# FXAUTOPAY Web Service Specification

| Version | Date | Author | Revision Details |
|---|---|---|---|
| 1.0 | 2 July 2010 | Sean Thompson | Initial version. |
| 1.1 | 6 July 2010 | Paul Sullivan | Adjusted to include demo login details and location of reference implementation. |
| 1.2 | 23 August 2010 | Sean Thompson | Added sell currency to payment spec; clarified the 'amount' description in payments. |
| 1.3 | 9 September 2010 | Sean Thompson | Addition of Error Codes and Past payments. Updated schemas to reflect changes. Formatting. |

## Table of Contents

FXAUTOPAY – WEB SERVICE SPECIFICATION ..... 3
Introduction ..... 3
Obtaining Access ..... 3
Requests ..... 3
Authenticating The Request ..... 3
HTTP_AUTH ..... 4
Token Authentication ..... 4
Request Format ..... 4
Requesting a Quote ..... 5
Making Payments ..... 7
Sample Payment ..... 12
Re-requesting Past Payments ..... 13
The Response Object ..... 14
Response Complex Element Descriptions ..... 16
Quote Response Element ..... 16
Error Handling ..... 19
Error Codes ..... 19

# List of Figures

Figure 1: Request Schema ..... 5
Figure 2: Get Quote Sub-request ..... 6
Figure 3: Sample Quote Request ..... 7
Figure 4: Payment Sub-request ..... 9
Figure 5: Beneficiary Object ..... 12
Figure 6: Sample Payment ..... 13
Figure 7: Past Payment Request Example ..... 13
Figure 8: Past Payment Example Response ..... 14
Figure 9: Response Schema ..... 15
Figure 10: Failed Quote Response ..... 15
Figure 11: Successful Quote Response ..... 16
Figure 12: Quote Response Schema ..... 17
Figure 13: Payment Response Schema ..... 18
Figure 14: Response Past-Payment Schema ..... 18
Figure 15: Error Response Schema ..... 19
Figure 16: Response Example - Bad Payment ..... 19

# List of Tables

Table 1: Elements Table Tag Definitions ..... 3
Table 2 Request Elements ..... 4
Table 3: Quote Request Elements Description ..... 5
Table 4: Payment Elements Description ..... 8
Table 5: Beneficiary Elements Description ..... 10
Table 6: past payment definition ..... 13
Table 7: Response Elements Definition ..... 14
Table 8: Error Codes ..... 21

# FXAUTOPAY – WEB SERVICE SPECIFICATION

## Introduction

This document describes the technical specifications of the World First FXAUTOPAY API web service.

The invocation of FXAUTOPAY involves the sending and receiving of XML messages to/from a web service via HTTPS.

The messages are described in this document via XML schemas, sample XML files and a message elements table that details the usage of each element in the request. The codes used in the elements table for usage are defined as shown in Table 1.

| Code | Description |
|---|---|
| M | Mandatory |
| O | Optional |
| C | Conditional (May be mandatory under certain conditions) |

**Table 1: Elements Table Tag Definitions**

## Obtaining Access

Before FXAUTOPAY can be accessed, an account must first be set up for clients of the API including a username and password, also a secret key must be set and exchanged between the parties in order to create the required tokens that are passed with each API request.

Once these pre-requisites have been met, the API can be accessed at:

https://trading.worldfirst.com/api/demo/

Sample PHP, C# and JAVA interface programs as well as schemas and test data are provided at:

https://trading.worldfirst.com/api/samples/

## Requests

### Authenticating The Request

Every request sent to FXAUTOPAY is authenticated on two levels: HTTP_AUTH and hash token authentication.

## HTTP_AUTH

HTTP_AUTH authentication is performed by Apache and should be included in the request header. A username and password for each client will be supplied by World First, to be used in HTTP_AUTH authentication.

| | |
|---|---|
| Username: | demo |
| Password: | d3m0u5r |

## Token Authentication

Once HTTP_AUTH authentication of the client has passed, each request will contain a token to be validated using a secret hashing key provided by World First to the client. Sending of an invalid token will result in the request packet not being processed. The token must be sent with each request to the FXAUTOPAY API.

The algorithm to create the token is as follows

1. Create an MD5 hash of the request XML packet. Note: create the MD5 hash of only the XML packet containing the request, not the entire request itself.
2. Use HMAC (Hash-based Message Authentication Code) to create a HMAC MD5 hash of the MD5 hash you just created, using the secret key.

