SecurePay

Secure XML API Integration Guide – Card Storage, Triggered and Scheduled Payments

Document Control

This is a control document

DESCRIPTION	SECURE XML API INTEGRATION GUIDE - CARD STORAGE, TRIGGERED AND SCHEDULED PAYMENTS		
CREATION DATE	15/05/2009	CREATED BY	SECUREPAY
VERSION	1.8	DATE UPDATED	07/10/2022
CHANGES	Updated Section 4.5.1.1 and Appendix K - added Fiserv FDMSA as an acquirer supporting Visa & Mastercard Scheme Mandates - Standing Instructions (Recurring/ Instalment)		

Table of Contents

1 Introduction

1.1 About this Guide

This guide provides technical information about integrating and configuring Secure XML API – Card Storage and Scheduled Payments within your environment.

1.2 What is Secure XML API – Card Storage and Scheduled Payments

The Secure XML API – Card Storage and Scheduled Payments is a method for transmitting information to SecurePay to securely store customer details including the full card number, trigger payments, or setup payment schedules. Each XML message sent to SecurePay contains one operation.

This guide covers the process of building a program within your web site or application in order to integrate the XML API and can be run on any platform and in any programming language.

The below operations are available through the Secure XML API and detailed in this guide

Credit Card Storage Operations

- Card Storage: Add a Payor
- Card Storage: Add a Token
- Card Storage: Look up a Token
- Edit a Payor or Token
- Delete a Payor or Token

Direct Entry Storage Operations

- Direct Debit Storage: Add a Payor
- Direct Credit Storage: Add a Payee
- Edit a Payor or Payee
- Delete a Payor or Payee

Triggered Payments Features

- Trigger a Payment

Scheduled Payments

- Add a Future Payment: Once off
- Add a Scheduled Payment: Reoccurring

Once composed by your application, an XML Message is sent via the POST method to a HTTPS URL at SecurePay for processing. Once processing is complete, a response message is sent via the POST method back to your server.

In the case of a triggered payment this gives your application a real time response on the outcome of the card transaction. Direct entry payments are not processed in real time; they are stored in SecurePay's database and processed daily at 4.30pm EST.

1.3 How Card Storage and Scheduled Payments work

Secure XML API - Card Storage and Scheduled Payments processes the following payment types:

- **Storage – Payor ID:** Allows you to store a set of customer details, either Card or Bank details in the SecurePay database against a unique reference generated by the merchant. Once stored in the database the merchant can trigger a transaction against the stored details at any time. This record may also include other details such as customer name or email address.
- **Storage – Token:** Allows you to store a set of customer Card details in the SecurePay database. Once stored SecurePay will return a Token that is used to trigger transactions at any time. If the same card number is sent a second time, the same token will be returned.
- **Storage – Lookup Token:** Return an existing Token and associated information by providing an existing card or Token.
- **Triggered Payments:** Allows you to trigger a once off immediate payment against a stored set of customer details.
- **Future Payment - Once off.** Allows you to set up a payment that will only occur once on a specified future date.
- **Scheduled Payments – Reoccurring:** Allows you to set up a schedule that will occur on a regular basis. The scheduled payment occurs for the first time on the start date and then at regular intervals until the requested number of payments have been taken.

The scheduled payments will commence processing at 4.30 PM for direct entry transactions or 5.30 PM for credit card transactions on the scheduled date.

All scheduled and triggered payments will appear on the customer's bank statement with the merchant's name enabling your customers to have a record of all payments made.

Scheduled payments can be cancelled at any time via the SecurePay Merchant Login or an XML delete request when they are no longer required.

It is recommended for merchants to have signed agreements from their customers before setting up scheduled payments for credit card or direct debit

1.4 Intended Audience

This document is intended for developers, integrating SecurePay's Secure XML API – Card Storage, Triggered and Scheduled Payments interface into their own applications or websites.

It is recommended that someone with web site, XML or application programming experience reads this guide and implements the Secure XML API – Card Storage, Triggered and Scheduled Payments.

1.5 Authentication, Communication & Encryption

To ensure security, each merchant is issued with transaction password. This password is authenticated with each request before it is processed. This makes sure that unauthorised users will be unable to use the interface.

The password can be changed by the merchant via the SecurePay Merchant Login.

The Secure XML API interface uses HTTPS protocol and SSL for communication with SecurePay's servers.

SecurePay's SSL certificate will automatically encrypt requests and decrypt responses from the Secure XML API.

1.6 Feedback

Continuous improvement is one of SecurePay's core values. We welcome any feedback you have on our integration guides as a way to help us improve any future changes to our products.

If you wish to leave feedback, please click here.

2 Payment URLs

The Secure XML API – Scheduled Payments and Card Storage messages must be sent to the following URLs.

For Triggered Payment, Scheduled Payments and Card Storage (Payor):

Test URL:	https://test.api.securepay.com.au/xmlapi/periodic
Live URL:	https://api.securepay.com.au/xmlapi/periodic

For Card Storage (Token):

Test URL:	https://test.api.securepay.com.au/xmlapi/token
Live URL:	https://api.securepay.com.au/xmlapi/token

2.1 How to use

2.2 the Test Environment

As you build your system, you can test functionality when necessary by submitting your XML request to the test URL.

2.2.1 *Public Test Account Details*

You can use the below details against the SecurePay test environment.

Integration Details

This is used as part of your transaction messages.

Merchant id:	ABC0001
Transaction password:	abc123

You can log in to the Test SecurePay Merchant Login with the below details to see the outcome of your testing.

Test Login Details

This is used to log in to the Merchant Login.

URL:	https://test.login.securepay.com.au
Merchant Id:	ABC
User name:	test
Login Password:	abc1234!!

2.2.2 *Test Card Number*

Use the following information when testing transactions:

```
Card Number: 4444333322221111
Card Type:   VISA
Card CCV:    123
Card Expiry: 08 / 23 (or any date greater than today)
```

2.2.3 *Simulating Approved and Declined Transactions*

You can simulate approved and declined transactions by submitting certain payment amounts.

If the payment amount ends in 00, 08, 11 or 16, the transaction will be approved once card details are submitted. All other values will cause a declined transaction.

```
Payment amounts to simulate approved transactions:


$1.00 (100)
$1.08 (108)
$105.00 (10500)
$105.08 (10508)
(or any total ending in 00, 08, 11 or 16)
```

```
Payment amounts to simulate declined transactions:


$1.51 (151)
$1.05 (105)
$105.51 (10551)
$105.05 (10505)
(or any totals not ending in 00, 08, 11 or 16)
```

Note that when using the live URL for payments, the bank determines the transaction result, independent of the payment amount.

3 XML Message Format and Contents

3.1 TLS Support

3.1.1 *TLS Protocol version*

In line with PCI DSS v3.2 requirements, connections to SecurePay shall be configured to only use the TLS v1.2 protocol to protect data in transit. Your servers must be configured to use TLS v1.2 only. TLS 1.1, TLS 1.0 or SSL 3.0 are not supported.

3.1.2 *TLS Encryption*

Your website must be configured to use HTTPS protocol to encrypt normal HTTP requests and responses, making it safer and secure.

3.1.3 *TLS Cipher Suites*

Cipher suites specify the cryptographic algorithms that will be used in a session. Your choice of cipher suites also impacts your ability to establish a secure connection with SecurePay. It is recommended to use those cipher suites that support perfect forward secrecy. For a list of supported cipher suites see *Appendix J.*

Cipher suites which are deemed unsecure over time will no longer be supported by SecurePay but merchants will be given ample lead time to use safer alternatives.

3.2 HTTP Message Structure

The structure of the HTTP request and response messages must conform to the HTTP 1.1 network protocol standard.

The HTTP communication between the client and SecurePay Payment Server must be done via SSL so that the sensitive information included in the request and response messages is encrypted.

Your web host cannot use Server Name Indicators (SNIs) for determining which SSL certificate to serve. This is not supported by SecurePay's systems.

The HTTP header must include a Content-Type of 'text/xml'.

The XML Encoding is plain 'UTF-8' text and must not be URL encoded.

If the request includes an 'Accept' header it must include 'text/xml' as a supported content type by the client. Use of the 'Accept' header is strongly discouraged.

Note: DOCTYPE declaration in the XML request is forbidden and will be rejected by the SecurePay server.

Below is an example standard credit card payment request and response including HTTP 1.1 headers.

This message is triggering a payment against a Payor named 'test3' for the amount of $14.00 which will have a reference 'Payment Reference' recorded against the payment.

3.2.1 *Request*

HTTP Headers:
```
POST /xmlapi/payment HTTP/1.1
host: test.securepay.com.au
content-type: text/xml
content-length: 828
```

Request Body:

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
   <MessageInfo>
      <messageID>8af793f9af34bea0ecd7eff71c94d6</messageID>
      <messageTimestamp>20040710050758444000+600</messageTimestamp>
      <timeoutValue>60</timeoutValue>
      <apiVersion>spxml-3.0</apiVersion>
      </MessageInfo>
   <MerchantInfo>
      <merchantID>ABC0001</merchantID>
      <password>abc123</password>
   </MerchantInfo>
   <RequestType>Periodic</RequestType>
   <Periodic>
      <PeriodicList count="1">
         <PeriodicItem ID="1">
            <actionType>trigger</actionType>
            <transactionReference>Payment Reference</transactionReference>
            <clientID>test3</clientID>
            <amount>1400</amount>
         </PeriodicItem>
      </PeriodicList>
   </Periodic>
</SecurePayMessage>
```

3.2.2 *Response*

The initial HTTP server response (100 continue) is to indicate that the request has been received and should be ignored. The 200 response should follow with the XML response message. If content length is 0 and no XML response is included then the request could not be understood and no response was produced.

Please Note: Example message header below, this can change.

