

SmartRoute Sample Code (Java)

version 1.0

STS PayOne

27 April 2017

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	7
3.4 REDIRECT PAYMENT RESPONSE	7
4 DIRECT POST PAYMENT MESSAGE	9
4.1 SECURE HASH GENERATION	9
4.2 DIRECT POST REQUEST PREPARATION	10
4.3 DIRECT POST REQUEST SUBMITTING	11
4.4 DIRECT POST PAYMENT RESPONSE	12
5 API PAYMENT	13
5.1 SECURE HASH GENERATION	13
5.2 API PAYMENT REQUEST	14
5.3 3DS POST REQUEST SUBMITTING TO ACS URL	17
5.4 SADAD AUTHENTICATION SUBMISSION	18
6 API MPAYMENT	19
6.1 SECURE HASH GENERATION	19
6.2 API MPAYMENT REQUEST	20
6.3 SADAD AUTHENTICATION SUBMISSION	23
7 API APPROVE	23
7.1 SECURE HASH GENERATION	23
7.2 API APPROVE REQUEST	25
8 INQUIRY MESSAGE	27
8.1 SECURE HASH GENERATION	27
8.2 INQUIRY REQUEST	28
9 REFUND MESSAGE	30
9.1 SECURE HASH GENERATION	30
9.2 REFUND REQUEST	31

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

```
// import statements
import org.apache.commons.codec.digest.DigestUtils;
```

3.1 Secure Hash Generation

```
//Step 1: Generate Secure Hash
String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store Your
Secret Key in safe Place (eg. database)

// put the parameters in a TreeMap to have the parameters to have them sorted
alphabetically.
Map <String,String> parameters = new TreeMap<String,String> ();

String transactionId = String.valueOf(System.currentTimeMillis());
// fill required parameters
parameters.put("TransactionID", transactionId);
parameters.put("MerchantID", "ANBRedirectM");
parameters.put("Amount", "2000");
parameters.put("CurrencyISOCode", "840");
parameters.put("MessageID", "1");
parameters.put("Quantity", "1");
parameters.put("Channel", "0");

//fill some optional parameters
parameters.put("Language", "en");
parameters.put("ThemeID", "1000000001");
parameters.put("ResponseBackURL", "https://MerchantSite/RedirectPaymentResponsePage");// if
this url is configured for the merchant it's not required
parameters.put("Version", "1.0");

//Create an Ordered String of The Parameters Map with Secret Key
StringBuilder orderedString = new StringBuilder();
orderedString.append(SECRET_KEY);
for (String treeMapKey : parameters.keySet()) {
    orderedString.append(parameters.get(treeMapKey));
}

System.out.println("orderdString: " + orderedString);
// Generate SecureHash with SHA256
// Using DigestUtils from appache.commons.codes.jar Library
String secureHash = new String(DigestUtils.sha256Hex(orderedString.toString()).getBytes());
```

3.2 Redirect Post Request Preparation

```
// Step 2: Prepare Payment Request and Send It to Redirect JSP Page (To Send a Post Request)

request.setAttribute("TransactionID", transactionId);
request.setAttribute("MerchantID", "ANBRedirectM");
request.setAttribute("Amount", "2000");
request.setAttribute("CurrencyISOCode", "840");
request.setAttribute("MessageID", "1");
request.setAttribute("Quantity", "1");
request.setAttribute("Channel", "0");

request.setAttribute("Language", "en");
request.setAttribute("ThemeID", "1000000001");
// if this url is configured for the merchant it's not required, else it is required
request.setAttribute("ResponseBackURL", "http://MerchantSite/RedirectPaymentResponsePage");
request.setAttribute("Version", "1.0");

request.setAttribute("RedirectURL", "http://SmartrouteURL/SmartRoutePaymentWEB/SRPayMsgHandler");

// set secure hash in the request
request.setAttribute("SecureHash", secureHash);

request.getRequestDispatcher(response.encodeURL("SubmitRedirectPaymentRequest.jsp")).forward(request, response);
```

3.3 Redirect Post Request Submitting

```
<!-- STEP 3: Create JSP Page send Request -->

<%
    // read the parameters from request
    String redirectURL = (String) request.getAttribute("RedirectURL");
    String amount = (String) request.getAttribute("Amount");
    String currencyCode = (String) request.getAttribute("CurrencyISOCode");
    String transactionID = (String) request.getAttribute("TransactionID");
    String merchantID = (String) request.getAttribute("MerchantID");
    String language = (String) request.getAttribute("Language");
    String messageID = (String) request.getAttribute("MessageID");
    String secureHash = (String) request.getAttribute("SecureHash");
    String themeID = (String) request.getAttribute("ThemeID");
    String responseBackURL = (String) request.getAttribute("ResponseBackURL");
    String channel = (String) request.getAttribute("Channel");
    String quantity = (String) request.getAttribute("Quantity");
    String version = (String) request.getAttribute("Version");
%>
<html>

<body onload="javascript:document.redirectForm.submit();">
<form action="<%=redirectURL%>" method="post" name="redirectForm">
    <input name="MerchantID" type="hidden" value="<%=merchantID%>" />
    <input name="Amount" type="hidden" value="<%=amount%>" />
    <input name="CurrencyISOCode" type="hidden" value="<%=currencyCode%>" />
    <input name="Language" type="hidden" value="<%=language%>" />
    <input name="MessageID" type="hidden" value="<%=messageID%>" />
    <input name="TransactionID" type="hidden" value="<%=transactionID%>" />
    <input name="ThemeID" type="hidden" value="<%=themeID%>" />
    <input name="ResponseBackURL" type="hidden" value="<%=responseBackURL%>" />
    <input name="Quantity" type="hidden" value="<%=quantity%>" />
    <input name="Channel" type="hidden" value="<%=channel%>" />
    <input name="Version" type="hidden" value="<%=version%>" />
    <input name="SecureHash" type="hidden" value="<%=secureHash%>" />
</form>
</body>
</html>
```

