

SMS Broker System
Software Integration

Our system uses
WCF
(Windows Communication Foundation)
&
.NET

Annex. ABI classes and interfaces.

(ver. from 8/3/2026)

Table of Contents

1. AMS (Accounting Manifest System) and ERF (Extract Reference File) Queries.	1
1.1 AMS query samples.	1
1.2 ERF query samples.	3
2. ISmsABIManager PutToQueue() and PutToQueueOneWay() parameters.	4
3. Samples of put to queue methods with parameters.	10
3.1 How to create new Contact object by request to Customs.	10
3.2 HTS update by tariff range.	11
3.3 Importer Name / Bond query without adding an importer contact to database.	12

1. AMS (Accounting Manifest System) and ERF (Extract Reference File) Queries.

1.1 AMS query samples.

All AMS special put to queue methods of [ISmsABIManager](#) interface:

PutToQueue_CargoManifestEntryReleaseQuery_Bill()	- bill of lading query,
PutToQueue_CargoManifestEntryReleaseQuery_AirBill()	- airway bill query,
PutToQueue_CargoManifestEntryReleaseQuery_Entry()	- entry query,
PutToQueue_CargoManifestEntryReleaseQuery_InBond()	- in-bond query

are synchronous and hence we can apply catch exception handling for validation and also use returned ABI transmission Id. Pay attention that in a catch blocks below we use [ActionFaults](#) validation class in contrast to a case with put to queue methods of document or directory class objects (there we use [EntryValidationFault](#) validation class).

More detailed description of [ISmsABIManager](#) interface methods for AMS queries you can see in *07. ABI classes and interfaces.pdf*, 3. *ABI SMS Manager*.

Detailed description of validation classes you can see in *05. Base, common usage classes and interfaces.pdf*, 13. *Validation*.

Sample 1.

Master Airway Bill plus House Airway Bill query:

```
public static long AMS_Bill_Request(string airMasterBOL, string airHouseBOL, string clientRef)
{
    // Initialize Client Context. It must be your connection data here:
```

```

string url = @"https://localhost:8130/brokerservice";
string username = "admin";
string database = "sandbox";
string password = "somepassword";
SMS.Broker.ClientContext.InitializeContext(url, username + "@" + database, password);

// Prepare manager:
SMS.Broker.ServiceContracts.ISmsABIManager mngrSmsABI =
SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.ISmsABIManager>();

long ABITransmissionId = 0;

try
{
    ABITransmissionId = mngrSmsABI.PutToQueue_CargoManifestEntryReleaseQuery_AirBill(airMasterBOL, airHouseBOL, clientRef);
}
catch (FaultException<ActionFaults> ex)
{
    // Your some catch handling, for example a record to log file.
}

return ABITransmissionId;
}

```

Sample 2.

Entry query (we change only put to queue method and input parameters here):

```

public static long AMS_Entry_Request(string entryFileCode, string entryNumber, bool requestBOLEntryData, string clientRef)
{
    // Initialize Client Context. It must be your connection data here:
    string url = @"https://localhost:8130/brokerservice";
    string username = "admin";
    string database = "sandbox";
    string password = "somepassword";
    SMS.Broker.ClientContext.InitializeContext(url, username + "@" + database, password);

    // Prepare manager
    SMS.Broker.ServiceContracts.ISmsABIManager mngrSmsABI =
    SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.ISmsABIManager>();

    long ABITransmissionId = 0;

    try
    {
        ABITransmissionId = mngrSmsABI.PutToQueue_CargoManifestEntryReleaseQuery_Entry(entryFileCode, entryNumber,
        requestBOLEntryData, clientRef);
    }
    catch (FaultException<ActionFaults> ex)
    {
        // Your some catch handling, for example a record to log file.
    }

    return ABITransmissionId;
}

```

Sample 3.

