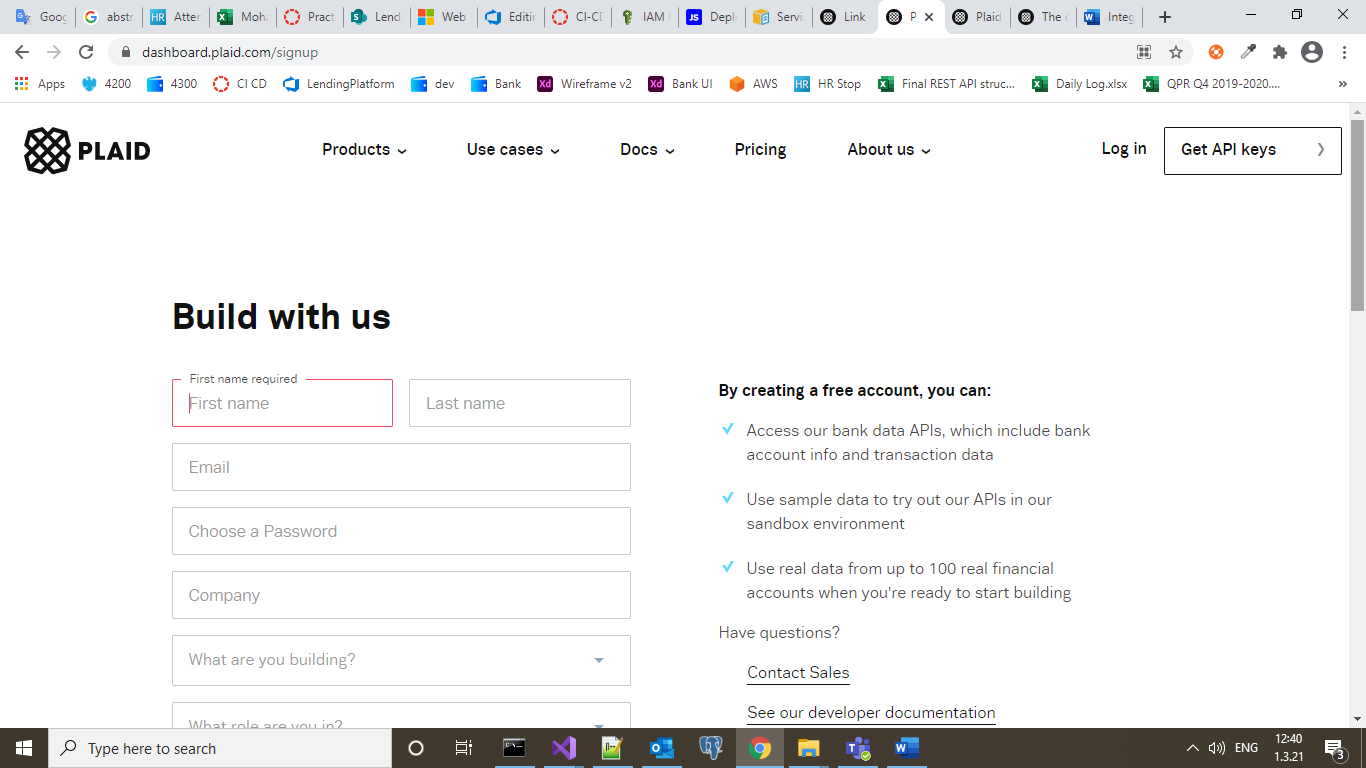
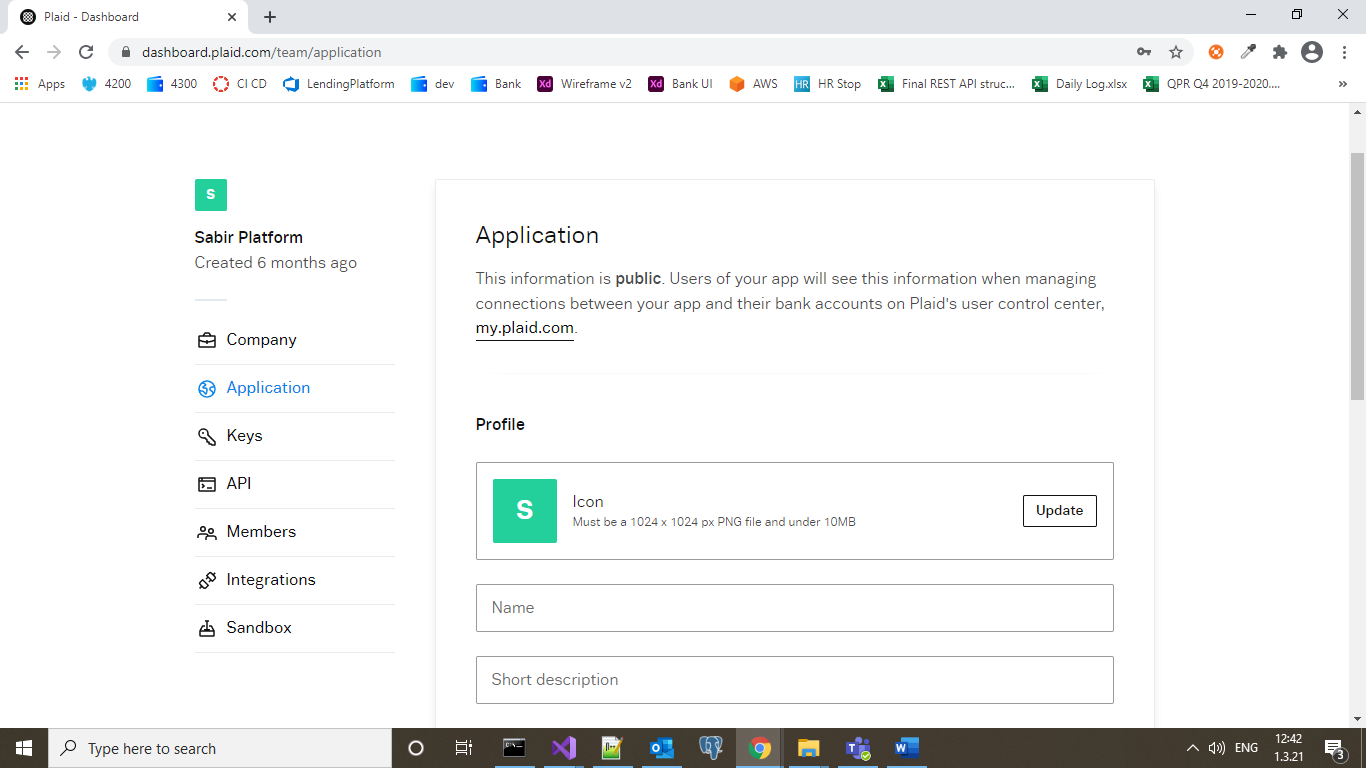
