eBridge Connections Web Services

Overview

Authorized developers can use the eBridge Connections SOAP API to send or receive documents from businesses on eiCloud. This documentation page has instructions and code samples for many common use cases. Additional resources are available at the end of this document, including XSD files and a WSDL document.

Get Started

In order to get started you will need the username and password for an eBridge eiCloud account that has API access enabled. You can request API access or request an eiCloud account by contacting eBridge.

Below is an example script that you can use to test your eiCloud credentials with the eBridge web services. Replace `JaneDoe` with your eiCloud username (or login) and `abc123` with your eiCloud password.

- PHP

```
<?php

$client = new SoapClient("https://eicloudservice.com/eportalservice.asmx?
wsdl",array('trace' => true));

$params = array(

        'username'=>'JaneDoe',

        'password'=>'abc123'

        );

try {

        $client->GetDocTypeList($params);

        $response = $client->__getLastResponse();

        }

catch (Exception $e) {
```

```
        $response = 'Error: '.$e->getMessage();

        }

echo $response;
```

The `$response` variable should contain a list of the document types supported by eBridge. If you do not see the document types then it is possible that your eBridge account does not have API access enabled. Contact eBridge Connections support for assistance with enabling API access on your account.

Sending Documents to eiCloud

The SendFile() method is used for transmitting data to eiCloud. This method is commonly used to send orders to an ERP through eiCloud (for example, from a custom web store) however it can also be used to transmit product data, inventory reports, shipping information, customer information and other business documents.

When a valid document is successfully transmitted using `SendFile()` it will appear in the outbox of the sending account. You can view your eiCloud mailboxes by logging into eiCloud.

SendFile

`SendFile()` has four string parameters; all are required:

Parameter	Type	Description
`login`	string	eiCloud username of sender
`password`	string	eiCloud password of sender
`content`	string	XML document being sent
`filename`	string	A unique filename

Things you should know

- eiCloud users cannot send documents to themselves. In order to do this, please request a dummy sender user from eBridge support.
- You can only send a document to a recipient account after eBridge has registered you to trade documents with them and only then for the document types that you are registered to trade.
- All documents must have sender and recipient IDs in the XML file. You can find more information about these IDs in the sub-sections for orders, product data and inventory data.

Writing eBridge XML documents

eBridge XML is based on xCBL 4.0 (the XML Common Business Library.) The elements in the XML files you send to eBridge will be determined by the document type you wish to send and the applications that you are integrating. You can find schemas and sample XML files for each document type in the sections for orders, product data and inventory data.

Some elements are optional or only apply to a specific application. You can use the eBridge blueprint builder to look up the XPath associated with a specific field. Here's how:

1. Choose an accounting package using the picklists on the left or choose an eCommerce or CRM system using the buttons and picklists on the right
2. Click on the arrow for the touchpoint you are interested in
3. Click on the *View Data Elements* link (located below the arrows)

A pop-up will appear, showing you each field in the touchpoint you chose and the corresponding eBridge XML XPaths.

Errors

The response from `SendFile()` will be `true` or `false`. A response of `true` means that the document was received by eiCloud. It does not necessarily mean that the document was delivered to the recipient. You can verify whether or not the document was delivered by logging into eiCloud and checking your Outbox to see if the document is there.

If there was an error in processing or delivering your document, you may see the document in your list along with a short error message that explains what went wrong.

Sample code

Below are some sample scripts that use `SendFile()`:

- PHP
- C#

```
<?php

$client = new SoapClient("https://eicloudservice.com/eportalservice.asmx?
wsdl");

$xml = file_get_contents("example.xml");

$params = array(

        'login'=>'JaneDoe',

        'password'=>'abc123',

        'content' => $xml,
```

```
        'filename' => '123.xml'

        );

try {

    $response = $client->SendFile($params);

    }

catch (Exception $e) {

    $response = 'Error: '.$e->getMessage();

    }

var_dump($response);
```

Orders

Orders can be sent to eiCloud using `SendFile()`. The `content` parameter of `SendFile()` should be an XML document structured according to the Order Management XSD of the xCBL 4.0 standard with `<Order>` as the root node.

The fields required for a sales order can vary depending on the application that the order will be integrated with. You can see the sales order fields for your application/ERP and identify their corresponding XML elements using the *Data Elements* link from eBridge's XPath blueprint builder.

