UPS Primary Interface

Reference

8/20/2013

Revision 3.91

Keith Musslewhite

PSG SC Engineering and Quality
Hewlett-Packard Company

Contents

Executive Summary

The UPS system is a set of software tools designed to create a systemic flow of data from the customer to the manufacturing site in order to fulfill specific CIS customization services. The UPS tools consist of one set of tools designed to facilitate the build of the CIS service while another set of tools are designed to provide automation at the manufacturing site. The UPS Primary Interface, commonly known as PUPS, is one of the automation tools. Its primary purpose is to provide a machine-to-machine interface between the manufacturing site's order system and the UPS system. The PUPS interface provides capabilities for the manufacturing site to determine UPS supported services and part numbers, notify UPS of impending factory builds, transfer critical data required for automated inline processing, and gather information required to accurately determine UUT routing.

Purpose

This document describes a machine-to-machine interface to UPS and how to use it. It describes what the UPS Primary Interface Web Service is, what is required to use it, how to make requests and how to process the response. This document should be the main guideline when implementing the UPS Primary Interface Web Service.

Audience

The indented reader of this document is anyone that wishes to implement and set up machine-to-machine communication with UPS to automate the setup of UPS supported customization skus. The reader is required to have a basic knowledge of how web services works and should feel confident using terms like SOAP and XML.

Use of this guide assumes you are familiar with the following:

- XML (for an overview, go to *W3 Schools XML Tutorial*)
- Basic understanding of web services (for an overview, go to *W3 Schools Web Services Tutorial*)
- A programming language for consuming a web service and any related tools

Overview

This document describes the UPS Primary Interface web service and how to set up machine-to-machine communications. It is basically divided into three parts:

1. an introduction to the UPS Primary Interface web service and machine-to-machine communication;
2. a technical description of the method interface and how to understand responses from method calls; and
3. a description of the UPS Primary Interface production environment.

Introduction

The UPS Primary Interface web service supports the SOAP message protocol for calling service actions over an HTTP connection. The easiest way to use the SOAP interface with your application is to use a SOAP toolkit appropriate for your programming platform. SOAP toolkits are available for most popular programming languages and platforms. The service's Web Services Description Language (WSDL) files (inbound WSDL and outbound WSDL) describe the operations along with the format and data types of the operations' requests and responses. Your SOAP toolkit interprets the WSDL files to provide your application access to the operations. For most toolkits, your application calls a service action using routines and classes provided or generated by the toolkit.

Structure of a SOAP Request

A SOAP request is an XML data structure that your SOAP toolkit generates and sends to the service. As described by the service WSDLs, the root element of this structure is named after the operations. You include the parameters for the request inside the root element, according to the UPS schema contained in each WSDL.

Example SOAP Request

The following example shows the XML for a SOAP message that calls the UPSSendBOM operation. Although you probably won't build the SOAP message for a service request manually, it is useful to see what your SOAP toolkit tries to produce when provided with the appropriate values.

Request:

```
POST /UPSService/UPSPrimaryService.asmx HTTP/1.1
Host: localhost
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <UPSSendBOM xmlns="http://hp.com/">
      <TransID>string</TransID>
      <HP_PO_Num>string</HP_PO_Num>
      <HP_SO_Num>string</HP_SO_Num>
      <Qty>int</Qty>
      <BOM>string</BOM>
    </UPSSendBOM>
  </soap12:Body>
</soap12:Envelope>
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <UPSSendBOMResponse xmlns="http://hp.com/">
      <UPSSendBOMResult>
        <retcode>int</retcode>
        <message>string</message>
        <datastring>string</datastring>
      </UPSSendBOMResult>
    </UPSSendBOMResponse>
  </soap12:Body>
</soap12:Envelope>
```

Interface details

- This section lists all methods available in the UPS Primary Interface web service.
- All methods will return a BomStatus structure.

```
public struct BomStatus

{
   public int retcode;
   public string message;
   public string datastring;
}
```

- OSI XML string will conform to the following format(individual xml tags may be empty):
  <OSITag1>OSIData1</OSITag1>…<OSITagx>OSIDatax</OSITagx>