```
HTTP/1.1 100 Continue
 Server: Apache
 Date: Tue, 02 Feb 2016 01:44:36 GMT
HTTP/1.1 200 OK
Content-Type: text/xml;charset=ISO-8859-1
Date: Tue, 02 Feb 2016 01:44:39 GMT
Server: Apache
Connection: close
Content-Length: 1049

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SecurePayMessage>
  <MessageInfo>
    <messageID>8af793f9af34bea0ecd7eff71c94d6</messageID>
    <messageTimestamp>20160202124439798000+660</messageTimestamp>
    <apiVersion>spxml-3.0</apiVersion>
  </MessageInfo>
  <RequestType>Periodic</RequestType>
  <MerchantInfo>
    <merchantID>ABC0001</merchantID>
  </MerchantInfo>
  <Status>
    <statusCode>0</statusCode>
    <statusDescription>Normal</statusDescription>
  </Status>
```

```
      <Periodic>
        <PeriodicList count="1">
          <PeriodicItem ID="1">
            <actionType>trigger</actionType>
            <clientID>test3</clientID>
            <responseCode>00</responseCode>
            <responseText>Approved</responseText>
            <successful>yes</successful>
            <txnType>3</txnType>
            <amount>1400</amount>
            <currency>AUD</currency>
            <txnID>710000</txnID>
            <receipt/>
            <ponum>Payment Reference</ponum>
            <settlementDate>20160202</settlementDate>
            <CreditCardInfo>
              <pan>444433...111</pan>
              <expiryDate>09/17</expiryDate>
              <recurringFlag>no</recurringFlag>
              <cardType>6</cardType>
              <cardDescription>Visa</cardDescription>
            </CreditCardInfo>
          </PeriodicItem>
        </PeriodicList>
      </Periodic>
    </SecurePayMessage>
```

3.3 Transaction Type-Required Element Map

The table below shows which elements are required for each transaction type sent to the Triggered Payments, Scheduled Payments and Card Storage URL. Elements are mandatory, optional or not required.

ACTION TYPE / ELEMENT	Storing a Payor	Future Payment	Payment Schedule	Deleting a Payor, Future or Scheduled Payment	Triggering a Payment	Editing a Payor, Future or Scheduled Payment
	add	**add**	**add**	**delete**	**trigger**	**edit**
`<messageID>`	M	M	M	M	M	M
`<messageTimestamp>`	M	M	M	M	M	M
`<timeoutValue>`	M	M	M	M	M	M
`<apiVersion>`	M	M	M	M	M	M
`<merchantID>`	M	M	M	M	M	M
`<password>`	M	M	M	M	M	M
`<RequestType>`	M	M	M	M	M	M
`<actionType>`	M	M	M	M	M	M
`<periodicType>`	M (4)	M (1)	M (2 or 3)	X	X	X
`<standingInstructionType>`	X	X	O	X	X	X
`<clientID>`	M	M	M	M	M	M
`<amount>`	M	M	M	X	O	O
`<currency>`	O (cc)	O (cc)	O (cc)	X	X	O
`<transactionReference>`	O	X	X	X	O	X
`<paymentInterval>`	X	X	M	X	X	X
`<startDate>`	X	M	M	X	X	X
`<numberOfPayments>`	X	X	M	X	X	X
`<cardNumber>`	M (cc)	M (cc)	M (cc)	X	X	O (cc)
`<cvv>`	X	X	O(cc)	X	O (cc)	O (cc)
`<expiryDate>`	M (cc)	M (cc)	M (cc)	X	X	O (cc)
`<recurringFlag>`	O (cc)	O (cc)	O (cc)	X	O (cc)	X
`<bsbNumber>`	M (de)	M (de)	M (de)	X	X	O (de)
`<accountNumber>`	M (de)	M (de)	M (de)	X	X	O (de)
`<accountName>`	M (de)	M (de)	M (de)	X	X	O (de)
`<creditFlag>`	O (de)	O (de)	O (de)	X	X	X

M – Mandatory
O – Optional
X – Not required (ignored)
cc – credit card payments only
de – direct entry payments only

The table below shows which elements are required for each transaction type sent to the Token URL. Elements are mandatory, optional or not required.

ACTION TYPE / ELEMENT	Store a Token	Lookup a Token
	addToken	**lookupToken**
`<messageID>`	M	M
`<messageTimestamp>`	M	M
`<timeoutValue>`	M	M
`<apiVersion>`	M	M
`<merchantID>`	M	M
`<password>`	M	M
`<RequestType>`	M	M
`<amount>`	M	X
`<currency>`	O	X
`<transactionReference>`	O	X
`<tokenType>`	O	X
`<tokenValue>`	X	O
`<cardNumber>`	M	O
`<expiryDate>`	M	X
`<recurringFlag>`	O	X
M – Mandatory **O** – Optional **X** – Not required (ignored)		

3.4 Element Types and Constraints

Here are the value types and constraints which describe XML elements found in the following sections:

Type	Constraint	Description
String	A	• Alphabetic characters • Value in the element is valid if it only contains characters in the specified set (alphabetic)
	N	• Numeric characters • Value in the element is valid if it only contains characters in the specified set (numeric)
	S	• Special characters • Will be followed with a list of allowed characters • Value in the element is valid if it only contains characters in the specified set (special characters)
	EBCDIC	• EBCDIC character set • See Appendix I: EBCDIC Character Set
	LEN	• Number of characters in the string • Value in the element is valid if the length of the value is equal to the defined length
	MINLEN	• Minimum number of characters in the string • Value in the element is valid if the length of the value is greater than or equal to the defined minimum length
	MAXLEN	• Maximum number of characters in the string • Value in the element is valid if the length of the value is less than or equal to the defined maximum length

Type	Constraint	Description
Integer	DIGNO	• Number of digits in the integer value • Value in the element is valid if the number of digits in the value is less than or equal to the defined digits number
	MINVAL	• Minimum numerical value • Value in the element is valid if it is numerically greater than or equal to the defined minimum value
	MAXVAL	• Maximum numerical value • Value in the element is valid if it is numerically less than or equal to the defined maximum value

4 XML Request Message Elements

Requests are the messages sent to SecurePay. The following sections describe each XML element in detail.

4.1 XML Header

The XML document will begin with an XML declaration that contains the following data:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Markup	Usage	Explanation
`<?`	required	Begins a processing instruction.
`xml`	required	Declares this to be an XML instruction.
`version=""`	required	Identifies the version of XML specification in use.
`encoding=""`	required	Indicates which international character set is used.
`?>`	required	Terminates the processing instruction.

The XML document must contain a following top level (root) element: <SecurePayMessage>

4.2 MessageInfo Element

Description:	Identifies the message.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Sub-elements:	Yes, see table below

`<MessageInfo>` sub-elements:

Element	Comments	
`<messageID>`	**Description:**	Unique identifier for the XML message. Generated by the merchant.
	Format type:	String
	Format constraints:	AN, MINLEN = 0, MAXLEN = 30
	Validated by SecurePay:	Yes
	Value:	Eg: "8af793f9af34bea0cf40f5fb5c630c"
	Sub-elements:	No
`<messageTimestamp>`	**Description:**	Time of the request.
	Format type:	String, see Appendix E: Timestamp String Format
	Format constraints:	NS ('+', '-'), LEN = 24
	Validated by SecurePay:	Yes
	Value:	Eg: "20041803161306527000+660"
	Sub-elements:	No
`<timeoutValue>`	**Description:**	Timeout value used, in seconds.
	Format type:	Integer
	Format constraints:	DIGNO = 3, MINVAL = 1
	Validated by SecurePay:	Yes
	Value:	Recommended "60"
	Sub-elements:	No
`<apiVersion>`	**Description:**	Version of the product used.
	Format type:	String
	Format constraints:	ANS ('-', '.'), MINLEN = 1, MAXLEN = 13
	Validated by SecurePay:	Yes
	Value:	Always "spxml-3.0"
	Sub-elements:	No

4.3 MerchantInfo Element

Description:	Identifies the merchant.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Sub-elements:	Yes, see table below

`<MerchantInfo>` sub-elements:

Element	Comments	
`<merchantID>`	**Description:**	Merchant ID. 5 or 7-character merchant ID supplied by SecurePay.
	Format type:	String
	Format constraints:	AN, MINLEN = 5, MAXLEN = 7
	Validated by SecurePay:	Yes
	Value:	5-character merchant ID for Direct Entry transactions, eg: "ABC00" 7-character merchant ID for Credit Card transactions, eg: "ABC0001" Note: Tokens and Payors are stored under the three letter code (E.g. ABC) and can be accessed from any subaccount.
	Sub-elements:	No
`<password>`	**Description:**	Transaction password. Password used for authentication of the merchant's request message, supplied by SecurePay. **Note:** The password can be changed via the SecurePay Merchant Login.
	Format type:	String
	Format constraints:	ANS (All characters are allowed), MINLEN = 6, MAXLEN = 20
	Validated by SecurePay:	Yes
	Value:	Eg: "password_01"
	Sub-elements:	No

4.4 RequestType Element

Description:	Defines the type of the request being processed.
Format type:	String
Format constraints:	A, MINLEN = 1, MAXLEN = 20
Validated by SecurePay:	Yes
Value:	One of the following: • "Periodic" • "addToken" • "lookupToken" • "Echo"
Sub-elements:	No

4.5 Periodic Element

The <RequestType> will vary depending on the type of action you are taking.

Description: Contains information about financial transactions to be processed.

Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Sub-elements:	Yes, see table below

`<Periodic>` sub-elements:

Element	Comments
`<PeriodicList>`	See PeriodicList Element

4.5.1 *PeriodicList Element*

Description:	Contains list of transactions to be processed.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Attributes:	Yes, see table below
Sub-elements:	Yes, see table below

`<PeriodicList>` sub-elements:

Element	Comments	
`<PeriodicList.count>`	**Description:**	Item count is an attribute of `<PeriodicList>` element and specifies number of `<PeriodicItem>` elements. **Note:** Currently only single item per request is supported. Requests submitted with more than one `<PeriodicItem>` element will be rejected with Status code "577".
	Format type:	Integer
	Format constraints:	DIGNO = 1, MINVAL = 1, MAXVAL = 1
	Validated by SecurePay:	Yes
	Value:	Currently always "1"
	Sub-elements:	No
`<PeriodicItem>`	See PeriodicItem Element	

4.5.1.1 PeriodicItem Element

Description:	Contains information about a Payor, Future Payment, Scheduled Payment or triggered transaction request.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Attributes:	Yes, see table below
Sub-elements:	Yes, see table below

`<PeriodicItem>` sub-elements:

Not all of the `<PeriodicItem>` sub-elements are required for different types of transactions. Please refer to section Transaction Type-Required Element Map for information what elements are required for various transaction types.

