ByDesign API & Replicated Sites

Documentation for Developers

Contents

Overview

ByDesign Online API

The Online API is the core API used when integrating with ByDesign. This API provides the ability to add leads, customers, and representatives; create orders; and update user information. In short, this API allows the core functionality needed to implement a representative's backoffice, shopping cart, or signup process. Commonly used functions include *CreateCustomer*, *CreateOrder*, and *CreateRep*.

ByDesign Extranet API

The Extranet API is the main API used for reporting. This API has numerous methods that allow for returning current volume amounts, current genealogy placements, and current commission amounts. This API has more methods that can be used to expand a representative's backoffice. Commonly used functions include *GetOrders*, *GetGenealogyReport*, and *GetCommissions*.

ByDesign Order API

The Order API allows for administrative order functions such as confirming order totals, modifying an order's status, modifying order notes, updating shipping information, and printing shipping pack slips. This API is normally used to provide an administrative interface or for integration into shipping warehouses.

ByDesign AutoShip API

The AutoShip API provides the functions necessary to maintain and create AutoShip profiles for both representatives and customers. Commonly used functions include *CreateCustomerProfile, CreateRepProfile,* and *GetProfileItemDetails*.

ByDesign Generic Web Service API

The generic web service API provides the functionality for a 3rd party to call a custom procedure or process without having to write a custom API. This keeps the functionality the same, but the integration costs lower as our developers are easily able to integrate into this API. This API only contains one method which is *GenericWebServiceCall*.

ByDesign Online Leads API

The Online Leads API allows for the entry and viewing of leads. This can be used to track people who are interested in joining the company. The most common methods used are *GetLeads*, *AddLead*, and *GetSourceTypes*.

ByDesign Promotions API

The Promotions API allows for the awarding and tracking of promotions related to an order being processed. The most common functions used are *CheckPromotions*, *ApplyPromotion*, and *GetPromotionDetailsByID*.

ByDesign Replicated Site API

The Replicated Site API allows one to manage the replicated sites for representatives. The API includes functions that allow one to verify a particular URL, retrieve social network information, and retrieve information for a representative's site. The most common functions used are *GetRepInfo_V2*, *VerifyRepURL*, and *GetRepSocialNetworks*.

ByDesign Restful API

The ByDesign Restful API allows object specific access to a lot of the items used within the Freedom service collection. For instance, if you'd like to retrieve all credit types, rep types, or rank types in a Restful API fashion, this would be the place to go. Also, if you'd like to edit warehouse territories or social networking types in a Restful API fashion, this would be where you go as well. This API also contains RESTful APIs that can be used to create a shopping cart browsing experience. Be aware, not all data types are available via the Restful API yet.

Using the ByDesign API

Credentials

The credentials are defined as the access keys that allow one to use an area of the ByDesign API. Prior to being able to successfully call the ByDesign API, one will need to be provided with API credentials. These credentials will allow one to access the ByDesign API.

Username and Password

The credentials come in the form of a username and password. This is done so that any changes can be attributed to a particular integration. In some cases, multiple credentials are given for different areas of an integration as they may be calling the same API for different reasons, and the different credentials allows for easier tracking.

Access Rights

Simply creating a user within the Freedom BackOffice will not allow that username and password to be used against the ByDesign API. Within the Freedom BackOffice, one can assign access rights to a particular user. There is a special section dedicated to the ByDesign API where you can provide what areas of the ByDesign API a user is allowed access to. In some instances, there are particular access rights to call a particular function with one f the APIs. If you are constantly receiving a 0 when calling an API, this could be related to not having a particular access right within the API. One should communicate what APIs will need to be called with the Freedom BackOffice administrator in order to make sure one's credentials have access to the necessary areas of the ByDesign API.

Recommended Tools

Programming Language

ByDesign recommends using the .NET Framework for any ByDesign API implementation. The reason for this being is that .NET allows for an easy conversion of any WSDL into a proxy class. This means rather than generating the soap xml manually, one can simply call a method just as if it was a local function[1]. Nearly all code examples in this document will be provided in C# 4.0 using the .NET Framework 2.0.

.NET Web Service Studio

WebServiceStudio is a free software tool provided by Adnan Masood on Codeplex[2]. This tool allows one to make test calls to the ByDesign API without writing a single line of code. The tool also automatically generates a C# proxy class for the referenced WSDL as well as provides the SOAP XML for each test call to the API. ByDesign recommends testing call on WebServiceStudio for troubleshooting should one incur any issues.

Restful APIs

For testing Restful APIs, there really is a plethora of API testing tools depending on your preferred language or OS. If you're looking to send request and response information to our support team, then we'd recommend the tools Postman[3] or REST Console[4] as they allow easy copy & pasting of your request and response information for easy debugging & recreating purposes.

[1] While some other language/frameworks can do this as well, ByDesign recommends the .NET Framework based on the simplicity of usage.

[2] Web Service Studio - http://webservicestudio.codeplex.com/

[3] https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop

[4] https://chrome.google.com/webstore/detail/rest-console/cokgbflfommojglbmbpenpphppikmonn

Examples & Programming Tips

Calling & Using SOAP APIs

Below are examples of how to call & use the SOAP APIs (all APIs except the Restful API)

Proxy Class

Most examples within this document use a generated proxy class. This class is generated via the WSDL command line tool for the .NET Framework, and allows for easy object based API calls[5]. One can generate a proxy class in one of three ways as follows: use the WSDL command line tool, import the WSDL into a Visual Studio Web Service reference, or use the generated proxy class from WebServiceStudio[6]. ByDesign strongly recommends using a proxy class as it allows for strongly typed references as well as easier API integrations.

Calling the API

The following code example displays how to make a call to the ByDesign API using a proxy class. For most instances of code examples, only the base code will be given. One can assume all object references are for a generated proxy class.

```
class Program
{
    //instance of the generated Credentials class
    Credentials _ByDesignAPICredentials;

    //instance of the generated API proxy class
    OrderAPISoap _ByDesignOrderAPI;

    public void Main(string[] args)
    {
        //Setup our credentials once
        SetupAPICredentials();

        //Call the ByDesign Order API (GetOrderInfo) for OrderID 12345
        GetOrderResponse GetOrderReturn = _ByDesignOrderAPI.GetOrderInfo(_ByDesignAPICredentials, 12345);

        //If the call was a success
        if (GetOrderReturn.Success == 1)
        {
            //Let's print the date the order was made
            Console.WriteLine("Returned information from order. The order date is {0}", GetOrderReturn.OrderDate);
        }
        else
        {
            //if not a success, let's print why the API message returned zero
            Console.WriteLine("Uh-oh, the order may not exist, API message: {0}", GetOrderReturn.Message);
        }
    }

    void SetupAPICredentials()
    {
        //setup our API credentials
        _ByDesignAPICredentials.Username = "FakeUserName";
        _ByDesignAPICredentials.Password = "FakeUserPassword";
    }
}
```

Debugging the API

After helping numerous clients get up and running with the API, ByDesign has created a list of common problems encountered when writing an API implementation. The following list contains a list of items to check for while writing an API implementation.

[5] WSDL Import Tool - http://msdn.microsoft.com/en-us/library/7h3ystb6.aspx
[6] Importing a Web Service - http://msdn.microsoft.com/en-us/library/d9w023sx.aspx

- When encountered a success of 0 seemingly no matter what, check that your API username still has access. ByDesign has had countless emergency calls related to the API credentials being deactivated or user rights being removed.
- Always check the success property or integer returned. If it is 0, something was wrong with the call. This can define multiple situations, and the situation is usually clearly defined in the message property of the object returned. One should **always** check each possible scenario related to creating customers, representatives, leads, or orders. If a payment returns zero, and one attempts to complete the order, it will fail because the order will have a balance due. These situations should be handled appropriately. Most flowcharts within this document will not document validating the return of payments as it is assumed one will validate the payment's success.
- If not using a proxy, always verify the type of object being returned. A Rep number could always be alphanumeric rather than an integer so remember to always treat it as an alphanumeric object.
- Always check encoding. ByDesign recommends using UTF-16 for encoding. When one ends up going international, this can lead to numerous issues should international encoding not be accounted for in all areas of the implementation.

