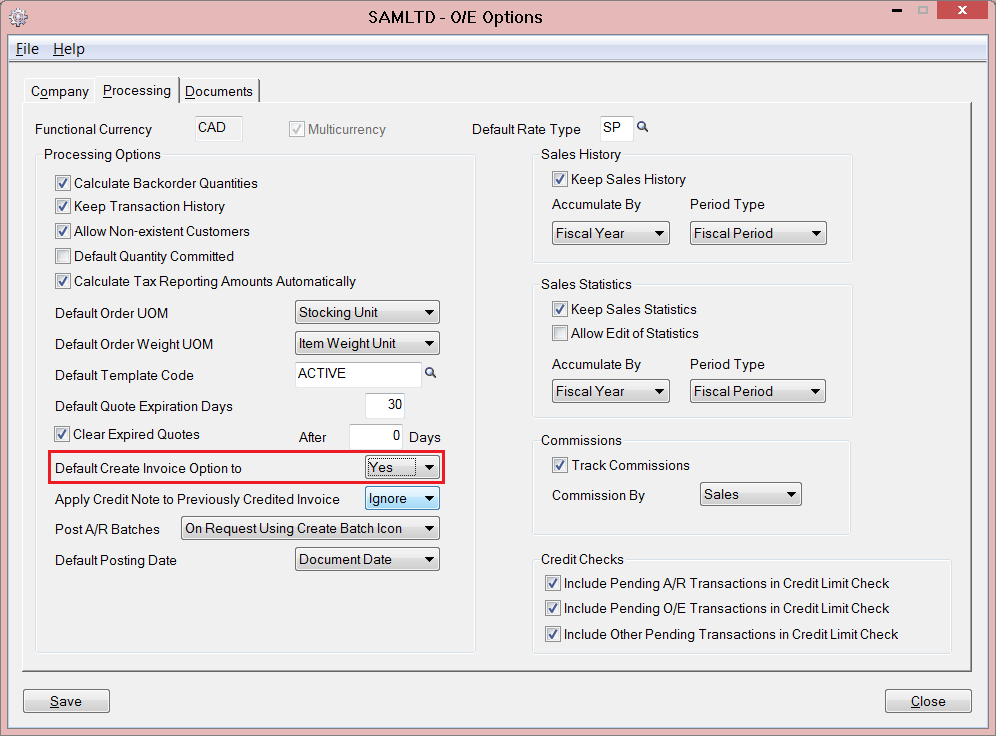
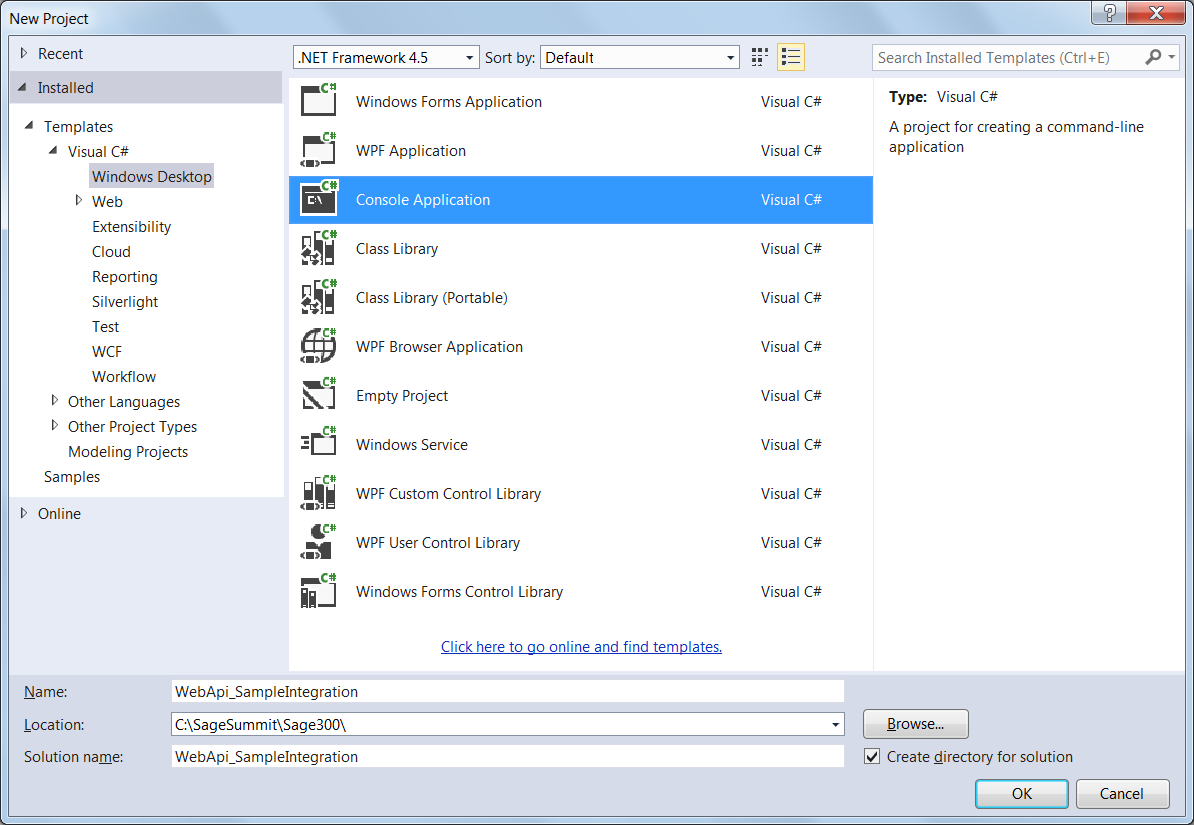
**Sage 300 Web API - Sample Integration**

