

Importer Query. SOAP messaging.

(ver. from 8/31/2025)

Table of Contents

1. Services.	1
2. Importer Query workflow.	2
3. HTTP/SOAP.	3
3.1 PutToQueue() method.	3
3.2 GetAnswers() method.	4
3.3 GetABIEvents() method.	4
3.4 GetByNumber() method.	5
3.5 Exceptions.	5

SMS-server was built on .NET platform but finally all data exchange between client and server is based on HTTP/SOAP messaging. Respectively, on a simple cases the interaction with server can dispense with .NET. The follow document describes how to built Importer Query client part using only HTTP/SOAP requests.

1. Services.

First of all we need to know necessary services which client should apply to. On Importer Query case there are four such services: SmsABIManager (to run PutToQueue() method), ABITransmissionManager (to run GetAnswers() method), EntityEventManager (to run GetABIEvents() method) and ContactManager (GetByNumber() method).

On **test** SMS server they can be accessed by URI-s:

<https://smstest.logisticaldatasolutions.com:8130/brokerservice/SmsABIManager>
<https://smstest.logisticaldatasolutions.com:8130/brokerservice/ABITransmissionManager>
<https://smstest.logisticaldatasolutions.com:8130/brokerservice/EntityEventManager>
<https://smstest.logisticaldatasolutions.com:8130/brokerservice/ContactManager>

Test SMS server URL: <https://smstest.logisticaldatasolutions.com:8130/brokerservice>

Metadata wsdl-files on **production** SMS server are available on:

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

2. Importer Query workflow.

Step 1: Client sends to SmsABIManager service a request to run PutToQueue() method. Service adds an outgoing transmission object with Importer query to transmission queue. Synchronous PutToQueue() method returns to client a transmission Id. This Id will be used to return the result of query.

Step 2: To know a result of Importer Query the client sends to ABITransmissionManager service a request to run GetAnswers() method. Transmission Id found out on the previous step is its input parameter. In general case GetAnswers() method returns a list of related incoming transmissions. In our case (Importer Query) it will be only one transaction. This transaction contains Customs response in a special Customs format. To get more readable conventional text we need a next step.

Step 3: Among other data the returned transmission object contains a link to a query object that can be used to apply to EntityEventManager service. Client sends to it a request to run GetABIEvents() method that returns an event object with Customs response in conventional text format.

PutToQueue() method is synchronous and it waits until service adds an outgoing transmission object on server side and returns transmission Id. However, there is also a time lag between when transmission object is put to queue and when incoming transmission object with Customs response appears.

This inconvenience can be overcome by using a loop in a code with a time delay for GetAnswer() method or by dividing the process into two parts: first part returns transmission Id by PutToQueue() request and another part sends GetAnswers() and GetABIEvents() methods using saved transmission Id.

If you are familiar with AMS Query you can see that Importer Query workflow steps look the same. However, PutToQueue() method has another set of parameters and depending on them the process can be "ABI Query based" or "Contact based".

1. "ABI Query based" scenario:

In this case you still don't have Contact object in SMS-database and know only its importer number (IRS, SSN, CBP Assigned Number). You run PutToQueue() method with ImporterNumber value and then the Transmission object and ABIQuery objects are created on SMS-database. You run GetAnswers() method, get a link to created ABIQuery object and by this link (using GetABIEvents() method) find event with Customs info about importer in event Text property.

Available Action input parameter values for PutToQueue() method on this scenario are QNA or QN.

Note 1. How to know if Contact object already exists in SMS-database.

QNA parameter of PutToQueue() method suggests that requested Importer will be added to SMS-database as Contact object if information successfully returns from Customs. Hence, before to run PutToQueue() method with Action = "QNA" the necessity can appear to know if importer with requested importer number already exists in SMS-database. For this purpose you can use GetByNumber() method of Contact Manager that returns Contact object by importer number (please see 3.1 PutToQueue() method. below).

2. "Contact based" scenario:

You already have Contact object of importer in database and know its Id.

You run PutToQueue() method with SourceLink equal to value constructed from known Contact Id as "Contact=" + Id.ToString() (for example "Contact=11").

Available Action input parameter values for PutToQueue() method on this scenario are B, I or N.

Here from the beginning you already know a link you can use to find an event with importer info when it appears but it is possible that Customs reply still didn't come to SMS-server at this moment. So, it is better to follow the same algorithm as on "ABI Query based" scenario - run GetAnswers() method to be sure that Customs has replied and this reply has been placed on SMS-server. And only then use Contact link as a link parameter value to find event with importer info (GetABIEvents()).