Calling and Using the Restful API

Reviewing the Object Types

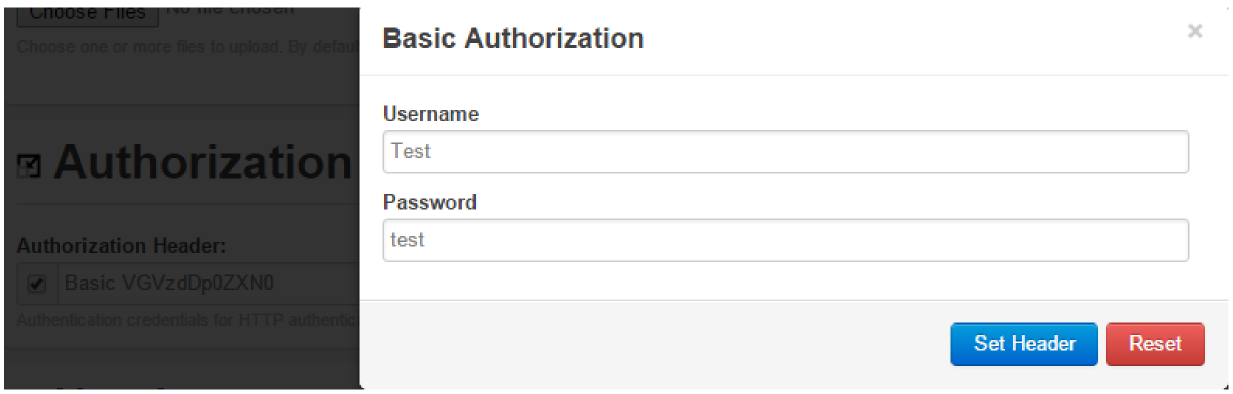
Before you begin, you'll want to find out which object type you'd like to interact with. Each object type has its own documented automatically generated based off of which request types and URLs are available. You'll notice that some object types have numerous request URLs available as some object types belong to numerous areas (such as Admin, Bonus, or Promotions). You can review your documentation at https://webapi.securefreedom.com/{Your-Web-Root}/Help where {Your-Web-Root} is your chosen web folder. If you'd like a sample API to review, please contact your implementation specialist and they can provide you with a sample client (which they will need to generate credentials for).

Generating a Request

Based on the object type you'd like to call, your request will typically look like GET ~/api/{AREA}/{OBJECT_TYPE}. For instance, a request to get all the rank types would look like GET ~/api/Admin/RankType. Of course, we'd still need to provide authentication, but this is the basics of our request.

Authentication

With the same credentials you'd use for any API, you'll want to add a basic authorization header to the request with the same user name & password. Most API testing suites will allow you to simply put in a user name & password and it will automatically generate the header. An example is shown below.



Username
Test
Password
test

If you're unfamiliar with generating the basic authentication header, we'd recommend the following link for guidance: https://en.wikipedia.org/wiki/Basic_access_authentication#Client_side

OAuth

The Restful API also supports OAuth for many client facing APIs such as InventoryShopping and InventoryCategories. This allows your application to let a representative or customer interact directly with the ByDesign Restful API[7]. The following sections will help you understand and use the OAuth authentication within the Restful API.

Username Separate and Application Client ID

Because OAuth by default only assume one user type, ByDesign has implemented a system to identify the user type with a prefix. By default, if no prefix is given, the user type of representative will be assumed. The two prefixes are “-->r_” and “-->c_” which identify representatives and customers respectively.

Also, in order to identify origin and that requests are not being made from an invalid source, each request for a token will need to send an application client id. This application client Id can be requested from ByDesign. You will need to provide which origins the requests will be coming from. A wildcard is allowed within the scope of an URL, but the origin cannot simply be a wildcard token (allowing any origin). For instance, if you are hosting on a site with url https://{CLIENTNAME}.{YOURSITE}.com, you could provide an origin of https://*.{YOURSITE}.com.

Getting an OAuth Token

You want to make sure that whatever authentication/requests that occur are directly from client to the web API. The Client ID you pass will be explicitly tied to a particular origin and requests will only work to and from that origin. In order to get an OAuth token from a particular username and password, you’ll want the client to send a grant request of password to https://webapi.securefreedom.com/{Your-Web-Root}/token. An example request for a representative is below:

```
POST https://webapi.securefreedom.com/{Your-Web-Root}/token
Accept:application/json
Content-Type:application/x-www-form-urlencoded
Form Data:
grant_type=password
&username=-->r_{REPDID}
&password={PASSWORD}
&client_id={APPLICATION_CLIENT_ID}
```

This will return the following response:

```
{
  "access_token":
"Fx9iNaORGprrSEx3VUYzbkS1mVu5OrffKrreLQUYK0vpVUz9Si1Oj83GLuu2jQJE7GV7t9pwRj2s_u7OuCqM1cKOvSLPurhu_
2DEkkN2LbDCTnOS0ilppDuFh7avU1b7GK0fEvR9TQfFqHBDp_Wbm-
7KgNanoLifn9G1ljCwdlLjTp1sgJZLz2yGREln353D2GrI1K_rlK3owTh1Zk2mE4v6tWjK4LYXk3WJj1JCERB190OhVUgrd51IpkbE
ddY5D5gX__TmjW2XOh41G3x2cdex5NDDriqDfg2eM9d_vjOouFDDmOqn_mPrNkiv383VQsb9Dw",
  "token_type": "bearer",
  "expires_in": 172799,
  "refresh_token": "d68b7ffc0726408c8fd36e4ea5e57a1c",
  "as:client_id": "{APPLICATION_CLIENT_ID}",
  "userName": "-->r_{REPDID}",
  "userType": "rep",
  ".issued": "Tue, 26 Jul 2016 12:41:07 GMT",
  ".expires": "Thu, 28 Jul 2016 12:41:07 GMT"
}
```

[7] This is the same API that ByDesign uses in the JS Cart. You can see this interaction by simply watching the requests back and forth in the JS Cart if desired.

You can now use this as a bearer token for a request in order to authenticate the user. Be aware that this token **will** expire. However, you can use the refresh token in order to request a new token once it expires.

Request a new token

After so much time (based on the expiration date), your authentication token will expire for your client, and you will need to get a new one. Rather than forcing your client to login again, you can refresh their authentication token instead. Sometimes, the refresh token request **will** fail (for various authentication reasons), and you will need to have the user login again anyways; however, almost every time the refresh token request will succeed. Here's a sample request:

```
POST https://webapi.bydesign.com/{Your-Web-Root}/token
Accept:application/json
Content-Type:application/x-www-form-urlencoded
Form Data:
grant_type=refresh_token
&refreshtoken=d68b7ffc0726408c8fd36e4ea5e57a1c
&client_id={APPLICATION_CLIENT_ID}
```

This will return the following response:

```
{
  "access_token":
"tuAcTI0yyv_NwV71eYyx7WpODEyQpgJcM0FnCpoudq0y3XzkLXP1Q2xaVSpU0TpbyKORYBQyrnW6xaj1nRXMNpwhJ3TSs-
5p_PwfggiAWh0Zy-
Aa_ZM3r4liX_yk3AvtZzHFvmar5s1xVFh13AxtnnA6DS3QU5w1ZC2RmfY095IUcjuEPJosN7GNt3l3Wa3BAZ2yJnwI92vUmeK
bx6weJU7hSbc23opD0LW8nKK7ZP9R45ErADgvfwKvEHXRZYL-
PDQ0S3TcXH_Ro5qikz0eSBR4g5S0QvhNXorh9ILnBiXPezctsVLs31DSDVD4ePoOibWJ7Q",
  "token_type": "bearer",
  "expires_in": 172799,
  "refresh_token": "d277e3f64336416895782Fa142523a8e",
  "as:client_id": "{APPLICATION_CLIENT_ID}",
  "userName": "-->r_{REPID}",
  "userType": "rep",
  ".issued": "Tue, 26 Jul 2016 12:57:08 GMT",
  ".expires": "Thu, 28 Jul 2016 12:57:08 GMT"
}
```

Specifying Content Type

Depending on what language you are working in, you may want the results in XML or JSON. You'll be happy to know that the Restful API supports both XML & JSON. All you need to do is specify which format you'd like in the "Accept" request header. The same goes for adding or editing current objects, if you want to update in XML or JSON, simply specify the "Content-Type" request header.

