

SmartRoute Sample Code (C#)

version 1.1

STS PayOne

30 July 2018

PROPRIETARY INFORMATION

STS PayOne and the STS PayOne logo are registered trademarks of STS Payment Network DMCC. Other trademarks and trade names may be used in this document to refer to either the entities claiming the marks and/or the names of their products. STS PayOne disclaims proprietary interest in the marks and names of others.

The information in this document has been reviewed and is believed to be accurate. However, neither STS PayOne nor its affiliates assume any responsibility for inaccuracies, errors, or omissions that may be contained herein. In no event will STS PayOne or its affiliates be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect or omission in this document, even if advised of the possibility of such damages.

STS PayOne reserves the right to make improvements or changes to this document and information contained within, and to the products and services described at any time, without notice or obligation.

STS PayOne understands that any award to supply the products/ provide the services that are the subject of this document are subject to the mutual execution of a definitive written agreement.

All information supplied for the purpose of this document is to be considered STS PayOne confidential.

© 2016 STS PayOne

1 Contents

2 Introduction	4
2.1 Who should read this document?	4
2.2 Purpose	4
2.3 Prerequisites	4
3 Redirect Payment Message	5
3.1 Secure Hash Generation	5
3.2 Redirect Post Request Preparation	6
3.3 Redirect Post Request Submitting	6
3.4 Redirect Payment Response	7
4 Direct Post Payment Message	9
4.1 Secure Hash Generation:	9
4.2 Direct Post Request Preparation	11
4.3 Direct Post Request Submitting	11
4.4 Direct Post Payment Response	12
5 API MPayment	13
5.1 Secure Hash Generation	14
5.2 API MPayment Request	16
5.3 3DS Post Request Submitting to ACS URL	18
5.4 Sadad authentication Submission	20
6 API Payment	21
6.1 Secure Hash Generation	21
6.2 API Payment Request	22
6.3 Sadad authentication Submission	25
7 API Approve	25
7.1 Secure Hash Generation	25
7.2 API Approve Request	27
8 Inquiry Message	28
8.1 Secure Hash Generation	28
8.2 Inquiry Request	29
9 Refund Message	32
9.1 Secure Hash Generation	32
9.2 Refund Request	33

2 Introduction

2.1 Who should read this document?

This document is intended for merchant system designer and developer planning to integrate with SmartRoute interface to perform e-Commerce. Readers of this document should be web application developers.

2.2 Purpose

This guide is written for merchants who have signed up through SmartRoute system to use it as their Integration point with Payment Gateways for handling electronic transactions (payment, refund, confirm.. etc.) and from different payment methods (credit card, debit card...etc.), by using the HTTPS Post as programming interface to perform the transactions. In particular, it describes the format for sending transactions and the corresponding received responses.

2.3 Prerequisites

- Merchant is contracted with Payment Gateway as an e-Commerce Merchant.
- Merchant is contracted with SmartRoute as an Integration point with the Payment Gateways.
- Merchant is provided with a SmartRoute profile to generate Authentication Token used for integration.
- Merchant has a profile at Payment Gateway side.

3 Redirect Payment Message

```
1. using System.Collections.Generic;
2. using System.Security.Cryptography;
3. using System.Text;
```

3.1 Secure Hash Generation

```
1. //Step 1: Generate Secure Hash
2. String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store
   Your Secret Key in safe Place (eg.database)
3.
4.     // put the parameters in a SortedDictionary to have the parameters to have
   them sorted alphabetically.
5.     SortedDictionary<string, string> parameters = new SortedDictionary<String,
   String>();
6.
7.     //getting time in milliseconds
8.
9.     Long transactionId = (DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond);
10.
11.     // fill required parameters
12.     parameters.Add("TransactionID", transactionId.ToString());
13.
14.     parameters.Add("MerchantID", "ANBRedirectM");
15.     parameters.Add("Amount", "2000");
16.     parameters.Add("CurrencyISOCODE", "840");
17.     parameters.Add("MessageID", "1");
18.     parameters.Add("Quantity", "1");
19.     parameters.Add("Channel", "0");
20.     //fill some optional parameters
21.     parameters.Add("Language", "en");
22.     parameters.Add("ThemeID", "10000000001");
23.     parameters.Add("ResponseBackURL", "https://MerchantSite/RedirectPaym
entResponsePage");// if this url is configured for the merchant it's not required
24.     parameters.Add("Version", "1.0");
25.
26.     //Create an Ordered String of The Parameters Dictionary with Secret
   Key
27.     StringBuilder orderedString = new StringBuilder();
28.     orderedString.Append(SECRET_KEY);
29.     foreach (KeyValuePair<string, string> kv in parameters)
30.     {
31.         orderedString.Append(kv.Value);
32.     }
33.     Console.WriteLine("orderdString: " + orderedString);
34.
35.     // Generate SecureHash with SHA256
36.     SHA256 sha256;
37.     byte[] bytes, hash;
38.     string secureHash = string.Empty;
39.
40.     bytes = Encoding.UTF8.GetBytes(orderedString.ToString().ToString());
41.     sha256 = SHA256Managed.Create();
42.     hash = sha256.ComputeHash(bytes);
43.     foreach (byte x in hash)
44.     {
45.         secureHash += String.Format("{0:x2}", x);
46.     }
```

3.2 Redirect Post Request Preparation

```
1.          // Step 2: Prepare Payment Request and Send It to Redirect ASP Page (To
Send a Post Request)
2.          this.Context.Items.Add("TransactionID", transactionId);
3.          this.Context.Items.Add("MerchantID", "ANBRedirectM");
4.          this.Context.Items.Add("Amount", "2000");
5.          this.Context.Items.Add("CurrencyISOCODE", "840");
6.          this.Context.Items.Add("MessageID", "1");
7.          this.Context.Items.Add("Quantity", "1");
8.          this.Context.Items.Add("Channel", "0");
9.          this.Context.Items.Add("Language", "en");
10.         this.Context.Items.Add("ThemeID", "1000000001");
11.         // if this url is configured for the merchant it's not required, else it
is required
12.         this.Context.Items.Add("ResponseBackURL", "http://MerchantSite/RedirectPaym
entResponsePage");
13.         this.Context.Items.Add("Version", "1.0");
14.         this.Context.Items.Add("RedirectURL", "http://SmartrouteURL/SmartRoutePayme
ntWEB/SRPayMsgHandler");
15.         // set secure hash in the request
16.         this.Context.Items.Add("SecureHash", secureHash);
17.         Server.Transfer("SubmitRedirectPaymentRequest.aspx", true);
```

3.3 Redirect Post Request Submitting

```
1.  <%@ Page
Language="C#" AutoEventWireup="true" CodeFile="SubmitRedirectPaymentRequest.aspx.c
s" Inherits="vs_WebSite2_SubmitRedirectPaymentRequest" %>
2.
3.  <!DOCTYPE html>
4.
5.  <html xmlns="http://www.w3.org/1999/xhtml">
6.  <head runat="server">
7.      <title></title>
8.  </head>
9.  <body onload="javascript:document.redirectForm.submit();">
10.     <!-- STEP 3: Create ASP Page send Request -->
11.     <%
12.         // read the parameters from request
13.         String redirectURL = (String) this.Context.Items["RedirectURL"];
14.         String amount = (String) this.Context.Items["Amount"];
15.         String currencyCode = (String) this.Context.Items["CurrencyISOCODE"];
16.         String transactionID = (String) this.Context.Items["TransactionID"];
17.         String merchantID = (String) this.Context.Items["MerchantID"];
18.         String language = (String) this.Context.Items["Language"];
19.         String messageID = (String) this.Context.Items["MessageID"];
20.         String secureHash = (String) this.Context.Items["SecureHash"];
21.         String themeID = (String) this.Context.Items["ThemeID"];
22.         String responseBackURL = (String) this.Context.Items["ResponseBackURL"];
23.         String channel = (String) this.Context.Items["Channel"];
24.         String quantity = (String) this.Context.Items["Quantity"];
25.         String version = (String) this.Context.Items["Version"];
26.     %>
27.
28.     <form action="<%=redirectURL%" method="post" name="redirectForm">
29.     <input name="MerchantID" type="hidden" value="<%=merchantID%" />
30.     <input name="Amount" type="hidden" value="<%=amount%" />
```

```

31.     <input name="CurrencyISOCode" type="hidden" value="<%=currencyCode%>"/>
32.     <input name="Language" type="hidden" value="<%=language%>"/>
33.     <input name="MessageID" type="hidden" value="<%=messageID%>"/>
34.     <input name="TransactionID" type="hidden" value="<%=transactionID%>"/>
35.     <input name="ThemeID" type="hidden" value="<%=themeID%>"/>
36.     <input name="ResponseBackURL" type="hidden" value="<%=responseBackURL%>"/>
37.     <input name="Quantity" type="hidden" value="<%=quantity%>"/>
38.     <input name="Channel" type="hidden" value="<%=channel%>"/>
39.     <input name="Version" type="hidden" value="<%=version%>"/>
40.     <input name="SecureHash" type="hidden" value="<%=secureHash%>"/>
41. </form>
42. </body>
43. </html>
44.

