Alcinia Limited

Shylock Interface Specification Documentation

Technical description of interfaces between Software Systems comprising the Shylock Billing service.

1. Revision History

Date	Version	Author	Description
6/02/2008	V0.1	Tasos Chatzipavlou	Initial creation of Documentation. • Chapters and sections defined. • Writing tasks assigned. • Introduction and glossary outlined
7/02/2008	V0.2	Stathis Alexopoulos	Revision and definition of used services for the Front End Interface and Network Interface.
7/02/2008	V0.3	Tasos Chatzipavlou	Definition of parameters for • Create Account • Get Account Information • Set Account Info
7/02/2008	V0.3a	Tasos Chatzipavlou	Definition of parameters for • rest of Front End Interface services and • Network Interface services.
8/02/2008	V0.4	Stathis Alexopoulos	Introduction of Accounting Interface. The scope and needed services defined.
8/02/2008	V0.5	Tasos Chatzipavlou	Definition of parameters for • Get Accounts Total • Get Totals
10/02/2008	V0.6	Stathis Alexopoulos	Redefinition of Network Interface because of new design. Now the interface is generic enough to hold any kind of usage.
10/02/2008	V0.7	Tasos Chatzipavlou	Definition of parameters for Accounting Interface and for the new Network Interface.
10/02/2008	V0.8	Stathis Alexopoulos	Redefinition of customer category design. The design of Offers Management leads to a total redesign of Customer category

10/02/2008	V0.9	Stathis Alexopoulos	Addition of sequence diagrams for the generic services of Network Interface.
10/02/2008	V0.10	Tasos Chatzipavlou	Addition of short description for each interface
10/02/2008	V0.11		

2. Contents

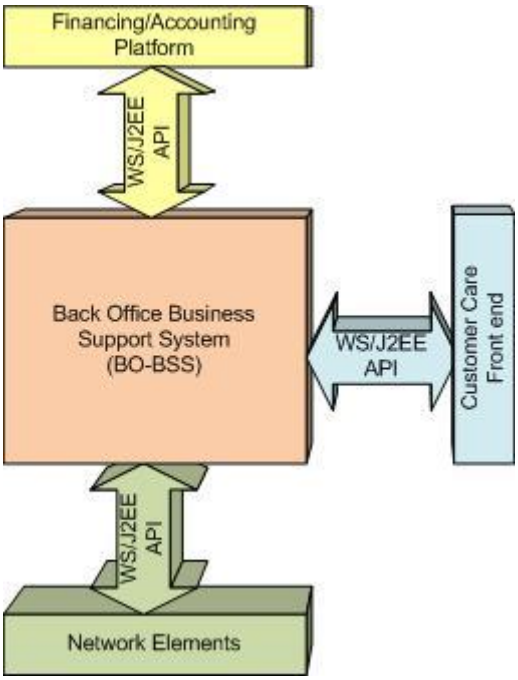
3. Introduction

3.1. Scope

This document's purpose is to present an initial proposal of the interfaces that are going to be built for communication purposes between Shylock platform and external OSS Platform.

3.2. Overview

The whole solution is based on three basic interfaces as mentioned in the following diagram:



Front End Interface: The purpose of this interface is to provide the appropriate methods for account manipulation (e.g. account creation. Usage events history etc)

Network Interface: The purpose of this interface is to provide the appropriate methods for service handling, such as service availability request, service provisioning, service expiration etc

Financing/Accounting Interface: The purpose of this interface is to provide the appropriate methods for accounting actions such as acquiring financial and usage data per customer for a given time period

3.3. Audience

The targeted audiences of this document are managers who want to validate business processes against the supported features of a software system. Engineers who want to implement specific feature on their system. Testers who want to test the behavior between cooperated systems.

What comes next, is an initial proposal of the methods, the above Interfaces are going to expose, along with a sort description for each on of them

4. Front End Interface

The purpose of this interface is to provide the appropriate methods for account manipulation such as account creation, modification, and account information retrieval. These methods are going to be exposed in the form of Web Service calls.

4.1. Create Account

The intention of Create Account service is to create a new Account. The responsibility of account creation remains always to the “Shop” application.