Sample Request & Response

Below is a sample request/response

Request:

Request Url: https://webapi.securefreedom.com/{Your-Web-Root}/api/admin/RepType

Request Method: GET

```
Accept: application/json
Authorization: Basic WW91ciBBUEkgVXNlck5hbWU6WW91ciBBUEkgUGFzc3dvcmQ=[8]
```

Response:

```
Status Code: 200
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache
Content-Length: 609
Expires: -1
Body:
[
  {
    "ID":1,
    "Description":"Consultant",
    "Abbreviation":"Cons",
    "OrderType":"Wholesale",
    "AutoShipOrderType":"Wholesale",
    "IsDefaultValue":true,
    "CreatedBy":"SYSTEM",
    "DateCreated":"\/Date(-2177434800000-0500)\/",
    "LastModifiedBy":"ByDesign User (bydesign)",
    "DateLastModified":"\/Date(1327532020983-0500)\/"
  },
  {
    "ID":2,
    "Description":"Special",
    "Abbreviation":"Sp",
    "OrderType":"Special",
    "AutoShipOrderType":"Special",
    "IsDefaultValue":false,
    "CreatedBy":"ByDesign User (bydesign)",
    "DateCreated":"\/Date(1204059297613-0500)\/",
    "LastModifiedBy":"alocaluser",
    "DateLastModified":"\/Date(1209586442190-0400)\/"
  }
]
```

Logging

ByDesign cannot encourage logging enough. We continually see situations where an error within an integration has occurred, but no logging is being made for the calls against the ByDesign API. In any integration situation, one should always be recording the request & response from the 3rd party and ByDesign suggests the same for you. We have had multiple clients call in complaining of something not working, only to find out that they accidently changed their credentials. Logging allows one to track any problems or situations over time making for an easier bug tracking and integration process.

[8] To authentication using OAuth, simple use Authorization: Bearer {AUTHENTICATION_TOKEN}

ByDesign Online API

URL

The ByDesign Online API is available at https://api.securefreedom.com/{CLIENTNAME}/Webservice/OnlineAPI.asmx?wsdl where {CLIENTNAME} is the client's main folder directory.

Main Tasks with Flow Charts

Overview

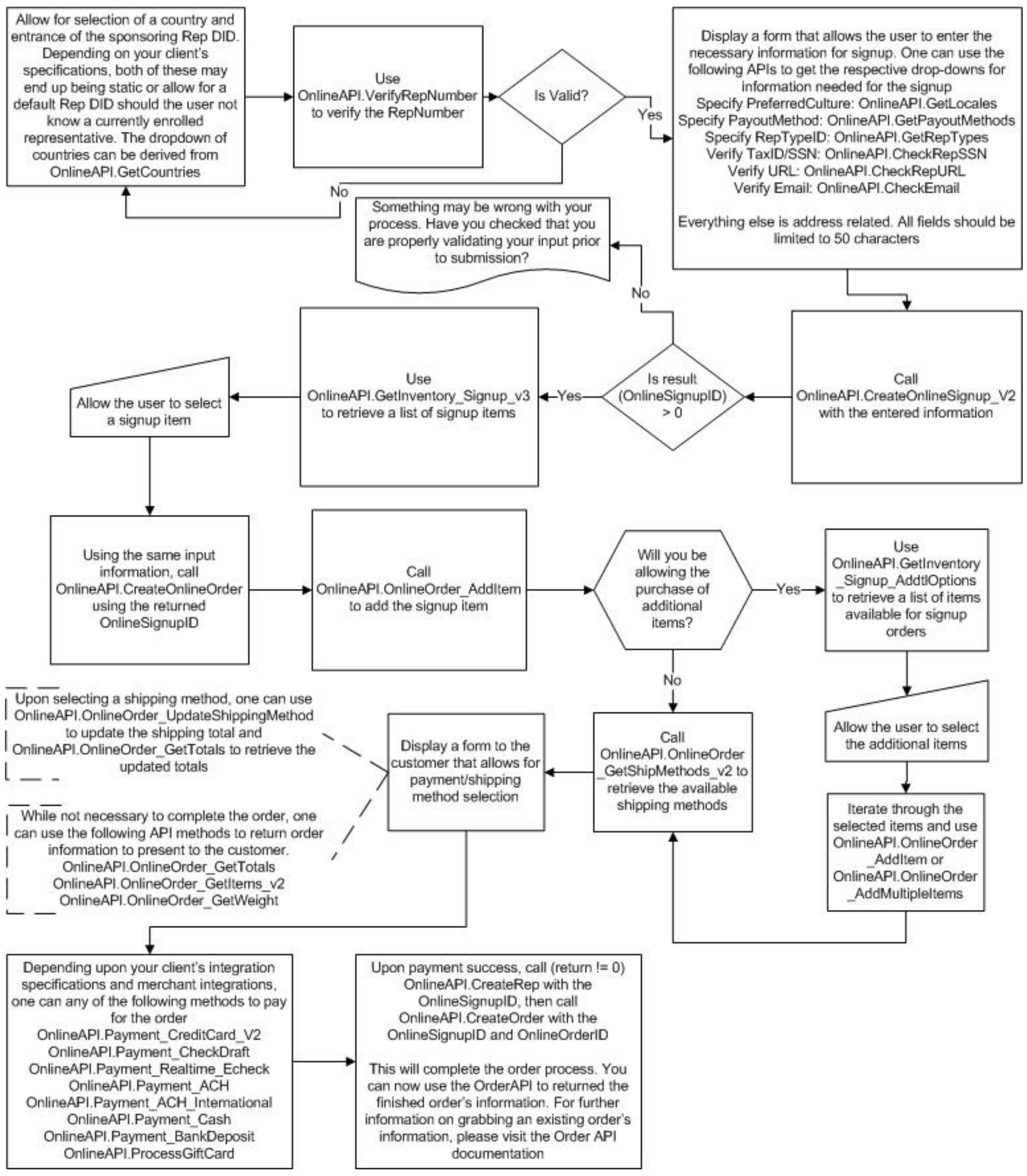
In order to make integration as easy as possible, ByDesign has created a series of flow charts that will help explain the flow of some of the more common tasks done within API integrations. While the flow charts are not meant to be the singular form of integration information, they provide an overview of tasks that will need to be accounted for upon designing the integration that will be completed.

One of the most common problems we see in some rushed integrations is that the proper time was not taken to glimpse all the tasks required for the most basic of signups and orders. The below flowcharts can help one realize all areas of validation and data querying that will be necessary in order to make a robust signup and/or order integration. It is strongly recommended to review the schema for all functions one will need to call for the desired integration. Reviewing the data types and necessary parameters prior to beginning implementation can result in a much smoother integration.

Depending on the size of the integration taking place, it may be best to talk with your account or implementation manager concerning setting up a call with a developer. These calls can usually help plan for all areas of a large integration as well as allow one to explain any further integrations desired, and if that functionality exists in the current API. For instance, there was a situation where a client desired to verify if a representative had purchased a particular item within a particular time period. The client had been going through every single order for a representative, and then going through every item within the order to verify this. Needless to say, this was extremely inefficient and making for an extremely slow verification process. After setting up a call, the developer informed them of a function in the ByDesign Order API (*CheckOrderedItemForRepDIDWithinDate*) that allowed this exact same functionality in one simple call allowing for a much faster verification process.

