



Leading the **Integration** Revolution

Your business problems have changed.
Why hasn't your integration solution?



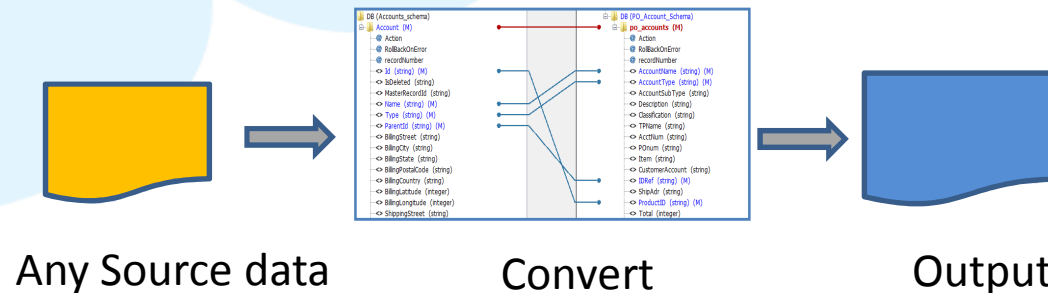
ADEPTIA



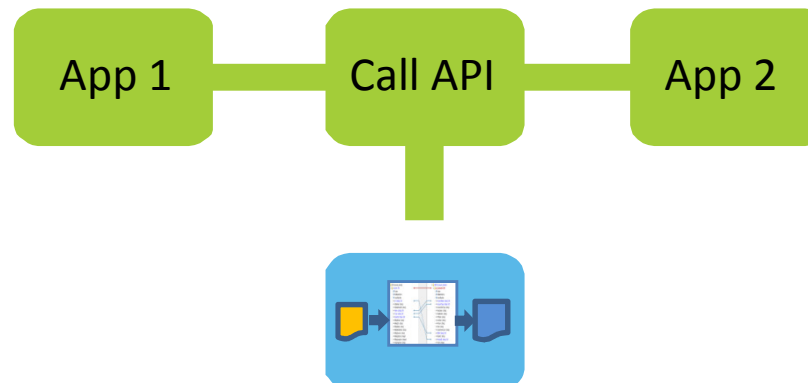
Dynamic Mapping Service API

Use Case: Dynamic Mapping Service

Step 1: Create a dynamic process that can take any source data, executes the appropriate Mapping to convert or transform the source into the desired output.



Step 2: Wrap this service into a REST API that can be called by applications automatically.

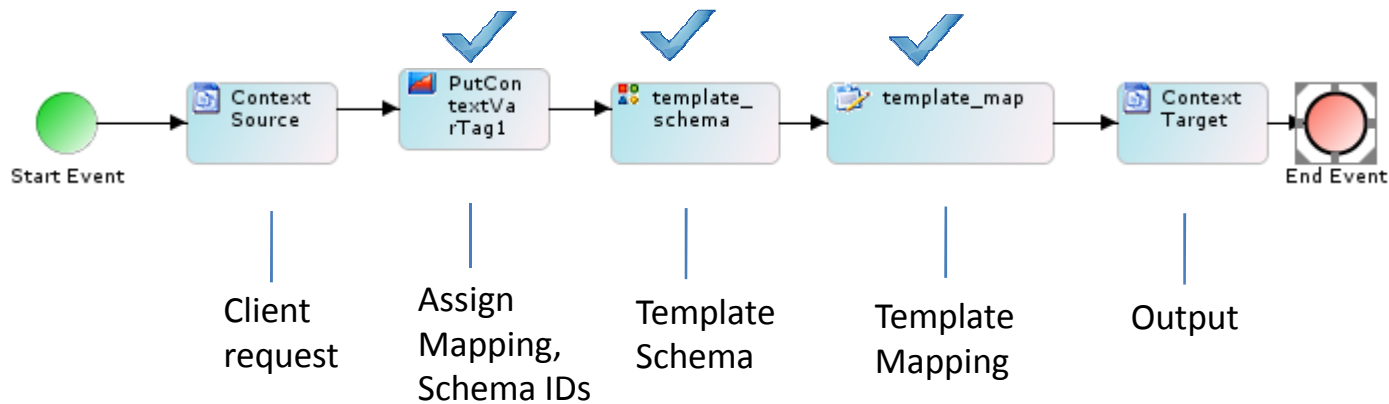


Step 1: Create a dynamic process (example 1)

Refer to the Use Case video on how to build the process flow.

