

SMS Broker System
Software Integration

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

Annex. In-Bond classes and interfaces.

Table of Contents

(ver. from 8/3/2026)

1. Sample. Context initialization.	1
2. Sample. In-Bond objects reading from database.	1
3. Sample. New In-Bond object creating.	2
4. Sample. How to put In-Bond submission to transmission queue.	4

1. Sample. Context initialization.

Please, look at *11. Annex. Importer Security Filing classes and interfaces.pdf, 1. Sample. Context initialization.*

2. Sample. In-Bond objects reading from database.

Here we read a list of In-Bond documents from database. `Common.InitializeServiceContext()` is a method defined in `Common` class in a previous sample.

```
protected void btGetList_Click(object sender, EventArgs e)
{
    Common.InitializeServiceContext();

    // Server part contains a realization of numerous managers (Service
    // Contracts). SMS.Broker.Services.ServiceTypes.Services is a collection
    // of the full list of managers. Each manager has a name (a part of uri)
    // and an interface (Service Contract).

    var mngr = ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Documents.IInBondManager>();

    EntityPageQuery q = new EntityPageQuery();

    q.Include = "Carrier";           // Read from a database also property Carrier by
    q.PageSize = 10;                 // Carrier_Id (Entity Framework "include")
    q.QueryTotalCount = true;        // Get first 10 records (the big value can
    q.RequestFullList = false;       // decrease an operation performance or cause a
    q.Position = 0;                  // timeout exception)
    // The result will have a total number of records
    // (It requires an additional SQL query and
    // decrease the operation performance)
    // Get all records (It is only for small tables)
    // Position of the first record in query result.
```

```

        // q.Navigation = NavigationType.First;    // Request will return the first
                                                // page list starting with the first
                                                // record
        // q.Navigation = NavigationType.Last;    // Request will return the last
                                                // page list starting with the last
                                                // record - q.PageSize
        // q.Navigation = NavigationType.Next;    // Request will return the next page
                                                // starting with q.Position +
                                                // q.PageSize
        // q.Navigation = NavigationType.Previous; // Request will return the previous
                                                // page starting with q.Position -
                                                // q.PageSize
        q.Navigation = NavigationType.Refresh;    // Request will return a list
                                                // starting with q.Position

// Order by Number descending
q.Order.Add(new OrderItem() { Name = "Number", IsDesc = true });

// Filter (where [Id] < 100L)
q.Criteria = "[Id] < 100L";

// Make a request:
var result = mngr.GetPage(q);

// Total count of records in full query result
// If q.QueryTotalCount = true
// result.TotalCount

// Number of record in Current page
// result.EntityPageCount

// true if it can be got the next non empty page
// result.HasNext

// true if it can be got the previous non empty page
// result.HasPrevious

// List of records in current page
// result.Entities;

// Query that is used to get this page
// result.Query;

GridView1.DataSource = result.Entities;
GridView1.DataBind();
}

```

3. Sample. New In-Bond object creating.

Here we create a new In-Bond object, fill its properties and collections with some values and save it to database. About navigation property and [PossibleValues](#) please see *02. Annex. Getting Started With SMS API, 1. Navigation property.* and *05. Base, common usage classes and interfaces.pdf, 11. Possible Values.* In this sample there are a lot of hardcoded string values are used. It is done specially in order not to complicate the samples. Real work, of course, suggests a work with a user interface or another source of an input data.

```

protected void btAddNewInBond_Click(object sender, EventArgs e)
{
    Common.InitializeServiceContext();

    SMS.Broker.ServiceContracts.Documents.IInBondManager mngrInBond =
        ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Documents.IInBondManager>();

    SMS.Broker.DataContracts.Documents.InBond inBond = mngrInBond.New(null);    // New In-Bond number is assigned
                                                                                // automatically.
    inBond.CreateCollections();

    // It is suggested that really this data is taken from a user interface or some another source.
    // This sample has no a sophisticated user interface. Its purpose is only to show as it works.

    // In-Bond data
    string InBondPortCode = "4901";
    string InBondType = "62";
    string ConveyanceType = "AMS";
    string PortOfDestinationCode = "2704";
}