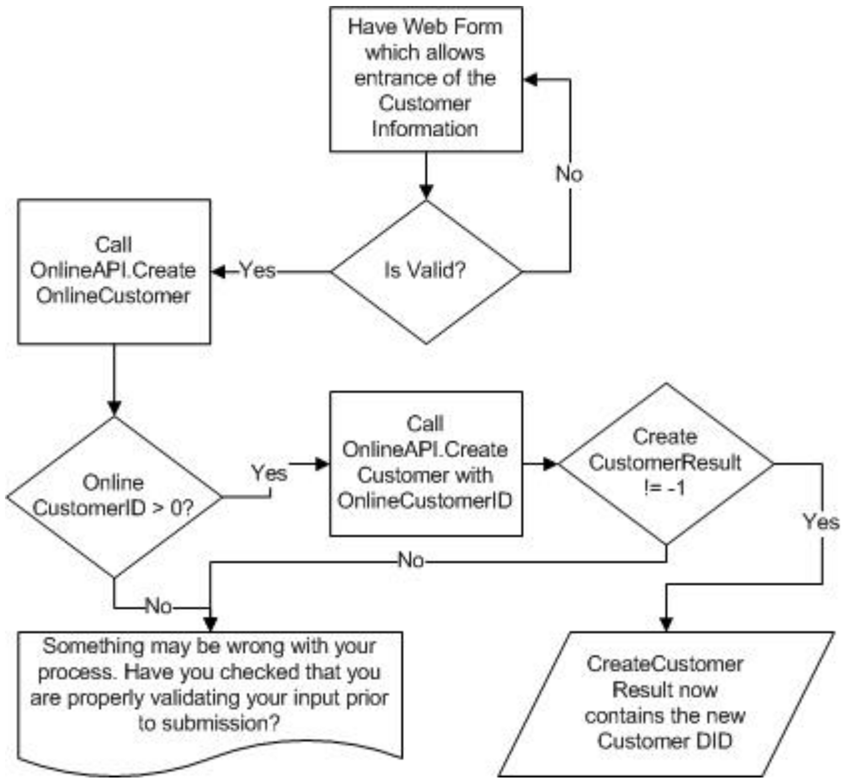
Signing Up a Representative

Below we define the methods and page interactions that should take place for signing up a representative.


Allow for selection of a country and entrance of the sponsoring Rep DID. Depending on your client's specifications, both of these may end up being static or allow for a default Rep DID should the user not know a currently enrolled representative. The dropdown of countries can be derived from OnlineAPI.GetCountries
Use OnlineAPI.VerifyRepNumber to verify the RepNumber
Is Valid?
Yes
No
Display a form that allows the user to enter the necessary information for signup. One can use the following APIs to get the respective drop-downs for information needed for the signup
Specify PreferredCulture: OnlineAPI.GetLocales
Specify PayoutMethod: OnlineAPI.GetPayoutMethods
Specify RepTypeID: OnlineAPI.GetRepTypes
Verify TaxID/SSN: OnlineAPI.CheckRepSSN
Verify URL: OnlineAPI.CheckRepURL
Verify Email: OnlineAPI.CheckEmail
Everything else is address related. All fields should be limited to 50 characters
Something may be wrong with your process. Have you checked that you are properly validating your input prior to submission?
No
Call OnlineAPI.CreateOnlineSignup_V2 with the entered information
Is result (OnlineSignupID) > 0
Yes
Use OnlineAPI.GetInventory_Signup_v3 to retrieve a list of signup items
Allow the user to select a signup item
Using the same input information, call OnlineAPI.CreateOnlineOrder using the returned OnlineSignupID
Call OnlineAPI.OnlineOrder_AddItem to add the signup item
Will you be allowing the purchase of additional items?
Yes
Use OnlineAPI.GetInventory_Signup_AddtlOptions to retrieve a list of items available for signup orders
Allow the user to select the additional items
Iterate through the selected items and use OnlineAPI.OnlineOrder_AddItem or OnlineAPI.OnlineOrder_AddMultipleItems
No
Call OnlineAPI.OnlineOrder_GetShipMethods_v2 to retrieve the available shipping methods
Display a form to the customer that allows for payment/shipping method selection
Upon selecting a shipping method, one can use OnlineAPI.OnlineOrder_UpdateShippingMethod to update the shipping total and OnlineAPI.OnlineOrder_GetTotals to retrieve the updated totals
While not necessary to complete the order, one can use the following API methods to return order information to present to the customer.
OnlineAPI.OnlineOrder_GetTotals
OnlineAPI.OnlineOrder_GetItems_v2
OnlineAPI.OnlineOrder_GetWeight
Depending upon your client's integration specifications and merchant integrations, one can any of the following methods to pay for the order
OnlineAPI.Payment_CreditCard_V2
OnlineAPI.Payment_CheckDraft
OnlineAPI.Payment_Realtime_Echeck
OnlineAPI.Payment_ACH
OnlineAPI.Payment_ACH_International
OnlineAPI.Payment_Cash
OnlineAPI.Payment_BankDeposit
OnlineAPI.ProcessGiftCard
Upon payment success, call (return != 0) OnlineAPI.CreateRep with the OnlineSignupID, then call OnlineAPI.CreateOrder with the OnlineSignupID and OnlineOrderID
This will complete the order process. You can now use the OrderAPI to returned the finished order's information. For further information on grabbing an existing order's information, please visit the Order API documentation