Here's an example of a dynamic process template.

Base process

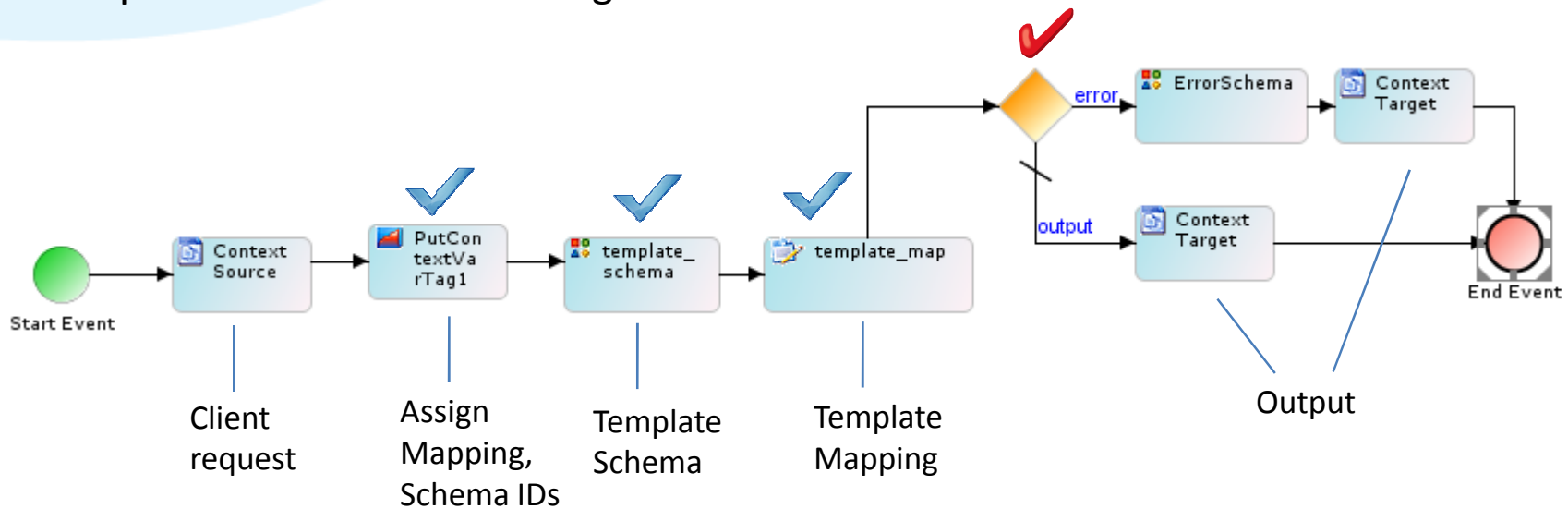


Step 1: Create a dynamic process (example 2)