Append the complete hash token to the URL in the request in the form of hash=token. Eg: https://trading.worldfirst.com/api/demo/?hash=[token] where [token] is the token from the HMAC hash you have just created.

| | |
|---|---|
| Secret Key: | abcdefghihjklmnopqrstuvwxyz0123456789 |

## Request Format

Every request sent to FXAUTOPAY must be in XML format and follow the schema shown in Figure 1.

| Tag | Type | Description | Usage |
|---|---|---|---|
| atomic | boolean | If true, tells World First to execute all sub-requests (payments and quote requests) in the packet or none. i.e. if one sub-request fails, kill the entire request. | M |
| testing | boolean | If true, indicates that the request is a “test” packet, i.e. not to be executed in the market. | M |
| getquote | getquote | Sub-request object for a rate on a particular symbol. Can include multiple instances of this object. See Figure 2. | O |
| payment | payment | Sub-request object for a payment on a quoted rate, or a “trade implied” payment. Can include multiple instances of this object. See Figure 4. | O |
| past-paymentid | string | The payment ID of a previously processed payment. | O |

**Table 2 Request Elements**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<xs:include schemaLocation="payment.xsd"/>
<xs:include schemaLocation="getquote.xsd"/>

<xs:element name="request">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="atomic" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="testing" type="xs:boolean" minOccurs="1" maxOccurs="1"/>

            <xs:element name="getquote" type="getquote" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="payment" type="payment" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

**Figure 1: Request Schema**

## Requesting a Quote

Each request may contain zero to *n* number of quote requests. Each quote request must take the format shown in Table 3.

| Tag | Type | Description | Usage |
|---|---|---|---|
| settlementdate | date (YYYY-MM-DD) | The value date of the trade you wish to book for. Must be a future date. | M |
| burcurr | string | The 3 letter code of the currency you wish to buy. | M |
| sellcurr | string | The 3 letter code of the currency you wish to pay World First. | M |
| side | string | Either 'B' or 'S'. This indicates which currency you are specifying the amount in, the buy ('B') or sell ('S') currency. | M |
| amount | decimal | The amount of the specified currency you want to trade. The decimal places may be omitted if trading JPY. | M |

**Table 3: Quote Request Elements Description**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:complexType name="getquote">
        <xs:sequence>
            <xs:element name="settlementdate" type="xs:date" minOccurs="1" maxOccurs="1"/>

            <xs:element name="buycurr" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="sellcurr" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="side" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="B"/>
                        <xs:enumeration value="S"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="amount" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:decimal">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

**Figure 2: Get Quote Sub-request**

The sample request XML file shown in Figure 3 depicts a single quote request, selling GBP to World First and buying 10,000 Euros:

```
<?xml version="1.0"?>
<request>
    <atomic>false</atomic>
    <testing>true</testing>

    <getquote>
        <settlementdate>2010-08-11</settlementdate>
        <buycurr>EUR</buycurr>
        <sellcurr>GBP</sellcurr>
        <side>B</side>
        <amount>10000</amount>
    </getquote>
</request>
```

**Figure 3: Sample Quote Request**

## Making Payments

| Tag | Type | Description | Usage |
|---|---|---|---|
| tradeid | string | The quote ID for a previously quoted trade. If this field is not set, the payment will be treated as “trade implied”. i.e. A rate will be booked automatically to cover the payment. | C |
| paymentid | string | Client’s unique identifier for the payment. | M |
| sellcurrency | string | Currency to pay World First. If not provided it will default to the client’s stored base currency. | O |
| amount | string | The amount to pay to the beneficiary in the beneficiarie’s currency. This will also be the same as the BUY currency. | M |
| paymentdate | date (YYYY-MM-DD) | The date you wish the payment to be paid on. | M |
| paymenttype | decimal | Payment type eg: SEPA, BACS, CHAPS etc... | M |
| notes1 | string | Any notes you wish to appear on the payment record. | O |
| notes2 | string | Same as notes1. | O |
| notes3 | string | Same as notes1. | O |
| reason | integer | ID of the reason this payment is being made from the following:<br>1: Emigration<br>3: Overseas Mortgage Payments<br>4: Sending Money Home<br>5: Transfer to Own Account<br>6: Other<br>7: Paying Overseas Suppliers<br>8: Repatriating Overseas Earnings | M |

| | | | |
|---|---|---|---|
| | | 9: Investing Abroad<br>16: Holiday Home/Second Home Purchase<br>17: Investment Property Purchase<br>18: Overseas Purchase<br>19: Property Sale<br>20: Returning From Abroad | |
| reason_if_other | string | Include only if reason is “other”. | C |
| beneficiary | beneficiary | Must include one and only one beneficiary object. See Figure 5. | M |