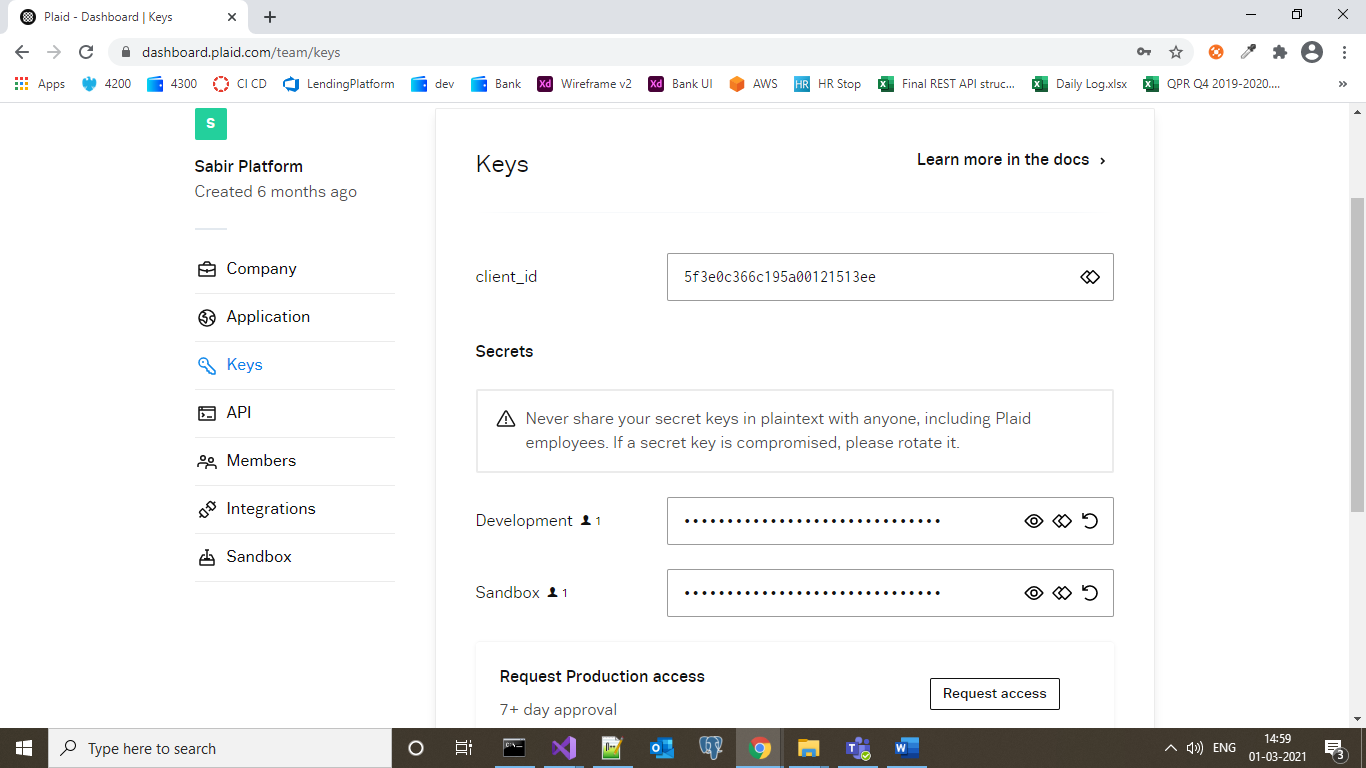
**Detailed Documentation**