3.4 Redirect Payment Response

```
String SECRET_KEY = " Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store Your
Secret Key in safe Place (eg. database)

// get All Request Parameters
Enumeration<String> parameterNames = request.getParameterNames();

// store all response Parameters to generate Response Secure Hash
// and get Parameters to use it later in your Code
Map<String, String> responseParameters = new TreeMap<String, String>();
while (parameterNames.hasMoreElements()) {
    String paramName = parameterNames.nextElement();
    String paramvalue = request.getParameter(paramName);
    responseParameters.put(paramName, paramvalue);
}
// Now that we have the map, order it to generate secure hash and compare it with the
received one

StringBuilder responseOrderdString = new StringBuilder();
```

```

responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : responseParameters.keySet()) {
    responseOrderdString.append(responseParameters.get(treeMapKey));
}

System.out.println("Response Ordered String is: " + responseOrderdString.toString());

// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());

// get the received secure hash from result map
String receivedSecurehash = responseParameters.get("Response.SecureHash");

if (!receivedSecurehash.equals(generatedsecureHash)) {

    // IF they are not equal then the response shall not be accepted
    System.out.println("Received Secure Hash does not Equal Generated Secure hash");
} else {

    // Complete the Action get other parameters from result map and do
    // your processes
    // Please refer to The Integration Manual to see the List of The
    // Received Parameters
    String status = responseParameters.get("Response.Status");
    System.out.println("Status is: " + status);
}

```

4 Direct Post Payment Message

```
// import statements
import org.apache.commons.codec.digest.DigestUtils;
```

4.1 Secure Hash Generation

```
//Step 1: Generate Secure Hash
String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store Your
Secret Key in safe Place (e.g. database)

// put the parameters in a TreeMap to have the parameters to have them sorted
alphabetically.
Map <String,String> parameters = new TreeMap<String,String> ();

String transactionId = String.valueOf(System.currentTimeMillis());

// fill required parameters
parameters.put("TransactionID", transactionId);
parameters.put("MerchantID", "ANBRedirectM");
parameters.put("Amount", "2000");
parameters.put("CurrencyISOCode", "840");
parameters.put("MessageID", "1");
parameters.put("Quantity", "1");
parameters.put("Channel", "0");
parameters.put("PaymentMethod", "1");

//for Card Payment (conditional;paymentMethod=1)
parameters.put("CardNumber", "4012001045873335");
parameters.put("ExpiryDateYear", "01");
parameters.put("ExpiryDateMonth", "19");
parameters.put("SecurityCode", "123");
parameters.put("CardHolderName", "1");

//for Sadad Payment (conditional; paymentMethod=2)
//parameters.put("SadadOlpId", "testSadad");

//fill some optional parameters
parameters.put("Language", "en");
parameters.put("ThemeID", "1000000001");
parameters.put("ResponseBackURL", "https://MerchantSite/RedirectPaymentResponsePage");// if
this URL is configured for the merchant it's not required
parameters.put("Version", "1.0");

//Create an ordered String of The Parameters Map with Secret Key
StringBuilder orderedString = new StringBuilder();
orderedString.append(SECRET_KEY);
for (String treeMapKey : parameters.keySet()) {
    orderedString.append(parameters.get(treeMapKey));
}

System.out.println("orderdString: " + orderedString);
// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String secureHash = new String(DigestUtils.sha256Hex(orderedString.toString()).getBytes());
```

4.2 Direct Post Request Preparation

```
// Step 2: Prepare Payment Request and Send It to Redirect JSP Page (To Send a Post Request)
request.setAttribute("TransactionID",transactionId);
request.setAttribute("MerchantID", "ANBRedirectM");
request.setAttribute("Amount", "2000");
request.setAttribute("CurrencyISOCode", "840");
request.setAttribute("MessageID", "1");
request.setAttribute("Quantity", "1");
request.setAttribute("Channel", "0");

request.setAttribute("PaymentMethod", "1");

//for Card Payment (conditional;paymentMethod=1)
request.setAttribute("CardNumber", "4012001045873335");
request.setAttribute("ExpiryDateYear", "01");
request.setAttribute("ExpiryDateMonth", "19");
request.setAttribute("SecurityCode", "123");
request.setAttribute("CardHolderName", "1");

//for Sadad Payment (conditional;paymentMethod=2)
//request.setAttribute("SadadOlpId", "testSadad");

request.setAttribute("Language", "en");
request.setAttribute("ThemeID", "1000000001");
request.setAttribute("ResponseBackURL",
"https://MerchantSite/RedirectPaymentResponsePage");// if this url is configured for the
merchant it's not required
request.setAttribute("Version", "1.0");
request.setAttribute("RedirectURL",
"http://localhost:9080/SmartRoutePaymentWEB/SRPayMsgHandler");

// set secure hash in the request
request.setAttribute("SecureHash", secureHash);

request.getRequestDispatcher(response.encodeURL("SubmitRedirectPaymentRequest.jsp")).forward(
request, response);
```