You will discover the power of Sage 300 Web API by writing a full featured integration with very little code. In this exercise, we will automate the creation of an OE order for a brand new customer and coordinate the workflow of the order's entire life cycle: from generating the associated OE shipment and OE invoice, to generating the AR invoice through IC day end and producing the eventual GL batch.

1. Start **Sage 300**, enter **ADMIN** for both the **User ID** and **Password**, select **Sample Company Limited** and click **OK**.
2. Open **Order Entry** then select **O/E Setup** and open **Options**.
3. In the **Processing** tab, change **Default Create Invoice Option** to **Yes.**

****

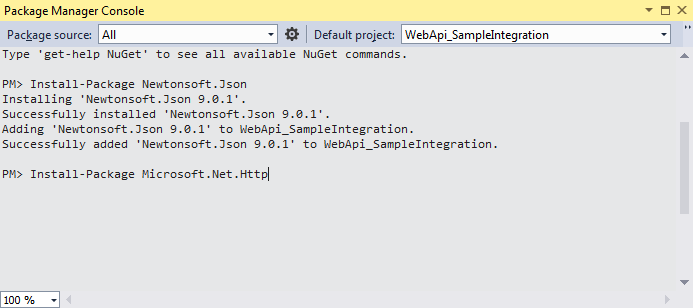
1. Start Visual Studio and from the **File** menu, select **New** and then **Project.**
2. In the **New Project** dialog, select **Installed**, then **Templates** and expand the **Visual C#** node. Next select **Windows Desktop**. In the list of project templates, select **Console Application**. Name the project **WebApi\_SampleIntegration** and click **OK**.



1. From the **View** menu, hover over **Other Windows** and select **Package Manager Console**. In the Package Manager Console window, type the following commands:

**Install-Package Newtonsoft.Json**

**Install-Package Microsoft.Net.Http**



1. Inside **Program.cs** replace the import section at the top of the class with the following:

using System;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using Newtonsoft.Json;

1. Add the following properties at the top of the **Program** class:

private static string NewCustomerNumber { get; set; }

private static string ItemNumber { get; set; }

private static int QuantityToOrder { get; set; }

These will serve as simulated input parameters from an external source.

1. Add a base method for making requests:

public static async Task<object> SendRequest(HttpMethod method, string requestUri, object payload = null)

{

HttpContent content = null;

string responsePayload = "";

// Serialize the payload if one is present

if (payload != null)

{

var payloadString = JsonConvert.SerializeObject(payload);

content = new StringContent(payloadString, Encoding.UTF8, "application/json");

}

// Create the Web API client with the appropriate authentication

using (var httpClientHandler = new HttpClientHandler { Credentials = new NetworkCredential("WEBAPI", "WEBAPI") })

using (var httpClient = new HttpClient(httpClientHandler))

{

Console.WriteLine("\n{0} {1}", method.Method, requestUri);

// Create the Web API request

var request = new HttpRequestMessage(method, requestUri)

{

Content = content

};

// Send the Web API request

try

{

var response = await httpClient.SendAsync(request);

responsePayload = await response.Content.ReadAsStringAsync();

var statusNumber = (int)response.StatusCode;

Console.WriteLine("\n{0} {1}", statusNumber, response.StatusCode);

if (statusNumber < 200 || statusNumber >= 300)

{

Console.WriteLine(responsePayload);

throw new ApplicationException(statusNumber.ToString());

}

}

catch (Exception e)

{

Console.WriteLine("\n{0} Exception caught.", e);

Console.WriteLine("\n\nPlease ensure the service root URI entered is valid.");

Console.WriteLine("\n\nPress any key to end.");

Console.ReadKey();

Environment.Exit(0);

}

}

return string.IsNullOrWhiteSpace(responsePayload) ? null : JsonConvert.DeserializeObject(responsePayload);

}

1. Add a method to create a customer:

public static async Task CreateCustomer(string uri)

{

var customer = new

{

CustomerNumber = NewCustomerNumber,

GroupCode = "WHL",

TaxGroup = "BCTAX",

ShortName = "Old Name",

};

await SendRequest(new HttpMethod("POST"), uri + @"AR/ARCustomers", customer);