Element	Comments	
`<PeriodicItem.ID>`	**Description:**	Transaction ID is an attribute of `<PeriodicItem>` element and specifies transaction ID. All transactions should be numbered sequentially starting at "1". **Note:** Currently only single transactions per request are supported. Transactions submitted with more than one `<PeriodicItem>` element will be rejected with Status code "577".
	Format type:	Integer
	Format constraints:	DIGNO = 1, MINVAL = 1, MAXVAL = 1
	Validated by SecurePay:	Yes
	Value:	Currently always "1"
	Sub-elements:	No
`<actionType>`	**Description:**	Action type specifies the type of action to be performed. Periodic and Triggered payments can either be added or deleted from SecurePay's database. Triggered payments can also be triggered.
	Format type:	String
	Format constraints:	A, MINLEN = 1, MAXLEN = 20
	Validated by SecurePay:	Yes
	Value:	One of the following: • "add" • "delete" • "trigger"
	Sub-elements:	No
`<periodicType>`	**Description:**	Periodic type specifies the type of transaction being processed.
	Format type:	Integer, see Appendix A: Periodic Types
	Format constraints:	DIGNO = 1, MINVAL = 1, MAXVAL = 4
	Validated by SecurePay:	Yes
	Value:	Eg: "1"
	Sub-elements:	No
`<standingInstructionType>`	**Description:**	Standing Instruction type specifies the type of standing instruction/payment schedule being created. To be used when <actiontype> = "add" and <periodicType> = "2" or "3". Applies to Visa and Mastercard cards only and is available on selected acquiring banks (NAB, ANZ, Westpac Qvalent and Fiserv FDMSA).
	Format type:	Integer, see Appendix K: Standing Instruction Type
	Format constraints:	DIGNO = 1, MINVAL =1, MAXVAL = 2
	Validated by SecurePay:	Yes
	Value:	Eg: "1"
	Sub-elements:	No
`<amount>`	**Description:**	Transaction amount in cents.
	Format type:	Integer
	Format constraints:	MINVAL = 1
	Validated by SecurePay:	Yes
	Value:	Eg: "123" for $1.23
	Sub-elements:	No

Element	Comments	
`<clientID>`	**Description:**	The Payor ID, Future Payment ID or Scheduled Payments ID. This must be a unique identifier, typically a Customer Reference Number. Note: This is used as the primary identifier for Payors, Future Payments and Payment Schedules. As such it must be unique across both lists for a single merchant account.
	Format type:	String
	Format constraints:	For Credit Card payments ANS (All characters allowed except spaces and "'" single quote), For Direct Entry payments EBCDIC (see Appendix I: EBCDIC Character Set), MINLEN = 1, MAXLEN = 20
	Validated by SecurePay:	Yes
	Value:	Eg: "CRN678123"
	Sub-elements:	No
`<startDate>`	**Description:**	Date indicating when the first payment should be processed for a Future Payment or Scheduled Payments.
	Format type:	String, in Format YYYYMMDD
	Format constraints:	N, LEN = 8
	Validated by SecurePay:	Yes
	Value:	Eg: "20241007" to process the first payment on October 10, 2004.
	Sub-elements:	No
`<numberOfPayments>`	**Description:**	Number of payments to occur for a particular periodic payment for Scheduled Payments.
	Format type:	Integer
	Format constraints:	MINVAL = 1
	Validated by SecurePay:	Yes
	Value:	Eg: "7" for the payment to occur 7 times
	Sub-elements:	No
`<paymentInterval>`	**Description:**	Payment interval specifies the frequency of the periodic payments for Scheduled Payments.
	Format type:	Integer. For Day Based Periodic Payments the payment interval is a number of days between payments. For Calendar Based Periodic Payments see Appendix B: Calendar Payment Intervals.
	Format constraints:	MINVAL = 1
	Validated by SecurePay:	Yes
	Value:	Eg: "1" to indicate a payment recurring every day for Day Based Periodic Payments. Eg: "1" to indicate a weekly payment for Calendar Based Periodic Payments.
	Sub-elements:	No
`<CreditCardInfo>`	See <CreditCardInfo> sub-elements.	
`<DirectEntryInfo>`	See <DirectEntryInfo> sub-elements.	

4.5.1.2 CreditCardInfo Element

Description:	Contains credit card information.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Sub-elements:	Yes, see table below

<CreditCardInfo> sub-elements:

Element	Comments	
`<cardNumber>`	**Description:**	Credit card number.
	Format type:	String
	Format constraints:	N, MINLEN = 13, MAXLEN = 16
	Validated by SecurePay:	Yes
	Value:	Eg: "4444333322221111"
	Sub-elements:	No
`<cvv>`	**Description:**	Card verification value. The CVV value assists the bank with detecting fraudulent transactions based on automatically generated card numbers, as the CVV number is printed on the physical card and cannot be generated in conjunction with a card number. If passed, the bank may check the supplied value against the value recorded against the card. See Appendix D: Location of CVV. The CVV is not stored by SecurePay.
	Format type:	String
	Format constraints:	N, MINLEN = 3, MAXLEN = 4
	Validated by SecurePay:	Yes
	Value:	Eg: "123"
	Sub-elements:	No
`<expiryDate>`	**Description:**	Credit card expiry date.
	Format type:	String
	Format constraints:	NS ('/'), LEN = 5
	Validated by SecurePay:	Yes
	Value:	Eg: "05/06" for May 2006
	Sub-elements:	No
`<recurringFlag>`	**Description:**	Marks the item as recurring. Recurring transactions are treated differently through the authorisation process. In most cases the expiry date and CVV are ignored by the bank. This allows for easy reoccurring billing, such as subscriptions. If <standingInstructionType> is provided, the recurring flag is not required and will be ignored.
	Format type:	String
	Format constraints:	A, MINLEN = 2, MAXLEN = 3
	Validated by SecurePay:	Yes
	Value:	Eg: either "yes" or "no"
	Sub-elements:	No

4.5.1.3 DirectEntryInfo Element

Description:	Contains direct entry information.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Sub-elements:	Yes, see table below

<DirectEntryInfo> sub-elements:

Element	Comments	
`<bsbNumber>`	**Description:**	BSB number.
	Format type:	String
	Format constraints:	N, LEN = 6
	Validated by SecurePay:	Yes
	Value:	Eg: "012012"
	Sub-elements:	No
`<accountNumber>`	**Description:**	Account number.
	Format type:	String
	Format constraints:	N, MINLEN = 1, MAXLEN = 9
	Validated by SecurePay:	Yes
	Value:	Eg: "00123"
	Sub-elements:	No
`<accountName>`	**Description:**	Account name.
	Format type:	String
	Format constraints:	EBCDIC (see Appendix I: EBCDIC Character Set), MINLEN = 0, MAXLEN = 32
	Validated by SecurePay:	Yes
	Value:	Eg: "John Smith"
	Sub-elements:	No
`<creditFlag>`	**Description:**	Marks the item as credit. All Direct Entry transactions are debits as a default.
	Format type:	String
	Format constraints:	A, MINLEN = 2, MAXLEN = 3
	Validated by SecurePay:	Yes
	Value:	Eg: either "yes" or "no"
	Sub-elements:	No

4.6 Token Element

Description:	Contains information about transactions to be processed.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Sub-elements:	Yes, see table below

`<Token>` sub-elements:

Element	Comments
`<TokenList>`	See TokenList Element

4.6.1 *TokenList Element*

Description:	Contains list of transactions to be processed.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Attributes:	Yes, see table below
Sub-elements:	Yes, see table below

`<TokenList>` sub-elements:

Element	Comments
`<TokenList.count>`	**Description:** Transaction count is an attribute of `<TokenList>` element and specifies number of `<TokenItem>` elements. **Note:** Currently only single transactions per request are supported. Payments submitted with more than one `<TokenItem>` element will be rejected with Status code "577". **Format type:** Integer **Format constraints:** DIGNO = 1, MINVAL = 1, MAXVAL = 1 **Validated by SecurePay:** Yes **Value:** Currently always "1" **Sub-elements:** No
`<TokenItem>`	See TokenItem Element

`<TokenItem>` sub-elements:

Not all of the <TokenItem> sub-elements are required for different transaction types. Please refer to section Required Element Map for information what elements are required for various payment types.

Elements	Comments
`<TokenItem.ID>`	**Description:** Token Item ID is an attribute of `<TokenItem>` element. All transactions should be numbered sequentially starting at "1". **Note:** Currently only single transactions per request are supported. Payments submitted with more than one `<TokenItem>` element will be rejected with Status code "577". **Format type:** Integer **Format constraints:** DIGNO = 1, MINVAL = 1, MAXVAL = 1 **Validated by SecurePay:** Yes **Value:** Currently always "1" **Sub-elements:** No
`<cardNumber>`	**Description:** Credit card number. **Format type:** String **Format constraints:** N, MINLEN = 13, MAXLEN = 16 **Validated by SecurePay:** Yes **Value:** Eg: "4444333322221111" **Sub-elements:** No
`<tokenType>`	**Description:** The type of token created. Defaults to 1 if absent. Type 1 is 16 digits, not based on the card number, failing the LUHN check **Format type:** Integer **Format constraints:** N LEN = 1 **Validated by SecurePay:** Yes **Value:** Eg: 1 **Sub-elements:** No

`<expiryDate>`	**Description:**	Credit card expiry date.
	Format type:	String
	Format constraints:	NS ('/'), LEN = 5
	Validated by SecurePay:	Yes
	Value:	Eg: "05/12" for May 2012
	Sub-elements:	No
`<tokenValue>`	**Description:**	The token value that represents a stored card within SecurePay
	Format type:	String
	Format constraints:	AN, MINLEN = 1, MAXLEN = 20
	Validated by SecurePay:	Yes
	Value:	Eg: "1234567890123456"
	Sub-elements:	No
`<transactionReference>`	**Description:**	The transaction identifier. This value will appear against all processed transactions. Typically an invoice number. E.g. "invoice12345". If absent the Token value will be used.
	Format type:	String
	Format constraints:	AN, MINLEN = 1, MAXLEN = 30
	Validated by SecurePay:	Yes
	Value:	Eg: "Customer 23"
	Sub-elements:	No
`<amount>`	**Description:**	Default amount in cents to be stored with Token. The amount can be overridden be passing an amount when triggering a payment.
	Format type:	Integer
	Format constraints:	MINVAL = 1
	Validated by SecurePay:	Yes
	Value:	Eg: "123" for $1.23
	Sub-elements:	No
