```
// IF YOU FOUND THIS CODE USEFUL, PLEASE FOLLOW THIS LINK TO PROVIDE A SMALL DONATION (PayPal
button in the article)!
//
http://www.tedkrapf.com/RantsRaveTips/tabid/278/EntryId/168/Transaction-Express-TransFirst-API-AC
H-Payment-Integration.aspx
// THANKS!  TKConsulting!

// This program is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// GNU General Public License available at <http://www.gnu.org/licenses/>.


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MgvBal.TransactionExpressAPI; //TE WSDL
using System.ServiceModel;

namespace MgvBal.Payments
{
    public class TransactionExpress
    {
        //'Certification' (TEST) Values
        //protected static string _MercGatewayID = "replaceme";
        //protected static string _MercRegKey = "replaceme";

        //Production Values
        protected static string _MercGatewayID = "replaceme";
        protected static string _MercRegKey = "replaceme";


        public class Response
        {
            public string ResponseCode { get; set; }
            public string TransactionNo { get; set; }
            public string Message { get; set; }
            public bool Successful { get; set; }
        }


        public static Response AchTransaction(string AcctNum, string RoutingNum, string
        customerFullName, string phoneNum, decimal Amount)
        {
            try
            {
```

```
MerchantWebServicesPortTypeClient client = new MerchantWebServicesPortTypeClient
();

Merc m = new Merc();
m.id = _MercGatewayID;  //gateway id
m.regKey = _MercRegKey;
m.inType = 1; // "Merchant Web Service;

//m.inTypeSpecified = true;

Contact contact = new Contact();
contact.fullName = customerFullName;

if (phoneNum.Length == 10)
{
    //add dashes when doesn't have dashes
    string ac = phoneNum.Substring(0, 3);
    string ex = phoneNum.Substring(3, 3);
    string nm = phoneNum.Substring(6, 4);
    phoneNum = ac + "-" + ex + "-" + nm;
}
Phone[] phones = new Phone[1];
phones[0] = new Phone();
phones[0].nr = phoneNum;
phones[0].type = 1; //home phone
contact.phone = phones;

AchEcheck ach = new AchEcheck();
ach.acctNr = AcctNum;
ach.bankRtNr = RoutingNum;
ach.seccCode = 2;
ach.seccCodeSpecified = true;
ach.acctType = 0;
ach.acctTypeSpecified = true;

SendTranRequest request = new SendTranRequest();
request.merc = m;
request.achEcheck = ach;
request.contact = contact;
request.tranCode = 11; //ACH Debit
request.reqAmt = "0" + Math.Round((Amount * 100), 0);  //amount is denoted with
a leading zero and in minor denominations (no decimal)
request.lclDtTm = DateTime.Now;

SendTranResponse response = new SendTranResponse();
response = ((MerchantWebServicesPortType)(client)).SendTran(request);

Response result = new Response();
result.TransactionNo = response.tranData.tranNr;

result.ResponseCode = response.rspCode;
if (result.ResponseCode == "00")
    result.Successful = true;
else
```

```
            result.Successful = false;
        if (response.achResponse != null && response.achResponse.Message != null)
        {
            result.Message = response.achResponse.Message;
        }
        else
        {
            result.Message = AchTransactionResponseCodes.Where(c => c.Key == response.