When we have returned outgoing ABI transmission Id (i.e. our request transmission Id) we can directly find Customs reply (our incoming) ABI transmission and bypassing [ABIQuery](#) class object get related event:

```

public static string GetEventText(long ABITransmissionId)
{
    // Initialize Client Context. It must be your connection data here:
    string url = @"https://localhost:8130/brokerservice";
    string username = "admin";
    string database = "sandbox";
    string password = "somepassword";
    SMS.Broker.ClientContext.InitializeContext(url, username + "@" + database, password);

    string text = "";

    // Prepare managers
    SMS.Broker.ServiceContracts.Documents.IABITransmissionManager mngrABITransmission =
    SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Documents.IABITransmissionManager>();
    SMS.Broker.ServiceContracts.Common.IEntityEventManager mngrEntityEvent =
    SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Common.IEntityEventManager>();

    // Get Customs replay ABITransmission:
    List<SMS.Broker.DataContracts.Documents.ABITransmission> ABITransmissions = mngrABITransmission.GetAnswers(ABITransmissionId);
    if (ABITransmissions.Count == 0) // If there is no response yet.
    {

```

```

    return "";
}

// Here we get an ABITransmission from returned list. Customs response to ABI query is always only the one
// but in other cases (for some documents) they can be more then one response to one outgoing ABI transmission.
SMS.Broker.DataContracts.Documents.ABITransmission ABITransmission = ABITransmissions[0];

// ABITransmission object contains a link to its source. Now it is an ABI Query.
// Its link looks like "ABIQuery=159" (Id = 159, for example)
// We get a link for our ABI Query:
string ABIQuerylink = ABITransmission.SourceLink;

// When we know a link we can use it to find related events.
// Here we read Customs ABI response (a list of ABI events)
// from SMS-server by found link and select an ABI event.
// On ABI Query case ABI event is only one also. (We can take it as ABIQueryEvents[0])

List<EntityEvent> ABIQueryEvents = mngrEntityEvent.GetABIEvents(ABIQuerylink, true);

if (ABIQueryEvents.Count != 0)
    text = ABIQueryEvents[0].Text; // This text is a response
                                   // from Customs ABI formatted
                                   // as a user-oriented text.

return text;
}

```

1.2 ERF query samples.

All AMS special put to queue methods of [ISmsABIManager](#) interface:

<code>_PutToQueue_ExtractReferenceFile_FIRMSByBeginDate()</code>	- update FIRMS directory by begin date,
<code>_PutToQueue_ExtractReferenceFile_FIRMSByCode()</code>	- update FIRMS directory by FIRMS code,
<code>PutToQueue_ExtractReferenceFile_Carriers()</code>	- update all carriers,
<code>PutToQueue_ExtractReferenceFile_Countries()</code>	- update all countries,
<code>PutToQueue_ExtractReferenceFile_CustomsPorts()</code>	- update all Customs ports,
<code>PutToQueue_ExtractReferenceFile_FIRMS()</code>	- update all FIRMS directory
<code>PutToQueue_ExtractReferenceFile_ForeignPorts()</code>	- update all foreign ports.

are synchronous. The same as in *1.1 AMS query samples* we can apply catch exception handling for validation and also use returned ABI transmission Id. However, the main purpose of this methods is to renew directories ([Carriers](#), [CountryInfoes](#), [CustomsPorts](#), [FIRMS](#) and [ForeignPorts](#)) with Customs data. Pay attention that in a catch blocks below we use [ActionFaults](#) validation class in contrast to a case with put to queue methods of document or directory class objects (there we use [EntryValidationFault](#) validation class).

More detailed description of [ISmsABIManager](#) interface methods for ERF queries you can see in *07. ABI classes and interfaces.pdf*, 3. *ABI SMS Manager*.

Detailed description of Extract Reference File classes you can see in *09. Extract Reference File And Currency Rate classes.pdf*.

Detailed description of validation classes you can see in *05. Base, common usage classes and interfaces.pdf*, 13. *Validation*.

Sample 1.

FIRMS query by FIRMS code:

```

public static long ERF_FIRMSQuery(string FIRMSCode)
{
    // Initialize Client Context. It must be your connection data here:
    string url = @"https://localhost:8130/brokerservice";
    string username = "admin";
    string database = "sandbox";
    string password = "somepassword";
    SMS.Broker.ClientContext.InitializeContext(url, username + "@" + database, password);

    // Prepare manager
    SMS.Broker.ServiceContracts.ISmsABIManager mngrSmsABI =

```

```

SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.ISmsABIManager>();

long ABITransmissionId = 0;
try
{
    ABITransmissionId = mngrSmsABI._PutToQueue_ExtractReferenceFile_FIRMSByCode(FIRMSCode);
}
catch (System.ServiceModel.FaultException<SMS.Broker.Faults.ActionFaults> ex)
{
    // Your some catch handling, for example a record to log file.
}

return ABITransmissionId;
}

```

Sample 2.