```

3.4 Redirect Payment Response

```

1.     String SECRET_KEY = " Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours,
    Please Store Your Secret Key in safe Place (eg.database)
2.     // get All Request Parameters
3.     System.Collections.ICollection parameterNames = this.Context.Items.Keys;
4.     // store all response Parameters to generate Response Secure Hash
5.     // and get Parameters to use it later in your Code
6.     SortedDictionary<string, string> responseParameters = new SortedDictionary
    <String, String>();
7.
8.     for (int i = 0; i < parameterNames.Count; i++)
9.     {
10.         String paramName =
11.             (String)parameterNames.GetEnumerator().Current;
12.         String paramvalue = (String)this.Context.Items[paramName];
13.         responseParameters.Add(paramName, paramvalue);
14.         parameterNames.GetEnumerator().MoveNext();
15.     }
16.     // Now that we have the dictionary, order it to generate secure hash
    and compare it with the received one
17.     StringBuilder responseOrderdString = new StringBuilder();
18.     responseOrderdString.Append(SECRET_KEY);
19.     foreach (KeyValuePair<string, string> kv in responseParameters)
20.     {
21.         responseOrderdString.Append(kv.Value);
22.     }
23.     Console.WriteLine("Response Orderd String is:
    " + responseOrderdString.ToString());
24.
25.     // Generate SecureHash with SHA256
26.     SHA256 sha256;
27.     byte[] bytes, hash;
28.     string generatedsecureHash = string.Empty;
29.
30.     bytes = Encoding.UTF8.GetBytes(responseOrderdString.ToString().ToStr
    ing());
31.     sha256 = SHA256Managed.Create();
32.     hash = sha256.ComputeHash(bytes);
33.     foreach (byte x in hash)
34.     {
35.         generatedsecureHash += String.Format("{0:x2}", x);
36.     }
37.
38.

```

```

39.         // get the received secure hash from result dictionary
40.         String receivedSecurehash =
41.         responseParameters["Response.SecureHash"];
42.         if (receivedSecurehash != generatedsecureHash.ToString())
43.         {
44.             // If they are not equal then the response shall not be accepted
45.             Console.WriteLine("Received Secure Hash does not Equal generated
Secure hash");
46.         }
47.         else
48.         {
49.             // Complete the Action get other parameters from result
dictionary and do
50.             // your processes
51.             // Please refer to The Integration Manual to See The List of The
52.             // Received Parameters
53.             String status = responseParameters["Response.Status"];
54.             Console.WriteLine("Status is: " + status);
55.         }

```

4 Direct Post Payment Message

```
1. using System.Collections.Generic;
2. using System.Security.Cryptography;
3. using System.Text;
4. using System.Web;
```

4.1 Secure Hash Generation:

```
1. //Step 1: Generate Secure Hash
2. String SECRET_KEY = " Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store
   Your Secret Key in safe Place (eg.database)
3. // get All Request Parameters
4. System.Collections.ICollection parameterNames = this.Context.Items.Keys;
5. // store all response Parameters to generate Response Secure Hash
6. // and get Parameters to use it later in your Code
7. SortedDictionary<string, string> responseParameters = new SortedDictionary
   <String, String>();
8.
9. for(int i=0; i< parameterNames.Count; i++)
10. {
11.     String
   paramName = (String)parameterNames.GetEnumerator().Current;
12.     String paramvalue = (String)this.Context.Items[paramName];
13.     responseParameters.Add(paramName, paramvalue);
14.     parameterNames.GetEnumerator().MoveNext();
15. }
16. // Now that we have the sorted dictionary, order it to generate
   secure hash and compare it with the received one
17. StringBuilder responseOrderdString = new StringBuilder();
18. responseOrderdString.Append(SECRET_KEY);
19. foreach (KeyValuePair<string, string> kv in
   responseParameters)
20. {
21.     responseOrderdString.Append(kv.Value);
22. }
23. Console.WriteLine("Response Orderd String is:
   " + responseOrderdString.ToString());
24.
25. // Generate SecureHash with SHA256
26. SHA256 sha256;
27. byte[] bytes, hash;
28. string generatedsecureHash = string.Empty;
29.
30. bytes = Encoding.UTF8.GetBytes(responseOrderdString.ToString());
31. sha256 = SHA256Managed.Create();
32. hash = sha256.ComputeHash(bytes);
33. foreach (byte x in hash)
34. {
35.     generatedsecureHash += String.Format("{0:x2}", x);
36. }
37.
38.
39. // get the received secure hash from result dictionary
40. String
   receivedSecurehash = responseParameters["Response.SecureHash"];
41. if (receivedSecurehash != generatedsecureHash.ToString())
42. {
43.     // IF they are not equal then the response shall not be accepted
```

```
44.         Console.WriteLine("Received Secure Hash does not Equal generated  
Secure hash");  
45.     }  
46.     else  
47.     {  
48.         // Complete the Action get other parameters from result  
dictionary and do  
49.         // your processes  
50.         // Please refer to The Integration Manual to See The List of The  
51.         // Received Parameters  
52.         String status = responseParameters["Response.Status"];  
53.         Console.WriteLine("Status is: " + status);  
54.     }
```

4.2 Direct Post Request Preparation

```
1.         this.Context.Items.Add("TransactionID", transactionId);
2.         this.Context.Items.Add("MerchantID", "ANBRedirectM");
3.         this.Context.Items.Add("Amount", "2000");
4.         this.Context.Items.Add("CurrencyISOCODE", "840");
5.         this.Context.Items.Add("MessageID", "1");
6.         this.Context.Items.Add("Quantity", "1");
7.         this.Context.Items.Add("Channel", "0");
8.         this.Context.Items.Add("PaymentMethod", "1");
9.         this.Context.Items.Add("Language", "en");
10.        this.Context.Items.Add("ThemeID", "1000000001");
11.        this.Context.Items.Add("ResponseBackURL",
12.            "https://MerchantSite/RedirectPaymentResponsePage");// if this url
            is configured for the merchant it's not required
13.        this.Context.Items.Add("Version", "1.0");
14.        this.Context.Items.Add("RedirectURL", "http://localhost:9080/SmartRo
            utePaymentWEB/SRPayMsgHandler");
15.        // set secure hash in the request
16.        this.Context.Items.Add("SecureHash", secureHash);
17.        this.Server.Transfer("SubmitRedirectPaymentRequest.aspx", true);
```

