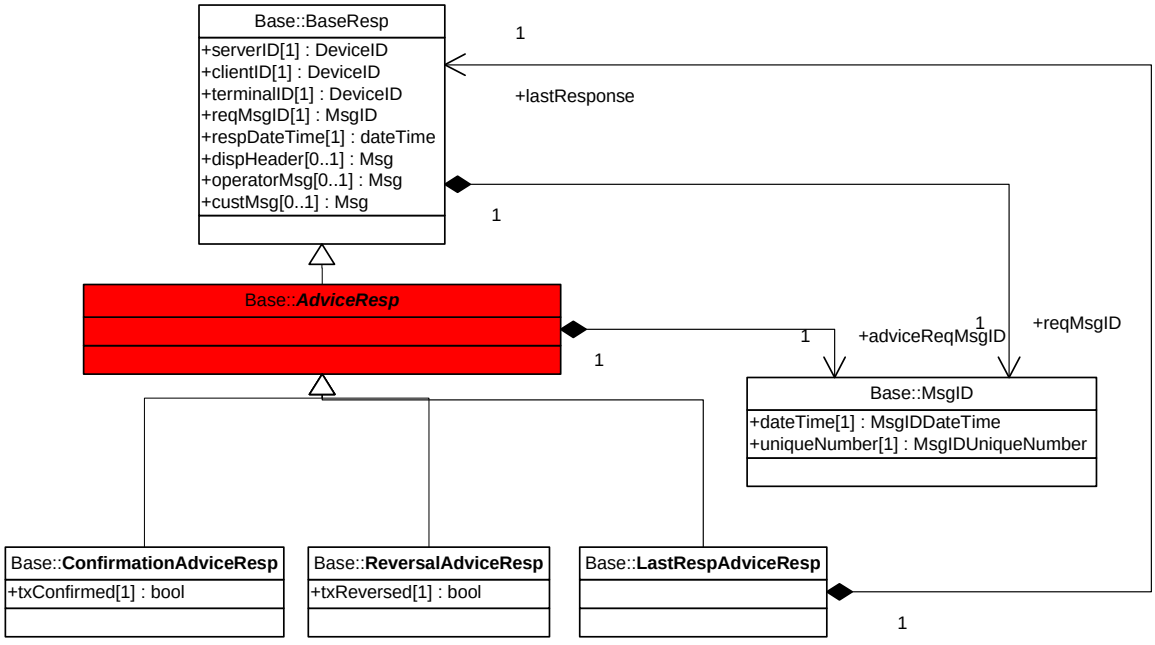

Figure 52 — Issue advice response message class diagram

4.7.3.12 Meter credit transfer token

The meter credit transfer request message class diagram is illustrated in figure 53.

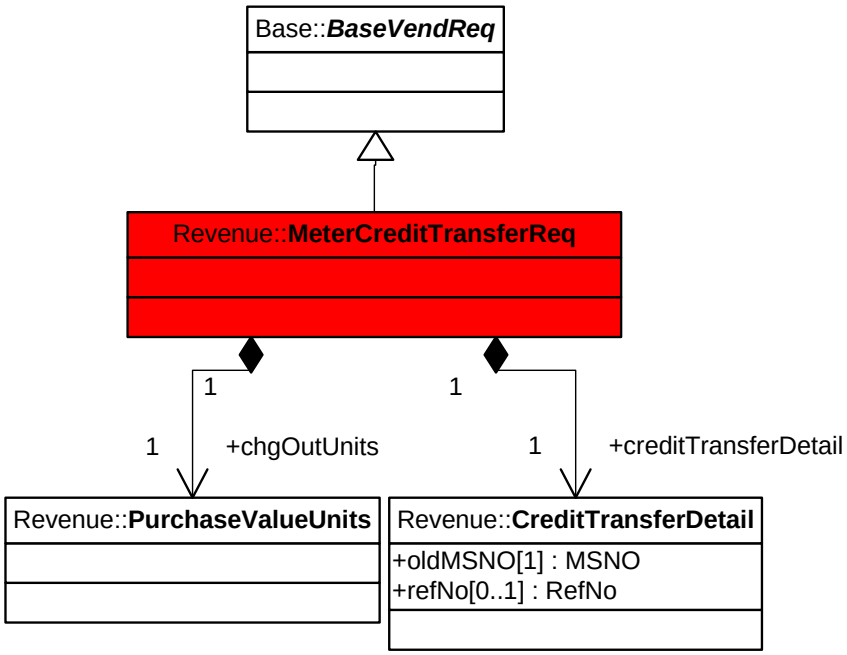

Figure 53 — Meter credit transfer request message class diagram

The meter credit transfer response message class diagram is illustrated in figure 54.

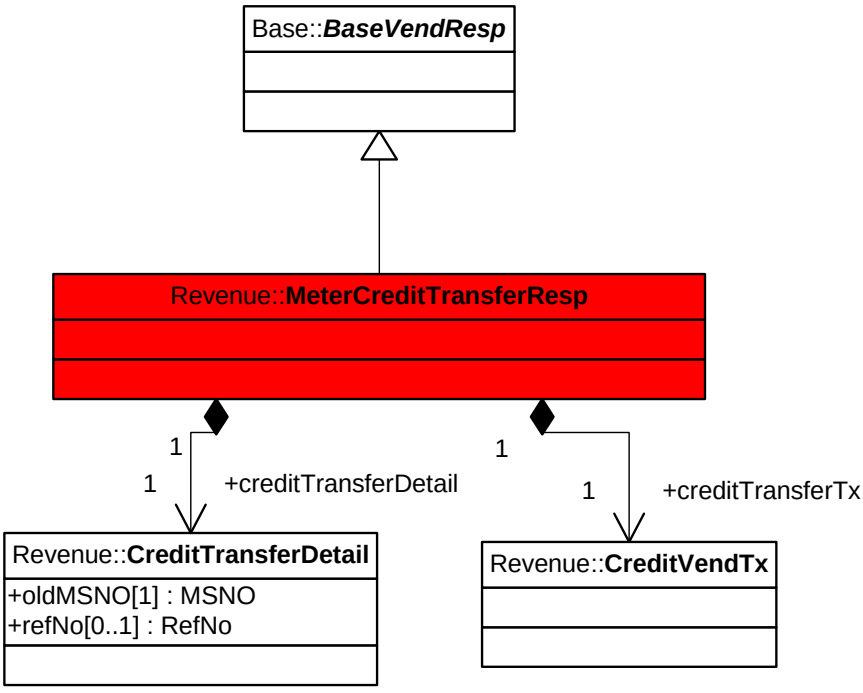

Figure 54 — Meter credit transfer response message class diagram

4.7.3.13 Meter-specific engineering token use case

The meter specific engineering token request message class diagram is illustrated in figure 55.

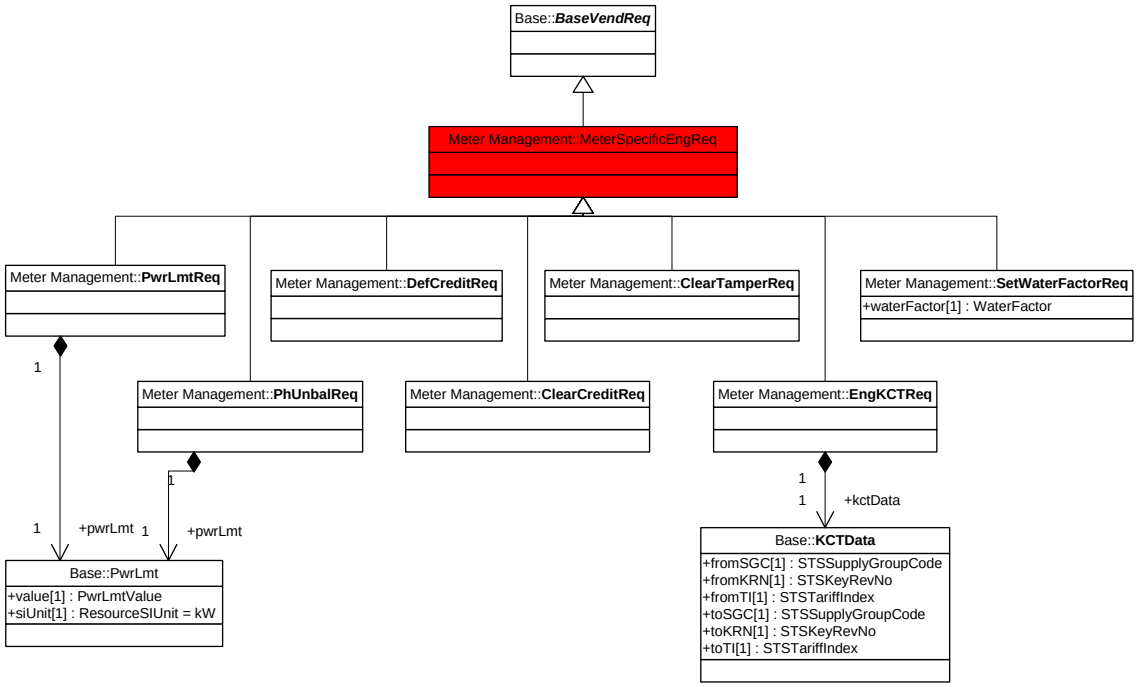

Figure 55 — Meter-specific engineering token request message class diagram

The meter-specific engineering token response message class diagram is illustrated in figure 56.

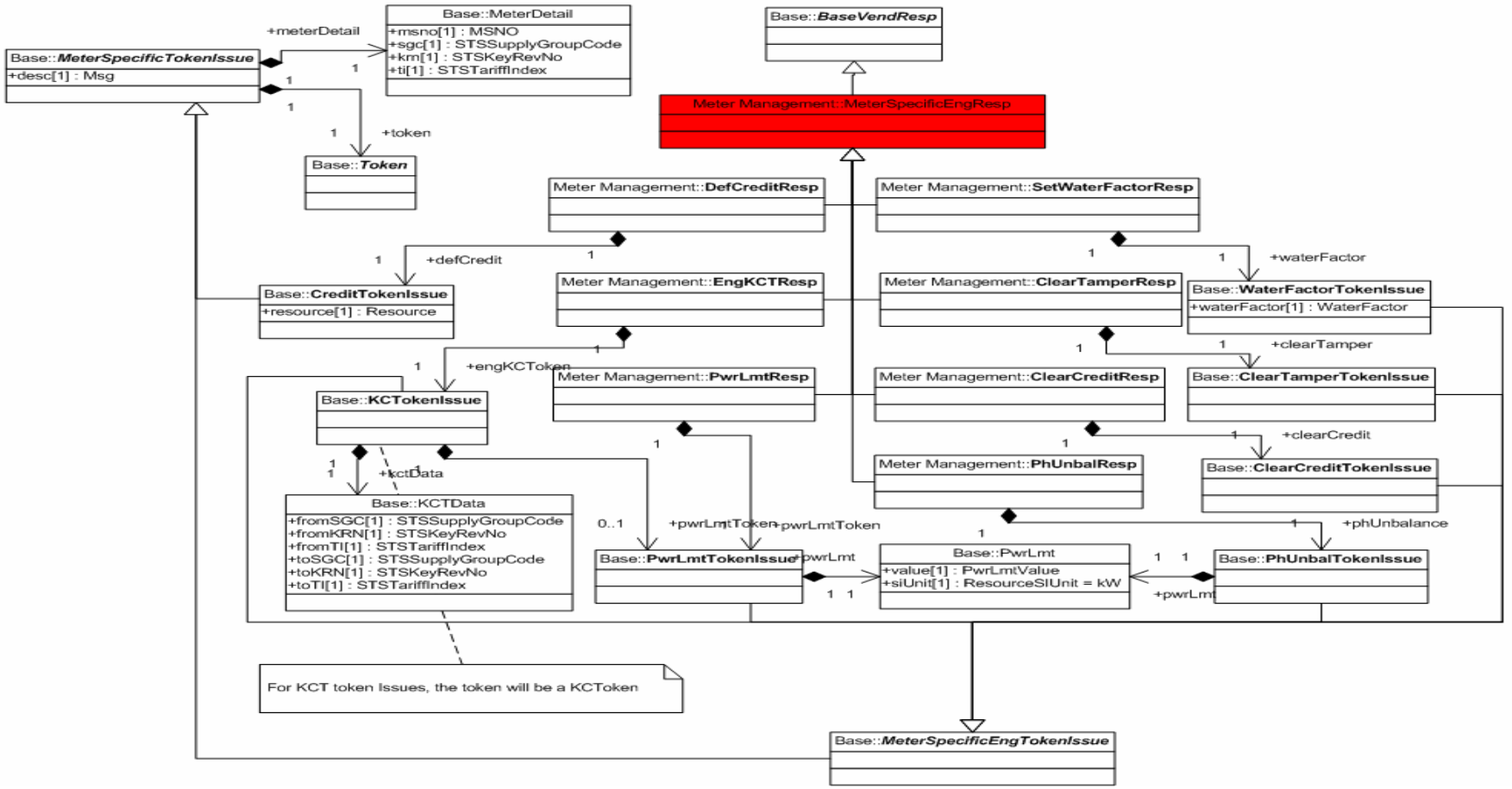

Figure 56 — Meter-specific engineering token response message class diagram

4.7.3.14 Non-meter-specific engineering token

The non-meter-specific engineering token request message class diagram is illustrated in figure 57.