* **Name:** [Plaid](https://github.com/thepirat000/Audit.NET/tree/master/src/Audit.EntityFramework)
* **API Version:**2020-09-14
* **Link:** https://plaid.com/
* **Official Documentation Link:** https://plaid.com/docs/
* **Researched Links:**
  1. Use Link to connect to your users' financial accounts with the Plaid API- <https://plaid.com/docs/link/>
  2. API token endpoints - <https://plaid.com/docs/api/tokens/>
  3. User key details - https://dashboard.plaid.com/team/keys
* **Explicit Integration Steps:**

1. Create a account in the Plaid.  
   **Link:** <https://dashboard.plaid.com/signup>   
     
   
2. After successfully signed in, Add the application details of the plaid.  
   **Link:** https://dashboard.plaid.com/team/application  
   
3. There are 3 environments in the plaid. Sandbox, Development and Production. It creates a   
   different secret keys by the environment. This is the link to view the Keys.   
   **Link:** https://dashboard.plaid.com/team/keys
4. Load the javascript in the frontend by calling the loadProviderScript().

loadProviderScript(): void {

// Add the plaid javascript.

var plaidjs = document.createElement("script");

plaidjs.type = "text/javascript";

plaidjs.src = "https://cdn.plaid.com/link/v2/stable/link-initialize.js";

document.getElementById('transactionPop-Up').append(plaidjs);

}

1. Call an API to create a link\_token, which is required as a parameter when initializing Link. CreateLinkTokenAsync(userId) funcation calls by passing the userid.  
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   There are following parameter use in the API.

/// <summary>

/// Generate link token from plaid.

/// </summary>

/// <param name="userId">loggedIn userid</param>

/// <returns>Link token</returns>

public async Task<string> CreateLinkTokenAsync(Guid userId)

{

string[] products = { "auth", "transactions" };

string[] countryCodes = { "US" };

// Create link token.

var result = await CreateLinkTokenAsync(new CreateLinkTokenRequestAC()

{

ClientId = \_configuration.GetValue<string>("PlaidService:ClientId"),

Secret = \_configuration.GetValue<string>("PlaidService:ClientSecret"),

ClientName = \_configuration.GetValue<string>("PlaidService:ClientName"),

Products = products,

CountryCodes = countryCodes,

Language = "en",

User = new PlaidUserAC() { ClientUserId = userId.ToString() }

});

return result.LinkToken;

}  
/// <summary>

/// Creates a Link link\_token.