BuyerParty IDs and SellerParty IDs

Orders must contain BuyerParty IDs and SellerParty IDs. The BuyerParty is the sender of the order on eiCloud. This could be a buyer or in some cases it could represent a third party webstore. The SellerParty represents the recipient of the order (normally the seller, however it could also stand for a warehouse or other intermediate endpoint). You can retrieve these ISA IDs by logging into eiCloud with the sender account and navigating to Connection Settings. In the sidebar at left, click on the partner that you wish to send orders to. On this page, you will find a table labelled Envelopes:

The User ISA ID is the BuyerParty ID. The Partner ISA ID is the SellerParty ID. You can find a sample Order XML document below. In this sample file, the BuyerParty ID (**ACME123** in this example) is labelled in **red** and the SellerParty ID (**XYZCORP**) is labelled in **gold**.

Sample XML

```
<?xml version="1.0"?>

<Order xmlns="rrn:org.xcbl:schemas/xcbl/v4_0/ordermanagement/v1_0/
ordermanagement.xsd" xmlns:core="rrn:org.xcbl:schemas/xcbl/v4_0/core/
core.xsd" xmlns:dgs="http://www.w3.org/2000/09/xmldsig#" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance">

  <OrderHeader>

    <OrderNumber>

      <BuyerOrderNumber>123</BuyerOrderNumber>

    </OrderNumber>

    <OrderIssueDate>2013-05-11T08:51:16</OrderIssueDate>

    <OrderReferences/>

    <Purpose>

      <core:PurposeCoded>Other</core:PurposeCoded>

    </Purpose>

    <RequestedResponse>

      <core:RequestedResponseCoded>Other</core:RequestedResponseCoded>

    </RequestedResponse>

    <OrderType>

      <core:OrderTypeCoded>Other</core:OrderTypeCoded>

      <core:OrderTypeCodedOther>InboundOrder</core:OrderTypeCodedOther>

    </OrderType>

    <OrderCurrency>

      <core:CurrencyCoded>USD</core:CurrencyCoded>
```

```
  </OrderCurrency>
  <OrderLanguage>
    <core:LanguageCoded>en</core:LanguageCoded>
  </OrderLanguage>
  <OrderDates/>
  <OrderParty>
    <BuyerParty xsi:type="core:PartyType">
      <core:PartyID>
        <core:Ident>ACME123<</core:Ident>
      </core:PartyID>
      <core:ListOfIdentifier>
        <core:Identifier>
          <core:Ident>1</core:Ident>
        </core:Identifier>
      </core:ListOfIdentifier>
    </BuyerParty>
    <SellerParty xsi:type="core:PartyType">
      <core:PartyID>
        <core:Ident>XYZCORP<</core:Ident>
      </core:PartyID>
      <core:ListOfIdentifier>
        <core:Identifier>
```

```
            <core:Ident>XYZCORP<</core:Ident>
          </core:Identifier>
        </core:ListOfIdentifier>
      </SellerParty>
      <ShipToParty xsi:type="core:PartyType">
        <core:PartyID>
          <core:Ident/>
        </core:PartyID>
        <core:ListOfIdentifier>
          <core:Identifier>
            <core:Ident/>
          </core:Identifier>
        </core:ListOfIdentifier>
        <core:NameAddress>
          <core:Name1>John Doe</core:Name1>
          <core:Street>123 Sesame Street</core:Street>
          <core:PostalCode>02101</core:PostalCode>
          <core:City>New York</core:City>
          <core:Region>
            <core:RegionCoded>Other</core:RegionCoded>
            <core:RegionCodedOther>NY</core:RegionCodedOther>
          </core:Region>
```

```
        <core:Country>
          <core:CountryCoded>US</core:CountryCoded>
        </core:Country>
      </core:NameAddress>
    </ShipToParty>
    <BillToParty xsi:type="core:PartyType">
      <core:PartyID>
        <core:Ident/>
      </core:PartyID>
      <core:ListOfIdentifier>
        <core:Identifier>
          <core:Ident/>
        </core:Identifier>
      </core:ListOfIdentifier>
      <core:NameAddress>
        <core:Name1>John Doe</core:Name1>
        <core:Street>123 Sesame Street</core:Street>