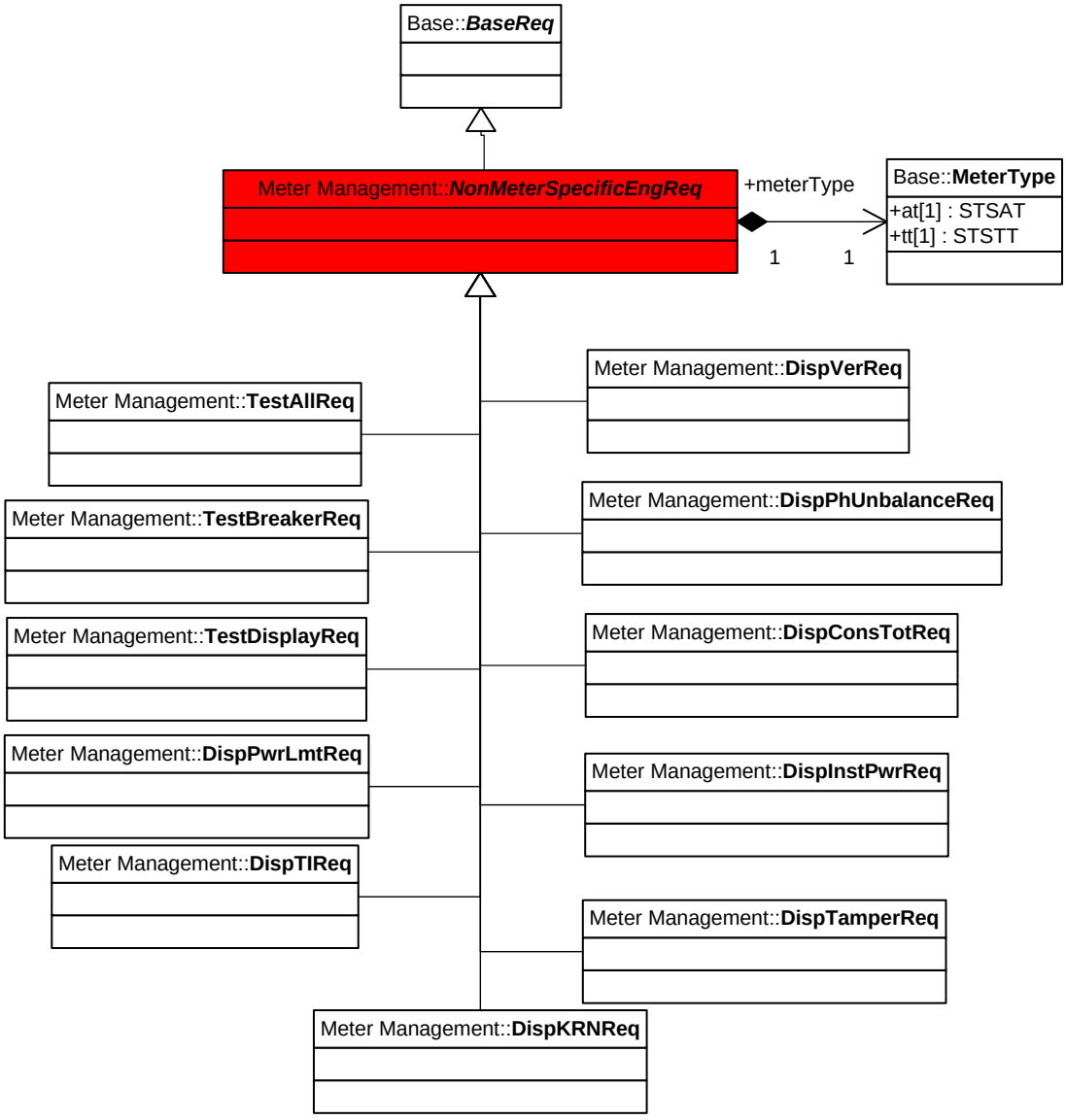

Figure 57 — Non-meter-specific engineering token request message class diagram

The non-meter-specific engineering token response message class diagram is illustrated in figure 58.

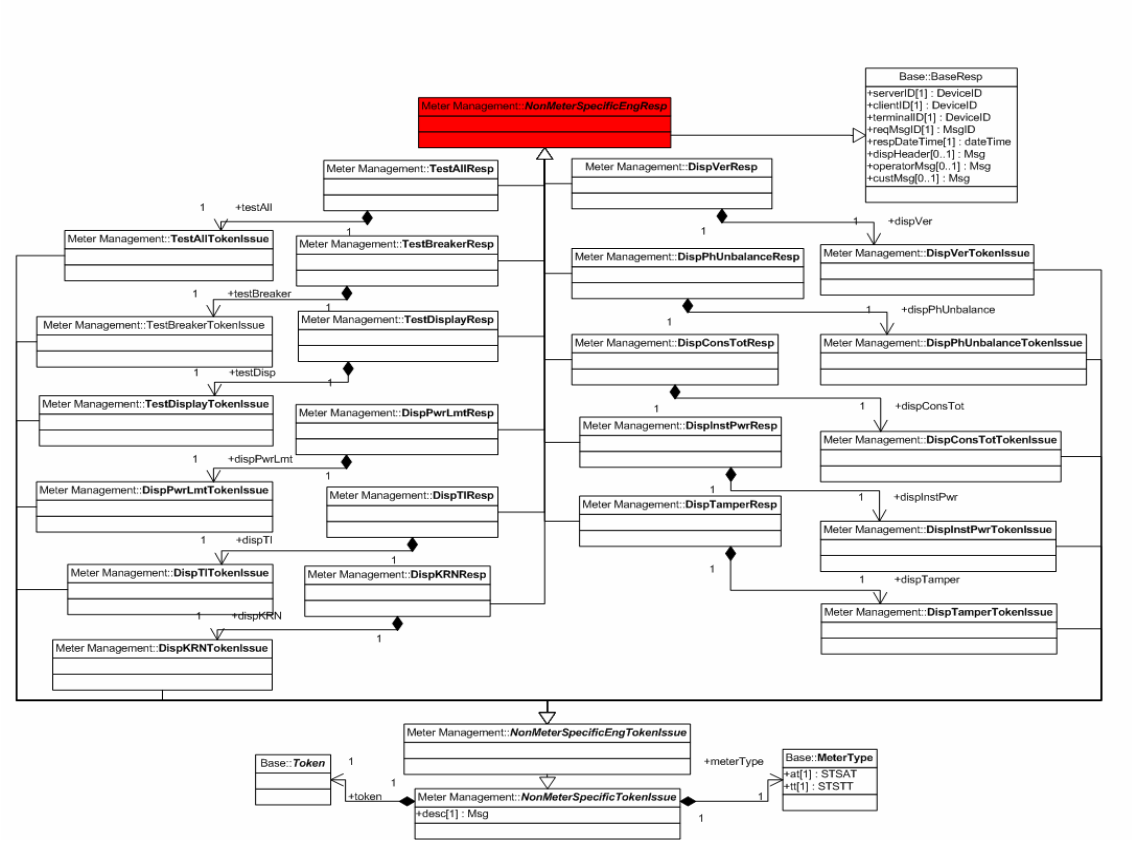

Figure 58 — Non-meter-specific engineering token response message class diagram

4.7.3.15 Pay account

The pay account request message class diagram is illustrated in figure 59.

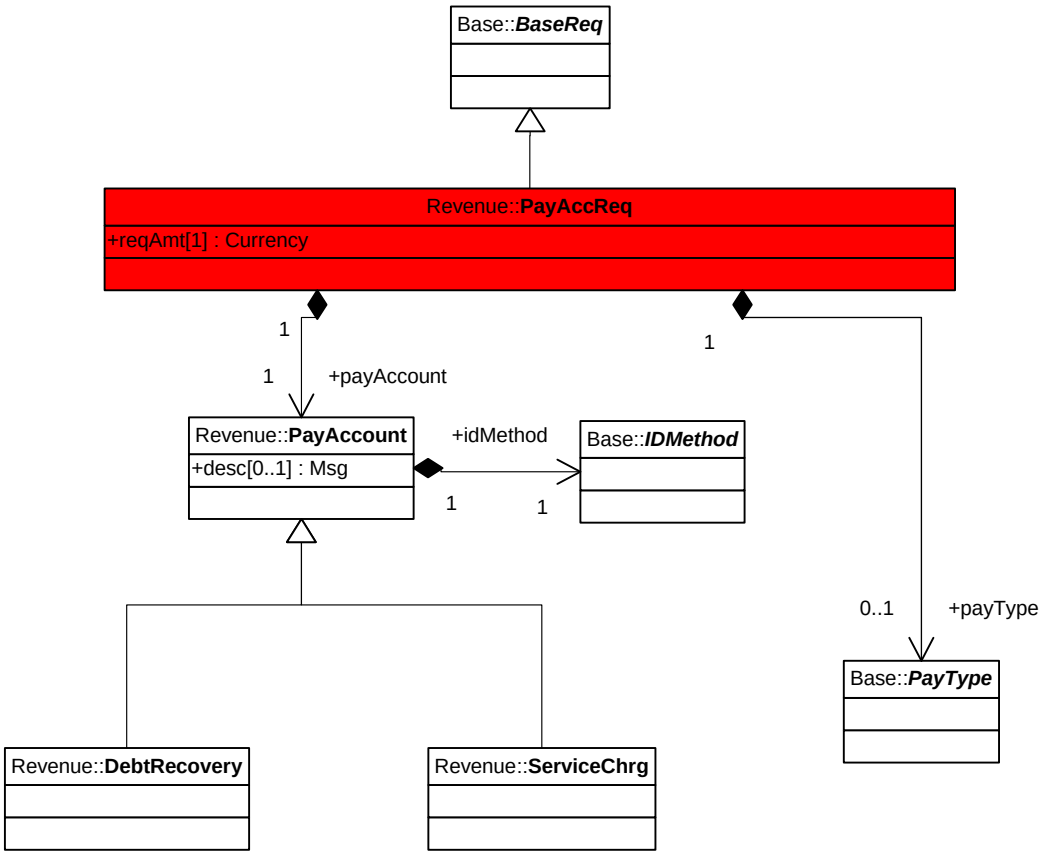

Figure 59 — Pay account request message class diagram

The pay account response message class diagram is illustrated in figure 60.

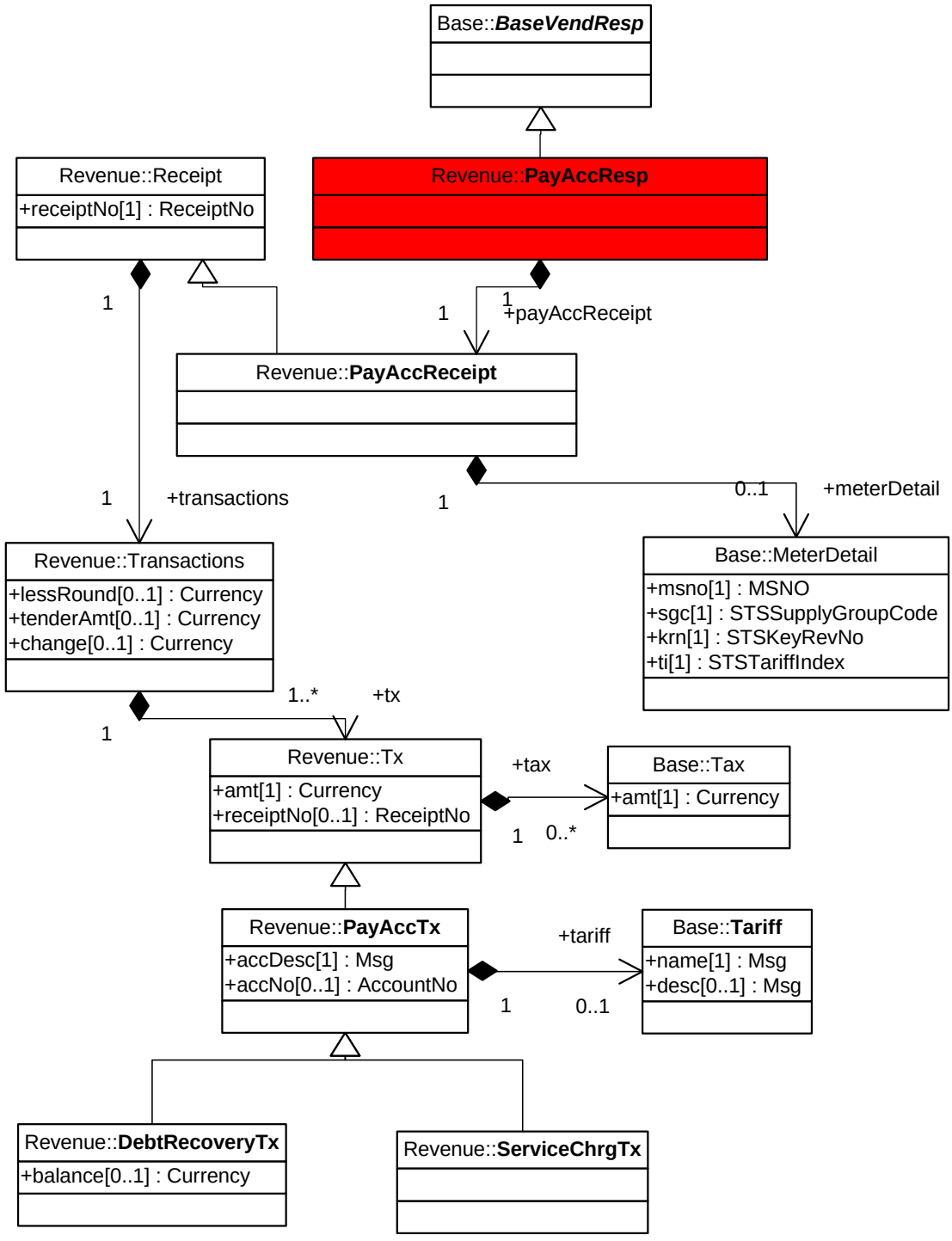

Figure 60 — Pay account response message class diagram

4.7.3.16 Purchase credit token

The purchase credit token request message class diagram is illustrated in figure 61.

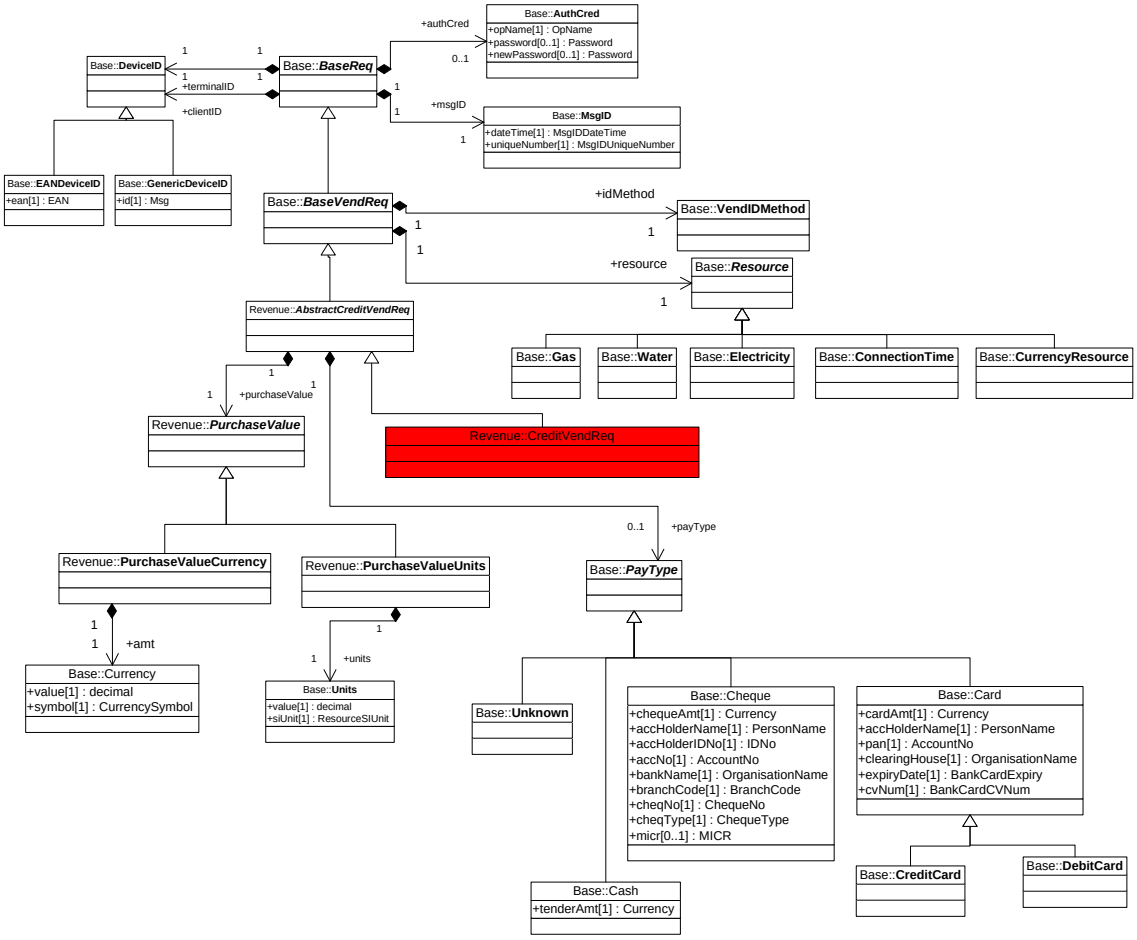

Figure 61 — Purchase credit token request message class diagram

The purchase credit token response message class diagram is illustrated in figure 62.