```

```

string InBondCarrierCode = "ANGTRA";
string PortOfForeignDest = "57035";
string ImportingCarrierCode = "COSU";
string CountryOfImportingCarrierCode = "US";
string MOT = "11";
string ConveyanceName = "TEST240409115434CARR";
string Voyage = "0003E";
string PortOfImportingConvArrivalCode = "4901";
DateTime estDateOfArrival = DateTime.Now.AddDays(10);
string LocationCode = "L064";

// Bill data
string MasterBillIssuerCode = "XXXC";
string MasterBillNumber = "CJ240409R005";

// Take entities from directories by their code:
SMS.Broker.ServiceContracts.Directories.ICustomsPortManager mngrCustomsPort =
    ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Directories.ICustomsPortManager>();
SMS.Broker.DataContracts.Directories.CustomsPort InBondPort = mngrCustomsPort.GetByCode(InBondPortCode, "");
SMS.Broker.DataContracts.Directories.CustomsPort PortOfDestination =
    mngrCustomsPort.GetByCode(PortOfDestinationCode, "");
SMS.Broker.DataContracts.Directories.CustomsPort PortOfImportingConveyanceArrival =
    mngrCustomsPort.GetByCode(PortOfImportingConvArrivalCode, "");

SMS.Broker.ServiceContracts.Directories.IForeignPortManager mngrForeignPort =
    ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Directories.IForeignPortManager>();
SMS.Broker.DataContracts.Directories.ForeignPort PortOfForeignDestination =
    mngrForeignPort.GetByCode(PortOfForeignDest, "");

SMS.Broker.ServiceContracts.Directories.ICarrierManager mngrCarrier =
    ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Directories.ICarrierManager>();
SMS.Broker.DataContracts.Directories.Carrier ImportingCarrier =
    mngrCarrier.GetByCode(ImportingCarrierCode, "");

SMS.Broker.ServiceContracts.Directories.IContactManager mngrContact =
    ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Directories.IContactManager>();
SMS.Broker.DataContracts.Directories.Contact InBondCarrier =
    mngrContact.GetByCode(InBondCarrierCode, ""); // InBondCarrierCode is a code of Contact object, not a
                                                    // Carrier object.

SMS.Broker.ServiceContracts.Directories.IFIRMManager mngrFIRMS =
    ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Directories.IFIRMManager>();
SMS.Broker.DataContracts.Directories.FIRM Location = mngrFIRMS.GetByCode(LocationCode, "");

// Add In-Bond data to In-Bond:
inBond.InBondPort_Id = InBondPort.Id;
inBond.InBondEntryType = InBondType;
inBond.ConveyanceType = ConveyanceType;
inBond.PortOfDestination_Id = PortOfDestination.Id;
inBond.Carrier_Id = InBondCarrier.Id;
inBond.PortOfForeignDestination_Id = PortOfForeignDestination.Id;
inBond.ImportingCarrier_Id = ImportingCarrier.Id;
inBond.ImportingCarrierCountry = CountryOfImportingCarrierCode;
inBond.ImportingTransportationMode = MOT;
inBond.ImportingConveyanceName = ConveyanceName;
inBond.VoyageFlightTripNumber = Voyage;
inBond.PortOfImportingConveyanceArrival_Id = PortOfImportingConveyanceArrival.Id;
inBond.EstimatedDateOfArrival = estDateOfArrival;
inBond.Location_Id = Location.Id;
inBond.Value = 1000m;

// Add Bill data to In-Bond bill:
SMS.Broker.DataContracts.Documents.InBondBill bill = new SMS.Broker.DataContracts.Documents.InBondBill();
SMS.Broker.DataContracts.Directories.Carrier MasterBillIssuer = mngrCarrier.GetByCode(MasterBillIssuerCode, "");

bill.MasterBillIssuer_Id = MasterBillIssuer.Id;
bill.MasterBillNumber = MasterBillNumber;

inBond.Bills.Add(bill);

// save an new In-Bond to database
var inb = mngrInBond.Save(inBond);

// Get Id of new In-Bond:
long Id = inb.Id;
}

```

4. Sample. How to put In-Bond submission to queue.

Here we synchronously put to queue In-Bond document (QP-application) to send it to Customs. In-Bond object's Id is taken from tbInBondId control.

```
protected void btInBondQue6_Click(object sender, EventArgs e)
{
    if (tbInBondId.Text.Trim() == "")
    {
        tbInBondMessage.Text = "In-Bond Id is absent.";
        return;
    }

    long InBondId = Convert.ToInt64(tbInBondId.Text.Trim());

    Common.InitializeServiceContext();

    SMS.Broker.ServiceContracts.Documents.IInBondManager mgrInBond =
        ClientContext.ServicesFactory.GetManager<SMS.Broker.ServiceContracts.Documents.IInBondManager>();

    InBond inb;
    inb = mgrInBond.Get(InBondId, "");

    if (inb == null)
    {
        tbInBondMessage.Text = "In-Bond is absent on database.";
        return;
    }

    long TrId;    // Transmission Id.
    try
    {
        // This method places a request to queue on SMS-server.
        TrId = mgrInBond.PutToQueue(InBondId, InBond.ActionQUE.Add);
    }
    catch (System.ServiceModel.FaultException<SMS.Broker.Faults.EntryValidationFault> ex)
    {
        // Your some catch handling, for example a record to log file.
    }
}
}
```