}

1. Change the **Main** method by assigning the class properties and calling **CreateCustomer**:

string Sage300WebAPIURI = "http://localhost/Sage300WebApi/v1.0/-/SAMLTD/";

// Set up the input parameters (can be customized to be passed in externally)

Console.Write("Enter a new customer number: ");

NewCustomerNumber = Console.ReadLine();

ItemNumber = @"A1-103/0";

QuantityToOrder = 2;

CreateCustomer(Sage300WebAPIURI).Wait();

Console.WriteLine("\nPress any key to end.");

Console.ReadKey();

1. Run the program by pressing **F5** and entering **ZCUST1** when prompted for the **new customer number**.
2. Add a method to update the customer:

public static async Task UpdateCustomer(string uri)

{

var customer = new

{

ShortName = "New Name"

};

await SendRequest(new HttpMethod("PATCH"), uri + @"AR/ARCustomers('" + NewCustomerNumber + "')", customer);

}

1. Call this new method inside the **Main** method:

static void Main(string[] args)

{

string Sage300WebAPIURI = "http://localhost/Sage300WebApi/v1.0/-/SAMLTD/";

// Set up the input parameters (can be customized to be passed in externally)

Console.Write("Enter a new customer number: ");

NewCustomerNumber = Console.ReadLine();

ItemNumber = @"A1-103/0";

QuantityToOrder = 2;

CreateCustomer(Sage300WebAPIURI).Wait();

UpdateCustomer(Sage300WebAPIURI).Wait();

Console.WriteLine("\nPress any key to end.");

Console.ReadKey();

}

1. Run the program by pressing **F5** and enter **ZCUST2** when prompted for the **new customer number**.
2. Add a method in the **Program** class which creates an OE entry document and interprets the results:

public static async Task CreateOEOrder(string uri)

{

var detail = new

{

Item = ItemNumber,

QuantityOrdered = QuantityToOrder,

QuantityShipped = QuantityToOrder

};

var order = new

{

CustomerNumber = NewCustomerNumber,

InvoiceWillBeProduced = true,

OrderDetails = new[] { detail }

};

dynamic newOrder = await SendRequest(new HttpMethod("POST"), uri + @"OE/OEOrders", order);

Console.WriteLine("Created OE Order Number: {0} with Shipment Number: {1} Invoice Number: {2}", newOrder.OrderNumber, newOrder.LastShipmentNumber, newOrder.LastInvoiceNumber);

}

1. Call this new method inside the **Main** method:

static void Main(string[] args)

{

string Sage300WebAPIURI = "http://localhost/Sage300WebApi/v1.0/-/SAMLTD/";

// Set up the input parameters (can be customized to be passed in externally)

Console.Write("Enter a new customer number: ");

NewCustomerNumber = Console.ReadLine();

ItemNumber = @"A1-103/0";

QuantityToOrder = 2;

CreateCustomer(Sage300WebAPIURI).Wait();

UpdateCustomer(Sage300WebAPIURI).Wait();

CreateOEOrder(Sage300WebAPIURI).Wait();

Console.WriteLine("\nPress any key to end.");

Console.ReadKey();

}

1. Run the program by pressing **F5** and entering **ZCUST3** when prompted for the **new customer number**.
2. Add methods to call various processes including IC Day End and Post AR Invoices:

public static async Task InvokeICDayEnd(string uri)

{

var dayendprocess = new

{

ClearHistory = false

};

await SendRequest(new HttpMethod("POST"), uri + @"IC/ICDayEndProcessing('$process')", dayendprocess);

}

public static async Task PostARInvoice(string uri)

{

var postinvoice = new

{

PostAllBatches = "Postallbatches"

};

await SendRequest(new HttpMethod("POST"), uri + @"AR/ARPostInvoices('$process')", postinvoice);

}

public static async Task CreateGLBatch(string uri)

{

var createglbatch = new

{

ProcessInvoiceBatch = "PostInvoicebatches",

InvoiceThroughPostingSequenceNumber = 999999999

};

await SendRequest(new HttpMethod("POST"), uri + @"AR/ARCreateGLBatch('$process')", createglbatch);

}

1. Call these new methods within the **Main** method:

static void Main(string[] args)

{

string Sage300WebAPIURI = "http://localhost/Sage300WebApi/v1.0/-/SAMLTD/";

// Set up the input parameters (can be customized to be passed in externally)

Console.Write("Enter a new customer number: ");

NewCustomerNumber = Console.ReadLine();

ItemNumber = @"A1-103/0";

QuantityToOrder = 2;

CreateCustomer(Sage300WebAPIURI).Wait();

UpdateCustomer(Sage300WebAPIURI).Wait();

CreateOEOrder(Sage300WebAPIURI).Wait();

InvokeICDayEnd(Sage300WebAPIURI).Wait();

PostARInvoice(Sage300WebAPIURI).Wait();

CreateGLBatch(Sage300WebAPIURI).Wait();

Console.WriteLine("\nPress any key to end.");

Console.ReadKey();

}

1. Run the final program by pressing **F5** and entering **ZCUST4** when prompted.