

SMS Broker System
Software Integration

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

Annex. Accounts Receivable classes and interfaces.

(ver. from 8/3/2026)

Table of Contents

1. Jobs, Sources and Charges.	1
2. Sample. Customer Invoice Save() and Post() methods.	3
3. Sample. Cash Receipt Save() and Post() methods.	5
4. Sample. Customer invoice objects reading from database.	6
5. Sample. How to get customer invoice from database.	7
6. How to get posted / unposted customer invoices.	7

1. Jobs, Sources and Charges.

Possible scenarios for Jobs & Sources invoicing

Job	Source	Charges	Description
Customs entry			If we make a Customs Entry.
	Customs entry	CE - CUSTOMS ENTRY CEART - CUSTOMS ENTRY ADDITIONAL ARTICLE CEBND - CUSTOMS ENTRY BOND CECNT - CUSTOMS ENTRY ADDITIONAL CONTAINERS CEDU - CUSTOMS ENTRY DUTY & FEE CEFDA - CUSTOMS ENTRY FOOD & DRUG (FDA) CEINSU - CUSTOMS ENTRY INSURANCE CEINVO - CUSTOMS ENTRY ADDITIONAL INVOICE CEPGA - CUSTOMS ENTRY PGA CESHP - CUSTOMS ENTRY ADDITIONAL SHIPMENT	
	BreakBulk	BBCFS - BREAKBULK CFS CONTAINER UNLOADING BBCH - BREAKBULK CHASSIS FEE FC - FREIGHT CHARGE	For Shipments of this Customs Entry.
	Delivery Order	DELIV - DELIVERY	For Cargoes of Shipments of this Customs Entry.

Break-Bulk	Shipment	DLCHAS - CONTAINER DELIVERY CHASSIS	Shipments in this Customs Entry.
		DLCN - CONTAINER DELIVERY	
		DLCNHQ - CONTAINER-HQ DELIVERY	
		DLCNOW - CONTAINER-OVERWEIGHT DELIVERY	
		DLTCF - PORT CONGESTION FEE	
	ISF	BBCF - BREAKBULK CONSOLIDATION FEE	If we NOT make Customs Entry and make Breakbulk document or in some specific cases for invoicing before Customs Entry will be made, excepting Shipments for coloaders. Not applicable.
		BBDT - BREAKBULK DOCUMENTATION FEE	
		BBDTA - BREAKBULK DOCUMENTATION FEE ADDITIONAL SHIPMENT	
		BBUNLD - BREAKBULK CONTAINER UNLOADING	
		ISF - ISF	
	Customs entry	ISFBND - ISF BOND	For shipments from this Breakbulk.
Delivery Order	Breakbulk	BBCFS - BREAKBULK CFS CONTAINER UNLOADING	For Cargoes for Shipments of this Breakbulk
		BBCH - BREAKBULK CHASSIS FEE	
		FC - FREIGHT CHARGE	
	Delivery Order	DELIV - DELIVERY	For Cargoes of Shipments of this Breakbulk.
		DLCHAS - CONTAINER DELIVERY CHASSIS	
		DLCN - CONTAINER DELIVERY	
		DLCNHQ - CONTAINER-HQ DELIVERY	
		DLCNOW - CONTAINER-OVERWEIGHT DELIVERY	
	Shipment	DLTCF - PORT CONGESTION FEE	If we DON'T make a Customs Entry and DON'T make a Breakbulk. Not applicable.
		BBCF - BREAKBULK CONSOLIDATION FEE	
		BBDT - BREAKBULK DOCUMENTATION FEE	
		BBDTA - BREAKBULK DOCUMENTATION FEE ADDITIONAL SHIPMENT	
		BBUNLD - BREAKBULK CONTAINER UNLOADING	
Delivery Order	ISF	ISF - ISF	For Cargoes of Shipments of this Breakbulk.
		ISFBND - ISF BOND	
	Customs entry		Not applicable.
	Break-Bulk		Not applicable.
	Delivery Order	DELIV - DELIVERY	For Cargoes of Shipments of this Breakbulk.

