BitPOS API

ABN: 13 166 377 528

Jun-14

1. Table of Contents

2. Table of Figures

3. Introduction

The BitPOS API allows you to integrate your service with the BitPOS platform – to get the current Bitcoin exchange rate and to raise and manage orders.

Integration is simple matter of making a call to BitPOS to create an order. Upon creation of the order, BitPOS will return either a URL for and iFrame, or a URL to redirect to.

This iframe / redirection is to enable the order to be presented to the customer, who will then proceed to pay with their bitcoins.

Upon successful payment BitPOS will redirect you to the 'success' callback URL provided in the order creation.

Likewise, if the order payment is unsuccessful, BitPOS will redirect you to the URL provided for a failed payment.

BitPOS provides a full test environment for you to test with. The test environment is configured to use the Bitcoin Testnet environment, which is a staging area provided by the Bitcoin network for testing purposes.

To move from the test platform to production, remove the word 'test' from all example URLs.

4. API Keys

An API Key is the identifier used to secure your access to the BitPOS Platform. API Keys are created through the Merchant Administration screen https://admin.test.bitpos.me.

Each API Key is granted certain user-selectable privileges that restrict what it can be used for.

An API Key consists of a username (An automatically generated random-looking string of letters), and a password.

4.1 Generating an API Key

4.1.1 Step 1 – Go to Administration Dashboard

In you browser, navigate to https://admin.test.bitpos.me

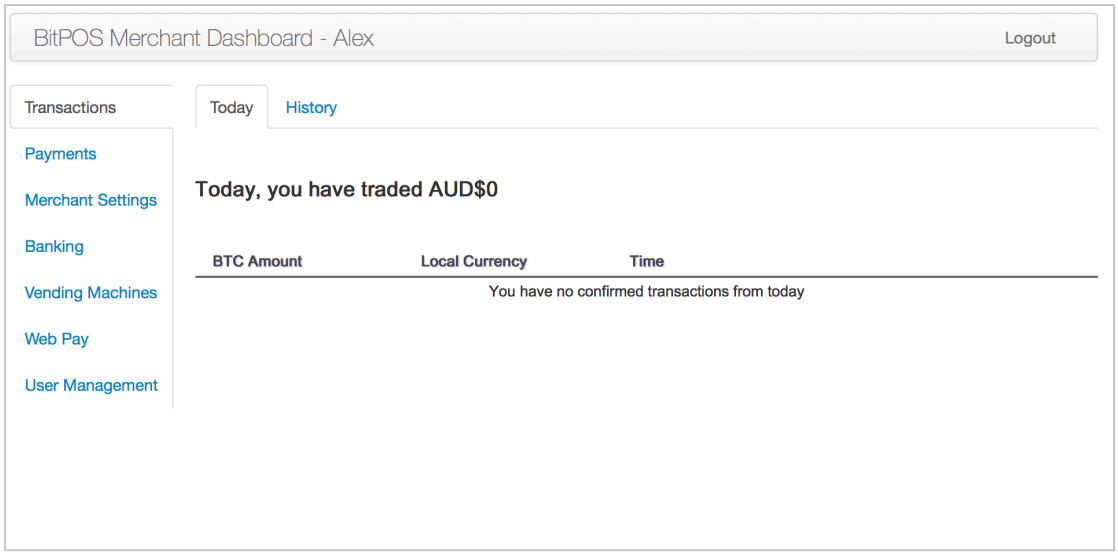

Figure 1 Administration screen

4.1.2 Step 2 – Select 'User Management'

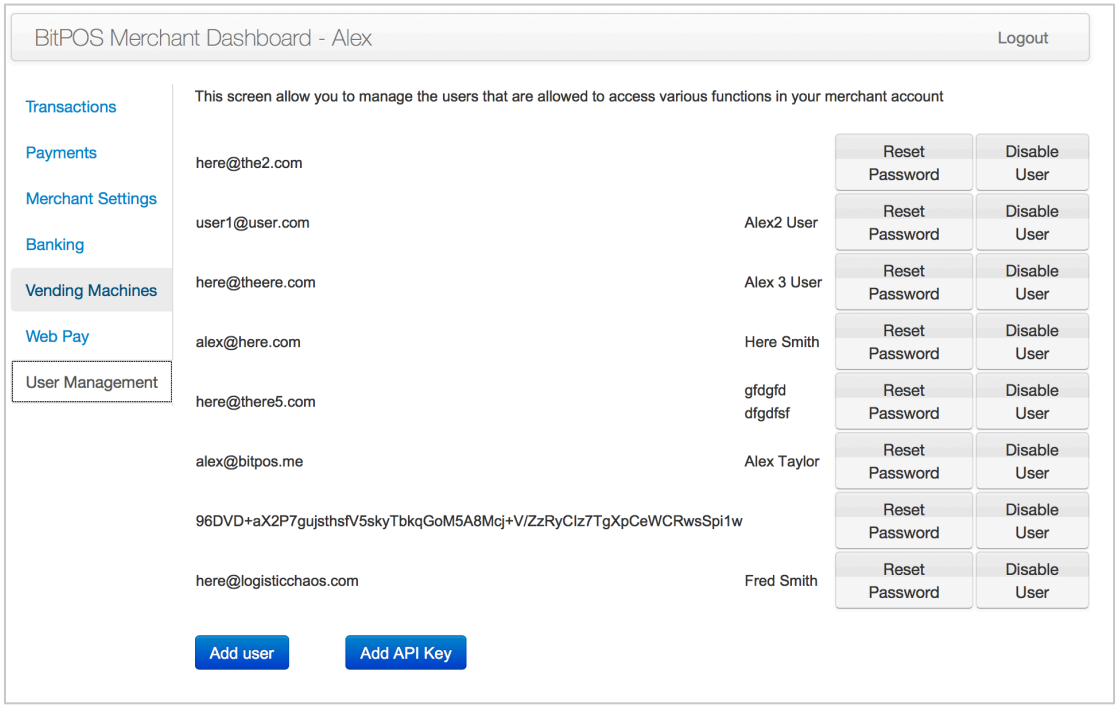

Figure 2 User Management screen

4.1.3 Step 3 – Select 'Add API Key'

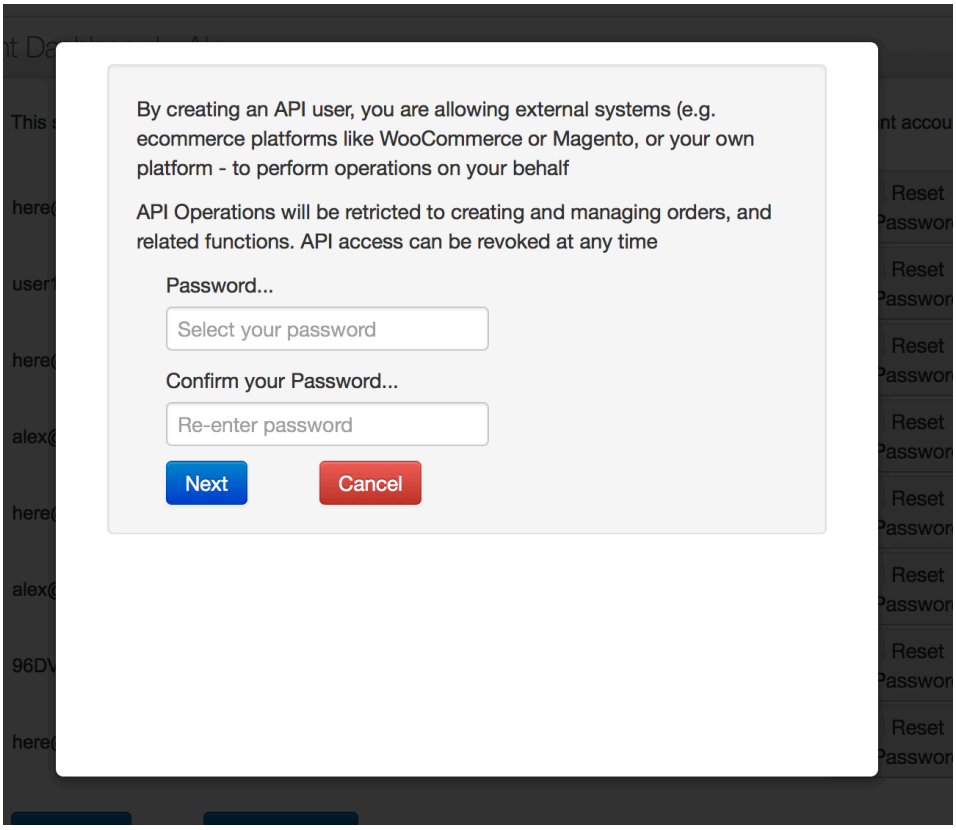

Figure 3 Add API key

4.1.4 Step 4 – Enter the code sent to your mobile phone

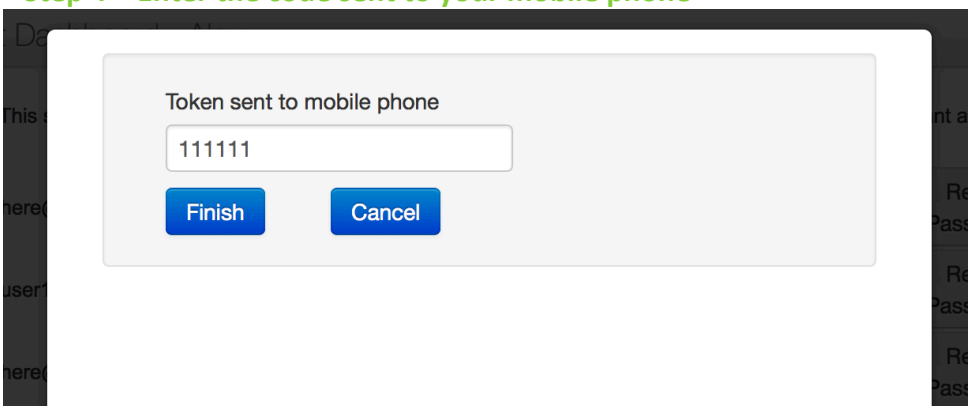

Figure 4 Token screen

Once you have completed Step 4, your new API Key is ready to use!

For all API calls, use the generated API key now visible in the user list as the username, and your selected password as the password.

5. Orders

A BitPOS *Order* represents a purchase that your customer wants to make, stored in the BitPOS platform for payment. It consists of parameters representing the sale amount in the base units of your local currency including an external reference number (eg receipt id) redirect URLs for success and failure.

Creating a BitPOS *Order* allows you to use the BitPOS Platform to sell a product or service. It locks in an exchange rate for 5 minutes, converts a price from a fiat-currency (e.g. Australian or US dollars) into a Bitcoin amount, and optionally generates a URL that can be either embedded into a page as an IFrame (seen below), or loaded via an HTTP redirect.

5.1 Creating an Order

Creating an order can be done using either our Restful or SOAP Webservices, according to your preference