Parameters

Name	Type	Mandatory	Description
accountId	Integer	Yes	The ID of this account. The system that initially created the account is responsible for the creation of this ID.
userName	String	Yes	This value has to be unique to the billing system.
password	String	Yes	A non empty string for user authentication. Any restrictions concerning the format of password remains in the responsibility of “Shop” application.
category	String	Yes	
status	String	Yes	The activation status of customer.
currency	String	No	
billingAddress	String	No	Billing Address
billingCity	String	No	Billing City
billingPostCode	String	No	Zip Code for the Billing.
billingCountry	String	No	Billing Country.
contactPhone	String	No	
mobilePhone	String	No	
email	String	No	This parameter is essential in the case that customer wanted to have bills in electronic form.

The system responds with a Boolean value denoting the success or failure of the requested service.

The following table describes the acceptable values for the property ***status*** and their meaning.

Value	Description
0	Suspended
1	Active
-1	Deactivated

4.2. Get Account Information

The intention of Get Account Info service is to inform the “Shop” application about the current status of any given account.

Parameters

Name	Type	Description
accountId	Integer	The id of account

The following table summarizes the properties that will be returned from the Billing as a response to Get Account Info.

Please note that for any property not defined during the account creation with ***createAccount*** service or its value was changed to null with a later ***setAccountInfo*** service, will not appear to the response.

The answer in the success case is:

Return Properties

Name	Type	Description
accountId	Integer	
userName	String	This value has to be unique to the billing system.
password	String	
category	String	
status	String	
balance	Money	
currency	String	
billingAddress	String	
billingCity	String	
billingPostCode	String	
billingCountry	String	
contactPhone	String	
mobilePhone	String	
email	String	

For a description of the semantics for the above properties, please consult the description of properties in ***createAccount*** service definition.

4.3. Set Account Info

The intention of Set Account Info service is to change the value of properties already assigned to an account.

Parameters

Name	Type	Mandatory	Description
accountId	Integer	Yes	The id of account. It is used only for finding the account. By no means it is allowed to change
password	String	No	
category	String	No	
status	String	No	
billingAddress	String	No	
billingCity	String	No	
billingPostCode	String	No	
billingCountry	String	No	
contactPhone	String	No	
mobilePhone	String	No	
email	String	No	

Please note that some of the fields that are considered as mandatory for account creation, are not present here. An example is the username which must be unique throughout the billing system, as well as balance and currency information.

For a complete description of parameter semantics see the parameters table of create account service.

4.4. Get Usage Events

The intention of Get Usage Events is to return a list of usage events for a specific account between a certain period of time.

Parameters

Name	Type	Mandatory	Description
accountId	Integer	Yes	This is the accountId that has been created in the CRM.
fromDate	Date	No	Specifies from which date and time the function will return usage events for the account. If this parameter is not passed, the function returns all the available usage events for the account until toDate (if toDate has been specified) (see Date below)
toDate	Date	No	Specifies until which date and time the function will return usage events for the account. If this parameter is not passed, we will get all usage events for the account from fromDate (if fromDate has been specified) until present. (see Date below)

In case that both fromDate and toDate are missing, the function returns all the usage events for the account specified.

Important!! If we don't want to specify neither fromDate nor toDate, we have to pass an empty array to query Items and not null.

Return Properties

Type	Description
UsageEvent[]	An array of UsageEvent (see UsageEvent below)

4.5. Get Payment Events

The intention of Get Payment Events is to return a list of payment events for a specific account between a certain period of time.

Parameters

Name	Type	Mandatory	Description
AccountId	Integer	Yes	This is an accountId that has been created in the CRM.
fromDate	Date	No	Specifies from which date and time the function will return payment events for the account. If this parameter is not passed, the function returns all the available payment events for the account until toDate (if toDate has been specified) (see Date below)
toDate	Date	No	Specifies until which date and time the function will return payment events for the account. If this parameter is not passed, we will get all payment events for the account from fromDate (if fromDate has been specified) until present. (see Date below)

In case that both fromDate and toDate are missing, the function returns all the payment events for the account specified.

Important!! If we don't want to specify neither fromDate nor toDate, we have to pass an empty array to query Items and not null.