		DLCHAS - CONTAINER DELIVERY CHASSIS	
		DLCN - CONTAINER DELIVERY	
		DLCNHQ - CONTAINER-HQ DELIVERY	
		DLCNOW - CONTAINER-OVERWEIGHT DELIVERY	
		DLTCF - PORT CONGESTION FEE	
	Shipment		
		BBCF - BREAKBULK CONSOLIDATION FEE	
		BBDT - BREAKBULK DOCUMENTATION FEE	
		BBDTA - BREAKBULK DOCUMENTATION FEE ADDITIONAL SHIPMENT	
		BBUNLD - BREAKBULK CONTAINER UNLOADING	
	ISF	ISF - ISF	
		ISFBND - ISF BOND	
ISF			If we DON'T make a Customs Entry and DON'T make a Breakbulk and DON'T make a Delivery Order.
	Shipment		For shipments with ISF only.
		ISF - ISF	
Shipment			For Breakbulk Shipments for co- loaders or for other shipments (with no other job was assigned).
	Customs entry		Not applicable.
	Break-Bulk		
		BBCFS - BREAKBULK CFS CONTAINER UNLOADING	
		BBCH - BREAKBULK CHASSIS FEE	
		FC - FREIGHT CHARGE	
	Delivery Order		Not applicable.
	ISF		Not applicable.
Invoice			For annual bonds and unusual cases.
	Invoice		
		CEBND - CUSTOMS ENTRY BOND	
		ISFBND - ISF BOND	

2. Sample. Customer Invoice Save() and Post() methods.

```

public static void Test3()
{
    ServicesFactory sf = new ServicesFactory(@"https://localhost:8130/brokerservice", "UserName@sandbox", "YourPassword");
    ICustomerInvoiceManager mngrCustomerInvoice = sf.GetManager<ICustomerInvoiceManager>();
    ICarrierManager mngrCarrier = sf.GetManager<SMS.Broker.ServiceContracts.Directories.ICarrierManager>();
    IContactManager mngrContact = sf.GetManager<SMS.Broker.ServiceContracts.Directories.IContactManager>();
    IChargeManager mngrCharge = sf.GetManager<SMS.Broker.ServiceContracts.Directories.IChargeManager>();

    #region prepare data to make a new customer invoice.

    // 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.

    string Description = "";

    //Some user data

```

```

string ClientRef = "Some ref.";

// Contact code (for Customer and BillTo):
string CustomerCode = "ABCIMP";

//SCAC code
string CarrierCode = "MAEU";

string CountryOfOriginCode = "CN";

#endregion

CustomerInvoice newCi = mngrCustomerInvoice.New(null);

// Get the next sequence number:
var ciNumber = newCi.Number;

newCi.ClientRef = ClientRef;

// Get Carrier from a database by Carrier Code
SMS.Broker.DataContracts.Directories.Carrier Carrier = mngrCarrier.GetByCode(CarrierCode, "");

// Important item - Carrier is a navigation property and we need to assign Carrier.Id to newCi.Carrier_Id.
// newCi.Carrier_Id has a higher priority against newCi.Carrier and exactly its value will be saved to database
// on save operation. Other words newCi.Carrier = Carrier is incorrect for save operation.
if (Carrier != null)
    newCi.Carrier_Id = Carrier.Id;

// Get Customer from a database by Customer Code
SMS.Broker.DataContracts.Directories.Contact Customer = mngrContact.GetByCode(CustomerCode, "");

// Customer and BillTo are navigation properties also.
newCi.Customer_Id = Customer.Id;
newCi.BillTo_Id = Customer.Id;

newCi.Description = Description;
newCi.CountryOfOrigin = CountryOfOriginCode;

#region Add Items

SMS.Broker.DataContracts.Documents.CustomerInvoiceItem item1 = new SMS.Broker.DataContracts.Documents.CustomerInvoiceItem();
SMS.Broker.DataContracts.Documents.CustomerInvoiceItem item2 = new SMS.Broker.DataContracts.Documents.CustomerInvoiceItem();
SMS.Broker.DataContracts.Documents.CustomerInvoiceItem item3 = new SMS.Broker.DataContracts.Documents.CustomerInvoiceItem();
SMS.Broker.DataContracts.Documents.CustomerInvoiceItem item4 = new SMS.Broker.DataContracts.Documents.CustomerInvoiceItem();

item1.Line = 1;
item2.Line = 2;
item3.Line = 3;
item4.Line = 4;

Charge charge1 = mngrCharge.GetByCode("CEDU", "");
Charge charge2 = mngrCharge.GetByCode("CC", "");
Charge charge3 = mngrCharge.GetByCode("CEFDA", "");
Charge charge4 = mngrCharge.GetByCode("CEFDAA", "");

item1.Charge_Id = charge1.Id;
item2.Charge_Id = charge2.Id;
item3.Charge_Id = charge3.Id;
item4.Charge_Id = charge4.Id;

item1.Amount = 50.45m;
item2.Amount = 70.00m;
item3.Amount = 80.45m;
item4.Amount = 100.45m;

newCi.Items.Add(item1);
newCi.Items.Add(item2);
newCi.Items.Add(item3);
newCi.Items.Add(item4);

#endregion Add Items

// save a new customer invoice to database
newCi = mngrCustomerInvoice.Save(newCi);

//=====
// In this sample Post operation works just after Save but in a real application
// it is a reason to run Post() method later by another button. Post() method
// has more detailed, "demanding" validation.

// Take an Id of a just created customer invoice:
long NewCild = newCi.Id;

try
{
    mngrCustomerInvoice.Post(NewCild);
}

