{COMPANY} Payment Flow and APIs

Document version 1.0

Legal Notice

Documentation version: API
Last release: N/A

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, etc. etc. etc. This is an example doc.

Table of Contents

Introduction

This document describes the Corp™ payment flow and payment APIs between the Corp device, the MON_MON server, the Payment Gateway, and the COMPANY POS. Payment processes enable restaurant customers to initiate and conclude payments through the Corp device.
Payment processes use the following components:

- Corp, the physical user interface device, Corp™
- MON_MON, the server between Corp and the POS
- COMPANY POS, the point of sale
- Payment gateway

Payment overview

Figure 1: Corp Payment Overview shows the high level view of payment flow using Corp, MON_MON, POS, and Payment Gateway.

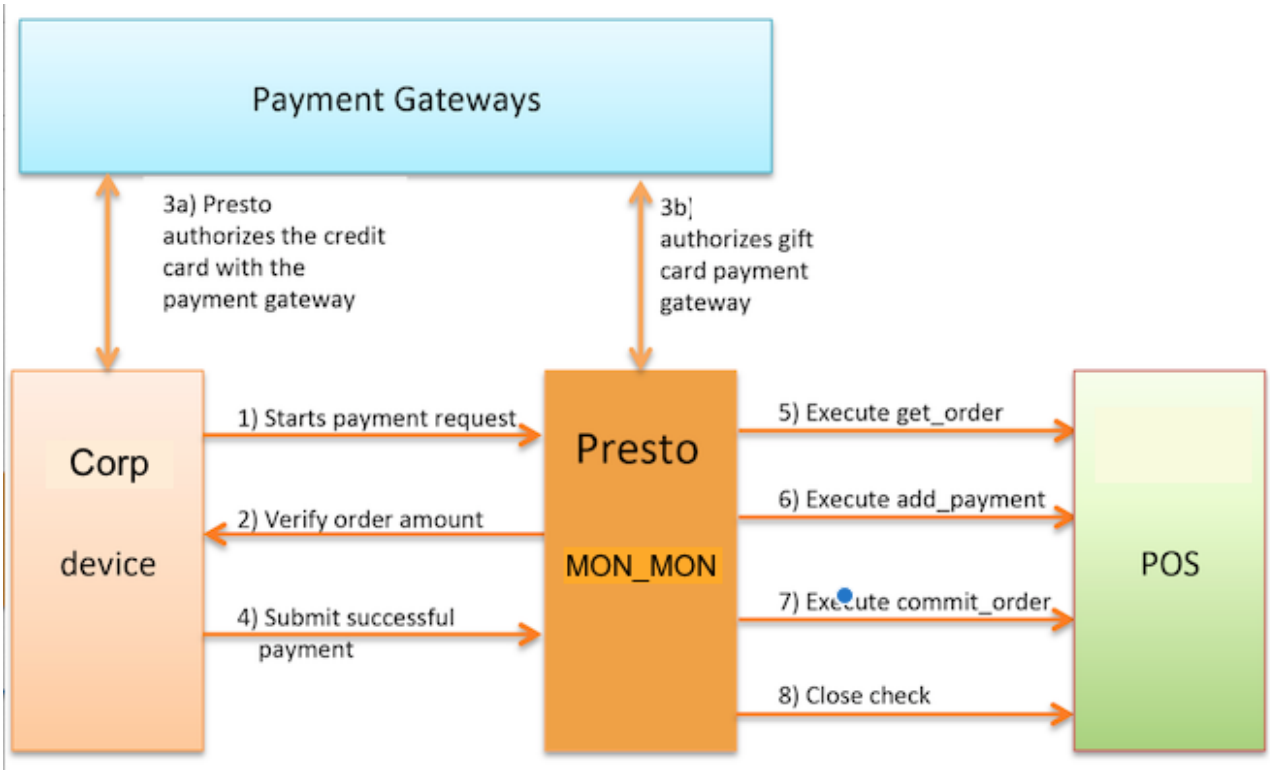

Figure 1: Corp Payment overview workflow

1. Corp starts a payment request with MON_MON.
2. MON_MON verified check with Corp.
3. Corp or MON_MON authorizes card with:
   - a payment gateway, then if credit/debit (bank) card, card is charged.
     or
   - a Store gift card payment gateway.
4. Corp submits successful payment to MON_MON using **submit_process_payment** API.
5. MON_MON executes **get_order** API. This gets the state of the order from POS and locks the check.
6. MON_MON adds a payment to the order with **add_payment** API.
7. MON_MON executes **commit_order** API to POS. The check is updated on POS.
8. MON_MON closes check.