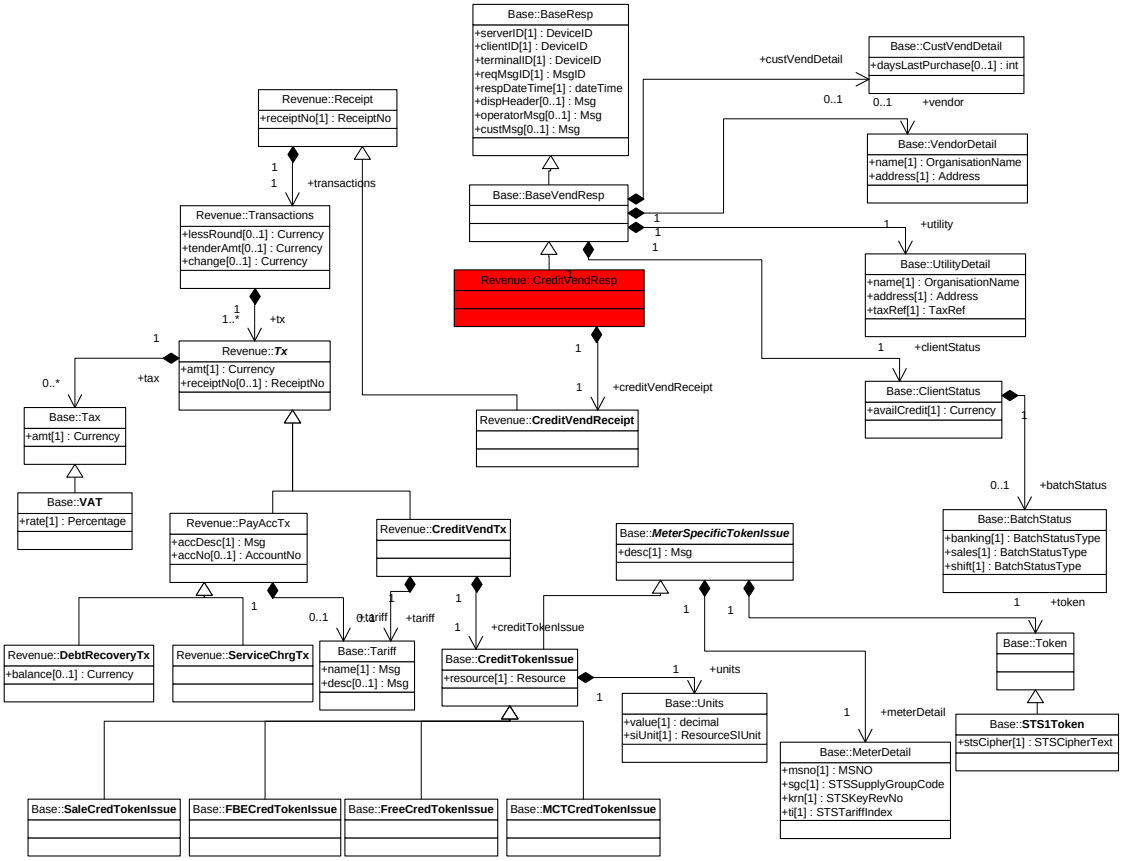

Figure 62 — Purchase credit token response message class diagram

4.7.3.17 Purchase trial credit token

The purchase trial credit token request message class diagram is illustrated in figure 63.

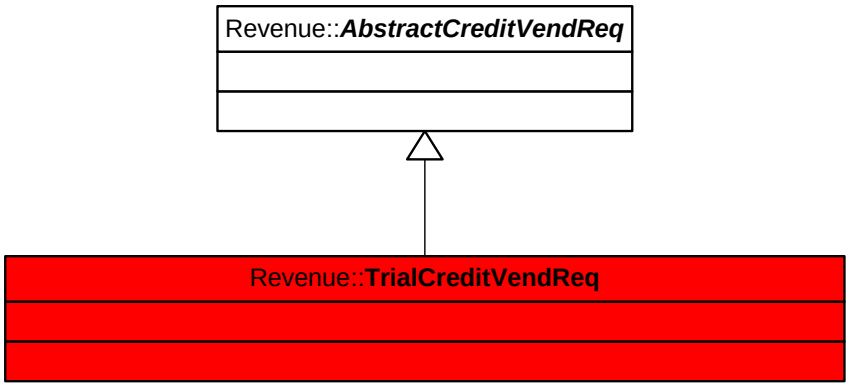

Figure 63 — Purchase trial credit token request message class diagram

The purchase trial credit token response message class diagram is illustrated in figure 64.

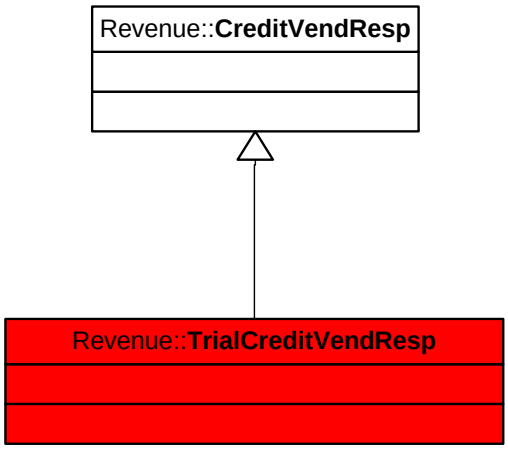

Figure 64 — Purchase trial credit token response message class diagram

4.7.3.18 Reprint transaction

The reprint transaction request message class diagram is illustrated in figure 65.

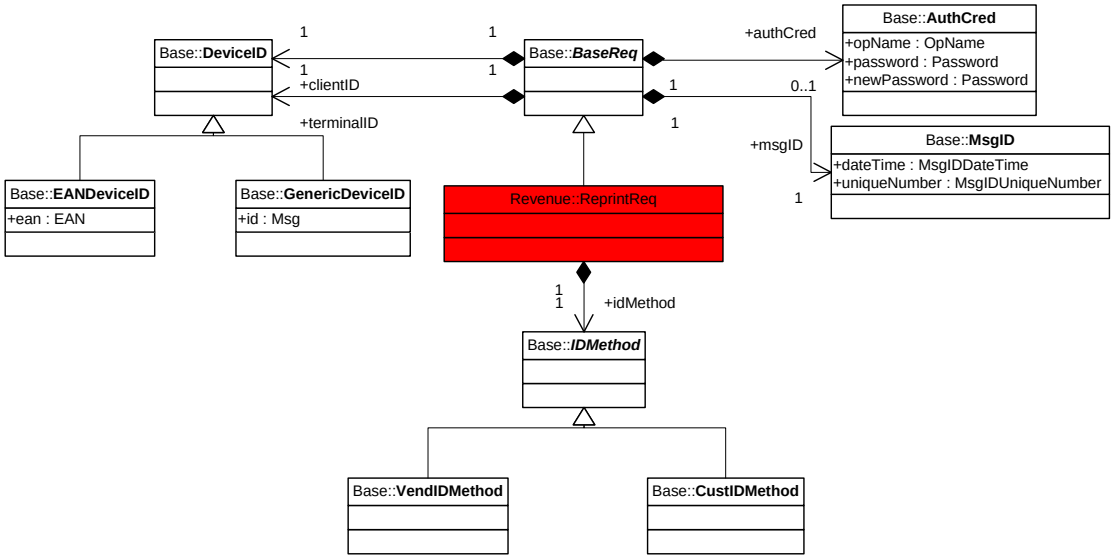

Figure 65 — Reprint transaction request message class diagram

The reprint transaction response message class diagram is illustrated in figure 66.

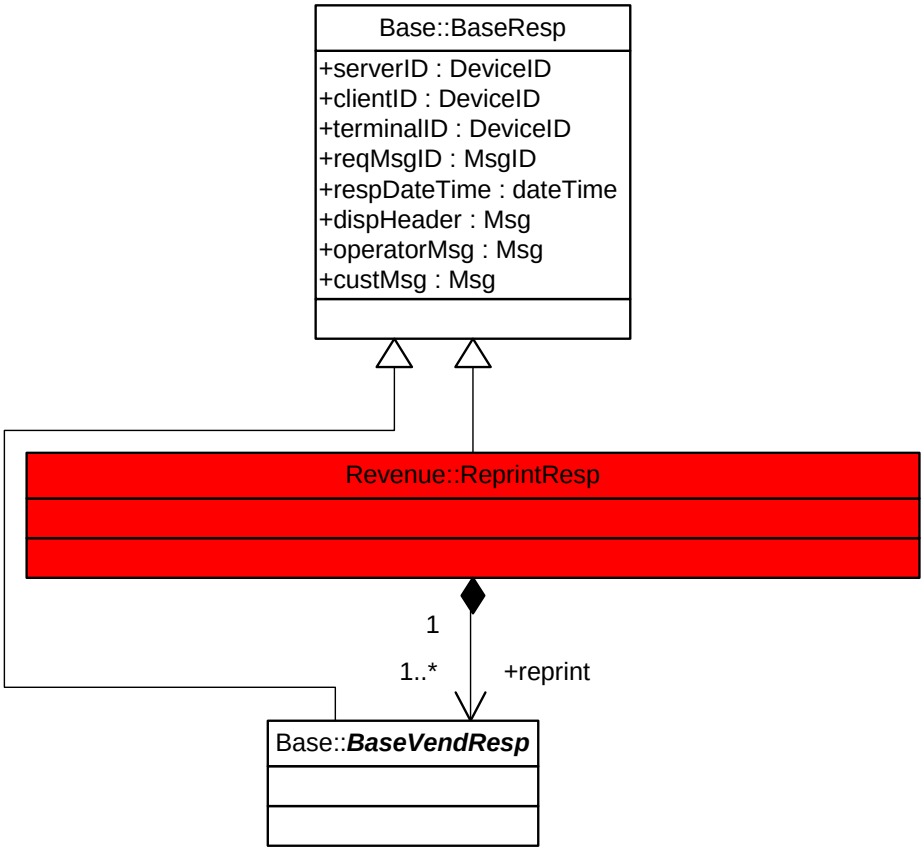

Figure 66 — Reprint transaction response message class diagram

4.7.3.19 Reprint deposit slip

The reprint deposit slip request message class diagram is illustrated in figure 67.

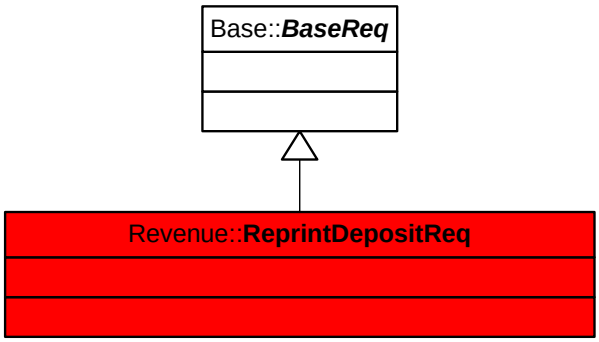

Figure 67 — Reprint deposit slip request message class diagram

The reprint deposit slip response message class diagram is illustrated in figure 68.

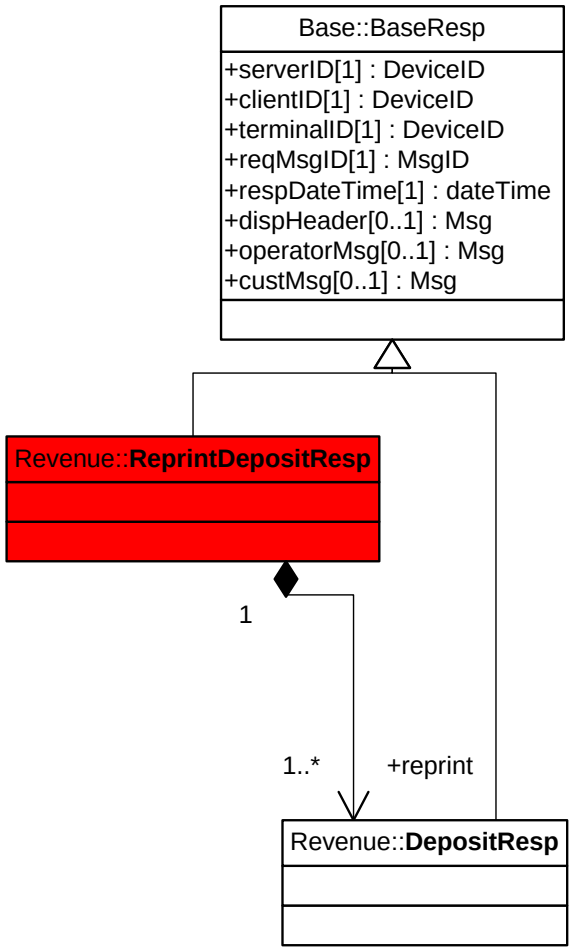

Figure 68 — Reprint deposit slip response message class diagram

4.7.3.20 Reprint end batch

The reprint end batch request message class diagram is illustrated in figure 69.

