

SMS Broker System
Software Integration

Annex. AMS Query. SOAP messaging.

(ver. from 8/3/2026)

Table of Contents

1. Sample of AMS Query implementation.	1
2. Samples of xml SOAP response message.	3
2.1 PutToQueue() method.	3
2.2 GetAnswers() method.	4
2.3 GetABIEvents() method.	4
2.4 PutToQueue_CargoManifestEntryReleaseQuery_InBond() method.	5
2.5 Exceptions.	6
3. How to prepare HTTP request using wsdl-file.	7
3.1 PutToQueue_CargoManifestEntryReleaseQuery_InBond() method.	7

1. Sample of AMS Query implementation.

Here you can see a simplified method that realizes an AMS Query by means of HTTP request process. I was written on C# but can be repeated on any programming language and environment that allows to call Web HTTP request. The sample was created only in demonstration purposes.

```
// This string is for PutToQueue() method xml-SOAP template.
// RequestRelatedBOL, RequestBOLEntryData already are added as false default values:
private string xmlPutToQueueTemplate = "<?xml version='1.0' encoding='utf-8'><s:Envelope xmlns:s='http://schemas.xmlsoap.org/soap/envelope/'><s:Header><o:Security s:mustUnderstand='1'><o:UsernameToken><o:Username /><o:PasswordType='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText' /></o:UsernameToken></o:Security></s:Header><s:Body><PutToQueue xmlns='http://tempuri.org/'><ApplId>CQ</ApplId><parameters xmlns:i='http://www.w3.org/2001/XMLSchema-instance'><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>EntryFilerCode</Name><Value i:type='a:string'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>EntryNumber</Name><Value i:type='a:string'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>InBondNumber</Name><Value i:type='a:string'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>SCAC</Name><Value i:type='a:string'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>BillNumber</Name><Value i:type='a:string'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>AirMasterBOL</Name><Value i:type='a:string'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>AirHouseBOL</Name><Value i:type='a:string'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>RequestRelatedBOL</Name><Value i:type='a:boolean'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>RequestBOLEntryData</Name><Value i:type='a:boolean'></Value></Parameter><Parameter xmlns='http://www.w3.org/2001/XMLSchema-instance'><Name>ClientRef</Name><Value i:type='a:string'></Value></Parameter></parameters></PutToQueue></s:Body></s:Envelope></pre></div>
```

```

// string requestType: define a type of request to be sent to Customs.
// "BOL"      - Bill of Lading Query (Sea)
// "Air"      - Airway Bill Query
// "In-Bond"  - In-Bond Query
// "Entry"    - Cargo Release Query
//
// object[] values - query parameter values

public string ProcessAMSQuery(string queryType, object[] values, string username, string password)
{
    string Result = ""; // Customs response to AMS Query in conventional text format.

    XmlDocument xmlDoc = new XmlDocument();
    xmlDoc.LoadXml(xmlPutToQueueTemplate); // Load xml SOAP template for PutToQueue() method of SmsABIManager service to xml document.
                                           // Template already contains hardcoded ApplId = "CQ" Customs application type that means "ACE Cargo
                                           // Manifest / In-Bond / Entry Status Query"

    // Prepare HTTP request to put the AMS query to transmission queue on SMS server.
    HttpRequest SmsABIManagerRequest = (HttpRequest)WebRequest.Create(@"https://smstest.logisticaldatasolutions.com:8130/brokerservice/
SmsABIManager/");
    AddHttpHeaders(SmsABIManagerRequest, @"http://tempuri.org/ISmsABIManager/PutToQueue");

    SetTagParameter(xmlDoc, "o:Username", username);
    SetTagParameter(xmlDoc, "o:Password", password);

    // PutToQueue() method with ApplId = "CQ" sends different types of AMS Query depending on a set of input parameters.
    // Input query parameter values to PutToQueue() method xml SOAP template:
    switch (queryType)
    {
        case "BOL": // Bill of Lading Query (Sea)
            SetParameter(xmlDoc, "SCAC", values[0]);
            SetParameter(xmlDoc, "BillNumber", values[1]);
            SetParameter(xmlDoc, "RequestRelatedBOL", values[2]);
            SetParameter(xmlDoc, "ClientRef", values[3]);
            break;
        case "Air": // Airway Bill Query
            SetParameter(xmlDoc, "AirMasterBOL", values[0]);
            SetParameter(xmlDoc, "AirHouseBOL", values[1]);
            SetParameter(xmlDoc, "ClientRef", values[2]);
            break;
        case "In-Bond": // In-Bond Query
            SetParameter(xmlDoc, "InBondNumber", values[0]);
            SetParameter(xmlDoc, "ClientRef", values[1]);
            break;
        case "Entry": // Cargo Release Query
            SetParameter(xmlDoc, "EntryFilerCode", values[0]);
            SetParameter(xmlDoc, "EntryNumber", values[1]);
            SetParameter(xmlDoc, "RequestBOLEntryData", values[2]);
            SetParameter(xmlDoc, "ClientRef", values[3]);
            break;
        default:
            return "";
    }
}

FillHttpBody(xmlDoc, SmsABIManagerRequest); // Add xml SOAP to HTTP request.

try
{
    // Get response from SMS-server:
    HttpResponse response = (HttpResponse)SmsABIManagerRequest.GetResponse();
    xmlDoc = GetXML(response);
    string transmissionId = GetTagValue(xmlDoc, "PutToQueueResult"); // We need this Id to get the response transmission object from SMS-server on
                                                                    // the next step.

    // Prepare HTTP request to get transmission object with Customs response from SMS server.
    HttpRequest ABITransmissionManagerRequest = (HttpRequest)WebRequest.Create(@"https://smstest.logisticaldatasolutions.com:8130/
brokerservice/ABITransmissionManager/");
    AddHttpHeaders(ABITransmissionManagerRequest, @"http://tempuri.org/IABITransmissionManager/GetAnswers");

    xmlDoc = new XmlDocument();
    xmlDoc.LoadXml(xmlGetAnswersTemplate); // Load xml SOAP template for GetAnswers() method of ABITransmissionManager service to xml
document.
    SetTagParameter(xmlDoc, "requestId", transmissionId); // Input Id that we just got to xml SOAP template

    SetTagParameter(xmlDoc, "o:Username", username);
    SetTagParameter(xmlDoc, "o:Password", password);

    FillHttpBody(xmlDoc, ABITransmissionManagerRequest); // Add xml SOAP to HTTP request.

    try
    {