            rspCode).Select(c => c.Value).First();
        }
        return result;
    }
    catch (Exception ex)
    {
        Response result = new Response();
        result.TransactionNo = "";
        result.Message = "Error Occurred, check values and try again: " + ex.ToString();
        result.ResponseCode = "Error";
        result.Successful = false;
        return result;
    }
}


public static Response ArbSubscription(string AcctNum, string RoutingNum, string
customerFullName, string address1, string address2, string city, string state,
                                       string zip, int dealID, string phoneNum, decimal
                                        Amount, DateTime StartDate)
{
    try
    {
        MerchantWebServicesPortTypeClient client = new MerchantWebServicesPortTypeClient
        ();
        Merc m = new Merc();
        m.id = _MercGatewayID;  //gateway id
        m.regKey = _MercRegKey;
        m.inType = 1; // "Merchant Web Service;
        //m.inTypeSpecified = true;

        #region CREATE CUSTOMER
        Cust cust = new Cust();
        cust.type = 0; //0=add, 1=update
        cust.typeSpecified = true;

        Contact c = new Contact();
        c.typeSpecified = true;
        //c.id = DateTime.Now.ToString("yyMMddhhmmssffff");
        c.fullName = customerFullName;

        if (phoneNum.Length == 10)
        {
            //add dashes when doesn't have dashes
            string ac = phoneNum.Substring(0, 3);
            string ex = phoneNum.Substring(3, 3);
```

```
        string nm = phoneNum.Substring(6, 4);
        phoneNum = ac + "-" + ex + "-" + nm;
    }
    Phone[] phones = new Phone[1];
    phones[0] = new Phone();
    phones[0].nr = phoneNum;
    phones[0].type = 1; //home phone
    c.phone = phones;

    c.addrLn1 = address1;
    if (address2 != "") c.addrLn2 = address2;
    c.city = city;
    c.state = (ContactState)Enum.Parse(typeof(ContactState), state.ToUpper());
    c.stateSpecified = true;
    c.zipCode = zip;

    c.type = 1; //recurring
    c.typeSpecified = true;
    c.stat = 1; //1=active, 0=inactive
    c.statSpecified = true;
    c.note = "Deal ID#: " + dealID.ToString();

    cust.contact = c;


    UpdtRecurrProfRequest requestCust = new UpdtRecurrProfRequest();
    requestCust.merc = m;
    requestCust.cust = cust;
    UpdtRecurrProfResponse responseCust = new UpdtRecurrProfResponse();
    responseCust = ((MerchantWebServicesPortType)(client)).UpdtRecurrProf(
    requestCust);
    #endregion Create Customer

    Console.WriteLine("Response: " + responseCust.custId.ToString());

    #region CREATE WALLET
    m.prodType = 4; //4=ach
    m.prodTypeSpecified = true;

    Contact cWallet = new Contact();
    cWallet.id = responseCust.custId.ToString();

    Cust custWallet = new Cust();
    custWallet.contact = cWallet;

    AchEcheck a = new AchEcheck();
    a.bankRtNr = RoutingNum;
    a.acctNr = AcctNum;
    a.acctType = 0; //checking acct
    a.acctTypeSpecified = true;

    Pmt[] pays = new Pmt[1];
    pays[0] = new Pmt();
    pays[0].type = 0; //0=add;
```

```
pays[0].typeSpecified = true;
pays[0].ordNr = dealID.ToString();
pays[0].desc = dealID.ToString();
pays[0].status = 1; //active
pays[0].statusSpecified = true;
pays[0].ach = a;

UpdtRecurrProfRequest requestWallet = new UpdtRecurrProfRequest();
requestWallet.merc = m;
requestWallet.cust = custWallet;
requestWallet.cust.pmt = pays;
UpdtRecurrProfResponse responseWallet = new UpdtRecurrProfResponse();
responseWallet = ((MerchantWebServicesPortType)(client)).UpdtRecurrProf(
requestWallet);
#endregion Create Wallter


#region CREATE ARB ("Reoccuring")
RecurProf rc = new RecurProf();
rc.type = 0; //add
rc.typeSpecified = true;

Recur r = new Recur();
r.recurProfStat = 1; //'status', 0=inactive, 1=active
r.dbtOrCdt = 0; //debit (take money)
r.amt = "0" + Math.Round((Amount * 100), 0);  //amount is denoted with a
leading zero and in minor denominations (no decimal)
r.startDt = StartDate;
r.blngCyc = 51; //00-daily, 10-weekly, 20-biweekly, 30-every 4 weeks, 40-every
8 wks, 51-specified day/30days, 52-last day of month, 60-every 3months,
70-quarterly, 80-yearly, 90-single payment
r.desc = dealID.ToString();
r.custId = responseCust.custId;
r.custIdSpecified = true;
r.pmtId = responseWallet.pmtId[0];
r.pmtIdSpecified = true;
r.ordNr = dealID.ToString();
r.seccCode = 2; //business to customers
r.seccCodeSpecified = true;

rc.recur = r;

UpdtRecurrProfRequest requestArb = new UpdtRecurrProfRequest();
requestArb.merc = m;
requestArb.recurProf = rc;