4.3 Direct Post Request Submitting

```
1.  <%@ Page
    Language="C#" AutoEventWireup="true" CodeFile="SubmitRedirectPaymentRequest.aspx.c
    s" Inherits="vs_WebSite2_SubmitRedirectPaymentRequest" %>
2.
3.  <!DOCTYPE html>
4.
5.  <html xmlns="http://www.w3.org/1999/xhtml">
6.  <head runat="server">
7.      <title></title>
8.  </head>
9.  <body>
10.     <!-- STEP 3: Create ASP Page send Request -->
11.     <%
12.         // read the parameters from request
13.         String redirectURL = (String) this.Context.Items["RedirectURL"];
14.         String amount = (String) this.Context.Items["Amount"];
15.         String currencyCode = (String) this.Context.Items["CurrencyISOCODE"];
16.         String transactionID = (String) this.Context.Items["TransactionID"];
17.         String merchantID = (String) this.Context.Items["MerchantID"];
18.         String language = (String) this.Context.Items["Language"];
19.         String messageID = (String) this.Context.Items["MessageID"];
20.         String secureHash = (String) this.Context.Items["SecureHash"];
21.         String themeID = (String) this.Context.Items["ThemeID"];
22.         String responseBackURL = (String) this.Context.Items["ResponseBackURL"];
23.         String channel = (String) this.Context.Items["Channel"];
24.         String quantity = (String) this.Context.Items["Quantity"];
25.         String version = (String) this.Context.Items["Version"];
26.     %>
27.
28.     <form action="<%=redirectURL%>" method="post" name="redirectForm">
```

```

29.     <input name="MerchantID" type="hidden" value="<%=merchantID%>"/>
30.     <input name="Amount" type="hidden" value="<%=amount%>"/>
31.     <input name="CurrencyISOCode" type="hidden" value="<%=currencyCode%>"/>
32.     <input name="Language" type="hidden" value="<%=language%>"/>
33.     <input name="MessageID" type="hidden" value="<%=messageID%>"/>
34.     <input name="TransactionID" type="hidden" value="<%=transactionID%>"/>
35.     <input name="ThemeID" type="hidden" value="<%=themeID%>"/>
36.     <input name="ResponseBackURL" type="hidden" value="<%=responseBackURL%>"/>
37.     <input name="Quantity" type="hidden" value="<%=quantity%>"/>
38.     <input name="Channel" type="hidden" value="<%=channel%>"/>
39.     <input name="Version" type="hidden" value="<%=version%>"/>
40.     <input name="SecureHash" type="hidden" value="<%=secureHash%>"/>
41.     <label>Card Number</label>
42.     <input name="CardNumber" type="text" value=""/>
43.     <br/>
44.     <label>Card Holder Name</label>
45.     <input name="CardHolderName" type="text" value=""/>
46.     <br/>
47.     <label>Security Code</label>
48.     <input name="SecurityCode" type="text" value=""/>
49.     <br/>
50.     <label>Year Expiry Date</label>
51.     <input name="ExpiryDateYear" type="text" value=""/>
52.     <br/>
53.     <label>Month Expiry Date</label>
54.     <input name="ExpiryDateMonth" type="text" value=""/>
55.     <br/>
56.     <input type="submit" value="Proceed" />
57. </form>
58. </body>
59. </html>

```

4.4 Direct Post Payment Response

```

1.     String SECRET_KEY = " Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours,
    Please Store Your Secret Key in safe Place (eg.database)
2.     // get All Request Parameters
3.     System.Collections.ICollection parameterNames = this.Context.Items.Keys;
4.     // store all response Parameters to generate Response Secure Hash
5.     // and get Parameters to use it later in your Code
6.     SortedDictionary<string, string> responseParameters = new SortedDictionary
    <String, String>();
7.
8.     for (int i = 0; i < parameterNames.Count; i++)
9.     {
10.         String
        paramName = (String)parameterNames.GetEnumerator().Current;
11.         String paramvalue = (String)this.Context.Items[paramName];
12.         responseParameters.Add(paramName, paramvalue);
13.         parameterNames.GetEnumerator().MoveNext();
14.     }
15.     // Now that we have the dictionary, order it to generate secure hash
    and compare it with the received one
16.     StringBuilder responseOrderdString = new StringBuilder();
17.     responseOrderdString.Append(SECRET_KEY);

```

```

18.         foreach (KeyValuePair<string, string> kv in responseParameters)
19.         {
20.             responseOrderdString.Append(kv.Value);
21.         }
22.         Console.WriteLine("Response Orderd String is:
" + responseOrderdString.ToString());
23.
24.         // Generate SecureHash with SHA256
25.         SHA256 sha256;
26.         byte[] bytes, hash;
27.         string generatedsecureHash = string.Empty;
28.
29.         bytes = Encoding.UTF8.GetBytes(responseOrderdString.ToString().ToStr
ing());
30.         sha256 = SHA256Managed.Create();
31.         hash = sha256.ComputeHash(bytes);
32.         foreach (byte x in hash)
33.         {
34.             generatedsecureHash += String.Format("{0:x2}", x);
35.         }
36.
37.
38.         // get the received secure hash from result dictionary
39.         String
receivedSecurehash = responseParameters["Response.SecureHash"];
40.         if (receivedSecurehash != generatedsecureHash.ToString())
41.         {
42.             // IF they are not equal then the response shall not be accepted
43.             Console.WriteLine("Received Secure Hash does not Equal generated
Secure hash");
44.         }
45.         else
46.         {
47.             // Complete the Action get other parameters from result
dictionary and do
48.             // your processes
49.             // Please refer to The Integration Manual to See The List of The
50.             // Received Parameters
51.             String status = responseParameters["Response.Status"];
52.             Console.WriteLine("Status is: " + status);
53.         }

```

5 API MPayment

```

1. //using statements
2. using System.Collections.Generic;
3. using System.Collections.Specialized;
4. using System.Net;
5. using System.Security.Cryptography;
6. using System.Text;

```

5.1 Secure Hash Generation

```
1.      String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWiz"; // Use Yours,
      Please Store Your Secret Key in safe Place(eg.database)
2.      // put the parameters in a SortedDictionary to have the parameters to have
      them sorted alphabetically.
3.      SortedDictionary<string, string> parameters = new SortedDictionary<String,
      String>();
4.
5.      long
transactionId = (DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond); //getting
time in milliseconds
6.
7.      // fill required parameters
8.      parameters.Add("TransactionID", transactionId.ToString());
9.      parameters.Add("MerchantID", "ANBRedirectM");
10.     parameters.Add("Amount", "2000");
11.     parameters.Add("CurrencyISOCODE", "840");
12.     parameters.Add("MessageID", "8");
13.     parameters.Add("Quantity", "1");
14.     parameters.Add("Channel", "0");
15.     parameters.Add("PaymentMethod", "1");
16.     parameters.Add("ClientIPAddress", "127.0.0.1");
17.     //for Card Payment (conditional;paymentMethod=1)
18.     parameters.Add("CardNumber", "4012001045873335");
19.     parameters.Add("ExpiryDateYear", "01");
20.     parameters.Add("ExpiryDateMonth", "19");
21.     parameters.Add("SecurityCode", "123");
22.     parameters.Add("CardHolderName", "1");
23.     //for Sadad Payment (conditional;paymentMethod=2)
24.     //parameters.Add("SadadOlpId", "testSadad");
25.     //parameters.Add("mfu", "https://MerchantSite/RedirectPaymentRequestP
age");
26.     //fill some optional parameters
27.     parameters.Add("Language", "en");
28.     parameters.Add("ThemeID", "10000000001");
29.     parameters.Add("Version", "1.0");
30.     //Create an Ordered String of The Parameters SortedDictionary with
Secret Key
31.     StringBuilder orderedString = new StringBuilder();
32.     orderedString.Append(SECRET_KEY);
33.     foreach (KeyValuePair<string, string> kv in parameters)
34.     {
35.         orderedString.Append(kv.Value);
36.     }
37.     Console.WriteLine("orderdString " + orderedString);
38.
39.     // Generate SecureHash with SHA256
40.     SHA256 sha256;
41.     byte[] bytes, hash;
42.     string secureHash = string.Empty;
43.
44.     bytes = Encoding.UTF8.GetBytes(orderedString.ToString().ToString());
45.     sha256 = SHA256Managed.Create();
46.     hash = sha256.ComputeHash(bytes);
47.     foreach (byte x in hash)
48.     {
```

```
49.         secureHash += String.Format("{0:x2}", x);
50.     }
51.
52.     Console.WriteLine("Secure Hash: " + secureHash.ToString());
53.
54.
```