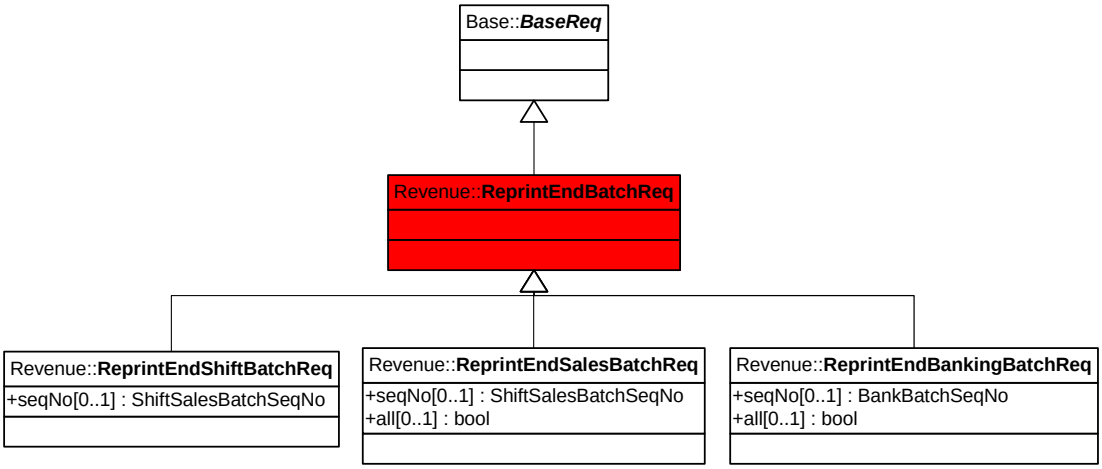

Figure 69 — Reprint end batch request message class diagram

The reprint end batch response message class diagram is illustrated in figure 70.

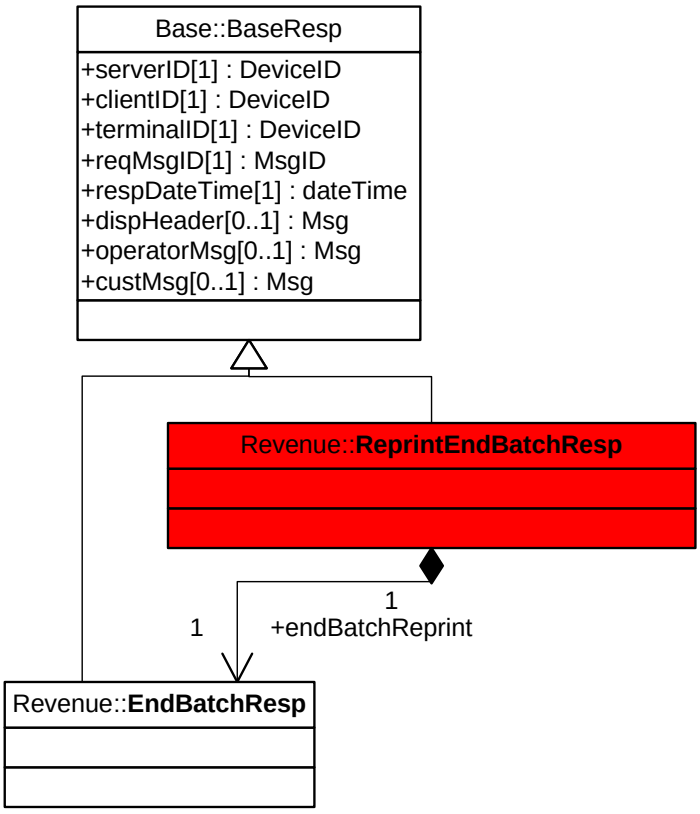

Figure 70 — Reprint end batch response message class diagram

4.7.3.21 Start batch

The start batch request message class diagram is illustrated in figure 71.

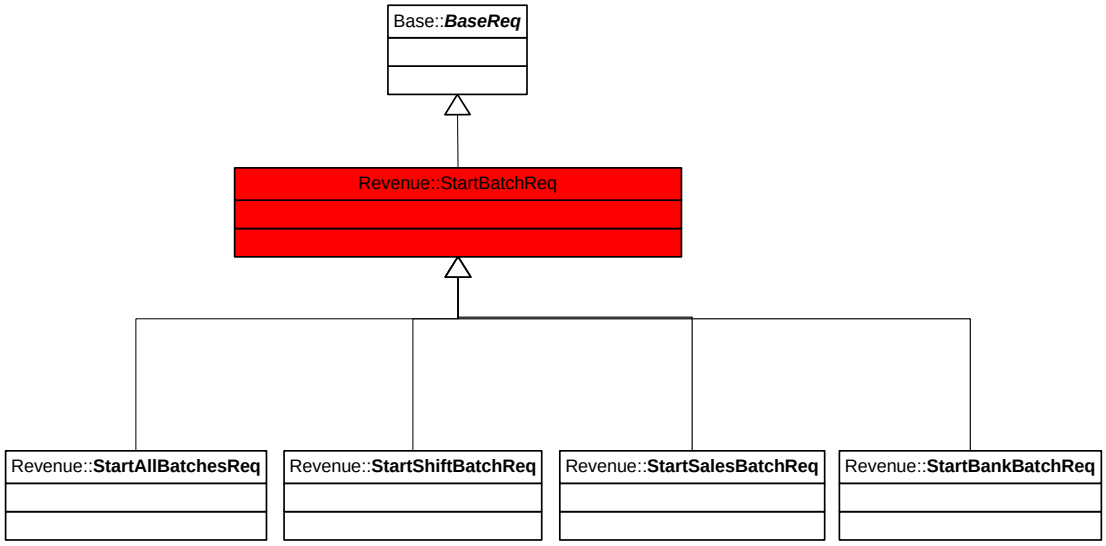

Figure 71 — Start batch request message class diagram

The start batch response message class diagram is illustrated in figure 72.

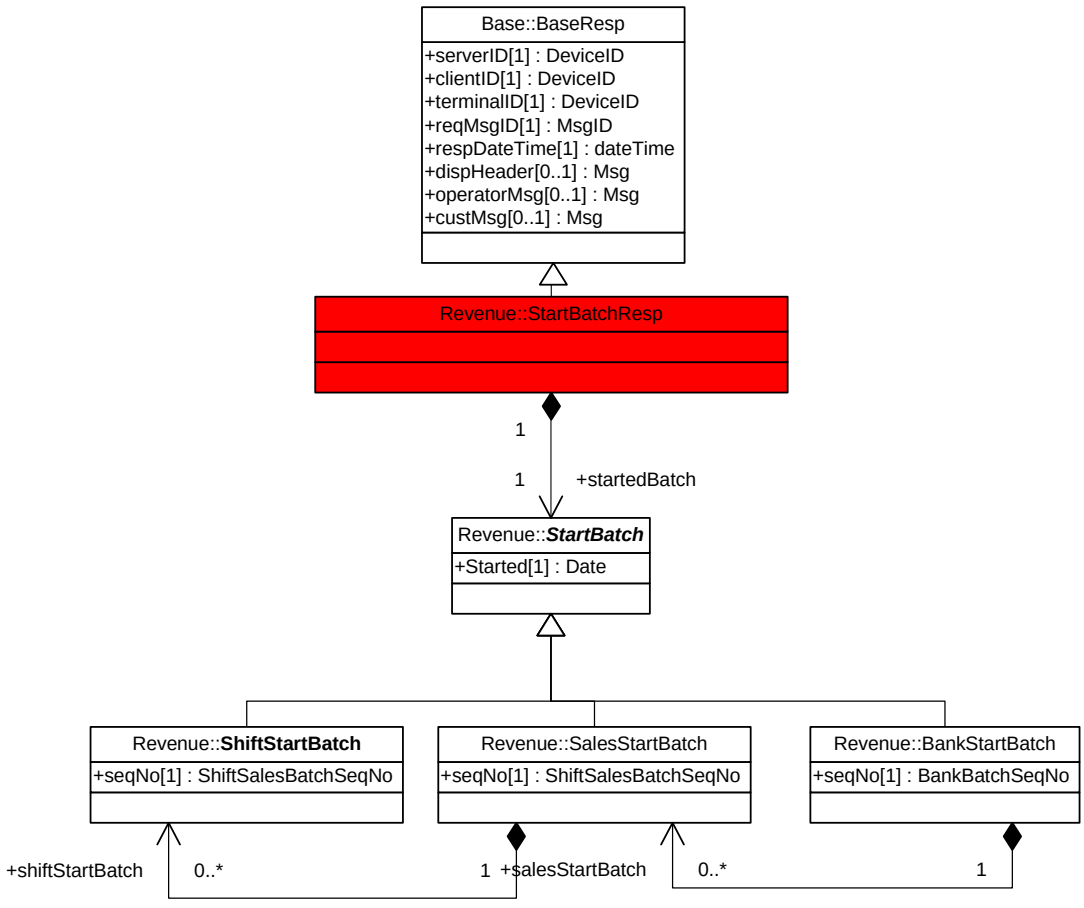

Figure 72 — Start batch response message class diagram

4.7.3.22 Update meter key

The update meter key request message class diagram is illustrated in figure 73.

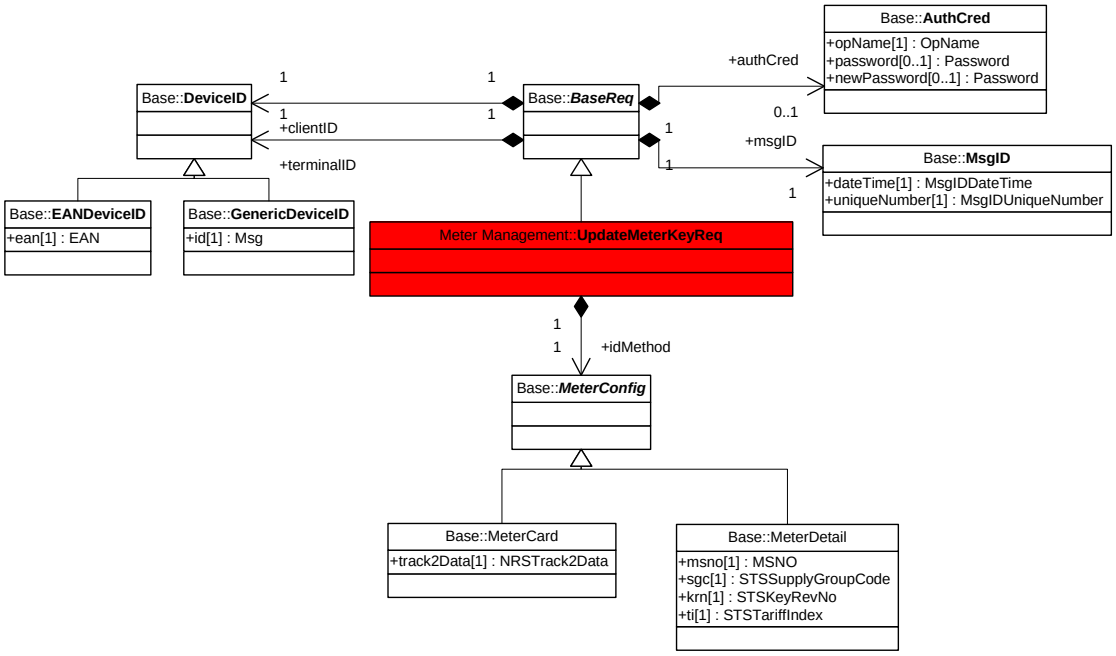

Figure 73 — Update meter key request message class diagram

The update meter key response message class diagram is illustrated in figure 74.

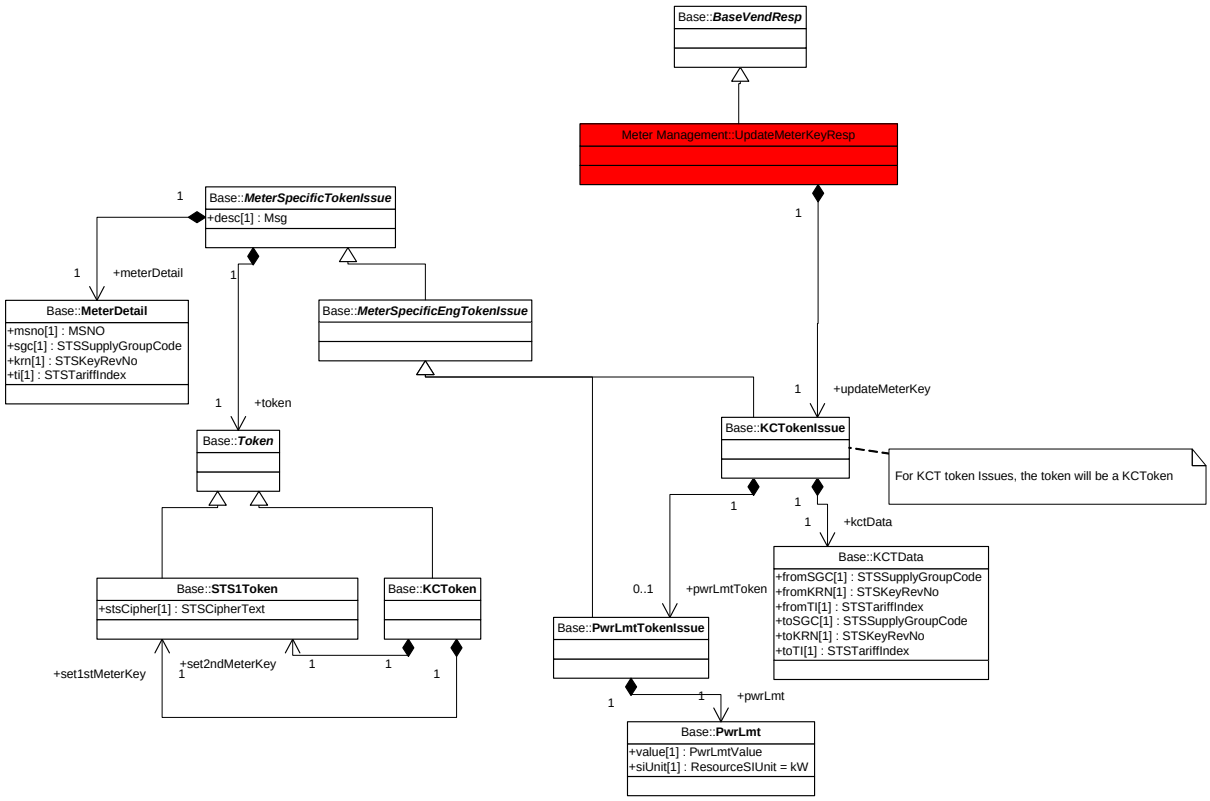

Figure 74 — Update meter key response message class diagram

4.7.3.23 Verify token

The verify token request message class diagram is illustrated in figure 75.