```

```

    }
    catch (System.ServiceModel.FaultException<SMS.Broker.Faults.ActionFault> ex)
    {
        // Your some catch handling, for example a record to log file.
        string errors = ex.Detail.ToString();
    }
}

```

3. Sample. Cash Receipt Save() and Post() methods.

```

public static void Test4()
{
    ServicesFactory sf = new ServicesFactory(@"https://localhost:8130/brokerservice", "UserName@sandbox", "YourPassword");
    ICashReceiptManager mngrCashReceipt = sf.GetManager<ICashReceiptManager>();
    IContactBankAccountManager mngrBankAccount = sf.GetManager<SMS.Broker.ServiceContracts.Directories.IContactBankAccountManager>();
    IContactManager mngrContact = sf.GetManager<SMS.Broker.ServiceContracts.Directories.IContactManager>();
    ICustomerInvoiceManager mngrCustomerInvoice = sf.GetManager<ICustomerInvoiceManager>();

    #region prepare data to make a new cash receipt

    // 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.

    //Some user data
    string ClientRef = "My ref.";

    // Contact code (for Payer):
    string PayerCode = "ABCIMP";

    //Check number. (Do not mix with Number property)
    //Check number is a number of incoming customers check document.
    string CheckNumber = "D1234556";

    string CheckTypeCode = "1"; // It is "Cash Receipt" type. Others are "2" - "Credit Card" and "3" - "Offsetting".

    long customerInvoiceId = 10100L;
    #endregion

    CashReceipt newCr = mngrCashReceipt.New(null);

    // Get the next sequence number:
    var ciNumber = newCr.Number;

    newCr.ClientRef = ClientRef;

    SystemConfiguration systemConfiguration = sf.GetManager<ISystemConfigurationManager>().GetActive();

    // Get owner's Bank account from a database. It is simplified here in a sample.
    // Really it should be a selection from the owner's bank accounts list.
    if (systemConfiguration.Contact != null)
    {
        newCr.BankAccount_Id = systemConfiguration.Contact.BankAccounts[0]?.Id;
    }

    newCr.CheckNumber = CheckNumber;
    newCr.PaymentType = CheckTypeCode;
    newCr.CheckAmount = 200.00m; // It must be equal or more then a sum of detail Allocated Amounts.

    // Get Payer from a database by Payer Code
    SMS.Broker.DataContracts.Directories.Contact Payer = mngrContact.GetByCode(PayerCode, "");

    // Payer is a navigation property also.
    newCr.Payer_Id = Payer.Id;

    #region Add Customer Invoices

    // Cash Receipt detail consists of CashReceiptCustomerInvoice class objects.
    // CashReceiptCustomerInvoice is some mid-layer class object that relates CashReceipt and CustomerInvoice object:
    SMS.Broker.DataContracts.Documents.CashReceiptCustomerInvoice cc1 = new
SMS.Broker.DataContracts.Documents.CashReceiptCustomerInvoice();

    cc1.Line = 1;

    // Get customer invoice by its Id:
    CustomerInvoice ci1 = mngrCustomerInvoice.Get(customerInvoiceId, "");
    cc1.CustomerInvoice_Id = ci1.Id;

```

```

cc1.AllocatedAmount = 200;

newCr.Invoices.Add(cc1);

#endregion Add Customer Invoices

// Save a new customer invoice to database:
newCr = mngrCashReceipt.Save(newCr);

//=====
// In this sample Post operation works just after Save but in a real application
// it is a reason to run Post() method later by another button. Post() method
// has more detailed, "demanding" validation.

// Take an Id of a just created cash receipt:
long NewCrId = newCr.Id;

try
{
    mngrCashReceipt.Post(NewCrId);
}
catch (System.ServiceModel.FaultException<SMS.Broker.Faults.ActionFault> ex)
{
    // Your some catch handling, for example a record to log file.
    string errors = ex.Detail.ToString();
}
}