**Table 4: Payment Elements Description**

The payments schema is defined in Figure 4.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<xs:include schemaLocation="beneficiary.xsd"/>

    <xs:complexType name="payment">
        <xs:sequence>
            <xs:element name="tradeid" type="xs:string" minOccurs="0" maxOccurs="1"/>

            <xs:element name="paymentid" type="xs:string" minOccurs="1" maxOccurs="1"/>

            <xs:element name="sellcurrency" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="amount" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:decimal">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="paymentdate" type="xs:date" minOccurs="1" maxOccurs="1"/>
            <xs:element name="paymenttype" type="xs:string" minOccurs="1" maxOccurs="1"/>

            <xs:element name="notes1" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="notes2" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="notes3" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

```
            <xs:element name="reason" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:integer">
                        <xs:enumeration value="1"/>
                        <xs:enumeration value="3"/>
                        <xs:enumeration value="4"/>
                        <xs:enumeration value="5"/>
                        <xs:enumeration value="6"/>
                        <xs:enumeration value="7"/>
                        <xs:enumeration value="8"/>
                        <xs:enumeration value="9"/>
                        <xs:enumeration value="16"/>
                        <xs:enumeration value="17"/>
                        <xs:enumeration value="18"/>
                        <xs:enumeration value="19"/>
                        <xs:enumeration value="20"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="reason_if_other" type="xs:string" minOccurs="0" maxOccurs="1"/>

            <xs:element name="beneficiary" type="beneficiary" minOccurs="1" maxOccurs="1"/>

        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

**Figure 4: Payment Sub-request**

A beneficiary must be included in the payment to indicate who to make the payment to. The beneficiary object is detailed in Table 5.

| Tag | Type | Description | Usage |
|---|---|---|---|
| name | string | The name of the beneficiary/account holder. | M |
| accholderadd1 | string | Address of the beneficiary | O |
| accholderadd2 | string | Same as accholderadd1. | O |
| accholderadd3 | string | Same as accholderadd1. | O |
| curr | string | The 3 letter currency code for the currency to be paid into this account. | M |
| bankname | string | Name of the beneficiarie's bank. | M |
| bankcode | string | Bank Code/Sort Code. Subject to beneficiary rules to determine whether required or not. | C |
| accno | string | Bank account number of the beneficiary. Subject to beneficiary rules to determine whether required or not. | C |
| bankadd1 | string | Bank address. | O |
| bankadd2 | string | Same as bankadd1. | O |
| bankadd3 | string | Same as bankadd1. | O |
| bankcountry | string | Country this bank account is held in. This is an international standard 2 letter country code. | M |
| iban | string | IBAN identifier for this account. Subject to beneficiary rules to determine whether required or not. | C |
| bic | string | BIC identifier for this account. Subject to beneficiary rules to determine whether required or not. | C |

**Table 5: Beneficiary Elements Description**

The beneficiary schema contained within the payment is defined in Figure 5.

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:complexType name="beneficiary">
        <xs:sequence>
            <xs:element name="accholder" type="xs:string" minOccurs="1" maxOccurs="1"/>

            <xs:element name="accholderadd1" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="35"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
```

```
<xs:element name="accholderadd2" minOccurs="0" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="35"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element name="accholderadd3" minOccurs="0" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="35"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element name="curr" minOccurs="1" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:pattern value="[a-zA-Z]{3}"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element name="bankname" minOccurs="1" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:minLength value="1"/>
            <xs:maxLength value="35"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element name="bankcode" type="xs:string" minOccurs="0" maxOccurs="1"/>
<xs:element name="accno" type="xs:string" minOccurs="0" maxOccurs="1"/>

<xs:element name="bankadd1" minOccurs="0" maxOccurs="1">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:maxLength value="35"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>

<xs:element name="bankadd2" minOccurs="0" maxOccurs="1">
```

```
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="35"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="bankadd3" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:maxLength value="35"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="bankcountry" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{2}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="iban" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="bic" type="xs:string" minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>

</xs:schema>
```