5.2 API MPayment Request

```
1. // if the Card was 3DS Enrolled, APIPayment Will be Divided into two requests.
2. //in the response, if the received status code was "20001" or "20002" this means
3. //that the Payment is 3DS supported, which means you need to authenticate with the
4. //Bank site, all needed parameters for 3DS in will be included in the response,
5. //and after Authentication, you will send APIApprove Request to SmartRoute.
6. //Note: The Difference between 3DS payment and none-3DS Payment, will start after
7. // getting the APIPayment response.
8.     StringBuilder requestQuery = new StringBuilder();
9.     requestQuery
10.         .Append("TransactionID").Append("=").Append(transactionId).Append("&")
11.         .Append("MerchantID").Append("=").Append("ANBRRedirectM").Append("&")
12.         .Append("Amount").Append("=").Append("2000").Append("&")
13.         .Append("CurrencyISOCode").Append("=").Append("840").Append("&")
14.         .Append("MessageID").Append("=").Append("8").Append("&")
15.         .Append("Quantity").Append("=").Append("1").Append("&")
16.         .Append("Channel").Append("=").Append("0").Append("&")
17.         .Append("PaymentMethod").Append("=").Append("1").Append("&")
18.         .Append("ClientIPAddress").Append("=").Append("127.0.0.1").Append("&")
19.         //for Card Payment (conditional.Append("&")paymentMethod=1)
20.         .Append("CardNumber").Append("=").Append("4012001045873335").Append("&")
21.         .Append("ExpiryDateYear").Append("=").Append("01").Append("&")
22.         .Append("ExpiryDateMonth").Append("=").Append("19").Append("&")
23.         .Append("SecurityCode").Append("=").Append("123").Append("&")
24.         .Append("CardHolderName").Append("=").Append("1").Append("&")
25.         //for Sadad Payment (conditional.Append("&")paymentMethod=2)
26.         // .Append("SadadOlpId").Append("=").Append("testSadad").Append("&")
27.         // .Append("mfu","https://MerchantSite/RedirectPaymentRequestPage").App
end("&")
28.     //fill some optional parameters
29.     .Append("Language").Append("=").Append("en").Append("&")
30.     .Append("ThemeID").Append("=").Append("10000000001").Append("&")
31.     .Append("Version").Append("=").Append("1.0")
32.     .Append("SecureHash").Append("=").Append(secureHash);
33.
34.     //Send the request
35.     string data = requestQuery.ToString().ToString();
36.     byte[] dataStream = Encoding.UTF8.GetBytes(data);
37.     string urlPath = "https://SR_URL";
38.     string request = urlPath;
39.     WebRequest webRequest = WebRequest.Create(request);
40.     webRequest.Method = "POST";
41.     webRequest.ContentType = "application/x-www-form-urlencoded";
42.     webRequest.ContentLength = dataStream.Length;
43.     Stream newStream = webRequest.GetRequestStream();
44.     // Send the data.
45.     newStream.Write(dataStream, 0, dataStream.Length);
46.     newStream.Close();
47.     // Get the response
48.     WebResponse webResponse = webRequest.GetResponse();
49.     String output = webResponse.ToString();
50.     //Output the response
51.     Console.WriteLine(output);
52.
53.     // this string is formatted as a "Query String" -
name=value&name2=value2.....
54.     String outputString = output.ToString();
55.
56.     // To read the output string you might want to split it
57.     // on '&' to get pairs then on '=' to get name and value
```

```

58.         // and for a better and ease on verifying secure hash you should put
        them in a SortedDictionary
59.         SortedDictionary<string, string> result = new SortedDictionary<String,
        String>();
60.         NameValueCollection qscoll = HttpUtility.ParseQueryString(output);
61.         foreach (KeyValuePair<string, string> kv in qscoll)
62.         {
63.             result.Add(kv.Key, kv.Value);
64.         }
65.
66.         // Now that we have the SortedDictionary, order it to generate secure
        hash and compare it with the received one
67.         StringBuilder responseOrderdString = new StringBuilder();
68.         responseOrderdString.Append(SECRET_KEY);
69.         foreach (KeyValuePair<string, string> kv in result)
70.         {
71.             responseOrderdString.Append(kv.Value);
72.         }
73.
74.         Console.WriteLine("Response Orderd String is
        " + responseOrderdString.ToString());
75.
76.         // Generate SecureHash with SHA256 from responseOrderedString
77.         bytes = Encoding.UTF8.GetBytes(responseOrderdString.ToString().ToStrin
        g());
78.         sha256 = SHA256Managed.Create();
79.         hash = sha256.ComputeHash(bytes);
80.         String generatedsecureHash = String.Empty;
81.         foreach (byte x in hash)
82.         {
83.             generatedsecureHash += String.Format("{0:x2}", x);
84.         }
85.
86.         // get the received secure hash from result dictionary
87.         String receivedSecurehash = result["Response.SecureHash"];
88.
89.         if (receivedSecurehash != generatedsecureHash.ToString())
90.         {
91.             //IF they are not equal then the response shall not be accepted
92.             Console.WriteLine("Received Secure Hash does not Equal generated
        Secure hash");
93.         }
94.         else
95.         {
96.             // Complete the Action get other parameters from result dictionary
            and do your processes
97.             // please refer to The Integration Manual to See the List of The
            Received Parameters
98.             String status = result["Response.Status"];
99.             Console.WriteLine("Status is :" + status);
100.            if ("20001" == status)
101.            {
102.                // if the received status code was 20001 this means that this
                transaction needs 3DS
103.                //Authentication , the parameters you need are received with
                the response too.16 | P a g e
104.                // prepare parameters to send to ASP , to Send it to 3DS in A
                Post Request
105.                String bankUrl = (String)result["Response.AcsURL"];
106.                String
                PaRequestMessage = (String)result["Response.PaRequestMessage"];
107.                // 3DS Response page ( the url that you want 3DS
                Authentication to forward the request to)
108.                String
                Merchant3DSResponseURL = "http://yoursite/your3DSResponsepage";

```

```

109.
110.         this.Context.Items.Add("ACSURL", bankUrl);
111.         this.Context.Items.Add("3DSPaMessage", PaRequestMessage);
112.         this.Context.Items.Add("TERMURL_PREFIX", Merchant3DSResponseURL);
113.     };
114.         // Verification Enrollment Result Used for 3DS payment.
115.         String veResult = (String)result["Response.ResponseVeResult"];
116.         /*****
117.         /*****
118.         /*STORE veResult IN DATABASE OR ANY SAFE PLACE TO USE IT IN
119.     APPROVE REQUEST*/
120.         /*****
121.         /*****
122.         this.Server.Transfer("RedirectTo3DS.aspx", true);
123.     }
124.     // this means that the transaction needs Sadad Authentication
125.     else if ("20002" == status)
126.     {
127.         String responseEstn = result["Response.estn"];
128.         String responseMfu = result["Response.mfu"];
129.         String
130.         responseAuthenticationUrl = result["Response.AuthenticationURL"];
131.         this.Context.Items.Add("responseEstn", responseEstn);
132.         this.Context.Items.Add("responseMfu", responseMfu);
133.         this.Context.Items.Add("responseAuthenticationUrl", responseAuthenticationUrl);
134.         this.Server.Transfer("AuthenticateSadad.aspx", true);
135.     }
136.     else
137.     {
138.         // then the card is not 3ds enrolled
139.         // this means your payment has been completed
140.         Console.WriteLine("Status is :" + status);
141.     }