        <core:PostalCode>02101</core:PostalCode>
        <core:City>NEW YORK</core:City>
        <core:Region>
          <core:RegionCoded>Other</core:RegionCoded>
          <core:RegionCodedOther>NY</core:RegionCodedOther>
```

```
          </core:Region>
          <core:Country>
            <core:CountryCoded>US</core:CountryCoded>
          </core:Country>
        </core:NameAddress>
      </BillToParty>
    </OrderParty>
    <OrderPaymentInstructions>
      <core:PaymentTerms>
        <core:PaymentTerm>
          <core:PaymentTermCoded>Other</core:PaymentTermCoded>
          <core:PaymentTermCodedOther>Dummy text since this is required</
core:PaymentTermCodedOther>
        </core:PaymentTerm>
      </core:PaymentTerms>
      <core:PaymentMethod>
        <core:PaymentMeanCoded>Other</core:PaymentMeanCoded>
        <core:PaymentMeanCodedOther>Credit Card</core:PaymentMeanCodedOther>
      </core:PaymentMethod>
    </OrderPaymentInstructions>
  </OrderHeader>
  <OrderDetail>
```

```
<ListOfItemDetail>
  <ItemDetail>
    <BaseItemDetail>
      <LineItemNum>
        <core:BuyerLineItemNum>1</core:BuyerLineItemNum>
      </LineItemNum>
      <ItemIdentifiers>
        <core:PartNumbers>
          <core:SellerPartNumber>
            <core:PartID>SKU123</core:PartID>
          </core:SellerPartNumber>
        </core:PartNumbers>
      </ItemIdentifiers>
      <TotalQuantity xsi:type="core:QuantityType">
        <core:QuantityValue>1</core:QuantityValue>
        <core:UnitOfMeasurement>
          <core:UOMCoded>Other</core:UOMCoded>
          <core:UOMCodedOther>EA</core:UOMCodedOther>
        </core:UnitOfMeasurement>
      </TotalQuantity>
    </BaseItemDetail>
    <PricingDetail>
```

```
        <core:ListOfPrice>
          <core:Price>
            <core:UnitPrice>
              <core:UnitPriceValue>1.99</core:UnitPriceValue>
            </core:UnitPrice>
          </core:Price>
        </core:ListOfPrice>
      </PricingDetail>
    </ItemDetail>
    <ItemDetail>
      <BaseItemDetail>
        <LineItemNum>
          <core:BuyerLineItemNum>2</core:BuyerLineItemNum>
        </LineItemNum>
        <ItemIdentifiers>
          <core:PartNumbers>
            <core:SellerPartNumber>
              <core:PartID>SKU124</core:PartID>
            </core:SellerPartNumber>
          </core:PartNumbers>
        </ItemIdentifiers>
        <TotalQuantity xsi:type="core:QuantityType">
```

```
            <core:QuantityValue>1</core:QuantityValue>
            <core:UnitOfMeasurement>
              <core:UOMCoded>Other</core:UOMCoded>
              <core:UOMCodedOther>EA</core:UOMCodedOther>
            </core:UnitOfMeasurement>
          </TotalQuantity>
        </BaseItemDetail>
        <PricingDetail>
          <core:ListOfPrice>
            <core:Price>
              <core:UnitPrice>
                <core:UnitPriceValue>27.50</core:UnitPriceValue>
              </core:UnitPrice>
            </core:Price>
          </core:ListOfPrice>
        </PricingDetail>
      </ItemDetail>
    </ListOfItemDetail>
  </OrderDetail>
</Order>
```

Product Data

Product Data (also known as `PRODAT`) is another form of data that can be exchanged using `SendFile()`. When sending `PRODAT` to eiCloud, the `content` parameter of `SendFile()` must be an XML document that complies with the `InventoryManagement2` XSD. This schema is an eBridge variation of the xCBL standard.

SenderParty IDs and ReceivingParty IDs

As with orders, the document must contain identifiers for the sender and recipient of the document. Refer to section on Orders for information on where to find these identifiers in eiCloud. In `PRODAT` documents, the buyer and seller identifiers are replaced with `<SenderParty>` and `<ReceivingParty>` respectively.