Return Properties

Type	Description
PaymentEvent[]	An array of Payment Event (see PaymentEvent in Glossary)

5. Network Interface

The purpose of this interface is to provide the appropriate methods for service handling, such as service availability requests, service provisioning, service cancellations and any other service related method needed. These methods are using the Authorization and Accounting modules.

5.1. Service Availability Request

The intention of Service Availability Request is to do an enquiry, whether a certain kind of service is available, given a specific account

Parameters

Name	Type	Description
Accounted	Integer	
serviceId	Integer	The Id of the selected Service
serviceName	String	
serviceType	Integer	

The answer in the success case is:

Name	Type	Description
Amount	Number	The current balance of Account
Price	Money	The price of the requested service
serviceProvisioningValid	Boolean	True if this service can be provided to the current account. False otherwise

5.2. Start Service Provisioning

The intention of Start Service Provisioning Request is to start service provisioning given a specific account

Parameters

Name	Type	Description
accountId	Integer	
Username	String	User name of a registered account
Password	String	Password of a registered account
serviceId	Integer	
serviceName	String	
serviceType	Integer	

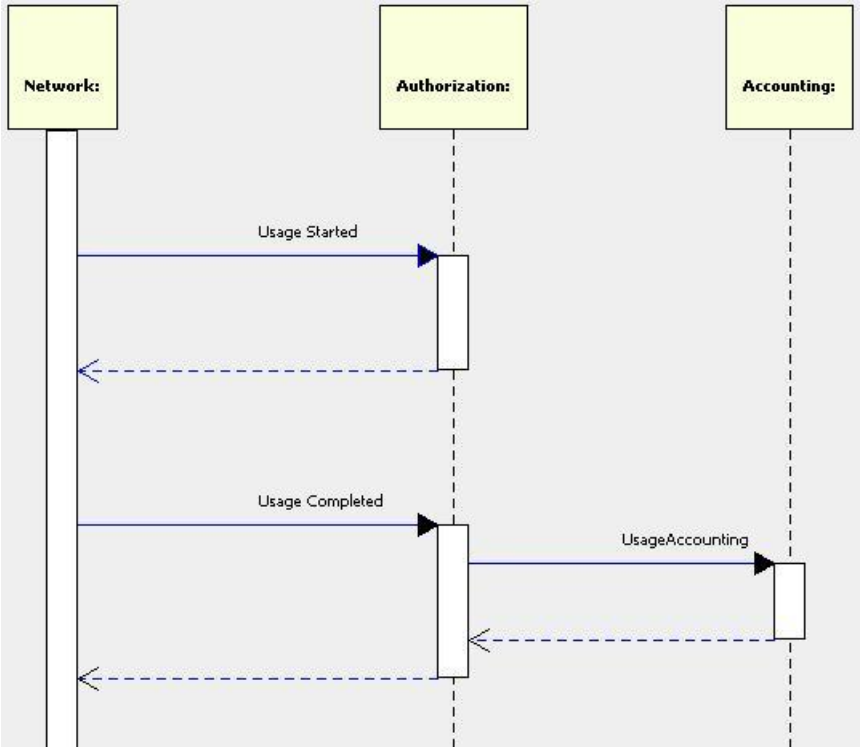
The answer in the success case is:

Name	Type	Description
Amount	Number	The current balance of Account
Price	Number	The price of the requested service
startDate	Date	The date from which , this service is going to be available to this account
endDate	Date	The expiration date of this service for the given account
transactionId	String	Unique id specifying the service provisioning request.

5.3. Service Provisioning Completed

The intention of Stop Service Provisioning Request is to stop service provisioning given a specific account and a transaction id

The complete transaction sequence in a case that a provision of service was completed normally, is explained in the following diagram.



