Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Web Services Μεταξύ SAP ECC και Εφαρμογές Android** **Web Services Between SAP ECC and Android Applications**
Ονοματεπώνυμο Φοιτητή	**Ευθύμιος-Νταβίντ Αλέπης**
Πατρώνυμο	**Θεόδωρος**
Αριθμός Μητρώου	**ΜΠΠΛ 14004**
Επιβλέπων	**Μαρία Βίρβου, Καθηγήτρια**

Ημερομηνία Παράδοσης **Νοέμβριος 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Μαρία Βίρβου Καθηγήτρια	Ευθύμιος Αλέπης Επίκουρος Καθηγητής	Κων/νος Πατσάκης Επίκουρος Καθηγητής

Table of Contents

Abstract

English

This Thesis focuses on the implementation of web services published by a SAP ECC system and consumed by an android application.
Two web services are implemented on a SAP ECC system and thoroughly presented, one using SOAP and one using REST and a single android application to consume them. A step by step implementation guide is presented for the SAP ECC system and a drill down in the coding used for the android application.
Also, an overview of the components used is present and an introduction to SAP ABAP architecture and design.
The goal of this thesis is to establish an overview of SAP ABAP, SAP ECC and the Android Platform and display in detail how these can work together via Web services.

Ελληνικά

Η παρούσα πτυχιακή επικεντρώνεται στην υλοποίηση web service που δημοσιεύονται από ένα σύστημα SAP ECC και καταναλώνονται από μια εφαρμογή Android.
Δύο Web Services υλοποιούνται σε ένα σύστημα SAP ECC και αναλύονται λεπτομερώς, ένα χρησιμοποιώντας SOAP και ένα χρησιμοποιώντας REST, επεισης υλοποιείτε και παρουσιάζεται μια εφαρμογή Android για να της καταναλώσει. Παρουσιάζεται ένας οδηγός βήμα προς βήμα για το σύστημα SAP ECC και μια παρουσιαση του κωδικα που χρησιμοποιείται για την εφαρμογή Android.
Επίσης, παρουσιάζεται μια επισκόπηση των χρησιμοποιούμενων στοιχείων και μια εισαγωγή στην αρχιτεκτονική και το σχεδιασμό SAP ABAP.
Ο στόχος αυτής της εργασίας είναι να παρουσιάσει μια επισκόπηση των SAP ABAP, SAP ECC και της πλατφόρμας Android και να παρουσιάσει λεπτομερώς πώς μπορούν να συνεργάζονται μέσω των υπηρεσιών Web services.

Introduction

Mobile applications perform more and more tasks both in the everyday life as well as in enterprise environments. Android is the most prevalent Operating system used in today's mobile phones and a multitude of applications exist in the "Play Store" ecosystem.
The integration of these applications with enterprise resource planning software has become crucial for companies worldwide. From master data collection to financial transactions and from point of sale applications to simple user registrations, the data which can be collected and processed in ERP systems of mobile applications is increasingly valuable.
Multiple organizations have abandoned the traditional Desktop or laptop setups and have moved to mobile applications to handle everyday tasks. Sales persons close sales with their tablets and executive staff obtain daily reports in custom-made applications directly on their mobile phones. As the bulk of the data is stored and processed in ERP systems the integration and quick access to the data becomes crucial.
ERPs generally adopt only proven and tested solutions to integrate into their platforms. This explains why Web services have been emerging and becoming mainstream in recent years only. Companies realized that ERP software can't be "Black boxes" but must be integrated with the Internet and all the services it provides.
SAP is one of the biggest Enterprise Software providers in the world. SAP systems provide the tools in required to both consume and publish Web Services. The SAP middleware Process Integration (PO) is specifically designed to handle web services among other things. However, not all enterprise structures see the need to implement an SAP PO system, whether due to cost reasons or to the need for integration of a high number of third party systems and web services. Web Services can also be consumed from the Core ERP software provided by SAP. With the rise of mobile applications, consultants and developers who can implement the consumption and publication of web services from SAP systems are in high demand. In the case of publishing web services, a good working knowledge of the client's systems architecture is needed. In such projects, a big part of the SAP consultant's role is to guide and assist the client's development team in order to ensure seamless integration. Not only is assistance needed as it related to the technical specifications, it is also essential when it comes to the SAP terminology, data structures and data relationships which are relatively foreign to the majority of IT professionals.
The following thesis focuses on the integration of the two leading companies in their respective fields.
SAP AG is the leader in the field of ERP software for businesses and Google Inc., with the help of the android operating system, is the leader of mobile platform operating systems.