Sample XML

A sample `PRODAT` document can be seen below:

```
<InventoryManagement2 xmlns="rrn:org.xcbl:schemas/xcbl/v4_0/catalog/v1_0/
catalog.xsd" xmlns:core="rrn:org.xcbl:schemas/xcbl/v4_0/core/core.xsd"
xmlns:dgs="http://www.w3.org/2000/09/xmldsig#"
xmlns:inv="rrn:org.xcbl:schemas/xcbl/v4_0/materialsmanagement/v1_0/
materialsmanagement.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="rrn:org.xcbl:schemas/scbl/v4_0/catalog/v1_0/
catalog.xsd C:/Schemas/eBRIDGE/Inventorymanagement2.xsd">

  <CatalogHeader>

    <CatalogID>PROD56345234</CatalogID>

    <CatalogDate xmlns:prod="rrn:org.xcbl:schemas/scbl/v4_0/catalog/v1_0/
catalog.xsd">2013-05-08T10:30:00</CatalogDate>

    <CatalogProvider xmlns:prod="rrn:org.xcbl:schemas/scbl/v4_0/catalog/v1_0/
catalog.xsd"/>

    <CatalogVersion xmlns:prod="rrn:org.xcbl:schemas/scbl/v4_0/catalog/v1_0/
catalog.xsd">1</CatalogVersion>

  </CatalogHeader>

  <ListOfNameValueSet>

    <core:NameValueSet>

      <core:SetName>HeaderReferences</core:SetName>
```

```
    <core:ListOfNameValuePair>
      <core:NameValuePair>
        <core:Name>DocType</core:Name>
        <core:Value>PRODAT</core:Value>
      </core:NameValuePair>
    </core:ListOfNameValuePair>
  </core:NameValueSet>
</ListOfNameValueSet>
<SenderParty>
  <core:PartyID>
    <core:Ident>XYZCORP</core:Ident>
  </core:PartyID>
  <core:ListOfIdentifier>
    <core:Identifier>
      <core:Ident>XYZCORP</core:Ident>
    </core:Identifier>
  </core:ListOfIdentifier>
  <core:MDFBusiness>true</core:MDFBusiness>
</SenderParty>
<ReceivingParty>
  <core:PartyID>
    <core:Ident>ACME123</core:Ident>
```

```
  </core:PartyID>
  <core:ListOfIdentifier>
    <core:Identifier>
      <core:Ident>ACME123</core:Ident>
    </core:Identifier>
  </core:ListOfIdentifier>
  <core:MDFBusiness>true</core:MDFBusiness>
</ReceivingParty>
<CatalogData>
  <Product>
    <ProductID>SKU123</ProductID>
    <ProductName>Case of 24 - Maple Syrup</ProductName>
    <UOM>
      <core:UOMCoded>Other</core:UOMCoded>
      <core:UOMCodedOther>Case</core:UOMCodedOther>
    </UOM>
    <ShortDescription>physical</ShortDescription>
    <LongDescription>24 bottles of maple syrup 9</LongDescription>
  </Product>
  <Pricing>
    <ProductPrice>
      <Amount>150.000000</Amount>
```

```
        <PriceType>price</PriceType>
      </ProductPrice>
      <ProductPrice>
        <Amount>60.000000</Amount>
        <PriceType>cost_price</PriceType>
      </ProductPrice>
      <ProductPrice>
        <Amount>150.000000</Amount>
        <PriceType>retail_price</PriceType>
      </ProductPrice>
    </Pricing>
    <InventoryReportDetail>
      <inv:LineItemNumber xmlns:inv="rrn:org.xcbl:schemas/xcbl/v4_0/
materialsmanagement/v1_0/materialsmanagement.xsd"/>
      <inv:InventoryItemIdentifiers>
        <core:PartNumbers>
          <core:SellerPartNumber>
            <core:PartID>SKU123</core:PartID>
          </core:SellerPartNumber>
          <core:OtherItemIdentifiers>
            <core:ProductIdentifierCoded>
              <core:ProductIdentifierQualifierCoded>SKU</
core:ProductIdentifierQualifierCoded>
```

```
            <core:ProductIdentifier>SKU123</core:ProductIdentifier>
          </core:ProductIdentifierCoded>
        </core:OtherItemIdentifiers>
      </core:PartNumbers>
    </inv:InventoryItemIdentifiers>
    <inv:TotalInventoryQuantity>
      <core:QuantityValue>0</core:QuantityValue>
      <core:UnitOfMeasurement>
        <core:UOMCoded>Other</core:UOMCoded>
        <core:UOMCodedOther>Case</core:UOMCodedOther>
      </core:UnitOfMeasurement>
    </inv:TotalInventoryQuantity>
    <inv:ListOfNameValueSet>
      <core:NameValueSet>
        <core:SetName>Warehouse</core:SetName>
        <core:ListOfNameValuePair>
          <core:NameValuePair>
            <core:Name>Is In Stock</core:Name>
            <core:Value>False</core:Value>
          </core:NameValuePair>
        </core:ListOfNameValuePair>
      </core:NameValueSet>
```

```
      </inv:ListOfNameValueSet>

    </InventoryReportDetail>

  </CatalogData>

</InventoryManagement2>
```