Parameters

Name	Type	Description
accountId	Integer	
transactionId	String	

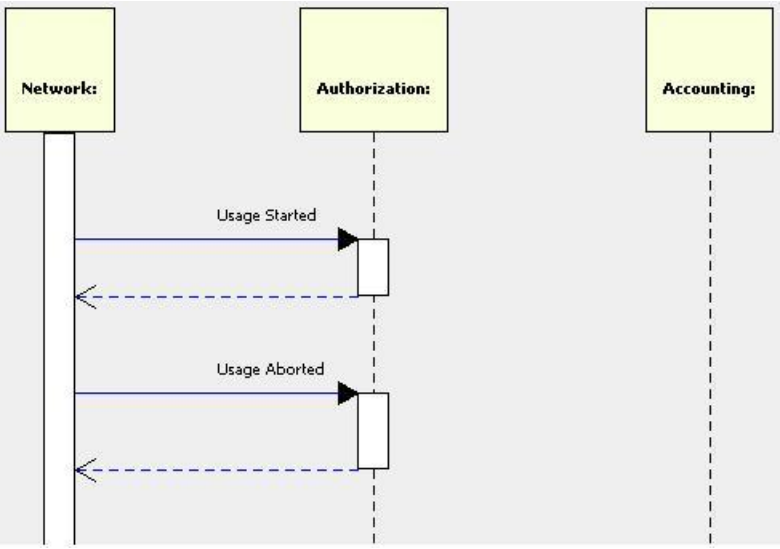
The answer in the success case is:

Name	Type	Description
amount	Number	The current balance of Account
price	Number	The price of the requested service

5.4. Service Provisioning Canceled

The intention of Cancel Service Provisioning Request is to cancel service provisioning given a specific account and a transaction id

The complete transaction sequence in a case that a provision of service was cancelled abnormally, before the complete provisioning is explained in the following diagram.



Parameters

Name	Type	Description
accountId	Integer	
transactionId	String	

The answer in the success case is:

Name	Type	Description
Amount	Number	The current balance of Account
Price	Number	The price of the requested service

6. Accounting Interface

The purpose of this interface is to provide the appropriate methods for accounting actions such as acquiring financial and usage data per customer for a given time period interface. Accounting module is used for this interface's methods.

6.1. Get Account Totals

The intention of Get Account Totals is to return the credit and debit summaries for a specific account between a certain period of time.

Parameters

Name	Type	Mandatory	Description
accountId	Integer	Yes	This is the accountId that has been created in the CRM.
fromDate	Date	No	Specifies from which date and time the function will return financial data for the account. If this parameter is not passed, the function returns all the available financial data for the account until toDate (if toDate has been specified) (see Date below)
toDate	Date	No	Specifies until which date and time the function will return financial data for the account. If this parameter is not passed, we will get all financial data for the account from fromDate (if fromDate has been specified) until present. (see Date below)

In case that both fromDate and toDate are missing, the function returns full summaries for the account specified.

Return Properties

Type	Description
credit	The summary of credit for the given period and account.
debit	The summary of debit for the given period and

	account.

6.2. Get Account Details

The intention of Get Account Details is to return a list of financial data for a specific account between a certain period of time.

Parameters

Name	Type	Mandatory	Description
accountId	Integer	Yes	This is the accountId that has been created in the CRM.
fromDate	Date	No	Specifies from which date and time the function will return financial data for the account. If this parameter is not passed, the function returns all the available financial data for the account until toDate (if toDate has been specified) (see Date below)
toDate	Date	No	Specifies until which date and time the function will return financial data for the account. If this parameter is not passed, we will get all financial data for the account from fromDate (if fromDate has been specified) until present. (see Date below)

In case that both fromDate and toDate are missing, the function returns all the financial data for the account specified.

Return Properties

Type	Description
FinancialData []	An array of Financial Data (see FinancialData in the Glossary)

6.3. Get Totals

The intention of Get All Account Totals is to return the credit and debit summaries for all registered accounts between a certain period of time.

Parameters

Name	Type	Mandatory	Description
fromDate	Date	No	Specifies from which date and time the function will return summaries for all of the accounts. If this parameter is not passed, the function returns summaries until toDate (if toDate has been specified) (see Date below)
toDate	Date	No	Specifies until which date and time the function will return summaries for all of the accounts. If this parameter is not passed, the function returns summaries from fromDate (if fromDate has been specified) until present. (see Date below)

In case that both fromDate and toDate are missing, the function returns full summaries for all registered accounts.