4.3 Direct Post Request Submitting

```
<!-- STEP 3: Create JSP Page send Request -->

<%
    // read the parameters from request
    String redirectURL = (String) request.getAttribute("RedirectURL");
    String amount = (String) request.getAttribute("Amount");
    String currencyCode = (String) request.getAttribute("CurrencyISOCODE");
    String transactionID = (String) request.getAttribute("TransactionID");
    String merchantID = (String) request.getAttribute("MerchantID");
    String language = (String) request.getAttribute("Language");
    String messageID = (String) request.getAttribute("MessageID");
    String secureHash = (String) request.getAttribute("SecureHash");
    String themeID = (String) request.getAttribute("ThemeID");
    String responseBackURL = (String) request.getAttribute("ResponseBackURL");
    String channel = (String) request.getAttribute("Channel");
    String quantity = (String) request.getAttribute("Quantity");
    String version = (String) request.getAttribute("Version");

    String paymentMethod = (String) request.getAttribute("PaymentMethod");
    String cardNumber = (String) request.getAttribute("CardNumber");
    String cardHolderName = (String) request.getAttribute("CardHolderName");
    String securityCode = (String) request.getAttribute("SecurityCode");
    String expiryDateYear = (String) request.getAttribute("ExpiryDateYear");
    String expiryDateMonth = (String) request.getAttribute("ExpiryDateMonth");
    String expiryDateMonth = (String) request.getAttribute("ExpiryDateMonth");
    String sadadOlpId = (String) request.getAttribute("SadadOlpId");
%>
<html>

<body onload="javascript:document.redirectForm.submit();">
<form action="<%=redirectURL%>" method="post" name="redirectForm">
    <input name="MerchantID" type="hidden" value="<%=merchantID%>" />
    <input name="Amount" type="hidden" value="<%=amount%>" />
    <input name="CurrencyISOCODE" type="hidden" value="<%=currencyCode%>" />
    <input name="Language" type="hidden" value="<%=language%>" />
    <input name="MessageID" type="hidden" value="<%=messageID%>" />
    <input name="TransactionID" type="hidden" value="<%=transactionID%>" />
    <input name="ThemeID" type="hidden" value="<%=themeID%>" />
    <input name="ResponseBackURL" type="hidden" value="<%=responseBackURL%>" />
    <input name="Quantity" type="hidden" value="<%=quantity%>" />
    <input name="Channel" type="hidden" value="<%=channel%>" />
    <input name="Version" type="hidden" value="<%=version%>" />
    <input name="PaymentMethod" type="hidden" value="<%=paymentMethod%>" />
    <input name="CardNumber" type="hidden" value="<%=cardNumber%>" />
    <input name="CardHolderName" type="hidden" value="<%=cardHolderName%>" />
    <input name="SecurityCode" type="hidden" value="<%=securityCode%>" />
    <input name="SadadOlpId" type="hidden" value="<%=sadadOlpId%>" />
    <input name="ExpiryDateYear" type="hidden" value="<%=expiryDateYear%>" />
    <input name="ExpiryDateMonth" type="hidden" value="<%=expiryDateMonth%>" />
    <input name="SecureHash" type="hidden" value="<%=secureHash%>" />
</form>
</body>
</html>
```

4.4 Direct Post Payment Response

```
String SECRET_KEY = " Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz";// Use Yours, Please Store Your
Secret Key in safe Place (eg. database)

// get All Request Parameters
Enumeration<String> parameterNames = request.getParameterNames();

// store all response Parameters to generate Response Secure Hash
// and get Parameters to use it later in your Code
Map<String, String> responseParameters = new TreeMap<String, String>();
while (parameterNames.hasMoreElements()) {
    String paramName = parameterNames.nextElement();
    String paramvalue = request.getParameter(paramName);
    responseParameters.put(paramName, paramvalue);
}
// Now that we have the map, order it to generate secure hash and compare it with the
received one

StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : responseParameters.keySet()) {
    responseOrderdString.append(responseParameters.get(treeMapKey));
}

System.out.println("Response Orderd String is: " + responseOrderdString.toString());

// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());

// get the received secure hash from result map
String receivedSecurehash = responseParameters.get("Response.SecureHash");

if (!receivedSecurehash.equals(generatedsecureHash)) {

    // IF they are not equal then the response shall not be accepted
    System.out.println("Received Secure Hash does not Equal generated Secure hash");
} else {

    // Complete the Action get other parameters from result map and do
    // your processes
    // Please refer to The Integration Manual to See The List of The
    // Received Parameters
    String status = responseParameters.get("Response.Status");
    System.out.println("Status is: " + status);
}
```

5 API Payment

```
// import statements
import org.apache.commons.codec.digest.DigestUtils;
```

5.1 Secure Hash Generation

```
//Step 1: Generate Secure Hash
String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store Your
Secret Key in safe Place(eg. database)

// put the parameters in a TreeMap to have the parameters to have them sorted
alphabetically.
Map <String,String> parameters = new TreeMap<String,String> ();
String transactionId=String.valueOf(System.currentTimeMillis());

// fill required parameters
parameters.put("TransactionID", transactionId);
parameters.put("MerchantID", "ANBRedirectM");
parameters.put("Amount", "2000");
parameters.put("CurrencyISOCODE", "840");
parameters.put("MessageID", "8");
parameters.put("Quantity", "1");
parameters.put("Channel", "0");
parameters.put("PaymentMethod", "1");
parameters.put("ClientIPAddress", "127.0.0.1");

//for Card Payment (conditional;paymentMethod=1)
parameters.put("CardNumber", "4012001045873335");
parameters.put("ExpiryDateYear", "01");
parameters.put("ExpiryDateMonth", "19");
parameters.put("SecurityCode", "123");
parameters.put("CardHolderName", "1");
//for Sadad Payment (conditional;paymentMethod=2)
//parameters.put("SadadOlpId", "testSadad");
//parameters.put("mfu", "https://MerchantSite/RedirectPaymentRequestPage");

//fill some optional parameters
parameters.put("Language", "en");
parameters.put("ThemeID", "1000000001");
parameters.put("Version", "1.0");

//Create an Ordered String of The Parameters Map with Secret Key
StringBuilder orderedString = new StringBuilder();
orderedString.append(SECRET_KEY);
for (String treeMapKey : parameters.keySet()) {
    orderedString.append(parameters.get(treeMapKey));
}

System.out.println("orderdString "+orderedString);
// Generate SecureHash with SHA256
// Using DigestUtils from appache.commons.codes.jar Library
String secureHash = new String(DigestUtils.sha256Hex(orderedString.toString()).getBytes());
```