Carriers query (we change only put to queue method and input parameters here):

```

public static long ERF_CountriesQuery()
{
    // Initialize Client Context. It must be your connection data here:
    string url = @"https://localhost:8130/brokerservice";
    string username = "admin";
    string database = "sandbox";
    string password = "somepassword";
    SMS.Broker.ClientContext.InitializeContext(url, username + "@" + database, password);

    // Prepare manager
    SMS.Broker.ServiceContracts.ISmsABIManager mngrSmsABI =
    SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.ISmsABIManager>();

    long ABITransmissionId = 0;
    try
    {
        ABITransmissionId = mngrSmsABI.PutToQueue_ExtractReferenceFile_Countries();
    }
    catch (System.ServiceModel.FaultException<SMS.Broker.Faults.ActionFaults> ex)
    {
        // Your some catch handling, for exmple a record to log file.
    }

    return ABITransmissionId;
}

```

2. ISmsABIManager PutToQueue() and PutToQueueOneWay() parameters.

ISmsABIManager put to queue operation methods:

synchronous

PutToQueue(string AppId, List<Parameter> parameters)

and asynchronous

PutToQueueOneWay(string AppId, List<Parameter> parameters)

have the same set of parameters.

string AppId – standard Customs application code from CATAIR ("AE", "SU" and so on);

List<Parameter> parameters – parameters of put to queue operation method. For more details about Parameter class see 05. Other classes and interfaces. Part 1, 4. Parameter class.

A list of Parameter class objects can be used in different methods but on this document we consider how they are used on put to queue operation methods.

a. If a document (or directory) object to which the put to queue operation is applied already exists then the list of parameters must contain an entity link to this object (see 05. Base, common usage classes and interfaces.pdf, 5. EntityLink and 02. Annex. Getting Started With SMS API.pdf, 4. Entity Link).

If a document (or directory) is created as result of put to queue operation (document – ABI query, Contact objects) entity link is not used.

b. For document and directory objects one of parameters must be Action parameter.

c. Additional parameters can be required.

[ISmsABIManager](#) realizes a universal approach to run put to queue operation methods. The special AMS and ERF methods of [ISmsABIManager](#) reviewed in 1.1 – 1.2 and also put to queue methods of document and directory class managers don't cover all possible cases. [ISmsABIManager](#) PutToQueue() and PutToQueueOneWay() methods gives an access to all options as for document submitting that available in SMS-system.

In a table below you can see parameters those are applied in each case. "Equivalent enumerator value" column is added for comparison only to see an equivalent form of put to queue methods used by another managers, for example by [ICustomsEntryManager](#) or [IContactManager](#). Enumerator values are not used by [ISmsABIManager](#) put to queue methods.

For example, we want to put to queue 7501 CBP form. We create a manager object and parameters collection:

```
SMS.Broker.ServiceContracts.ISmsABIManager mngrSmsABI =  
SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.ISmsABIManager>();  
  
List<SMS.Broker.DataContracts.Parameter> parameters = new List<SMS.Broker.DataContracts.Parameter>();
```

Then we find in a table a line for "Entry Summary (7501 CBPF) add" in "Description" column. In "SourceLink" column it contains "Yes." It means that we have to add this parameter to the list (ce.Link is a [string](#) type entity link to [CustomsEntry](#) class object. In general "Class" column defines an object type for the link.):

```
parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "SourceLink", Value = ce.Link });
```

We also see that "Action" parameter must be "A" and add it as the second parameter:

```
parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "Action", Value = "A" });
```

We take [string](#) AppId parameter (it is not included to parameters list variable) from "Customs app Id" column and now we can run put to queue method:

```
mngrSmsABI.PutToQueue("AE", parameters);
```

See samples 2. *How to create new Contact object by request to Customs.* and 3. *HTS update by tariff range.* below in this document.