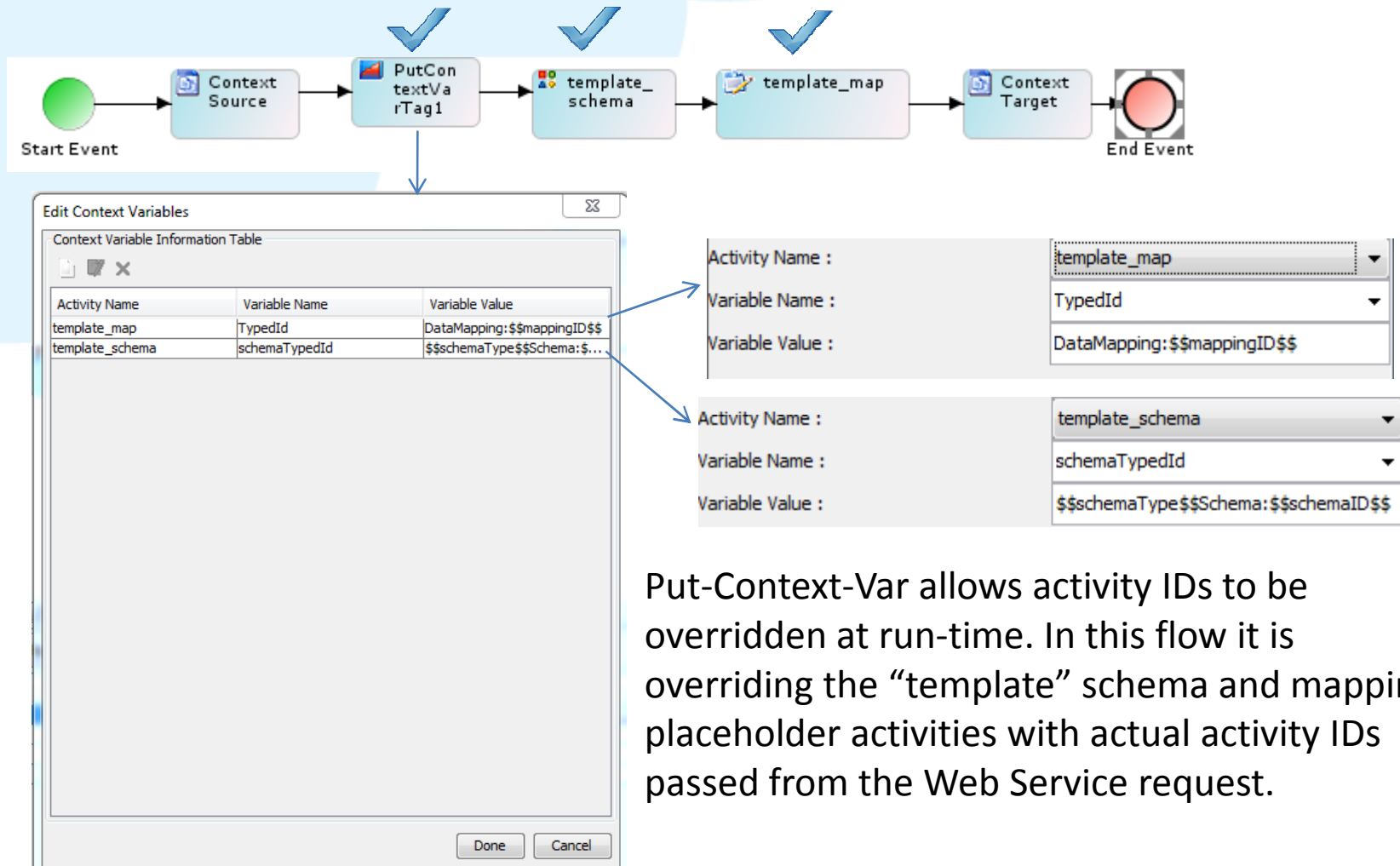
Refer to the Use Case video on how to build the process flow.

We can add Error Handling steps after the gateway and make the process extensible as per your business needs.

Base process with error handling



Overriding the Activity IDs in process flow



Put-Context-Var allows activity IDs to be overridden at run-time. In this flow it is overriding the “template” schema and mapping placeholder activities with actual activity IDs passed from the Web Service request.

Gateway condition to check errors

The image illustrates the configuration of a gateway condition in a BPMN diagram. The diagram shows a gateway (diamond) with an 'error' event (red circle) leading to 'ErrorSchema' and 'Context Target' tasks. A red arrow points from the gateway to the 'Condition Wizard' dialog box.

Condition Wizard

Other Condition
Please enter arbitrary name-value:

Name :

Operands Type :

Operators :

Value :

Decimal Precision :

Check if varError is true.