Payment flow processes

This section contains flowcharts and steps that describe the following processes:

- Successful payment process flow
- Check values do not match
- Error submitting POS payment
- Card authorization failure

Successful credit/ debit and Store gift card payment process flow

Figure 2 shows the payment flow processes of a successful payment.

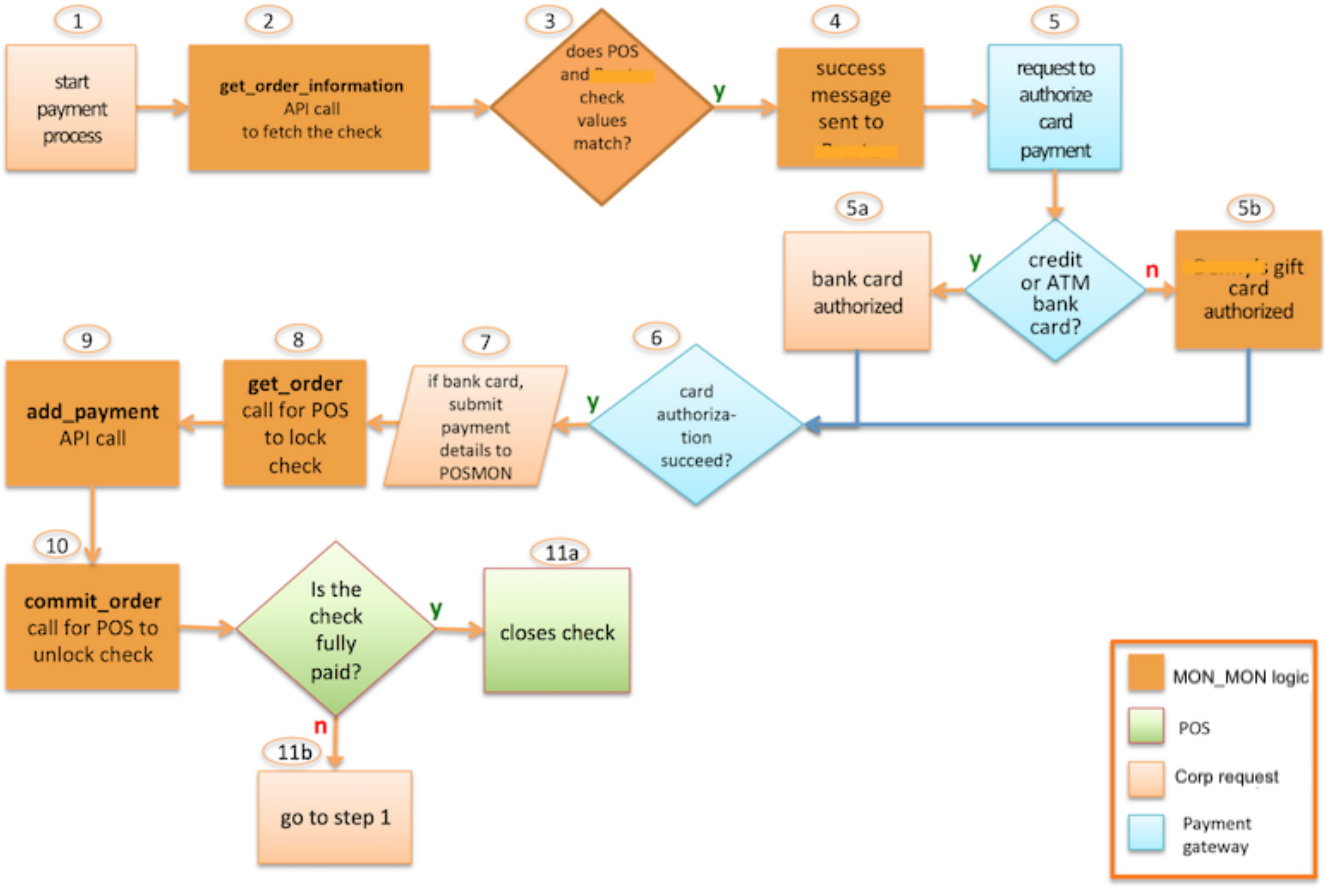

Figure 2: Successful Corp payment process workflow

1. Corp starts a payment process with MON_MON.
2. MON_MON fetches the check from POS with **get_order_information** API.
3. MON_MON compares POS check value with Corp check value.
   They match.
4. MON_MON sends success message to Corp.
5. Credit/debit/gift card payment request authorization/charge with payment gateway.
   a. Corp authorizes/charges bank cards.
   b. MON_MON charges Store gift cards.
6. Did authorization/charge succeed?
   The authorization was successful.
7. (*Only If credit/debit*) Corp submits processed payment details to MON_MON.
8. MON_MON executes **get_order** API to POS.
   POS changes the check state to lock.