/// </summary>

/// <param name="request"></param>

/// <returns></returns>

private async Task<CreateLinkTokenResponseAC> CreateLinkTokenAsync(CreateLinkTokenRequestAC request)

{

return await PostAsync<CreateLinkTokenResponseAC>("link/token/create", request);

}

private async Task<TResponse> PostAsync<TResponse>(string path, SerializableContentAC request) where TResponse : ResponseBaseAC

{

using (var http = new HttpClient())

{

string url = GetEndpoint(path);

string json = request.ToJson();

var body = Body(json);

body.Headers.Add("Plaid-Version", \_configuration.GetValue<string>("PlaidService:Version"));

using (HttpResponseMessage response = await http.PostAsync(url, body))

{

json = await response.Content.ReadAsStringAsync();

if (response.IsSuccessStatusCode)

{

TResponse result = JsonConvert.DeserializeObject<TResponse>(json,

new JsonSerializerSettings

{

ContractResolver = new DefaultContractResolver { NamingStrategy = new SnakeCaseNamingStrategy() },

Formatting = Formatting.Indented

});

result.StatusCode = response.StatusCode;

#if DEBUG

result.RawJsonForDebugging = json;

#endif

return result;

}

else

{

var error = JObject.Parse(json);

throw new HttpRequestException(error["error\_message"].Value<string>());

}

}

}

}

* + [**client\_id**](https://plaid.com/docs/api/tokens/#link-token-create-request-client-id) : Your Plaid API client\_id.
  + [secret](https://plaid.com/docs/api/tokens/#link-token-create-request-secret) : Your Plaid API secret.
  + **client\_name** : The name of your application, as it should be displayed in Link.
  + **Language** : The language that Link should be displayed in.eg. ‘en’ for English.
  + **country\_codes**: Specify an array of Plaid-supported country codes using the ISO-3166-1 alpha-2 country code standard. We use for ['US']
  + **user** : An object specifying information about the end user who will be linking their account. We pass **client\_user\_id** in this object that we passed in this function.
  + [**products**](https://plaid.com/docs/api/tokens/#link-token-create-request-products) : List of Plaid product(s) you wish to use. We use ['auth', 'transactions'].  
    **Note** **:** This is the link to see the client keys. https://dashboard.plaid.com/team/keys

1. We got a link token at frontend. We have to open the dialogue for select a bank option. We need to pass the link token in the config.  
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   Above code is for the open the dialogue. In frontend, User will select the bank and enter the credential.

// Fetch a link token from server and pass it back to app to initialize Link.

this.companyInfoService.getPlaidLinkToken().subscribe((linkToken: string) => {

this.openPlaidDialogue(window, linkToken);

}, (err) => {

this.loader = false;

if (err.status === 400) {

this.toastrService.error(err.response);

}

});

/\*\*

\* Plaid script for open the plaid dialogue.

\* @param window

\* @param fetchLinkToken

\*/

openPlaidDialogue(window: any, fetchLinkToken: string) {

const self = this;

const configs = {

token: fetchLinkToken,

onSuccess: async function (public\_token) {

// Send the public\_token to server and fetch the bank transactions.

self.getPlaidBankTransactions(public\_token);

},

onExit: async function (err) {

self.loader = false;

if (err != null) {

// The user encountered a Plaid API error prior to exiting.

handler.destroy();

self.toastrService.error(err.error\_message);

}

}

};

const handler = window.Plaid.create(configs);

handler.open();

}

1. If credential is valid then invoke **onSuccess** method else invoke **onExit** method.   
   In **onSuccess**, We got public token for user. Public token is used for fetch the user bank transaction.
2. In backend, GetTransactionInfoAsync(publicToken) function call for fetching the bank transaction. There are 2 Plaid API call for fetch the bank.
   * 1st API call for fetch the access token by passing the public token.

/// <summary>

/// Generate access token from plaid.

/// </summary>

/// <param name="publicToken">public token</param>

/// <returns>Access token</returns>

private async Task<string> CreateAccessTokenAsync(string publicToken)

{

// Create access token.

var result = await ExchangeTokenAsync(new ExchangeTokenRequestAC()

{

ClientId = \_configuration.GetValue<string>("PlaidService:ClientId"),

Secret = \_configuration.GetValue<string>("PlaidService:ClientSecret"),

PublicToken = publicToken

});

return result.AccessToken;

}  
/// <summary>

/// Exchanges a Link public\_token for an API access\_token.