5.2 API Payment Request

// if the Card was 3DS Enrolled, APIPayment Will be Divided into two requests.
//in the response, if the received status code was "20001" or "20002" this means
//that the Payment is 3DS supported, which means you need to authenticate with the
//Bank site, all needed parameters for 3DS in will be included in the response,
//and after Authentication, you will send APIApprove Request to SmartRoute.
//Note: The Difference between 3DS payment and none-3DS Payment, will start after
getting the APIPayment response.

```
StringBuffer requestQuery = new StringBuffer();

requestQuery
.append("TransactionID").append("=").append(transactionId).append("&")
.append("MerchantID").append("=").append("ANBRedirectM").append("&")
.append("Amount").append("=").append("2000").append("&")
.append("CurrencyISOCode").append("=").append("840").append("&")
.append("MessageID").append("=").append("8").append("&")
.append("Quantity").append("=").append("1").append("&")
.append("Channel").append("=").append("0").append("&")
.append("PaymentMethod").append("=").append("1").append("&")
.append("ClientIPAddress").append("=").append("127.0.0.1").append("&")

//for Card Payment (conditional.append("&")paymentMethod=1)
.append("CardNumber").append("=").append("4012001045873335").append("&")
.append("ExpiryDateYear").append("=").append("01").append("&")
.append("ExpiryDateMonth").append("=").append("19").append("&")
.append("SecurityCode").append("=").append("123").append("&")
.append("CardHolderName").append("=").append("1").append("&")

//for Sadad Payment (conditional.append("&")paymentMethod=2)
//.append("SadadOlpId").append("=").append("testSadad").append("&")
//.append("mfu", "https://MerchantSite/RedirectPaymentRequestPage").append("&")

//fill some optional parameters
.append("Language").append("=").append("en").append("&")
.append("ThemeID").append("=").append("1000000001").append("&")
.append("Version").append("=").append("1.0")
.append("SecureHash").append("=").append(secureHash);

//Send the request
URL url = new URL("https://SR_URL");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");

//write parameters
writer.write(requestQuery.toString());
writer.flush();

// Get the response
StringBuffer output = new StringBuffer();
BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));
String line;
while ((line = reader.readLine()) != null) {
    output.append(line);
}
writer.close();
reader.close();
```

```

//Output the response
System.out.println(output.toString());

// this string is formatted as a "Query String" - name=value&name2=value2.....
String outputString=output.toString();

// To read the output string you might want to split it
// on '&' to get pairs then on '=' to get name and value
// and for a better and ease on verifying secure hash you should put them in a TreeMap
String [] pairs=outputString.split("&");

Map<String,String> result=new TreeMap<String,String>();

// now we have separated the pairs from each other {"name1=value1","name2=value2",....}
for(String pair:pairs){

    // now we have separated the pair to {"name","value"}
    String[] nameValue=pair.split("=");

    String name=nameValue[0];//first element is the name
    String value=nameValue[1];//second element is the value

    // put the pair in the result map
    result.put(name,value);
}

// Now that we have the map, order it to generate secure hash and compare it with the
received one

StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : result.keySet()) {
    responseOrderdString.append(result.get(treeMapKey));
}

System.out.println("Response Orderd String is " + responseOrderdString.toString());

// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());

// get the received secure hash from result map
String receivedSecurehash=result.get("Response.SecureHash");

if(!receivedSecurehash.equals(generatedsecureHash)){

    //IF they are not equal then the response shall not be accepted
    System.out.println("Received Secure Hash does not Equal generated Secure hash");
}
else{
    // Complete the Action get other parameters from result map and do your processes
    // please refer to The Integration Manual to See the List of The Received Parameters
    String status=result.get("Response.Status");
    System.out.println("Status is :"+ status);

    if ("20001".equalsIgnoreCase(status)) {
        // if the received status code was 20001 this means that this transaction needs 3DS
        //Authentication , the parameters you need are received with the response too.
    }
}