UpdtRecurrProfResponse responseArb = new UpdtRecurrProfResponse();
responseArb = ((MerchantWebServicesPortType)(client)).UpdtRecurrProf(requestArb);


#endregion CREATE ARB ("Reoccuring")

Response result = new Response();
result.TransactionNo = responseArb.id.ToString();
result.Message = "";
```

```
            result.ResponseCode = responseArb.rspCode;
            if (responseArb.rspCode == "00")
                result.Successful = true;
            else
                result.Successful = false;

            return result;

        }
        catch (Exception ex)
        {
            Response result = new Response();
            result.TransactionNo = "";
            result.Message = "Error Occurred, check values and try again: " + ex.ToString();
            result.ResponseCode = "Error";
            result.Successful = false;
            return result;
        }
    }


    #region Codes

    public static string GetAchTransactionResponseCodeMessage(string code)
    {
        string msg = "not found";
        try
        {
            msg = AchTransactionResponseCodes.Where(c => c.Key == code).First().Value;
            return msg;
        }
        catch
        {
            return msg;
        }
    }

    public static Dictionary<string, string> AchTransactionResponseCodes = new Dictionary<
    string, string>()
    {
        { "00", "Approved or completed successfully" },
        { "01", "Refer to card issuer" },
        { "02", "Refer to card issuer, special condition" },
        { "03", "Invalid merchant" },
        { "04", "Pick-up card" },
        { "05", "Do not honor" },
        { "06", "Error" },
        { "07", "Pick-up card, special condition" },
        { "08", "Honor with identification (this is a decline response when a card not
        present transaction) If you receive an approval in a card not present environment,
        you will need to void the transaction." },
        { "09", "Request in progress" },
        { "0A", "Reserved for future Realtime use" },
        { "10", "Approved, partial authorization" },
```

```
{ "11", "VIP Approval (this is a decline response for a card not present
transaction)" },
{ "12", "Invalid transaction" },
{ "13", "Invalid amount" },
{ "14", "Invalid card number" },
{ "15", "No such issuer" },
{ "16", "Approved, update track 3" },
{ "17", "Customer cancellation" },
{ "18", "Customer dispute" },
{ "19", "Re-enter transaction" },
{ "20", "Invalid response" },
{ "21", "No action taken" },
{ "22", "Suspected malfunction" },
{ "23", "Unacceptable transaction fee" },
{ "24", "File update not supported" },
{ "25", "Unable to locate record" },
{ "26", "Duplicate record" },
{ "27", "File update field edit error" },
{ "28", "File update file locked" },
{ "29", "File update failed" },
{ "30", "Format error" },
{ "31", "Bank not supported" },
{ "32", "Completed partially" },
{ "33", "Expired card, pick-up" },
{ "34", "Suspected fraud, pick-up" },
{ "35", "Contact acquirer, pick-up" },
{ "36", "Restricted card, pick-up" },
{ "37", "Call acquirer security, pick-up" },
{ "38", "PIN tries exceeded, pick-up" },
{ "39", "No credit account" },
{ "40", "Function not supported" },
{ "41", "Lost card, pick-up" },
{ "42", "No universal account" },
{ "43", "Stolen card, pick-up" },
{ "44", "No investment account" },
{ "45", "Account closed" },
{ "46", "Identification required" },
{ "47", "Identification cross-check required" },
{ "48", "No customer record" },
{ "49", "Reserved for future Realtime use" },
{ "50", "Reserved for future Realtime use" },
{ "51", "Not sufficient funds" },
{ "52", "No checking account" },
{ "53", "No savings account" },
{ "54", "Expired card" },
{ "55", "Incorrect PIN" },
{ "56", "No card record" },
{ "57", "Transaction not permitted to cardholder" },
{ "58", "Transaction not permitted on terminal" },
{ "59", "Suspected fraud" },
{ "60", "Contact acquirer" },
{ "61", "Exceeds withdrawal limit" },
{ "62", "Restricted card" },
{ "63", "Security violation" },
```

```
{ "64", "Original amount incorrect" },
{ "65", "Exceeds withdrawal frequency" },
{ "66", "Call acquirer security" },
{ "67", "Hard capture" },
{ "68", "Response received too late" },
{ "69", "Advice received too late" },
{ "70", "Reserved for future use" },
{ "71", "Reserved for future Realtime use" },
{ "72", "Reserved for future Realtime use" },
{ "73", "Reserved for future Realtime use" },
{ "74", "Reserved for future Realtime use" },
{ "75", "PIN tries exceeded" },
{ "76", "Reversal: Unable to locate previous message (no match on Retrieval
Reference Number)/ Reserved for future Realtime use" },
