Open Catalog Interface

Release 2.0B

SAP
## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚨</td>
<td>Caution</td>
</tr>
<tr>
<td>🗣️</td>
<td>Example</td>
</tr>
<tr>
<td>💡</td>
<td>Note</td>
</tr>
<tr>
<td>🔍</td>
<td>Recommendation</td>
</tr>
<tr>
<td>🔧</td>
<td>Syntax</td>
</tr>
<tr>
<td>💡</td>
<td>Tip</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Open Catalog Interface</td>
<td>5</td>
</tr>
<tr>
<td>Open Catalog Interface: Outbound Section</td>
<td>6</td>
</tr>
<tr>
<td>Open Catalog Interface: Inbound Section</td>
<td>9</td>
</tr>
<tr>
<td>Example of a Catalog Interface</td>
<td>14</td>
</tr>
</tbody>
</table>
Open Catalog Interface

Purpose

The Open Catalog Interface (OCI) is the interface between catalogs and SAP Business-to-Business Procurement. SAP’s Open Catalog Interface uses standard Internet protocols. It has already been implemented between several catalog applications and SAP Business-to-Business Procurement.

Integration

The graphic below shows how catalogs are integrated with SAP Business-to-Business Procurement.

Structure

The catalog interface consists of two separate and distinct sections: the outbound section and the inbound section.

- The **outbound section** defines the information being sent from the SAP Business-to-Business Procurement application to the catalog application. This includes information such as the catalog URL and login data.
- The **inbound section** consists of the information being sent from the catalog application to the SAP Business-to-Business Procurement application. This section contains data on the items selected in the catalog, such as the items descriptions, quantities ordered, and prices. For more details, see the interface diagram under **Inbound Section**.
Open Catalog Interface: Outbound Section

The following graphic shows how the purchasing application redirects the browser from SAP Business-to-Business Procurement to the catalog using the parameters of the outbound section:

The outbound section consists of information that is sent to the catalog application by the SAP Business-to-Business Procurement application. This information originates in the SAP system, where it is created and maintained using SAP transaction SM34, view cluster VC_CATALOG. You also have to define user attribute CAT to make the catalog visible for the user in the SAP Business-to-Business Procurement application.

Using this information, the SAP Business-to-Business Procurement application constructs a URL call to the catalog application and redirects the browser to this URL. The CGI script of the catalog application then has to parse and decode this information.

In the SAP Business-to-Business Procurement system, you can set up this information in Customizing.

Path in the SAP Implementation Guide (IMG):

*Business-to-Business Procurement → Define Catalogs.*

Note the following when setting up this information:

- Many of the field names and values are dependent on the particular catalog application.
- Every field contains a name and a value (*Contents* column) and has a type. Possible types are:
  - URL
  - SAP field
  - Fixed value
  - Field from service
  - Return URL
- You must structure the fields in the order shown in the table below. In particular, define the catalog-specific information directly after the catalog URL and the ~OkCode and ~Target, and ~Caller fields directly after the Return URL field.
- Typically, field values are entered as literals. For example, the LOGIN field can have a literal value of GUEST. However, system variables such as SY-UNAME can also be used...
as values for a field, allowing the actual value to be determined at runtime. In this case, you should assign the type *SAP field* to the relevant field.

- As of Release 2.0, the name/value pair OCI_VERSION="2.0" is passed to the catalog. This indicates that the SAP B2B Procurement system involved is a Release 2.0 system.

The outbound data you maintain is structured as described in the table below. In this table, *Fixed* indicates that the field name must be exactly as specified, whereas *Variable* indicates that the field name is catalog-specific.