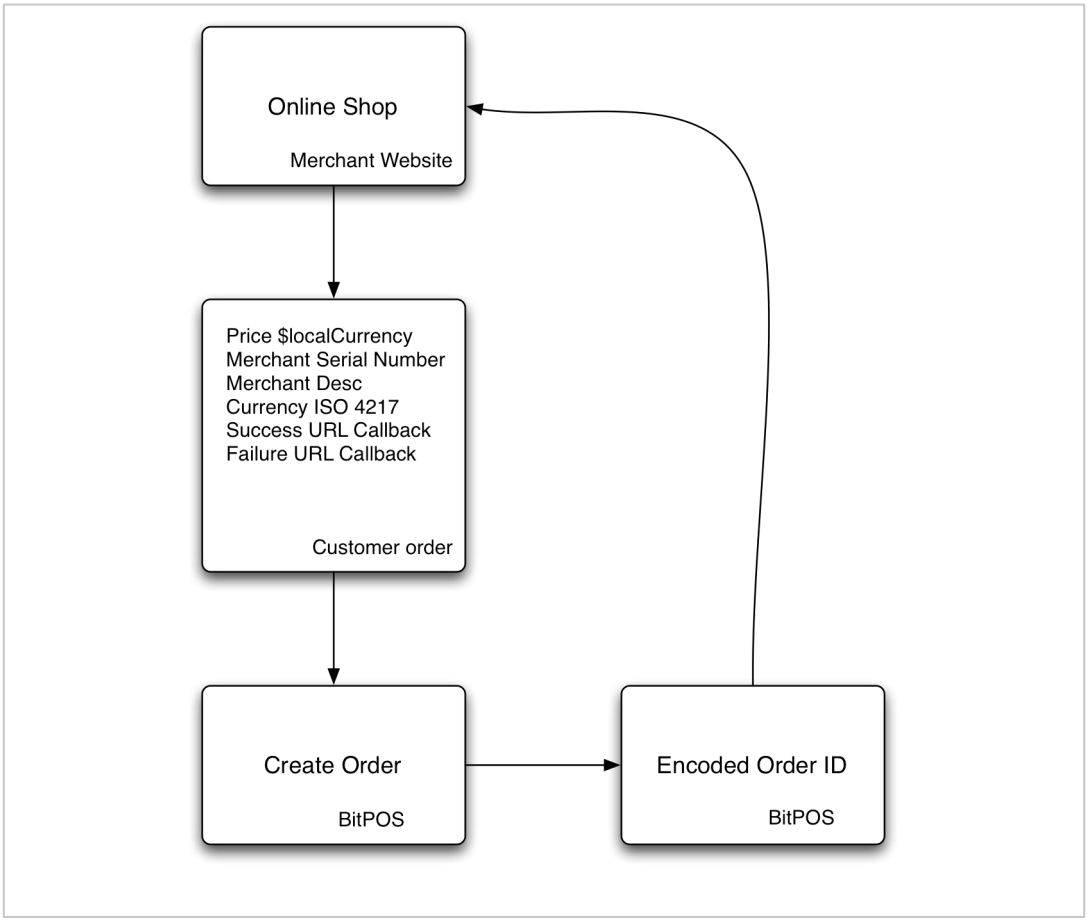

Figure 5 Order creation process

When BitPOS responds with the encoded order ID, you can embed an iFrame in your page using the URL

```
https://payment.test.bitpos.me/payment.jsp?orderId=<encodedOrderId>&frame=1
```

Figure 6 Order iframe URL

or redirect to a BitPOS page

```
https://payment.test.bitpos.me/payment.jsp?orderId=<encodedOrderId>
```

Figure 7 Order redirect URL

BitPOS then accepts the payment and redirects to the success URL, or the failure URL supplied in the original order.

5.1.1 Order confirmation

You must independently determine the status of the order. Failure to do so may result in an attacker performing their own redirection and confirming the order in a fraudulent manner.

To confirm an order use the following URL

```
https://rest.test.bitpos.me/services/webpay/order/status/<encodedOrderId>
```

Figure 8 Order confirmation URL

BitPOS is not responsible for any orders that have not been confirmed using this method.

5.2 Parameters

An order has several basic parameters that govern the way in which the order will behave. They are described in the following table.

Parameter	Definition
currency	The ISO4217 Currency Code for the currency that the 'amount' parameter is specified in. Examples are AUD, NZD, USD, BTC etc
amount	The amount of the order in the **base unit** of the selected currency. For example, if your currency is AUD, 100 equals $1.00
reference	A user selectable reference code for this order. This would normally be a foreign-key to the equivalent order in your own system.
description	A human-readable description for this order
successURL	The URL to redirect to upon successful completion of the order (i.e payment)
failureURL	The URL to redirect to if the order fails (e.g non-payment, timeout, etc)

Figure 9 Order parameters

5.3 Example Order (JSON)

The below is the JSON struture to create an order in Australian dollars for $2.50

```
{
    "currency" => "AUD",
    "amount" => 250,
    "reference" => "123456-ab",
    "description" => "Widgets - Large",
    "successURL" => "http://www.example.com/success",
    "failureURL" => "http://www.example.com/failure",

}
```

Figure 10 Example order data

6. Theming

It is possible to theme the QR code payment screen (iframe and redirect using CSS you may upload to BitPOS. Custom CSS is uploaded in the "Web Pay" tab in the "BitPOS Merchant dashboard".