```

5.3 3DS Post Request Submitting to ACS URL

```

1.     <%@ Page
2.         Language="C#" AutoEventWireup="true" CodeFile="RedirectTo3DS.aspx.cs" Inherits="vs_W
3.         ebSite2_RedirectTo3DS" %>
4.     <%
5.     //get the error code message
6.     String ACSURL="";
7.     String PaMessage="";
8.     String TermsURL = "";
9.     try{ //set the labels
10.         ACSURL = (String)this.Context.Items["ACSURL"];
11.         if (ACSURL == null) { //the errorCode is null
12.             ACSURL="";
13.         } else { //there is an error occurred
14.             PaMessage = (String)this.Context.Items["3DSPaMessage"];
15.         }
16.     }catch(Exception e){
17.         Console.WriteLine("Exception:" + e.Message);

```

```
18.     }
19.     %>
20.     <head id="Head1" runat="server">
21.     <body onload="javascript:document.ACSAutoSubmitForm.submit();">
22.     <form name="ACSAutoSubmitForm" action="<%= ACSURL%>" method="POST">
23.     <input type="hidden" name="PaReq" value="<%= PaMessage%>">
24.     <input type="hidden" name="TermUrl" value="<%=
25.     HttpContext.Current.Server.UrlEncode((String)this.Context.Items["TERMURL_PREFI
X"])%>">
26.     <input type="hidden" name="MD" value="testing">
27.     </form>
28.     </body>
29.     </html>
```

5.4 Sadad authentication Submission

```
1. <%@ Page
   Language="C#" AutoEventWireup="true" CodeFile="AuthenticateSadad.aspx.cs" Inherits="
   AuthenticateSadad" %>
2.
3. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4.
5. <html xmlns="http://www.w3.org/1999/xhtml">
6. <head id="Head1" runat="server">
7.     <%
8.         String responseEstn = (String)this.Context.Items["responseEstn"];
9.         String responseMfu = (string)this.Context.Items["responseMfu"];
10.        String ibUrl = (String) this.Context.Items["responseAuthenticationUrl"];
11.    %>
12.    <meta http-equiv="Content-Type" content="text/html; charset=windows-1256">
13.    <meta http-equiv="Pragma" content="no-cache">
14.    <meta http-equiv="Expires" content="-1">
15.    <title>3DS Authentication</title>
16. </head>
17.     <script>
18.         function redirectForm() {
19.             document.AutoSubmitForm.submit();
20.         }
21.     </script>
22. <body>
23.
24.     <form name="AutoSubmitForm" action="<%=ibUrl%>" method="GET">
25.     <input type="hidden" name="mfu" value="<%=responseMfu%>">
26.     <input type="hidden" name="estn" value="<%=responseEstn%>">
27.     </form>
28.     <input name="submit" value="Authenticate" type="button"
   onclick="redirectForm()" />
29. </body>
30. </html>
```

6 API Payment

```
1. //using statements
2. using System.Collections.Generic;
3. using System.Collections.Specialized;
4. using System.Net;
5. using System.Security.Cryptography;
6. using System.Text;
```

6.1 Secure Hash Generation

```
1. String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours,
   Please Store Your Secret Key in safe Place(e.g.database)
2. // put the parameters in a SortedDictionary to have the parameters to have
   them sorted alphabetically.
3. SortedDictionary<string, string> parameters = new SortedDictionary<String, S
   tring>();
4.
5. long
   transactionId = (DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond); //getting time
   in milliseconds
6.
7. // fill required parameters
8. parameters.Add("TransactionID", transactionId.ToString());
9. parameters.Add("MerchantID", "ANBRedirectM");
10. parameters.Add("Amount", "2000");
11. parameters.Add("CurrencyISOCODE", "840");
12. parameters.Add("MessageID", "9");
13. parameters.Add("Quantity", "1");
14. parameters.Add("Channel", "0");
15. parameters.Add("PaymentMethod", "1");
16. parameters.Add("ClientIPAddress", "127.0.0.1");
17. //for Card Payment (conditional;paymentMethod=1)
18. parameters.Add("CardNumber", "4012001045873335");
19. parameters.Add("ExpiryDateYear", "01");
20. parameters.Add("ExpiryDateMonth", "19");
21. parameters.Add("SecurityCode", "123");
22. parameters.Add("CardHolderName", "1");
23. //for Sadad Payment (conditional;paymentMethod=2)
24. //parameters.Add("SadadOlpId", "testSadad");
25. //parameters.Add("mfu", "https://MerchantSite/RedirectPaymentRequestPag
   e");
26. //fill some optional parameters
27. parameters.Add("Language", "en");
28. parameters.Add("ThemeID", "1000000001");
29. parameters.Add("Version", "1.0");
30. //Create an Ordered String of The Parameters dictionary with Secret
   Key
31. StringBuilder orderedString = new StringBuilder();
32. orderedString.Append(SECRET_KEY);
33. foreach (KeyValuePair<string, string> kv in parameters)
34. {
35.     orderedString.Append(kv.Value);
36. }
37. Console.WriteLine("orderdString " + orderedString);
38.
39. // Generate SecureHash with SHA256
40. SHA256 sha256;
41. byte[] bytes, hash;
42. string secureHash = string.Empty;
```

```

43.
44.         bytes = Encoding.UTF8.GetBytes(orderedString.ToString().ToString());
45.         sha256 = SHA256Managed.Create();
46.         hash = sha256.ComputeHash(bytes);
47.         foreach (byte x in hash)
48.         {
49.             secureHash += String.Format("{0:x2}", x);
50.         }
51.
52.         Console.WriteLine("Secure Hash: " + secureHash.ToString());

```

6.2 API Payment Request

```

1.    //in the response, if the received status code was "20002" it needs Sadad
    authentication,
2.    //and after Authentication, you will send APIApprove Request to SmartRoute.
3.        StringBuilder requestQuery = new StringBuilder();
4.        requestQuery
5.            .Append("TransactionID").Append("=").Append(transactionId).Append("&")
6.            .Append("MerchantID").Append("=").Append("ANBRedirectM").Append("&")
7.            .Append("Amount").Append("=").Append("2000").Append("&")
8.            .Append("CurrencyISOCode").Append("=").Append("840").Append("&")
9.            .Append("MessageID").Append("=").Append("9").Append("&")
10.           .Append("Quantity").Append("=").Append("1").Append("&")
11.           .Append("Channel").Append("=").Append("0").Append("&")
12.           .Append("PaymentMethod").Append("=").Append("1").Append("&")
13.           .Append("ClientIPAddress").Append("=").Append("127.0.0.1").Append("&")
14.           //for Card Payment (conditional.Append("&")paymentMethod=1)
15.           .Append("CardNumber").Append("=").Append("4012001045873335").Append("&")
16.           ")
17.           .Append("ExpiryDateYear").Append("=").Append("01").Append("&")
18.           .Append("ExpiryDateMonth").Append("=").Append("19").Append("&")
19.           .Append("SecurityCode").Append("=").Append("123").Append("&")
20.           .Append("CardHolderName").Append("=").Append("1").Append("&")
21.           .Append("SecureHash").Append("=").Append(secureHash).Append("&")
22.           //for Sadad Payment (conditional.Append("&")paymentMethod=2)
23.           //Append("SadadOlpId").Append("=").Append("testSadad").Append("&")
24.           //Append("mfu","https://MerchantSite/RedirectPaymentRequestPage").App
end("&")
25.           //fill some optional parameters
26.           .Append("Language").Append("=").Append("en").Append("&")
27.           .Append("ThemeID").Append("=").Append("1000000001").Append("&")
28.           .Append("Version").Append("=").Append("1.0")
29.           .Append("SecureHash").Append("=").Append(secureHash);
30.
31.           //Send the request
32.           string data = requestQuery.ToString().ToString();
33.           byte[] dataStream = Encoding.UTF8.GetBytes(data);
34.           string urlPath = "https://SR_URL";
35.           string request = urlPath;
36.           WebRequest webRequest = WebRequest.Create(request);
37.           webRequest.Method = "POST";
38.           webRequest.ContentType = "application/x-www-form-urlencoded";
39.           webRequest.ContentLength = dataStream.Length;
40.           Stream newStream = webRequest.GetRequestStream();
41.           // Send the data.
42.           newStream.Write(dataStream, 0, dataStream.Length);
43.           newStream.Close();
44.           // Get the response
45.           WebResponse webResponse = webRequest.GetResponse();
46.           String output = webResponse.ToString();
47.           //Output the response
48.           Console.WriteLine(output);