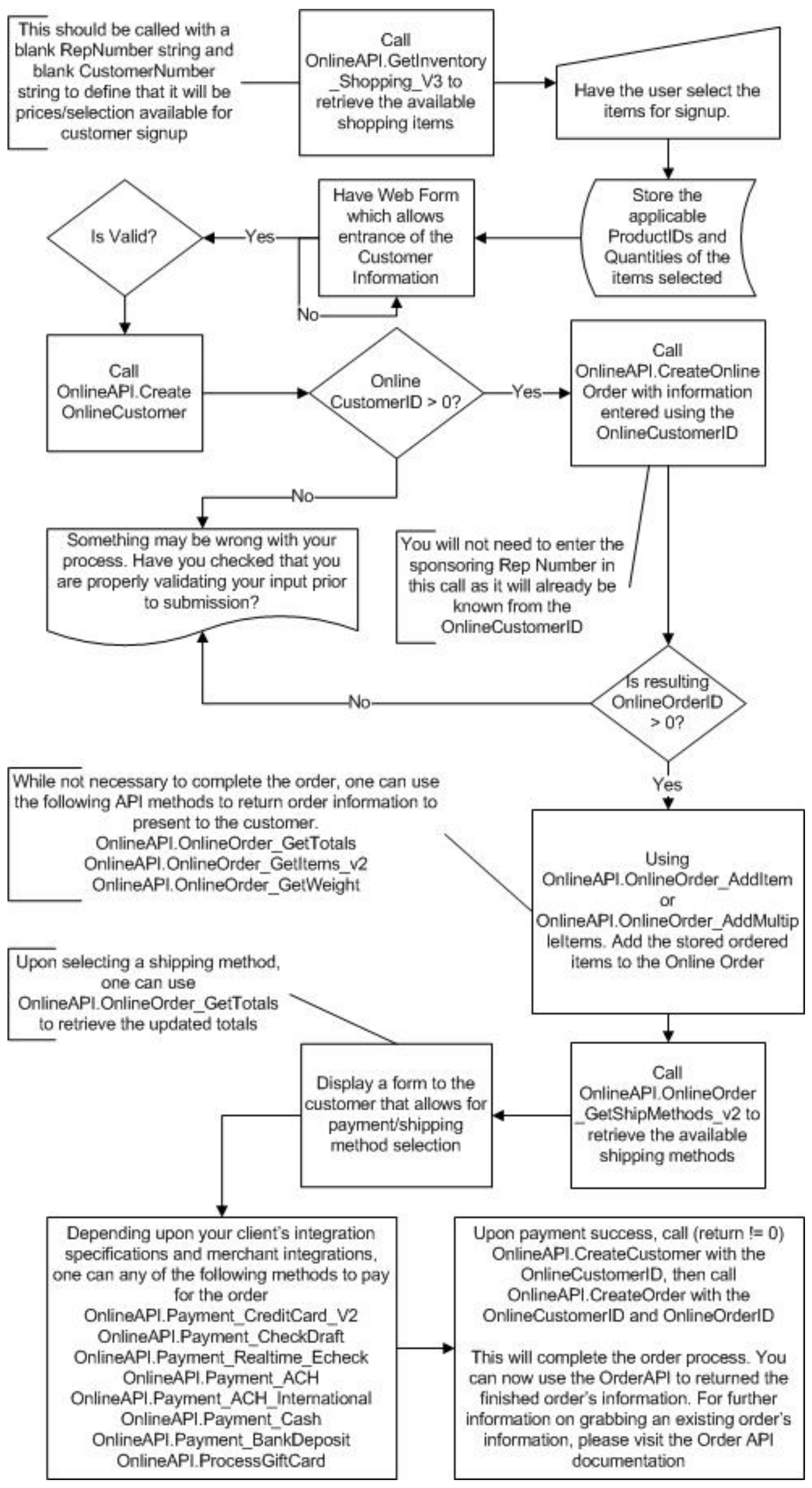
Signing Up a Customer without Purchase

Below we define the methods and page interactions that should take place for signing up a customer when no purchase is necessary


Have Web Form which allows entrance of the Customer Information
No
Is Valid?
Yes
Call OnlineAPI.Create OnlineCustomer
Online CustomerID > 0?
Yes
Call OnlineAPI.Create Customer with OnlineCustomerID
Create CustomerResult != -1
Yes
No
No
Something may be wrong with your process. Have you checked that you are properly validating your input prior to submission?
CreateCustomer Result now contains the new Customer DID

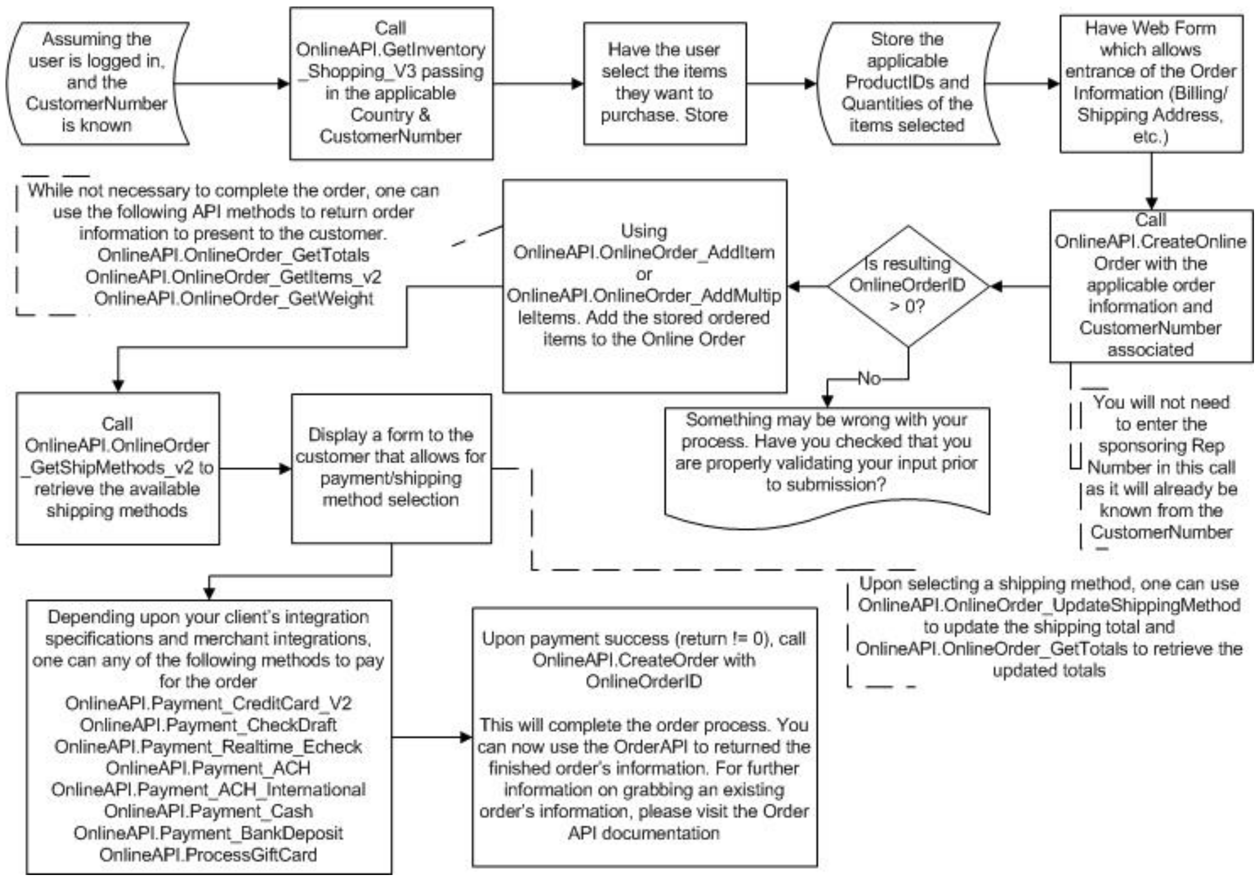
Signing Up a Customer with Signup Items

Below we define how to sign up a customer where the signup is dependent upon the purchase of items.


This should be called with a blank RepNumber string and blank CustomerNumber string to define that it will be prices/selection available for customer signup
Call OnlineAPI.GetInventory_Shopping_V3 to retrieve the available shopping items
Have the user select the items for signup.
Store the applicable ProductIDs and Quantities of the items selected
Have Web Form which allows entrance of the Customer Information
Is Valid?
Yes
No
Call OnlineAPI.CreateOnlineCustomer
Online CustomerID > 0?
Yes
No
Call OnlineAPI.CreateOnlineOrder with information entered using the OnlineCustomerID
Something may be wrong with your process. Have you checked that you are properly validating your input prior to submission?
You will not need to enter the sponsoring Rep Number in this call as it will already be known from the OnlineCustomerID
Is resulting OnlineOrderID > 0?
No
Yes
While not necessary to complete the order, one can use the following API methods to return order information to present to the customer. OnlineAPI.OnlineOrder_GetTotals OnlineAPI.OnlineOrder_GetItems_v2 OnlineAPI.OnlineOrder_GetWeight
Using OnlineAPI.OnlineOrder_AddItem or OnlineAPI.OnlineOrder_AddMultipleItems. Add the stored ordered items to the Online Order
Upon selecting a shipping method, one can use OnlineAPI.OnlineOrder_GetTotals to retrieve the updated totals
Display a form to the customer that allows for payment/shipping method selection
Call OnlineAPI.OnlineOrder_GetShipMethods_v2 to retrieve the available shipping methods
Depending upon your client's integration specifications and merchant integrations, one can any of the following methods to pay for the order OnlineAPI.Payment_CreditCard_V2 OnlineAPI.Payment_CheckDraft OnlineAPI.Payment_Realtime_Echeck OnlineAPI.Payment_ACH OnlineAPI.Payment_ACH_International OnlineAPI.Payment_Cash OnlineAPI.Payment_BankDeposit OnlineAPI.ProcessGiftCard
Upon payment success, call (return != 0) OnlineAPI.CreateCustomer with the OnlineCustomerID, then call OnlineAPI.CreateOrder with the OnlineCustomerID and OnlineOrderID
This will complete the order process. You can now use the OrderAPI to returned the finished order's information. For further information on grabbing an existing order's information, please visit the Order API documentation

