# xpedx.com Next generation

# *UOM List*

**Authors: Sterling Commerce**

**Date Created:** 05/14/10

**Last Updated:** 05/14/10

**File Name:** C:\Documents and Settings\bfurman\My Documents\Temp\Methodology v1.1\Project Management\TEMPLATE - DOCUMENT - Use Case Definition.docUOM List Low level design

Document Revision History

This chart tracks the changes introduced by the revisions to the document as the project progresses through the stages of the System Development Life Cycle (SDLC).

| Version | **Date** | **Description (Changes Made)** | **Author(s)** |
| --- | --- | --- | --- |
| 1.0 | 05/14/2010 | Initial Draft | Sterling |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Related or Reference Documents

| Document Name | Description | Owner | Location |
| --- | --- | --- | --- |
| SCI\_Xpedx Solution Definition Document v1.0 | Solution Definition document | Sterling Commerce |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

TABLE OF CONTENTS

1. Introduction 4

1.1 Document Purpose 4

1.2 Document Audience 4

2 UOM List 5

2.1 Overview 5

2.1.1 Sequence Flow 6

2.1.2 Input XML 6

2.1.3 Output XML 6

3 Classes and Services 6

4 Unit test cases 13

# Introduction

## Document Purpose

This document develops helps developers to code/maintain UOM list functionality.

## Document Audience

This document is intended for developers/architects/support to understand the logic involved to retrieving list of UOMs specific to xpedx requirements.

# UOM List

## Overview

The following aspects are involved in retrieving UOM list.

1. Items’ base UOM
2. Alternate UOMs
3. Ordering UOMs
4. Customer specific UOM
5. Customer-Item-UOM

System retrieves CustomerID, itemID and storefrontID as inputs to give list of valid UOMs to the client.

## Sequence Flow

1. Get the item details by passing Item ID and Organization Code, this gives item details plus Item Vs Customer details
2. Obtain the order multiple from Item details.
3. Get alternate UOM list and get IsOrderingUOM from alternate UOM.
4. Get the customer details by passing customer ID, this gives all customer details including Customer Number and Customer Branch
5. Break the input CustomerID into three parts- <Customer Div>-<Legacy Customer Number>-Suffix, here ‘–‘is a delimiter.
6. While looping through customer Item details, ignore the records where @Customer Number does not match <Legacy Customer Number> OR @Customer Branch does not match Customer Div
7. Navigate through alternate UOMs and add to list of UOMs and conversion factors to a map if its IsOrderingUOM is set to ‘Y’
8. Navigate through all Item customer details; get Customer Unit, Conv Factor and ExtnUseOrderMulUOMFlag
9. Add above Customer Unit and conversion factors to a map.
10. If ExtnUseOrderMulUOMFlag is set to ‘Y’, Get conv = @ConvFactor / OrderMultiple, IF abs (conv) = conv, this can be lowest conversion factor if this least among all while iterating through the loop. This UOM should be top of the map.
11. If customer specific UOM is defined, only that UOM will be returned to client else return the list of UOMs and conversion factors.

## Input XML

<UOMList CustomerID=*""* ItemID=*""* OrganizationCode=*""*/>

## Output XML

<UOMList>

<UOM UnitOfMeasure=*""* ConversionFactor=*""* />

<UOM UnitOfMeasure=*""* ConversionFactor=*""* />

<UOM UnitOfMeasure=*""* ConversionFactor=*""* />

</UOMList>

# Classes and Services

A new service called ‘XPXUOMListAPI’ is defined to fetch the valid UOMs. This will be defined in the master Config. This service uses the following class to retrieve the UOMs.

package com.xpedx.nextgen.uom.api;

import java.rmi.RemoteException;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Properties;

import java.util.Set;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NamedNodeMap;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import com.sterlingcommerce.baseutil.SCXmlUtil;

import com.yantra.interop.japi.YIFApi;

import com.yantra.interop.japi.YIFClientCreationException;

import com.yantra.interop.japi.YIFClientFactory;

import com.yantra.interop.japi.YIFCustomApi;

import com.yantra.yfc.dom.YFCDocument;

import com.yantra.yfc.dom.YFCElement;

import com.yantra.yfc.log.YFCLogCategory;

import com.yantra.yfs.japi.YFSEnvironment;

import com.yantra.yfs.japi.YFSException;

public class **XPXUOMListAPI** implements YIFCustomApi {

/\*\* API object. \*/

private static YIFApi api = null;

private static YFCLogCategory log;

private String lowestConvUOM = "";

private int currentConversion;

private String ExtnIsCustUOMExcl = "";

public void setProperties(Properties params) throws Exception {

// this.props = props;