```

4. Sample. Customer invoice objects reading from database.

```

public static void Test2()
{
    ServicesFactory sf = new ServicesFactory(@"https://localhost:8130/brokerservice", "UserName@sandbox", "YourPassword");

    // 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 mngrCustomerInvoice = sf.GetManager<ICustomerInvoiceManager>();

    EntityPageQuery q = new EntityPageQuery();
    q.Option = "";

    q.Include = "Items.Charge,LastStatuses"; // Read from a database also Items collection with charges.
    q.PageSize = 10; // Get first 10 records (the big value can

    // decrease an operation performance or cause a
    // timeout exception)
    q.QueryTotalCount = true; // The result will have a total number of records

    // (It requires an additional SQL query and
    // decrease the operation performance)
    q.RequestFullList = false; // Get all records (It is only for small tables)
    q.Position = 0; // 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 = mngrCustomerInvoice.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 the next non empty page can be got
// result.HasNext

// true if the previous non empty page can be got
// 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();

}

```

5. Sample. How to get customer invoice from database.

```

public static void Test2()
{
    ServicesFactory sf = new ServicesFactory(@"https://localhost:8130/brokerservice", "UserName@sandbox", "YourPassword");
    var mngrCustomerInvoice = sf.GetManager<ICustomerInvoiceManager>();

    var ci = mngrCustomerInvoice.Get(25L, "BillTo,Customer,Items.Charge");
}

```

6. How to get posted / unposted customer invoices.

```

public static void Test3()
{
    //... Some your code before

    long maxId = 15000; // this value is only a sample, you need to take it, saved after the previous session on your side.
    List<CustomerInvoice> cis = GetPostedAndUnpostedInvoices(ref maxId);

    // Changed maxId should be saved for the next session.

    //... Some your code after
}

public static List<CustomerInvoice> GetPostedAndUnpostedInvoices(ref long maxId)
{
    List<CustomerInvoice> cis = null;

    ServicesFactory sf1 = new ServicesFactory(@"https://smstest.logisticaldatasolutions.com:8130/brokerservice", "user1@database1", "passw1");
}

```

```

SMS.Broker.ServiceContracts.Common.IEntityEventManager mngrEvent
    = sf1.GetManager<SMS.Broker.ServiceContracts.Common.IEntityEventManager>();

SMS.Broker.DataContracts.EntityPageQuery q = new SMS.Broker.DataContracts.EntityPageQuery();
q.PageSize = 1000;
// Maximum number of returned records.
// It is suggested that the number of new (or changed) invoices
// on each new session doesn't exceed 1000.
// If it can exceed then encrease PageSize.
q.Position = 0;
// Position of the first record analyzed by query.
q.Navigation = SMS.Broker.DataContracts.NavigationType.Refresh; // Request will return a list starting with q.Position

// You need to select events:
// 1. Only for customer invoices: [SourceLink] LIKE 'CustomerInvoice'
// 2. Only posted/unposted: [Message] = 'Document posted -> Yes' OR [Message] = 'Document posted -> No
// 3. Only posted or unposted after the previous session (events which Id-s those are bigger then previous max Id saved on the previous session).
q.Criteria = "[SourceLink] LIKE 'CustomerInvoice' AND [Message] LIKE 'Document posted ->' AND [Id] > " + maxId.ToString();

// Make a request:
List<EntityEvent> entityEvents = mngrEvent.GetPage(q).Entities;

if (entityEvents == null)
{
    return cis; // return null (there are no new posted/unposted invoices), maxId is not changed.
}

maxId = entityEvents.Max(e => e.Id); // You need to save it for the next session.

string[] sourceLinks = entityEvents.Select(e => e.SourceLink) // Invoice can be posted and unposted many times.
    .Distinct<string>().ToArray(); // We need only one "mention" of posted/unposted invoice.

// Get invoices by their links. They are only invoices posted/unposted since the previous session those could be potentially added, voided, modified and
// simply posted/unposted.
SMS.Broker.ServiceContracts.Documents.ICustomerInvoiceManager mngrCustomerInvoice
    = sf1.GetManager<SMS.Broker.ServiceContracts.Documents.ICustomerInvoiceManager>();

cis = new List<CustomerInvoice>();
foreach (string s in sourceLinks)
{
    long Id = new SMS.Broker.DataContracts.EntityLink(s).Id; // Extract Id from the source link.
    CustomerInvoice ci = mngrCustomerInvoice.Get(Id, "Customer,BillTo,Carrier,Items.Charge,DebtTurns,LastStatuses");
    cis.Add(ci);
}

// cis list contains the result;
return cis;
}

```