Web Methods

UPSCancelPO

```
[WebMethod]
// parameters
//      HP_PO_Num  = the HP Purchase order number
//      CustPO  = the customer purchase order order number (REF*PO* line in EDI850)
public BomStatus UPSCancelPO(string HP_PO_Num, string CustPO)
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	`"PO " + HP_PO_Num + " cancelled"`	n/a	PO successfully cancelled

Give back range and USI information to the UPS Master.

Use Case:

Cancel HPPO

Notes:

- When canceling PO's, the UPS Master will be given back any of the following which apply:
  - Unused Range numbers that were allocated
  - Unused USI ((User Specific Information) which is uniquely allocated for UUT's
    - An example would be static IP's

UPSGetOSI

```
[WebMethod]
// parameters
//  TransID - Transaction ID (string max len = 50)
//  HP_PO_Num  = the HP Purchase order number (string max len = 25)
//                        OSI returned in the BomStatus.Orderinfo string
//                        (e.g., "<CustomerPO>100338</CustomerPO><OrderDate>09/14/09</OrderDate>")
//
public BomStatus UPSGetOSI( string HP_PO_Num)
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	"Order information found"	OSI XML string	All requirements either available or queued.

Use Case:

Get OSI Information

Notes:

- This returns all the OSI requirements plus any of the data we already have for OSI.
- Designer can determine what information is available and what is missing by parsing the XML data.

UPSGetPrintRouting

```
[WebMethod]
// parameters
//  PartNumber  = the asset tagging part number from the BOM
//
public BomStatus UPSGetPrintRouting(string PartNumber)
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL		Error communicating with UPS
1	"Routing information not found"		This part number does not need to be routed to asset tag printing station
0	"Routing information found"	Routing code	This part number needs to be routed to asset tag printing station

Routing Codes:

Routing Code	Description
R1	Offline chassis label without asset#
R2	Offline chassis label with asset#
R3	Automated chassis tag without asset#
R4	Automated chassis tag with asset#
R5	Automated packaging tag with or without asset#
R6	CAMS tag with asset tag#
R7	CAMS without asset tag#

Use Case:

Get Print Routing Information

Notes:

UPSGetPrintRoutingAll

```
[WebMethod]
// parameters
//  HPPO  = HP purchase order number
//
public BomStatus UPSGetPrintRoutingAll(string HPPO)
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL		Error communicating with UPS
0	"Routing information not found"		This UUT does not need to be routed to asset tag printing station
1	"Routing information found"	Routing codes delimited by semicolon.	This UUT needs to be routed to asset tag printing station

Routing Codes:

Routing Code	Description
R1	Offline chassis label without asset#
R2	Offline chassis label with asset#
R3	Automated chassis tag without asset#
R4	Automated chassis tag with asset#
R5	Automated packaging tag with or without asset#
R6	CAMS tag with asset tag#
R7	CAMS without asset tag#

UPSSendAll

```
public BomStatus UPSSendAll(string TransID, string HP_PO_Num, string CustPO, int Qty, string
CT05, string BOM, string OSI)


[WebMethod]
// parameters
//  TransID - Transaction ID
//  HP_PO_Num  = the HP Purchase order number
//  CustPO  = the customer purchase order order number (REF*PO* line in EDI850)
//  Qty = ordered quantity
//  CT05 part number (blank if n/a)
 // BOM = XML containing part number, description and revision of the master bom in the following format:  (tag names are not case
sensitive but the spelling is enforced)
//     <BOM><BomItem><PartNumber>KP725AV</PartNumber><Description>dc7900 SFF
chassis</Description><Revision>C</Revision></BomItem></BOM>
//  OSI - Order Specific Information in psuedo XML format.
//      example: <CustomerPO>po123</CustomerPO><HPOrderNumber>812764383746</HPOrderNumber>
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	Action taken	n/a	All requirements either available or queued.
1	Action required	XML string of OSI requirements	Need to supply OSI data to UPS
2	“Ready to build”	Print routing code(s) delimited by semicolon	All requirements are available
3	“BuildReadyState Unknown”	n/a	PUPS unable to determine state of BOM
4	“No supported AV found in BOM”	n/a	The BOM sent to PUPS does not contain an AV number the is supported by UPS

NOTES:

Example of Datastring when Retcode equal to 2: “R2 - Offline chassis Label with asset#; R5 - Automated packaging tag with or without asset#”

Use Case:

Partner IT System sends HPPO plus OSI all at once

UPSSendBOM