9. MON_MON executes **add_payment** API to POS.
   POS changes the state of the check with payment.
10. MON_MON executes **commit_order** API to POS to save the fully paid check state.
    POS changes the check state to unlock.
11. Is the check fully paid?
    a. (*Yes-is fully paid*) The payment flow is complete.
    b. (*No-is not fully paid*) Go to step 1.

Check values do not match

Figure 3 shows a payment flow process when the check values in Corp and POS don't match.

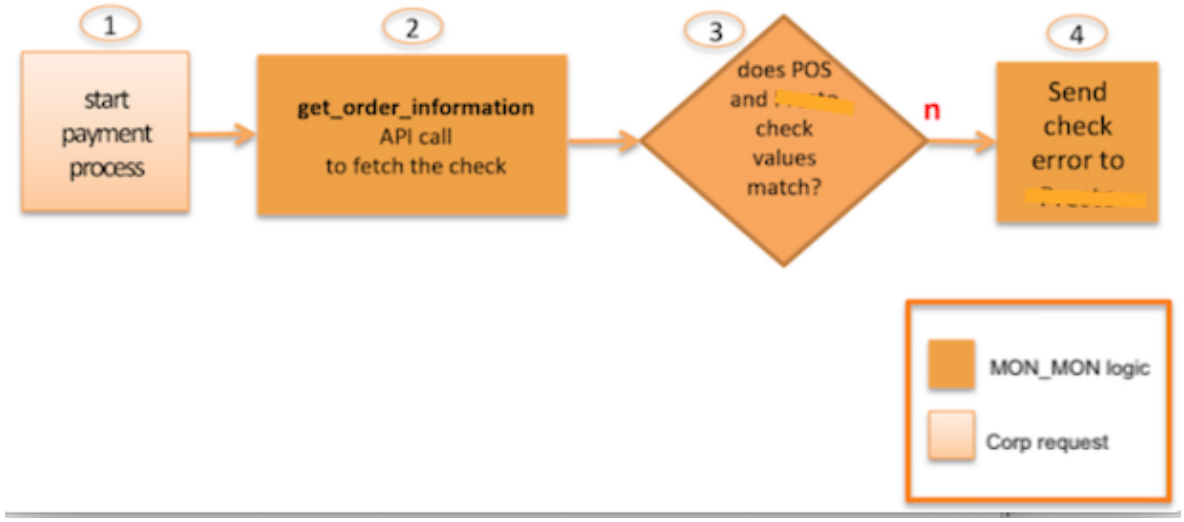

Figure 3: Corp Payment check value does not match workflow

1. Corp starts a payment process with MON_MON.
2. MON_MON fetches the check from POS with **get_order_information** API.
3. MON_MON compares POS check value with Corp check value.
   They do not match.
4. MON_MON sends an error message to Corp, "*Your check has changed.*"

Error submitting POS payment

Figure 4 shows a payment flow process when MON_MON sends a payment to POS and it does not succeed.