```

```

// PutToQue() is synchronous method and if it returned transmission Id outgoing transmission object is already added on SMS-server
// but it can be a time lag between response to PutToQue() method (when we already added an outgoing transmission object on SMS
// server and already sent a query to Customs side) and Customs response and hence the appearance of incoming transmission object
// on SMS-server. To wait a Customs response there is a loop here 5 times 5 seconds.
// The better approach is to use asynchronous PutToQueueOneWay() method but it would be more complex sample.
string Link = null;
for (int i = 0; i < 5; i++)
{
    System.Threading.Thread.Sleep(5000);
    response = (HttpWebResponse)ABITransmissionManagerRequest.GetResponse();
    xmlDoc = GetXML(response);
    Link = GetTagValue(xmlDoc, "SourceLink", "http://schemas.datacontract.org/2004/07/SMS.Broker.DataContracts.Documents");
    if (Link != null)
        break;
}

if (Link == null)
    return "Customs didn't responded yet.";

HttpWebRequest EntityEventManagerRequest = (HttpWebRequest)WebRequest.Create(@"https://smstest.logisticaldatasolutions.com:8130/
brokerservice/EntityEventManager/");
AddHTTPHeaders(EntityEventManagerRequest, "\"http://tempuri.org/IEntityEventManager/GetABIEvents\"");

xmlDoc = new XmlDocument();
xmlDoc.LoadXml(xmlGetEventTemplate); // Load xml SOAP template for GetEvent() method of EntityEventManager service to xml document.
// Incoming transaction object we got on previous step contains Customs response in special Customs
// format. Applying to EntityEventManager we get more readable conventional text format.
SetTagParameter(xmlDoc, "link", Link); // Input known link to xml SOAP template (link looks something like "ABISearch=628")
SetTagParameter(xmlDoc, "includeText", true); // Input "true" to xml SOAP template to return Text (it is not transmitted by default)

SetTagParameter(xmlDoc, "o:Username", username);
SetTagParameter(xmlDoc, "o:Password", password);

FillHTTPBody(xmlDoc, EntityEventManagerRequest); // Add xml SOAP to HTTP request.

try
{
    response = (HttpWebResponse)EntityEventManagerRequest.GetResponse();
    xmlDoc = GetXML(response);
    Result = GetTagValue(xmlDoc, "Text", "http://schemas.datacontract.org/2004/07/SMS.Broker.DataContracts.Common");
}
catch (WebException ex)
{
    Result = FaultProcessing(ex); // Exception processing is simplified.
}
catch (WebException ex)
{
    Result = FaultProcessing(ex); // Exception processing is simplified.
}
catch (WebException ex)
{
    Result = FaultProcessing(ex); // Exception processing is simplified.
}
catch (Exception e)
{
    Result = e.ToString();
}

return Result;
}