**Figure 5: Beneficiary Object**

## Sample Payment

```
<?xml version="1.0"?>
<request>
    <atomic>false</atomic>
    <testing>false</testing>

    <payment>
        <paymentid>nontrade_payment1</paymentid>
        <sellcurrency>GBP</sellcurrency>
        <amount>2000.00</amount>
        <paymentdate>2010-08-11</paymentdate>
        <paymenttype>SEPA</paymenttype>
        <notes1>note</notes1>
        <notes2>note</notes2>
```

```
        <notes3>note</notes3>
        <reason>6</reason>
        <reason_if_other>some stuff</reason_if_other>

        <beneficiary>
            <accholder>Sean Thompson</accholder>
            <accholderadd1>18 lombard road</accholderadd1>
            <accholderadd2>london sw11</accholderadd2>
            <accholderadd3></accholderadd3>
            <curr>EUR</curr>
            <bankname>Santander</bankname>
            <bankcode>13-24-56</bankcode>
            <accno>11112222</accno>
            <bankadd1>999 lombard street</bankadd1>
            <bankadd2>Barcelona</bankadd2>
            <bankadd3></bankadd3>
            <bankcountry>ES</bankcountry>
            <iban>ES9121000418450200051332</iban>
            <bic>WFSTGB01</bic>
        </beneficiary>
    </payment>
</request>
```

**Figure 6: Sample Payment**

## Re-requesting Past Payments

If you need to request a previously processed payment, this can be done by using the past-payment request element. All that is required in the request is the **paymentid** of the payment originally sent to FX AUTO PAY (this is the paymentid field specified in Table 4). This is the ID that you provided to ID this payment. The number of past-payment elements in the request that can be included is presently unlimited.

| Tag | Type | Description | Usage |
|---|---|---|---|
| past-paymentid | string | The payment ID of a previously processed payment. | O |

**Table 6: past payment definition**

A typical request for past payments may look something like this:

```
<?xml version="1.0"?>
<request>
    <atomic>false</atomic>
    <testing>false</testing>
    <past-paymentid>nontrade_payment1</past-paymentid>
    <past-paymentid>test_no_payment</past-paymentid>
</request>
```

**Figure 7: Past Payment Request Example**

The response will be either the information of any payments matching the provided paymentid, or an error message notifying that the payment was not found, as depicted in Figure 8.

```
<?xml version="1.0"?>
<response>
    <atomic>false</atomic>
    <testing>false</testing>

    <past-payment>
        <requested-paymentid>nontrade_payment1</requested-paymentid>
        <success>true</success>
        <created>2010-09-06T17:08:40Z</created>
        <tradeid>FXAUTOPAY_14c8512065022e</tradeid>
        <buycurr>EUR</buycurr>
        <sellcurr>AUD</sellcurr>
        <amount>2000.00</amount>
        <rate>1.406766</rate>
        <paymentdate>2010-09-09</paymentdate>
    </past-payment>

    <past-payment>
        <requested-paymentid>test_no_payment</requested-paymentid>
        <success>false</success>
        <error>
            <code>500</code>
            <message>Payment not found for: test_no_payment</message>
        </error>
    </past-payment>

</response>
```

**Figure 8: Past Payment Example Response**

## The Response Object

The elements that may be included in a response packet are defined in Table 7.

| Tag | Type | Description | Usage |
|---|---|---|---|
| atomic | string | Indicates how the corresponding request was handled. Will take the value passed in the request it is responding to. | C |
| testing | string | Indicates whether the request was treated as a test request or a live one. Will take the value passed in the request it is responding to. Note: If the client is flagged for testing on World First systems, the packet will be treated as “testing” regardless of what value was sent in the initial request. | C |
| error | error | Any errors that may have occurred in high level packet processing. | C |
| quote | quote | Response details of any <getquote>s in the client request. | C |
| payment | payment | Response details of any <payment>s in the client request. | C |
| past-payment | past-payment | Response details of any <past-paymentid>s in the client request. | C |

