

AMS Query. SOAP messaging.

(ver. from 5/26/2026)

Table of Contents

1. Services.	1
2. ABI Query workflow.	1
3. HTTP/SOAP.	2
3.1 PutToQueue() method.	2
3.2 GetAnswers() method.	4
3.3 GetABIEvents() method.	4
3.4 Exceptions.	5
4. Sample of AMS Query implementation.	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 AMS Query client part using only HTTP/SOAP requests.

1. Services.

First of all we need to know necessary services which client should apply to. There are three such services: SmsABIManager (to run PutToQueue() method), ABITransmissionManager (to run GetAnswers() method) and EntityEventManager (to run GetABIEvents() 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>

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>

2. ABI Query workflow.

Step 1: Client sends to SmsABIManager service a request to run PutToQueue() method. Service adds an outgoing transmission object with AMS 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 AMS 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 (AMS 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 in two parts: first part returns transmission Id by PutToQueue() request and another part sends GetAnswers() and GetABIEvents() methods using saved transmission Id.

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 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

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/>
</s:Envelope>
```

```

</s:Header>
<s:Body>
  <PutToQueue xmlns="http://tempuri.org/">
    <ApplId>CQ</ApplId>
    <parameters xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
      <Parameter xmlns="">
        <Name>EntryFilerCode</Name>
        <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
      </Parameter>
      <Parameter xmlns="">
        <Name>EntryNumber</Name>
        <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
      </Parameter>
      <Parameter xmlns="">
        <Name>InBondNumber</Name>
        <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
      </Parameter>
      <Parameter xmlns="">
        <Name>SCAC</Name>
        <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
      </Parameter>
      <Parameter xmlns="">
        <Name>BillNumber</Name>
        <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
      </Parameter>
      <Parameter xmlns="">
        <Name>AirMasterBOL</Name>
        <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
      </Parameter>
      <Parameter xmlns="">
        <Name>AirHouseBOL</Name>
        <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
      </Parameter>
      <Parameter xmlns="">
        <Name>RequestRelatedBOL</Name>
        <Value i:type="a:boolean" xmlns:a="http://www.w3.org/2001/XMLSchema">false</Value>
      </Parameter>
      <Parameter xmlns="">
        <Name>RequestBOLEntryData</Name>
        <Value i:type="a:boolean" xmlns:a="http://www.w3.org/2001/XMLSchema">false</Value>
      </Parameter>
      <Parameter xmlns="">
        <Name>ClientRef</Name>
        <Value i:type="a:string" xmlns:a="http://www.w3.org/2001/XMLSchema"/>
      </Parameter>
    </parameters>
  </PutToQueue>
</s:Body>
</s:Envelope>

```

Input parameters.

A set of input parameters depends on a type of AMS Query you send to Customs: Bill of Lading Query (non-air), Airway Bill, In-Bond, Cargo Release (Entry). In brackets the maximum size is indicated.

Bill of Lading Query:

string (4) SCAC	- Standard Carrier Alpha Code (mandatory)
string (12) BillNumber	- Bill of Lading Number for non-air delivery (mandatory)
boolean RequestRelatedBOL	- request for related bill (optional for Customs)
string (20) ClientRef	- special option to relate request and some client info (optional)

Airway Bill Query:

string 11 AirMasterBOL	- Master Bill for air delivery (mandatory)
string 12 AirHouseBOL	- House Bill for air delivery (optional for Customs)
string (20) ClientRef	- special option to relate request and some client info (optional)

In-Bond Query:

string (12) InBondNumber	- In-Bond Number (mandatory)
string (20) sClientRef	- special option to relate request and some client info (optional)

Cargo Release (Entry):

string (3) EntryFilerCode	- Filer Code (mandatory)
---------------------------	--------------------------

string (9) EntryNumber - Cargo Release Entry Number (mandatory)
 boolean RequestBOLEntryData - (optional for Customs)
 string (20) ClientRef - special option to relate request and some client info (optional)

Besides that this SOAP template already contains hardcoded Appld = "CQ" Customs application type that means "ACE Cargo Manifest / In-Bond / Entry Status Query".

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*, 2.1 *PutToQueue() method*.

Output parameter.

long PutToQueueResult - transmission Id.

3.2 GetAnswers() method.

HTTP header to be added: SOAPAction: "http://tempuri.org/IABITransmissionManager/GetAnswers"
 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>
    <GetAnswers xmlns="http://tempuri.org/">
      <requestId/>
    </GetAnswers>
  </s:Body>
</s:Envelope>
```

Input parameters.

long requestId - transmission id returned by PutToQueue() method in PutToQueueResult parameter.

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*, 2.2 *GetAnswers() method*.

Output parameters.

GetAnswersResult - contains transmission objects. On the case of ABI Query it is always one object (we need its SourceLink parameter)

3.3 GetABIEvents() method.

HTTP header to be added: SOAPAction: "http://tempuri.org/IEntityEventManager/GetABIEvents"
 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>
    <GetABIEvents xmlns="http://tempuri.org/">
      <requestId/>
    </GetABIEvents>
  </s:Body>
</s:Envelope>
```

```

    </o:UsernameToken>
  </o:Security>
</s:Header>
<s:Body>
  <GetABIEvents xmlns="http://tempuri.org/">
    <link/>
    <includeText/>
  </GetABIEvents>
</s:Body>
</s:Envelope>

```

Input parameters.

<code>string</code> link	- ABIQuery link (it looks something like "ABIQuery=628", for example)
<code>boolean</code> includeText	- include a text of Customs response in conventional format.

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, 2.3 GetABIEvents() method.*

Output parameters.

GetABIEventsResult	- contains event objects. On the case of ABI Query it is also always one object (we need its Text parameter)
--------------------	--

3.4 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*.

4. Sample of AMS Query implementation.

In 13. *Annex. AMS Query. SOAP messaging, Sample of AMS Query implementation.* you can see a simplified realization of all three steps mentioned above. The sample was created only in demonstrating purposes.