```

```

49.         // this string is formatted as a "Query String" -
        name=value&name2=value2.....
50.         String outputString = output.ToString();
51.
52.         // To read the output string you might want to split it
53.         // on '&' to get pairs then on '=' to get name and value
54.         // and for a better and ease on verifying secure hash you should put
        them in a SortedDictionary
55.         SortedDictionary<string, string> result = new SortedDictionary<String,
        String>();
56.         NameValueCollection qscoll = HttpUtility.ParseQueryString(output);
57.         foreach (KeyValuePair<string, string> kv in qscoll)
58.         {
59.             result.Add(kv.Key, kv.Value);
60.         }
61.
62.         // Now that we have the SortedDictionary, order it to generate secure
        hash and compare it with the received one
63.         StringBuilder responseOrderdString = new StringBuilder();
64.         responseOrderdString.Append(SECRET_KEY);
65.         foreach (KeyValuePair<string, string> kv in result)
66.         {
67.             responseOrderdString.Append(kv.Value);
68.         }
69.
70.         Console.WriteLine("Response Orderd String is
        " + responseOrderdString.ToString());
71.         // Generate SecureHash with SHA256
72.
73.         // Generate SecureHash with SHA256 from responseOrderedString
74.         bytes = Encoding.UTF8.GetBytes(responseOrderdString.ToString()
        g());
75.         sha256 = SHA256Managed.Create();
76.         hash = sha256.ComputeHash(bytes);
77.         String generatedsecureHash = String.Empty;
78.         foreach (byte x in hash)
79.         {
80.             generatedsecureHash += String.Format("{0:x2}", x);
81.         }
82.
83.         // get the received secure hash from result dictionary
84.         String receivedSecurehash = result["Response.SecureHash"];
85.         if (receivedSecurehash != generatedsecureHash.ToString())
86.         {
87.             //IF they are not equal then the response shall not be accepted
88.             Console.WriteLine("Received Secure Hash does not Equal generated
        Secure hash");
89.         }
90.         else
91.         {
92.             // complete the Action get other parameters from result dictionary
            and do your processes
93.             // please refer to The Integration Manual to See The List of The
            Received Parameters
94.             String status = result["Response.Status"];
95.             Console.WriteLine("Status is :" + status);
96.             if ("20002" == status)
97.             {
98.                 String responseEstn = result["Response.estn"];
99.                 String responseMfu = result["Response.mfu"];
100.                 String
        responseAuthenticationUrl = result["Response.AuthenticationURL"];
101.                 this.Context.Items.Add("responseEstn", responseEstn);
102.                 this.Context.Items.Add("responseMfu", responseMfu);

```

```
103.         this.Context.Items.Add("responseAuthenticationUrl", responseAu
    thenticationUrl);
104.         this.Server.Transfer("AuthenticateSadad.aspx", true);

105.     } else {
106.         // then the card is not 3ds enrolled
107.         // this means your payment has been completed
108.         Console.WriteLine("Status is :" + status);
109.
110.     }
```

6.3 Sadad authentication Submission

```
1. <%@ Page
   Language="C#" AutoEventWireup="true" CodeFile="AuthenticateSadad.aspx.cs" Inherits="
   AuthenticateSadad" %>
2.
3. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4.
5. <html xmlns="http://www.w3.org/1999/xhtml">
6. <head id="Head1" runat="server">
7. <%
8. String responseEstn = (String)this.Context.Items["responseEstn"];
9. String responseMfu = (string)this.Context.Items["responseMfu"];
10. String ibUrl = (String) this.Context.Items["responseAuthenticationUrl"];
11. %>
12. <meta http-equiv="Content-Type" content="text/html; charset=windows-1256">
13. <meta http-equiv="Pragma" content="no-cache">
14. <meta http-equiv="Expires" content="-1">
15. <title>3DS Authentication</title>
16. </head>
17. <script>
18. function redirectForm() {
19. document.AutoSubmitForm.submit();
20. }
21. </script>
22. <body>
23.
24. <form name="AutoSubmitForm" action="<%=ibUrl%>" method="GET">
25. <input type="hidden" name="mfu" value="<%=responseMfu%>">
26. <input type="hidden" name="estn" value="<%=responseEstn%>">
27. </form>
28. <input name="submit" value="Authenticate" type="button"
   onclick="redirectForm()" />
29. </body>
30. </html>
```

7 API Approve

```
1. //using statements
2. using System.Text;
3. using System.Security.Cryptography;
4. using System.Net;
5. using System.IO;
6. using System.Collections.Specialized;
7. using System.Collections.Generic;
```

7.1 Secure Hash Generation

```
1. String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please
   Store Your Secret Key in safe Place(e.g.database)
2.
3. // put the parameters in a SortedDictionary to have the parameters to have
   them sorted alphabetically.
4. SortedDictionary<string, string> parameters = new SortedDictionary<String, S
   tring>();
5.
```

```

6.         long
transactionId = (DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond); //getting time
in milliseconds
7.
8.         // fill required parameters
9.         parameters.Add("MessageID", "1");
10.        parameters.Add("TransactionID", transactionId.ToString());
11.        parameters.Add("MerchantID", "ANBRedirectM");
12.        parameters.Add("PaymentMethod", "1");
13.
14.        //it's real length is very longer than this this value is just a
sample
15.        parameters.Add("PAREs", HttpContext.Current.Server.UrlEncode("eJzFV2mT
okoW/Ssd9T4a3ewiHZQvkh2"));
16.        //in case of sadad payment
17.        //parameters.Add("estn", "Test123");
18.        //for Card Payment (conditional;paymentMethod=1)
19.        parameters.Add("CardNumber", "4012001045873335");
20.        parameters.Add("ExpiryDateYear", "01");
21.        parameters.Add("ExpiryDateMonth", "19");
22.        parameters.Add("SecurityCode", "123");
23.        parameters.Add("CardHolderName", "1");
24.
25.        //Create an Ordered String of The Parameters SortedDictionary with
Secret Key
26.        StringBuilder orderedString = new StringBuilder();
27.        orderedString.Append(SECRET_KEY);
28.        foreach (KeyValuePair<string, string> kv in parameters)
29.        {
30.            orderedString.Append(kv.Value);
31.        }
32.        Console.WriteLine("orderdString " + orderedString);
33.
34.        // Generate SecureHash with SHA256
35.        SHA256 sha256;
36.        byte[] bytes, hash;
37.        string secureHash = string.Empty;
38.
39.        bytes = Encoding.UTF8.GetBytes(orderedString.ToString().ToString());
40.        sha256 = SHA256Managed.Create();
41.        hash = sha256.ComputeHash(bytes);
42.        foreach (byte x in hash)
43.        {
44.            secureHash += String.Format("{0:x2}", x);
45.        }
46.
47.        Console.WriteLine("Secure Hash: " + secureHash.ToString());
48.