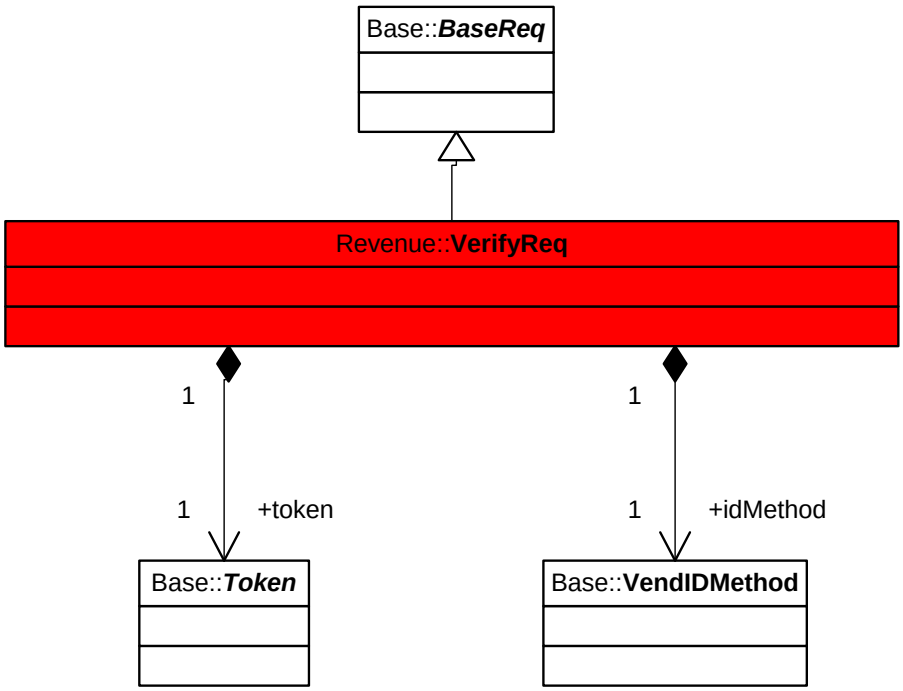

Figure 75 — Verify token request message class diagram

The verify token response message class diagram is illustrated in figure 76.

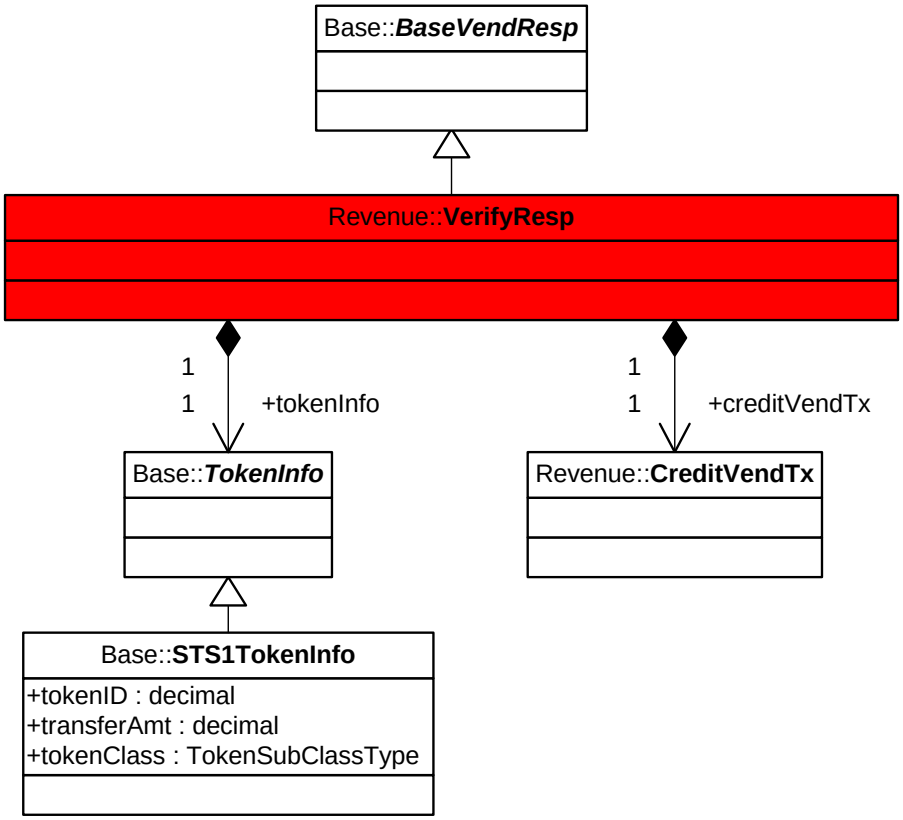

Figure 76 — Verify token response message class diagram

4.7.3.24 Vendor statement

The vendor statement request message class diagram is illustrated in figure 77.

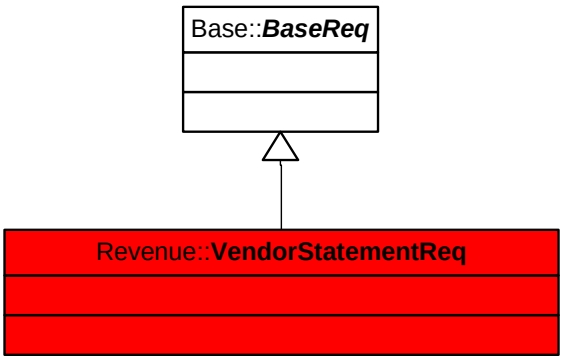

Figure 77 — Vendor statement request message class diagram

The vendor statement response message class diagram is illustrated in figure 78.

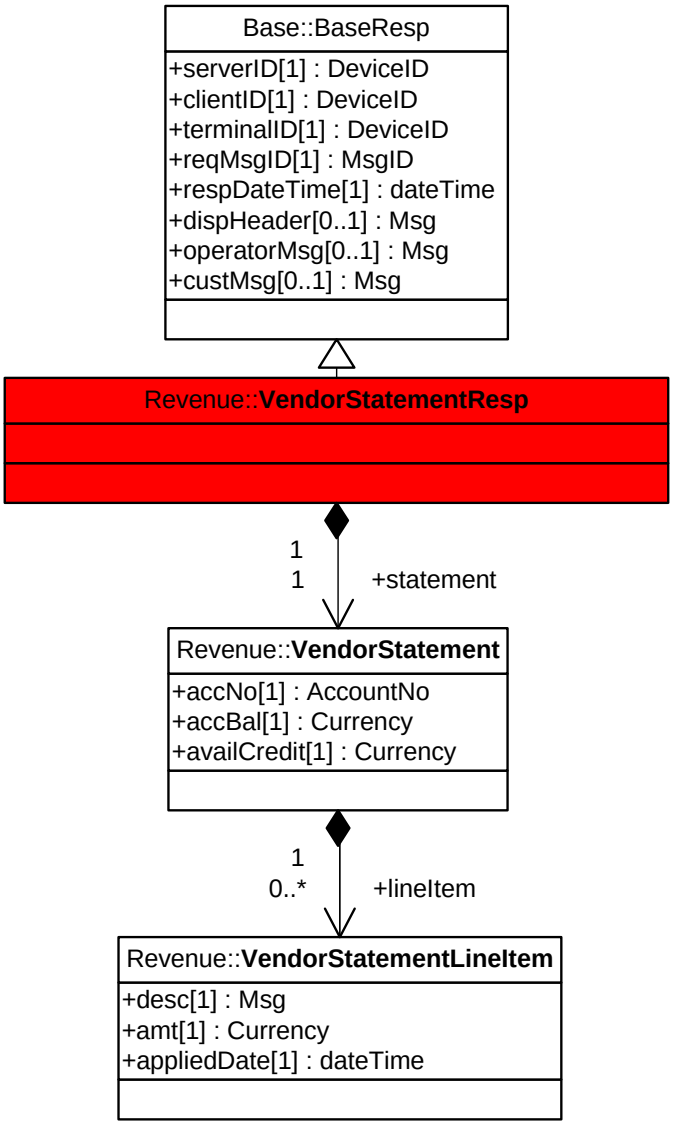

Figure 78 — Vendor statement response message class diagram

4.7.4 Class and attribute descriptions

The class and associated field definitions have been embedded into the schema as <annotations>. They are available on http://www.nrs.eskom.co.za/xmlvend.

5 Web services implementation

NOTE Clause 5 is aimed mainly at suppliers of XMLVend clients and servers. It describes the technical detail of how the messages are required to be securely exchanged between the XMLVend client and server.

5.1 Introduction

XMLVend currently specifies web services as the method to implement the use case message pairs described in 4.7. Web services provide the following functions:

a) platform-independent format language for structured data exchanged. This is achieved through the use of eXtensible markup language (XML);

b) a way of describing the structure of the data being exchanged. This is achieved through the use of XML Schema;

c) a standard method of packaging the data for transmission over the communications network. This is achieved through the use of protocol known as “SOAP”;

d) 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); and

e) a standard method of transporting the data across the network. This is achieved through the use of hypertext transport protocol (HTTP) and TCP/IP.

To further assist with interoperability and adoption of web services the Web Service Interoperability (WS-I) organization has published guidelines for implementation of web services. The guidelines or profiles have been adopted by this specification (XMLVend). Further details on web services and WS-I is provided in annex C.

This clause describes the mapping of the use case class diagrams to a web service, implementing the web services stack, message exchange workflow, and fault handling using Web Services.

5.2 Mapping use case class diagrams to the XMLVend Web service

5.2.1 The development process

The Web service specification deliverables are the XMLVend schema and WSDL. The process used to develop the schema and WSDL is based on the “Contract-First” development approach. This approach is illustrated in figure 79.

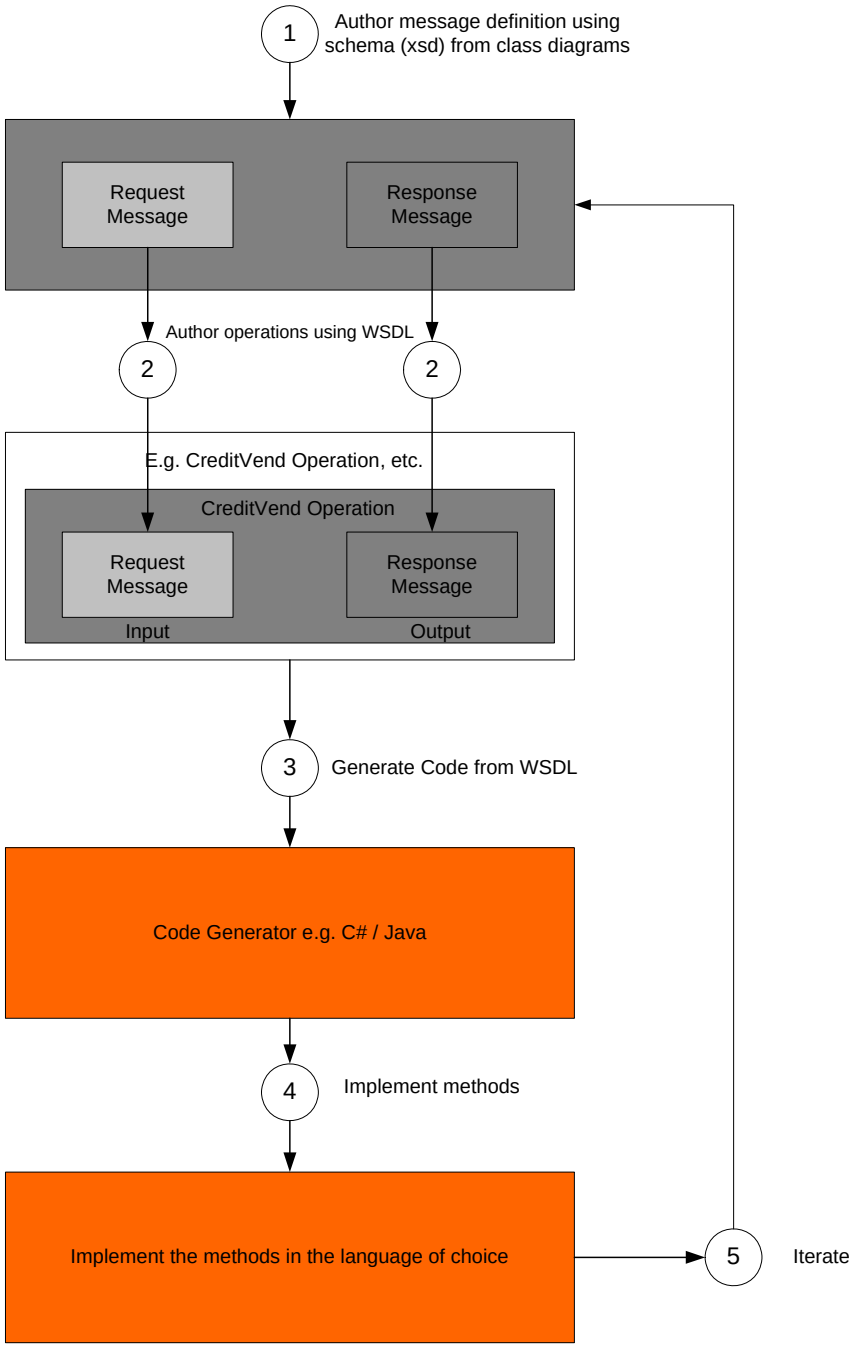

Figure 79 — Schema and WSDL development approach

Steps 1, 2 and 5 are part of the XMLVend specification development process. Steps 3 and 4 are used by suppliers to implement an XMLVend compliant server or client (or both) and therefore not discussed in this specification.

Step 1 is used to author the request or response messages using schema, modelled on the class diagrams described in 4.7. This step focuses on naming and structuring the data messages that are exchanged between client and server.

Step 2 is used to author the service operations and transport bindings using WSDL. The WSDL operation defines the schema message pairs that shall be exchanged to implement a particular use case. The operations are then grouped into a WSDL interface (or portType). WSDL also defines the transport binding details that are needed to exchange messages between client and server. WSDL contains all the information required for code generators to produce the implementation code for both client and server.

Step 5 provides feedback from the implementations to update and enhance the specification.

5.2.2 The schema and WSDL

5.2.2.1 General

The schemas have been mapped from the class diagrams described in 4.7. This mapping process is currently manual and some unintentional errors have been introduced. Therefore, the schemas shall be considered the definitive specification for the XMLVend message pairs, supported by the class diagrams.

Figure 80 shows the relationships between the domain-specific schemas and WSDLs. The diagram also shows how the core schemas can be extended to support user-specific requirements. The extension of the core XMLVend schema is described in 5.2.2.3.