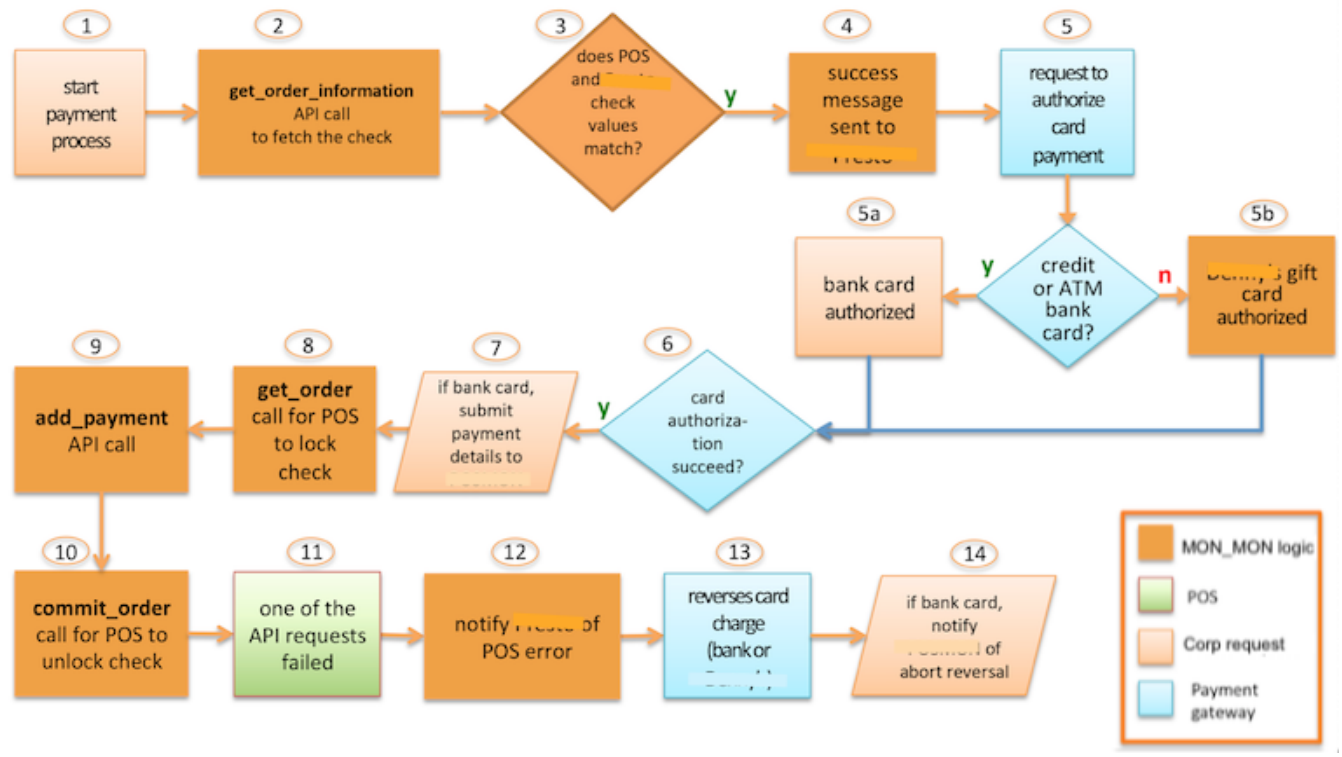

Figure 4: Corp Payment error submitting POS payment workflow

1. Corp starts a payment process with MON_MON.
2. MON_MON fetches the check with **get_order_information** API.
3. MON_MON compares POS check value with Corp check value.
   They match.
4. MON_MON sends a success message to Corp.
5. Credit/debit/gift card payment request authorization/charge with payment gateway.
   a. Corp authorizes/charges bank cards.
   b. MON_MON charges Store gift cards.
6. Did the payment go through?
   The authorization was successful.
7. (*if bank card*) Corp submits processed payment details to MON_MON.
8. MON_MON executes **get_order** API to POS.
   POS changes the check state to lock.
9. MON_MON executes **add_payment** API to POS.
   POS changes the state of the check with payment.
10. MON_MON executes **commit_order** API to POS to save the check state.

POS changes the check state to unlock.

11. Did payment submission to POS succeed?
    The payment submission **was not successful due to failure of an API request.**
12. MON_MON notifies Corp of POS error.
13. *(if Store gift card)* MON_MON reverses payment with SVS payment gateway.
14. (*If bank card)* Corp reverses payment with Payment gateway and notifies MON_MON abort reversal.

Card authorization failure

Figure 5 shows a payment flow process when the credit card is declined.