The themeable classes are:

merchantNameStyle
invoiceBorderStyle
satoshiStype
paymentReferenceStyle
timerlabelStyle
cancelButtonStyle
qrCodeStyle

Figure 11 QR code style reference

7. Examples

7.1 Ruby

7.1.1 SOAP

```
require 'rubygems'
require 'savon'
require 'pp'

base_url = "https://api.test.bitpos.me"
payment_base_url = https://payment.test.bitpos.me
api_key = "example_api_key_5452435432543"
api_key_password = "password"

client = Savon.client(wsdl: base_url +
"/BitPOSWs/OrderService/OrderServiceEndPoint?wsdl", basic_auth:
[api_key, api_key_password]) do
    ssl_verify_mode :none
end


print "Operations:"
print client.operations

response = client.call(:create_order, message: {
currency: "AUD",
amount: 100,
reference: "TICKET15924",
description: "A Concert Ticket",
successURL: "http://example.com.au/orders/success/15924",
failureURL: "http://example.com.au/orders/failure/15924" },
soap_action: "")


#Get response values, and check order status
respHash = response.to_hash

print "Order id: " + respHash[:order][:order_id] + "\n"
print "Value in bitcoin satoshis: " + respHash[:order][:satoshis]
+ "\n\n"
print "URL For Order (iframe): " + payment_base_url +
"/payment.jsp?orderId=" + respHash[:order][:order_id] + "\n\n"

#Now get the order status...
response = client.call(:get_order_status, message:
respHash[:order][:order_id], soap_action: "")

respHash2 = response.to_hash
print "Order status is currently: " +
respHash2[:order_detail][:status] + "\n"
```

Figure 12 Ruby SOAP example

7.1.2 REST

```
require 'rest_client'
require 'json'

base_url = “https://rest.test.bitpos.me”
payment_base_url = https://payment.test.bitpos.me
api_key = “example_api_key_5452435432543”
api_key_password = “password”

jdata = JSON.generate(
{
    "currency" => "AUD",
    "amount" => 250,
    "reference" => "TEST2",
    "description" => "Desc",
    "successURL" => "http://www.example.com/success",
    "failureURL" => "http://www.example.com/failure",

})


resource = RestClient::Resource.new( base_url + '
/services/webpay/order/create', api_key, api_key_password )
response = resource.post jdata, {:content_type => :json}
respHash = JSON.parse(response.to_str)

print "Created order with amount (satoshis): " +
respHash["satoshis"].to_s + "\n\n"

orderId = respHash["encodedOrderId"]
print "Web order URL is: “ + payment_base_url +
“/payment.jsp?orderId=" + orderId + "\n\n"

#Getting status..
resource = RestClient::Resource.new( base_url + '
/services/webpay/order/status/' + orderId, api_key,
api_key_password )
response = resource.get( {:content_type => :json} )

respHash2 = JSON.parse(response.to_str)

print "Status is: " + respHash2["status"] + "\n"
```