And one important addition. In "ABI Query based" case there is always only one event related to Customs response. In contrast to them, on "Contact based" scenario they can be many event objects accumulated by Contact object during a long time. The event related to Customs response will be the last (added to the end

of events collection). It is necessary to take it into account when parsing incoming GetABIEventsResponse xml-file.

To send/receive HTTP requests for method calls we need HTTP parameters and xml SOAP templates. In general case you can get all necessary metadata from our SMS-server (please, see item 1. of this document). Then you can select methods and their parameters manually from wsdl-files (Please see 13. Annex. AMS Query. SOAP messaging, 3. How to prepare HTTP request using wsdl-file.) or using some special tool like SoapUI.

Detailed description of data types you can see in 07. ABI classes and interfaces.pdf.

On our case we already have necessary SOAP templates prepared. You need only to add a necessary input parameters before to send an HTTP requests.

3. HTTP/SOAP.

HTTP messages for all four 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

Also SOAP xml must include username (local name Username) and password (local name Password) values.

3.1 PutToQueue() method.

HTTP header to be added: SOAPAction: "http://tempuri.org/ISmsABIManager/PutToQueue"

SOAP template looks as follows:

```
<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 xmlns="http://tempuri.org/">
      <Appld>KI</Appld>
      <parameters xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <Parameter xmlns="">
          <Name>SourceLink</Name>
          <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
        </Parameter>
        <Parameter xmlns="">
          <Name>ImporterNumber</Name>
          <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
        </Parameter>
        <Parameter xmlns="">
          <Name>Action</Name>
          <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
        </Parameter>
      </parameters>
    </PutToQueue>
  </s:Body>
</s:Envelope>
```

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

Input parameters.

A set of input parameters depends on used scenario.

ABI based scenario:

string (2) Appld - it is always equal to KI. It is Customs code of "Importer/Bond query" application,
string (3) Action - it depends on what result you want to get (to add a new Contact to database, to know only info about Contact.),

"ABI Query based" scenario:

QNA – it means that besides returning of info about importer the **SMS-system creates a new Contact object in SMS-database** using this info. Contact for importer is created despite on if the importer already exists in SMS-database or not.

QN – only info about importer is returned (without new Contact object creating).

"Contact based" scenario:

B – Info without address is returned and Contact object is updated with the new info about existed bonds.

N – Info including name and address is returned and Contact name/address is updated with the name/address info.

I – Info without address is returned (without Contact object update).

string (12) ImporterNumber - IRS, SNN (Social Security Number) or CBP Assigned Number of importer in standard Customs format,

string SourceLink - a link to a source object (also see details below).

Output parameter:

long PutToQueueResult – transmission Id returned by SMS-server.

Besides that this SOAP template already contains hardcoded Appld = "KI" Customs application type that means "Importer/Bond Query".

A sample of response xml SOAP message you can see in 15. *Annex. Importer Query. SOAP messaging.pdf, 1. Samples of xml SOAP response message, 1.1 PutToQueue() method.*

Output parameter.

long PutToQueueResult – transmission Id.

3.2 GetAnswers() method.

Please, see 12. *AMS Query. SOAP messaging.pdf, 3.2 GetAnswers() method.*

3.3 GetABIEvents() method.

Please, see 12. *AMS Query. SOAP messaging.pdf, 3.3 GetABIEvents() method.*

3.4 GetByNumber() method.

HTTP header to be added: SOAPAction: "http://tempuri.org/IContactManager/GetByNumber"
SOAP template looks as follows:

```
<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>
    <GetByNumber xmlns="http://tempuri.org/">
      <number/>
    </GetByNumber>
  </s:Body>
</s:Envelope>
```

Input parameters.

string (12) number – IRS, SNN (Social Security Number) or CBP Assigned Number of importer in standard Customs format.

A sample of response xml SOAP message you can see in 15. *Annex. Importer Query. SOAP messaging.pdf*, 1. *Samples of xml SOAP response message*, 2.2 *GetAnswers() method*.

Output parameters.

GetByNumberResult – contains Contact objects.

3.5 Exceptions.

A sample of response xml SOAP message you can see in 13. *Annex. AMS Query. SOAP messaging.pdf*, 2. *Samples of xml SOAP response message*, 2.4 *Exceptions*.

The detailed description of **ActionFaults** object you can see in 05. *Base, common usage classes and interfaces.pdf*.