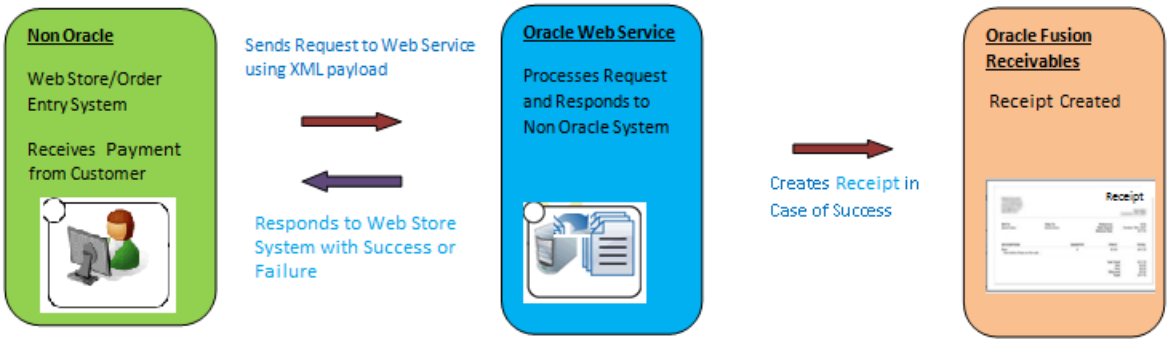
Usage Notes for Receivables Create and Apply Receipt Service

Table of Contents

1 Overview

A receipt is a Receivables document that describes the payment received in exchange for goods or services. An online web-store front receives payments from their customers for the sale of goods. The **Receivables Create and Apply Receipt Service** integrates the deploying company's non-Oracle order entry/online web-store systems to Oracle Fusion Receivables. This service receives the payment information from the online user and creates a corresponding receipt and applies the receipt automatically to the invoice.



Web Services Description Language (WSDL) of the **Receivables Create and Apply Receipt Service** can be accessed from the SOAP Web Services for Oracle Financials Cloud guide and is available to any user who has access to Oracle Fusion Receivables. This service supports both synchronous and asynchronous processing.

The **Receivables Create and Apply Receipt Service** is a SOAP encoded Web service which uses HTTPS transport as defined in the WSDL. A SOAP client can be manually created to invoke this service (example described in appendix 3.1). Another alternative is to automatically invoke the service via WSDL invocation tools, such as the Web Services Invocation Framework (WSIF) for Java clients or SOAP::Lite for Perl.

2 Service Details

2.1 Supported Operations

The following operations are available in the Receivables Create and Apply Receipt Service.

createAndApplyReceipt

This operation is used to create and apply the receipt to the invoice in synchronous mode. It accepts an object of Receivables Standard Receipt and Receipt Application SDO as the input parameter and returns a response of the same object.

createAndApplyReceiptAsync

This operation is used to create and apply the receipt to the invoice document in asynchronous mode. It accepts an object of Receivables Standard Receipt and Receipt Application SDO as the input parameter and returns a response of the same object.

createAndApplyReceiptAsyncResponse

This operation is used to get the response of the asynchronous invocation.

The asynchronous requests to Oracle web service are placed in a processing queue and handled asynchronously with other requests. The client application does not wait for a response. Once a job is submitted, a job ID is returned in the Web services response. The client application can then check on the status and result of the request by referencing the job ID.

2.2 Setups and Security

Prerequisite Setups

The prerequisite setups needed to invoke the Receivables Create and Apply Receipt service are the same as creating a receipt. These can be set up in the Fusion instance through the Functional Setup Manager.

- Customers
- Receivables System Options
- Receipt Class and Receipt Methods

Security Details

The users and their credentials in the source system should be synchronized with the Oracle Fusion instance so that the user invoking the web service from the source system will get authorized at the Oracle Fusion instance. Once the service is invoked, the XML request payload is submitted to the web service which processes the request and creates a receipt in Oracle Fusion Receivables. The user who has the Receivables Specialist or Receivables Manager Job Role will be able view the receipt successfully created in Oracle Fusion Receivables

2.3 Invoking Receivables Create and Apply Receipt Service using Web Service Proxy Client

Step 1: Get the service description from the createAndApplyReceipt WSDL URL.
http://<HostName>:<PortNumber>/finArRcptsSharedCommonService/CreateAndApplyReceiptService?WSDL