`<currency>`	**Description:**	Default currency to be stored with Token. The amount can be overridden be passing an amount when triggering a payment.
	Format type:	Integer
	Format constraints:	MINVAL = 1
	Validated by SecurePay:	Yes
	Value:	Eg: "AUD" for Australian Dollars
	Sub-elements:	No

4.7 Echo Request Messages

Echo requests do not have any additional elements.

The following `<RequestType>` element value must be used for all Echo messages:

`<RequestType>Echo</RequestType>`

The Echo messages should not be sent more often than every 5 minutes and only if there were no real transactions processed in the last 5 minutes.

4.7.1 *Echo URLs*

Echo requests can be sent to any of the Payment URLs to verify if the service is available. The Status Code returned in the Echo response will be "000" if the service is up.

5 XML Response Messages Elements

Responses are the messages sent from SecurePay to the merchant in a response to a request message. Following sections describe elements common to all responses.

5.1 MessageInfo Element

Description:	Identifies the message.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Sub-elements:	Yes, see table below

`<MessageInfo>` sub-elements:

Element	Comments	
`<messageID>`	**Description:**	Unique identifier for the XML message. Returned unchanged from the request.
	Format type:	String
	Format constraints:	AN, MINLEN = 0, MAXLEN = 30
	Value:	Eg: "8af793f9af34bea0cf40f5fb5c630c"
	Sub-elements:	No
`<messageTimestamp>`	**Description:**	Time of the response.
	Format type:	String, see Appendix E: Timestamp String Format
	Format constraints:	NS ('+', '-'), LEN = 24
	Value:	Eg: "20041803161306527000+660"
	Sub-elements:	No
`<apiVersion>`	**Description:**	Version of the product used. Returned unchanged from the request.
	Format type:	String
	Format constraints:	ANS ('-', '.'), MINLEN = 1, MAXLEN = 13
	Value:	Eg: "spxml-3.0"
	Sub-elements:	No

5.2 MerchantInfo Element

Description:	Identifies the merchant.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Sub-elements:	Yes, see table below

`<MerchantInfo>` sub-elements:

Element	Comments	
`<merchantID>`	**Description:**	Merchant ID. 5 or 7-character merchant ID supplied by SecurePay. Returned unchanged from the request.
	Format type:	String
	Format constraints:	AN, MINLEN = 5, MAXLEN = 7
	Value:	5-character merchant ID for Direct Entry transactions, eg: "ABC00" 7-character merchant ID for Credit Card transactions, eg: "ABC0001"
	Sub-elements:	No

5.3 RequestType Element

Description:	Defines the type of the request being processed. Returned unchanged from the request.
Format type:	String
Format constraints:	A, MINLEN = 1, MAXLEN = 20
Value:	One of the following: • "Periodic" • "addToken" • "lookupToken" • "Echo"
Sub-elements:	No

5.4 Status Element

Description:	Status of the processing of merchant's request. Always present in the response.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Sub-elements:	Yes, see table below

`<Status>` sub-elements:

Element	Comments	
`<statusCode>`	**Description:**	Status code.
	Format type:	String, see Appendix F: SecurePay Status Codes
	Format constraints:	N, MINLEN = 1, MAXLEN = 3
	Value:	Eg: "0" or "000" for Normal
	Sub-elements:	No
`<statusDescription>`	**Description:**	Status description.
	Format type:	String, see Appendix F: SecurePay Status Codes
	Format constraints:	ANS (All characters are allowed), MINLEN = 0, MAXLEN = 40
	Value:	Eg: "Normal"
	Sub-elements:	No

5.5 Periodic Element

Following sections describe elements used in Periodic responses. The following elements will only be returned if Status received in the response is "000 – Normal" or "0 – Normal".

Description:	Contains information about transactions processed.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Sub-elements:	Yes, see table below

`<Periodic>` sub-elements:

Element	Comments
`<PeriodicList>`	See PeriodicList Element

5.5.1 *PeriodicList Element*

Description:	Contains list of transactions processed.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Attributes:	Yes, see table below
Sub-elements:	Yes, see table below

`<PeriodicList>` sub-elements:

Element	Comments
`<PeriodicList.count>`	**Description:** Transaction count is an attribute of `<PeriodicList>` element and specifies number of `<PeriodicItem>` elements. Returned unchanged from the request. **Note:** Currently only single transactions per request are supported. Transactions submitted with more than one `<PeriodicItem>` element will be rejected with Status code "577". **Format type:** Integer **Format constraints:** DIGNO = 1, MINVAL = 1, MAXVAL = 1 **Value:** Currently always "1" **Sub-elements:** No
`<PeriodicItem>`	See PeriodicItem Element

5.5.2 *PeriodicItem Element*

Description:	Contains information about a transaction.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Attributes:	Yes, see table below
Sub-elements:	Yes, see table below

`<PeriodicItem>` sub-elements:

Not all of the `<PeriodicItem>` sub-elements will be returned in a response.

Element	Comments
`<PeriodicItem.ID>`	**Description:** ID is an attribute of `<PeriodicItem>` element and specifies transaction ID. All transactions returned should be numbered sequentially starting at "1" just as they were in the request message. Returned unchanged from the request. **Note:** Currently only single transactions per request are supported. Payments submitted with more than one `<Txn>` element will be rejected with Status code "577". **Format type:** Integer **Format constraints:** DIGNO = 1, MINVAL = 1, MAXVAL = 1 **Value:** Currently always "1" **Sub-elements:** No

Element	Comments
`<actionType>`	**Description:** Action type specifies the type of action to be performed. Returned unchanged from the request. **Format type:** String **Format constraints:** A, MINLEN = 1, MAXLEN = 20 **Validated by SecurePay:** Yes **Value:** One of the following: • "add" • "delete" • "trigger" **Sub-elements:** No
`<periodicType>`	**Description:** Periodic type specifies the type of transaction being processed. Returned unchanged from the request. **Format type:** Integer, see Appendix A: Periodic Types **Format constraints:** DIGNO = 1, MINVAL = 1, MAXVAL = 4 **Validated by SecurePay:** Yes **Value:** Eg: "1" **Sub-elements:** No
`<paymentInterval>`	**Description:** Payment interval specifies the frequency of the periodic payments. Returned unchanged from the request. **Format type:** Integer. For Day Based Periodic Payments the payment interval is a number of days between payments. For Calendar Based Periodic Payments see Appendix B: Calendar Payment Intervals. **Format constraints:** MINVAL = 1 **Validated by SecurePay:** Yes **Value:** Eg: "1" to indicate a payment recurring every day for Day Based Periodic Payments. Eg: "1" to indicate a weekly payment for Calendar Based Periodic Payments. **Sub-elements:** No
`<amount>`	**Description:** Transaction amount in cents. Returned unchanged from the request. **Format type:** Integer **Format constraints:** MINVAL = 1 **Validated by SecurePay:** Yes **Value:** Eg: "123" for $1.23 **Sub-elements:** No
`<currency>`	**Description:** Transaction currency. **Note:** Only applicable to Credit Card payments. If not set a default currency is used. Default currency is "AUD" – Australian Dollars. **Format type:** String **Format constraints:** A, LEN = 3 **Validated by SecurePay:** Yes **Value:** Eg: "AUD" for Australian Dollars **Sub-elements:** No

Element	Comments	
`<clientID>`	**Description:**	Unique identifier, typically a Customer Reference Number. Returned unchanged from the request.
	Format type:	String
	Format constraints:	For Credit Card payments ANS (All characters allowed except spaces and "’" single quote), For Direct Entry payments EBCDIC (see Appendix I: EBCDIC Character Set), MINLEN = 1, MAXLEN = 20
	Validated by SecurePay:	Yes
	Value:	Eg: "CRN45187"
	Sub-elements:	No
`<startDate>`	**Description:**	Date indicating when the first payment should be processed. Returned unchanged from the request.
	Format type:	String, in Format YYYYMMDD
	Format constraints:	N, LEN = 8
	Validated by SecurePay:	Yes
	Value:	Eg: "20441007" to process the first payment on October 10, 2044.
	Sub-elements:	No
`<numberOfPayments>`	**Description:**	Number of payments to occur for a particular periodic payment. Returned unchanged from the request.
	Format type:	Integer
	Format constraints:	MINVAL = 1
	Validated by SecurePay:	Yes
	Value:	Eg: "7" for the payment to occur 7 times
	Sub-elements:	No
`<endDate>`	**Description:**	Date of the last scheduled payment.
	Format type:	String, in Format YYYYMMDD
	Format constraints:	N, LEN = 8
	Validated by SecurePay:	Yes
	Value:	Eg: "20041007" to indicate that the last payment will be processed on October 10, 2004.
	Sub-elements:	No
`<responseCode>`	**Description:**	Response code of the transaction. Either a 2-digit bank response or a 3-digit SecurePay Periodic response. Element `<responseText>` provides more information in a textual formal. Refer to the SecurePay Response Codes document via the Developer documentation link on the SecurePay website.
	Format type:	String
	Format constraints:	AN, MINLEN = 2, MAXLEN = 3
	Value:	Eg: "00"
	Sub-elements:	No
`<responseText>`	**Description:**	Textual description of the response code received.
	Format type:	String
	Format constraints:	ANS (All characters allowed), MINLEN = 0, MAXLEN = 40
	Value:	Eg: "Approved"
	Sub-elements:	No

Element	Comments	
`<successful>`	**Description:**	Indicates whether the transaction processed has been successfully added, deleted or triggered.
	Format type:	String
	Format constraints:	A, MINLEN = 2, MAXLEN = 3
	Value:	Either "yes" or "no"
	Sub-elements:	No
`<settlementDate>`	**Description:**	Bank settlement date when the funds will be settled into the merchant's account (only for triggered payments). This will be the current date mostly, however after the bank's daily cut-off time, or on non-banking days, the settlement date will be the next business day. Will not be returned if the bank did not receive the transaction. (A settlement date may be returned for declined transactions.)
	Format type:	String