```

2. Samples of xml SOAP response message.

2.1 PutToQueue() method.

A sample of response xml SOAP message:

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <SMSMessageHeader xmlns="http://SMS/DataServer/2014/12">
      <SMSMessageHeader xmlns="Key">&lt;?xml version="1.0" encoding="utf-16"?>
&lt;HeaderProperties xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
&lt;ServiceMessages />
&lt;/HeaderProperties></SMSMessageHeader>

```

```

</SMSMessageHeader>
<o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <u:Timestamp u:Id="_0">
    <u:Created>2024-04-27T18:51:26.642Z</u:Created>
    <u:Expires>2024-04-27T18:56:26.642Z</u:Expires>
  </u:Timestamp>
</o:Security>
</s:Header>
<s:Body>
  <PutToQueueResponse xmlns="http://tempuri.org/">
    <PutToQueueResult>7305</PutToQueueResult>
  </PutToQueueResponse>
</s:Body>
</s:Envelope>

```

2.2 GetAnswers() method.

A sample of response xml SOAP message:

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <SMSMessageHeader xmlns="http://SMS/DataServer/2014/12">
      <SMSMessageHeader xmlns="Key">&lt;?xml version="1.0" encoding="utf-16"?>
&lt;HeaderProperties xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
&lt;ServiceMessages />
&lt;/HeaderProperties></SMSMessageHeader>
    </SMSMessageHeader>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <u:Timestamp u:Id="_0">
        <u:Created>2024-04-27T20:33:23.658Z</u:Created>
        <u:Expires>2024-04-27T20:38:23.658Z</u:Expires>
      </u:Timestamp>
    </o:Security>
  </s:Header>
  <s:Body>
    <GetAnswersResponse xmlns="http://tempuri.org/">
      <GetAnswersResult xmlns:a="http://schemas.datacontract.org/2004/07/SMS.Broker.DataContracts.Documents"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <a:ABITransmission z:Id="1" xmlns:z="http://schemas.microsoft.com/2003/10/Serialization/">
          <Id xmlns="">7306</Id>
          <RowVersion xmlns="">AAAAAAAXI+Q=</RowVersion>
          <a:ABIMessage>A4701SY1 042724 C1 01 7305 ABETATEST B 4701SY1C1
01 7305 WR1 XXXCTEST230411074707CARR0001E041123 WS4EX022924022924
62 WR4800000843 CJ230411R002 00000005CTN XXXC 0N1 WN1800000843 46014601 46012704
Y 4701SY1C100004 01 Z4701SY1 042724 01</a:ABIMessage>
          <a:AppliactionId>C1</a:AppliactionId>
          <a:Content>800000843XXXCCJ230411R002</a:Content>
          <a:ProcessingTime>2024-04-27T18:51:28.597</a:ProcessingTime>
          <a:Request_Id>7305</a:Request_Id>
          <a:SourceLink>ABIQuery=709</a:SourceLink>
          <a:LastStatusSubject>CQ</a:LastStatusSubject>
          <a:Text/>
          <a:Time>2024-04-27T18:51:28.517</a:Time>
          <a:User_Id>3</a:User_Id>
        </a:ABITransmission>
      </GetAnswersResult>
    </GetAnswersResponse>
  </s:Body>
</s:Envelope>

```

2.3 GetABIEvents() method.

A sample of response xml SOAP message:

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <SMSMessageHeader xmlns="http://SMS/DataServer/2014/12">
      <SMSMessageHeader xmlns="Key">&lt;?xml version="1.0" encoding="utf-16"?>
&lt;HeaderProperties xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
&lt;ServiceMessages />
&lt;/HeaderProperties></SMSMessageHeader>
    </SMSMessageHeader>

```

```

<o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <u:Timestamp u:Id="_0">
    <u:Created>2024-04-27T20:57:59.950Z</u:Created>
    <u:Expires>2024-04-27T21:02:59.950Z</u:Expires>
  </u:Timestamp>
</o:Security>
</s:Header>
<s:Body>
  <GetABIEventsResponse xmlns="http://tempuri.org/">
    <GetABIEventsResult xmlns:a="http://schemas.datacontract.org/2004/07/SMS.Broker.DataContracts.Common"
      xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
      <a:EntityEvent z:Id="i1" xmlns:z="http://schemas.microsoft.com/2003/10/Serialization/">
        <Id xmlns="">1104806</Id>
        <a:ABITransmission_Id>7306</a:ABITransmission_Id>
        <a:Message>ABI-> C1</a:Message>
        <a:SourceLink>ABIQuery=709</a:SourceLink>
        <a:LastStatus z:Id="i2">
          <Id xmlns="">143109</Id>
          <RowVersion xmlns="">AAAAAAAXI+M=</RowVersion>
          <a:ABIQuery_Id>709</a:ABIQuery_Id>
          <a:Source>C1</a:Source>
          <a:Status>ABI App QUE status</a:Status>
          <a:Subject>CQ</a:Subject>
          <a:Time>2024-04-27T18:51:28.58</a:Time>
          <a:User z:Id="i3" xmlns:b="http://schemas.datacontract.org/2004/07/SMS.Broker.DataContracts.Directories">
            <Id xmlns="">3</Id>
            <RowVersion xmlns="">AAAAAAAWjqE=</RowVersion>
            <Code xmlns="">ROBOT</Code>
            <Name xmlns="">ROBOT</Name>
          </a:User>
          <a:User_Id>3</a:User_Id>
          <a:Value>Incoming</a:Value>
        </a:LastStatus>
        <a:LastStatus_Id>143109</a:LastStatus_Id>
        <a:Text>Importer #:
Carrier Code: XXXC Vessel: TEST230411074707CARR Voyage/Flight: 0001E
Arrival Date: 4/11/2023
In-bond Status: { }
In-bond Arrival Date: 2/29/2024
In-bond Export Date: 2/29/2024
In-bond Entry Type: {62} Transportation and Exportation (T&E)
In Bond Number: 800000843
Master bill #: XXXC CJ230411R002
House bill #:
Manifest Qty: 5 CTN
Bill of Lading Type: {0} Regular Bill of Lading
Importer Security Filing Indicator: {N} ISF not on file
Mode of Transportation Code: {1} Ocean

In-bond Entry Number: 800000843
Manifested Port of Unlading/Import: 4601
Actual Port of Unlading/Import: 4601
In-bond Originating Port: 4601
Manifested In-bond Destination Port: 2704</a:Text>
      <a:Time>2024-04-27T18:51:28.603</a:Time>
      <a:Type>ABI</a:Type>
      <a:User z:Ref="i3" xmlns:b="http://schemas.datacontract.org/2004/07/SMS.Broker.DataContracts.Directories">
        <a:User_Id>3</a:User_Id>
      </a:EntityEvent>
    </GetABIEventsResult>
  </GetABIEventsResponse>
</s:Body>
</s:Envelope>

```

2.4 PutToQueue_CargoManifestEntryReleaseQuery_InBond() method.

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <SMSMessageHeader xmlns="http://SMS/DataServer/2014/12">
      <SMSMessageHeader xmlns="Key">&lt;?xml version="1.0" encoding="utf-16"?>
&lt;HeaderProperties xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
&lt;ServiceMessages />
&lt;/HeaderProperties></SMSMessageHeader>
    </SMSMessageHeader>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <u:Timestamp u:Id="_0">