Step 2: Use WSDL and write a Java class (for example CreateAndApplyReceiptServiceClient.java)
to invoke the createAndApplyReceipt operation of CreateAndApplyReceiptService web service and return the response.
Complete Details of a sample Service Client Java code are found in Section 3.1
The Java client class (CreateAndApplyReceiptServiceClient.java) should have the following two main methods:

a) ConstructPayload() : Create the Java method to construct the XML payload based on the functional parameter values. This XML format should follow the standard SOAP (Simple Object Access Protocol) specifications. The XML request payload will be passed while invoking the service. The XML Tag names for the input parameters (Service Data Object attributes) can be obtained from the SDO Section [2.4]. Example of a sample request payload can be found in Section [3.4].

b) createAndApplyReceipt() : Create the java method to invoke the service via http connection and get the response. This method will accept the following parameters:
   - Service URL
   - Username/Password
   - Keystore details
   - Input Payload

Step 3: Prepare the Keystore information for certificate authentication.

- Launch the WSDL URL in the browser, click on the Lock icon appearing on the starting of the URL and export the certificate to the local system.
- Get the keystore file from the Financial Domain of the Fusion instance.

- Use the key tool command to import the certificate into the keystore
  `keytool -import -file <sslcertfile> -keystore <keystorefile>.`
- Use this keystore file in the CreateAndApplyReceiptServiceClient.Java class.

Step 4: Integrate the CreateAndApplyReceiptServiceClient.java class to the source system to create the receipt and apply it to an invoice.

There are four major areas used in the source system to integrate the client java class:

- **Set the input parameters**
  The input parameters are the functional parameter values that are required for receipt creation. The values entered here will be formulated as XML payload in ConstructPayload() method of CreateAndApplyReceiptServiceClient.java class. The XML tags for the functional parameters are shown in the SDO attribute names mentioned in SDO Section (2.4).

- **Set the Security information**
  The combination of keystore information (obtained from step 3) and the user credentials are required for authentication of the service call at the Oracle Fusion instance.

- **Call the Client code for service invocation and receipt creation**
  This steps calls the createAndApplyReceipt() method of CreateAndApplyReceiptServiceClient.java class with the input parameters, host information and the security information.

- **Parse the service response and handling error messages**
  The invocation of the service returns the response upon completion. The response is in a XML format and it can be parsed using any standard java class (like XPathFactory) to read the values of the tags and analyze them to determine the status of the service call. If the service call reports an error status, then the service error message can be sent to the user to review the cause for the service invocation failure.

Sample code for invocation of the CreateAndApplyReceiptServiceClient is available in Appendix 3.2

Step 5: Test the CreateAndApplyReceiptService web service.

- **Install Java and set PATH system**
  Install the Java Development Kit (JDK) and set the system environment path and variables.

Details of the testing is available in Appendix 3.3

2.4 Service Data Objects

The **Receivables Create and Apply Receipt Service** Web service supports the following Service Data Object (SDO):

Receivables Standard Receipt and Receipt Application SDO

This is the Service SDO that contains information pertaining to the receipt. The SDO Attributes are:

Attribute Name	Attribute Description	Data Type	Required
BusinessUnitName	A unit of an enterprise that performs one or more business functions that can be rolled up in a management hierarchy.	java.lang.String	No. Value will be derived from the Profile Option "Default Business Unit".
ReceiptMethodName	Attribute that associates receipt class, remittance bank, and receipt account information with receipts.	java.lang.String	Yes
ReceiptNumber	Number that identifies a receipt.	java.lang.String	Yes
ReceiptDate	Date when the receipt is created.	java.sql.Date	No.
GlDate	The date, referenced from Oracle Fusion General Ledger, used to determine the accounting period for the receipt.	java.sql.Date	No. Value will be derived from the receipt date. If the receipt date is in a closed period, then the first date in the next open period will be used.
Amount	Amount of the receipt.	java.math.BigDecimal	Yes
CustomerName	Name that identifies a customer.	java.lang.String	Yes
CustomerAccountNumber	Number that identifies a bill-to customer account.	java.lang.String	Yes
CurrencyCode	Currency of the Receivables receipt.	java.lang.String	Yes
TransactionNumber	Number that identifies a transaction that is getting applied.	java.lang.String	Yes
CustomerTrxId	Unique Identifier of the transaction that is getting applied.	java.lang.Long	No.