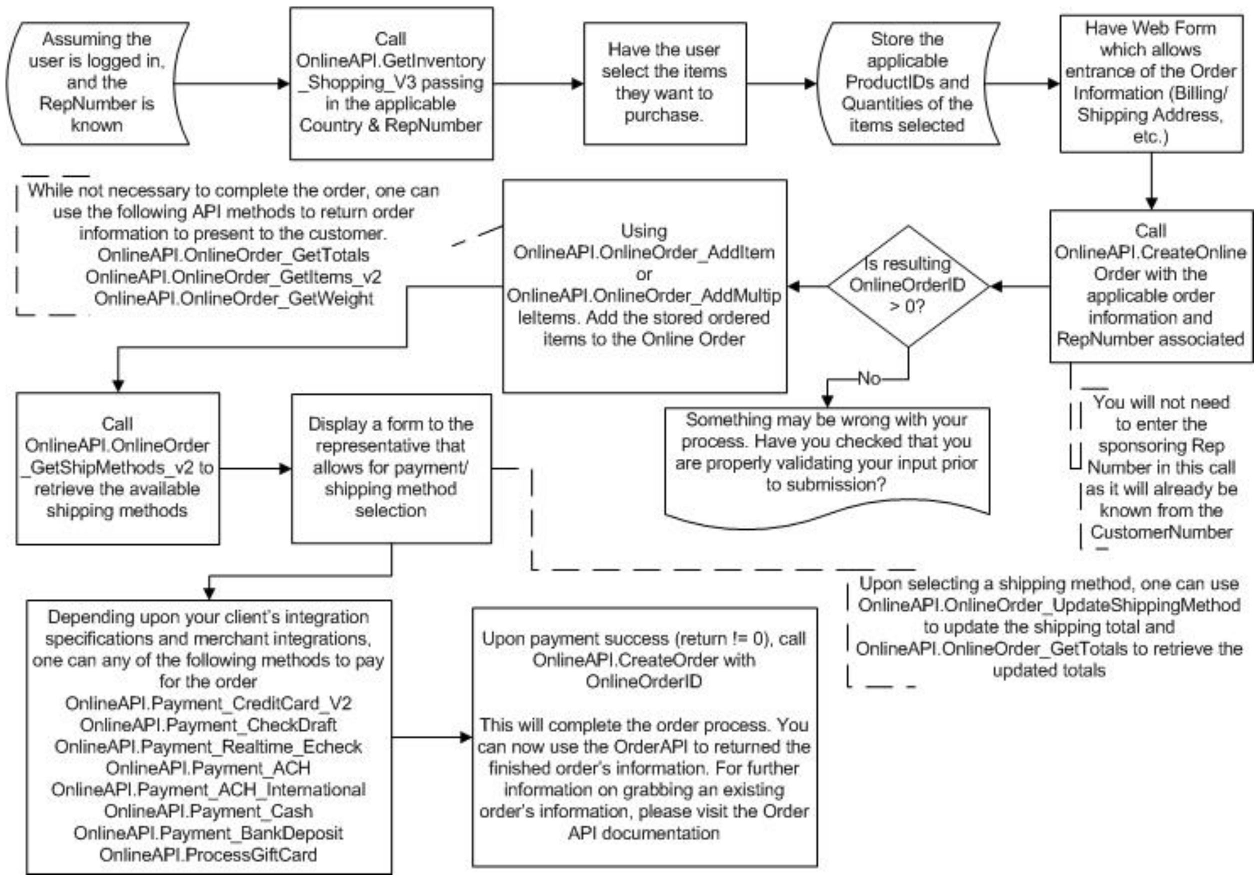
Placing an Order for a Customer

Below we define how to create an order for a customer.


Assuming the user is logged in, and the CustomerNumber is known
Call OnlineAPI.GetInventory_Shopping_V3 passing in the applicable Country & CustomerNumber
Have the user select the items they want to purchase. Store
Store the applicable ProductIDs and Quantities of the items selected
Have Web Form which allows entrance of the Order Information (Billing/ Shipping Address, etc.)
Call OnlineAPI.CreateOnlineOrder with the applicable order information and CustomerNumber associated
You will not need to enter the sponsoring Rep Number in this call as it will already be known from the CustomerNumber
Is resulting OnlineOrderID > 0?
No
Something may be wrong with your process. Have you checked that you are properly validating your input prior to submission?
Using OnlineAPI.OnlineOrder_AddItem or OnlineAPI.OnlineOrder_AddMultipleItems. Add the stored ordered items to the Online Order
While not necessary to complete the order, one can use the following API methods to return order information to present to the customer. OnlineAPI.OnlineOrder_GetTotals OnlineAPI.OnlineOrder_GetItems_v2 OnlineAPI.OnlineOrder_GetWeight
Call OnlineAPI.OnlineOrder_GetShipMethods_v2 to retrieve the available shipping methods
Display a form to the customer that allows for payment/shipping method selection
Upon selecting a shipping method, one can use OnlineAPI.OnlineOrder_UpdateShippingMethod to update the shipping total and OnlineAPI.OnlineOrder_GetTotals to retrieve the updated totals
Depending upon your client's integration specifications and merchant integrations, one can any of the following methods to pay for the order OnlineAPI.Payment_CreditCard_V2 OnlineAPI.Payment_CheckDraft OnlineAPI.Payment_Realtime_Echeck OnlineAPI.Payment_ACH OnlineAPI.Payment_ACH_International OnlineAPI.Payment_Cash OnlineAPI.Payment_BankDeposit OnlineAPI.ProcessGiftCard
Upon payment success (return != 0), call OnlineAPI.CreateOrder with OnlineOrderID
This will complete the order process. You can now use the OrderAPI to returned the finished order's information. For further information on grabbing an existing order's information, please visit the Order API documentation

Placing an Order for a Representative

Below, we define how to create an order for a representative.


Assuming the user is logged in, and the RepNumber is known
Call OnlineAPI.GetInventory_Shopping_V3 passing in the applicable Country & RepNumber
Have the user select the items they want to purchase.
Store the applicable ProductIDs and Quantities of the items selected
Have Web Form which allows entrance of the Order Information (Billing/ Shipping Address, etc.)
Call OnlineAPI.CreateOnline Order with the applicable order information and RepNumber associated
You will not need to enter the sponsoring Rep Number in this call as it will already be known from the CustomerNumber
Is resulting OnlineOrderID > 0?
No
Something may be wrong with your process. Have you checked that you are properly validating your input prior to submission?
Using OnlineAPI.OnlineOrder_AddItem or OnlineAPI.OnlineOrder_AddMultipleItems. Add the stored ordered items to the Online Order
While not necessary to complete the order, one can use the following API methods to return order information to present to the customer. OnlineAPI.OnlineOrder_GetTotals OnlineAPI.OnlineOrder_GetItems_v2 OnlineAPI.OnlineOrder_GetWeight
Call OnlineAPI.OnlineOrder_GetShipMethods_v2 to retrieve the available shipping methods
Display a form to the representative that allows for payment/ shipping method selection
Upon selecting a shipping method, one can use OnlineAPI.OnlineOrder_UpdateShippingMethod to update the shipping total and OnlineAPI.OnlineOrder_GetTotals to retrieve the updated totals
Depending upon your client's integration specifications and merchant integrations, one can any of the following methods to pay for the order OnlineAPI.Payment_CreditCard_V2 OnlineAPI.Payment_CheckDraft OnlineAPI.Payment_Realtime_Echeck OnlineAPI.Payment_ACH OnlineAPI.Payment_ACH_International OnlineAPI.Payment_Cash OnlineAPI.Payment_BankDeposit OnlineAPI.ProcessGiftCard
Upon payment success (return != 0), call OnlineAPI.CreateOrder with OnlineOrderID
This will complete the order process. You can now use the OrderAPI to returned the finished order's information. For further information on grabbing an existing order's information, please visit the Order API documentation

Tasks with Process Flow

Overview

There are tons of ways one could use the ByDesign Online API for integrations so rather than attempt to cover all possible tasks from a large view point; the following list provides a short call list for a particular sub task. Each task defines the API(s) that should be called as well as any specification as to what parameter should be passed or actions to take place upon receiving the result.