/// </summary>

/// <param name="request">The request.</param>

/// <returns>Task&lt;Management.ExchangeTokenResponse&gt;.</returns>

private async Task<ExchangeTokenResponseAC> ExchangeTokenAsync(ExchangeTokenRequestAC request)

{

return await PostAsync<ExchangeTokenResponseAC>("item/public\_token/exchange", request);

}

* + 2nd API call for fetch the bank transaction by using the public token.

/// <summary>

/// Fetch the user bank transaction.

/// </summary>

/// <param name="publicToken"></param>

/// <returns>Returns GetTransactionsResponse object contains account and transaction.</returns>

public async Task<GetTransactionsResponseAC> GetTransactionInfoAsync(string publicToken)

{

// Generate access token.

string accessToken = await CreateAccessTokenAsync(publicToken);

int lastNYears = \_configuration.GetValue<int>("FinancialYear:Years");

var fromYear = DateTime.UtcNow.Year - lastNYears;

var fromDate = new DateTime(year: fromYear, month: 1, day: 1);

var endDate = DateTime.UtcNow;

uint offset = 0, fetchedRecords = 0, totalFetchRecords = 500;

GetTransactionsResponseAC getTransactionsResponseAC = new GetTransactionsResponseAC();

do

{

// Retrieving a user's recent transactions.

var result = await FetchTransactionsAsync(new GetTransactionsRequestAC()

{

ClientId = \_configuration.GetValue<string>("PlaidService:ClientId"),

Secret = \_configuration.GetValue<string>("PlaidService:ClientSecret"),

AccessToken = accessToken,

StartDate = fromDate,

EndDate = endDate,

Options = new PaginationOptionsAC()

{

Offset = offset,

Count = totalFetchRecords

}

});

if (offset == 0)

{

getTransactionsResponseAC = result;

}

else

{

getTransactionsResponseAC.Transactions.AddRange(result.Transactions);

}

fetchedRecords += totalFetchRecords;

offset++;

} while (getTransactionsResponseAC.TotalTransactions > fetchedRecords);

return getTransactionsResponseAC;

}

/// <summary>

/// Retrieves user-authorized transaction data for credit and depository-type <see cref="Entity.Account"/>.

/// </summary>

/// <param name="request">The request.</param>

/// <returns>Task&lt;Transactions.GetTransactionsResponse&gt;.</returns>

private async Task<GetTransactionsResponseAC> FetchTransactionsAsync(GetTransactionsRequestAC request)

{

return await PostAsync<GetTransactionsResponseAC>("transactions/get", request);

}

**Note** : API returns only 500 transactions. If user wants more transactions then we use loop to fetch more transactions till end of transactions. In API response, We got total user transaction in the TotalTransactions field. We pass start date and end date for transaction.

1. frontend, HTML code for display the bank transaction in the frontend.

<!-- Transactions table -->

<div class="sticky-table mt-3" \*ngIf="transactionsArray.length > 0">

<table class="table table-bordered invoice-table table-fixed mb-0">

<tr>

<th scope="col" class="width30"><span></span></th>

<th scope="col" class="col-120"><span>Date</span></th>

<th scope="col" class="col-120"><span>Amount</span></th>

<th scope="col" class="col-180"><span>Type</span></th>

<th scope="col"><span>Description</span></th>

</tr>

<tbody>

<tr \*ngFor="let x of transactionsArray">

<td></td>

<td>**{{**x.transactionDate**}}**</td>

<td>$ **{{**x.amount.amount | mask: 'separator':','**}}**</td>

<td>**{{**x.type**}}**</td>

<td>**{{**x.description.simple**}}**</td>

</tr>

</tbody>

</table>

</div>

1. There are following schema for API.

/// <summary>

/// Provides methods and properties for making a standard request.

/// </summary>

/// <seealso cref="Plaid.SerializableContentAC" />

public abstract class RequestBaseAC : SerializableContentAC

{

/// <summary>

/// Gets or sets the secret.

/// </summary>

/// <value>The secret.</value>

public string Secret { get; set; }

/// <summary>

/// Gets or sets the client\_id.

/// </summary>

/// <value>The client identifier.</value>

public string ClientId { get; set; }

/// <summary>

/// Gets or sets the access\_token.