Class	Customs app. Id	"SourceLink" parameter	"Action" parameter	Other parameters	Equivalent enumerator value	Description
CustomsEntry	"AE"	Yes.	"A"		CustomsEntry.ActionQueue.AddEntrySummary	Entry Summary (7501 CBPF) add.
	"AE"	Yes.	"R"		CustomsEntry.ActionQueue.ReplaceEntrySummary	Entry Summary (7501 CBPF) replace.
	"AE"	Yes.	"D"		CustomsEntry.ActionQueue.DeleteEntrySummary	Entry Summary (7501 CBPF) delete.
	"AE"	Yes.	"A+"		CustomsEntry.ActionQueue.AddEntrySummaryAndCargoRelease	Entry Summary (7501 CBPF) and Cargo Release

						(3461 CBPF) add.
	"AE"	Yes.	"R+"		CustomsEntry.ActionQUE.ReplaceEntrySummaryAndCargoRelease	Entry Summary (7501 CBPF) and Cargo Release (3461 CBPF) replace.
	"AE"	Yes.	"AP"		Absent.	Paid Entry Summary (7501 CBPF) replace.
	"CA"	Yes.	"R"	"Articles": ShipmentArticle object Id values as string separated by semicolons.	Absent.	PGA Data Corrections
	"SU"	Yes.	"A"		CustomsEntry.ActionQUE.ACHUpdate	ACH update.
	"JC"	No.	"A"	"EntryFilerCode" "EntryNumber"	CustomsEntry.ActionQUE.EntrySummaryQuery	Entry Summary Query.
	"TE"	Yes.	"E"		CustomsEntry.ActionQUE.TemporaryImportationBondExtend	Temporary Importation Bond – Extension.
	"TE"	Yes.	"C"		CustomsEntry.ActionQUE.TemporaryImportationBondClose	Temporary Importation Bond – Closure.
	"SE"	Yes.	"A"		CustomsEntry.ActionQUE.AddCargoRelease	Cargo Release (3461 CBPF) add.
	"SE"	Yes.	"R"		CustomsEntry.ActionQUE.ReplaceCargoRelease	Cargo Release (3461 CBPF) replace.
	"SE"	Yes.	"D"		CustomsEntry.ActionQUE.CancelCargoRelease	Cargo Release (3461 CBPF) cancel.
	"SE"	Yes.	"U"		CustomsEntry.ActionQUE.BOLUpdate	Cargo Release (3461 CBPF) Bill of Lading & Manifest information update.
ImporterSecurityFiling	"SF"	Yes.	"A"		ImporterSecurityFiling.ActionQUE.Add	Importer Security Filing (ISF-5, ISF-10) add.
	"SF"	Yes.	"R"		ImporterSecurityFiling.ActionQUE.Replace	Importer Security Filing (ISF-5, ISF-10) replace.
	"SF"	Yes.	"D"		ImporterSecurityFiling.ActionQUE.Delete	Importer Security Filing (ISF-5, ISF-10) delete.
Contact	"\$I"	Yes.	"A"		Contact.ActionQUE.AddManufacturerNameAddress	Add Manufacturer Name and Address.
	Absent.	Absent.	Absent.		Contact.ActionQUE.AddManufacturerNameAddressIfNeeded	Add Manufacturer Name and Address.