Buttons: Back, Next, Done, Cancel

Context Source properties and multiple stream

The screenshot displays a software interface with a flowchart on the left and a 'Multiple Stream Dialog' window in the foreground. The flowchart shows a 'Start Event' connected to a 'Context Source' activity, which is marked with a red checkmark. The 'Multiple Stream Dialog' window contains the following sections:

- Stream Name Entry:** Stream name: [] Add Stream
- Stream And Activity Mapping:** Streams: ContextSource, Activities: template_schema, Map [✓]
- Stream Information Table:**

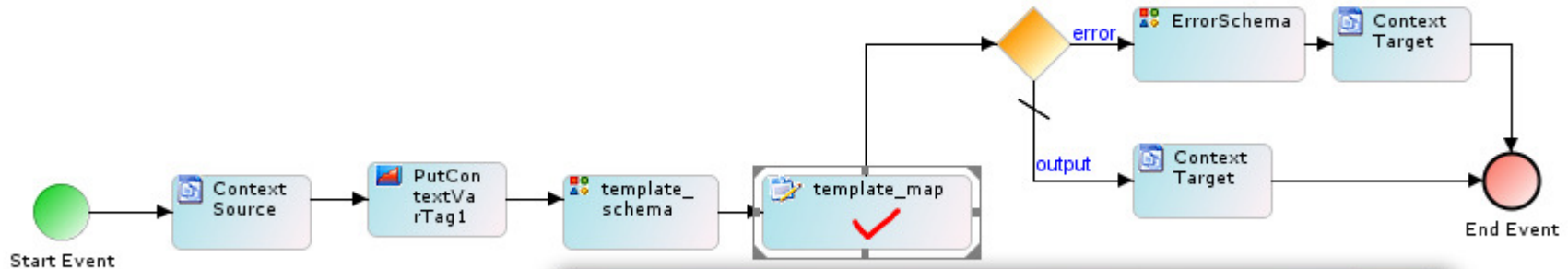
Stream Name	Error Stream	Stream Consumer ...	Explicit Stream
ContextSource	<input type="checkbox"/>	template_schema	<input checked="" type="checkbox"/>

At the bottom of the dialog, the 'Done' button is highlighted with a red checkmark. Below the dialog, a 'Properties' window is partially visible, showing a table of properties for the 'ContextSource' activity:

Name	ContextSource
parameterName	restRequest
source	<u>ContextSource</u>
streamNames	ContextSource
Synch	true
Type	ContextSource

Template Map multiple stream option

25 50 75 100 125 150 175 200 225 250 275 300 325 350 375 400 425 450 475 500 525 550 575 600 625 650 675 700 725 750 775 800 825 850 875



Multiple Stream Dialog

Stream Name Entry
Stream name : Add Stream

Stream And Activity Mapping
Streams: error
Activities: ErrorSchema
Map

Stream Information Table

Stream Name	Error Stream	Stream Consumer	Explicit Stream
output	<input type="checkbox"/>	ContextTarget ✓	<input type="checkbox"/>
error	<input type="checkbox"/>	ErrorSchema ✓	<input type="checkbox"/>

Properties

OnException Scripts Alerts Process

Maximum Retries on Failure
Wait Time(in seconds) between Retries
Character Set Encoding
Name Stream
Records
Site Stream

Two output streams in the Mapping

In the template and the actual map name the two output streams as 'output' and 'error'

Structure_of_source_XSD

- Root (UseCase2_Text_schema)
 - Record (F, CM -)
 - @ recordNumber
 - <> _OrderNumber (string) (M)
 - <> Document_Date (date) (M)
 - <> CustomerID (string) (M)
 - <> CustomerDUNS (string) (M)
 - <> SupplierID (string) (M)
 - <> SupplierDUNS (string) (M)
 - <> TotalAmount (integer) (M)
 - <> PaymentDetails (string) (M)
 - <> LineNumber (integer)
 - <> Action_Status (string)
 - <> Material (string)
 - <> TotalLineQTY (integer)
 - <> UOM (string)
 - <> ScheduleID (string)
 - <> OpenQTY (integer)
 - <> ConfirmedQTY (integer)
 - <> PromisedShipDate (date)