2.5 Error Handling

When a Web service request is being processed and an error is encountered, the nature of the error will be communicated to the client. The SOAP specification defines a standard, platform-independent way of describing the error within the SOAP message using a SOAP fault.

SOAP faults can be one of the following types:

- **Modeled**
  This refers to an exception that is thrown explicitly from the business logic and mapped to wsdl:fault definitions in the WSDL file.
- **Unmodeled**
  This refers to an exception that is generated at run-time when no business logic fault is defined in the WSDL. In this case, Java exceptions are represented as generic SOAP fault exceptions, javax.xml.ws.soap.SOAPFaultException.

The faults are returned to the sender only for synchronous mode. If a Web service invocation is asynchronous, the SOAP fault is not returned to the sender but stored for further processing. For more information about exception handling refer to the Exception Handling SOAP documentation.

The table below shows the error message names and the accompanying the text that describes each error.

Receivables Create and Apply Receipt Service Error Messages Table

Message Name	Message Text
AR_NO_ROW_IN_SYSTEM_PARAMETERS	The business unit defined in the MO: Business Unit profile option is invalid.
AR_RAPI_CUS_NAME_INVALID	The customer name is invalid.
AR_RAPI_CUS_NUM_INVALID	The customer account number is invalid.
AR_RAPI_CUS_NAME_NUM_INVALID	The combination of customer name or account number is invalid.
AR_RAPI_RCPT_MD_NAME_INVALID	The receipt method name {REC_METH} is invalid.
AR_RAPI_REM_BK_AC_NAME_INVALID	The remittance bank account name is invalid.
AR_RAPI_CURR_CODE_INVALID	The currency is invalid.

AR_RAPI_TRX_NUM_INVALID	The transaction number is invalid.
AR_INVALID_APP_GL_DATE	The accounting date {GL_DATE} is not in an open or future-enterable period.
AR_RAPI_RCPT_AMOUNT_NULL	The receipt amount is null
AR_RW_RCT_AMOUNT_NEGATIVE	You cannot enter a negative receipt amount for cash receipts.

3 Appendix

3.1 Sample of the Service Client Java code.

CreateAndApplyReceiptServiceClient.Java code:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.HashMap;

public class CreateAndApplyReceiptServiceClient {
    public CreateAndApplyReceiptServiceClient() {
        super();
    }

    private static String serviceURL;
    private static String inputPayload;
    private static String username;
    private static String password;
    private static String userToken;
    private static String outputPayload;


    public static String httpPost(String destUrl, String postData,
                                  String authStr, String keyStoreLocation,
                                  String keyStorePassword) throws Exception {

        System.out.println();
        System.out.println("Invoking the Service");

       // Setting the KeyStore Properties

        System.setProperty("javax.net.ssl.trustStore", keyStoreLocation);
        System.setProperty("javax.net.ssl.trustStorePassword",
                           keyStorePassword);

       //Open the HTTP connection and set the connection properties

        URL url = new URL(destUrl);
        HttpURLConnection conn = (HttpURLConnection)url.openConnection();
        if (conn == null) {
            return null;
        }

        conn.setRequestProperty("Content-Type", "text/xml;charset=UTF-8");
        conn.setDoOutput(true);
        conn.setDoInput(true);
        conn.setUseCaches(false);
        conn.setFollowRedirects(true);
        conn.setAllowUserInteraction(false);
        conn.setRequestMethod("POST");

       //Set the Authorization property for the HTTP connection using the
       username and password

        byte[] authBytes = authStr.getBytes("UTF-8");
```

```
    String auth = Base64.byteArrayToBase64(authBytes);
    conn.setRequestProperty("Authorization", "Basic " + auth);

   //Post the http request. This will invoke the Create and Apply Receipt
   Web Service for creating the Receipt and applying it to the invoice.