						Check if manufacturer identifier (MID) code already presents in SMS-database and if not, sends the request to Customs.
	"KI"	Yes.	"B"		Contact.ActionQUE.ImporterBondQuery	Importer/Bond Query. Returns only Bond info (without Name and Address).
	"KI"	Yes.	"N"		Contact.ActionQUE.ImporterBondQueryNameAndAddresses	Importer/Bond Query. Returns Bond info with Name and Address.
	"TP"	Yes.	"A"		Contact.ActionQUE.ImporterConsigneeCreate	Importer/Consignee Create/Update (CBPF 5106) add.
	"TP"	Yes.	"U"		Contact.ActionQUE.ImporterConsigneeUpdate	Importer/Consignee Create/Update (CBPF 5106) update.
	"TP"	Yes.	"N"		Contact.ActionQUE.ApplyForCBPAssignedNumber	Importer/Consignee Create/Update (CBPF 5106) apply for a CBP assigned number.
	"GE"	Yes.	"A"		Contact.ActionQUE.GlobalBusinessIdentifierAdd	Global Business Identifier (GE) Enrollment add.
	"GE"	Yes.	"U"		Contact.ActionQUE.GlobalBusinessIdentifierUpdate	Global Business Identifier (GE) Enrollment update.
	"GE"	Yes.	"D"		Contact.ActionQUE.GlobalBusinessIdentifierDelete	Global Business Identifier (GE) Enrollment delete.
StatementDaily	"RM"	Yes.	"A"		Absent.	Daily Statement authorizing.
Drawback	"DE"	Yes.	"A"		Absent.	Drawback Entry Summary add.
	"DE"	Yes.	"R"		Absent.	Drawback Entry Summary replace.
FTZe214	"FT"	Yes.	"A"		FTZe214.ActionQUE.AddFTZe214	FTZ Admission (CBPF214) add.

	"FT"	Yes.	"R"		FTZe214.ActionQUE.ReplaceFTZe214	FTZ Admission (CBPF214) replace.
	"FT"	Yes.	"D"		FTZe214.ActionQUE.DeleteFTZe214	FTZ Admission (CBPF214) delete.
	"FT"	Yes.	"S"		FTZe214.ActionQUE.ChangeZoneStatusFTZe214	FTZ Admission (CBPF214) Merchandise Zone Status change.
	"FZ"	Yes.	"H"	"LevelLink": "1"	FTZe214.ActionQUE.AdmissionAcceptance	Operator transmits Acceptance.
	"FZ"	Yes.	"G"	"LevelLink": "1"	FTZe214.ActionQUE.AdmissionRefusal	Operator transmits Refusal.
	"FZ"	Yes.	"F"	"LevelLink": "2", "4"	FTZe214.ActionQUE.Bill ePTTRequest FTZe214.ActionQUE.ContainerPTTRequest	Operator transmits ePTT.
	"FZ"	Yes.	"K"	"LevelLink": "2"	FTZe214.ActionQUE.Bill ePTTCancelation	Operator transmit Permit to Transfer cancel.
	"FZ"	Yes.	"I"	"LevelLink": "2", "3", "4"	FTZe214.ActionQUE.Bill GoodsArrival FTZe214.ActionQUE.InBondGoodsArrival FTZe214.ActionQUE.ContainerGoodsArrival	Operator transmits Arrival.
	"FZ"	Yes.	"A"	"LevelLink": "1"	FTZe214.ActionQUE.AdmissionConcurrence	Operator transmits Concurrence on admission level.
	"FZ"	Yes.	"B"	"LevelLink": "2"	FTZe214.ActionQUE.Bill Concurrence	Operator transmits Concurrence on bill level.
	"FZ"	Yes.	"C"	"LevelLink": "3"	FTZe214.ActionQUE.InBondConcurrence	Operator transmits Concurrence on In-Bond level.
	"FZ"	Yes.	"D"	"LevelLink": "4"	FTZe214.ActionQUE.ContainerConcurrence	Operator transmits Concurrence on container level.
	"FZ"	Yes.	"J"	"LevelLink": "1"	FTZe214.ActionQUE.AdmissionPostCorrection	Operator transmit Post Admission Correction.
InBond	"QP"	Yes.	"A"		Absent.	In-Bond (CBPF-7512) add.
	"QP"	Yes.	"B"	"Bill_Id"	Absent.	Delete In-Bond