Structure_of_destination_XSD

- Exercise2 (UseCase2_Order_XML) ✓
 - Transaction (F, LV)
 - HeaderLevel
 - <> OrderNumber (M)
 - <> DocumentDate (M)
 - <> CustomerID (M)
 - <> CustomerDUNS (M)

Assign Streams

Order	Schema Name	Root Element	Stream Name
1	UseCase2_Order_XML	Exercise2	output ✓
2	ErrorSchema	Root	error ✓

Parameters

- Global Variables
 - \$varOrdNum (\$Input_UseCase2_Text_schema/Ro
- Local Variables
- Custom Methods

Context variable in actual Mapping

Set context variable with value as true. Use for each and apply any error rule on this schema's record node that would result in this error schema to be populated by error records at run-time.

The screenshot displays the Data Mapper interface with two XSD schemas side-by-side. The left schema, 'Structure_of_source_XSD', shows a 'Record' node with attributes like '@ recordNumber', '_OrderNumber', 'Document_Date', 'CustomerID', 'CustomerDUNS', 'SupplierID', 'SupplierDUNS', 'TotalAmount', 'PaymentDetails', 'LineNumber', 'Action_Status', 'Material', 'TotalLineQTY', 'UOM', and 'ScheduleID'. The right schema, 'Structure_of_destination_XSD', shows a 'Transaction' node with similar attributes and an 'ErrorSchema' node containing a 'Record' node with '@ recordNumber' and 'Error (string) (M, LV)'. A red line connects the 'recordNumber' attribute of the source to the 'recordNumber' attribute of the destination. Below the schemas, a table shows a local variable 'varError' with the value 'set-context('varError','true')'.

Local Variable Name	Local Variable Value	Add Comment
varError	set-context('varError','true')	

Step 2: Publish the dynamic process as a REST API

Web Service Provider: mappingservice

Standard Properties

Name* mappingservice

Description* convert any data to standard OAGIS format

Character Set Encoding* ISO-8859-1

Refresh

Publish Type SOAP REST

Resource End Path* /convert

Process Flow Name* Generic_Mapping_Service

Select the Dynamic Process flow

#	Name	Value	Style
1			Query
2			Query
3			Query
4			Query

Resource Parameter

No. of Rows 1 at Position 5

Add Row

Remove Row

Add Method

Method Type* Post

#	Name	Default Value	Style
1			Query
2			Query
3			Query
4			Query

Method Parameter

No. of Rows 1 at Position 5

Add Row Remove Row

Type	Media Type	Status Code	Variable Name
REQUEST	application/text		restRequest
RESPONSE	application/text		restResponse
FAULT	application/text		restFault

Step 2: Publish the dynamic process as a REST API

After saving the Provider go to Manage page and click on the View to see the REST endpoint. Now use this endpoint in your client application to pass the payload and the Activity IDs and get the mapping results.

Services > Web Services > Provider

Synchronize Delete Create New

<input type="checkbox"/>	Name	Description	Owner	Style	WSDL	Project Name	Modified	Action
<input type="checkbox"/>	mappingservice ✓	convert any data to standard OAGIS for...	admin	REST	View ✓	Default	10/28/14 ...	☰

Properties	Value	
RESOURCE PARAMETER		
POST METHOD	URL	http://23.21.250.133:8080/adeptia/publishProviderByRest/mappingservice/convert?
	XSD (REQUEST,RESPONSE,FAULT)	

Close Window

Running the Mapping Service API: Request

chrome-extension://hgml0ofddfdnphfgcellkdfbfjeloo/RestClient.html#RequestPlace:saved/1

Advanced Rest Client

dynamic_api_call

Save Open

http://23.21.250.133:8080/adeptia/publishProviderByRest/mappingservice/convert? ✓

GET POST ✓ PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

Add new header

mappingID	010070010106140806706463300065 ✓
schemaID	010070010106140806224172300038 ✓
schemaType	EDI ✓