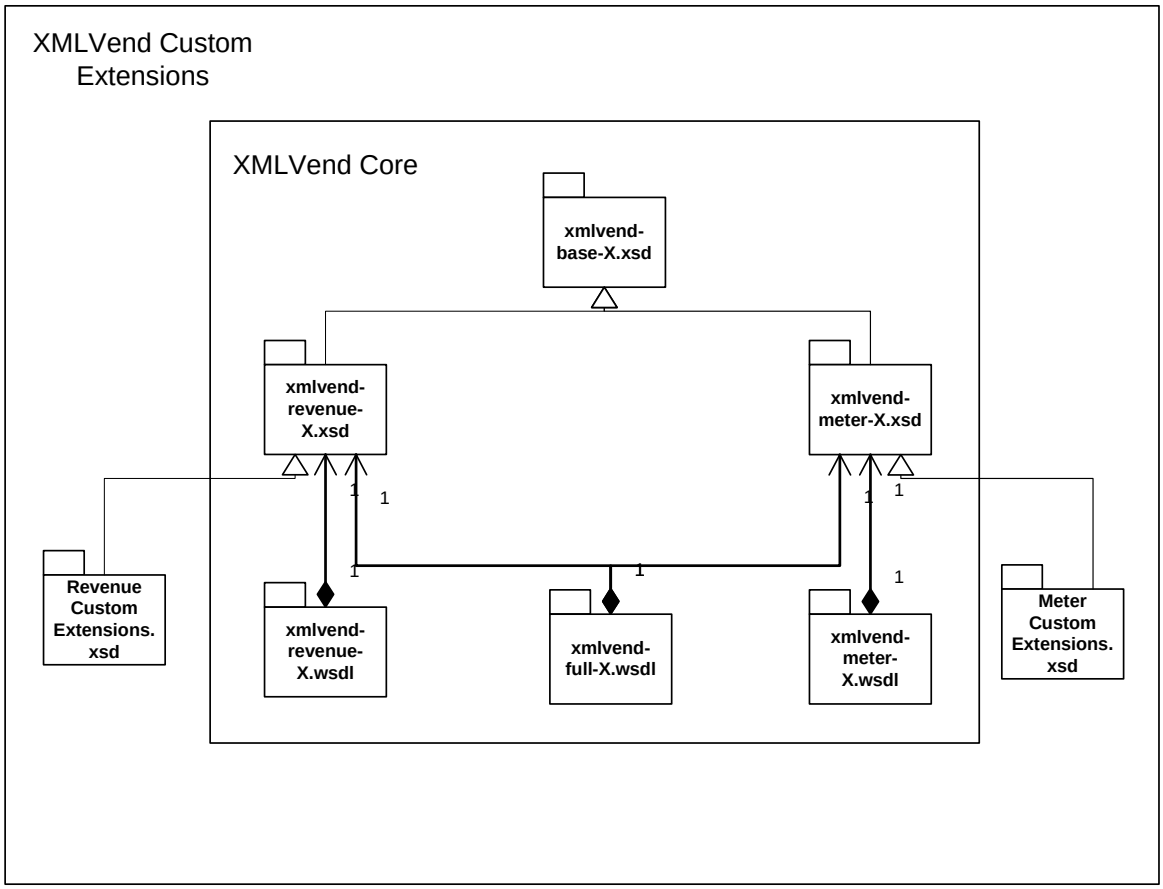

Figure 80 — Schema and WSDL for each domain

The core XMLVend schema and WSDL consist of the following:

a) schemas:
   1) xmlvend-base-X.xsd – the base schema contains common types, constraints and message pairs that are used by both the revenue and meter schemas;

   2) xmlvend-revenue-X.xsd – the revenue schema contains all message pairs and types that are specific to the revenue management domain; and

   3) xmlvend-meter-X.xsd – the meter schema contains all message pairs and types that are specific to the meter management domain.

b) WSDLs and

   1) xmlvend-full-X.wsdl – the full WSDL includes all operations (use cases). This WSDL would usually be used by suppliers that implement use case from both the revenue and meter management domains;

   2) xmlvend-revenue-X.wsdl – the revenue WSDL contains the revenue management specific operations (use cases). This WSDL shall be used by suppliers that implement revenue management operations only; and

   3) xmlvend-meter-X.wsdl – the meter WSDL contains the meter management specific operations (use cases). This WSDL shall be used by suppliers that implement meter management operations only.

The schema file name naming convention is xmlvend-“domain”-“version”.xsd

The WSDL file name naming convention is xmlvend-“domain”-“version”.wsdl

The WSDLs and schema files are available for download on
<http://www.nrs.eskom.co.za/xmlvend>.

5.2.2.2 Versioning with namespaces

The XML namespaces serve two purposes, as follows:

a) to ensure uniqueness of XML element and attribute names (eliminate name collisions); and

b) to provide a URI to specify the language associated with a given XML element or attribute.

The schema namespace naming convention is
http://www.nrs.eskom.co.za/xmlvend/“domain”/“version”/schema

The WSDL (service) namespace 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 “full” WSDL namespace, which includes the meter and revenue domain services does not have a “domain” specified.

NOTE The namespaces of the individual schemas may change independent of each other.

5.2.2.3 User-specific schema extensions

Extensions shall be specified in a separate schema file. The file name naming convention to be used is “utility”-xmlvend-“domain”-“version”.xsd, e.g., eskom-xmlvend-revenue-2.0.xsd.

The extension schema namespace naming convention to be used is
http://www.”utility”/xmlvend/“domain”-ext/“version”/schema,
e.g. http://www.eskom.co.za/xmlvend/revenue-ext/2.1/schema

An example of user-specific schema extensions is provided on
http://www.nrs.eskom.co.za/xmlvend.

The user-specific extensions may be implemented by generating the required code from the schema file discussed above and using the late binding mechanism in the XML, using the `<xsi:type>` attribute of the relevant element in the XML instance file.

5.2.2.4 Schema optional elements and attributes

Elements are specified as optional in the schema using minOccurs attribute, for example <element name=”KRN” minOcurrs=”0” />. XML instance documents that specify <KRN xsi:nil=”true”/> or <KRN/> for an omitted optional element is not valid since optional elements shall either be present or totally omitted. Optional attributes shall also be treated similarly.

5.2.3 The XMLVend Web services stack

5.2.3.1 Overview

Each layer of the Web services stack represents one of the fundamental functional areas of a web service instance. These layers are depicted in figure 81.

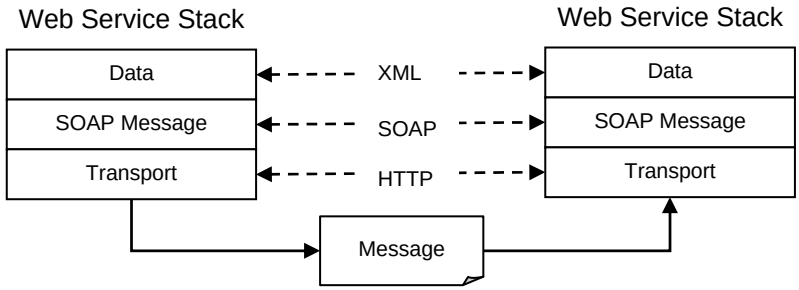

Figure 81 — Web services stack

A web service application may include several logical layers incorporating functions such as the web service instance and application business logic. The WS-I basic profile and usage scenarios do not address application business logic except where the functionality of any part of the web services stack is implemented within the business logic.

A set of activities is defined for each layer of the Web services stack. Activities are the fundamental operations that comprise a Web service. A single activity has several constraints applied to it from the basic profile. For example, one activity might be “Send HTTP” and the specifications and guidelines for how to fulfil that activity come from the SOAP 1.1 and HTTP sections of the basic profile. Activities are summarized in table 4 below.

Table 4 — Activities grouped by Web services stack layer

1	2
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

5.2.3.2 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 following activities are part of the data layer.

a) Write XML: Application-level messages that are to be exchanged during a Web services interaction shall be written to a serialized form that can be transported with the underlying transport protocol. These messages use the data types and formats declared in the schemas. Writing the message data is the responsibility of the application component sending a message to a recipient. The output of this activity is often referred to as the SOAP payload.

b) Process XML: Application-level messages that are exchanged in a Web services interaction are passed to application components responsible for receiving, interpreting and acting upon the received messages. Application components process message data according to the types and formats declared in the schemas.

The data layer is the most important layer with respect to XMLVend specification because the data layer describes and transports the XMLVend messages pairs between XMLVend client and XMLVend server.

5.2.3.3 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 portTypes and bindings. On the receiving side the message layer processes the SOAP messages and dispatches requests to the correct application or method.

The following activities are executed with the SOAP message layer:

a) SOAP envelope;

   1) write SOAP envelope; and
   2) process SOAP envelope.

b) SOAP body;

The SOAP body is used for transporting application-specific information included in the application message data. The activities in this layer are different from the data payload writing and processing activities described in the data layer activities section.

1) write SOAP body; and
2) process SOAP body.

c) SOAP header;

The SOAP header provides a modular mechanism for extending a SOAP message.

1) write SOAP header; and
2) process SOAP header.

5.2.3.4 Transport layer

The transport layer sends and receives messages. For the basic profile, this includes only HTTP client and server platforms. This layer maps to the wsdl:binding and wsdl:port definitions with the WSDL document.

SOAP messages shall be sent using the send and receive HTTP transport protocols.

NOTE GZIP compression and TLS security are also applied at this layer.

5.2.4 The XMLVend message exchange pattern

5.2.4.1 Overview

The XMLVend message pairs exchanged between XMLVend Client and Server are implemented using synchronous request / response pattern as described in the WS-I 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.

The high-level interactions between an XMLVend client and server using a synchronous request/response pattern are illustrated in figure 82.

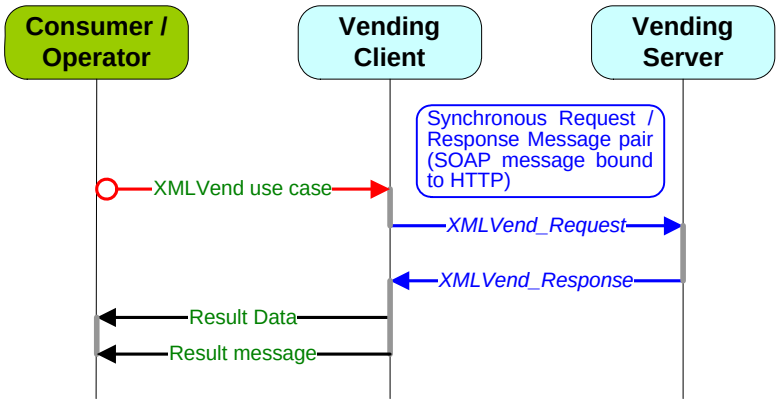

Figure 82 — Synchronous request/response (SOAP message bound to HTTP) sequence diagram

5.2.4.2 Request/response message pair flow

5.2.4.2.1 The detailed flow for this scenario, using the activities defined in 5.2.3, is described below. Each bulleted item represents the activities performed within one layer of the Web services stack required to complete the flow. Each activity has constraints imposed upon it from the WS-I Basic Profile.

5.2.4.2.2 The XMLVend client initiates the following SOAP request (see figure 83):

a) data layer;
   1) write XML.

b) SOAP message layer; and
   1) write SOAP envelope;
   2) write SOAP body.

c) transport layer;
   1) optional compress HTTP body message;
   2) send HTTP.

5.2.4.2.3 The XMLVend server receives the SOAP request:

a) transport layer;
   1) receive HTTP;
   2) uncompress HTTP body message (if required).

b) SOAP message layer;
   1) process SOAP envelope;
   2) process SOAP body.

c) data layer; and
   1) process XML. The data payload is processed according to the data model and dispatched to the server application.

d) application layer;
   1) process security credentials as described in 5.2.6;
   2) process business rule compliance.

5.2.4.2.4 Request flow is illustrated in figure 83.

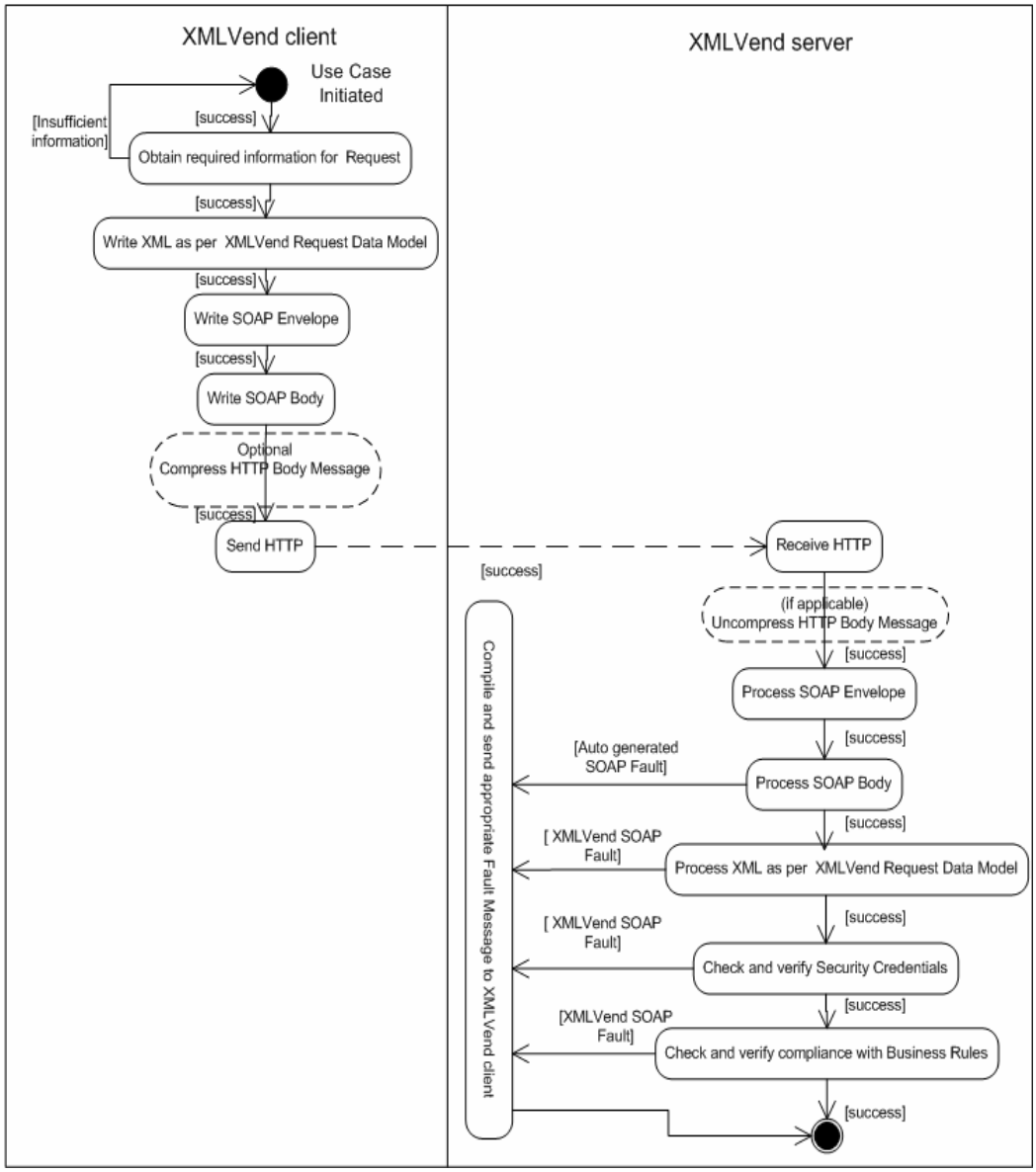

Figure 83 — Request flow

5.2.4.2.5 The vend server generates an appropriate SOAP response as follows (see figure 84):

a) data layer;
   1) write XML. The payload is created according to the XMLVend data model.

b) SOAP message layer; and
   1) write SOAP envelope;
   2) write SOAP body;

c) transport layer;
   1) optional compress the HTTP body;
   2) send HTTP;

5.2.4.2.6 The XMLVend Client receives the SOAP response:

a) transport layer;
   1) receive HTTP;

b) SOAP message layer;
   1) process SOAP envelope;
   2) process SOAP body;

c) data layer; and
   1) process XML - the data payload is processed according to the data model and dispatched to the client application.

d) application layer;
   1) process security credentials as described in 5.2.6;
   2) process business rule compliance.