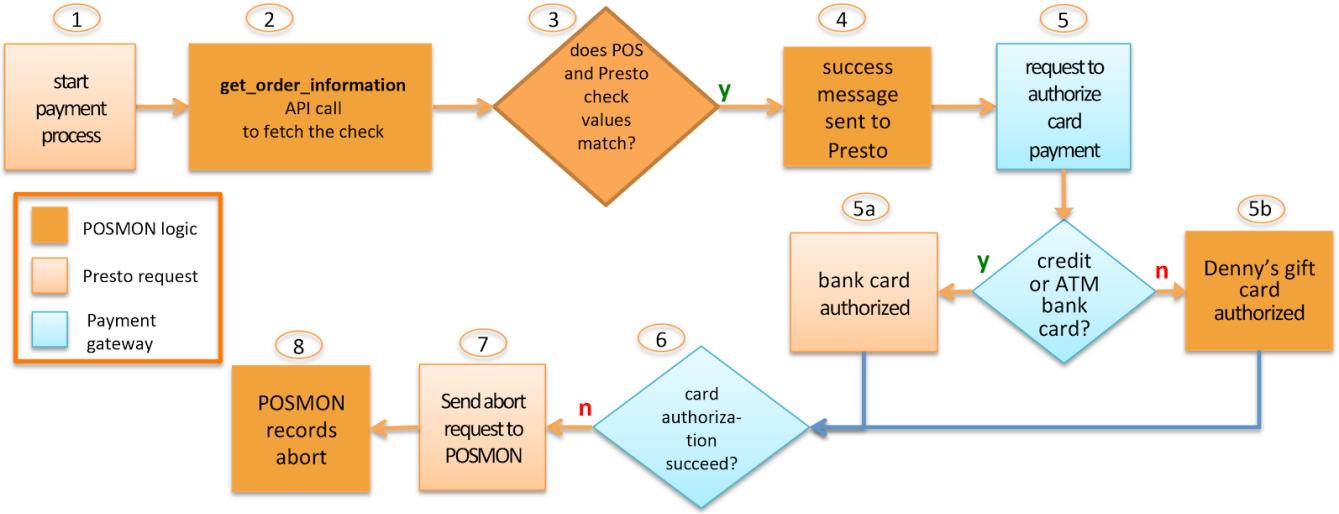

Figure 5: Corp Payment card authorization failure workflow

1. Corp starts a payment process with MON_MON.
2. MON_MON fetches the check from POS with **get_order_information** API.
3. MON_MON compares POS check value with Corp check value.
   They match.
4. MON_MON sends success message to Corp.
5. Credit/debit/gift card payment request authorization/charge with payment gateway.
   a. Corp authorizes/charges bank cards.
   b. MON_MON charges Store gift cards.
6. Did authorization/charge succeed?
   Authorization was not successful.
7. Corp aborts payment and notifies MON_MON.
8. MON_MON records the abort.

Technical notes

To keep checks current at payment time, Corp uses real-time check data sent directly from POS, which keeps the check amount in sync at the time the check is locked.

Item update requests are stored in the MON_MON cache. Corp periodically checks for item updates such as item prices, and refreshes the cache.

API calls during the payment process

Table 1 describes API calls during the payment process. The details explain payment transaction call details and clarify logical processes in Corp and MON_MON.

Table 1: API call list and definitions

API calls	Description
payment_process_start	The **payment_process_start** call verifies that the Corp check is in synch with the POS check, and that Corp can lock the check before it authorizes the credit card.
submit_process_payment	The **submit_process_payment** call is for Corp to submit the payment authorization details to POS.
payment_process_abort	The **payment_process_abort** call is for record keeping purpose for MON_MON. It closes the transaction in the POSMAN database, keeps Corp and MON_MON in synch, and clears the state for the next transaction. The information stays in MON_MON.
get_order	MON_MON calls **get_order** to lock the check and to get the current check state. Payload
add_payment	MON_MON calls add_payment to add an authorized payment to the check. **add_payment** response has updated check state with the added payment
commit_order	MON_MON calls commit_order to unlock the check and save the check state after modification. **commit_order** response has updated check state with the added payment

Status statements

Table 2: Status statements

Statuses
payment success
closed check success
abort acknowledgement

Error types

Table 3: Customer-facing error types

Errors	Description
Check changed	This could happen when the customer has loaded the check contents and then tries to pay on Corp, at the same time *or* before Corp was able to update the check contents, the waiter has done something to modify the check. When the two transactions arrive at POS simultaneously, the state is out of synch and the ErrorCheckChanged error occurs.
Table in Use / User in Use	A check would not be locked if a waiter is accessing the check on the POS when the customer tries to pay the check from Corp.
Payment submit error	Error submitting POS payment happens when MON_MON sends a payment to POS and it doesn't succeed. This is a rare condition and happens for some unknown issue, e.g. POS or network issue.

APIs request and response examples

This section lists parameters and sample API call requests and responses between MON_MON and POS.

GetOrderInformation

Request Parameters:
orderpoint: configurable field that identifies which client is calling the POS
orderIdList: list of unique iIDs associated with each order

Sample Request:

{'orderpoint': <orderpoint type="WEBCLIENT">{{ ORDER_POINT }}</orderpoint>,
'orderIdList': [check_id]}