	Format constraints:	N, LEN = 8
	Value:	Eg: "20040326" for 26th March 2004
	Sub-elements:	No
`<txnID>`	**Description:**	Bank transaction ID (only for triggered payments). Will not be returned if the transaction has not been processed or in some cases if it was not received by the bank.
	Format type:	String
	Format constraints:	AN, MINLEN = 6, MAXLEN = 16
	Value:	Eg: "123456"
	Sub-elements:	No
`<ponum>`	**Description:**	The transaction identifier. This value will appear in your transaction report to identify the payment after it has been processed. It is also used as part of the refund message.
	Format type:	String
	Format constraints:	AN, MINLEN = 1, MAXLEN = 30
	Value:	Eg: "Order10001"
	Sub-elements:	No
`<CreditCardInfo>`	See CreditCardInfo Element	
`<DirectEntryInfo>`	See DirectEntryInfo Element	

5.5.2.1 CreditCardInfo Element

Description:	Contains credit card information.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Sub-elements:	Yes, see table below

<CreditCardInfo> sub-elements:

Element	Comments	
`<pan>`	**Description:**	Truncated credit card number. Contains first 6 digits of the card number, followed by "..." and then last 3 digits of the card number. May not be returned for invalid transactions.
	Format type:	String
	Format constraints:	N, LEN = 12
	Value:	Eg: "444433...111"
	Sub-elements:	No

`<expiryDate>`	**Description:** **Format type:** **Format constraints:** **Value:** **Sub-elements:**	Credit card expiry date. May not be returned for invalid transactions. String NS ('/'), LEN = 5 Eg: "05/06" for May 2006 No
`<cardType>`	**Description:** **Format type:** **Format constraints:** **Value:** **Sub-elements:**	Card type used. May not be returned for invalid transactions. Integer, see Appendix C: Card Types DIGNO = 1 Eg: "6" for Visa cards No
`<cardDescription>`	**Description:** **Format type:** **Format constraints:** **Value:** **Sub-elements:**	Card description. May not be returned for invalid transactions. String, see Appendix C: Card Types A, MINLEN = 0, MAXLEN = 20 Eg: "Visa" No

5.5.2.2 DirectEntryInfo Element

Description:	Contains direct entry information.
Format type:	(No value)
Format constraints:	(No value)
Validated by SecurePay:	Yes
Value:	(No value)
Sub-elements:	Yes, see table below

<DirectEntryInfo> sub-elements:

Element	Comments	
`<bsbNumber>`	**Description:** **Format type:** **Format constraints:** **Validated by SecurePay:** **Value:** **Sub-elements:**	BSB number. May not be returned for invalid transactions. String N, LEN = 6 Yes Eg: "012012" No
`<accountNumber>`	**Description:** **Format type:** **Format constraints:** **Validated by SecurePay:** **Value:** **Sub-elements:**	Account number. May not be returned for invalid transactions. String N, MINLEN = 1, MAXLEN = 9 Yes Eg: "00123" No
`<accountName>`	**Description:** **Format type:** **Format constraints:** **Validated by SecurePay:** **Value:** **Sub-elements:**	Account name. May not be returned for invalid transactions. String EBCDIC (see Appendix I: EBCDIC Character Set), MINLEN = 0, MAXLEN = 32 Yes Eg: "John Smith" No

`<creditFlag>`	**Description:**	Marks the item as credit. All Direct Entry transactions are debits as a default. May not be returned for invalid transactions.
	Format type:	String
	Format constraints:	A, MINLEN = 2, MAXLEN = 3
	Validated by SecurePay:	Yes
	Value:	Eg: either "yes" or "no"
	Sub-elements:	No

5.6 Token Element

Following sections describe elements used in Token responses. The following elements will only be returned if Status received in the response is "000 – Normal" or "0 – Normal".

Description:	Contains information about transactions processed.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Sub-elements:	Yes, see table below

`<Token>` sub-elements:

Element	Comments
`<TokenList>`	See TokenList Element

5.6.1 *TokenList Element*

Description:	Contains list of operations processed.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Attributes:	Yes, see table below
Sub-elements:	Yes, see table below

`<TokenList>` sub-elements:

Element	Comments	
`<TokenList.count>`	**Description:**	Transaction count is an attribute of `<TokenList>` element and specifies number of `<TokenItem>` elements. Returned unchanged from the request. **Note:** Currently only single transactions per request are supported. Transactions submitted with more than one `<TokenItem>` element will be rejected with Status code "577".
	Format type:	Integer
	Format constraints:	DIGNO = 1, MINVAL = 1, MAXVAL = 1
`<TokenItem>`	See TokenItem Element	

5.2.5.2 TokenItem Element

Description:	Contains information about an operation.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Attributes:	Yes, see table below
Sub-elements:	Yes, see table below

`<TokenItem>` sub-elements:

Only relevant `<TokenItem>` sub-elements will be returned in a response.

Element	Comments
`<TokenItem.ID>`	**Description:** ID is an attribute of `<TokenItem>` element and specifies transaction ID. All transactions returned should be numbered sequentially starting at "1" just as they were in the request message. Returned unchanged from the request. **Note:** Currently only single transactions per request are supported. Requests submitted with more than one `<TokenItem>` element will be rejected with Status code "577". **Format type:** Integer **Format constraints:** DIGNO = 1, MINVAL = 1, MAXVAL = 1 **Value:** Currently always "1" **Sub-elements:** No
`<tokenValue>`	**Description:** The new token generated or previously stored token value **Format type:** String **Format constraints:** ANS (All characters allowed), MINLEN = 0, MAXLEN = 20 **Value:** Eg: "1234123412341234" **Sub-elements:** No

Element	Comments
`<responseCode>`	**Description:** Response code of the transaction. Either a 2-digit bank response (for triggered payments) or a 3-digit SecurePay Periodic response. Element `<responseText>` provides more information in a textual format. Refer to the SecurePay Response Codes document via the Developer documentation link on the SecurePay website. **Format type:** String **Format constraints:** AN, MINLEN = 2, MAXLEN = 3 **Value:** Eg: "00" **Sub-elements:** No

`<responseText>`	**Description:** Textual description of the response code received. **Format type:** String **Format constraints:** ANS (All characters allowed), MINLEN = 0, MAXLEN = 40 **Value:** Eg: "Successful" **Sub-elements:** No
`<successful>`	**Description:** Indicates whether the Token has been successfully added or looked up. **Format type:** String **Format constraints:** A, MINLEN = 2, MAXLEN = 3 **Value:** Either "yes" or "no" **Sub-elements:** No
`<transactionReference>`	**Description:** The default Transaction Reference stored with the Token. This value will appear against all processed transactions. Typically an invoice number. E.g. "invoice12345". **Format type:** String **Format constraints:** A, MINLEN = 1, MAXLEN = 30 **Validated by SecurePay:** Yes **Value:** Eg: "Customer 23" **Sub-elements:** No
`<amount>`	**Description:** For an addToken request, if an amount is provided in the request, it will be returned in this element. For a tokenLookup request, the default amount stored, if present, will be returned. **Format type:** Integer **Format constraints:** MINVAL = 1 **Validated by SecurePay:** Yes **Value:** Eg: "123" for $1.23 **Sub-elements:** No
`<CreditCardInfo>`	See CreditCardInfo Element

5.6.1.1 CreditCardInfo Element

Description:	Contains credit card information.
Format type:	(No value)
Format constraints:	(No value)
Value:	(No value)
Sub-elements:	Yes, see table below

<CreditCardInfo> sub-elements:

Element	Comments
`<pan>`	**Description:** Truncated credit card number. Contains first 6 digits of the card number, followed by "..." and then last 3 digits of the card number. May not be returned for invalid transactions. **Format type:** String **Format constraints:** N, LEN = 12 **Value:** Eg: "444433...111" **Sub-elements:** No
`<expiryDate>`	**Description:** Credit card expiry date. May not be returned for invalid transactions. **Format type:** String **Format constraints:** NS (‘/’), LEN = 5 **Value:** Eg: "05/06" for May 2006 **Sub-elements:** No

`<cardType>`	**Description:**	Card type used. May not be returned for invalid transactions.
	Format type:	Integer, see Appendix C: Card Types
	Format constraints:	DIGNO = 1
	Value:	Eg: "6" for Visa cards
	Sub-elements:	No
`<cardDescription>`	**Description:**	Card description. May not be returned for invalid transactions.
	Format type:	String, see Appendix C: Card Types
	Format constraints:	A, MINLEN = 0, MAXLEN = 20
	Value:	Eg: "Visa"
	Sub-elements:	No

5.7 Echo Response Messages

Echo responses do not return any additional elements. The `<Status>` element will return a response code "000" if the service is available.

6 Sample XML Messages

This section contains example messages for some of the more common operations through the Card Storage and Scheduled Payments interface.

All examples below are for using card details. To use direct entry, simply substitute the <CreditCardInfo> element for the <DirectEntryInfo> element.

6.1 *Card Storage*

6.1.1 *Storing a Card as a Payor*

This message will store a card in a Payor record. A Payor can be used to trigger payments through an XML message, the merchant login or a batch upload.

6.1.1.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0ecd7eff71b37ef</messageID>
  <messageTimestamp>20040710044409342000+600</messageTimestamp>
  <timeoutValue>60</timeoutValue>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <MerchantInfo>
  <merchantID>ABC0001</merchantID>
  <password>abc123</password>
 </MerchantInfo>
 <RequestType>Periodic</RequestType>
 <Periodic>
  <PeriodicList count="1">
   <PeriodicItem ID="1">
    <actionType>add</actionType>
    <clientID>test3</clientID>
    <CreditCardInfo>
     <cardNumber>4444333322221111</cardNumber>
     <cvv>123</cvv>
     <expiryDate>09/25</expiryDate>
    </CreditCardInfo>
    <amount>1100</amount>
    <periodicType>4</periodicType>
   </PeriodicItem>
  </PeriodicList>
 </Periodic>
</SecurePayMessage>
```