Retrieving a Rep's Information

One can retrieve a rep's information by calling OnlineAPI.GetRepInfo with the specified RepNumber or Rep URL. An example is shown below:

```
GetRepInfoResponse resp = _ByDesignOnlineAPI.GetRepInfo(_ByDesignAPICredentials, "1");

if (resp.Success != 0)
        Console.WriteLine("The Rep's first name is {0}", resp.Firstname);
```

Updating a Rep's Information

One can update a rep's information by calling OnlineAPI.UpdateRepInfo_V2 with the updated information in a UpdateRepInfo_v2 request. One does not need to fill out all information being updated, but can simply pass the items needed for update. An example is shown below:

```
UpdateRepInfo_v2 Request = new UpdateRepInfo_v2();
Request.Email = "newemail@email.com";

GenericResponse resp = _ByDesignOnlineAPI.UpdateRepInfo_v2(_ByDesignAPICredentials, Request, "12345");
```

```
    if (resp.Success == 1)
        Console.WriteLine("Save Successful");
```

Retrieving a Customer's Information

One can retrieve a customer's information by calling OnlineAPI.GetCustomerInfo with the specified CustomerNumber. An example is shown below:

```
CustomerInfo resp = _ByDesignOnlineAPI.GetCustomerInfo(_ByDesignAPICredentials, "101");

if (resp.Success != 0)
        Console.WriteLine("The Customer's first name is {0}", resp.FirstName);
```

Updating a Customer's Information

One can update a customer's information by calling OnlineAPI.UpdateCustomerInfo with the updated information in an UpdateCustomerInfo request. One does not need to fill out all information being updated, but can simply pass the items needed for update. An example is shown below:

```
    UpdateCustomerInfo Request = new UpdateCustomerInfo();
    Request.Email = "newemail@email.com";

    GenericResponse resp = _ByDesignOnlineAPI.UpdateCustomerInfo(_ByDesignAPICredentials, Request, 1);

    if (resp.Success == 1)
        Console.WriteLine("Save Successful");
```

Retrieving all Customer's for a Representative

One can retrieve all customers for a given Representative number by calling OnlineAPI.GetRepCustomers. An example is shown below:

```
RepCustomersResponse[] resp = _ByDesignOnlineAPI.GetRepCustomers(_ByDesignAPICredentials, "1");

Console.WriteLine("The Rep has {0} customers.",resp.Count<RepCustomersResponse>());
```

Associate an Order to a Party

One can associate a shopping cart order to a particular party by having the user enter the given party ID. Once validated, the order can be associated to the given party. An example is shown below:

```
ValidatePartyIDResponse resp = _ByDesignOnlineAPI.ValidatePartyID(_ByDesignAPICredentials, EnteredPartyID);

if(resp.Success == 0)
        Console.WriteLine("The Party ID {0} was not found",EnteredPartyID);
        //this would normally be done via ajax upon attempting to complete the order, but is placed here as
example
else
{
        //Complete online order and return OrderID from successful completion

if (OrderCreationSuccess)
        _ByDesignOnlineAPI.AssociateOrderToParty(_ByDesignAPICredentials, OrderID, EnteredPartyID);
}
```

Login a Representative

If you want to verify a rep's authentication, one can do so by calling OnlineAPI.LoginCheck_Rep. An example is shown below:

```
LoginReturn_Rep resp = _ByDesignOnlineAPI.LoginCheck_Rep(_ByDesignAPICredentials, username, password);

if (resp.Success != 0)
        Console.WriteLine("Rep# {0} has successfully logged in", resp.RepNumber);
```

Login a Customer

If you want to verify a customer's authentication, one can do so by calling OnlineAPI.LoginCheck_Customer. An example is shown below:

```
LoginReturn_Customer resp = _ByDesignOnlineAPI.LoginCheck_Customer(_ByDesignAPICredentials, username, password);

if (resp.Success != 0)
        Console.WriteLine("Customer# {0} has successfully logged in", resp.CustomerNumber);
```

Generating AutoResponders

A common problem encountered during new integrations lies within emails being sent. There are multiple ways to send AutoResponders. If one would like, ByDesign can turn on a setting to automatically send AutoResponders when a rep is created or when a customer is created, but the simplest method of integrating AutoResponders is to always call OnlineAPI.GenerateAR after creating a new Order, Rep, Customer, or Lead. This can be done by calling OnlineAPI.GenerateAR with the correct context. An example is shown below:[9][10]

```
GenericResponse resp = _ByDesignOnlineAPI.GenerateAR(_ByDesignAPICredentials, ARTypes.NewOrder, 1, "1001");

if (resp.Success == 1)
    Console.WriteLine("AutoResponder request sent successfully");
```

[9] The Locale ID can be retrieved from OnlineAPI.GetLocales. By default, the only Locale ID is 1 (English,US), but when going international, multiple locales could be applicable. For instance, if I am integrating a website for Mexico, I may also get LocaleID 52 from GetLocales, and that locale should be used in GenerateAR to allow for the properly translated email to be sent.

[10] The Subject will be on the following IDs. Rep #, Customer #, Order #, or Lead #. Please watch the status message if encountering trouble as it will inform you if an improper subject was passed.

ByDesign Extranet API

URL

The ByDesign Extranet API is available at https://api.securefreedom.com/{CLIENTNAME}/Webservice/ExtranetAPI.asmx?wsdl where {CLIENTNAME} is the client's main folder directory.

Overview

As of right now, no detailed documentation is available for this API. ByDesign has numerous APIs and we are working to provide better documentation. If you have any questions concerning this API, please contact support using the Technical Support information within this document.

Tasks With Process Flow

ByDesign Order API

URL

The ByDesign Order API is available at https://api.securefreedom.com/{CLIENTNAME}/Webservice/OrderAPI.asmx?wsdl where {CLIENTNAME} is the client's main folder directory.

Overview

As of right now, no detailed documentation is available for this API. ByDesign has numerous APIs and we are working to provide better documentation. If you have any questions concerning this API, please contact support using the Technical Support information within this document.

ByDesign Generic Web Service API

URL

The ByDesign Generic Web Service API is available at https://api.securefreedom.com/{CLIENTNAME}/Webservice/GenericWebService.asmx?wsdl where {CLIENTNAME} is the client's main folder directory.

Overview

As of right now, no detailed documentation is available for this API. ByDesign has numerous APIs and we are working to provide better documentation. If you have any questions concerning this API, please contact support using the Technical Support information within this document.

ByDesign Online Leads API

URL

The ByDesign Online Leads API is available at https://api.securefreedom.com/{CLIENTNAME}/Webservice/OnlineLeadsAPI.asmx?wsdl where {CLIENTNAME} is the client's main folder directory.

Overview

As of right now, no detailed documentation is available for this API. ByDesign has numerous APIs and we are working to provide better documentation. If you have any questions concerning this API, please contact support using the Technical Support information within this document.

ByDesign Promotions API

URL

The ByDesign API is available at https://api.securefreedom.com/{CLIENTNAME}/Webservice/PromotionsAPI.asmx?wsdl where {CLIENTNAME} is the client's main folder directory.

Overview

As of right now, no detailed documentation is available for this API. ByDesign has numerous APIs and we are working to provide better documentation. If you have any questions concerning this API, please contact support using the Technical Support information within this document.

ByDesign Replicated Site API

URL

The ByDesign API is available at https://api.securefreedom.com/{CLIENTNAME}/Webservice/ReplicatedSiteAPI.asmx?wsdl where {CLIENTNAME} is the client's main folder directory.

Overview on Replicated Sites

What is a Replicated Site?