Figure 13 Ruby REST example

7.2 Perl

7.2.1 REST

```
use LWP::Simple;
use JSON qw( decode_json );     # From CPAN

my $api_key = "example_api_key_5452435432543";
my $api_key_password = "password";
my $base_url = https://rest.test.bitpos.me;


my $url = $base_url . '/services/webpay/order/create';

#Note - turn on SSL check for production
my $browser = LWP::UserAgent->new(
        ssl_opts => { verify_hostname => 0 },
);

my $req = HTTP::Request->new('POST', $url);
$req->header( 'Content-Type' => 'application/json' );
$req->authorization_basic($api_key, $api_key_password );

#Create order
my $json = ' {
    "currency" : "AUD",
    "amount" : 250,
    "reference" : "TEST2",
    "description" : "Desc",
    "successURL" : "http://www.example.com/success",
    "failureURL" : "http://www.example.com/failure"
        }';

$req->content($json);
my $response = $browser->request($req);

print $response->as_string;

die 'Error getting $url' unless $response->is_success;
print 'Content type is ', $response->content_type;
print 'Content is:';
print $response->content;
print "\n\n";

my $decoded = decode_json($response->content);

print "Order URL:
http://payment.test.bitpos.me/payment.jsp?orderId=$decoded-
>{'encodedOrderId'}&frame=1\n\n";

#continued on next page
```

```
################################
#Get order status

my $statusURL =
"https://rest.test.bitpos.me/services/webpay/order/status/$decoded
->{'encodedOrderId'}";

print "URL for order status: $statusURL\n\n";
my $statusRequest = HTTP::Request->new('GET', $statusURL);
$statusRequest->header( 'Content-Type' => 'application/json' );
$statusRequest->authorization_basic( $api_key, $api_key_password
);

my $statusResponse = $browser->request($statusRequest);

print 'Content type is ', $statusResponse->content_type;
print 'Content is:';
print $statusResponse->content;
print "\n\n";

$decoded = decode_json($statusResponse->content);
print "Current order status is: $decoded->{'status'}\n\n";
```

Figure 14 Perl REST example

7.3 PHP

7.3.1 REST

```
$arr = array('currency' => 'AUD',
'amount' => 1000, //Amount in cents
'reference' => 'TESTORDERID1',
'description' => 'Products',
'successURL' => 'http://yoururl.com/success',
'failureURL' => 'http://yoururl.com/failure');

$data = json_encode($arr);

Mage::Log("JSON Encoded order: " . $data);

$username = 'yourapikey';
$password = 'your api key password';
Mage::Log("Username: $username Password: ********");

$ch = curl_init();
curl_setopt($ch,CURLOPT_URL,
'https://rest.test.bitpos.me/services/webpay/order/create');
curl_setopt($ch, CURLOPT_USERPWD, $username . ":" . $password);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");  //for updating we have
to use PUT method.
curl_setopt($ch, CURLOPT_HTTPHEADER,array('Content-Type:
application/json'));
curl_setopt($ch,CURLOPT_POSTFIELDS,$data);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

$result = curl_exec($ch);
$httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);

$resArray = json_decode($result);
$encodedOrderId = $resArray->{'encodedOrderId'};

print('BitPOS encoded order id is ' . $encodedOrderId);

print "Order URL: http://payment.test.bitpos.me/payment.jsp?orderId=" .
$encodedOrderId . "&frame=1\n\n";

//Check the order status...
$ch = curl_init();
curl_setopt($ch,CURLOPT_URL,
'https://rest.test.bitpos.me/services/webpay/order/status/' +
$encodedOrderId);
curl_setopt($ch, CURLOPT_USERPWD, $username . ":" . $password);
curl_setopt($ch, CURLOPT_HTTPHEADER,array('Content-Type:
application/json'));
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

$result = curl_exec($ch);
$httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);

$resArray = json_decode($result);
$status = $resArray->{'status'};

print "Order status is currently: " . $status;
```