6.1.1.2 Response

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0ecd7eff71b37ef</messageID>
  <messageTimestamp>20040710144410220000+600</messageTimestamp>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <RequestType>Periodic</RequestType>
 <MerchantInfo>
  <merchantID>ABC0001</merchantID>
```

```
      </MerchantInfo>
      <Status>
        <statusCode>0</statusCode>
        <statusDescription>Normal</statusDescription>
      </Status>
      <Periodic>
        <PeriodicList count="1">
          <PeriodicItem ID="1">
            <actionType>add</actionType>
            <clientID>test3</clientID>
            <responseCode>00</responseCode>
            <responseText>Successful</responseText>
            <successful>yes</successful>
            <CreditCardInfo>
              <pan>444433...111</pan>
              <expiryDate>09/25</expiryDate>
              <recurringFlag>no</recurringFlag>
            </CreditCardInfo>
            <amount>1100</amount>
            <periodicType>4</periodicType>
          </PeriodicItem>
        </PeriodicList>
      </Periodic>
    </SecurePayMessage>
```

6.1.2 *Storing a Card as a Token*

This message will store a card in a Token record. This will generate a Token entry in the Payor List. A Token can be used to trigger payments through an XML message, the merchant login or a batch upload.

6.1.2.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0ecd7eff711c9d3</messageID>
  <messageTimestamp>20110710015921142000+600</messageTimestamp>
  <timeoutValue>60</timeoutValue>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <MerchantInfo>
  <merchantID>ABC00</merchantID>
  <password>abc123</password>
 </MerchantInfo>
 <RequestType>addToken</RequestType>
 <Token>
  <TokenList count="1">
   <TokenItem ID="1">
    <cardNumber>4444333322221111</cardNumber>
    <expiryDate>11/25</expiryDate>
    <tokenType>1</tokenType>
    <amount>1100</amount>
    <transactionReference>MyCustomer</transactionReference>
    </TokenItem>
  </TokenList>
 </Token>
</SecurePayMessage>
```

6.1.2.2 Response

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
    <MessageInfo>
        <messageID>8af793f9af34bea0ecd7eff711c9d3</messageID>
        <messageTimestamp>20110710115921873000+600</messageTimestamp>
        <apiVersion>spxml-3.0</apiVersion>
    </MessageInfo>
    <RequestType>addToken</RequestType>
    <MerchantInfo>
        <merchantID>ABC00</merchantID>
    </MerchantInfo>
    <Status>
        <statusCode>0</statusCode>
        <statusDescription>Normal</statusDescription>
    </Status>
    <Token>
        <TokenList count="1">
            <TokenItem ID="1">
                <responseCode>00</responseCode>
                <responseText>Successful</responseText>
                <successful>no</successful>
                <tokenValue>1234123412341234</tokenValue>
                <CreditCardInfo>
                    <pan>444433...111</pan>
                    <expiryDate>11/15</expiryDate>
                    <cardType>6</cardType>
                    <cardDescription>Visa</cardDescription>
                </CreditCardInfo>
                <amount>1100</amount>
                <transactionReference>MyCustomer</transactionReference>
            </TokenItem>
        </TokenList>
    </Token>
</SecurePayMessage>
```

6.1.3 *Look up a Token*

The lookup function will allow you to retrieve some details about a previously stored Token. The below example looks up details of a Token using the `<tokenValue>` element.

6.1.3.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
    <MessageInfo>
        <messageID>8af793f9af34bea0ecd7eff71a55f6</messageID>
        <messageTimestamp>20110710042843759000+600</messageTimestamp>
        <timeoutValue>60</timeoutValue>
        <apiVersion>spxml-3.0</apiVersion>
    </MessageInfo>
    <MerchantInfo>
        <merchantID>ABC00</merchantID>
        <password>abc123</password>
    </MerchantInfo>
    <RequestType>lookupToken</RequestType>
    <Token>
        <TokenList count="1">
            <TokenItem ID="1">
                <tokenValue>1234123412341234</tokenValue>
            </TokenItem>
        </TokenList>
    </Token>
</SecurePayMessage>
```

6.1.3.2 Response

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
    <MessageInfo>
        <messageID>8af793f9af34bea0ecd7eff711c9d3</messageID>
        <messageTimestamp>20110710115921873000+600</messageTimestamp>
        <apiVersion>spxml-3.0</apiVersion>
    </MessageInfo>
    <RequestType>addToken</RequestType>
    <MerchantInfo>
        <merchantID>ABC00</merchantID>
    </MerchantInfo>
    <Status>
        <statusCode>0</statusCode>
        <statusDescription>Normal</statusDescription>
    </Status>
    <Token>
        <TokenList count="1">
            <TokenItem ID="1">
                <responseCode>00</responseCode>
                <responseText>Successful</responseText>
                <successful>no</successful>
                <tokenValue>1234123412341234</tokenValue>
                <CreditCardInfo>
                    <pan>444433...111</pan>
                    <expiryDate>11/25</expiryDate>
                    <cardType>6</cardType>
                    <cardDescription>Visa</cardDescription>
                </CreditCardInfo>
                <amount>1100</amount>
```

```
                    <transactionReference>MyCustomer</transactionReference>
                </TokenItem>
            </TokenList>
        </Token>
    </SecurePayMessage>
```

6.2 Triggered Payments

6.2.1 *Triggering a Payment*

This message will trigger a payment against an existing Payor.

6.2.1.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
   <MessageInfo>
      <messageID>8af793f9af34bea0ecd7eff71c94d6</messageID>
      <messageTimestamp>20040710050758444000+600</messageTimestamp>
      <timeoutValue>60</timeoutValue>
      <apiVersion>spxml-3.0</apiVersion>
      </MessageInfo>
   <MerchantInfo>
      <merchantID>ABC0001</merchantID>
      <password>abc123</password>
   </MerchantInfo>
   <RequestType>Periodic</RequestType>
   <Periodic>
      <PeriodicList count="1">
         <PeriodicItem ID="1">
            <actionType>trigger</actionType>
            <transactionReference>Payment Reference</transactionReference>
            <clientID>test3</clientID>
            <amount>1400</amount>
         </PeriodicItem>
      </PeriodicList>
   </Periodic>
</SecurePayMessage>
```

6.2.1.2 Response

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SecurePayMessage>
    <MessageInfo>
        <messageID>8af793f9af34bea0ecd7eff71c94d6</messageID>
        <messageTimestamp>20162201153843948000+660</messageTimestamp>
        <apiVersion>spxml-3.0</apiVersion>
    </MessageInfo>
    <RequestType>Periodic</RequestType>
    <MerchantInfo>
        <merchantID>ABC0001</merchantID>
    </MerchantInfo>
    <Status>
        <statusCode>0</statusCode>
        <statusDescription>Normal</statusDescription>
    </Status>
    <Periodic>
        <PeriodicList count="1">
            <PeriodicItem ID="1">
                <actionType>trigger</actionType>
                <clientID>test3</clientID>
                <responseCode>00</responseCode>
                <responseText>Approved</responseText>
                <successful>yes</successful>
                <txnType>3</txnType>
                <amount>1400</amount>
```

```
                    <currency>AUD</currency>
                    <txnID>400470</txnID>
                    <receipt/>
                    <ponum>Payment Reference</ponum>
                    <settlementDate>20160122</settlementDate>
                    <CreditCardInfo>
                        <pan>444433...111</pan>
                        <expiryDate>09/17</expiryDate>
                        <recurringFlag>no</recurringFlag>
                        <cardType>6</cardType>
                        <cardDescription>Visa</cardDescription>
                    </CreditCardInfo>
                </PeriodicItem>
            </PeriodicList>
        </Periodic>
    </SecurePayMessage>
```

6.3 Scheduled Payments

Scheduled Payments are used to take transactions on specified future dates, or on a reoccurring basis. This section contains example messages for the more common operations

> *Note: Scheduled Payments can only be set up with the full customer details, not through an existing Payor or Token.*

6.3.1 *Adding a Future Payment*

A Future payment is used to set up a once off transaction which will happen on a future date.

6.3.1.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0ecd7eff711c9d3</messageID>
  <messageTimestamp>20040710015921142000+600</messageTimestamp>
  <timeoutValue>60</timeoutValue>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <MerchantInfo>
  <merchantID>ABC00</merchantID>
  <password>abc123</password>
 </MerchantInfo>
 <RequestType>Periodic</RequestType>
 <Periodic>
  <PeriodicList count="1">
   <PeriodicItem ID="1">
    <actionType>add</actionType>
    <clientID>test</clientID>
    <CreditCardInfo>
     <cardNumber>4444333322221111</cardNumber>
     <cvv>123</cvv>
     <expiryDate>09/15</expiryDate>
    </CreditCardInfo>
    <amount>1100</amount>
    <currency>AUD</currency>
    <periodicType>1</periodicType>
    <startDate>20151101</startDate>
   </PeriodicItem>
  </PeriodicList>
 </Periodic>
</SecurePayMessage>
```

6.3.1.2 Response

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SecurePayMessage>
    <MessageInfo>
        <messageID>8af793f9af34bea0ecd7eff711c9d3</messageID>
        <messageTimestamp>20162201155505312000+660</messageTimestamp>
        <apiVersion>spxml-3.0</apiVersion>
    </MessageInfo>
    <RequestType>Periodic</RequestType>
    <MerchantInfo>
        <merchantID>ABC0001</merchantID>
    </MerchantInfo>
    <Status>
```