/// </summary>

/// <value>The access token.</value>

public string AccessToken { get; set; }

}

/// <summary>

/// Provide an method for converting an object to a JSON string.

/// </summary>

public abstract class SerializableContentAC

{

/// <summary>

/// The null value handling.

/// </summary>

protected NullValueHandling NullValueHandling = NullValueHandling.Ignore;

/// <summary>

/// Returns a string that represents the current object in JSON format.

/// </summary>

/// <returns>System.String.</returns>

public virtual string ToJson()

{

return JsonConvert.SerializeObject(this, new JsonSerializerSettings()

{

ContractResolver = new DefaultContractResolver { NamingStrategy = new SnakeCaseNamingStrategy() },

DateFormatString = "yyyy-MM-dd",

NullValueHandling = this.NullValueHandling,

#if DEBUG

Formatting = Formatting.Indented

#else

Formatting = Formatting.None

#endif

});

}

}

/// <summary>

/// Provides common members for all Plaid API responses.

/// </summary>

public abstract class ResponseBaseAC

{

#if DEBUG

public string RawJsonForDebugging;

#endif

/// <summary>

/// Gets or sets the request identifier.

/// </summary>

/// <value>The request identifier.</value>

public string RequestId { get; set; }

/// <summary>

/// Gets the http status code.

/// </summary>

/// <value>The status code.</value>

public HttpStatusCode StatusCode { get; internal set; }

}

/// <summary>

/// Represents a request for plaid's '/link/token/create' endpoint. Create a link\_token.

/// </summary>

/// <seealso cref="Plaid.SerializableContentAC" />

public class CreateLinkTokenRequestAC : SerializableContentAC

{

/// <summary>

/// Gets or sets the client identifier.

/// </summary>

/// <value>The client identifier.</value>

public string ClientId { get; set; }

/// <summary>

/// Gets or sets the secret.

/// </summary>

/// <value>The secret.</value>

public string Secret { get; set; }

/// <summary>

/// Gets or sets the client name.

/// </summary>

/// <value>The client name.</value>

public string ClientName { get; set; }

/// <summary>

/// Gets or sets the language.

/// </summary>

/// <value>The language.</value>

public string Language { get; set; }

/// <summary>

/// Gets or sets the country codes.

/// </summary>

/// <value>The country codes.</value>

public string[] CountryCodes { get; set; }

/// <summary>

/// Gets or sets the user.

/// </summary>

/// <value>The user.</value>

public PlaidUserAC User { get; set; }

/// <summary>

/// Gets or sets the products.

/// </summary>

/// <value>The products.</value>

public string[] Products { get; set; }

}

/// <summary>

/// Represents a response from plaid's '/link/token/create' endpoint. Create a link\_token.

/// </summary>

/// <seealso cref="Plaid.ResponseBaseAC" />

public class CreateLinkTokenResponseAC : ResponseBaseAC

{

/// <summary>

/// Gets or sets the link token.

/// </summary>

/// <value>The link token.</value>

public string LinkToken { get; set; }

/// <summary>

/// Gets or sets the expiration.

/// </summary>

/// <value>The expiration.</value>

public string Expiration { get; set; }

}

/// <summary>

/// Represents a request for plaid's '/item/public\_token/exchange' endpoint. Exchange a Link public\_token for an API access\_token.

/// </summary>

/// <seealso cref="Plaid.SerializableContentAC" />

public class ExchangeTokenRequestAC : SerializableContentAC

{

/// <summary>

/// Gets or sets the public\_token.

/// </summary>

/// <value>The public token.</value>

public string PublicToken { get; set; }

/// <summary>

/// Gets or sets the client identifier.

/// </summary>

/// <value>The client identifier.</value>

public string ClientId { get; set; }

/// <summary>

/// Gets or sets the secret.

/// </summary>

/// <value>The secret.</value>

public string Secret { get; set; }

}

/// <summary>

/// Represents a response from plaid's '/item/public\_token/exchange' endpoint. Exchange a Link public\_token for an API access\_token.

/// </summary>

/// <seealso cref="Plaid.ResponseBaseAC" />

public class ExchangeTokenResponseAC : ResponseBaseAC

{

/// <summary>

/// Gets or sets the <see cref="Entity.ItemAC"/> identifier.

/// </summary>

/// <value>The item identifier.</value>

public string ItemId { get; set; }

/// <summary>

/// Gets or sets the access token.