Raw Form Files (0) Payload ✓

Encode payload Decode payload

```
ISA*00* *00* *ZZ*TEST22222222 *01*TEST99999 *140730*1700*U*00401*000000286*0*P*>~
GS*SH*TEST22222222*TEST99999*20140730*1700*216*X*004010~
ST*856*301700225~
BSN*00*800999225*20140730*1700*0004~
DTM*011*20140730~
DTM*017*20140804~
HL*1**S~
TD1*CTN*60***G*21.772*LB~
ID3*TL~
REF*BM*8005265837~
FOB*PP~
HL*2*1*O~
PRF*6000243259~
HL*3*2*I~
LIN*000001*BP*DCF-00203~
SN1**30*EA~
REF*BV*000001~
HL*4*2*I~
LIN*000002*BP*DCF-00205~
SN1**30*EA~
```

application/x-www-form-urlencoded ✓ Set "Content-Type" header to overwrite this value.

Clear Send ✓

Schema type, Mapping and Schema IDs in the Headers

Sending Request data in Text

Running the Mapping Service API: Response

Scroll to top

Status **202 Accepted** Loading time: 552 ms

Request headers
schemaType: EDI
Origin: chrome-extension://hgml0ofddfdnphfgcellkdfbfjeloo
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
schemalD: 010070010106140806224172300038
mappingID: 010070010106140806706463300065
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8
Cookie: m=open; m6=open; m7=open; m2=open; m3=open; yPos=!~0-!; JSESSIONID=ut74pbexk6fa1tgdwrtpjiwh; JSESSIONID=g22jkqm8s5px1r7bc7iuhfkiw; lastService=WsProvider

Response headers
Date: Tue, 28 Oct 2014 19:56:23 GMT
Content-Type: application/text; charset=ISO-8859-1
Transfer-Encoding: chunked

Request triggers the Mapping Service and sends the mapping result back as a response.

Raw Parsed **Response**

[Open output in new window](#) [Copy to clipboard](#) [Save as file](#) [Open in JSON tab](#)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ns0:NotifyShipment xmlns:ns0="http://www.openapplications.org/oagis/9_4" xmlns:e2o="http://www.e2open.com/ns">
  <ns0:ApplicationArea>
    <ns0:Sender>
      <ns0:LogicalID>TEST22222222 </ns0:LogicalID>
      <ns0:TaskID>ASN</ns0:TaskID>
      <ns0:ReferenceID>na</ns0:ReferenceID>
      <ns0:ConfirmationCode>na</ns0:ConfirmationCode>
    </ns0:Sender>
    <ns0:Receiver>
      <ns0:LogicalID>TEST99999 </ns0:LogicalID>
    </ns0:Receiver>
  </ns0:ApplicationArea>
  <ns0:DataArea>
    <ns0:Shipment>
      <ns0:ShipmentHeader>
        <ns0:DocumentID>
          <ns0:ID>800999225</ns0:ID>
        </ns0:DocumentID>
        <ns0:DocumentDateTime>20140730</ns0:DocumentDateTime>
      </ns0:ShipmentHeader>
      <ns0:Status>
        <ns0:Code name="Payment">PP</ns0:Code>
      </ns0:Status>
      <ns0:BillOfLadingReference>
        <ns0:DocumentID>
          <ns0:ID>800999225</ns0:ID>
        </ns0:DocumentID>
        <ns0:Note type="PackCode">CTN</ns0:Note>
        <ns0:Quantity>60</ns0:Quantity>
      </ns0:BillOfLadingReference>
      <ns0:ScheduledDeliveryDateTime>2014-08-04T00:00:00.000+0100</ns0:ScheduledDeliveryDateTime>
      <ns0:ActualDeliveryDateTime>2014-07-30T14:00:27+00:00</ns0:ActualDeliveryDateTime>
    </ns0:Shipment>
  </ns0:DataArea>
</ns0:NotifyShipment>
```

Receiving Response data in XML



Thank you!

sales@adeptia.com