}

static {

try {

log = YFCLogCategory.instance(XPXUOMListAPI.class);

api = YIFClientFactory.getInstance().getApi();

} catch (YIFClientCreationException e1) {

log.info(e1.getMessage());

}

}

public Document getXpedxUOMList(YFSEnvironment env, Document inXML)

throws YFSException, YIFClientCreationException, RemoteException {

log.beginTimer("XPXUOMListAPI:getUOMList started...");

HashMap<String, String> htBaseUOMConv = new HashMap<String, String>();

String LegacyCustomerNumber = "";

String companyCode = "";

String useOrderMulUOMFlag = "";

String orderMultiple = "";

String customerDivision = "";

Element documentElement = inXML.getDocumentElement();

String itemID = documentElement.getAttribute("ItemID");

String customerID = documentElement.getAttribute("CustomerID");

String storeFrontId = documentElement.getAttribute("OrganizationCode");

String[] customerIDTokens = customerID.split("\\-");

if (customerIDTokens != null && customerIDTokens.length > 1) {

LegacyCustomerNumber = customerIDTokens[1];

}

HashMap<String, String> customerDetails = getCustomerDetails(env, inXML);

companyCode = customerDetails.get("companyCode");

customerDivision = customerDetails.get("customerDivision");

useOrderMulUOMFlag = customerDetails.get("useOrderMulUOMFlag");

YFCDocument inputDocument = YFCDocument.createDocument("Item");

YFCElement inputElement = inputDocument.getDocumentElement();

inputElement.setAttribute("ItemID", itemID);

inputElement.setAttribute("OrganizationCode", storeFrontId);

env.setApiTemplate("getItemList", SCXmlUtil.createFromString(""

+ "<ItemList><Item><AlternateUOMList><AlternateUOM />"

+ "</AlternateUOMList><Extn><XPXItemExtnList><XPXItemExtn/>"

+ "</XPXItemExtnList><XPXItemcustXrefList><XPXItemcustXref />"

+ "</XPXItemcustXrefList></Extn></Item></ItemList>"));

api = YIFClientFactory.getInstance().getApi();

Document outputListDocument = api.invoke(env, "getItemList",

inputDocument.getDocument());

Element outputListElement = outputListDocument.getDocumentElement();

NodeList itemListNodes = outputListElement.getChildNodes();

int length = itemListNodes.getLength();

for (int i = 0; i < length; i++) {

Node itemNode = itemListNodes.item(i);

NodeList itemNodeChildren = itemNode.getChildNodes();

int length1 = itemNodeChildren.getLength();

for (int j = 0; j < length1; j++) {

Node itemNodeChild = itemNodeChildren.item(j);

if (itemNodeChild != null

&& itemNodeChild.getNodeName().equals("Extn")) {

NodeList extnChildList = itemNodeChild.getChildNodes();

int length2 = extnChildList.getLength();

for (int k = 0; k < length2; k++) {

Node extnChild = extnChildList.item(k);

if (extnChild != null

&& extnChild.getNodeName().equals( "XPXItemExtnList")) {

orderMultiple = getOrderMultipleValue(extnChild,

companyCode, customerDivision);

} else if (extnChild != null

&& extnChild.getNodeName().equals(

"XPXItemcustXrefList")) {

handleXpxItemcustXrefList(extnChild,

LegacyCustomerNumber, customerDivision,

useOrderMulUOMFlag, orderMultiple,

htBaseUOMConv);

if (ExtnIsCustUOMExcl != null

&& ExtnIsCustUOMExcl.equals("Y")) {

return getOutputDocument(htBaseUOMConv, "");

}

}

}

} else if (itemNodeChild != null

&& itemNodeChild.getNodeName().equals(

"AlternateUOMList")) {

handleAternateUOMs(itemNodeChild, htBaseUOMConv);