```
[WebMethod]
/// parameters
//  TransID - Transaction ID reference generated by Partner IT system
//  HP_PO_Num  = the HP Purchase order number
//  CustPO  = the customer purchase order order number (REF*PO* line in EDI850)
//  Qty = ordered quantity (integer max  =  32,767)
//  BOM = XML containing part number, description and revision of the master bom in the
                    following format: (tag names are not case sensitive but the spelling is enforced)
                   <BOM><BomItem><PartNumber>KP725AV</PartNumber><Description>dc7900 SFF
                   chassis</Description><Revision>C</Revision></BomItem></BOM>
                     (string max len =  nvarchar(max))
//
public BomStatus UPSSendBOM(string TransID, string HP_PO_Num, string CustPO,  int Qty, string BOM)
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	Action taken	n/a	All requirements either available or queued.
1	Action required	XML string of OSI requirements	Need to supply OSI data to UPS
2	“Ready to build”	Print routing code(s) delimited by semicolon	All requirements are available
3	“BuildReadyState Unknown”	n/a	PUPS unable to determine state of BOM
4	“No supported AV found in BOM”	n/a	The BOM sent to PUPS does not contain an AV number the is supported by UPS

NOTES:

Example of Datastring when Retcode equal to 2: “R2 - Offline chassis Label with asset#; R5 - Automated packaging tag with or without asset#”

Use Case:

Partner IT System Sends HPPO Information to Site UPS

UPSSendBOMWithCTO5

```
[WebMethod]
// parameters
//  TransID - Transaction ID (string max len = 50)
//  HP_PO_Num  = the HP Purchase order number (string max len = 25)
//  CustPO  = the customer purchase order order number (REF*PO* line in EDI850)
//  Qty = ordered quantity(integer max  =  32,767)
//  CT05 part number (string max len = 15)
//  BOM = XML containing part number, description and revision of the master bom in the
            following format:  (tag names are not case sensitive but the spelling is enforced)
            <BOM><BomItem><PartNumber>KP725AV</PartNumber><Description>dc7900 SFF
            chassis</Description><Revision>C</Revision></BomItem></BOM>
          (string max len =  nvarchar(max))
//
public BomStatus UPSSendBOMWithCTO5(string TransID, string HP_PO_Num, string CustPO, int Qty, string CT05, string BOM)
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	Action taken	n/a	All requirements either available or queued.
1	Action required	XML string of OSI requirements	Need to supply OSI data to UPS
2	“Ready to build”	Print routing code(s) delimited by semicolon	All requirements are available
4	“No supported AV found in BOM”	n/a	The BOM sent to PUPS does not contain an AV number the is supported by UPS
3	“BuildReadyState Unknown”	n/a	PUPS unable to determine state of BOM

NOTES:

Example of Datastring when Retcode equal to 2: “R2 - Offline chassis Label with asset#; R5 -

Use Case:

Partner IT System sends HPPO information to Site UPS.

Remarks:

Some sites use a lot of CTO5’s, and this is a method to include the CTO5 part number when they send us the BOM. The CTO5 number is just used as a reference. Providing a CTO5 number does not offer any additional processing capability.

UPSSendOSI

```
[WebMethod]
// parameters
//  TransID - Transaction ID (string max len = 50)
//  HP_PO_Num  = the HP Purchase order number (string max len = 25)
//  orderinfo = XML string of order data.
            (e.g., "<CustomerPO>100338</CustomerPO><OrderDate>09/14/09</OrderDate>")
            (string max len = 1024)

public BomStatus UPSSendOSI(string TransID, string HP_PO_Num, string orderinfo)
***** ALL PREVIOUS DATA STORED FOR THE PO WILL BE OVERWRITTEN WITH orderinfo STRING ******
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	"Order information saved"	n/a	OSI table updated with information provided

Use Case:

Send OSI Information

UPSVerifyPOReady

```
public BomStatus UPSVerifyPOReady(string HP_PO_Num)


[WebMethod]
// parameters
//  HP_PO_Num  = the HP Purchase order number (string max len = 25)
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	n/a	n/a	Ready to build
1	Specific requirements that are still pending	XML string of OSI information if applicable	Not ready to build

Use Case:

Verify Build Status

UPSSendBOMAndPOLine