```

7.2 API Approve Request

```
1.      StringBuilder requestQuery = new StringBuilder();
2.      requestQuery
3.          .Append("TransactionID").Append("=").Append(transactionId).Append("&")
4.          .Append("MerchantID").Append("=").Append("ANBRedirectM").Append("&")
5.          .Append("MessageID").Append("=").Append("1").Append("&")
6.          .Append("PaymentMethod").Append("=").Append("1").Append("&")
7.          //for Sadad Payment (conditional.Append("&")paymentMethod=2)
8.          //.Append("estn").Append("=").Append("Test123").Append("&")
9.          //for Card Payment (conditional.Append("&")paymentMethod=1)
10.         .Append("CardNumber").Append("=").Append("4012001045873335").Append("&")
11.         ")
12.         .Append("ExpiryDateYear").Append("=").Append("01").Append("&")
13.         .Append("ExpiryDateMonth").Append("=").Append("19").Append("&")
14.         .Append("SecurityCode").Append("=").Append("123").Append("&")
15.         .Append("CardHolderName").Append("=").Append("1").Append("&")
16.         .Append("SecureHash").Append("=").Append(secureHash).Append("&")
17.         .Append("PAREs").Append("=").Append(HttpContext.Current.Server.UrlEnco
18.         de("eJzFV2mTokoW/Ssd9T4a3ewiHZQvkh2"))
19.         .Append(" & ");
20.
21.         //Send the request
22.         string data = requestQuery.ToString().ToString();
23.         byte[] dataStream = Encoding.UTF8.GetBytes(data);
24.         string urlPath = "https://SR_URL";
25.         string request = urlPath;
26.         WebRequest webRequest = WebRequest.Create(request);
27.         webRequest.Method = "POST";
28.         webRequest.ContentType = "application/x-www-form-urlencoded";
29.         webRequest.ContentLength = dataStream.Length;
30.         Stream newStream = webRequest.GetRequestStream();
31.         // Send the data.
32.         newStream.Write(dataStream, 0, dataStream.Length);
33.         newStream.Close();
34.         // Get the response
35.         WebResponse webResponse = webRequest.GetResponse();
36.         String output = webResponse.ToString();
37.         //Output the response
38.         Console.WriteLine(output);
39.
40.         // this string is formatted as a "Query String" -
41.         name=value&name2=value2.....
42.
43.         // To read the output string you might want to split it
44.         // on '&' to get pairs then on '=' to get name and value
45.         // and for a better and ease on verifying secure hash you should put
46.         them in a SortedDictionary
47.         SortedDictionary<string, string> result = new SortedDictionary<String,
48.         String>();
49.         NameValueCollection qscoll = HttpUtility.ParseQueryString(output);
50.         foreach (KeyValuePair<string, string> kv in qscoll)
51.         {
52.             result.Add(kv.Key, kv.Value);
53.         }
54.
55.         // Now that we have the SortedDictionary, order it to generate secure
56.         hash and compare it with the received one
57.         StringBuilder responseOrderdString = new StringBuilder();
58.         responseOrderdString.Append(SECRET_KEY);
59.         foreach (KeyValuePair<string, string> kv in result)
60.         {
```

```

56.         responseOrderdString.Append(kv.Value);
57.     }
58.
59.     Console.WriteLine("Response Orderd String is
" + responseOrderdString.ToString());
60.
61.     // Generate SecureHash with SHA256 from responseOrderedString
62.     bytes = Encoding.UTF8.GetBytes(responseOrderdString.ToString());
63.     sha256 = SHA256Managed.Create();
64.     hash = sha256.ComputeHash(bytes);
65.     String generatedsecureHash = String.Empty;
66.     foreach (byte x in hash)
67.     {
68.         generatedsecureHash += String.Format("{0:x2}", x);
69.     }
70.
71.     // get the received secure hash from result sorted dictionary
72.     String receivedSecurehash = result["Response.SecureHash"];
73.     if (generatedsecureHash.ToString() != receivedSecurehash)
74.     {
75.         //IF they are not equal then the response shall not be accepted
76.         Console.WriteLine("Received Secure Hash does not Equal generated
Secure hash");
77.     }
78.     else
79.     {
80.         // Complete the Action get other parameters from result sorted
dictionary and do your processes
81.         // please refer to The Integration Manual to See The List of The
Received Parameters
82.         String status = result["Response.Status"];
83.         Console.WriteLine("Status is : " + status);
84.     }

```

8 Inquiry Message

```

1. //using statements
2. using System.Collections.Generic;
3. using System.Collections.Specialized;
4. using System.Net;
5. using System.Security.Cryptography;
6. using System.Text;

```

8.1 Secure Hash Generation

```

1.     String SECRET_KEY = "Y2FkMTdlOWZiMzJmZjY4ZGFkMzhkMWIz"; // Use Yours,
Please Store Your Secret Key in safe Place (e.g.database)
2.     // put the parameters in a SortedDictionary to have the parameters to have
them sorted alphabetically.
3.     SortedDictionary<string, string> parameters = new SortedDictionary<String, S
tring>();
4.
5.     // fill required parameters
6.     parameters.Add("MessageID", "2");
7.     parameters.Add("OriginalTransactionID", "1440954863817");

```

```

8.         parameters.Add("MerchantID", "ANBRedirectM");
9.         parameters.Add("Version", "1.0");
10.
11.         //Create an ordered String of The Parameters SortedDictionary with Secret Key
12.         StringBuilder orderedString = new StringBuilder();
13.         orderedString.Append(SECRET_KEY);
14.         foreach (KeyValuePair<string, string> kv in parameters)
15.         {
16.             orderedString.Append(kv.Value);
17.         }
18.         String orderTest = orderedString.ToString().ToString();
19.         Console.WriteLine("orderdString " + orderTest);
20.
21.         // Generate SecureHash with SHA256
22.         SHA256 sha256;
23.         byte[] bytes, hash;
24.         string secureHash = string.Empty;
25.
26.         bytes = Encoding.UTF8.GetBytes(orderedString.ToString().ToString());
27.         sha256 = SHA256Managed.Create();
28.         hash = sha256.ComputeHash(bytes);
29.         foreach (byte x in hash)
30.         {
31.             secureHash += String.Format("{0:x2}", x);
32.         }
33.
34.         Console.WriteLine("Secure Hash: " + secureHash.ToString());

```

8.2 Inquiry Request

```

1.         StringBuilder requestQuery = new StringBuilder();
2.         requestQuery
3.         .Append("OriginalTransactionID").Append("=").Append(1440954863817).Append("&
4.         ")
5.         .Append("MerchantID").Append("=").Append("ANBRedirectM").Append("&")
6.         .Append("MessageID").Append("=").Append("2").Append("&")
7.         .Append("Version").Append("=").Append("1.0").Append("&")
8.         .Append("SecureHash").Append("=").Append(secureHash).Append("&");
9.
10.        //Send the request
11.        string data = requestQuery.ToString().ToString();
12.        byte[] dataStream = Encoding.UTF8.GetBytes(data);
13.        string urlPath = "https://SR_URL";
14.        string request = urlPath;
15.        WebRequest webRequest = WebRequest.Create(request);
16.        webRequest.Method = "POST";
17.        webRequest.ContentType = "application/x-www-form-urlencoded";
18.        webRequest.ContentLength = dataStream.Length;
19.        Stream newStream = webRequest.GetRequestStream();
20.        // Send the data.
21.        newStream.Write(dataStream, 0, dataStream.Length);
22.        newStream.Close();
23.        WebResponse webResponse = webRequest.GetResponse();
24.        String output = null;
25.
26.        using (Stream stream = webResponse.GetResponseStream())
27.        {
28.            StreamReader reader = new StreamReader(stream, Encoding.UTF8);
29.            output = reader.ReadToEnd();
30.        }