Inventory Data

Inventory data, also known as `INVRPT`, uses the same schema definition as product data: the `InventoryManagement2` XSD. Inventory documents also require `<SenderParty>` and `<ReceivingParty>` elements as described in the product data section.

Sample XML

```
<?xml version="1.0" encoding="UTF-16"?>

<InventoryManagement2 xmlns="rrn:org.xcbl:schemas/xcbl/v4_0/catalog/v1_0/
catalog.xsd" xmlns:core="rrn:org.xcbl:schemas/xcbl/v4_0/core/core.xsd"
xmlns:dgs="http://www.w3.org/2000/09/xmldsig#"
xmlns:inv="rrn:org.xcbl:schemas/xcbl/v4_0/materialsmanagement/v1_0/
materialsmanagement.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="rrn:org.xcbl:schemas/scbl/v4_0/catalog/v1_0/
catalog.xsd C:/Schemas/eBRIDGE/Inventorymanagement2.xsd">

  <CatalogHeader>

    <CatalogID>INVT3413</CatalogID>

    <CatalogDate xmlns:prod="rrn:org.xcbl:schemas/scbl/v4_0/catalog/v1_0/
catalog.xsd">2013-04-05T10:39:00</CatalogDate>

    <CatalogProvider xmlns:prod="rrn:org.xcbl:schemas/scbl/v4_0/catalog/v1_0/
catalog.xsd" />

    <CatalogVersion xmlns:prod="rrn:org.xcbl:schemas/scbl/v4_0/catalog/v1_0/
catalog.xsd">1</CatalogVersion>

  </CatalogHeader>

  <SenderParty>
```

```
  <core:PartyID>
    <core:Ident>XYZCORP</core:Ident>
  </core:PartyID>
  <core:ListOfIdentifier>
    <core:Identifier>
      <core:Ident>XYZCORP</core:Ident>
    </core:Identifier>
  </core:ListOfIdentifier>
  <core:MDFBusiness>true</core:MDFBusiness>
</SenderParty>
<ReceivingParty>
  <core:PartyID>
    <core:Ident>ACME123</core:Ident>
  </core:PartyID>
  <core:ListOfIdentifier>
    <core:Identifier>
      <core:Ident>ACME123</core:Ident>
    </core:Identifier>
  </core:ListOfIdentifier>
  <core:MDFBusiness>true</core:MDFBusiness>
</ReceivingParty>
<CatalogData>
```

```
    <Product>
      <ProductID>SKU123</ProductID>
      <ProductName>Case of 24 - Maple Syrup</ProductName>
      <UOM>
        <core:UOMCoded>Other</core:UOMCoded>
        <core:UOMCodedOther>PAIR</core:UOMCodedOther>
      </UOM>
      <ShortDescription />
      <LongDescription />
    </Product>
    <InventoryReportDetail>
      <inv:LineItemNumber xmlns:inv="rrn:org.xcbl:schemas/xcbl/v4_0/
materialsmanagement/v1_0/materialsmanagement.xsd">1</inv:LineItemNumber>
      <inv:InventoryItemIdentifiers>
        <core:PartNumbers>
          <core:SellerPartNumber>
            <core:PartID>SKU123</core:PartID>
          </core:SellerPartNumber>
          <core:OtherItemIdentifiers>
            <core:ProductIdentifierCoded>
              <core:ProductIdentifierQualifierCoded>SKU</
core:ProductIdentifierQualifierCoded>
              <core:ProductIdentifier>SKU123</core:ProductIdentifier>
```

```
          </core:ProductIdentifierCoded>
        </core:OtherItemIdentifiers>
      </core:PartNumbers>
    </inv:InventoryItemIdentifiers>
    <inv:TotalInventoryQuantity>
      <core:QuantityValue>0</core:QuantityValue>
      <core:UnitOfMeasurement>
        <core:UOMCoded>Other</core:UOMCoded>
        <core:UOMCodedOther>PAIR</core:UOMCodedOther>
      </core:UnitOfMeasurement>
    </inv:TotalInventoryQuantity>
    <inv:ListOfNameValueSet>
      <core:NameValueSet>
        <core:SetName>Warehouse</core:SetName>
        <core:ListOfNameValuePair>
          <core:NameValuePair>
            <core:Name>Is In Stock</core:Name>
            <core:Value>False</core:Value>
          </core:NameValuePair>
        </core:ListOfNameValuePair>
      </core:NameValueSet>
    </inv:ListOfNameValueSet>
```

```
    </InventoryReportDetail>

  </CatalogData>

</InventoryManagement2>
```