Return Properties

Type	Description
credit	The full summary of credit for all registered accounts and the given period.
debit	The full summary of debit for all registered accounts and the given period.

6.4. Get All Account Totals

The intention of Get Totals is to return a list of credit and debit summaries for all registered accounts between a certain period of time.

Parameters

Name	Type	Mandatory	Description
fromDate	Date	Yes	Specifies from which date and time the function will return summaries for all registered accounts. If this parameter is not passed, the function returns summaries for all registered accounts until toDate (if toDate has been specified) (see Date below)
toDate	Date	Yes	Specifies until which date and time the function will return summaries for all registered accounts. If this parameter is not passed, the function returns summaries for all registered accounts from fromDate (if fromDate has been specified) until present. (see Date below)

In case that both fromDate and toDate are missing, the function returns summaries for all registered accounts.

7. Reference

7.1. Glossary

Term	Definition
WSDL	Web Service Definition Language. An XML based document which describes the available operations for the specific Web Service.
CRM	Customer Relationship Management. A system responsible for the communication with customers. Usually these systems they gather information from other BSS and present an integrated view to users.
Network Element	
North Bound	
South Bound	
J2EE	This is the code name for the Java applications targeted to Enterprise environments and they are ready to deployed in Application Servers.
Web Service	Web Service. This is common language for different technology platforms to communicate through the web, sending messages in XML format, using Http communication protocol
BSS	Business Support System. An abstract terminology for those systems that supports the business of companies. In TelCo industry, such kind of systems are CRM, ERP, Billing e.t.c
OSS	Operation Support System. An abstract terminology for those systems that support the operation of companies. In TelCo industry, such kind of systems are the Network Access Elements, Service providers, e.t.c

7.2. Data Types

The following table describes all the data types that have been referenced in this document. Complex data type they have their own separate description whether this is necessary.

Type	Description
Integer	The usual representation of Integer as described by Java JLS specification
String	The typical String data type.
Numeric	Represents numeric values with decimal point.
Property	Is a set of two strings. The first one shows the name of the value and the second one is the value. (see wsProperty below)
Money	Represents monetary amounts with currency. Its form is a string consisted by a numeric value followed by a space character and the country currency code according to ISO 4217. (e.g. 52.22 EUR)
Volume	Represents Is a string containing a numeric value. In case of a voice call contains the number of seconds the call lasted. In case of SMS contains the number of SMS rated.
UsageEvent	Represents usage events that happened to an account. (see UsageEvent below)
Usage Type	It is a string that describes the type of the usage.
Date	It is a string that describes a certain point of time. The format of the string must be: «yyyy-MM-dd HH:mm:ss»
PaymentEvent	Represents payment events that happened to an account. (see Payment Event below)
FinancialData	Represents finance data they were created as result of either a Usage Event or Payment Event. (see FinancialData below)

7.3. PaymentEvent

A Payment Event is an event that produced from a payment for a specific account. Shylock produces various types of Payment Events. A Payment Event gives detailed information for a specific account concerning the type of service he/she used, the price the customer was charged for this service etc.

Name	Type	Description
paymentId	Integer	A unique id of this payment event
accountId	Integer	This is the accountId of a customer that is stored in the CRM.
amount	Money	
eventType	Integer	1=SMS 2=OTE 3=Credit Card
transDate	Date	The date and time the system was informed about this payment event.
dueDate	Date	The date the payment transaction took place.
transactionId	String	A unique id, used to correlate the specific event between two different systems

7.4. UsageEvent

Any service that is provided to customer creates one or more usage events. These usage events hold detailed information about the type of the provided service.

Name	Type	Description
usageId	Integer	A unique id of this usage event
accountId	Integer	
usageType	String	Type of usage.
usageStartTime	Date	When the usage event started
usageEndTime	Date	When the usage event ended
usageVolume	Volume	
usagePrice	Money	The amount of money, charged to the account.
usageStatus	Integer	Reserved for internal use
transactionId	String	A unique id, used to correlate the specific event between two different systems

7.5. FinancialData

Financial Data is an object holding all the necessary information for a financial system which produced from a “financial” event.