5.2.4.2.7 Response flow is illustrated in figure 84.

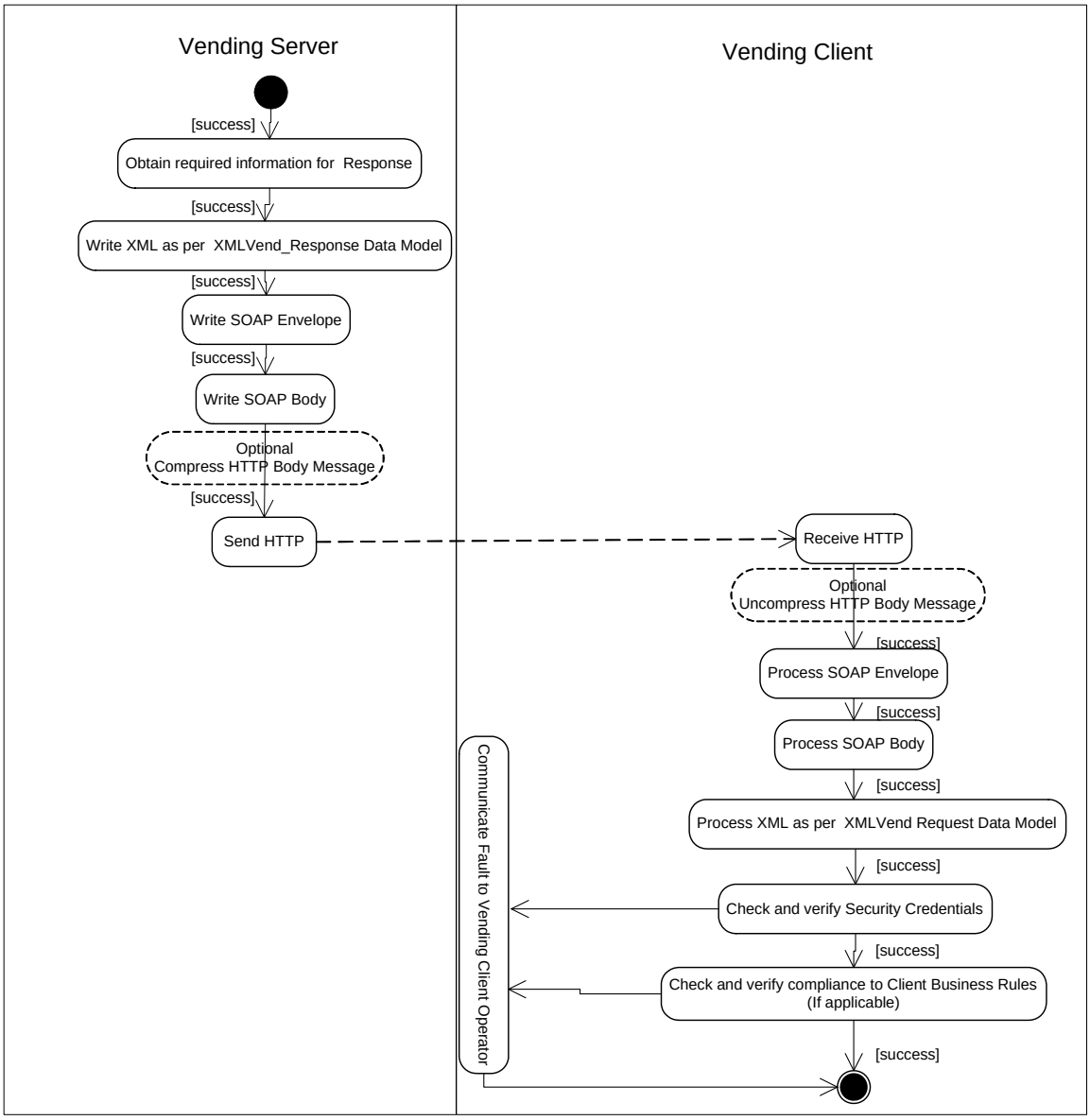

Figure 84 — Response flow

5.2.4.2.8 The following activities shall adhere to the referenced constraints described for the synchronous request/response usage scenario in WS-I usage scenarios specification:

a) write XML;

b) write SOAP envelope;

c) write SOAP body;

d) send HTTP;

e) receive HTTP;

f) process SOAP;

g) process SOAP body; and

h) process XML.

5.2.4.3 Message uniqueness

All request messages contain a unique message identifier in the <msgID/> field. The XMLVend client application is responsible for generating the unique message identifier. The unique message identifier consists of a sequence number and a date time stamp. Client application developers shall take appropriate care not to generate duplicate message identifiers should the client date time settings be manipulated.

The XMLVend server shall verify that each client request message identifier is unique with respect to that client. The server shall raise the appropriate XMLVendFaultEx (see 4.6) should it encounter a duplicate message identifier for a specific client.

5.2.4.4 Application layer error handling

As described in 4.6.3 and 4.7.3.9, a standard fault response message has been defined to communicate fault scenarios to the client. This subclause describes how the XMLVend fault response message is integrated with SOAP's fault communication mechanism.

Application layer errors that occur while a request is being serviced are communicated to the client using the SOAP fault mechanism. The specific application error information will be compiled as per XMLVend fault Response message (see 4.7.3.9) and included into the `<detail>` section of a SOAP `<fault>` element.

Fault generation shall adhere to constraints and behaviour as described for the synchronous request/response scenario usage scenario in WS-I Usage Scenarios specification.

5.2.4.5 Data transport

The data transport protocol between XMLVend Client and XMLVend Server will be HTTP as specified for the synchronous request / response usage scenario in WS-I Usage Scenarios specification.

The HTTP SOAPAction header shall be omitted or, if present, contain an empty string (“”). Therefore, no decision making by the server or client shall be based on this header. This removes the close coupling between the XMLVend protocol and HTTP, allowing other transports to be utilized in the future.

5.2.5 Message compression

5.2.5.1 General

XMLVend compression is implemented at the HTTP layer of the protocol stack and shall be implemented as specified in the HTTP 1.1 protocol specification.

5.2.5.2 Compression algorithm

The supported compression algorithm is GZIP <http://www.gzip.org>. It is recommended that the latest official release of GZIP be implemented.

5.2.5.3 Compression implementation rules

HTTP messages will optionally be compressed (in both directions) by making use of the standard HTTP content encoding mechanism. This is achieved by the client setting the appropriate HTTP request headers, for example:

a) content-encoding: gzip; and

b) accept-encoding: gzip.

As per HTTP protocol, the 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.

5.2.6 Security

5.2.6.1 Security overview

Web services' as with other information technologies, security consists of understanding the potential threats an attacker may mount, and applying operational, physical, and technological countermeasures to reduce the risk of a successful attack to an acceptable level. Because an "acceptable level of risk" varies hugely depending on the application and because costs of implementing countermeasures are also highly variable, there can be no universal "right answer" for securing Web services. Choosing the absolutely correct balance of countermeasures and acceptable risk can only be done on a case-by-case basis.

See annex D for details on security risks and threats that web service base systems shall consider.

5.2.6.2 Security mechanisms

5.2.6.2.1 Options

XMLVend may use the security mechanisms given in 5.2.6.2.2 and 5.2.6.2.3.

5.2.6.2.2 Dedicated communications link

A dedicated communications link is used between the XMLVend server and XMLVend client. An example of such a link is a Virtual Private Network (VPN) link.

Client authentication at the application layer may be enabled using the username and password fields contained in request messages.

5.2.6.2.3 Transport Layer Security (TLS)

5.2.6.2.3.1 General

XMLVend adopts and recommends the use of HTTP secured with either TLS 1.0 or SSL 3.0 (HTTPS) as specified in the WS-I profile. HTTPS is widely regarded as a mature standard for encrypted transport connections to provide a basic level of confidentiality. HTTPS thus forms the first and simplest means of achieving some basic security features that are required by many real-world web service applications.

5.2.6.2.3.2 Compliance

TLS implementations shall comply with the constraints specified for the Synchronous Request / Response usage scenario in WS-I usage scenarios specification.

5.2.6.2.3.3 Authentication

The XMLVend implementation of HTTPS will require both server and client authentication through the use of server and client digital certificates. The client and server private keys should be stored in a secure location. A hardware storage medium is recommended. For the clients that also act as POS terminals a portable hardware storage medium such as a smart card is recommended. This scenario would have the added security benefit of disenabling the XMLVend client when the smart card is removed, reducing the risk of unauthorized use of the XMLVend client when it is unattended.

The server shall also verify that the <clientID/> field in every request message matches the client certificate common name (CN) field. If the fields do not match, the appropriate XMLVendFaultEx shall be raised (see 4.6).

5.2.6.2.3.4 TLS session timeout

The TLS session timeout determines the time period that a session key is valid after which a new session key shall be negotiated.

A lengthy session timeout could provide sufficient time for “hacker” to crack the session key. It is therefore recommended that the maximum session timeout be 12 h, i.e. a new session shall be re-negotiated every 12 h cycle.

5.2.6.2.3.5 Certificate management

The management of the digital certificates is beyond the scope of this specification but the certificate management process should be a joint agreement between transacting parties.
A typical process for certificate request and signing has been compiled as a guide for testing and piloting XMLVend. This document is available on <http://www.nrs.eskom.co.za/XMLVend>.

5.2.7 Testing and verifying compliance

Implementations can check compliance with WS-I requirements by utilizing the WS-I test tools. Supplier interoperability can be accomplished by testing between supplier equipment. An XMLVend reference server and client implementation and compliance test suite has been developed and the latest version is available on <http://www.nrs.eskom.co.za/XMLVend>.

6 Change management process

The change management process is illustrated in figure 85. Updates and modifications to the specification shall be managed through this process.

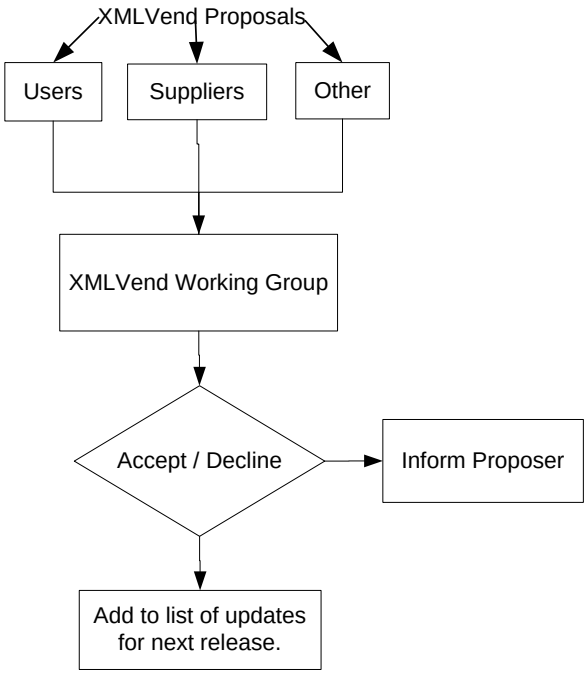

Figure 85 — XMLVend specification change management process

7 Constraints concerning group-coded SGCs

Group-coded SGCs depend on the token being erased after use. The token technology is restricted magnetic cards. However, online vending servers cannot ensure that the token is finally encoded on a magnetic card. Therefore, vending to group-coded SGCs shall be avoided by online vending systems. This can be achieved by preventing group-coded SGCs from being loaded on the server's security module. This should be controlled and monitored by utilities and server suppliers.

Annex A
(informative)

Example fault descriptions

1	2	3	4
<BusinessRuleEx/> specialization	**Suggested text for <desc/> field**	**Suggested text for <operatortMsg/> field**	**Suggested text for <custMsg/> field**
ClientIDSSLEx	The client ID in the SSL certificate does not match the ClientID in the request message.	The vending client application has caused security exception error on the server related to its ClientID. Please contact your service provider.	N/A
LastResponseEx	No associated request message on the server.	The server did not process the requested transaction. The required action can be retried.	N/A
UseCaseSupportEx	The server does not support the requested use case.	The server does not support the request function. Contact your service provider.	N/A
XMLVendSchemaEx	The request message does not conform to the XMLVend schema.	The vending client application has caused a message validation error on the server. Verify the data and retry. If the problem persists, contact your service provider.	N/A
BlockedMeterEx	The meter is blocked.	The meter has been blocked. The transaction cannot be completed.	The meter has been blocked. Please contact your service provider.
FBEEx	The customer does not qualify for FBE.	The customer does not qualify for FBE.	You are not configured for FBE. Please contact your service provider.
LatestKRNEx	KRN = X for SGC = YYYYYY has expired.	KRN = X for SGC = YYYYYY has expired. Do Update Meter Key request and encode Meter Card.	
MSNOCheckDigitEx	Meter serial number (MSNO) fails the meter check digit validation.	Invalid Meter Number – Retry with meter number from an old token or use a meter card.	

Annex A
(continued)

1	2	3	4
<BusinessRuleEx/> specialization	**Suggested text for <desc/> field**	**Suggested text for <operatortMsg/> field**	**Suggested text for <custMsg/> field**
RequestAuthorizationEx	Vendor not authorized to perform the requested function.	Vendor not authorized to perform the requested function. Please contact your service provider or try another function.	
SGCAuthorizationEx	Vendor not configured for SGC=XXXXXX.	Vendor not configured for SGC=XXXXXX. Please contact your service provider.	
STSDataEx	Incorrect STS data is supplied.	STS parameter X = YYY, is valid. Verify the data provided and retry.	
VendorCreditEx	Insufficient vendor credit available to perform the transaction.	Insufficient vendor credit available to perform the transaction. Please update credit.	
ServiceChrgEx	No outstanding service charge for payment.	The customer has no outstanding service charge for payment.	
UnknownMeterEx	The meter XXX is not registered on the server.	Meter XXX is not registered on the server. Request cannot be processed. Please contact the service provider.	
UnknownMeterUpdate MtrKeyEx	Cannot Update Meter Key, meter not registered.	Meter XXX, not registered. Do Engineering Key Change or contact your service provider to register the meter.	Meter XXX, not registered. Contact your service provider to register the meter.
UpdateMtrKeySameEx	“From” meter key data same as “To” meter key data.	MSNO=XXXXXXXX, “FROM” key data is the same as the “TO” key data, SGC=XXXXXX, KRN=X, TI=XX”. Key Change not required.	
VerifyTokenEx	Unable to verify token	Not verified – Token does not match meter information supplied.	Not verified – Token does not match meter information supplied.
ClientIDAuthorizationEx	Client ID not registered or blocked.	Client ID XXX, not registered or blocked. Contact Service Provider.	This client is not registered. Please try another vendor.
ConfirmCustomerEx	Customer not found.	Customer not found. Try different search criteria.	
CancelEx	Token cannot be cancelled.	The token cannot be cancelled. Contact service provider.	