    OutputStream out = conn.getOutputStream();
    OutputStreamWriter writer = new OutputStreamWriter(out, "UTF-8");
    writer.write(postData);
    writer.close();
    out.close();

    try {
        InputStream errIs = conn.getErrorStream();
        if (errIs != null) {
            String err = getString(errIs);
            if (err != null && !err.isEmpty()) {
                System.out.println(err);
            }
            errIs.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }


   // Read the response and return it to the calling Java API

    String response = null;
    try {
        InputStream in = conn.getInputStream();
        if (in != null) {
            response = getString(in);
            in.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    conn.disconnect();
    return response;
}

public static String getString(InputStream errIs) {

    BufferedReader br = null;
    StringBuilder sb = new StringBuilder();

    String line;
    try {

        br = new BufferedReader(new InputStreamReader(errIs));
        while ((line = br.readLine()) != null) {
            sb.append(line);
        }

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (br != null) {
            try {
                br.close();
```

```
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return sb.toString();

}

public void setInputPayload(String inputPayload) {
    this.inputPayload = inputPayload;
}

public String getInputPayload() {
    return this.inputPayload;
}

public void setWebService(String webService) {
    this.serviceURL = webService;
}

public static String getWebService() {
    return serviceURL;
}

public void setUsername(String username) {
    this.username = username;
}

public static String getUsername() {
    return username;
}

public void setPassword(String password) {
    this.password = password;
}

public static String getPassword() {
    return password;
}

public void setUserToken(String userToken) {
    this.userToken = userToken;
}

public String getUserToken() {
    return userToken;
}

public void setOutputPayload(String outputPayload) {
    this.outputPayload = outputPayload;
}

public String getOutputPayload() {
    return outputPayload;
}

public String constructPayload(HashMap createAndApplyReceiptValues,) {
    String payload =
              "<soap:Envelope
   xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
        "<soap:Body>" +
```

```
            "<ns1:createAndApplyReceiptAsync
   xmlns:ns1=\"http://xmlns.oracle.com/apps/financials/receivables/receipts/
   shared/createAndApplyReceiptService/commonService/types/\">" +
        "<ns1:createAndApplyReceipt
   xmlns:ns2=\"http://xmlns.oracle.com/apps/financials/receivables/receipts/
   shared/createAndApplyReceiptService/commonService\">" +
        "<ns2:ReceiptMethodName>" +
        createAndApplyReceiptValues.get("ReceiptMethodName") +
        "</ns2:ReceiptMethodName>" + "<ns2:ReceiptNumber>" +
        createAndApplyReceiptValues.get("ReceiptNumber") +
        "</ns2:ReceiptNumber>" + "<ns2:Amount>" +
        createAndApplyReceiptValues.get("Amount") +
        "</ns2:Amount>" +
               "<ns2:CustomerName>" +
   createAndApplyReceiptValues.get("CustomerName") +
        "</ns2:CustomerName>" + "<ns2:CustomerAccountNumber>" +
               createAndApplyReceiptValues.get("CustomerAccountNumber") +
   "</ns2:CustomerAccountNumber>" +
        "<ns2:TransactionNumber>" +
   createAndApplyReceiptValues.get("TransactionNumber") +
        "</ns2:TransactionNumber>" +
        "</ns1:createAndApplyReceipt>" +
        "</ns1:createAndApplyReceiptAsync>" + "</soap:Body>" +
        "</soap:Envelope>";
    return payload;
}

public String createAndApplyReceipt(String hostName, int port, String
username,String password,HashMap createAndApplyReceiptValues, String
keyStoreLocation,String keyStorePassword) throws Exception {

    if (port < 0)
        this.setWebService("https://" + hostName +

          "/finArRcptsSharedCommonService/CreateAndApplyReceiptService");
    else
        this.setWebService("https://" + hostName + ":" + port +

          "/finArRcptsSharedCommonService/CreateAndApplyReceiptService");
    this.setUsername(username);
    this.setPassword(password);

   // construct the XML input payload

    String reqPayload = this.constructPayload(createAndApplyReceiptValues);
    this.setInputPayload(reqPayload);

   // Invoke the service via a http secure connection

    String response =
               httpPost(getWebService() + "?invoke=", getInputPayload(),
   getUsername() +
                 ":" + getPassword(), keyStoreLocation, keyStorePassword);
    return response;
                 }}
```

Base64.Java code:

```
 public class Base64 {
       public Base64() {
           super();
       }
```