Sample Response:

```
<response>
  <result>0</result>
  <resultmessage>ok</resultmessage>
  <function>getorderinformation</function>
  <orders>
    <orderinfo>
      <orderid>00400008-04-07112006</orderid>
      <order>
        <orderheader>
          <ordernum>400008</ordernum>
          <registernum>4</registernum>
          <cashiernum></cashiernum>
          <cashdrawernum>1</cashdrawernum>
          <businessdate>07/11/2006</businessdate>
          <origbusinessdate>07/11/2006</origbusinessdate>
          <customername><![CDATA[customername]]></customername>
          <state>4</state>
          <substate>4</substate>
          <destination>2</destination>
          <conceptid></conceptid>
          <tablenum></tablenum>
          <seats>
```

```
      <seat>1</seat>
    </seats>
    <guestcount></guestcount>
    <orderpoint>kiosk1</orderpoint>
    <total>$3.41</total>
    <subtotal>$3.18</subtotal>
    <tax>$0.23</tax>
    <paytotal>$0.00</paytotal>
    <amountdue>$3.41</amountdue>
    <changedue>$0.50</changedue>
    <disctotal>$0.00</disctotal>
    <linedisctotal>$0.00</linedisctotal>
  </orderheader>
  <items>
    <item>
      <itemnum>108</itemnum>
      <modifier>1</modifier>
      <qty>1</qty>
      <serialnum>1</serialnum>
      <seatnum></seatum>
      <parentserialnum></parentserialnum>
      <state>4</state>
      <destination>2</destination>
      <description>famous star</description>
      <price>$1.59</price>
      <tax>$0.11</tax>
      <discountcode></discountcode>
      <kitchenstatus></kitchenstatus>
    </item>
    <item>
      <itemnum>108</itemnum>
      <modifier>1</modifier>
      <qty>1</qty>
      <serialnum>2</serialnum>
      <seatnum></seatnum>
      <parentserialnum></parentserialnum>
      <state>4</state>
      <destination>2</destination>
      <description>famous star</description>
      <price>$1.59</price>
      <tax>$0.12</tax>
      <discountcode></discountcode>
```

```
            <kitchenstatus></kitchenstatus>
          </item>
        </items>
        <paytypes>
          <paytype>
            <Id>201</Id>
            <state>1</state>
            <serialnum>1</serialnum>
            <description>Visa</description>
            <amount>3.41</amount>
            <amountapproved>3.41</amountapproved>
            <account>99999xxxxxx9999</account>
          </paytype>
        </paytypes>
      </order>
    </orderinfo>
  </orders>
</response>
```

GetOrder

Request Parameters:

orderpoint: configurable field that identifies which client is calling POS
orderid: unique id associated with order

Sample Request:

```
{'orderpoint': <orderpoint type="WEBCLIENT">{{ ORDER_POINT }}</orderpoint>,
 'orderid': check_id}
```

Sample Response:

```
<response>
        <result>0</result>
        <resultmessage>Ok</resultmessage>
        <function>getorder</function>
        <orderpoint>Corp1</orderpoint>
        <timestamp>07/21/2009 16:07:42</timestamp>
```

```
<orderid>00200307-02-07212009</orderid>
<order>
  <orderheader>
    <ordernum>200307</ordernum>
    <registernum>2</registernum>
    <cashiernum>3</cashiernum>
    <cashdrawernum>2103</cashdrawernum>
    <businessdate>07/21/2009</businessdate>
    <origbusinessdate>07/21/2009</origbusinessdate>
    <customername></customername>
    <state>4</state>
    <destination>8</destination>
    <conceptid>0</conceptid>
    <tablenum>0</tablenum>
    <guestcount>1</guestcount>
    <orderpoint>WebOrder1</orderpoint>
    <total>113.08</total>
    <subtotal>104.97</subtotal>
    <subtax>
      <type></type>
      <amt>9.0983</amt>
    </subtax>
    <tax>9.10</tax>
    <paytotal>0.00</paytotal>
    <amountdue>113.08</amountdue>
    <changedue>0.00</changedue>
    <disctotal>0.00</disctotal>
    <linedisctotal>0.99</linedisctotal>
  </orderheader>
  <customerdetail>
    <firstname>name</firstname>
    <lastname>lname</lastname>
    <address>200 main st</address>
    <city>charlotte</city>
    <state>NC</state>
    <zip>28226</zip>
    <phoneno>7045551212</phoneno>
    <idno></idno>
    <pickupdate>7/22/2009 4:04:00 PM</pickupdate>
    <futuresendtime>7/22/2009 4:04:00 PM</futuresendtime>
  </customerdetail>
  <items>
```

```
    <item>
      <itemnum>2196</itemnum>
      <modifier>0</modifier>
      <qty>2</qty>
      <serialnum>1</serialnum>
      <seatnum>0</seatnum>
      <parentserialnum>0</parentserialnum>
      <state>4</state>
      <destination>8</destination>
      <description><![CDATA[HARVEST LG]]></description>
      <price>39.99</price>
      <tax>6.93</tax>
      <discountcode>0</discountcode>
      <kitchenstatus>0</kitchenstatus>
      <appdata><![CDATA[User Defined Data]]></appdata>
    </item>
    <item>
      <itemnum>1031</itemnum>
      <modifier>0</modifier>
      <qty>1</qty>
      <serialnum>2</serialnum>
      <seatnum>0</seatnum>
      <parentserialnum>0</parentserialnum>
      <state>4</state>
      <destination>8</destination>
      <description><![CDATA[VIA FRUIT SLD LG]]></description>
      <price>24.99</price>
      <tax>2.17</tax>
      <discountcode>0</discountcode>
      <kitchenstatus>0</kitchenstatus>
      <appdata><![CDATA[Saved with the Order]]></appdata>
    </item>
  </items>
  <discounts>
    <item>
      <itemnum>9996</itemnum>
      <modifier>0</modifier>
      <qty>1</qty>
      <serialnum>3</serialnum>
      <seatnum>-200</seatnum>
      <parentserialnum>0</parentserialnum>
      <state>4</state>
```

```
            <destination>8</destination>
            <description><![CDATA[Online Order $]]></description>
            <price>0.99</price>
            <tax>0.00</tax>
            <discountcode>1051</discountcode>
            <kitchenstatus>0</kitchenstatus>
          </item>
        </discounts>
      </order>
    </response>
```