```

```
// prepare parameters to send to JSP ,to Send it to 3DS in A Post Request
String bankUrl= (String)result.get("Response.AcsURL");
String PaRequestMessage= (String)result.get("Response.PaRequestMessage");

// 3DS Response page ( the url that you want 3DS Authentication to forward the
request to)
String Merchant3DSResponseURL= "http://yoursite/your3DSResponsepage";

request.setAttribute("ACSURL", bankUrl);
request.setAttribute("3DSPaMessage", PaRequestMessage);
request.setAttribute("TERMURL_PREFIX", Merchant3DSResponseURL);

// Verification Enrollment Result Used for 3DS payment.
String veResult= (String)result.get("Response.ResponseVeResult");

/*****
/*****
/*****
/*STORE veResult IN DATABASE OR ANY SAFE PLACE TO USE IT IN APPROVE REQUEST*/
/*****
/*****
/*****

request.getRequestDispatcher("RedirectTo3DS.jsp").forward(request, response);
}
// this means that the transaction needs Sadad Authentication
else if("20002".equalsIgnoreCase(status)) {
String responseEstn = result.get("Response.estn");
String responseMfu = result.get("Response.mfu");
String responseAuthenticationUrl = result.get("Response.AuthenticationURL");

request.setAttribute("responseEstn", responseEstn);
request.setAttribute("responseMfu", responseMfu);
request.setAttribute("responseAuthenticationUrl", responseAuthenticationUrl);

request.getRequestDispatcher("AuthenticateSadad.jsp").forward(request, response);
}
else {
// then the card is not 3ds enrolled
// this means your payment has been completed
System.out.println("Status is :"+ status);
}
} }
```

5.3 3DS Post Request Submitting to ACS URL

```
<%
//get the error code message
String ACSURL="";
String PaMessage="";
String TermsURL = "";

try{ //set the labels
    ACSURL = (String)request.getAttribute("ACSURL");
    if (ACSURL == null) { //the errorCode is null
        ACSURL="";
    } else { //there is an error ocured
        PaMessage = (String)request.getAttribute("3DSPaMessage");
    }
} catch (Exception e){
    System.out.println("Exception:" + e.getMessage());
}
}%>
<html>
<body onload="javascript:document.ACSAutoSubmitForm.submit();">
<form name="ACSAutoSubmitForm" action="<%= ACSURL%>" method="POST">
<input type="hidden" name="PaReq" value="<%= PaMessage%>">
<input type="hidden" name="TermUrl" value="<%=
response.encodeUrl(request.getAttribute("TERMURL_PREFIX"))%>">
<input type="hidden" name="MD" value="testing">
</form>

</body>
</html>
```

5.4 Sadad authentication Submission

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%
String responseEstn = (String) request.getAttribute("responseEstn");
String responseMfu = (String) request.getAttribute("responseMfu");
String ibUrl = (String) request.getAttribute("responseAuthenticationUrl");
%>

<html>
  <head>
    <%@ page language="java" contentType="text/html; charset=windows-1256"
pageEncoding="windows-1256" %>

    <meta http-equiv="Content-Type" content="text/html; charset=windows-1256">
    <meta http-equiv="Pragma" content="no-cache">
    <meta http-equiv="Expires" content="-1">

    <title>3DS Authentication</title>

    <script>
      function redirectForm() {
        document.AutoSubmitForm.submit();
      }
    </script>
  </head>

  <body>
    <form name="AutoSubmitForm" action="<%= ibUrl %>" method="GET">
      <input type="hidden" name="mfu" value="<%= responseMfu %>">
      <input type="hidden" name="estn" value="<%= responseEstn %>">
    </form>

    <input name="submit" value="Authenticate" type="button" onclick="redirectForm()" >
  </body>
</html>
```

6 API mPayment

```
// import statements
import org.apache.commons.codec.digest.DigestUtils;
```

6.1 Secure Hash Generation

```
//Step 1: Generate Secure Hash
String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store Your
Secret Key in safe Place(e.g. database)

// put the parameters in a TreeMap to have the parameters to have them sorted
alphabetically.
Map <String,String> parameters = new TreeMap<String,String> ();
String transactionId=String.valueOf(System.currentTimeMillis());

// fill required parameters
parameters.put("TransactionID", transactionId);
parameters.put("MerchantID", "ANBRedirectM");
parameters.put("Amount", "2000");
parameters.put("CurrencyISOCODE", "840");
parameters.put("MessageID", "9");
parameters.put("Quantity", "1");
parameters.put("Channel", "0");
parameters.put("PaymentMethod", "1");
parameters.put("ClientIPAddress", "127.0.0.1");

//for Card Payment (conditional;paymentMethod=1)
parameters.put("CardNumber", "4012001045873335");
parameters.put("ExpiryDateYear", "01");
parameters.put("ExpiryDateMonth", "19");
parameters.put("SecurityCode", "123");
parameters.put("CardHolderName", "1");
//for Sadad Payment (conditional;paymentMethod=2)
//parameters.put("SadadOlpId", "testSadad");
//parameters.put("mfu", "https://MerchantSite/RedirectPaymentRequestPage");

//fill some optional parameters
parameters.put("Language", "en");
parameters.put("ThemeID", "1000000001");
parameters.put("Version", "1.0");

//Create an Ordered String of The Parameters Map with Secret Key
StringBuilder orderedString = new StringBuilder();
orderedString.append(SECRET_KEY);
for (String treeMapKey : parameters.keySet()) {
    orderedString.append(parameters.get(treeMapKey));
}

System.out.println("orderdString "+orderedString);
// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String secureHash = new String(DigestUtils.sha256Hex(orderedString.toString()).getBytes());
```

6.2 API mPayment Request

```
//in the response, if the received status code was
"20002" it needs Sadad authentication,
//and after Authentication, you will send APIApprove Request to SmartRoute.
StringBuffer requestQuery = new StringBuffer();

requestQuery
.append("TransactionID").append("=").append(transactionId).append("&")
.append("MerchantID").append("=").append("ANBRedirectM").append("&")
.append("Amount").append("=").append("2000").append("&")
.append("CurrencyISOCode").append("=").append("840").append("&")
.append("MessageID").append("=").append("9").append("&")
.append("Quantity").append("=").append("1").append("&")
.append("Channel").append("=").append("0").append("&")
.append("PaymentMethod").append("=").append("1").append("&")
.append("ClientIPAddress").append("=").append("127.0.0.1").append("&")