<table>
<thead>
<tr>
<th>Description</th>
<th>Mandatory</th>
<th>Catalog-specific</th>
<th>Field Name</th>
<th>Field Name is Fixed/Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalog URL</td>
<td>Yes</td>
<td>Yes</td>
<td>&lt;blank&gt;</td>
<td>Fixed</td>
<td>The URL of the catalog. This should refer to the location of the catalog CGI script file.</td>
</tr>
<tr>
<td>All catalog-specific fields</td>
<td>As relevant</td>
<td>Yes</td>
<td></td>
<td>Variable</td>
<td>The set of catalog-specific fields. An example of catalog-specific fields is provided in the table below.</td>
</tr>
<tr>
<td>Return URL</td>
<td>Yes</td>
<td>No</td>
<td>HOOK_URL</td>
<td>Variable</td>
<td>The URL used to return to the SAP Business-to-Business Procurement application from the catalog application. Set the value of this field to blank. It is automatically filled at runtime by the procurement application. (See also Note below.)</td>
</tr>
<tr>
<td>OK Code</td>
<td>Yes</td>
<td>No</td>
<td>~OkCode</td>
<td>Fixed</td>
<td>Contains the transaction code indicating that the function Add Items to SAP shopping basket is to be performed. Must be set to ADDI for SAP Business-to-Business Procurement.</td>
</tr>
<tr>
<td>Target</td>
<td>Yes</td>
<td>No</td>
<td>~TARGET</td>
<td>Fixed</td>
<td>Specifies the frame to which a catalog is to return in a frame-based environment. If this field is not set, the catalog application must provide a default target of _top.</td>
</tr>
<tr>
<td>Caller</td>
<td>Yes</td>
<td>No</td>
<td>~CALLER</td>
<td>Fixed</td>
<td>Indicates that the data was sent by an external catalog. Content must be set to CTLG.</td>
</tr>
</tbody>
</table>

The HOOK_URL can have a different name, but the type of this field must be set to Return URL. The HOOK_URL is encoded so that special characters such as ':' and '/' are represented by '%' and the hex digits for the ASCII code for the characters. The catalog application is responsible for decoding this string into a valid URL. Note that the catalog application must not make any assumptions about the HOOK_URL, for example, it cannot assume that a question mark is present to delineate the URL variables.
Example of catalog-specific parameters:

<table>
<thead>
<tr>
<th>Description</th>
<th>Field Name</th>
<th>Field Name is Fixed/Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request type</td>
<td>REQTYPE</td>
<td>Variable</td>
<td>Requests a login when the catalog is first accessed.</td>
</tr>
<tr>
<td>Login</td>
<td>USERNAME</td>
<td>Variable</td>
<td>The user ID for logging in to the catalog.</td>
</tr>
<tr>
<td>Password</td>
<td>PASSWORD</td>
<td>Variable</td>
<td>The login password.</td>
</tr>
<tr>
<td>Database identifier</td>
<td>SERVICE</td>
<td>Variable</td>
<td>The database identifier. This may be required for some catalogs.</td>
</tr>
</tbody>
</table>

**Additional Catalog Functions**

An external catalog may support additional functionality. This additional functionality can be triggered by several name/value pairs which are passed over to the catalog within the query string of the catalog URL when the functionality is needed. The following functions are possible:

- **Detail of a product**
  
  The following name/value pairs are additionally transferred to the catalog:
  
  FUNCTION="DETAIL" and
  
  PRODUCTID="database key of product in the catalog"

- **Validation of a product**
  
  The following name/value pairs are additionally transferred to the catalog:
  
  FUNCTION="VALIDATE" and
  
  PRODUCTID="database key of product in the catalog"

- **Sourcing of a product**
  
  The following name/value pairs are additionally transferred to the catalog:
  
  FUNCTION="VALIDATE" and
  
  SEARCHSTRING="string to directly start the catalog search" and
  
  VENDOR="vendornumber in the buyer's system"

  The functions "DETAIL" and "VALIDATE" will only work if the parameter NEW_ITEM-EXT_PRODID[n] was filled from the catalog in a previous call.

If a catalog supports one or more of these three additional functions, the corresponding flag has to be set in the SAP transaction SM34, view cluster VC_CATALOG. Path in the SAP Implementation Guide (IMG): Business-to-Business Procurement → Define Catalogs.

The expected data to be returned from the catalog is described in this document under **Inbound Section** [Page 8]
Open Catalog Interface: Inbound Section

Use

The inbound section consists of information that is sent to the SAP Business-to-Business Procurement application by the catalog application.

For each item selected in the catalog and sent to the SAP Business-to-Business Procurement application, all mandatory fields have to be sent, along with the relevant optional fields.

The fields can be sent back to the purchasing application by either the GET or the POST method (because of the limitations of GET, POST is strongly recommended).