AddPayment

Request Parameters:

orderpoint: configurable field that identifies which client is calling the POS
orderid: unique id associated with order
id: unique id identifying payment tender type (VISA, MASTERCARD, etc)
amount: total authorized payment, including tax and tip
tip: tip amount

Sample Request:

```
{'orderpoint': <orderpoint type="WEBCLIENT">{{ ORDER_POINT }}</orderpoint>,
 'orderid': check_id
 'addpaymentxml':
          <paytypes>
            <paytype>
              <id>{{ PAY_TYPE_ID }}</id>
              <transactionid></transactionid>
              <account></account>
              <expdate></expdate>
              <amount>{{ PAYMENT_AMOUNT }}</amount>
              <tip>{{ TIP_AMOUNT }}</tip>
              <swipetype></swipetype>
              <authcode></authcode>
              <token></token>
            </paytype>
          </paytypes>
}
```

Sample Response:

```
<response>
  <result>0</result>
  <resultmessage>Ok</resultmessage>
  <function>addpayment</function>
  <orderpoint>WebOrder1</orderpoint>
  <timestamp>07/21/2009 16:44:19</timestamp>
  <orderid>00200309-02-07212009</orderid>
  <order>
    <orderheader>
      <ordernum>200309</ordernum>
      <registernum>2</registernum>
      <cashiernum>3</cashiernum>
      <cashdrawernum>2103</cashdrawernum>
      <businessdate>07/21/2009</businessdate>
      <origbusinessdate>07/21/2009</origbusinessdate>
      <customername><![CDATA[name lname]]></customername>
      <state>0</state>
      <destination>8</destination>
      <conceptid>0</conceptid>
      <tablenum>0</tablenum>
      <guestcount>1</guestcount>
      <orderpoint>WebOrder1</orderpoint>
      <total>114.15</total>
      <subtotal>104.97</subtotal>
      <subtax>
        <type></type>
        <amt>9.1849</amt>
      </subtax>
      <tax>9.18</tax>
      <paytotal>114.15</paytotal>
      <amountdue>0.00</amountdue>
      <changedue>0.00</changedue>
      <disctotal>0.00</disctotal>
      <linedisctotal>0.00</linedisctotal>
    </orderheader>
    <customerdetail>
      <firstname>name</firstname>
      <lastname>lname</lastname>
      <address>200 main st</address>
```

```
    <city>Charlotte</city>
    <state>NC</state>
    <zip>28226</zip>
    <phoneno>7045551212</phoneno>
    <idno></idno>
    <pickupdate>7/22/2009 4:04:00 PM</pickupdate>
    <futuresendtime>7/22/2009 4:04:00 PM</futuresendtime>
  </customerdetail>
  <items>
    <item>
      <itemnum>2196</itemnum>
      <modifier>0</modifier>
      <qty>2</qty>
      <serialnum>1</serialnum>
      <seatnum>0</seatnum>
      <parentserialnum>0</parentserialnum>
      <state>0</state>
      <destination>8</destination>
      <description><![CDATA[HARVEST LG]]></description>
      <price>39.99</price>
      <tax>6.99</tax>
      <discountcode>0</discountcode>
      <kitchenstatus>0</kitchenstatus>
      <appdata><![CDATA[User Defined Data]]></appdata>
    </item>
    <item>
      <itemnum>1031</itemnum>
      <modifier>0</modifier>
      <qty>1</qty>
      <serialnum>2</serialnum>
      <seatnum>0</seatnum>
      <parentserialnum>0</parentserialnum>
      <state>0</state>
      <destination>8</destination>
      <description><![CDATA[VIA FRUIT SLD LG]]></description>
      <price>24.99</price>
      <tax>2.19</tax>
      <discountcode>0</discountcode>
      <kitchenstatus>0</kitchenstatus>
      <appdata><![CDATA[Saved with the Order]]></appdata>
    </item>
  </items>
```

```
      <paytypes>
        <paytype>
          <id>201</id>
          <state>1</state>
          <serialnum>1</serialnum>
          <description>Visa</description>
          <amount>114.15</amount>
          <amountapproved>0.00</amountapproved>
          <account>400555xxxxxx1114</account>
          <token>4005000000000007</token>
        </paytype>
      </paytypes>
    </order>
  </response>
```