SAP and Google

SAP is the world leader in enterprise applications of software especially though their SAP ERP software and through their CRM software. It boasts more than 345.000 customers in over 180 countries and more than 15.000 partnered companies globally. More impressively, SAP AG is the third largest software manufacturer globally.
Founded by 5 IBM employees in 1972 in Germany and called Systemanalyse und Programmentwicklung ("System Analysis and Program Development") it has been in the forefront of software development for enterprise customers since the beginning.
The most popular of SAP products is SAP R/3 closely followed by ne newer SAP S4 HANA which is the platform on which all other parts are build.
SAP software is modular. SAP R/3 is organized into distinct modules, each of them covering typical functions in a business organization. The most commonly used modules are Financials and Controlling (FICO), Human Resources (HR), Materials Management (MM), Sales & Distribution (SD), and Production Planning (PP).
Each module handles the relevant business tasks on its own but is fully integrated and linked with the other modules. For example, a purchase order of materials is handled through the MM module, the invoice for the received goods is handled of the FI module, the sales are handled by the SD module and the invoice for the sale is again handled by the FI module and CO module. SAP recently produced specialized modules, referred to as IS or Industry Specific, some of them are IS-T (Industry Specific – Telecommunications) IS-U (Industry Solutions-Utilities)

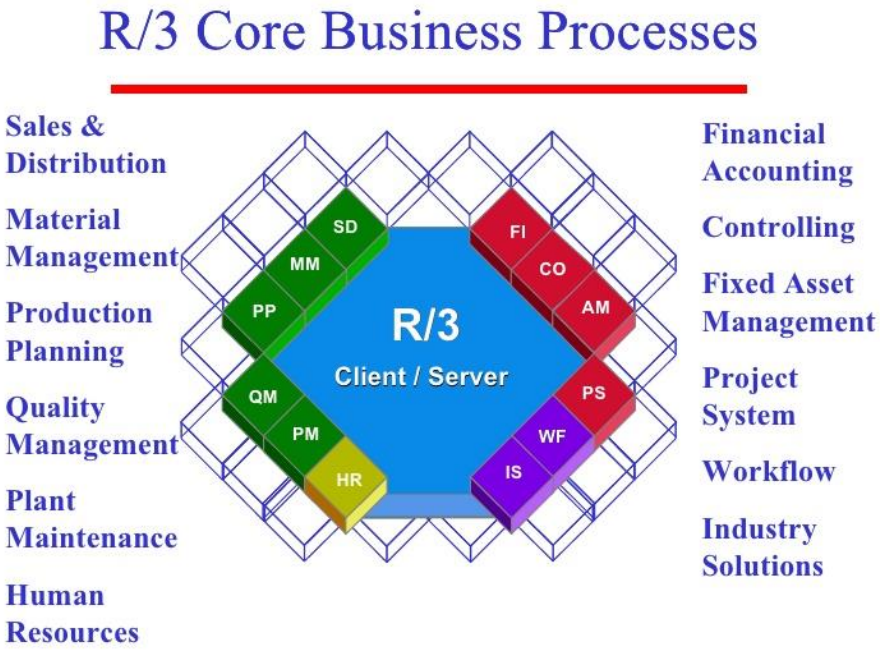

Figure 1: R/3 Core Business Processes, Source: ://www.slideshare.net/Agcristi/sap-r3-381369

Alphabet Inc., and its better-known subsidiary Google, specializes in internet related services and products, and among others is the global leader in mobile device Operating systems with a staggering 85% of mobile devises using Android OS.

Period	Android	iOS	Windows Phone	Others
2016Q1	83.4%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%
2016Q4	81.4%	18.2%	0.2%	0.2%
2017Q1	85.0%	14.7%	0.1%	0.1%

Source: IDC, May 2017

Founded in 1998 by Larry Page and Sergey Brin Google has been in the forefront of technological advancement from the beginning. Google offers Web based services like Gmail,, Google calendar, Google maps and many other from which the first product and the most notable one is the Google search engine.
In on September 23 of 2008 Google releases the first version of the Android operating system after acquiring Android Inc. in 2005. Despite the initial reluctance of mobile phone manufacturing companies, the Android operating system proved the most installed OS in mobile devices in the following years.
Android variations for tablets IOT devices, TVs and even Android for car infotainment systems followed.
The introduction of Google play on the android platform cemented the dominance of the OS in the mobile market even more. Over 2.7 third party applications are published on play store as of February 2017. Companies chose to develop their own applications to serve their customers and integrate their systems with apps of other companies. Some which operate in the technology sector have even proceeded in integrating android apps fully meaning that company employees perform their daily tasks without ever logging in the back-office systems but only though custom-made applications. The integration between Android and ERP seems more relevant than ever.