```
    public static String byteArrayToBase64(byte[] a) {
        return byteArrayToBase64(a, a.length);
    }

    public static String byteArrayToBase64(byte[] a, int aLen) {
        int numFullGroups = aLen / 3;
        int numBytesInPartialGroup = aLen - 3 * numFullGroups;
        int resultLen = 4 * ((aLen + 2) / 3);
        StringBuffer result = new StringBuffer(resultLen);
        char intToAlpha[] = intToBase64;

        int inCursor = 0;
        for (int i = 0; i < numFullGroups; i++) {
            int byte0 = a[inCursor++] & 0xff;
            int byte1 = a[inCursor++] & 0xff;
            int byte2 = a[inCursor++] & 0xff;
            result.append(intToAlpha[byte0 >> 2]);
            result.append(intToAlpha[byte0 << 4 & 0x3f | byte1 >> 4]);
            result.append(intToAlpha[byte1 << 2 & 0x3f | byte2 >> 6]);
            result.append(intToAlpha[byte2 & 0x3f]);
        }

        if (numBytesInPartialGroup != 0) {
            int byte0 = a[inCursor++] & 0xff;
            result.append(intToAlpha[byte0 >> 2]);
            if (numBytesInPartialGroup == 1) {
                result.append(intToAlpha[byte0 << 4 & 0x3f]);
                result.append("==");
            } else {
                int byte1 = a[inCursor++] & 0xff;
                            result.append(intToAlpha[byte0 << 4 & 0x3f
              | byte1 >> 4]);
                result.append(intToAlpha[byte1 << 2 & 0x3f]);
                result.append('=');
            }
        }
        return result.toString();
    }


    private static final char intToBase64[] =
      { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M',
      'N',
      'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a',
      'b',
      'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
      'p',
      'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2',
      '3',
      '4', '5', '6', '7', '8', '9', '+', '/' };

}
```

3.2 ReceiptProcessInvoke.java code

```
import java.io.StringReader;
import java.util.HashMap;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.xpath.XPath;
```

```
import javax.xml.xpath.XPathFactory;
import org.w3c.dom.Document;
import org.xml.sax.InputSource;


public class ReceiptProcessInvoke {
    public ReceiptProcessInvoke() {
        super();
    }

    public static void main(String[] args) throws Exception {

        HashMap createAndApplyReceiptValues = new HashMap();

      // Setting necessary input parameter values utilizing the SDO
      Attributes. See Section 2.2 for list of SDOs and Attributes

        createAndApplyReceiptValues.put("ReceiptMethodName", "USD");
        createAndApplyReceiptValues.put("ReceiptNumber", "Customer 1");
        createAndApplyReceiptValues.put("Amount", "100.00");
        createAndApplyReceiptValues.put("CustomerName", "Customer 1");
        createAndApplyReceiptValues.put("CustomerAccountNumber", "1001");
        createAndApplyReceiptValues.put("TransactionNumber", "4");

      // Setting keyStore Location

        String keyStoreLocation = "C:\\default-keystore.jks";
        String keyStorePass = "welcome1";
        String username = "guest";
        String password = "guest";

      // Calling the Create and Apply Receipt operation
      // The createAndApplyReceipt method calls the Http Post to invoke
      the service

        createAndApplyReceiptServiceClient receipt = new
      createAndApplyReceiptServiceClient();
        String response =
                  receipt.createAndApplyReceipt("efops-rel8-cdrmdit-
      external-fin.us.oracle.com",-1, username, password,
      createAndApplyReceiptValues,
                          keyStoreLocation, keyStorePass);

      // Parse the response to read the service output details and
      handle errors if any.

        if (response != null && !response.isEmpty()) {
            System.out.println();
                  InputSource source = new InputSource(new
      StringReader(response));
            DocumentBuilderFactory dbf =
      DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            Document document = db.parse(source);
            XPathFactory xpathFactory = XPathFactory.newInstance();
            XPath xpath = xpathFactory.newXPath();
            String receiptNumber =

      xpath.evaluate("Envelope/Body/createAndApplyReceiptAsyncResponse/r
      esult/ReceiptNumber",  document);
                  if ("".equals(receiptNumber) ) {
                System.out.println("Service Errored. Parse the Response
      to review the Error Message ");
```