```

```

30.
31.         //Output the response
32.         Console.WriteLine(output);
33.         // this string is formatted as a "Query String" -
        name=value&name2=value2.....
34.         String outputString = output.ToString();
35.
36.         // To read the output string you might want to split it
37.         // on '&' to get pairs then on '=' to get name and value
38.         // and for a better and ease on verifying secure hash you should put
        them in a SortedDictionary
39.         SortedDictionary<string, string> result = new SortedDictionary<String,
        String>();
40.         NameValueCollection qscoll = HttpUtility.ParseQueryString(output);
41.         foreach (KeyValuePair<string, string> kv in qscoll)
42.         {
43.             result.Add(kv.Key, kv.Value);
44.         }
45.
46.         // Now that we have the sorted dictionary, order it to generate secure
        hash and compare it with the received one
47.         StringBuilder responseOrderdString = new StringBuilder();
48.         responseOrderdString.Append(SECRET_KEY);
49.         foreach (KeyValuePair<string, string> kv in result)
50.         {
51.             responseOrderdString.Append(kv.Value);
52.         }
53.         Console.WriteLine("Response Orderd String is
        " + responseOrderdString.ToString());
54.
55.         // Generate SecureHash with SHA256
56.         SHA256 sha256Generated;
57.         byte[] bytesGenerated, hashGenerated;
58.
59.         bytesGenerated = Encoding.UTF8.GetBytes(responseOrderdString.ToString(
        ).ToString());
60.
61.         sha256Generated = SHA256Managed.Create();
62.         hashGenerated = sha256.ComputeHash(bytesGenerated);
63.         String generatedsecureHash = String.Empty;
64.         foreach (byte x in hashGenerated)
65.         {
66.             generatedsecureHash += String.Format("{0:x2}", x);
67.         }
68.
69.         // get the received secure hash from result SortedDictionary
70.         String receivedSecurehash = result["Response.SecureHash"];
71.         if (generatedsecureHash.ToString() != receivedSecurehash)
72.         {
73.             //IF they are not equal then the response shall not be accepted
74.             Console.WriteLine("Received Secure Hash does not Equal generated
        Secure hash");
75.         }
76.         else
77.         {
78.             // Complete the Action get other parameters from result dictionary
        and do your processes
79.             // Please refer to The Integration Manual to See The List of The
        Received Parameters
80.             String status = result["Response.Status"];
81.             Console.WriteLine("Status is :" + status);
82.         }

```


9 Refund Message

This message is intended to perform a transaction refund. It's based on the Back-To-Back communication model as described in [Communication Model](#) section.

```
1. //using statements
2. using System.Collections.Generic;
3. using System.Collections.Specialized;
4. using System.Net;
5. using System.Security.Cryptography;
6. using System.Text;
```

9.1 Secure Hash Generation

```
1.     String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours,
    Please Store Your Secret Key in safe Place(e.g.database)
2.     // put the parameters in a SortedDictionary to have the parameters to have
    them sorted alphabetically.
3.     SortedDictionary<string, string> dictionary = new SortedDictionary<String, S
tring>();
4.
5.     long transactionId = (DateTime.Now.Ticks / TimeSpan.TicksPerMillisecond);
6.     // fill required parameters
7.     dictionary.Add("MessageID", "4");
8.     dictionary.Add("TransactionID", transactionId.ToString());
9.     dictionary.Add("OriginalTransactionID", "1440954863817");
10.    dictionary.Add("MerchantID", "ANBRedirectM");
11.    dictionary.Add("Amount", "2000");
12.    dictionary.Add("CurrencyISOCODE", "840");
13.    dictionary.Add("Version", "1.0");
14.
15.    //Create an Ordered String of The Parameters Dictionary with Secret Key
16.    StringBuilder stringBuilder = new StringBuilder();
17.    stringBuilder.Append(SECRET_KEY);
18.
19.    foreach (KeyValuePair<string, string> kv in dictionary)
20.    {
21.        stringBuilder.Append(kv.Value);
22.    }
23.    Console.WriteLine("ordered string: " + stringBuilder.ToString().ToString());
24.    // Generate SecureHash with SHA256
25.    SHA256 sha256;
26.    byte[] bytes, hash;
27.    StringBuilder secureHash;
28.
29.    bytes = Encoding.UTF8.GetBytes(stringBuilder.ToString().ToString());
30.
31.    sha256 = SHA256Managed.Create();
32.    hash = sha256.ComputeHash(bytes);
33.    secureHash = new StringBuilder(hash.Length * 2);
34.
35.    Console.WriteLine("Secure Hash: " + stringBuilder.ToString().ToString());
```

9.2 Refund Request

```
1.     StringBuilder requestQuery = new StringBuilder();
2.     requestQuery
3.         .Append("TransactionID").Append("=").Append(transactionId).Append("&")
4.         .Append("MerchantID").Append("=").Append("ANBRedirectM").Append("&")
5.         .Append("MessageID").Append("=").Append("4").Append("&")
6.         .Append("Amount").Append("=").Append("2000").Append("&")
7.         .Append("OriginalTransactionID").Append("=").Append("1440954863817").Append(
            "&")
8.         .Append("CurrencyISOCODE").Append("=").Append("840").Append("&")
9.         .Append("SecureHash").Append("=").Append(secureHash).Append("&")
10.        .Append("Version").Append("=").Append("1.0").Append("&");
11.
12.        //Send the request
13.        string data = requestQuery.ToString().ToString();
14.        byte[] dataStream = Encoding.UTF8.GetBytes(data);
15.        string urlPath = "https://SR_URL";
16.        string request = urlPath;
17.        WebRequest webRequest = WebRequest.Create(request);
18.        webRequest.Method = "POST";
19.        webRequest.ContentType = "application/x-www-form-urlencoded";
20.        webRequest.ContentLength = dataStream.Length;
21.        Stream newStream = webRequest.GetRequestStream();
22.        // Send the data.
23.        newStream.Write(dataStream, 0, dataStream.Length);
24.        newStream.Close();
25.        WebResponse webResponse = webRequest.GetResponse();
26.        String output = webResponse.ToString();
27.        //Output the response
28.        Console.WriteLine(output);
29.        // this string is formatted as a "Query String" -
        name=value&name2=value2.....
30.
31.        SortedDictionary<string, string> dictionaryOutput = new SortedDictionary<String, String>();
32.        // To read the output string you might want to split it
33.        // on '&' to get pairs then on '=' to get name and value
34.        // and for a better and ease on verifying secure hash you should put
        them in a SortedDictionary
35.        NameValueCollection qscoll = HttpUtility.ParseQueryString(output);
36.        foreach (KeyValuePair<string, string> kv in qscoll)
37.        {
38.            dictionaryOutput.Add(kv.Key, kv.Value);
39.        }
40.
41.        // Now that we have the sorted dictionary, order it to generate secure
        hash and compare it with the received one
42.        StringBuilder responseOrderdString = new StringBuilder();
43.        responseOrderdString.Append(SECRET_KEY);
44.        foreach (KeyValuePair<string, string> kv in dictionaryOutput)
45.        {
46.            responseOrderdString.Append(kv.Value);
47.        }
48.        Console.WriteLine("Response Orderd String is
        " + responseOrderdString);
49.        string generatedsecureHash = string.Empty;
50.
51.        bytes = Encoding.UTF8.GetBytes(responseOrderdString.ToString().ToString());
52.
53.        sha256 = SHA256Managed.Create();
54.        hash = sha256.ComputeHash(bytes);
55.        foreach (byte x in hash)
56.        {
```

```

56.         generatedsecureHash += String.Format("{0:x2}", x);
57.     }
58.
59.     // get the received secure hash from result dictionary
60.     String receivedSecurehash = dictionaryOutput["Response.SecureHash"];
61.     if (!receivedSecurehash.Equals(generatedsecureHash))
62.     {
63.         //IF they are not equal then the response shall not be accepted
64.         Console.WriteLine("Received Secure Hash does not Equal generated
Secure hash");
65.     }
66.     else
67.     {
68.         // Complete the Action get other parameters from result dictionary
and do your processes
69.         // please refer to The Integration Manual to See The List of The
Received Parameters
70.         String status = dictionaryOutput["Response.Status"];
71.         Console.WriteLine("Status is :" + status);
72.     }

```