```
            <statusCode>0</statusCode>
            <statusDescription>Normal</statusDescription>
        </Status>
        <Periodic>
            <PeriodicList count="1">
                <PeriodicItem ID="1">
                    <actionType>add</actionType>
                    <clientID>test</clientID>
                    <responseCode>00</responseCode>
                    <responseText>Successful</responseText>
                    <successful>yes</successful>
                    <CreditCardInfo>
                        <pan>444433...111</pan>
                        <expiryDate>09/15</expiryDate>
                        <recurringFlag>no</recurringFlag>
                    </CreditCardInfo>
                    <amount>1100</amount>
                    <currency>AUD</currency>
                    <periodicType>1</periodicType>
                    <paymentInterval/>
                    <numberOfPayments/>
                    <startDate>20151101</startDate>
                    <endDate>20151101</endDate>
                </PeriodicItem>
            </PeriodicList>
        </Periodic>
    </SecurePayMessage>
```

6.3.2 *Adding a Payment Schedule*

A Payment Schedule with attempt to take a payment from the customer at regular intervals.

> *Note: Payment Schedules will need to be monitored through the Merchant Login or via a daily report to ascertain the outcome of each payment. Whether the payment is approved or declined is determined by the banking network at the time of processing.*

6.3.2.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0ecd7eff71a55f6</messageID>
  <messageTimestamp>20040710042843759000+600</messageTimestamp>
  <timeoutValue>60</timeoutValue>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <MerchantInfo>
  <merchantID>ABC0001</merchantID>
  <password>abc123</password>
 </MerchantInfo>
 <RequestType>Periodic</RequestType>
 <Periodic>
  <PeriodicList count="1">
   <PeriodicItem ID="1">
    <actionType>add</actionType>
    <clientID>test2</clientID>
    <CreditCardInfo>
     <cardNumber>4444333322221111</cardNumber>
     <cvv>123</cvv>
     <expiryDate>09/15</expiryDate>
    </CreditCardInfo>
    <amount>1100</amount>
    <currency>AUD</currency>
    <periodicType>2</periodicType>
    <paymentInterval>10</paymentInterval>
    <startDate>20151101</startDate>
    <numberOfPayments>2</numberOfPayments>
   </PeriodicItem>
  </PeriodicList>
 </Periodic>
</SecurePayMessage>
```

6.3.2.2 Response

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0ecd7eff71a55f6</messageID>
  <messageTimestamp>20040710142844382000+600</messageTimestamp>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <RequestType>Periodic</RequestType>
 <MerchantInfo>
  <merchantID>ABC0001</merchantID>
 </MerchantInfo>
 <Status>
  <statusCode>0</statusCode>
  <statusDescription>Normal</statusDescription>
```

```
    </Status>
    <Periodic>
     <PeriodicList count="1">
      <PeriodicItem ID="1">
       <actionType>add</actionType>
       <clientID>test2</clientID>
       <responseCode>00</responseCode>
       <responseText>Successful</responseText>
       <successful>yes</successful>
       <CreditCardInfo>
        <pan>444433...111</pan>
        <expiryDate>15/08</expiryDate>
        <recurringFlag>no</recurringFlag>
       </CreditCardInfo>
       <amount>1100</amount>
       <currency>AUD</currency>
       <periodicType>2</periodicType>
       <paymentInterval>10</paymentInterval>
       <startDate>20041101</startDate>
       <endDate>20151111</endDate>
      </PeriodicItem>
     </PeriodicList>
    </Periodic>
   </SecurePayMessage>
```

6.3.3 *Edit a Payor, Future Payment or Scheduled Payments*

The edit function will allow you to edit some of the details against a future payment, payment schedule or Payor.

6.3.3.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
<MessageInfo>
    <messageID>fc1f5f1355800852d33ccd1d66b52</messageID>
    <messageTimestamp>20161002154123000000+660</messageTimestamp>
    <timeoutValue>60</timeoutValue>
    <apiVersion>spxml-4.2</apiVersion>
  </MessageInfo>
  <MerchantInfo>
    <merchantID>ABC00</merchantID>
    <password>abc123</password>
  </MerchantInfo>
  <RequestType>Periodic</RequestType>
  <Periodic>
    <PeriodicList count="1">
      <PeriodicItem ID="1">
        <actionType>edit</actionType>
        <clientID>TestPayor</clientID>
        <CreditCardInfo>
         <cardNumber>4444333322221111</cardNumber>
         <expiryDate>09/24</expiryDate>
        </CreditCardInfo>
      </PeriodicItem>
    </PeriodicList>
  </Periodic>
</SecurePayMessage>
```

6.3.3.2 Response

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SecurePayMessage>
  <MessageInfo>
    <messageID>fc1f5f1355800852d33ccd1d66b52</messageID>
    <messageTimestamp>20161002154447163000+660</messageTimestamp>
    <apiVersion>spxml-4.2</apiVersion>
  </MessageInfo>
  <RequestType>Periodic</RequestType>
  <MerchantInfo>
    <merchantID>ABC00</merchantID>
  </MerchantInfo>
  <Status>
    <statusCode>0</statusCode>
    <statusDescription>Normal</statusDescription>
  </Status>
  <Periodic>
    <PeriodicList count="1">
      <PeriodicItem ID="1">
        <actionType>edit</actionType>
        <clientID>TestPayor</clientID>
        <responseCode>00</responseCode>
```

```
          <responseText>Successful</responseText>
          <successful>yes</successful>
        </PeriodicItem>
      </PeriodicList>
    </Periodic>
</SecurePayMessage>
```

6.3.4 *Delete*

The delete function can be used to delete a future payment, payment schedule or mark a Payor as deleted. When a Payor is marked as deleted you will still be able to see it through the merchant login, however you will be able to re-use the Payor ID to store another set of customer details.

6.3.4.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0ecd7eff71c3ef1</messageID>
  <messageTimestamp>20040710050206632000+600</messageTimestamp>
  <timeoutValue>60</timeoutValue>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <MerchantInfo>
  <merchantID>ABC0001</merchantID>
  <password>abc123</password>
 </MerchantInfo>
 <RequestType>Periodic</RequestType>
 <Periodic>
  <PeriodicList count="1">
   <PeriodicItem ID="1">
    <actionType>delete</actionType>
    <clientID>test2</clientID>
   </PeriodicItem>
  </PeriodicList>
 </Periodic>
</SecurePayMessage>
```

6.3.4.2 Response

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0ecd7eff71c3ef1</messageID>
  <messageTimestamp>20040710150207549000+600</messageTimestamp>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <RequestType>Periodic</RequestType>
 <MerchantInfo>
  <merchantID>ABC0001</merchantID>
 </MerchantInfo>
 <Status>
  <statusCode>0</statusCode>
  <statusDescription>Normal</statusDescription>
 </Status>
 <Periodic>
  <PeriodicList count="1">
```

```
   <PeriodicItem ID="1">
    <actionType>delete</actionType>
    <clientID>test2</clientID>
    <responseCode>00</responseCode>
    <responseText>Successful</responseText>
    <successful>yes</successful>
   </PeriodicItem>
  </PeriodicList>
 </Periodic>
</SecurePayMessage>
```

6.4 Echo

Echo requests can be sent to any of the Payment URLs to verify if the service is available. The Status Code returned in the Echo response will be "000" if the service is up.

> *The Echo messages should not be sent more often than every 5 minutes and only if there were no real transactions processed in the last 5 minutes.*

6.4.1 *Request*

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0cf40f5fb79f383</messageID>
  <messageTimestamp>20042403095953349000+660</messageTimestamp>
  <timeoutValue>60</timeoutValue>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <MerchantInfo>
  <merchantID>ABC0001</merchantID>
  <password>abc123</password>
 </MerchantInfo>
 <RequestType>Echo</RequestType>
</SecurePayMessage>
```

6.4.2 *Response*

```
<?xml version="1.0" encoding="UTF-8"?>
<SecurePayMessage>
 <MessageInfo>
  <messageID>8af793f9af34bea0cf40f5fb79f383</messageID>
  <messageTimestamp>20042403095956732000+660</messageTimestamp>
  <apiVersion>spxml-3.0</apiVersion>
 </MessageInfo>
 <MerchantInfo>
  <merchantID>ABC0001</merchantID>
 </MerchantInfo>
 <RequestType>Echo</RequestType>
 <Status>
  <statusCode>000</statusCode>
  <statusDescription>Normal</statusDescription>
 </Status>
</SecurePayMessage>
```

Appendix A: Periodic Types

Periodic types define the type of transaction to be processed by SecurePay.

Code	Description
1	Once Off Payment
2	Day Based Periodic Payment
3	Calendar Based Periodic Payment
4	Add a Payor ID

Appendix B: Calendar Payment Intervals

The Periodic payment interval values in the table below apply only to payments where periodic type is type 3, Calendar Based Payment.

Code	Description
1	Weekly
2	Fortnightly
3	Monthly
4	Quarterly
5	Half Yearly
6	Annually

Appendix C: Card Types

SecurePay uses numeric codes to refer to credit card types in our system.

Code	Description
0	Unknown
1	JCB
2	American Express (Amex)
3	Diners Club
4	Bankcard
5	MasterCard
6	Visa

Appendix D: Location of CVV

The Card Verification Value is an anti-fraud measure used by some banks to prevent payments from generated card numbers. The CVV number is printed on the physical card, and is randomly assigned, therefore cannot be auto-generated.

The CVV number can be found in the following places:

Card Type	Location
Visa	Signature strip on back of card. Last digits of card number are re-printed in reverse italics, followed by 3-digit CVV.
MasterCard	Signature strip on back of card. Last digits of card number are re-printed in reverse italics, followed by 3-digit CVV.
Bankcard	Signature strip on back of card. Last digits of card number are re-printed in reverse italics, followed by 3-digit CVV.
Amex	4 digit CVV above card number on front of card.
Diners Club	Signature strip on back of card. Last digits of card number are re-printed in reverse italics, followed by 3-digit CVV.
JCB	Not used

Appendix E: Timestamp String Format

The format of the Timestamp or Log Time strings returned by SecurePay xmlapi is:

YYYYDDMMHHNNSSKKK000sOOO

where:

- YYYY is a 4-digit year
- DD is a 2-digit zero-padded day of month
- MM is a 2-digit zero-padded month of year (January = 01)
- HH is a 2-digit zero-padded hour of day in 24-hour clock format (midnight =0)
- NN is a 2-digit zero-padded minute of hour
- SS is a 2-digit zero-padded second of minute
- KKK is a 3-digit zero-padded millisecond of second
- 000 is a Static 0 characters, as SecurePay does not store nanoseconds
- sOOO is a Time zone offset, where s is "+" or "-", and OOO = minutes, from GMT.