```
            } else {
                System.out.println("Service Status = Success");
             }
} } }
```

3.3 Testing the CreateAndApplyReceiptService web service.

One-time step required for running any Java executable on a Windows system is given below.
Download and install java and set PATH system environment variable.

- Get the latest JDK version for appropriate operating system if Java is not installed already.
- Install the Java from the executable.
- Note the path of the bin in the Java installation directory.
  The default path is `C:\Program Files\Java\jdkx.x.x\bin`.

Set the System Environment variables.

- Right-click on 'My Computer' and select 'Properties'.
- Go to Advance System Setting.
- Click Environment Variable.
- Set the PATH variable.
  If present double click it and append the bin path like ";`C:\Program Files\Java\jdkx.x.x\bin`.;"
  Else create a New Variable by clicking NEW.
  Provide variable name as "Path" and Variable value as path of bin directory
  " `C:\Program Files\Java\jdkx.x.x\bin`.;"
- Click OK

Javac ReceiptProcessInvoke.java – This generates the Java class file.
Java ReceiptProcessInvoke – This invokes the service and prints the output.

Output :

Invoking Create and Apply Receipt Web Service
Response received.
Receipt Number = 1566

The receipt number = 1566 can be queried on the Fusion instance to review the receipt.

3.4 Example of a Sample Payload

Request Payload

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns1:createAndApplyReceiptAsync
xmlns:ns1="http://xmlns.oracle.com/apps/financials/receivables/receipts/s
hared/createAndApplyReceiptService/commonService/types/">
            <ns1:createAndApplyReceipt
xmlns:ns2="http://xmlns.oracle.com/apps/financials/receivables/receipts/s
hared/createAndApplyReceiptService/commonService/">
          <ns2:BusinessUnitName> Vision Operations</ ns2:BusinessUnitName>
           <ns2:ReceiptMethodName>Check - BofA-ED</ns2:ReceiptMethodName>
           <ns2:ReceiptNumber>Rec_1163</ns2:ReceiptNumber>
          <ns1:CurrencyCode>USD</ns1:CurrencyCode>
           <ns2:Amount>362.80</ns2:Amount>
           <ns2:CustomerName>AR_Customer</ns2:CustomerName>
          <ns2:CustomerAccountNumber>1001</ns2:CustomerAccountNumber>
           <ns2:TransactionNumber>1163</ns2:TransactionNumber>
           </ns1:createAndApplyReceipt>
        </ns1:createAndApplyReceiptAsync>
    </soap:Body>
</soap:Envelope>
```

Response Payload

The Response payload will contain the receipt number if the invocation is successful.

```
<ns0:createAndApplyReceiptAsyncResponse xmlns=""
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns0="http://xmlns.oracle.com/apps/financials/receivables/receipts/s
hared/createAndApplyReceiptService/commonService/types/"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
<ns2:result xmlns:ns0="http://xmlns.oracle.com/adf/svc/types/"
xmlns:ns1="http://xmlns.oracle.com/apps/financials/receivables/receipts/s
hared/createAndApplyReceiptService/commonService/"
xmlns:ns2="http://xmlns.oracle.com/apps/financials/receivables/receipts/s
hared/createAndApplyReceiptService/commonService/types/"
xmlns:tns="http://xmlns.oracle.com/adf/svc/errors/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="ns1:CreateAndApplyReceiptResult"><ns1:Value>
       <ns2:BusinessUnitName> Vision Operations</ ns2:BusinessUnitName>
       <ns1:ReceiptMethodName> Check - BofA-ED</ns1:ReceiptMethodName>
       <ns1:ReceiptNumber>Rec_1163</ns1:ReceiptNumber>
       <ns1:ReceiptDate/><ns1:GlDate xsi:nil="true"/>
       <ns1:Amount>435.35</ns1:Amount>
       <ns1:CustomerName>AR_Customer</ns1:CustomerName>
       <ns1:CustomerAccountNumber>10011</ns1:CustomerAccountNumber>
       <ns1:CurrencyCode>USD</ns1:CurrencyCode>
       <ns1:TransactionNumber>1163</ns1:TransactionNumber>
       </ns1:Value>
</ns2:result>
</ns0:createAndApplyReceiptAsyncResponse>
```