The concept of site replication is that an end-user (Customer, Distributor, Prospect, etc) can access a Rep (Distributor, Independent Business Owner, Consultant, etc) website by going to a URL that is assigned to this Rep. The basic steps to setting up a Replicated website are as follows:

1. Company produces a Rep Website that is the basis for the Replicated Website. This can be a variation of your existing website that displays some Rep information (e.g.: "John Smith welcomes you to…"), or as far as the Replicated Website Text and/or Rep Image.
2. Company integrates into the Replicated Website API to display appropriate information.
3. Company creates links to pages that they want to have on their websites, such as:
   a. Rep Login
   b. New Rep Enrollment
   c. Shopping Cart and Purchasing of Products

Technical Concept for Internet Information Server (IIS) Environment

The most common implementation of the replicated site in a Windows Server and Microsoft IIS configuration is by utilizing a custom File Not Found ("404") handler. The concept is that when the Rep's URL is entered, for example: http://www.YOUR-COMPANY-DOMAIN-NAME.com/JohnSmith

The folder "JohnSmith" does not exist and the "404" handler is triggered. By default, this would show a "File Not Found" message to the end-user. The method of replication that is used in this instance is overriding the "404" handler to use our "404.asp" (included), which does the following actions:

1. Parses the "JohnSmith" from the "File Not Found" message.
2. Calls the Replicated Website API on ByDesign's Servers to whether or not the URL entered ("JohnSmith") is a valid URL and if so, returns the details of that Rep.
3. Shows the main page of the Replicated Website, displaying the current Rep's information.

Tasks with Process Flow

Validating a Rep URL

Once one has collected the 404 information and received the text appended to the site path, one can make a call to the Replicate Site API to validate the URL.[11]

```
string ReplicatedSiteURL = "JohnSmith";

VerifyRepURLRequest Request = new VerifyRepURLRequest();
Request.Body.creds = _ByDesignAPICredentials;
```

[11] The example will show how to use GetRepInfo_v2, but if one does not wish to retrieve the rep's information, but simply verify the URL, one can use the method VerifyRepURL which simply returns a base Request ID, Success, & Message with no rep information.

```
Request.Body.RepURL = ReplicatedSiteURL;

VerifyRepURLResponse resp = _ByDesignReplicatedSiteAPI.VerifyRepURL(Request);

if (resp.Body.VerifyRepURLResult.Success == 1)
{
    Console.WriteLine("The URL is valid");
}
```

Retrieving Reps' Information using GetRepInfo

If one would not only like to validate the rep's replicated site access, but also retrieve the representative's information for display, one can use the method *GetRepInfo* to both validate and retrieve the reps' information.

```
string ReplicatedSiteURL = "JohnSmith";

GetRepInfo_v2Request Request = new GetRepInfo_v2Request();
Request.Body.Credentials = _ByDesignAPICredentials;
Request.Body.Replicated = ReplicatedSiteURL;

GetRepInfo_v2Response resp = _ByDesignReplicatedSiteAPI.GetRepInfo_v2(Request);

if (resp.Body.GetRepInfo_v2Result.Success == 1)
{
    Console.WriteLine("This URL belongs to {0}",resp.Body.GetRepInfo_v2Result.FirstName);
}
```

Replicated Site Marketing Links

Links below explain the different types of links that may be found on your marketing site. These links are always directed away from your site and onto ByDesign's Freedom servers. For this reason, it is generally a good idea to navigate the user to a new window so they do not lose sight of your own web site.

The following are the areas that can be linked to from your marketing site:

1. Login Link
2. Join Now Link
3. Customer Signup
4. Purchase Link

The look and feel of these pages can be controlled by modifying the headers, footers, background colors/images and/or banners. Due to problems with IIS, putting our site inside a FRAME is NOT supported.

Login Link

There is a separate section of this documented "Extranet Login Page Instructions" available that explains general and advanced details with the login page. However, method to send visitors to your Login Link is the following:

https://extranet.SecureFreedom.com/YOUE-COMPANY-NAME/login.aspx

Join Now Link

This link is designed to take people to your signup form. At the conclusion of the signup process, the visitor will have signed up as a Rep based on the placement information passed in. If no RepDID is passed along to EnrollNew.asp, the company has a few options:

1. They can solicit the user to enter the Rep Number of the person who referred to them to the site.
2. They can automatically place the visitor under a specific Rep Number.

The best way to link to the enrollment is as follows:
https://extranet.securefreedom.com/YOUR-COMPANY-NAME/Signup/EnrollNew.asp?RepDID=<RepDID>

Customer Signup

This link is designed to take customers to your Customer Signup form on the ByDesign Servers. Once they complete the form, they will be assigned to a Customer Number under the passed Rep.

The best way to link to the customer enrollment is as follows:
https://extranet.securefreedom.com/YOUR-COMPANY-NAME/Shopping/ShoppingCart_LoadPage.asp?OrderType=C&RepID=<RepID>&NewCust=1

Shopping Cart Links

This Shopping Cart Links are designed to allow a Customer or Rep to order from your website. This can be used in multiple ways, such as:

1. Linking directly to the Shopping Cart for a visitor to browse your product categories.
2. Linking directly to a specific Category of Products
3. Linking directly to a specific product.
4. Automatically going to a shopping cart login.
5. Automatically adding items to the Shopping Cart from links on your marketing site.

Customer Ordering for a Specific Rep

This link is used for Customer Ordering for a Specific Rep.
https://extranet.securefreedom.com/YOUR-COMPANY-NAME/Shopping/ShoppingCart_LoadPage.asp?OrderType=C&RepID=<RepID>

Linking directly to a specific Category of Products

For linking directly to a Category of Products, the following link is used:
https://extranet.securefreedom.com/YOUR-COMPANY-NAME/Shopping/ShoppingCart_LoadPage.asp?OrderType=C&RepID=<RepID>&Cat=<CategoryName>

Linking directly to a Specific Product

For linking directly to a specific product, the following link is used:
https://extranet.securefreedom.com/YOUR-COMPANY-NAME/Shopping/ShoppingCart_LoadPage.asp?OrderType=C&RepID=<RepID>&ProdID=<ProductID>

Automatically going to a Shopping Cart Login

For going automatically to the Shopping Cart Login, the following are the parameters that need to be passed in:
https://extranet.securefreedom.com/YOUR-COMPANY-NAME/Shopping/ShoppingCart_LoadPage.asp?OrderType=C&RepID=<RepID>&GoLogin=1

ByDesign Restful API

URL

The ByDesign Restful API is available at https://webapi.securefreedom.com/{CLIENTNAME} where {CLIENTNAME} is the client's main folder directory.

Overview

This API provides basic GET functionality to most the major object types available within the Freedom system. This API uses Restful APIs authenticated via basic authorization that allow the API consumer to request information in either JSON or XML. This API is critical for those situations in which you want to provide drop downs of important information provided during signup, shopping, or order review. This API is quickly growing with new functionality and will be the preferred format for future APIs.

Tasks with Process Flow

Retrieving Rank Type information

Using the information from the "Calling and Using the Restful API", you can make a request to GET ~/api/Admin/RankType using the credentials provided from ByDesign. You can change the request to return either XML or JSON depending on what your application prefers. The returned response will list out all the rank types with the information you need. Done!

Adding a new Warehouse Territory

The Restful API also has various object types that allow additions and modifications. This example will cover how to add a new object type (Warehouse Territory). For this example, we recommend connecting to your testing environment (which you can request from your contact at ByDesign) so as to not mess up anything when adding or modifying object types. First, make a request to get a sample Warehouse Territory at GET ~/api/Admin/WarehouseTerritory. Using a sample object as an example, make a request to POST ~/api/admin/WarehouseTerritory giving all the parameters for the properties in either the query string or the content-body. (We recommend the content-body.) The API will return the appropriate response (letting you know if there's an incorrect property values or if the addition was successful). Done!

Technical Support

If you need further help, please contact ByDesign Support at 813-253-2275 or at support@mlmbydesign.com. Also, you can always contact your Implementation Manager or Account Manager concerning setting up a meeting with a developer for further technical support.

Revision History

Author	Date	Version	Notes
Christopher Brower	11/16/2010	1.0	Created Initial Documentation
Christopher Brower	12/10/2010	1.01	Added base URLs for other APIs
Christopher Brower	2/11/2011	1.02	Added Replicated Site API