}

}

}

env.clearApiTemplate("getItemList");

Document document = getOutputDocument(htBaseUOMConv, lowestConvUOM);

log.endTimer("XPXUOMListAPI:getUOMList ended...");

return document;

}

private void handleXpxItemcustXrefList(Node extnChild,

String legacyCustomerNumber, String customerDivision,

String useOrderMulUOMFlag, String orderMultiple,

HashMap<String, String> htBaseUOMConv) {

NodeList XpxItemcustXrefList = extnChild.getChildNodes();

int length3 = XpxItemcustXrefList.getLength();

for (int m = 0; m < length3; m++) {

Node XpxItemcustXref = XpxItemcustXrefList.item(m);

NamedNodeMap XpxItemcustXrefAttributes = XpxItemcustXref

.getAttributes();

if (!isCustomerMatching(XpxItemcustXrefAttributes,

legacyCustomerNumber, customerDivision)) {

continue;

}

Node ExtnIsCustUOMExclNode = XpxItemcustXrefAttributes

.getNamedItem("ExtnIsCustUOMExcl");

if (ExtnIsCustUOMExclNode != null) {

ExtnIsCustUOMExcl = ExtnIsCustUOMExclNode.getTextContent();

}

Node customerUnitNode = XpxItemcustXrefAttributes

.getNamedItem("CustomerUnit");

String customerUnit = customerUnitNode.getTextContent();

Node ConvFactorNode = XpxItemcustXrefAttributes

.getNamedItem("ConvFactor");

String ConvFactor = ConvFactorNode.getTextContent();

if (ExtnIsCustUOMExcl != null && ExtnIsCustUOMExcl.equals("Y")) {

htBaseUOMConv.clear();

htBaseUOMConv.put(customerUnit, ConvFactor);

return;

}

if (useOrderMulUOMFlag.equals("Y")) {

int conversion = getConversion(ConvFactor, orderMultiple);

if (conversion != -1 && customerUnit != null

&& customerUnit.length() > 0) {

if (currentConversion == 0

|| (currentConversion != 0 && conversion < currentConversion)) {

lowestConvUOM = customerUnit;

currentConversion = conversion;

}

}

}

htBaseUOMConv.put(customerUnit, ConvFactor);

}

}

private void handleAternateUOMs(Node itemNodeChild,

HashMap<String, String> htBaseUOMConv) {

NodeList AternateUOMList = itemNodeChild.getChildNodes();

int length2 = AternateUOMList.getLength();

for (int k = 0; k < length2; k++) {

Node AlternateUOM = AternateUOMList.item(k);

NamedNodeMap namedNodeMap = AlternateUOM.getAttributes();

Node IsOrderingUOMNode = namedNodeMap.getNamedItem("IsOrderingUOM");

String IsOrderingUOM = IsOrderingUOMNode.getTextContent();

if (IsOrderingUOM != null && IsOrderingUOM.equals("Y")) {

Node unitOfMeasureNode = namedNodeMap

.getNamedItem("UnitOfMeasure");

String unitOfMeasure = unitOfMeasureNode.getTextContent();

Node quantityNode = namedNodeMap.getNamedItem("Quantity");

String quantity = quantityNode.getTextContent();

htBaseUOMConv.put(unitOfMeasure, quantity);

}

}

}

private boolean isCustomerMatching(NamedNodeMap xpxItemcustXrefAttributes,

String LegacyCustomerNumber, String customerDivision) {

Node CustomerNumberNode = xpxItemcustXrefAttributes

.getNamedItem("CustomerNumber");

String customerNumber = CustomerNumberNode.getTextContent();

Node customerBranchNode = xpxItemcustXrefAttributes

.getNamedItem("CustomerBranch");

String customerBranch = customerBranchNode.getTextContent();

if ((customerNumber.equals(LegacyCustomerNumber) && customerBranch

.equals(customerDivision))) {

return true;

}

return false;

}

private Document getOutputDocument(HashMap<String, String> htBaseUOMConv,

String lowestConvUOM) {

YFCDocument inputDocument = YFCDocument.createDocument("UOMList");