//for Card Payment (conditional.append("&")paymentMethod=1)
.append("CardNumber").append("=").append("4012001045873335").append("&")
.append("ExpiryDateYear").append("=").append("01").append("&")
.append("ExpiryDateMonth").append("=").append("19").append("&")
.append("SecurityCode").append("=").append("123").append("&")
.append("CardHolderName").append("=").append("1").append("&")
.append("SecureHash").append("=").append(secureHash).append("&");

//for Sadad Payment (conditional.append("&")paymentMethod=2)
//.append("SadadOlpId").append("=").append("testSadad").append("&")
//.append("mfu", "https://MerchantSite/RedirectPaymentRequestPage").append("&")

//fill some optional parameters
.append("Language").append("=").append("en").append("&")
.append("ThemeID").append("=").append("1000000001").append("&")
.append("Version").append("=").append("1.0")
.append("SecureHash").append("=").append(secureHash);

//Send the request
URL url = new URL("https://SR_URL");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");

//write parameters
writer.write(requestQuery.toString());
writer.flush();

// Get the response
StringBuffer output = new StringBuffer();
BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));
String line;
while ((line = reader.readLine()) != null) {
    output.append(line);
}
writer.close();
reader.close();

//Output the response
System.out.println(output.toString());

// this string is formatted as a "Query String" - name=value&name2=value2.....
```

```

String outputString=output.toString();

// To read the output string you might want to split it
// on '&' to get pairs then on '=' to get name and value
// and for a better and ease on verifying secure hash you should put them in a TreeMap
String [] pairs=outputString.split("&");

Map<String,String> result=new TreeMap<String,String>();

// now we have separated the pairs from each other {"name1=value1","name2=value2",....}
for(String pair:pairs){

    // now we have separated the pair to {"name","value"}
    String[] nameValue=pair.split("=");

    String name=nameValue[0];//first element is the name
    String value=nameValue[1];//second element is the value

    // put the pair in the result map
    result.put(name,value);
}

// Now that we have the map, order it to generate secure hash and compare it with the
received one

StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : result.keySet()) {
    responseOrderdString.append(result.get(treeMapKey));
}

System.out.println("Response Orderd String is " + responseOrderdString.toString());

// Generate SecureHash with SHA256
// Using DigestUtils from appache.commons.codec.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());

// get the received secure hash from result map
String receivedSecurehash=result.get("Response.SecureHash");

if(!receivedSecurehash.equals(generatedsecureHash)){

    //IF they are not equal then the response shall not be accepted
    System.out.println("Received Secure Hash does not Equal generated Secure hash");
}
else{
    // complete the Action get other parameters from result map and do your processes
    // please refer to The Integration Manual to See The List of The Received Parameters
    String status=result.get("Response.Status");
    System.out.println("Status is :"+ status);

    if("20002".equalsIgnoreCase(status)) {
        String responseEstn = result.get("Response.estn");
        String responseMfu = result.get("Response.mfu");
        String responseAuthenticationUrl = result.get("Response.AuthenticationURL");

        request.setAttribute("responseEstn", responseEstn);
        request.setAttribute("responseMfu", responseMfu);
    }
}

```

```
request.setAttribute("responseAuthenticationUrl", responseAuthenticationUrl);

request.getRequestDispatcher("AuthenticateSadad.jsp").forward(request, response);
}
else {
    // then the card is not 3ds enrolled
    // this means your payment has been completed
    System.out.println("Status is :"+ status);
}
} }
```

6.3 Sadad authentication Submission

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%
String responseEstn = (String) request.getAttribute("responseEstn");
String responseMfu = (String) request.getAttribute("responseMfu");
String ibUrl = (String) request.getAttribute("responseAuthenticationUrl");
%>

<html>
  <head>
    <%@ page language="java" contentType="text/html; charset=windows-1256"
pageEncoding="windows-1256" %>

    <meta http-equiv="Content-Type" content="text/html; charset=windows-1256">
    <meta http-equiv="Pragma" content="no-cache">
    <meta http-equiv="Expires" content="-1">

    <title>3DS Authentication</title>

    <script>
      function redirectForm() {
        document.AutoSubmitForm.submit();
      }
    </script>
  </head>

  <body>
    <form name="AutoSubmitForm" action="<%= ibUrl %>" method="GET">
      <input type="hidden" name="mfu" value="<%= responseMfu %>">
      <input type="hidden" name="estn" value="<%= responseEstn %>">
    </form>

    <input name="submit" value="Authenticate" type="button" onclick="redirectForm()" >
  </body>
</html>
```

7 API Approve

7.1 Secure Hash Generation

```
//Step 1: Generate Secure Hash
String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store Your
Secret Key in safe Place(e.g. database)

// put the parameters in a TreeMap to have the parameters to have them sorted
alphabetically.
Map <String,String> parameters = new TreeMap<String,String> ();

String transactionId=String.valueOf(System.currentTimeMillis());

// fill required parameters
parameters.put("MessageID", "1");
parameters.put("TransactionID", transactionId);
parameters.put("MerchantID", "ANBRedirectM");
parameters.put("PaymentMethod", "1");
```

```

//it's real length is very longer than this this value is just a sample
parameters.put("PAREs", URLEncoder.encode("eJzFV2mTokow/Ssd9T4a3ewiHZQvkh2", "UTF-8"));

//in case of sadad payment
//parameters.put("estn", "Test123");

//for Card Payment (conditional;paymentMethod=1)
parameters.put("CardNumber", "4012001045873335");
parameters.put("ExpiryDateYear", "01");
parameters.put("ExpiryDateMonth", "19");
parameters.put("SecurityCode", "123");
parameters.put("CardHolderName", "1");

//Create an Ordered String of The Parameters Map with Secret Key
StringBuilder orderedString = new StringBuilder();
orderedString.append(SECRET_KEY);
for (String treeMapKey : parameters.keySet()) {
    orderedString.append(parameters.get(treeMapKey));
}

System.out.println("orderdString "+orderedString);
// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String secureHash = new String(DigestUtils.sha256Hex(orderedString.toString()).getBytes());