Annex A
(concluded)

1	2	3	4
<BusinessRuleEx/> specialization	**Suggested text for <desc/> field**	**Suggested text for <operatortMsg/> field**	**Suggested text for <custMsg/> field**
CheckBatchTotalEx	The batch total cannot be checked.	The batch total cannot be checked as the batch is not open.	
DebtEx	No outstanding debt.	No outstanding debt available for customer/meter.	No outstanding debt. Please contact your service provider for assistance.
EndBatchEx	The batch cannot be closed.	The batch cannot be closed as the batch is not open, or child batches are still open. Contact service provider if problem persists.	
InsufficientMeterDataEx	Meter XXXXXXXX, not registered. Additional meter information required.	Meter XXXXXXXX, not registered. Additional meter information required from a Meter Card or Old Token.	Meter XXXXXXXX, not registered. Additional meter information required from a Meter Card or Old Token.
ReprintDepositSlipEx	No deposit slips available for reprint.	No deposit slips found for vendor XXXXX.	
ReprintEndBatchEx	No closed batches available to reprint.	No closed batches available for reprint for vendor, XXXX.	
ReprintEx	No transactions found for meter XXXX.	No transactions found for meter XXXX.	No transactions found for meter XXXX.
RequireMeterCardEx	Require meter card.	Must swipe a valid meter card to enter meter data.	Must swipe a valid meter card to enter meter data.
StartBatchEx	The batch cannot be opened.	The batch cannot be opened due to an already open batch. Close the open batch and retry.	

Annex B
(informative)

UML notation overview

The XMLVend message pair class diagrams are defined using UML notation. This annex presents an overview of the UML notation used in the diagrams.

The class diagrams make use of two class relationships, specialization and composition.

Figure B.1 illustrates the specialization relationship. Specialization is a very strong relationship – it is an ***is a*** relationship. This is like where the “Sedan” is a special instance of “Car” class. Specialization allows for substitutability and inheritance. The relationship is shown in figure B.2.

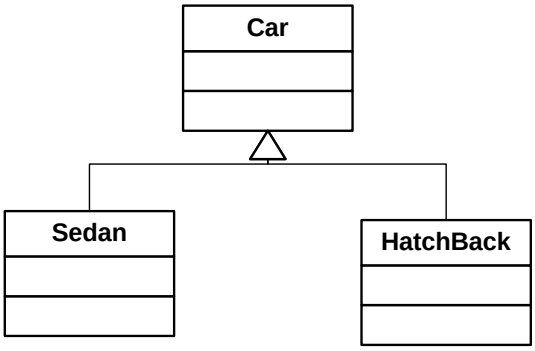

Figure B.1 — Specialization relationship (*is a*)

Figure B.2 illustrates the composition relationship. Composition is a strong ***has a*** relationship. This means that the “car” has one or more doors. It further implies that the doors are only accessible through the car class.

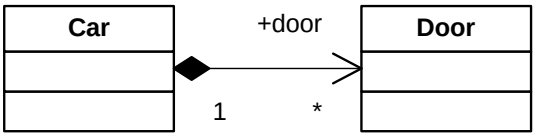

Figure B.2 — Composition relationship (*has a*)

Composition relationships also specify multiplicity. In the above example, the “1” and “*” mean each car instance can have many tyres associated with it. The following are other multiplicity notations:

a) “0.1” – None or one only;

b) “0..*” – None or one and more; and

c) “1.* – At least one but can be more than one.

Annex C
(informative)

Overview of SOAP/XML Web services and WS-I

C.1 At the simplest level, a Web service is a programmable logic accessible using standard Internet protocols. It consists of a service-agnostic request handler (a listener/server) that receives SOAP/XML message requests, and a facade layer that exposes the operations supported by the underlying business logic. The responses are then also packaged and sent as standardized SOAP/XML messages.

SOAP and XML are establishing themselves as standard protocols for business-to-business (B2B) integration. Both protocols are public standards maintained by the World-Wide Web Consortium (W3C), which do not impose any licensing constraints.

C.2 The benefits of using SOAP are given in (a) to (d).

a) SOAP is becoming the standard for B2B integration.

b) 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.

c) SOAP supports publication of services in registries (UDDI and ebXML registries) for lookup by clients.

d) There are a number of frameworks, which automate the mapping from normal programming languages onto/from SOAP. These include JAX_RPC from Sun, Axis from Apache, .Net from Microsoft and others.

C.3 With the plumbing (XML, SOAP, WSDL, UDDI, HTTP) more or less complete, the universal adoption of Web services is assured as long as those who develop the applications keep to the three tenets given in (a) to (c).

a) Systems are only loosely coupled together by nothing more than SOAP messages transmitted over HTTP or another open transport protocol (such as TCP or SMTP).

b) A service shall be described in a widely supported open interface definition language (such as WSDL).

c) If a service and client need to exchange data, the exchange shall be done in a universal data format with agreement on how data types are serialized (using XML and XML schemas, for example).

C.4 However, the primary challenge is that Web services standards are currently evolving in an emerging market and they have yet to prove the notion that Web services standards can provide interoperability. Simply put, plug-and-play interoperability among different Web service implementations is simply not a reality. There are three main reasons for these interoperability problems:

a) ambiguity among the interpretations of the standards that have already been agreed upon;

b) differences among specifications that have yet to gain widespread adoption; and

c) insufficient understanding of the interactions among specifications.

Annex C
(concluded)

Fortunately, the IT leaders behind the Web services specifications realized that interoperability is in the best interests of all industry participants. In early 2002, key industry leaders created the Web Services Interoperability Organization (WS-I), an industry organization focused on promoting Web Service interoperability across platforms, operating systems and programming languages.

C.5 Rather than creating new specifications, WS-I aims to comply with its interoperability goals through the following deliverables:

a) Web service profiles that specify collections of specifications, along with clarifications of the ambiguities, so that they can be adopted in an interoperable fashion;

b) Web services that test and implement guidance to accelerate customer deployments;

c) development and encouragement of Web service best practices, usage scenarios, use case and sample implementations that illustrate how Web service profiles can be applied to solve interoperability challenges; and

d) creation of self-administered and self-validating test suites for conformance on implementations with the profiles created by WS-I.

Annex D
(informative)

Web services security risks and countermeasures

D.1 General

This annex details common Web service threats and suggests possible countermeasures that are compliant with the Basic Profile. The countermeasures detailed here are best applied through a risk assessment of your Web service application.

The information presented here is not intended to be an exhaustive or encyclopaedic treatment of the security issues that confront Web service developers. Rather, it is designed to provide an intermediate assessment of security issues that briefly explores the intersection between traditional security issues and their manifestation in the Web service architecture.

D.2 Authentication

Authentication is a mechanism or a protocol that demonstrates proof of an asserted identity. Using an authentication mechanism, a Web service can draw conclusions about the sender of a request or a response message, and then act on the message. Many types of authentication mechanisms and protocols have been developed, including password schemes, Secure Sockets Layer (SSL), Kerberos, and public key infrastructure.

Each mechanism has advantages and limitations. With respect to interoperability, each mechanism introduces a variety of challenges. Web services can usually rely on the software platform to provide interoperable transport authentication. Authentication requirements are usually asymmetrical between service requestors and service providers. Therefore, authentication for Web Services can be further subdivided as given in D.3 and D.4.

D.3 Request authentication

D.3.1 Threat

The threats to a Web Service that do not authenticate utilities include access to data or resources by unauthorized entities, and “man-in-the-middle attacks”. In man-in-the-middle attacks, an unauthorized entity intercepts messages between requestor and responder, enabling eavesdropping and data manipulation. In very sophisticated Web service environments, a Web service provider may not be able to authenticate all parties involved in a transaction, and may therefore be required to delegate trust to other Web services.

Since Web services may rely on directory services to find providers of services (such as UDDI), authentication shall be ensured in certain processes such as consulting UDDI registries or downloading WSDL files. If authentication is not required by a directory, a relatively easy attack would be to falsify a WSDL file, causing the reliant Web services to bind to improper ports.

D.3.2 Countermeasure

A Web service should authenticate the sender of a request. Some specific situations in which Web services should authenticate requests include those in which the underlying state is changed, in which there is a charge for using the service, or where the information returned by the service is privileged.
Authentication of the service requester is the appropriate countermeasure. Client authentication can be performed using agreed upon digital certificates in the client authentication piece of an SSL/TLS exchange. The digital certificates exchanged during the SSL handshake shall chain to a certificate of authority agreed upon by both client and server.

Annex D
(continued)

D.4 Response authentication

D.4.1 Threat

An attack on a service requester is one that interposes a false or “spoofed” service that supplies responses resembling those provided by the expected service. For example, a “man-in-the-middle” might substitute a false response message for a genuine response, leading to request/response mismatches.

D.4.2 Countermeasure

Authentication of the service is the appropriate countermeasure. An SSL/TLS connection can provide server authentication and is typically sufficient protection from Web service provider spoofing for point-to-point transactions.

D.5 Authorization

Authorization is the process of determining the capabilities granted to an entity by a service provider or another trusted entity. While authentication determines which entities can access a Web service, authorization determines which features of that Web service can be accessed by the authenticated entity. In some cases even authenticated entities shall be restricted to a subset of functions provided by a Web service.

D.6 Request authorization

D.6.1 Threat

Unauthorized access to computational resources or protected data.

D.6.2 Countermeasure

Apply authorization mechanisms. Web services requests are fulfilled based on the authorization assigned to a particular requestor by the service provider. A Web service may need to communicate its authorization requirements through a policy.

A simple Web service may have one authorization level: i.e., I will execute process X for any user authenticated using a recognized token. However, more sophisticated mechanisms may be required for Web services designed to service a range of customers.

D.7 Confidentiality

D.7.1 Threat

A threat is posed by a compromise of privileged information through unauthorized access. In a messaging environment (as opposed to a session environment) it is important to evaluate the message protection characteristics of a Web service, because a Web service may not know the ultimate destination or the full route of the data being sent. Intermediaries may be traversed and if the data is unprotected, might read the confidential contents of a message, or they might be able to deduce confidential information by the mere fact that a particular message (or a message of a certain type, or messages in a certain frequency) was sent.

Annex D
(continued)

D.7.2 Countermeasure

Encryption is the primary defence against a breach of confidentiality. How encryption is applied can vary widely. The SSL/TLS protocol encrypts messages for the duration of the session. However, at each end-point, the message will be fully decrypted. An exception to this situation is SSL proxy tunnelling, in which a client proxy opens a connection to a secure server and copies data in both directions without intervening in the secure transaction.

There are ways to address the problem of end-to-end confidentiality while remaining compliant, though out of scope, of the Basic Profile. As an example, XML Encryption can be used to selectively encrypt elements or the entire message. There are many configurations, but one is to have the SOAP implementation encrypt the message payload, while leaving other information "in the clear" in the SOAP header.

D.8 Data integrity

D.8.1 Threat

Loss of data integrity is the unauthorized modification of a request or response. The threat to Web services is the malicious alteration or the accidental corruption of data.

D.8.2 Countermeasure

Messages sent using SSL/TLS have guaranteed data integrity for the duration of the exchange.

Another technique compliant, yet out of scope, with the Basic Profile is the use of digital signatures and message digests to provide proof of data integrity using XML Digital Signature. These can be applied to complete XML messages, or to portions of XML documents according to the XML Digital Signature specification.

D.9 Replay

D.9.1 Threat

A basic attack on a Web service is an attempt to re-use a once valid message. Certain elements of a Web services message, such as a security token, can also be reused as part of a different message to give the impression of a valid request or response.

D.9.2 Countermeasure

Replay attacks can be addressed by using message time stamps and caching, and through the use of universally unique identifiers on all messages.

D.10 Non-repudiation

This guarantees that both the Web service provider and customer client are protected against any claims from either party that the transaction did or did not occur at a later point in time.

D.11 Logging and auditing

One of the best countermeasures for any of the above security issues is a robust auditing/logging mechanism. In combination with authentication mechanisms, auditing and logging mechanisms can provide chains of evidence that permit runtime infractions of trust policies to be remedied by the offline trust infrastructure of business agreements and contractual law.

Annex D
(concluded)

D.12 Other risks

Like any networked application, Web services are exposed to standard network security vulnerabilities such as

a) unauthorized users who gain direct access to network resources;

b) virus or Trojan horse programs being transmitted within otherwise valid XML messages;

c) misconfiguration or improper coordination of internal resources by a Web services provider;

d) exploitation of known weaknesses; and

e) denial of service attacks.

D.13 Security standards

The security implementation shall adhere to the implemented open security protocol standards so that interoperability can be ensured.

SABS – Standards Division

The objective of the SABS Standards Division is to develop, promote and maintain South African National Standards. This objective is incorporated in the Standards Act, 2008 (Act No. 8 of 2008).

Amendments and Revisions

South African National Standards are updated by amendment or revision. Users of South African National Standards should ensure that they possess the latest amendments or editions.

The SABS continuously strives to improve the quality of its products and services and would therefore be grateful if anyone finding an inaccuracy or ambiguity while using this standard would inform the secretary of the technical committee responsible, the identity of which can be found in the foreword.
Tel: +27 (0) 12 428 6666 Fax: +27 (0) 12 428 6928

The SABS offers an individual notification service, which ensures that subscribers automatically receive notification regarding amendments and revisions to South African National Standards.
Tel: +27 (0) 12 428 6883 Fax: +27 (0) 12 428 6928 E-mail: sales@sabs.co.za

Buying Standards

Contact the Sales Office for South African and international standards, which are available in both electronic and hardcopy format.
Tel: +27 (0) 12 428 6883 Fax: +27 (0) 12 428 6928 E-mail: sales@sabs.co.za
South African National Standards are also available online from the SABS website http://www.sabs.co.za

Information on Standards

The Standards Information Centre provides a wide range of standards-related information on both national and international standards, and is the official WTO/TBT enquiry point for South Africa. The Centre also offers an individual updating service called INFOPLUS, which ensures that subscribers automatically receive notification regarding amendments to, and revisions of, international standards.
Tel: +27 (0) 12 428 6666 Fax: +27 (0) 12 428 6928 E-mail: info@sabs.co.za

Copyright

Details and advice can be obtained from the Senior Manager.
Tel: +27 (0) 12 428 6666 Fax: +27 (0) 12 428 6928 E-mail: info@sabs.co.za