```
[WebMethod]
/// parameters
//  TransID - Transaction ID reference generated by Partner IT system
//  HP_PO_Num  = the HP Purchase order number
//  POLine = HPPO line number
//  CustPO  = the customer purchase order order number (REF*PO* line in EDI850)
//  Qty = ordered quantity (integer max  =  32,767)
//  BOM = XML containing part number, description and revision of the master bom in the
                        following format: (tag names are not case sensitive but the spelling is enforced)
                       <BOM><BomItem><PartNumber>KP725AV</PartNumber><Description>dc7900 SFF
                        chassis</Description><Revision>C</Revision></BomItem></BOM>
                    (string max len =  nvarchar(max))
// FinalBOM = True/False value indicating that this is the last line of a PO
// FinalQty = sum of the quantities for HPPO# (only counting lines that are sent to UPS). This value is ignored when FinalBOM equal to False.
//
public BomStatus UPSSendBOMAndPOLine(string TransID, string HP_PO_Num, int POLine, string CustPO,  int Qty, string BOM, bool FinalBOM, int FinalQty )
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	“BOM Received”	n/a	
1	“PO mismatch”	PO number and Summary of PO lines received with quantity	Only manifested when FinalQty does not match sum of received quantities and FinalBOM equal to True

UPSSendBOMWithCTO5AndPOLine

```
[WebMethod]
/// parameters
//  TransID - Transaction ID reference generated by Partner IT system
//  HP_PO_Num  = the HP Purchase order number
//  POLine = HPPO line number
//  CustPO  = the customer purchase order order number (REF*PO* line in EDI850)
//  Qty = ordered quantity (integer max  =  32,767)
//  CT05 part number (string max len = 15)
//  BOM = XML containing part number, description and revision of the master bom in the
                    following format: (tag names are not case sensitive but the spelling is enforced)
                   <BOM><BomItem><PartNumber>KP725AV</PartNumber><Description>dc7900 SFF
                    chassis</Description><Revision>C</Revision></BomItem></BOM>
                (string max len =  nvarchar(max))
// FinalBOM = True/False value indicating that this is the last line of a PO
// FinalQty = sum of the quantities for HPPO# (only counting lines that are sent to UPS). This value is ignored when FinalBOM equal to False.

//
public BomStatus UPSSendBOMAndPOLine(string TransID, string HP_PO_Num, int POLine, string CustPO,  int Qty, string CT05, string BOM, bool FinalBOM, int FinalQty )
```

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	“BOM Received”	n/a	
1	“PO mismatch”	PO number and Summary of PO lines received with quantity	Only manifested when FinalQty does not match sum of received quantities and FinalBOM equal to True

UPSSendAllAndPOLine

[WebMethod]
```
// parameters
//  TransID - Transaction ID
//  HP_PO_Num  = the HP Purchase order number
//  POLine = HPPO line number
//  CustPO  = the customer purchase order order number (REF*PO* line in EDI850)
//  Qty = ordered quantity
//  CT05 part number (blank if n/a)
// BOM = XML containing part number, description and revision of the master bom in the following format:  (tag names are not case sensitive but the spelling is enforced)
//     <BOM><BomItem><PartNumber>KP725AV</PartNumber><Description>dc7900 SFF chassis</Description><Revision>C</Revision></BomItem></BOM>
//  OSI - Order Specific Information in psuedo XML format.
//      example: <CustomerPO>po123</CustomerPO><HPOrderNumber>812764383746</HPOrderNumber>
// FinalBOM = True/False value indicating that this is the last line of a PO
// FinalQty = sum of the quantities for HPPO# (only counting lines that are sent to UPS). This value is ignored when FinalBOM equal to False.
//
```

public BomStatus UPSSendAllAndPOLine(string TransID, string HP_PO_Num, int POLine, string CustPO, int Qty, string CT05, string BOM, string OSI, bool FinalBOM. int FinalQty)

Returns BomStatus with the following:

Retcode	Message	Datastring	Meaning
-1	Error message from SQL	n/a	Error communicating with UPS
0	“BOM Received”	n/a	
1	“PO mismatch”	PO number and Summary of PO lines received with quantity	Only manifested when FinalQty does not match sum of received quantities and FinalBOM equal to True

UPSGetCISServices

```
public DataSet UPSGetCISServices()

[WebMethod]
// parameters
//  none

Returns a dataset of all CIS service

Example:
```