```

7.2 API Approve Request

```
StringBuffer requestQuery = new StringBuffer();

requestQuery
.append("TransactionID").append("=").append(transactionId).append("&")
.append("MerchantID").append("=").append("ANBRedirectM").append("&")
.append("MessageID").append("=").append("1").append("&")
.append("PaymentMethod").append("=").append("1").append("&")

//for Sadad Payment (conditional.append("&")paymentMethod=2)
//.append("estn").append("=").append("Test123").append("&")

//for Card Payment (conditional.append("&")paymentMethod=1)
.append("CardNumber").append("=").append("4012001045873335").append("&")
.append("ExpiryDateYear").append("=").append("01").append("&")
.append("ExpiryDateMonth").append("=").append("19").append("&")
.append("SecurityCode").append("=").append("123").append("&")
.append("CardHolderName").append("=").append("1").append("&")
.append("SecureHash").append("=").append(secureHash).append("&");

.append("PAREs").append("=").append(URLEncoder.encode("eJzFV2mTokow/Ssd9T4a3ewiHZQvkh2", "UTF-8")).append("&");

//Send the request
URL url = new URL("https://SR_URL");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");

//write parameters
writer.write(requestQuery.toString());
writer.flush();

// Get the response
StringBuffer output = new StringBuffer();
BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream(), "UTF-8"));
String line;
while ((line = reader.readLine()) != null) {
    output.append(line);
}
writer.close();
reader.close();

//Output the response
System.out.println(output.toString());

// this string is formatted as a "Query String" - name=value&name2=value2.....
String outputString=output.toString();

// To read the output string you might want to split it
// on '&' to get pairs then on '=' to get name and value
// and for a better and ease on verifying secure hash you should put them in a TreeMap
String [] pairs=outputString.split("&");

Map<String,String> result=new TreeMap<String,String>();

// now we have separated the pairs from each other {"name1=value1","name2=value2",....}
for(String pair:pairs){
```

```

    // now we have separated the pair to {"name","value"}
    String[] nameValue=pair.split("=");

    String name=nameValue[0];//first element is the name
    String value=nameValue[1];//second element is the value

    // put the pair in the result map
    result.put(name,value);
}
// Now that we have the map, order it to generate secure hash and compare it with the
received one
StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : result.keySet()) {
    responseOrderdString.append(result.get(treeMapKey));
}

System.out.println("Response Orderd String is " + responseOrderdString.toString());

// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());

// get the received secure hash from result map
String receivedSecurehash=result.get("Response.SecureHash");

if(!receivedSecurehash.equals(generatedsecureHash)){

    //IF they are not equal then the response shall not be accepted
    System.out.println("Received Secure Hash does not Equal generated Secure hash");
}
else{
    // Complete the Action get other parameters from result map and do your processes
    // please refer to The Integration Manual to See The List of The Received Parameters
    String status=result.get("Response.Status");
    System.out.println("Status is :"+ status);
}

```

8 Inquiry Message

8.1 Secure Hash Generation

```
//Step 1: Generate Secure Hash
String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store Your
Secret Key in safe Place (e.g. database)

// put the parameters in a TreeMap to have the parameters to have them sorted
alphabetically.
Map <String,String> parameters = new TreeMap<String,String> ();

String transactionId=String.valueOf(System.currentTimeMillis());

// fill required parameters
parameters.put("MessageID", "2");
parameters.put("OriginalTransactionID", "1440954863817");
parameters.put("MerchantID", "ANBRedirectM");
parameters.put("Version", "1.0");

//Create an ordered String of The Parameters Map with Secret Key
StringBuilder orderedString = new StringBuilder();
orderedString.append(SECRET_KEY);
for (String treeMapKey : parameters.keySet()) {
    orderedString.append(parameters.get(treeMapKey));
}

System.out.println("orderdString "+orderedString);
// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String secureHash = new String(DigestUtils.sha256Hex(orderedString.toString()).getBytes());
```