As “financial events” we consider any UsageEvent or PaymentEvent. The Financial Data structure is a virtual structure produced from Shylock

Name	Type	Description
paymentId	Integer	A unique id of this payment event
accountId	Integer	This is the accountId of a customer that is stored in the CRM.
amount	Money	
eventType	Character	C=Credit D=Debit
dueDate	Date	The date the financial event took place.
transactionId	String	A unique id, used to correlate the specific event between two different systems

7.6. Example WSDL

The following WSDL is an example of the operation Get Usage Events. It does not consist a full working example.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://ws.bobss.rege.org"
xmlns:tns="http://ws.bobss.rege.org"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc11="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soapenc12="http://www.w3.org/2003/05/soap-encoding"
xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
      <wsdl:types>
            <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://ws.bobss.rege.org">

                  <xsd:complexType name="SimpleUsageEvent">
                        <xsd:sequence>
                              <xsd:element minOccurs="0"
name="accountId" nillable="true" type="xsd:int"/>
                              <xsd:element minOccurs="0" name="date"
type="xsd:dateTime"/>
                              <xsd:element minOccurs="0"
name="destination" nillable="true" type="xsd:string"/>
                              <xsd:element minOccurs="0"
name="duration" type="xsd:int"/>
                              <xsd:element minOccurs="0" name="price"
type="xsd:double"/>
                              <xsd:element minOccurs="0"
name="source" nillable="true" type="xsd:string"/>
                              <xsd:element minOccurs="0"
name="usageId" type="xsd:int"/>
                              <xsd:element minOccurs="0"
name="usageType" nillable="true" type="xsd:string"/>
                        </xsd:sequence>
                  </xsd:complexType>
                  <xsd:element name="getUsageEvents">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element maxOccurs="1"
minOccurs="1" name="in0" type="xsd:int"/>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:complexType name="ArrayOfSimpleUsageEvent">
                        <xsd:sequence>
                              <xsd:element maxOccurs="unbounded"
minOccurs="0" name="SimpleUsageEvent" nillable="true"
type="tns:SimpleUsageEvent"/>
                        </xsd:sequence>
                  </xsd:complexType>
```

```
                  <xsd:element name="getUsageEventsResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element maxOccurs="1"
minOccurs="1" name="out" nillable="true"
type="tns:ArrayOfSimpleUsageEvent"/>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
            </xsd:schema>
      </wsdl:types>

      <wsdl:message name="getUsageEventsRequest">
            <wsdl:part name="parameters"
element="tns:getUsageEvents">
            </wsdl:part>
      </wsdl:message>
      <wsdl:message name="getUsageEventsResponse">
            <wsdl:part name="parameters"
element="tns:getUsageEventsResponse">
            </wsdl:part>
      </wsdl:message>

      <wsdl:portType name="SimpleCrmAccountManagerPortType">
            <wsdl:operation name="getUsageEvents">
                  <wsdl:input name="getUsageEventsRequest"
message="tns:getUsageEventsRequest">
                  </wsdl:input>
                  <wsdl:output name="getUsageEventsResponse"
message="tns:getUsageEventsResponse">
                  </wsdl:output>
            </wsdl:operation>
      </wsdl:portType>

      <wsdl:binding name="SimpleCrmAccountManagerHttpBinding"
type="tns:SimpleCrmAccountManagerPortType">
            <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <wsdl:operation name="getUsageEvents">
                  <wsdlsoap:operation soapAction=""/>
                  <wsdl:input name="getUsageEventsRequest">
                        <wsdlsoap:body use="literal"/>
                  </wsdl:input>
                  <wsdl:output name="getUsageEventsResponse">
                        <wsdlsoap:body use="literal"/>
                  </wsdl:output>
            </wsdl:operation>
      </wsdl:binding>

      <wsdl:service name="SimpleCrmAccountManager">
            <wsdl:port name="SimpleCrmAccountManagerHttpPort"
binding="tns:SimpleCrmAccountManagerHttpBinding">
                  <wsdlsoap:address
location="http://127.0.0.1:8080/bo-bss-ws-
0.1/services/SimpleCrmAccountManager"/>
            </wsdl:port>
      </wsdl:service>
</wsdl:definitions>
```