						(CBPF-7512) from bill.
	"QP"	Yes.	"D"		Absent.	Delete In-Bond (CBPF-7512) from all associated bills.
InBondUpdate	"WP"	Yes.	"A"		Absent.	In-Bond Arrival / Export / Transfer of Liability transmission.
ABIQuery	"AD"	No.	Absent.	"CaseNumber" "CompanyCaseStatus" "CountryCode" "HTSNumber" "TSUSANumber" "ManufacturerId" "ForeignExporterId" "DateSinceLastUpdate"	Absent.	AD/CVD Case Information Query submitting.
	"CQ"	No.	Absent.	"EntryFilerCode" "EntryNumber" "InBondNumber" "SCAC" "BillNumber" "AirMasterBOL" "AirHouseBOL" "RequestRelatedBOL" "RequestBOLEntryData" "ClientRef"	Absent.	
	"CJ"	No.	Absent.	"DistrictPortOfEntry" "RequestedFromDate" "RequestedToDate" "RequestedToTime" "EntryNumber1" "EntryNumber2" "EntryNumber3" "EntryNumber4" "EntryNumber5"	Absent.	Census Warning Query submitting.
	"JC"	No.	Absent.	"ReturnDetailRequestIndicator" "EntryFilerCode" "EntryNumber" "CriteriaQueryType" "RequestedFromDateTime" "RequestedToDateTime"	Absent.	Entry Summary Query submitting.
	"FQ"	No.	Absent.	"Type": "F106" "CarriersRefreshNext500": true, false	Absent.	Extract Reference File - request for carriers.
	"FQ"	No.	Absent.	"Type": "F102"	Absent.	Extract Reference File - request for countries.
	"FQ"	No.	Absent.	"Type": "F101"	Absent.	Extract Reference File - request for Customs ports.
	"FQ"	No.	Absent.	"Type": "F111" "FIRMSCode"	Absent.	Extract Reference File -

				"FIRMName" "FIRMSDistrictCode" "FIRMBeginDate"		request for FIRMS.
	"FQ"	No.	Absent.	"Type": "F104"	Absent.	Extract Reference File - request for foreign ports.
	"HA"	No.	Absent.	"FromTariffNumber" "ToTariffNumber" "AsOfDate"	Absent.	Extract Reference File - request for HTS update by range.
	"KI"	No.	"QN"	"ImporterNumber"	Contact.ActionQUE.ImporterBondQuery	Importer/Bond Query. Creates query document and returns Bond and Name / Address info without adding a new contact to database.
ABIQuery, Contact	"KI"	No.	"QNA"	"ImporterNumber"	Contact.ActionQUE.ImporterBondQueryNameAndAddress	Importer/Bond Query. Creates contact using Customs info about the importer.
ABIQuery, StatementDaily, StatementMonthly	"MO"	No.		"TransmissionDateOfStatements" "StatementNumber" "ImporterOfRecordNumber" "ClientBranchIndicator" "ScopeIndicator": true, false "PreliminaryDailyStatementRequest": true, false "FinalDailyStatementRequest": true, false "PreliminaryPeriodicMonthlyStatementRequest": true, false "FinalPeriodicMonthlyStatementRequest": true, false	Absent.	Statement Request Reroute submitting.

3. Samples of put to queue methods with parameters.

3.1 How to create new Contact object by request to Customs.

On this sample [string](#) ImporterNumber is a string importer number in one of the follows formats:

1. **NN-NNNNNNXX** Internal Revenue Service (IRS) Number
2. **NNN-NN-NNNN** Social Security Number
3. **NNNNNN-NNNNN** Previously assigned CBP Number (for updates only)

In these codes, **N** = number, and **X** = alphanumeric.

If the importer number is in IRS format, the last two-position suffix (XX) cannot be the letters O, I and/or Z

```
public static bool CreateImporterBy5106(string ImporterNumber)
{
    // Initialize Client Context. It must be your connection data here:
    string url = @"https://localhost:8130/brokerservice";
    string username = "admin";
    string database = "sandbox";
    string password = "somepassword";
    SMS.Broker.ClientContext.InitializeContext(url, username + "@" + database, password);

    // Prepare manager
    SMS.Broker.ServiceContracts.ISmsABIManager mngrSmsABI =
    SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.ISmsABIManager>();

    // Create collection of parameters:
    List<SMS.Broker.DataContracts.Parameter> parameters = new List<SMS.Broker.DataContracts.Parameter>();
    // Add parameters to collection.
    parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "ImporterNumber", Value = ImporterNumber });
    parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "Action", Value = "QNA" });

    bool Added = false;

    try
    {
        mngrSmsABI.PutToQueue("KI", parameters);
        Added = true;
    }
    catch (FaultException<EntryValidationFault> ex)
    {
        Added = false;
    }

    return Added;
}
```

Detailed description of [Contact](#) class you can see in *06. Contact classes and interfaces. Part 1.pdf*.