8.2 Inquiry Request

```
StringBuffer requestQuery = new StringBuffer();

requestQuery
.append("OriginalTransactionID").append("=").append(1440954863817).append("&")
.append("MerchantID").append("=").append("ANBRedirectM").append("&")
.append("MessageID").append("=").append("2").append("&")
.append("Version").append("=").append("1.0").append("&")
.append("SecureHash").append("=").append(secureHash).append("&");

//Send the request
URL url = new URL("https://SR_URL");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");

//write parameters
writer.write(requestQuery.toString());
writer.flush();

// Get the response
StringBuffer output = new StringBuffer();
BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));
String line;
while ((line = reader.readLine()) != null) {
    output.append(line);
}
writer.close();
reader.close();

//Output the response
System.out.println(output.toString());

// this string is formatted as a "Query String" - name=value&name2=value2.....
String outputString=output.toString();

// To read the output string you might want to split it
// on '&' to get pairs then on '=' to get name and value
// and for a better and ease on verifying secure hash you should put them in a TreeMap
String [] pairs=outputString.split("&");

Map<String,String> result=new TreeMap<String,String>();

// now we have separated the pairs from each other {"name1=value1","name2=value2",...}
for(String pair:pairs){

    // now we have separated the pair to {"name","value"}
    String[] nameValue=pair.split("=");

    String name=nameValue[0];//first element is the name
    String value=nameValue[1];//second element is the value

    // put the pair in the result map
    result.put(name,value);
}
// Now that we have the map, order it to generate secure hash and compare it with the
received one
StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
```

```

for (String treeMapKey : result.keySet()) {
    responseOrderdString.append(result.get(treeMapKey));
}

System.out.println("Response Orderd String is " + responseOrderdString.toString());

// Generate SecureHash with SHA256
// Using DigestUtils from apache.commons.codes.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());

// get the received secure hash from result map
String receivedSecurehash=result.get("Response.SecureHash");

if(!receivedSecurehash.equals(generatedsecureHash)){

    //IF they are not equal then the response shall not be accepted
    System.out.println("Received Secure Hash does not Equal generated Secure hash");
}
else{
    // Complete the Action get other parameters from result map and do your processes
    // Please refer to The Integration Manual to See The List of The Received Parameters
    String status=result.get("Response.Status");
    System.out.println("Status is :"+ status);
}

```

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](#).

```
// import statements
import org.apache.commons.codec.digest.DigestUtils;
```

9.1 Secure Hash Generation

```
//Step 1: Generate Secure Hash
String SECRET_KEY = "Y2FkMTdlOWZiMzJjMzY4ZGFkMzhkMWIz"; // Use Yours, Please Store Your
Secret Key in safe Place(e.g. database)

// put the parameters in a TreeMap to have the parameters to have them sorted
alphabetically.
Map <String,String> parameters = new TreeMap<String,String> ();

String transactionId=String.valueOf(System.currentTimeMillis());

// fill required parameters
parameters.put("MessageID", "4");
parameters.put("TransactionID", transactionId);
parameters.put("OriginalTransactionID", "1440954863817");
parameters.put("MerchantID", "ANBRedirectM");
parameters.put("Amount", "2000");
parameters.put("CurrencyISOCODE", "840");
parameters.put("Version", "1.0");

//Create an Ordered String of The Parameters Map with Secret Key
StringBuilder orderedString = new StringBuilder();
orderedString.append(SECRET_KEY);
for (String treeMapKey : parameters.keySet()) {
    orderedString.append(parameters.get(treeMapKey));
}

System.out.println("orderdString "+orderedString);
// Generate SecureHash with SHA256
// Using DigestUtils from appache.commons.codes.jar Library
String secureHash = new String(DigestUtils.sha256Hex(orderedString.toString()).getBytes());
```

9.2 Refund Request

```
StringBuffer requestQuery = new StringBuffer();

requestQuery
.append("TransactionID").append("=").append(transactionId).append("&")
.append("MerchantID").append("=").append("ANBRedirectM").append("&")
.append("MessageID").append("=").append("4").append("&")
.append("Amount").append("=").append("2000").append("&")

.append("OriginalTransactionID").append("=").append("1440954863817").append("&")
.append("CurrencyISOCode").append("=").append("840").append("&")
.append("SecureHash").append("=").append(secureHash).append("&")
.append("Version").append("=").append("1.0").append("&");

//Send the request
URL url = new URL("https://SR_URL");
URLConnection conn = url.openConnection();
conn.setDoOutput(true);
OutputStreamWriter writer = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");

//write parameters
writer.write(requestQuery.toString());
writer.flush();

// Get the response
StringBuffer output = new StringBuffer();
BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream(),
"UTF-8"));
String line;
while ((line = reader.readLine()) != null) {
    output.append(line);
}
writer.close();
reader.close();

//Output the response
System.out.println(output.toString());

// this string is formatted as a "Query String" - name=value&name2=value2.....
String outputString=output.toString();

// To read the output string you might want to split it
// on '&' to get pairs then on '=' to get name and value
// and for a better and ease on verifying secure hash you should put them in a TreeMap
String [] pairs=outputString.split("&");

Map<String,String> result=new TreeMap<String,String>();

// now we have separated the pairs from each other {"name1=value1","name2=value2",....}
for(String pair:pairs){

    // now we have separated the pair to {"name", "value"}
    String[] nameValue=pair.split("=");

    String name=nameValue[0];//first element is the name
    String value=nameValue[1];//second element is the value

    // put the pair in the result map
    result.put(name,value);
}
```

```

// Now that we have the map, order it to generate secure hash and compare it with the
received one
StringBuilder responseOrderdString = new StringBuilder();
responseOrderdString.append(SECRET_KEY);
for (String treeMapKey : result.keySet()) {
    responseOrderdString.append(result.get(treeMapKey));
}

System.out.println("Response Orderd String is " + responseOrderdString.toString());

// Generate SecureHash with SHA256
// Using DigestUtils from appache.commons.codes.jar Library
String generatedsecureHash = new
String(DigestUtils.sha256Hex(responseOrderdString.toString()).getBytes());

// get the received secure hash from result map
String receivedSecurehash=result.get("Response.SecureHash");

if(!receivedSecurehash.equals(generatedsecureHash)){

    //IF they are not equal then the response shall not be accepted
    System.out.println("Received Secure Hash does not Equal generated Secure hash");
}
else{
    // Complete the Action get other parameters from result map and do your processes
    // please refer to The Integration Manual to See The List of The Received Parameters
    String status=result.get("Response.Status");
    System.out.println("Status is :"+ status);
}

```