YFCElement documentElement = inputDocument.getDocumentElement();

if (lowestConvUOM != null && lowestConvUOM.length() > 0) {

YFCElement uOMElement = documentElement.createChild("UOM");

uOMElement.setAttribute("UnitOfMeasure", lowestConvUOM);

uOMElement.setAttribute("Conversion", htBaseUOMConv

.get(lowestConvUOM));

}

Set<String> set = htBaseUOMConv.keySet();

Iterator<String> iterator = set.iterator();

while (iterator.hasNext()) {

YFCElement uOMElement = documentElement.createChild("UOM");

String UnitOfMeasure = (String) iterator.next();

String Conversion = htBaseUOMConv.get(UnitOfMeasure);

if (!UnitOfMeasure.equals(lowestConvUOM)) {

uOMElement.setAttribute("UnitOfMeasure", UnitOfMeasure);

uOMElement.setAttribute("Conversion", Conversion);

}

}

return inputDocument.getDocument();

}

private int getConversion(String convFactor, String orderMultiple) {

if (convFactor != null && convFactor.length() > 0

&& orderMultiple != null && orderMultiple.length() > 0) {

double convFactorD = Double.parseDouble(convFactor);

double orderMultipleD = Double.parseDouble(orderMultiple);

double factor = (convFactorD / orderMultipleD);

if (Math.abs(factor) == factor) {

return (int) Math.abs(factor);

}

}

return -1;

}

private String getOrderMultipleValue(Node extnChild, String companyCode,

String customerDiv) {

String orderMultiple = "";

NodeList XpxItemExtnList = extnChild.getChildNodes();

int length3 = XpxItemExtnList.getLength();

for (int m = 0; m < length3; m++) {

Node XpxItemExtn = XpxItemExtnList.item(m);

NamedNodeMap XpxItemExtnAttributes = XpxItemExtn.getAttributes();

Node companyCodeNode = XpxItemExtnAttributes

.getNamedItem("CompanyCode");

String companyCodeL = companyCodeNode.getTextContent();

Node XPXDivisionNode = XpxItemExtnAttributes

.getNamedItem("XPXDivision");

String XPXDivision = XPXDivisionNode.getTextContent();

if (companyCodeL.equals(companyCode)

&& XPXDivision.equals(customerDiv)) {

Node orderMultipleNode = XpxItemExtnAttributes

.getNamedItem("OrderMultiple");

orderMultiple = orderMultipleNode.getTextContent();

}

}

return orderMultiple;

}

private HashMap<String, String> getCustomerDetails(YFSEnvironment env,

Document inXML) throws YIFClientCreationException, YFSException,

RemoteException {

HashMap<String, String> customerDetails = new HashMap<String, String>();

Element documentElement = inXML.getDocumentElement();

String customerID = documentElement.getAttribute("CustomerID");

String storeFrontId = documentElement.getAttribute("OrganizationCode");

YFCDocument inputDocument = YFCDocument.createDocument("Customer");

YFCElement inputElement = inputDocument.getDocumentElement();

inputElement.setAttribute("CustomerID", customerID);

inputElement.setAttribute("OrganizationCode", storeFrontId);

env.setApiTemplate("getCustomerList", SCXmlUtil.createFromString(""

+ "<CustomerList><Customer><Extn>"

+ "</Extn></Customer></CustomerList>"));

api = YIFClientFactory.getInstance().getApi();

Document outputListDocument = api.invoke(env, "getCustomerList",

inputDocument.getDocument());

Element outputListElement = outputListDocument.getDocumentElement();

NodeList wNodeList = outputListElement.getChildNodes();

int length = wNodeList.getLength();

for (int i = 0; i < length; i++) {

Node customerNode = wNodeList.item(i);

NodeList customerChildNodes = customerNode.getChildNodes();

int length1 = customerChildNodes.getLength();

for (int j = 0; j < length1; j++) {

Node customerChildNode = customerChildNodes.item(j);

String companyCode = "";

String customerDivision = "";

String useOrderMulUOMFlag = "";

if (customerChildNode.getNodeName().equals("Extn")) {

NamedNodeMap extnAttributes = customerChildNode

.getAttributes();