CommitOrder

Request Parameters:

orderpoint: configurable field that identifies which client is calling the POS
orderid: unique id associated with order

Sample Request:

```
{'orderpoint': <orderpoint type="WEBCLIENT">{{ ORDER_POINT }}</orderpoint>,
 'orderid': check_id}
```

Sample Response:

```
<response>
    <result>0</result>
    <resultmessage>Ok</resultmessage>
    <function>commitorder</function>
    <orderpoint>HH1</orderpoint>
    <timestamp>01/08/2010 16:13:22</timestamp>
    <orderid>00900004-09-01082010</orderid>
    <order>
      <orderheader>
        <ordernum>900004</ordernum>
        <registernum>9</registernum>
```

```
    <cashiernum>1</cashiernum>
    <cashdrawernum>9001</cashdrawernum>
    <businessdate>01/08/2010</businessdate>
    <origbusinessdate>01/08/2010</origbusinessdate>
    <state>4</state>
    <substate>30100</substate>
    <destination>8</destination>
    <conceptid>0</conceptid>
    <tablenum>0</tablenum>
    <guestcount>1</guestcount>
    <seats>
      <seat>0</seat>
    </seats>
    <orderpoint>HH1</orderpoint>
    <total>5.40</total>
    <subtotal>4.99</subtotal>
    <subtax>
      <type></type>
      <amt>0.4117</amt>
    </subtax>
    <tax>0.41</tax>
    <taxexemptid>0</taxexemptid>
    <paytotal>5.40</paytotal>
    <amountdue>0.00</amountdue>
    <changedue>0.00</changedue>
    <disctotal>0.00</disctotal>
    <linedisctotal>0.00</linedisctotal>
    <surveycode>5090-0139-6201-2265</surveycode>
  </orderheader>
  <items>
    <item>
      <itemnum>11001</itemnum>
      <modifier>0</modifier>
      <qty>1</qty>
      <serialnum>1</serialnum>
      <seatnum>0</seatnum>
      <parentserialnum>0</parentserialnum>
      <state>4</state>
      <destination>8</destination>
      <description><![CDATA[PANDA BOWL]]></description>
      <price>4.99</price>
      <tax>0.41</tax>
```

```
      <discountcode>0</discountcode>
      <kitchenstatus>257</kitchenstatus>
      <appdata><![CDATA[Panda Bowl.]]></appdata>
      <childitems>
        <item>
          <itemnum>10082</itemnum>
          <modifier>0</modifier>
          <qty>1</qty>
          <serialnum>2</serialnum>
          <seatnum>0</seatnum>
          <parentserialnum>1</parentserialnum>
          <state>4</state>
          <destination>8</destination>
          <description><![CDATA[STEAMED]]></description>
          <price>0.00</price>
          <tax>0.00</tax>
          <discountcode>0</discountcode>
          <kitchenstatus>257</kitchenstatus>
          <appdata><![CDATA[Steamed Rice.]]></appdata>
        </item>
        <item>
          <itemnum>10027</itemnum>
          <modifier>0</modifier>
          <qty>1</qty>
          <serialnum>3</serialnum>
          <seatnum>0</seatnum>
          <parentserialnum>1</parentserialnum>
          <state>4</state>
          <destination>8</destination>
          <description><![CDATA[ORANGE CHICKEN]]></description>
          <price>0.00</price>
          <tax>0.00</tax>
          <discountcode>0</discountcode>
          <kitchenstatus>257</kitchenstatus>
          <appdata><![CDATA[Orange Chicken.]]></appdata>
        </item>
      </childitems>
    </item>
  </items>
  <paytypes>
    <paytype>
      <id>201</id>
```

```
            <state>1</state>
            <serialnum>1</serialnum>
            <description>Mastercard</description>
            <amount>5.40</amount>
            <amountapproved>0.00</amountapproved>
            <account>542418xxxxxx1732</account>
            <expdate>04/30/2012</expdate>
          </paytype>
        </paytypes>
      </order>
    </response>
```

Document Edit History

Version	Date	Additions/Modifications	Prepared/Revised by
0.01	November-15-2018	Prepared draft from intake meeting	Kraft Consulting
0.02	November -20-2018	First tech review entering live text edits with the engineering team. Gfx edits to add before final review.	Kraft Consulting
0.03	November -29-2018	Final review for end of month release.	Kraft Consulting
1.0	November -30-2018	Release version	Kraft Consulting

- EOF -