**Table 7: Response Elements Definition**

The XML schema that governs the responses FXAUTOPAY can send back is depicted in Figure 9.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

<xs:include schemaLocation="response_payment.xsd"/>
<xs:include schemaLocation="response_quote.xsd"/>

<xs:element name="response">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="atomic" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="testing" type="xs:boolean" minOccurs="0" maxOccurs="1"/>
            <xs:element name="error" type="error " minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="quote" type="quote" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="payment" type="payment" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

**Figure 9: Response Schema**

```
<?xml version="1.0"?>
<response>
    <atomic>false</atomic>
    <testing>false</testing>
    <quote>
        <success>false</success>
        <buycurr>EUR</buycurr>
        <sellcurr>GBP</sellcurr>
        <settlementdate>2010-09-02</settlementdate>
        <side>B</side>
        <amount>10000</amount>
        <error>
            <code>301</code>
            <message>Value Date must be in the future</message>
        </error>
    </quote>
</response>
```

**Figure 10: Failed Quote Response**

```
<?xml version="1.0"?>
<response>
    <atomic>false</atomic>
    <testing>false</testing>
    <quote>
        <success>true</success>
```

```
        <tradeid>API_14c2e093f54251</tradeid>
        <buycurr>EUR</buycurr>
        <sellcurr>GBP</sellcurr>
        <buyamt>10000</buyamt>
        <sellamt>8288.97</sellamt>
        <rate>0.828897</rate>
        <settlementdate>2010-08-11</settlementdate>
        <expiry>1278086040</expiry>
    </quote>
</response>
```

**Figure 11: Successful Quote Response**

## Response Complex Element Descriptions

As shown in Table 7 and the response schema, there are complex elements that may be sent in the response. These types are: quote, payment and past-payment. The schemas for all these elements are given here.

Although errors are also a complex element, error responses are explained in the following section.

### Quote Response Element

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:complexType name="quote">
        <xs:sequence>
            <xs:element name="success" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="tradeid" type="xs:string" minOccurs="0" maxOccurs="1"/>


            <xs:element name="buycurr" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="sellcurr" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="buyamt" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:decimal">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="sellamt" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
```

```
                    <xs:restriction base="xs:decimal">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="rate" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="settlementdate" type="xs:date" minOccurs="0" maxOccurs="1"/>
            <xs:element name="expiry" type="xs:string" minOccurs="0" maxOccurs="1"/>

            <xs:element name="side" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="B"/>
                        <xs:enumeration value="S"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="amount" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="error" type="error" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

**Figure 12: Quote Response Schema**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:complexType name="payment">
        <xs:sequence>
            <xs:element name="success" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="tradeid" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="paymentid" type="xs:string" minOccurs="1" maxOccurs="1"/>

            <xs:element name="buycurr" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="sellcurr" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="amount" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:decimal">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
```

```
            <xs:element name="rate" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="paymentdate" type="xs:date" minOccurs="0" maxOccurs="1"/>

            <xs:element name="error" type="error" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

**Figure 13: Payment Response Schema**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

    <xs:complexType name="past-payment">
        <xs:sequence>
            <xs:element name="requested-paymentid" type="xs:string" minOccurs="1"
maxOccurs="1"/>

            <xs:element name="success" type="xs:boolean" minOccurs="1" maxOccurs="1"/>
            <xs:element name="created" type="xs:dateTime" minOccurs="0" maxOccurs="1"/>

            <xs:element name="tradeid" type="xs:string" minOccurs="0" maxOccurs="1"/>

            <xs:element name="buycurr" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="sellcurr" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="[a-zA-Z]{3}"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="amount" minOccurs="0" maxOccurs="1">
                <xs:simpleType>
                    <xs:restriction base="xs:decimal">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>

            <xs:element name="rate" type="xs:string" minOccurs="0" maxOccurs="1"/>
            <xs:element name="paymentdate" type="xs:date" minOccurs="0" maxOccurs="1"/>

            <xs:element name="error" type="error" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

**Figure 14: Response Past-Payment Schema**

# Error Handling

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:complexType name="error">
        <xs:sequence>
            <xs:element name="code" type="xs:string" minOccurs="1" maxOccurs="1"/>
            <xs:element name="message" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