Receiving Documents from eiCloud

Data can be retrieved from eiCloud using GetDocumentList() or GetDocumentList2() in combination with GetDocument().

GetDocumentList

`GetDocumentList()` returns an array of document IDs based on filter criteria in its parameters:

Parameter	Type	Description
`login`	string	eiCloud username of sender
`password`	string	eiCloud password of sender
`status`	string	Either `All` or `New`
`docType`	string	A document type such as `ORDERS`, `INVRPT` or `PRODAT` (optional)
`partner`	string	The name of the partner or endpoint (optional)
`fromDate`	datetime	Start of date range
`toDate`	datetime	End of date range

`GetDocumentList2()` works the same way as `GetDocumentList()` however the `fromDate` and `toDate` fields must be sent as a string type instead of datetime. When using `fromDate` and `toDate` parameters with `GetDocumentList2`, the dates should be formatted `YYYY-MM-DD`.

Documents have the status `New` until they are retrieved using `GetDocument()` or an eBridge application adapter.

GetDocument

Once you have the list of document IDs that you wish to retrieve, you can use the `GetDocument()` method to fetch them. The parameters of `GetDocument()` are listed below:

Parameter	Type	Description
`login`	string	eiCloud username

`password`	string	eiCloud password
`sys_no`	integer	Document ID as returned from `GetDocumentList()` or `GetDocumentList2()`

Sample code

Here is an example script that uses `GetDocumentList()` and `GetDocument()` to retrieve orders from an eiCloud account:

- PHP
- C#

```
<?php

$login = 'JaneDoe';

$password = 'abc123';

$client = new SoapClient("https://eicloudservice.com/eportalservice.asmx?
wsdl");

$params = array(

        'login' => $login,

        'password'=> $password,

        'status' => 'All',

        'docType' => '850',

        'fromDate' => '2011-03-28',

        'toDate' => '2013-07-28'

        );

try {

        $response = $client->GetDocumentList2($params);

        }
```

```
catch (Exception $e) {

        echo 'Error: '.$e->getMessage();

        }

$xml = new SimpleXMLElement($response->GetDocumentList2Result);

$docs = $xml->xpath("//@doc_sys_no");

$params = array(

        'login' => $login,

        'password'=> $password

        );

foreach ($docs as $doc) {

        $params['sys_no'] = (string) $doc;

        try {

                $response = $client->GetDocument($params);

                }

        catch (Exception $e) {

                echo 'Error: '.$e->getMessage();

                }

        var_dump($response->GetDocumentResult);

        }
```

Other Methods

Get Document Type List

The GetDocTypeList() method allows you to retrieve a list of all document types used on eiCloud. Below you can find the required parameters and sample scripts.

Parameter	Type	Description
username	string	eiCloud username
password	string	eiCloud password

- PHP

```
<?php

$client = new SoapClient("https://eicloudservice.com/eportalservice.asmx?
wsdl");

$params = array(

        'username'=>'JaneDoe',

        'password'=>'abc123'

        );

try {

        $response = $client->GetDocTypeList($params);

        }

catch (Exception $e) {

        $response = 'Error: '.$e->getMessage();

        }

var_dump($response);
```

More Resources

- WSDL
- ordermanagement.xsd (for orders)
- inventorymanagement2.xsd (for product and inventory data)

- Contact eBridge Connections Support
- eBridge Connections Blueprint Builder