E.g. June 24, 2002 5:12:16.789 PM, Australian EST is:

20022406171216789000+600

Appendix F: SecurePay Status Codes

For a full list of SecurePay's response codes, please refer to the SecurePay Response Codes document via the Developer Documentation page on the SecurePay website.

Appendix G: XML Request DTD

```
<!ELEMENT SecurePayMessage (MessageInfo, RequestType, MerchantInfo, Periodic?,
Token?)>

<!-- define elements for SecurePayMessage -->
<!ELEMENT MessageInfo (messageID, messageTimestamp, apiVersion)>
<!ELEMENT RequestType (#PCDATA)>
<!ELEMENT MerchantInfo (merchantID)>
<!ELEMENT Periodic (PeriodicList)>

<!-- define elements for MessageInfo -->
<!ELEMENT messageID (#PCDATA)>
<!ELEMENT messageTimestamp (#PCDATA)>
<!ELEMENT apiVersion (#PCDATA)>

<!-- define elements for MerchantInfo -->
<!ELEMENT merchantID (#PCDATA)>

<!-- define elements for Periodic -->
<!ELEMENT PeriodicList (PeriodicItem*)>
<!ATTLIST PeriodicList
    count CDATA #REQUIRED>

<!-- define elements for PeriodicList -->
<!ELEMENT PeriodicItem (actionType, clientID, CreditCardInfo?, DirectEntryInfo?,
    amount?,    periodicType?,    standingInstructionType?,    paymentInterval?,
startDate?, numberOfPayments?)>
<!ATTLIST PeriodicItem ID CDATA #REQUIRED>

<!-- define elements for PeriodicItem -->
<!ELEMENT actionType (#PCDATA)>
<!ELEMENT clientID (#PCDATA)>
<!ELEMENT   CreditCardInfo   (cardNumber,   cvv?,   expiryDate,   recurringFlag?,
transactionReference?)>
<!ELEMENT DirectEntryInfo (bsbNumber, accountNumber, accountName, creditFlag?)>
<!ELEMENT amount (#PCDATA)>
<!ELEMENT currency (#PCDATA)>
<!ELEMENT periodicType (#PCDATA)>
<!ELEMENT standingInstructionType (#PCDATA)>
<!ELEMENT paymentInterval (#PCDATA)>
<!ELEMENT startDate (#PCDATA)>
<!ELEMENT numberOfPayments (#PCDATA)>

<!-- define elements for CreditCardInfo -->
<!ELEMENT cardNumber (#PCDATA)>
<!ELEMENT cvv (#PCDATA)>
<!ELEMENT expiryDate (#PCDATA)>
<!ELEMENT recurringFlag (#PCDATA)>
<!ELEMENT transactionReference (#PCDATA)>


<!-- define elements for DirectEntryInfo -->
<!ELEMENT bsbNumber (#PCDATA)>
<!ELEMENT accountNumber (#PCDATA)>
<!ELEMENT accountName (#PCDATA)>
<!ELEMENT creditFlag (#PCDATA)>
```

```
<!-- define elements for Token-->
<!ELEMENT Token (TokenList)>
<!ATTLIST TokenItem ID NMTOKEN #REQUIRED>
<!ELEMENT TokenList (TokenItem)>
<!ATTLIST TokenList count NMTOKEN #REQUIRED>
<!ELEMENT   TokenItem   (cardNumber?,   expiryDate?,   tokenType?,   amount?,
transactionReference?, tokenValue?)>
<!ELEMENT amount (#PCDATA)>
<!ELEMENT cardNumber (#PCDATA)>
<!ELEMENT expiryDate (#PCDATA)>
<!ELEMENT tokenType (#PCDATA)>
<!ELEMENT tokenValue (#PCDATA)>
<!ELEMENT transactionReference (#PCDATA)>
```

Appendix H: XML Response DTD

```
<!ELEMENT  SecurePayMessage  (MessageInfo,  RequestType,  MerchantInfo,  Status,
Periodic?, Token?)>

<!-- define elements for SecurePayMessage -->
<!ELEMENT MessageInfo (messageID, messageTimestamp, apiVersion)>
<!ELEMENT RequestType (#PCDATA)>
<!ELEMENT MerchantInfo (merchantID)>
<!ELEMENT Status (statusCode, statusDescription)>
<!ELEMENT Periodic (PeriodicList)>

<!-- define elements for MessageInfo -->
<!ELEMENT messageID (#PCDATA)>
<!ELEMENT messageTimestamp (#PCDATA)>
<!ELEMENT apiVersion (#PCDATA)>

<!-- define elements for MerchantInfo -->
<!ELEMENT merchantID (#PCDATA)>

<!-- define elements for Status -->
<!ELEMENT statusCode (#PCDATA)>
<!ELEMENT statusDescription (#PCDATA)>

<!-- define elements for Periodic -->
<!ELEMENT PeriodicList (PeriodicItem*)>
<!ATTLIST PeriodicList
    count CDATA #REQUIRED>

<!-- define elements for PeriodicList -->
<!ELEMENT PeriodicItem (actionType, clientID, CreditCardInfo?, DirectEntryInfo?,
    amount?,    periodicType?,    standingInstructionType,    paymentInterval?,
startDate?, endDate?, numberOfPayments?,
    responseCode, responseText, successful, settlementDate?, txnID?)>
<!ATTLIST PeriodicItem ID CDATA #REQUIRED>

<!-- define elements for PeriodicItem -->
<!ELEMENT actionType (#PCDATA)>
<!ELEMENT clientID (#PCDATA)>
<!ELEMENT   CreditCardInfo   (pan,   expiryDate,   recurringFlag?,   cardType?,
cardDescription?)>
<!ELEMENT DirectEntryInfo (bsbNumber, accountNumber, accountName, creditFlag?)>
<!ELEMENT amount (#PCDATA)>
<!ELEMENT periodicType (#PCDATA)>
<!ELEMENT standingInstructionType (#PCDATA)>
<!ELEMENT paymentInterval (#PCDATA)>
<!ELEMENT startDate (#PCDATA)>
<!ELEMENT endDate (#PCDATA)>
<!ELEMENT numberOfPayments (#PCDATA)>
<!ELEMENT responseCode (#PCDATA)>
<!ELEMENT responseText (#PCDATA)>
<!ELEMENT successful (#PCDATA)>
<!ELEMENT settlementDate (#PCDATA)>
<!ELEMENT txnID (#PCDATA)>

<!-- define elements for CreditCardInfo -->
<!ELEMENT pan (#PCDATA)>
<!ELEMENT expiryDate (#PCDATA)>
```

```
<!ELEMENT recurringFlag (#PCDATA)>
<!ELEMENT cardType (#PCDATA)>
<!ELEMENT cardDescription (#PCDATA)>
<!ELEMENT transactionReference (#PCDATA)>

<!-- define elements for DirectEntryInfo -->
<!ELEMENT bsbNumber (#PCDATA)>
<!ELEMENT accountNumber (#PCDATA)>
<!ELEMENT accountName (#PCDATA)>
<!ELEMENT creditFlag (#PCDATA)>

<!-- define elements for Token -->
<!ELEMENT Token (TokenList) >
<!ELEMENT   TokenItem   (responseCode,   responseText,   successful,   tokenValue,
CreditCardInfo, amount, transactionReference)>
<!ATTLIST TokenItem ID NMTOKEN #REQUIRED>
<!ELEMENT TokenList (TokenItem )>
<!ATTLIST TokenList count NMTOKEN #REQUIRED>
<!ELEMENT amount (#PCDATA)>
<!ELEMENT tokenValue (#PCDATA)>
```

Appendix I: EBCDIC Character Set

Description	Characters allowed
Numeric	0 - 9
Alphabetic	a – z, A - Z
Oblique slash	/
Hyphen	-
Ampersand	&
Period	.
Asterisk	*
Apostrophe	‘
Blank space	

Appendix J: TLS Cipher Suites

The following cipher suites are accepted for TLS 1.2:

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CCM
TLS_DHE_RSA_WITH_AES_256_CCM
TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8
TLS_DHE_RSA_WITH_AES_256_CCM_8
TLS_ECDHE_ECDSA_WITH_AES_128_CCM
TLS_DHE_RSA_WITH_AES_128_CCM
TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8
TLS_DHE_RSA_WITH_AES_128_CCM_8

Appendix K: Standing Instruction Type

The Standing Instruction type is used by SecurePay to identify the payment schedule as either recurring or instalment. The values in the table below can be used when creating a new payment schedule, when the:

- action type is "add"
- periodic type is "2" (Day Based Payment) or "3" (Calendar Based Payment)
- card type is "5" (Mastercard) or "6" (Visa).

The standing instruction type (recurring or instalment) supports Visa & Mastercard Scheme Mandates and allows SecurePay to provide the required parameters to the acquirer/scheme. If a standing instruction type is provided in the request, an Account Verification will be performed to verify the customer's card prior to saving the schedule. If successful, the Account Verification becomes the initial/first transaction in the standing instruction series. The subsequent scheduled payments are linked to the initial transaction via a standing instruction ID and are visible in the Merchant Portal.

NOTE: This is available on selected acquirers (NAB, ANZ, Westpac Qvalent and Fiserv FDMSA).

Code	Description
1	Recurring *When the card is being used for recurring charges e.g. fortnightly gym membership*
2	Instalment *When the card is being used to pay instalments on a single purchase e.g. pay four $100 monthly payments for a $400 sofa*

Merchants must have cardholder consent (e.g. a signed agreement) before setting up a payment schedule.