{ "77", "Previous message located for a repeat or reversal, but repeat or reversal
data is inconsistent with original message/ Intervene, bank approval required" },
{ "78", "Invalid/non-existent account – Decline (MasterCard specific)/ Intervene,
bank approval required for partial amount" },
{ "79", "Already reversed (by Switch)/ Reserved for client-specific use (declined)"
},
{ "80", "No financial Impact (Reserved for declined debit)/ Reserved for
client-specific use (declined)" },
{ "81", "PIN cryptographic error found by the Visa security module during PIN
decryption/ Reserved for client-specific use (declined)" },
{ "82", "Incorrect CVV/ Reserved for client-specific use (declined)" },
{ "83", "Unable to verify PIN/ Reserved for client-specific use (declined)" },
{ "84", "Invalid Authorization Life Cycle – Decline (MasterCard) or Duplicate
Transaction Detected (Visa)/ Reserved for client-specific use (declined)" },
{ "85", "No reason to decline a request for Account Number Verification or Address
Verification/ Reserved for client-specific use (declined)" },
{ "86", "Cannot verify PIN/ Reserved for client-specific use (declined)" },
{ "87", "Reserved for client-specific use (declined)" },
{ "88", "Reserved for client-specific use (declined)" },
{ "89", "Reserved for client-specific use (declined)" },
{ "90", "Cut-off in progress" },
{ "91", "Issuer or switch inoperative" },
{ "92", "Routing error" },
{ "93", "Violation of law" },
{ "94", "Duplicate Transmission (Integrated Debit and MasterCard)" },
{ "95", "Reconcile error" },
{ "96", "System malfunction" },
{ "97", "Reserved for future Realtime use" },
{ "98", "Exceeds cash limit" },
{ "99", "Reserved for future Realtime use" },
{ "A0", "Reserved for future Realtime use" },
{ "A1", "ATC not incremented" },
{ "A2", "ATC limit exceeded" },
{ "A3", "ATC configuration error" },
{ "A4", "CVR check failure" },
{ "A5", "CVR configuration error" },
{ "A6", "TVR check failure" },
{ "A7", "TVR configuration error" },
{ "B1", "Surcharge amount not permitted on Visa cards or EBT Food Stamps/ Reserved
for future Realtime use" },
```

```
            { "B2", "Surcharge amount not supported by debit network issuer/ Reserved for
            future Realtime use" },
            { "C0", "Unacceptable PIN" },
            { "C1", "PIN Change failed" },
            { "C2", "PIN Unblock failed" },
            { "C3 to D0", "Reserved for future Realtime use" },
            { "D1", "MAC Error" },
            { "D2 to E0", "Reserved for future Realtime use" },
            { "E1", "Prepay error" },
            { "E2 to MZ", "Reserved for future Realtime use" },
            { "N0 to ZZ", "Reserved for client-specific use (declined)" },
            { "N0", "Force STIP/ Reserved for client-specific use (declined)" },
            { "N3", "Cash service not available/ Reserved for client-specific use (declined)" },
            { "N4", "Cash request exceeds Issuer limit/ Reserved for client-specific use
            (declined)" },
            { "N5", "Ineligible for re-submission/ Reserved for client-specific use (declined)"
            },
            { "N7", "Decline for CVV2 failure/ Reserved for client-specific use (declined)" },
            { "N8", "Transaction amount exceeds preauthorized approval amount/ Reserved for
            client-specific use (declined)" },
            { "P0", "Approved; PVID code is missing, invalid, or has expired" },
            { "P1", "Declined; PVID code is missing, invalid, or has expired/ Reserved for
            client-specific use (declined)" },
            { "P2", "Invalid biller Information/ Reserved for client-specific use (declined)/
            Reserved for client-specific use (declined)" },
            { "Q1", "Card Authentication failed/ Reserved for client-specific use (declined)" },
            { "R0", "The transaction was declined or returned, because the cardholder requested
            that payment of a specific recurring or installment payment transaction be stopped/
            Reserved for client-specific use (declined)" },
            { "R1", "The transaction was declined or returned, because the cardholder requested
            that payment of all recurring or installment payment transactions for a specific
            merchant account be stopped/ Reserved for client-specific use (declined)" },
            { "XA", "Forward to Issuer/ Reserved for client-specific use (declined)" },
            { "XD", "Forward to Issuer/ Reserved for client-specific use (declined)" }
        };

        #endregion Codes
    }
}
```