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://hp.com/">
- <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
- <xs:element name="NewDataSet" msdata:IsDataSet="true"
msdata:UseCurrentLocale="true">
- <xs:complexType>
- <xs:choice minOccurs="0" maxOccurs="unbounded">
- <xs:element name="CISServices">
- <xs:complexType>
- <xs:sequence>
<xs:element name="CISService" type="xs:string" minOccurs="0" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>
- <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
- <NewDataSet xmlns="">
- <CISServices diffgr:id="CISServices1" msdata:rowOrder="0">
<CISService>AY100AV</CISService>
</CISServices>
- <CISServices diffgr:id="CISServices2" msdata:rowOrder="1">
<CISService>AY101AV</CISService>
</CISServices>
- <CISServices diffgr:id="CISServices3" msdata:rowOrder="2">
<CISService>AY102AV</CISService>
</CISServices>
- <CISServices diffgr:id="CISServices4" msdata:rowOrder="3">
<CISService>AY103AV</CISService>
</CISServices>
</NewDataSet>
</diffgr:diffgram>
</DataSet>
```

UPSGetUPSServices

```
public DataSet UPSGetUPSServices()


[WebMethod]
// parameters
//  none

Returns a dataset of all UPS supported CIS services

Example:
```

```
  <?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://hp.com/">
- <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
- <xs:element name="NewDataSet" msdata:IsDataSet="true"
msdata:UseCurrentLocale="true">
- <xs:complexType>
- <xs:choice minOccurs="0" maxOccurs="unbounded">
- <xs:element name="UPSServices">
- <xs:complexType>
- <xs:sequence>
  <xs:element name="CISService" type="xs:string" minOccurs="0" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:choice>
  </xs:complexType>
  </xs:element>
  </xs:schema>
- <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
- <NewDataSet xmlns="">
- <UPSServices diffgr:id="UPSServices1" msdata:rowOrder="0">
  <CISService>AY110AV</CISService>
  </UPSServices>
- <UPSServices diffgr:id="UPSServices2" msdata:rowOrder="1">
  <CISService>AY111AV</CISService>
  </UPSServices>
  </NewDataSet>
  </diffgr:diffgram>
  </DataSet>
```

UPSGetSupportedAV

```
public DataSet UPSGetSupportedAV(DateTime cutoffdate)

[WebMethod]
// parameters
// cutoffdate  = date of last query

Returns a dataset of all UPS supported AV numbers added to UPS since cutoffdate

Example:
```

```
<?xml version="1.0" encoding="utf-8" ?>
- <DataSet xmlns="http://hp.com/">
+ <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
- <diffgr:diffgram xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"
xmlns:diffgr="urn:schemas-microsoft-com:xml-diffgram-v1">
- <NewDataSet xmlns="">
- <SupportedAVs diffgr:id="SupportedAVs1" msdata:rowOrder="0">
  <TemplateName>AB100AV</TemplateName>
  </SupportedAVs>
- <SupportedAVs diffgr:id="SupportedAVs2" msdata:rowOrder="1">
  <TemplateName>AC115AV</TemplateName>
  </SupportedAVs>
- <SupportedAVs diffgr:id="SupportedAVs3" msdata:rowOrder="2">
  <TemplateName>AE110AV</TemplateName>
  </SupportedAVs>
- <SupportedAVs diffgr:id="SupportedAVs4" msdata:rowOrder="3">
  <TemplateName>AM114AV</TemplateName>
  </SupportedAVs>
  </NewDataSet>
  </diffgr:diffgram>
  </DataSet>
```

Using the web service in production

This section describes the UPS Primary Interface web service production environment and how it should be used.

BOM information needs to be passed to the UPS system when received by site. The site should initiate this communication when they receive a purchase order from HP. UPS will receive the BOM information and compare the data to the setup information in UPS to determine requirements (asset numbers, order specific info, etc …).

A significant number of asset tagging services require information specific to an individual order. The data items can either be a standard order field such as *shipping address* or a more abstract customer defined data element. HP's ordering systems do not contain a method for mapping all order specific data elements into specific fields that can be extracted in a consistent manner. The complexity of this requirement prohibits full automation requiring some input from the partner.

Example

```
using System;
using System.Collections.Generic;
using System.Text;

public struct BomStatus
{
    public int retcode;
    public string message;
    public string datastring;
}



namespace UPSPrimaryConsumer
{
    class Program
    {
        static void Main(string[] args)
        {


            UPSPrimaryConsumer.localhost.BomStatus stat;