```

```

        <u:Created>2024-05-01T01:17:12.281Z</u:Created>
        <u:Expires>2024-05-01T01:22:12.281Z</u:Expires>
    </u:Timestamp>
</o:Security>
</s:Header>
<s:Body>
    <PutToQueue_CargoManifestEntryReleaseQuery_InBondResponse xmlns="http://tempuri.org/">

<PutToQueue_CargoManifestEntryReleaseQuery_InBondResult>7487</PutToQueue_CargoManifestEntryReleaseQuery_InBondResult
>
    </PutToQueue_CargoManifestEntryReleaseQuery_InBondResponse>
</s:Body>
</s:Envelope>

```

2.5 Exceptions.

A sample of fault response message to PutToQueue() method:

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <SMSMessageHeader xmlns="http://SMS/DataServer/2014/12">
      <SMSMessageHeader xmlns="Key">&lt;?xml version="1.0" encoding="utf-16"?>
&lt;HeaderProperties xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
&lt;ServiceMessages />
&lt;/HeaderProperties></SMSMessageHeader>
    </SMSMessageHeader>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <u:Timestamp u:Id="_0">
        <u:Created>2024-04-27T21:15:01.305Z</u:Created>
        <u:Expires>2024-04-27T21:20:01.305Z</u:Expires>
      </u:Timestamp>
    </o:Security>
  </s:Header>
  <s:Body>
    <s:Fault>
      <faultcode>s:ACTION_FAULTS</faultcode>
      <faultstring xml:lang="en-US">Action faults</faultstring>
      <detail>
        <ActionFaults xmlns="http://schemas.datacontract.org/2004/07/SMS.Broker.Faults"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
          <Faults>
            <ActionFaultEntry>
              <ErrorCode>Z049</ErrorCode>
              <Message>Cargo Manifest/In-Bond/Entry Status query has no parameters enough!</Message>
              <Parameters>BillNumber, InBond, EnrtyNumber</Parameters>
              <Subject>Cargo Release Query Transactions</Subject>
            </ActionFaultEntry>
          </Faults>
        </ActionFaults>
      </detail>
    </s:Fault>
  </s:Body>
</s:Envelope>

```

A sample of fault response message to PutToQueue_CargoManifestEntryReleaseQuery_InBond() method:

```

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <SMSMessageHeader xmlns="http://SMS/DataServer/2014/12">
      <SMSMessageHeader xmlns="Key">&lt;?xml version="1.0" encoding="utf-16"?>
&lt;HeaderProperties xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
&lt;ServiceMessages />
&lt;/HeaderProperties></SMSMessageHeader>
    </SMSMessageHeader>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <u:Timestamp u:Id="_0">
        <u:Created>2024-05-01T01:10:51.332Z</u:Created>
        <u:Expires>2024-05-01T01:15:51.332Z</u:Expires>
      </u:Timestamp>
    </o:Security>
  </s:Header>
  <s:Body>
    <s:Fault>

```

```

<faultcode>s:ACTION_FAULT</faultcode>
<faultstring xml:lang="en-US">In-bond number required!</faultstring>
<detail>
  <ActionFault xmlns="http://schemas.datacontract.org/2004/07/SMS.Broker.Faults"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
    <ErrorCode>Z017</ErrorCode>
    <Message>In-bond number required!</Message>
    <Subject>Cargo manifest entry release query</Subject>
  </ActionFault>
</detail>
</s:Fault>
</s:Body>
</s:Envelope>

```

3. How to prepare HTTP request using wsdl-file.

3.1 PutToQueue_CargoManifestEntryReleaseQuery_InBond() method.

As already was mentioned in *12. AMS Query. SOAP messaging.pdf*, 3. HTTP/SOAP HTTP the messages for all three methods have one and the same HTTP verb and headers (exclusion is for SOAPAction header only):

Verb = POST
ContentType: text/xml; charset="UTF-8"
Host: smstest.logisticaldatasolutions.com:8130
Content - Length: <n>
Expect: 100 -continue
Accept - Encoding: gzip, deflate

SOAPAction header we will find in wsdl-file.

HTTP body consists of SOAP xml which in its turn has own envelop part with header and body parts included. Envelope stuff is a standard parts of xml SOAP messages and header stuff is a standard part in SMS-system.

SOAP header is always:

```

<s:Header>
  <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <o:UsernameToken>
      <o:Username/>
      <o:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText"/>
    </o:UsernameToken>
  </o:Security>
</s:Header>

```

Respectively, to complete SOAP part of HTTP request we still need to define only:

1. SOAPAction header for HTTP
2. SOAP body part.

From *07. ABI classes and interfaces.pdf*, 3. *ABI SMS Manager* we know that PutToQueue_CargoManifestEntryReleaseQuery_InBond() is a method of `SmsABIManager` class. It is available as service via `SMS.Broker.ServiceContracts.ISmsABIManager` interface. Wsdl-file of `SmsABIManager` service can be accessed on:

<http://smssystem.logisticaldatasolutions.com:8110/BrokerService/SmsABIManager/MEX>

We enter this URI to browser address bar and get wsdl-file. Then we find

```
wsdl:operation name="PutToQueue_CargoManifestEntryReleaseQuery_InBond"
```

line in the text. We see two such lines but needs the one that is placed in `wsdl:binding` section. It has a child element `soap:operation` that contains attribute:

```
soapAction="http://tempuri.org/ISmsABIManager/PutToQueue_CargoManifestEntryReleaseQuery_InBond".
```

This URI is a desired value for SOAPAction HTTP parameter. It should be added with quotations.

As for SOAP part for now we still have a body part unknown:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <o:UsernameToken>
        <o:Username/>
        <o:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText"/>
      </o:UsernameToken>
    </o:Security>
  </s:Header>
  <s:Body>

</s:Body>
</s:Envelope>
```

Method's name PutToQueue_CargoManifestEntryReleaseQuery_InBond is an operation contract name. This wsdl-file uses "document-literal wrapped" binding type and hence PutToQueue_CargoManifestEntryReleaseQuery_InBond is not only a name of operation but also a name of element defined in type section. In `<wsdl:types>` section we select the schema with schema location

<http://smstest.logisticaldatasolutions.com:8110/BrokerService/SmsABIManager/MEX?xsd=xsd4>

and use the location as URI (enter it to browser address bar and get the second wsdl-file). In this file we find

```
element name="PutToQueue_CargoManifestEntryReleaseQuery_InBond"
```

line.

As a result we see:

```
<xs:element name="PutToQueue_CargoManifestEntryReleaseQuery_InBond">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="inBondNumber" nillable="true" type="xs:string"/>
      <xs:element minOccurs="0" name="clientRef" nillable="true" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

In particular it means that PutToQueue_CargoManifestEntryReleaseQuery_InBond() method has two parameters: `string` inBondNumber and `string` clientRef.

Target namespace in wsdl-files is `http://tempuri.org/` (`targetNamespace="http://tempuri.org/"`). We have to add this namespace to PutToQueue_CargoManifestEntryReleaseQuery_InBond element:

```
<PutToQueue_CargoManifestEntryReleaseQuery_InBond xmlns="http://tempuri.org/">
```

And taking into account known input parameters we can write:

```
<PutToQueue_CargoManifestEntryReleaseQuery_InBond xmlns="http://tempuri.org/">
  <inBondNumber/>
  <clientRef/>
</PutToQueue_CargoManifestEntryReleaseQuery_InBond>
```

We add it as a body to our xml SOAP and finally have:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <s:Header>
    <o:Security s:mustUnderstand="1" xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <o:UsernameToken>
        <o:Username/>
```

```

        <o:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText"/>
        </o:UsernameToken>
    </o:Security>
</s:Header>
<s:Body>
    <PutToQueue_CargoManifestEntryReleaseQuery_InBond xmlns="http://tempuri.org/">
        <inBondNumber/>
        <clientRef/>
    </PutToQueue_CargoManifestEntryReleaseQuery_InBond>
</s:Body>
</s:Envelope>

```

Just after PutToQueue_CargoManifestEntryReleaseQuery_InBond element definition we see

```

<xs:element name="PutToQueue_CargoManifestEntryReleaseQuery_InBondResponse">
    <xs:complexType>
        <xs:sequence>
            <xs:element minOccurs="0" name="PutToQueue_CargoManifestEntryReleaseQuery_InBondResult"
type="xs:long"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

```

It means that request output parameter name is

PutToQueue_CargoManifestEntryReleaseQuery_InBondResponse and parameter has a [long](#) type (really it is an Id of outgoing transmission added to queue on SMS-server). A sample of the response you can see in [2.4 PutToQueue_CargoManifestEntryReleaseQuery_InBond\(\) method](#) of this document.