/// </summary>

/// <value>The access token.</value>

public string AccessToken { get; set; }

}

/// <summary>

/// Represents pagination options.

/// </summary>

public class PaginationOptionsAC

{

/// <summary>

/// Gets or sets the number of transactions to fetch, where 0 &lt; count &lt;= 500.

/// </summary>

/// <value>The number of transactions to return.</value>

public uint Count { get; set; }

/// <summary>

/// Gets or sets the number of transactions to skip, where offset &gt;= 0.

/// </summary>

/// <value>The offset.</value>

public uint Offset { get; set; }

/// <summary>

/// Gets or sets the list of account ids to retrieve for the <see cref="Entity.ItemAC" />. Note: An error will be returned if a provided account\_id is not associated with the <see cref="Entity.ItemAC" />.

/// </summary>

/// <value>The account ids.</value>

public string[] AccountIds { get; set; }

}

public class GetTransactionsRequestAC : RequestBaseAC

{

/// <summary>

/// Initializes a new instance of the <see cref="GetTransactionsRequestAC"/> class.

/// </summary>

public GetTransactionsRequestAC()

{

EndDate = DateTime.Now;

StartDate = DateTime.Now.Subtract(TimeSpan.FromDays(30));

}

/// <summary>

/// Gets or sets the start date.

/// </summary>

/// <value>The start date.</value>

public DateTime StartDate { get; set; }

/// <summary>

/// Gets or sets the end date.

/// </summary>

/// <value>The end date.</value>

public DateTime EndDate { get; set; }

/// <summary>

/// Gets or sets the pagination options.

/// </summary>

/// <value>The pagination options.</value>

public PaginationOptionsAC Options { get; set; }

}

public class GetTransactionsResponseAC : ResponseBaseAC

{

/// <summary>

/// Gets or sets the number of transactions returned.

/// </summary>

/// <value>The number of transactions returned.</value>

public int TotalTransactions { get; set; }

/// <summary>

/// Gets or sets the transactions.

/// </summary>

/// <value>The transactions.</value>

public List<PlaidTransactionAC> Transactions { get; set; }

/// <summary>

/// Gets or sets the item.

/// </summary>

/// <value>The item.</value>

public ItemAC Item { get; set; }

/// <summary>

/// Gets or sets the accounts.

/// </summary>

/// <value>The accounts.</value>

public List<AccountAC> Accounts { get; set; }

}

public class PlaidTransactionAC

{

public string Name { get; set; }

public string TransactionId { get; set; }

public string AccountId { get; set; }

public string[] Category { get; set; }

public string CategoryId { get; set; }

public string TransactionType { get; set; }

public decimal Amount { get; set; }

public string IsoCurrencyCode { get; set; }

public string UnofficialCurrencyCode { get; set; }

public string CurrencyCode => IsoCurrencyCode ?? UnofficialCurrencyCode ?? null;

public DateTime Date { get; set; }

public LocationInfoAC Location { get; set; }

public PaymentInfoAC PaymentMeta { get; set; }

public bool Pending { get; set; }

public string PendingTransactionId { get; set; }

public string AccountOwner { get; set; }

public TransactionType Type

{

get

{

bool valid = Enum.TryParse(TransactionType, out TransactionType type);

return valid ? type : Enums.Plaid.TransactionType.Unresolved;

}

}

}

* **Roadblocks faced/Solutions derived**

1. There are two services use for fetch the bank transactions. First is Plaid and second is Yodlee so we have to load the .js file when it needed. We need to create a configs and pass in the plaid js function. Function returns the handler then open the dialogue by use of the hander method.  
   **Problem:** In this, We don’t get the javascript function from the typescript and we had tried directly use window to call the js function. Like this **window.Plaid.create(configs);** but couldn’t get the method.  
   **Solution:** We pass window object from the function parameter then call a method like this **window.Plaid.create(configs);** then it works.
2. In above function, If user enters valid credential then invoke success method and returns public key. We use of public key and we have to call a typescript method from success method but success method is in the js function.

**Problem:** We call the getPlaidBankTransactions(public\_token); function from the success method but couldn’t find the method.

**Solution:** we need to create a self const and assign **this** keyword then call a typescript function like this. **self.getPlaidBankTransactions(public\_token);** then it works.