Note the following:

- Fields should not contain leading spaces.
- All numeric fields that allow fractional amounts must explicitly include the decimal point ".” at the correct byte position.
- Apart from the decimal point, numeric fields may not include commas or any other non-numeric characters.
- As of Release 2.0, no distinction is made between a service master and a material master. Both will be stored as a product master.
- As of Release 2.0 the inbound section can be sent back to the SAP Business-to-Business Procurement system as follows:
  - As an HTML version (as in Release 1.0)
  - As an XML version

HTML version

The following graphic illustrates the data flow of the inbound section:

To handle multiple items selected in the catalog, each field name must be an array using the bracket subscript convention as per the C programming language. Subscripts must start at 1 (not zero). For example, the description of an item would be returned as a name such as "NEW_ITEM-DESCRIPTION [1]" (refer to the Example of a Catalog Interface [Page 14]).
<table>
<thead>
<tr>
<th>Item</th>
<th>Name</th>
<th>Required/Optional</th>
<th>Details</th>
<th>Type Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>NEW_ITEM-DESCRIPTION[n]</td>
<td>Required if the SAP product master number is not specified.</td>
<td>The description of the item to be ordered</td>
<td>CHAR - 40</td>
</tr>
<tr>
<td>Product master</td>
<td>NEW_ITEM-MATNR[n]</td>
<td>Required if the Description field is not specified.</td>
<td>The SAP product master number</td>
<td>CHAR - 18</td>
</tr>
<tr>
<td>Product group</td>
<td>NEW_ITEM-MATGROUP[n]</td>
<td>Optional</td>
<td>The SAP product group</td>
<td>CHAR - 10</td>
</tr>
<tr>
<td>Quantity</td>
<td>NEW_ITEM-QUANTITY[n]</td>
<td>Required</td>
<td>Quantity to add. There can be a maximum of 11 digits to the left of the decimal point, and there must be 3 digits to the right.</td>
<td>CHAR - 15</td>
</tr>
<tr>
<td>Unit of measure</td>
<td>NEW_ITEM-UNIT[n]</td>
<td>Required if product master field is not filled</td>
<td>Unit of measure of the item. Must be the standard ISO code.</td>
<td>CHAR - 3</td>
</tr>
<tr>
<td>Price</td>
<td>NEW_ITEM-PRICE[n]</td>
<td>Optional</td>
<td>Price of the item in the catalog. There can be a maximum of 11 digits to the left of the decimal point, and there must be 3 digits to the right.</td>
<td>CHAR - 15</td>
</tr>
<tr>
<td>Price unit</td>
<td>NEW_ITEM-PRICEUNIT[n]</td>
<td>Optional</td>
<td>The number of units that must be purchased at the given price. This quantity defaults to one.</td>
<td>CHAR - 9</td>
</tr>
<tr>
<td>Currency</td>
<td>NEW_ITEM-CURRENCY[n]</td>
<td>Required if a price is returned, otherwise optional</td>
<td>Must be the ISO code for the currency.</td>
<td>CHAR - 5</td>
</tr>
<tr>
<td>Lead time</td>
<td>NEW_ITEM-LEADTIME[n]</td>
<td>Optional</td>
<td>Number of days from today until the product will be available. If not specified, no assumptions will be made about the lead time.</td>
