https://paypal.github.io/PayPal-NET-SDK/

```
sing System;
using PayPal.Api;
using System.Collections.Generic;
using PayPal.Sample.Utilities;

namespace PayPal.Sample
{
   public partial class PaymentWithPayPal : BaseSamplePage
   {
      protected override void RunSample()
      {
```

Api Context

Pass in a APIContext object to authenticate the call and to send a unique request id (that ensures idempotency). The SDK generates a request id if you do not pass one explicitly. See Configuration.cs to know more about APIContext.

```
         var apiContext = Configuration.GetAPIContext();

         string payerId = Request.Params["PayerID"];
         if (string.IsNullOrEmpty(payerId))
         {
```

Items

Items within a transaction.

```
            var itemList = new ItemList()
            {
               items = new List<Item>()
               {
                  new Item()
                  {
                     name = "Item Name",
                     currency = "USD",
                     price = "15",
                     quantity = "5",
                     sku = "sku"
                  }
               }
            };
```

Payer

A resource representing a Payer that funds a payment Payment Method as paypal

```
            var payer = new Payer() { payment_method = "paypal" };
```

Redirect URLS
These URLs will determine how the user is redirected from PayPal once they have either approved or canceled the payment.

```
var baseURI = Request.Url.Scheme + "://" + Request.Url.Authority +
"/PaymentWithPayPal.aspx?";
var guid = Convert.ToString((new Random()).Next(100000));
var redirectUrl = baseURI + "guid=" + guid;
var redirUrls = new RedirectUrls()
{
   cancel_url = redirectUrl + "&cancel=true",
   return_url = redirectUrl
};
```

Details
Let's you specify details of a payment amount.

```
var details = new Details()
{
   tax = "15",
   shipping = "10",
   subtotal = "75"
};
```

Amount
Let's you specify a payment amount.

```
var amount = new Amount()
{
   currency = "USD",
   total = "100.00", // Total must be equal to sum of shipping, tax and subtotal.
   details = details
};
```

Transaction
A transaction defines the contract of a payment - what is the payment for and who is fulfilling it.

```
var transactionList = new List<Transaction>();
```

The Payment creation API requires a list of Transaction; add the created Transaction to a List

```
transactionList.Add(new Transaction()
{
   description = "Transaction description.",
   invoice_number = Common.GetRandomInvoiceNumber(),
   amount = amount,
   item_list = itemList
});
```

Payment
A Payment Resource; create one using the above types and intent as sale or authorize

```
            var payment = new Payment()
            {
                intent = "sale",
                payer = payer,
                transactions = transactionList,
                redirect_urls = redirUrls
            };
```

Create a payment using a valid APIContext

```
            var createdPayment = payment.Create(apiContext);
```

Using the links provided by the createdPayment object, we can give the user the option to redirect to PayPal to approve the payment.

```
            var links = createdPayment.links.GetEnumerator();
            while (links.MoveNext())
            {
                var link = links.Current;
                if (link.rel.ToLower().Trim().Equals("approval_url"))
                {
                    this.flow.RecordRedirectUrl("Redirect to PayPal to approve the payment...",
link.href);
                }
            }
            Session.Add(guid, createdPayment.id);
            Session.Add("flow-" + guid, this.flow);
        }
        else
        {
            var guid = Request.Params["guid"];
```

Using the information from the redirect, setup the payment to execute.

```
            var paymentId = Session[guid] as string;
            var paymentExecution = new PaymentExecution() { payer_id = payerId };
            var payment = new Payment() { id = paymentId };
```

Execute the payment.

```
            var executedPayment = payment.Execute(apiContext, paymentExecution);
```

For more information, please visit PayPal Developer REST API Reference.

```
        }
```

```
    }
```
Create Future Payment Using PayPal
This sample code demonstrates how you can process a future payment made using a PayPal account.

```
    /// <summary>
    /// Code example for creating a future payment object.
    /// </summary>
    /// <param name="correlationId"></param>
    /// <param name="authorizationCode"></param>
    private Payment CreateFuturePayment(string correlationId, string authorizationCode, string
redirectUrl)
    {
```
Payer
A resource representing a Payer that funds a payment Payment Method as paypal

```
        Payer payer = new Payer()
        {
            payment_method = "paypal"
        };
```
Amount
Let's you specify a payment amount.

```
        var amount = new Amount()
        {
            currency = "USD",
```
Total must be equal to sum of shipping, tax and subtotal.

```
            total = "100",
```
Details
Let's you specify details of a payment amount.

```
            details = new Details()
            {
                tax = "15",
                shipping = "10",
                subtotal = "75"
            }
        };
```
Redirect URLS
```
        var redirUrls = new RedirectUrls()
        {
            cancel_url = redirectUrl,
            return_url = redirectUrl
```

```
        };
```
Items
Items within a transaction.

```
        var itemList = new ItemList() { items = new List<Item>() };
        itemList.items.Add(new Item()
        {
            name = "Item Name",
            currency = "USD",
            price = "15",
            quantity = "5",
            sku = "sku"
        });
```
Transaction
A transaction defines the contract of a payment - what is the payment for and who is fulfilling it.

```
        var transactionList = new List<Transaction>();
```
The Payment creation API requires a list of Transaction; add the created Transaction to a List

```
        transactionList.Add(new Transaction()
        {
            description = "Transaction description.",
            amount = amount,
            item_list = itemList
        });

        var authorizationCodeParameters = new CreateFromAuthorizationCodeParameters();
                authorizationCodeParameters.setClientId(Configuration.ClientId);
                authorizationCodeParameters.setClientSecret(Configuration.ClientSecret);
                authorizationCodeParameters.SetCode(authorizationCode);
```
Api Context
Pass in a APIContext object to authenticate the call and to send a unique request id (that ensures idempotency). The SDK generates a request id if you do not pass one explicitly. See Configuration.cs to know more about APIContext.

```
        var apiContext = Configuration.GetAPIContext();


        var tokenInfo =
Tokeninfo.CreateFromAuthorizationCodeForFuturePayments(apiContext,
authorizationCodeParameters);
        var accessToken = string.Format("{0} {1}", tokenInfo.token_type,
tokenInfo.access_token);
```

```
            var futurePaymentApiContext = Configuration.GetAPIContext(accessToken);
Payment
A FuturePayment Resource

            var futurePayment = new FuturePayment()
            {
                intent = "authorize",
                payer = payer,
                transactions = transactionList,
                redirect_urls = redirUrls
            };
            return futurePayment.Create(futurePaymentApiContext, correlationId);
        }
    }
}
```