Figure 15 PHP REST example

7.4 SOAP WSDL

The WSDL for the testing platform can be found at:
https://api.test.bitpos.me/BitPOSWs/OrderService/OrderServiceEndPoint?wsdl

```
<?xml version='1.0' encoding='UTF-8'?><wsdl:definitions
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="https://api.bitpos.me/order/create"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
name="OrderCreateService"
targetNamespace="https://api.bitpos.me/order/create">   <wsdl:types>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="https://api.bitpos.me/order/create"
attributeFormDefault="unqualified" elementFormDefault="unqualified"
targetNamespace="https://api.bitpos.me/order/create">   <xs:element
name="orderCreateRequest" type="tns:orderCreateRequest"/>
<xs:element name="orderCreateResponse"
type="tns:orderCreateResponse"/>   <xs:element
name="orderDetailResponse" type="tns:orderDetailResponse"/>
<xs:complexType name="orderCreateRequest">     <xs:sequence>
<xs:element minOccurs="0" name="currency" type="xs:string"/>
<xs:element minOccurs="0" name="amount" type="xs:long"/>
<xs:element minOccurs="0" name="reference" type="xs:string"/>
<xs:element minOccurs="0" name="description" type="xs:string"/>
<xs:element minOccurs="0" name="successURL" type="xs:string"/>
<xs:element minOccurs="0" name="failureURL" type="xs:string"/>
</xs:sequence>   </xs:complexType>   <xs:complexType
name="orderCreateResponse">     <xs:sequence>       <xs:element
minOccurs="0" name="orderId" type="xs:string"/>       <xs:element
minOccurs="0" name="currency" type="xs:string"/>       <xs:element
minOccurs="0" name="satoshis" type="xs:long"/>     </xs:sequence>
</xs:complexType>   <xs:complexType name="orderDetailResponse">
<xs:sequence>       <xs:element minOccurs="0" name="orderId"
type="xs:string"/>       <xs:element minOccurs="0" name="currency"
type="xs:string"/>       <xs:element minOccurs="0" name="satoshis"
type="xs:long"/>       <xs:element minOccurs="0" name="status"
type="xs:string"/>     </xs:sequence>   </xs:complexType>
<xs:element name="orderCreate" nillable="true"
type="tns:orderCreateRequest"/>   <xs:element name="order"
nillable="true" type="tns:orderCreateResponse"/>   <xs:element
name="orderId" nillable="true" type="xs:string"/>   <xs:element
name="orderDetail" nillable="true" type="tns:orderDetailResponse"/>
</xs:schema>   </wsdl:types>   <wsdl:message
name="getOrderStatusResponse">     <wsdl:part
element="tns:orderDetail" name="orderDetail">     </wsdl:part>
</wsdl:message>   <wsdl:message name="createOrderResponse">
<wsdl:part element="tns:order" name="order">     </wsdl:part>
</wsdl:message>   <wsdl:message name="createOrder">     <wsdl:part
element="tns:orderCreate" name="orderCreate">     </wsdl:part>
</wsdl:message>   <wsdl:message name="getOrderStatus">     <wsdl:part
element="tns:orderId" name="orderId">     </wsdl:part>
</wsdl:message>   <wsdl:portType name="OrderCreate">
<wsdl:operation name="createOrder">       <wsdl:input
message="tns:createOrder" name="createOrder">     </wsdl:input>
<wsdl:output message="tns:createOrderResponse"
```

```
name="createOrderResponse">     </wsdl:output>     </wsdl:operation>
<wsdl:operation name="getOrderStatus">       <wsdl:input
message="tns:getOrderStatus" name="getOrderStatus">     </wsdl:input>
<wsdl:output message="tns:getOrderStatusResponse"
name="getOrderStatusResponse">     </wsdl:output>
</wsdl:operation>   </wsdl:portType>   <wsdl:binding
name="OrderCreateServiceSoapBinding" type="tns:OrderCreate">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="createOrder">       <soap:operation
soapAction="" style="document"/>       <wsdl:input
name="createOrder">         <soap:body use="literal"/>
</wsdl:input>       <wsdl:output name="createOrderResponse">
<soap:body use="literal"/>       </wsdl:output>     </wsdl:operation>
<wsdl:operation name="getOrderStatus">       <soap:operation
soapAction="" style="document"/>       <wsdl:input
name="getOrderStatus">         <soap:body use="literal"/>
</wsdl:input>       <wsdl:output name="getOrderStatusResponse">
<soap:body use="literal"/>       </wsdl:output>     </wsdl:operation>
</wsdl:binding>   <wsdl:service name="OrderCreateService">
<wsdl:port binding="tns:OrderCreateServiceSoapBinding"
name="OrderCreatePort">       <soap:address
location="https://api.test.bitpos.me:443/BitPOSWs/OrderService/OrderS
erviceEndPoint"/>     </wsdl:port>   </wsdl:service>
</wsdl:definitions>
```

7.5 Style CSS example

```
.merchantNameStyle {
   font-weight: bold;
   font-size: 20px;
}

.invoiceBorderStyle {
 border:2px solid;
 border-radius:25px;
 border-color: #62BB46;
 border-collapse: separate;
}

.satoshiStyle {
   color: #57a957;
   font-size: 20px;
}
```

Figure 16 CSS style fragment