<td>CHAR - 5</td>
</tr>
<tr>
<td>Vendor</td>
<td>NEW_ITEM-VENDOR[n]</td>
<td>Optional</td>
<td>The business partner number in the SAP B2B Procurement system, if the product is purchased through a multi-supplier catalog.</td>
<td>CHAR - 10</td>
</tr>
<tr>
<td>Item</td>
<td>Name</td>
<td>Required/Optional</td>
<td>Details</td>
<td>Type Length</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------</td>
<td>------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Vendor product number</td>
<td>NEW_ITEM-VENDORMAT[n]</td>
<td>Optional</td>
<td>The vendor product part number of the product</td>
<td>CHAR - 22</td>
</tr>
<tr>
<td>Manufacturer's code</td>
<td>NEW_ITEM-MANUFACTCODE[n]</td>
<td>Optional</td>
<td>The manufacturer's code</td>
<td>CHAR - 10</td>
</tr>
<tr>
<td>Manufacturer's product number</td>
<td>NEW_ITEM-MANUFACTMAT[n]</td>
<td>Optional</td>
<td>The manufacturer's product part number of the product.</td>
<td>CHAR - 40</td>
</tr>
<tr>
<td>Contract number</td>
<td>NEW_ITEM-CONTRACT[n]</td>
<td>Optional</td>
<td>The number of the contract with the vendor.</td>
<td>CHAR - 10</td>
</tr>
<tr>
<td>Item of a contract</td>
<td>NEW_ITEM-CONTRACT_ITEM[n]</td>
<td>Optional</td>
<td>The number of an item within a contract with the vendor.</td>
<td>CHAR - 5</td>
</tr>
<tr>
<td>Service flag</td>
<td>NEW_ITEM-SERVICE[n]</td>
<td>Optional</td>
<td>A flag which indicates if the line refers to a service or to goods.</td>
<td>CHAR - 1</td>
</tr>
<tr>
<td>Quotation</td>
<td>NEW_ITEM-EXT_QUOTE_ID[n]</td>
<td>Optional</td>
<td>A reference to an external quotation.</td>
<td>CHAR - 35</td>
</tr>
<tr>
<td>Quotation item</td>
<td>NEW_ITEM-EXT_QUOTE_ITEM[n]</td>
<td>Optional</td>
<td>A reference to an external quotation item.</td>
<td>CHAR - 10</td>
</tr>
<tr>
<td>Product ID</td>
<td>NEW_ITEM-EXT_PRODUCT_ID[n]</td>
<td>Optional</td>
<td>Key to identify a product in the catalog</td>
<td>CHAR - 40</td>
</tr>
<tr>
<td>Description</td>
<td>NEW_ITEM-LONGTEXT_n:132[] (see Note below)</td>
<td>Optional</td>
<td>Description of a configuration, for example.</td>
<td>CHAR - No restriction</td>
</tr>
<tr>
<td>Customer-specific field</td>
<td>NEW_ITEM-CUST_FIELD1[n]</td>
<td>Optional</td>
<td>Customer-specific field, which may be handled in a business add-in in the SAP Business-to-Business Procurement system</td>
<td>CHAR - 10</td>
</tr>
<tr>
<td>Customer-specific field</td>
<td>NEW_ITEM-CUST_FIELD2[n]</td>
<td>Optional</td>
<td>As above</td>
<td>CHAR - 10</td>
</tr>
<tr>
<td>Customer-specific field</td>
<td>NEW_ITEM-CUST_FIELD3[n]</td>
<td>Optional</td>
<td>As above</td>
<td>CHAR - 10</td>
</tr>
<tr>
<td>Customer-specific field</td>
<td>NEW_ITEM-CUST_FIELD4[n]</td>
<td>Optional</td>
<td>As above</td>
<td>CHAR - 20</td>
</tr>
<tr>
<td>Customer-specific field</td>
<td>NEW_ITEM-CUST_FIELD5[n]</td>
<td>Optional</td>
<td>As above</td>
<td>CHAR - 50</td>
</tr>
</tbody>
</table>