Node ExtnCompanyCodeNode = extnAttributes

.getNamedItem("ExtnCompanyCode");

if (ExtnCompanyCodeNode != null) {

companyCode = ExtnCompanyCodeNode.getTextContent();

if (companyCode != null

&& companyCode.trim().length() > 0) {

customerDetails.put("companyCode", companyCode);

}

}

Node ExtnCustomerDivisionNode = extnAttributes

.getNamedItem("ExtnCustomerDivision");

if (ExtnCustomerDivisionNode != null) {

customerDivision = ExtnCustomerDivisionNode

.getTextContent();

if (customerDivision != null

&& customerDivision.trim().length() > 0) {

customerDetails.put("customerDivision",

customerDivision);

}

}

Node ExtnUseOrderMulUOMFlagNode = extnAttributes

.getNamedItem("ExtnUseOrderMulUOMFlag");

if (ExtnUseOrderMulUOMFlagNode != null) {

useOrderMulUOMFlag = ExtnUseOrderMulUOMFlagNode

.getTextContent();

if (useOrderMulUOMFlag != null

&& useOrderMulUOMFlag.trim().length() > 0) {

customerDetails.put("useOrderMulUOMFlag",

useOrderMulUOMFlag);

}

}

}

}

}

env.clearApiTemplate("getItemList");

return customerDetails;

}

}

For web channel, a utility is provided in **XPEDXOrderUtils** to which will call this service to get the list of UOMs. Here is the utility method. Here.

public static HashMap<String, String> **getXpedxUOMList**(

String customerID, String ItemID, String StoreFrontID) {

HashMap<String, String> wUOMsAndConFactors = new HashMap<String, String>();

YFSEnvironment env=null;

try {

YFCDocument inputDocument = YFCDocument.createDocument("UOMList");

YFCElement documentElement = inputDocument.getDocumentElement();

documentElement.setAttribute("ItemID", ItemID);

documentElement.setAttribute("CustomerID", customerID);

documentElement.setAttribute("OrganizationCode", StoreFrontID);

IWCContext context = WCContextHelper

.getWCContext(ServletActionContext.getRequest());

SCUIContext wSCUIContext = context.getSCUIContext();

env = (YFSEnvironment) wSCUIContext

.getTransactionContext(true)

.getTransactionObject(

SCUITransactionContextFactory.YFC\_TRANSACTION\_OBJECT);

YIFApi api = YIFClientFactory.getInstance().getApi();

env.setApiTemplate("XPXUOMListAPI", SCXmlUtil.createFromString(""

+ "<UOMList><UOM>"

+ "</UOM></UOMList>"));

Document outputListDocument =api.executeFlow(env, "XPXUOMListAPI", inputDocument.getDocument());

Element wElement = outputListDocument.getDocumentElement();

NodeList wNodeList = wElement.getChildNodes();

if (wNodeList != null) {

int length = wNodeList.getLength();

for (int i = 0; i < length; i++) {

Node wNode = wNodeList.item(i);

if (wNode != null) {

NamedNodeMap nodeAttributes = wNode.getAttributes();

if (nodeAttributes != null) {

Node UnitOfMeasure = nodeAttributes

.getNamedItem("UnitOfMeasure");

Node Conversion = nodeAttributes

.getNamedItem("Conversion");

if (UnitOfMeasure != null && Conversion != null) {

wUOMsAndConFactors.put(UnitOfMeasure

.getTextContent(), Conversion

.getTextContent());

}

}

}

}

}

} catch (Exception ex) {

log.error(ex.getMessage());

}

env.clearApiTemplate("XPXUOMListAPI");

return wUOMsAndConFactors;

}

# Unit test cases

1. Checked customer’s preferred UOM, then only system retrieves only preferred UOMs
2. Checked by giving CustomerID in the format of <Customer Div>-<Legacy Customer Number>-Suffix and tested it whether system is honors customer division, legacy number while retrieving UOM list.
3. Checked by enabling Order Multiple flag for unit of measure and its lowest conversion factor. Verified that this unit of measure is on the top of the retrieval list.
4. Checked whether alternate UOMs for that item are retrieved successfully.
5. Checked whether web channel utility is successfully able to retrieve the list of UOMs