            /* variables for testing purposes. These should come from the partner IT system used
to consume the web service */
            string transid = "123457";
            string ponumber = "test2";
            string customerpo = "ABCD0001";
            int qty = 10;
            string bominfo = "<BOM>" +
                             "<BomItem><PartNumber>KP725AV</PartNumber><Description>dc7900 SFF
chassis with 85% PSU ALL</Description><Revision>C</Revision></BomItem>" +
                             "<BomItem><PartNumber>KP739AV</PartNumber><Description>Intel Core 2
Duo E8500 Processor</Description><Revision>A</Revision></BomItem>" +
                             "<BomItem><PartNumber>KV968AV</PartNumber><Description>AY111AV Asset
tagging for customer XYZ</Description><Revision> A</Revision></BomItem>" +
                             "<BomItem><PartNumber>KV926AV</PartNumber><Description>160GB SATA 3.5
#1 Hard Drive</Description><Revision>A</Revision></BomItem>" +
                             "<BomItem><PartNumber>KV906AV</PartNumber><Description>ATI Radeon HD
2400 XT PCIe x16 (#1) Card</Description><Revision>A</Revision></BomItem>" +
                             "<BomItem><PartNumber>GD779AV#ABA</PartNumber><Description>HP USB
Standard Keyboard</Description><Revision>E</Revision></BomItem>" +
                             "</BOM>";


            localhost.UPSPrimaryWebSvc svc = new localhost.UPSPrimaryWebSvc();
```

```
            stat = svc.UPSSendBOM(transid, ponumber, customerpo, qty, bominfo);
            switch (stat.retcode)
            {
                case 0:  // order accepted - no furthur requirements
                    break;

                case 1:  // OSI required
                    break;

                case 2:  // ready to build
                    string PrintRouting = stat.datastring;
                    break;
                case 3:  // BuildReadyState unknown
                    break;

                case -1: // error condition
                    break;
            }

            stat = svc.UPSVerifyPOReady(ponumber);
            switch (stat.retcode)
            {
                case 0:  // ready to build
                    break;
                case 1:  // not ready to build
                    break;
                case -1: // error condition
                    break;
            }

        }
    }
}
```

Change History

Keith Musslewhite	1/11/2011	Rev 1.2. Initial document
Chuck Elliott	5/10/2011	Rev 1.3 Added formatting, Table of Contents and change history section
Chuck Elliott	6/17/2011	Rev 1.4 Added use case references and notes sections to methods. Put methods in alphabetical order .
Keith Musslewhite	7/13/2011	Rev 1.5 Added HP sales order number to all SendBOM type methods.
Keith Musslewhite	7/28/2011	Rev 1.7 Added new webmethod - GetPrintRouting
Keith Musslewhite	1/23/2012	Rev 1.8 Added three new methods - UPSGetCISServices, UPSGetUPSServices, UPSGetSupportedAV
Keith Musslewhite	3/8/2012	Rev 2.0 Change methods to accept CustPO rather than HP_SO_Num
Keith Musslewhite	3/14/2012	Rev 2.1 Removed reference for inability to use UPSCancelPO when 1 or more units already built.
Keith Musslewhite	4/11/2012	Rev 3.0 Added return codes 2 & 3 to UPSSendBOM, UPSSendBOMWithCT05, UPSSendAll methods
Keith Musslewhite	4/19/2011	Rev 3.1 added definition of OSI xml string.
Keith Musslewhite	7/1/2012	Rev 3.2 added 3 sendbomwithpoline webmethods to support FXPA
Claudia Castro	9/12/2012	Update date in the first page to be 7/1/2012 instead of 4/19/2012
Keith Musslewhite	11/06/2012	Rev 3.3 added UPSGetPrintRoutingAll webmethod
Keith Musslewhite	11/08/2012	Rev 3.4 corrected web method name UPSGetPrintRoutingAll cut and paste omission.
Keith Musslewhite	11/08/2012	Rev 3.5 clarified meaning of “routing code” in status table for UPSGetPrintRoutingAll.
Keith Musslewhite	12/04/2012	Rev 3.6 added Executive Summary
Claudia Castro	1/11/2013	Rev 3.7 added appendix with document: FX PA Data Exchange Interfaces
Keith Musslewhite	4/9/2013	Rev 3.8 added return code 4 definition to all ‘SendBOM’ methods
Keith Musslewhite	6/21/2013	Rev 3.9 edited for accuracy regarding EDI and XML tag references.
Keith Musslewhite	8/20/2013	Rev 3.91 Removed appendix with document: FX PA Data Exchange Interfaces. Regional implementations do not belong in this reference document.