The field NEW_ITEM-LONGTEXT forms an exception. In this case, the index must be attached with an underscore followed by 132 and empty brackets: NEW_ITEM-LONGTEXT_n:132[] (see also the Example of a Catalog Interface [Page 14]).

The fields shown in **bold type** are new with Release 2.0B.
The functions DETAIL and VALIDATE will only work if the parameter NEW_ITEM-EXT_PRODUCT_ID[n] was filled from the catalog in a previous call. The index of the line items is shown as 'n' in the table above.

The content of the five customer fields can be handled using business add-in BBP_CATALOG_TRANSFER. For further documentation on how to develop business add-ins, see:


**XML Version**

With Release 2.0 the interface is also able to process XML as input. The prerequisite for this is that you install the SAP Business Connector on the receiving side, which is used as a mapping tool.

The following graphic illustrates the data flow of the inbound interface:

If XML is used, the XML file has to be passed over to the browser in an HTML form in an input field with the name ~xmlDocument. (This should have the type hidden.) The content of the generated XML file is expected to be bound to the input field ~xmlDocument as a base64-encoded string (see the Example of a Catalog Interface [Page 14]). The encoding can be done directly on the server side, or, as in the example, on the client side.

In addition the type of XML to be transferred has to be passed over in another input field. The name of this field is ~xml_Type. This variable is used to distinguish between several XML schemas on the SAP Business Connector. Possible values for this field are: ESAPO (Encoded SAP Object) and ERNO (Encoded RosettaNet Object). To see the SAP XML schema, refer to the file pdi_oci.xsd at http://www.sap.com/csp/scenarios SAP B2B Procurement B2B-OCI.
### Additional Catalog Functions

The table below shows what data is expected to be returned to the catalog if one of the additional functions is called:

<table>
<thead>
<tr>
<th>Function</th>
<th>Data To Be Returned From The Catalog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail of a product</td>
<td>No data has to be returned from the catalog. The detail view of the product has to be shown in the browser.</td>
</tr>
<tr>
<td>Validation of a product</td>
<td>The application expects the complete product data to be returned. If the product no longer exists, the catalog is not expected to return any data. Data transfer must be done in the background.</td>
</tr>
<tr>
<td>Sourcing of a product</td>
<td>The catalog is expected to transfer the data in the <em>normal</em> way.</td>
</tr>
</tbody>
</table>
Example of a Catalog Interface

The following excerpts are examples of the HTML and XML source code generated by a catalog engine after a user has finished selecting items for a shopping basket. You can extract the source code and display the data in a Web browser.

**HTML version**

In the HTML version, all the line items are placed as name/value pairs in an HTML form as shown below. This can be done as text or hidden fields. The parameter `HOOK_URL` has to be placed in the action parameter of that form, so that the HTML form can be sent to the SAP Business-to-Business Procurement system.

**HTML Form (interface.asp):**

```html
<HTML>
<HEAD>
</HEAD>

<H1>SAP Open Catalog Interface: HTML - Example</H1>

This is an example, what needs to be generated by the catalog engine, if the user is done with picking items in his shopping basket.

View the source of this file to see all the text inputs.

To view this page in a Web browser, add the following line into the browser address bar:
```
The best pen in our program, lightweight and easy to handle.

Palm Pilot, the second item.
The award-winning Palm Computing organizers, designed as companion products to personal computers, enable mobile users to manage their schedules, contacts and other critical personal and business information on their desktops and remotely. Palm Computing organizers automatically synchronize their information's with a personal computer locally or over a local or wide area network at the touch of a button. Their most distinguishing features include shirt-pocket size, instant response, an elegant graphical user interface and an innovative desktop docking cradle which facilitates two way synchronization between the PC and organizer.
The award-winning Palm Computing organizers, designed as companion products to personal computers, enable mobile users to manage their schedules, contacts and other critical personal and business information on their desktops and remotely. Palm Computing organizers automatically synchronize their information's with a personal computer locally or over a local or wide area network at the touch of a button. Their most distinguishing features include shirt-pocket size, instant response, an elegant graphical user interface and an innovative desktop docking cradle which facilitates two way synchronization between the PC and organizer.

Palm Pilot, the fourth item
The award-winning Palm Computing organizers, designed as companion products to personal computers, enable mobile users to manage their schedules, contacts and other critical personal and business information on their desktops and remotely. Palm Computing organizers automatically synchronize their information's with a personal computer locally or over a local or wide area network at the touch of a button. Their most distinguishing features include shirt-pocket size, instant response, an elegant graphical user interface and an innovative desktop docking cradle which facilitates two way synchronization between the PC and organizer.

Cleaning the Palm Pilot, the fifth item
The award-winning Palm Computing organizers, designed as companion products to personal computers, enable mobile users to manage their schedules, contacts and other critical personal and business information on their desktops and remotely. Palm Computing organizers automatically synchronize their information's with a personal computer locally or over a local or wide area network at the touch of a button. Their most distinguishing features include shirt-pocket size, instant response, an elegant graphical user interface and an innovative desktop docking cradle which facilitates two way synchronization between the PC and organizer.
The award-winning Palm Computing organizers, designed as companion products to personal computers, enable mobile users to manage their schedules, contacts and other critical personal and business information on their desktops and remotely. Palm Computing organizers automatically synchronize their information with a personal computer locally or over a local or wide area network at the touch of a button. Their most distinguishing features include shirt-pocket size, instant response, an elegant graphical user interface and an innovative desktop docking cradle which facilitates two way synchronization between the PC and organizer.

In the XML version, the data from a shopping basket can be generated into a separate file which is then linked to the HTML form. The XML data is expected to be in the value of an input field with the name ~xmlDocument. It can be set there dynamically using the function SAP_encode_b64(Str), as shown in the example below. In the input field ~xml_Type, you have to specify the type of schema used to generate the XML data. Possible values for this field are: ESAPO (Encoded SAP Object) and ERNO (Encoded RosettaNet Object). To view the SAP XML schema refer to the file pdi_oci.xsd in SAPNet. To access this files, enter the URL http://www.sap.com/, specifying the alias CSP/scenarios/ and choose the menu options SAP B2B Procurement → B2B-OCI.