**Figure 15: Error Response Schema**

```
<response>
    <atomic>false</atomic>
    <testing>false</testing>
    <payment>
        <success>false</success>
        <paymentid>nontrade_payment1</paymentid>
        <error>
            <code>405</code>
            <message>Payment date is in the past</message>
        </error>
        <error>
            <code>407</code>
            <message>The payment date chosen falls on a UK or US non-working public
holiday</message>
        </error>
    </payment>
</response>
```

**Figure 16: Response Example - Bad Payment**

## Error Codes

All errors returned by FX AUTO PAY take the format described by the schema in Figure 15. Every error has a code and a corresponding error message. Each of these codes and messages are listed in Table 8 for reader convenience:

| | |
|---|---|
| 000 | API Offline |
| 100 | Authentication Failed: User record not found in Database |
| 101 | Authentication Failed: User has been removed |
| 102 | Authentication Failed: Key is not set |
| 103 | Authentication Failed: Bad token |
| 104 | Authentication Failed: User disabled |
| 200 | Request XML not well-formed |
| 201 | World First Error: Cannot locate validation schema |
| 202 | Request XML does not validate against schema |
| 300 | Bad value date, general |
| 301 | Value date must be set in future |
| 302 | Quote request amount must be greater than zero |
| 303 | Mismatched currencies |

| | |
|---|---|
| 304 | Same currencies requested in buy/sell |
| 500 | Past Payment Not Found |
| 900 | FXAUTOPAY Database connection error |
| 901 | Failed selecting FXAUTOPAY database |
| 902 | Query on FXAUTOPAY database has failed |
| 910 | Back Office database connection error |
| 911 | Failed selecting Back Office database |
| 912 | Query on Back Office database has failed |
| 400 | Duplicate payment IDs found |
| 401 | Beneficiary not found in Payment |
| 402 | Payment Amount must be greater than zero |
| 403 | Invalid Payment Reason ID |
| 404 | If Payment Reason ID == 6 (other), a 'reason_if_other' must be given |
| 405 | Payment Date is in the past |
| 406 | Payment Date falls on a weekend, please ensure payment dates are set for weekdays |
| 407 | The Payment Date chosen falls on a UK or US non-working public holiday |
| 450 | Payment attempted on an expired quote - quote expired |
| 451 | The accrued total of payments for this trade does not match the amount that was quoted |
| 452 | The user sell currency does not match the Sell Currency specified in the quote |
| 453 | The payment/beneficiary buy currency does not match the Buy Currency specified in the quote |
| 454 | The referenced trade has already been previously covered |
| 455 | The referenced trade is currently unavailable |
| 460 | The trade ID associated with this payment did not match any quoted trades |
| 461 | A payment referencing this trade ID has failed, so the trade was cancelled |
| 462 | Payment not processed because atomic was on and there were 1 or more failed payments |
| 463 | Failed placing order for this quote |
| 464 | Payment aborted because "atomic" was set and other payments failed in this request |
| 600 | Currencies specified did not resolve to a valid trading symbol |
| 601 | Account Holder name is not set <accholder> |
| 602 | Account Holder Name can only be 34 characters in length - length exceeded |
| 603 | Account Holder address was not found in the beneficiary node |
| 604 | Account Holder address may only have alpha-numeric characters, spaces and these symbols: . - = + \' , : ( ) |
| 605 | Bank Address was not found in the beneficiary node |
| 606 | Bank Address may only have alpha-numeric characters, spaces and these symbols: . - = + \' , : ( ) |
| 607 | BIC/SWIFT numbers must be either 8 or 11 characters in length |
| 608 | BIC/SWIFT numbers may only have alpha-numeric characters, spaces and these symbols: . - = + \' , : ( ) |
| 609 | UK: Either a Sort Code or a BIC must be provided |

| | |
|---|---|
| 610 | UK: If a BIC is provided, an IBAN or an account number must also be provided |
| 611 | Missing IBAN |
| 612 | Invalid IBAN |
| 613 | Sort Code is missing |
| 614 | Invalid Sort Code |
| 615 | Missing Bank Code |
| 616 | Invalid Bank Code |
| 617 | Account Number is missing |
| 618 | Either a Bank Code or a BIC/SWIFT must be provided |
| 800 | Quote received but failed placing order on quote, trade not booked |
| 801 | [general connector error] - comes from rate provider |

**Table 8: Error Codes**