3.2 HTS update by tariff range.

This method puts to queue HTS Query to renew tariffs in some tariff number range. Since it creates ABI query document and doesn't use previously saved document the "SourceLink" and "Action" parameters are not added.

```
public static bool TariffRangeUpdate(string StartTariff, string EndTariff, DateTime asOfDate)
{
    // Initialize Client Context. It must be your connection data here:
    string url = @"https://localhost:8130/brokerservice";
    string username = "admin";
    string database = "sandbox";
    string password = "somepassword";
    SMS.Broker.ClientContext.InitializeContext(url, username + "@" + database, password);

    // Prepare manager
    SMS.Broker.ServiceContracts.ISmsABIManager mngrSmsABI =
    SMS.Broker.ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.ISmsABIManager>();

    // Create collection of parameters:
    List<SMS.Broker.DataContracts.Parameter> parameters = new List<SMS.Broker.DataContracts.Parameter>();
    // Add parameters to collection.
    parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "FromTariffNumber", Value = StartTariff });
    parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "ToTariffNumber", Value = EndTariff });
    parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "AsOfDate", Value = asOfDate });

    bool Added = false;

    try
    {
        mngrSmsABI.PutToQueue("HA", parameters);
    }
```

```

        Added = true;
    }
    catch (FaultException<EntryValidationFault> ex)
    {
        Added = false;
    }

    return Added;
}

```

Detailed description of [HarmonizedTariff](#) class you can see in *09. Extract Reference File And Currency Rate classes, 8. Harmonized Tariff*.

3.3 Importer Name / Bond query without adding an importer contact to database.

This method puts to queue Importer/Bond Query but the info is returned as a formatted text only and not added to database as a new contact.

```

// Initialize Client Context. It must be your connection data here:
string url = @"https://localhost:8130/brokerservice";
string username = "admin";
string database = "sandbox";
string password = "somepassword";

ServicesFactory sf1 = new ServicesFactory(url, username + "@" + database, password);

// We need to prepare these managers first.
SMS.Broker.ServiceContracts.ISmsABIManager mngrSmsABI = sf1.GetManager<SMS.Broker.ServiceContracts.ISmsABIManager>();
SMS.Broker.ServiceContracts.Documents.IABITransmissionManager mngrABITransmission =
sf1.GetManager<SMS.Broker.ServiceContracts.Documents.IABITransmissionManager>();
SMS.Broker.ServiceContracts.Common.IEntityEventManager mngrEntityEvent =
sf1.GetManager<SMS.Broker.ServiceContracts.Common.IEntityEventManager>();

// Importer number can be IRS, CBP Assigned Number or Social Security Number (SSN is not unique and its usage can
// lead to error)
string ImporterNumber = "57-123456789"; // This time we use IRS. It is present on Customs Cert environment and hence
// is available in your test database.

// There is no a simple query method, we need to use a general PutToQueue() with parameters:
List<SMS.Broker.DataContracts.Parameter> parameters = new List<SMS.Broker.DataContracts.Parameter>();
parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "ImporterNumber", Value = ImporterNumber });
parameters.Add(new SMS.Broker.DataContracts.Parameter() { Name = "Action", Value = "QN" }); // "QN" means "do not add
// a contact to database".
// It is only a query.

long ABITransmissionId = mngrSmsABI.PutToQueue("KI", parameters); // "KI" is Importer/Bond Query application code in
// CATAIR.

// We need to make a delay here because of a time lapse when PutToQueue method was run and completed but Customs didn't // reply yet.
// It is only as a sample:
string Link = null;
for (int i = 0; i < 5; i++)
{
    System.Threading.Thread.Sleep(5000);
    var answs = mngrABITransmission.GetAnswers(ABITransmissionId);

    Link = answs?.FirstOrDefault()?.SourceLink;

    if (Link != null)
        break;
}

// qr.Text property will contain a formatted text with info about importer including bond info.
var qr = mngrEntityEvent.GetABIEvents(Link, true); // true means "return Text property"

```