**XML version**

In the XML version, the data from a shopping basket can be generated into a separate file which is then linked to the HTML form. The XML data is expected to be in the value of an input field with the name ~xmlDocument. It can be set there dynamically using the function SAP_encode_b64(Str), as shown in the example below. In the input field ~xml_Type, you have to specify the type of schema used to generate the XML data. Possible values for this field are: ESAPO (Encoded SAP Object) and ERNO (Encoded RosettaNet Object). To view the SAP XML schema refer to the file pdi_oci.xsd in SAPNet. To access this files, enter the URL http://www.sap.com/, specifying the alias CSP/scenarios/ and choose the menu options SAP B2B Procurement → B2B-OCI.

**HTML Form (interface.asp)**

```html
<HTML>
<script language="JavaScript1.2">
function SAP_encode_b64(Str) {
    var encStr = "";
    var base64 = [ 'A','B','C','D','E','F','G','H','I','J','K','L','M',
                   'N','O','P','Q','R','S','T','U','V','W','X','Y','Z',
                   'a','b','c','d','e','f','g','h','i','j','k','l','m',
                   'n','o','p','q','r','s','t','u','v','w','x','y','z',
                   '+','/'];
    for (var i = 0; i < Str.length; i++) {
        var c = Str.charCodeAt(i);
        if (c < 32) encStr += base64[c];
        else encStr += base64[(c >> 2) & 63] + base64[(c >> 4) & 63] + base64[(c >> 6) & 63] + base64[c & 63];
    }
    return encStr;
}
</script>

<FORM>
<input type="hidden" name="NEW_ITEM-LONGTEXT_6:132[]" value = "longtext_6:
2.1">
<input type="hidden" name="NEW_ITEM-CUST_FIELD1[6]" value = "custf6.1">
<input type="hidden" name="NEW_ITEM-CUST_FIELD2[6]" value = "custf6.2">
<input type="hidden" name="NEW_ITEM-CUST_FIELD3[6]" value = "custf6.3">
<input type="hidden" name="NEW_ITEM-CUST_FIELD4[6]" value = "custf6.4">
<input type="hidden" name="NEW_ITEM-CUST_FIELD5[6]" value = "custf6.5">
<!--and so on...-->

<input type="submit" value="Transfer Items to B2B shopping basket" id=submit1 name=submit1><br>
</FORM>

© 2000, SAP AG
</BODY>
</HTML>
```
for (var i = 0; i < Str.length; i += 3) {
    encStr += base64[(Str.charCodeAt(i) >>> 2)];
    if(!Str.charAt(i+1)) {encStr += '=='; break;}
    encStr += base64[((Str.charCodeAt(i) & 0x03) << 4) | Str.charCodeAt(i+1) >>> 4]);
    if(!Str.charAt(i+2)) {encStr += '='; break;}
    encStr += base64[((Str.charCodeAt(i+1) & 0x0F) << 2) | Str.charCodeAt(i+2) >>> 6]);
    encStr += base64[(Str.charCodeAt(i+2) & 0x3F)];
}
return encStr;
}
</script>

<HEAD>
<H1>SAP Open Catalog Interface: XML - Example</H1>
</HEAD>

<P>This is an example, what needs to be generated by the catalog engine, if the user is done with picking items in his shopping basket.</P>
<P>View the source of the file 'example1.xml' to see all the items.</P>

<!-- insert the order -->
<BODY bgcolor="#DED6C2">
<XML ID=xmlid src="example1.xml"></XML>
<!-- Transfer the order to the B2B Application -->
<form action="<%=Request.QueryString("HOOK_URL")%>" method="post"
name="OrderForm" onSubmit="OrderForm['~xmlDocument'].value =
SAP_encode_b64(xmlid.xml)">

    <input type="hidden" name="~xmlDocument" value="" />
    <input type="hidden" name="~xml_type" value="ESAPO">
    <input type="submit" value="Transfer Items to B2B shopping basket"
id=submit1 name=submit1>
</form>

© 2000, SAP AG
</BODY>
</HTML>

XML data (example1.xml)
<?xml version ="1.0"?>
<BusinessDocument>
  <CatalogHeader/>
</CatalogHeader>

<Catalog>
  <CatalogID>11</CatalogID>
  <Product ProductType = "Good">
    <CatalogKey>11111</CatalogKey>
    <ParentCategoryID>44120000</ParentCategoryID>
    <Description Language = "EN">Pen, Easytouch, Medium, Black</Description>
    <ShoppingBasketItem RefVendorDescription = "0"
      RefManufacturerDescription = "1">
      <Quantity UoM = "EA">1</Quantity>
      <NetPrice>
        <Price Currency = "USD">0.50</Price>
        <PriceUnit>5</PriceUnit>
      </NetPrice>
      <LeadTime>1</LeadTime>
      <Quote>
        <QuoteID>111</QuoteID>
        <QuoteItemID>1111</QuoteItemID>
      </Quote>
      <ItemText Language = "EN">
        The best pen in our program, lightweight and easy to handle
      </ItemText>
    </ShoppingBasketItem>
    <ManufacturerDescription ID = "1">
      <PartnerProductID Code = "Other">4712</PartnerProductID>
      <PartnerID Code = "Other">4711</PartnerID>
    </ManufacturerDescription>
    <VendorDescription ID = "0">
      <PartnerProductID Code = "Other">648570</PartnerProductID>
      <PartnerID Code = "Other">1768</PartnerID>
      <LeadTime>1</LeadTime>
      <BuyerContract>
        <ContractID>111</ContractID>
        <ContractItemID>1111</ContractItemID>
      </BuyerContract>
    </VendorDescription>
  </Product>
  <Product ProductType = "Good">
    <ProductID Code = "Buyer">KB-SPEZ</ProductID>
    <CatalogKey>22222</CatalogKey>
  </Product>
</Catalog>
</BusinessDocument>
<Description Language = "EN">Palm Pilot, the second item</Description>

<ShoppingBasketItem RefVendorDescription = "2" RefManufacturerDescription = "3">
  <Quantity UoM = "EA">1</Quantity>
  <NetPrice>
    <Price Currency = "USD">225</Price>
    <PriceUnit>222</PriceUnit>
  </NetPrice>
  <LeadTime>1</LeadTime>
  <Quote>
    <QuoteID>222</QuoteID>
    <QuoteItemID>2222</QuoteItemID>
  </Quote>
  <ItemText Language = "EN">
    The award-winning Palm Computing organizers, designed as companion products to personal computers, enable mobile users to manage their schedules, contacts and other critical personal and business information on their desktops and remotely. Palm Computing organizers automatically synchronize their information with a personal computer locally or over a local or wide area network at the touch of a button. Their most distinguishing features include shirt-pocket size, instant response, an elegant graphical user interface and an innovative desktop docking cradle which facilitates two-way synchronization between the PC and organizer.
  </ItemText>
</ShoppingBasketItem>

<ManufacturerDescription ID = "3">
  <PartnerProductID Code = "Other">222</PartnerProductID>
  <PartnerID Code = "Other">2222</PartnerID>
</ManufacturerDescription>

<VendorDescription ID = "2">
  <PartnerProductID Code = "Other">12345</PartnerProductID>
  <PartnerID Code = "Other">1768</PartnerID>
  <LeadTime>1</LeadTime>
  <BuyerContract>
    <ContractID>222</ContractID>
    <ContractItemID>2222</ContractItemID>
  </BuyerContract>
</VendorDescription>

</Product>

<Product ProductType = "Good">
  <CatalogKey>33333</CatalogKey>
  <Description Language = "EN">Palm Pilot, the third item</Description>
  <ShoppingBasketItem RefVendorDescription = "4" RefManufacturerDescription = "5">
    <Quantity UoM = "EA">77</Quantity>
  </ShoppingBasketItem>
</Product>
The award-winning Palm Computing organizers, designed as companion products to personal computers, enable mobile users to manage their schedules, contacts, and other critical personal and business information on their desktops and remotely. Palm Computing organizers automatically synchronize their information with a personal computer locally or over a local or wide area network at the touch of a button. Their most distinguishing features include shirt-pocket size, instant response, an elegant graphical user interface and an innovative desktop docking cradle which facilitates two-way synchronization between the PC and organizer.