ABAP

ABAP is the programming language SAP developed to build its platform and enable the creation of custom programs and to enable enhancement of SAP Applications from its customers.
ABAP stands for **A**llgemeiner ***B****erichts****A****ufbereitungs****P****rozessor*, German for "generic report preparation processor" and was renamed later to English to **A**dvanced **B**usiness **A**pplication **P**rogramming.
In 1999 SAP released ABAP Objects. This was an Object-Oriented extension to the existing Procedural ABAP.
The development platform for SAP ABAP is The Netweaver Platform which additionally supports JAVA development.
SAP Netweaver runs on a few different platforms. More notably AIX, HP-UX Solaris and linux in the Unix family, AS/400 and S/390 form the IBM system family and Microsoft Windows i5/OS.
The databases supported are IBM DB2, Informix, MaxDB, Oracle, and Microsoft SQL Server.

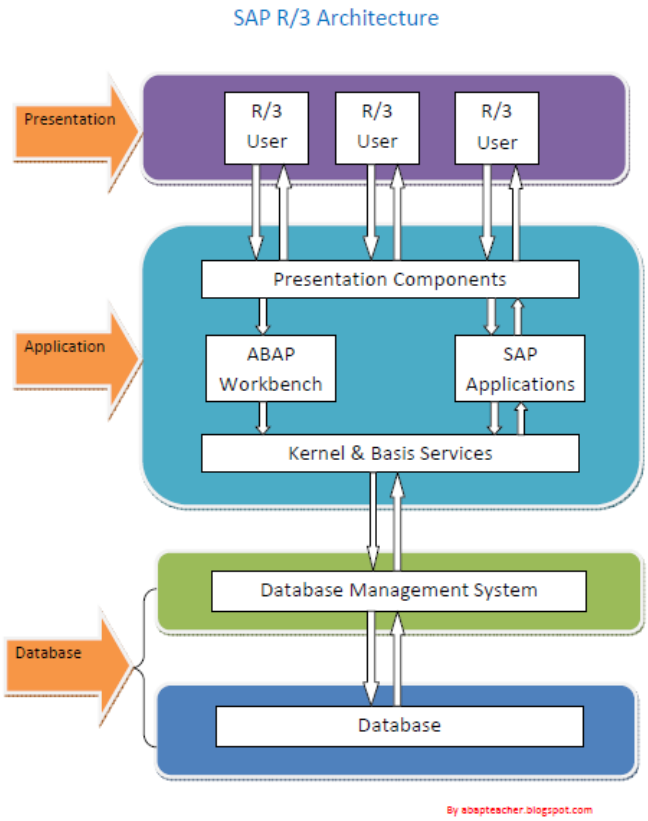

Figure 2 SAP R/3 Architecture Source: Shubhakar.com

All SAP software and data runs in the context of a SAP system. A SAP system consists of a relational database and at least one application instance, the application server. Typically, a SAP system has more than one application server and load balancing technologies are applied for performance reasons.
The most usual landscape for a SAP implementation cosists of three systems. A Development System, were all new implementations are performed. A Quality Assurance System, were all the Unit Testing is performed. And the Production system which is the system the company actually uses. The transport of objects from one system to another as well as the concurrency control is

managed by the Change and Transport system (CTS). A Simple example of the usage of the CTS system will be presented in this thesis as part of the ABAP development of the sample Web Service.
Each individual SAP system consists of three layers which may run on the same or on different hardware. The database layer, which in turn consists of the relational database and the database software. The application layer which contains the instances of the system. And the presentation layer which is responsible for the User interface. The proprietary Graphic interface of a SAP system is the SAP GUI. All further screen shots of sap systems are images of the SAP GUI component.

Android

Figure 3Why choose android, Source: tutorialspoint.com

Google Android provides a robust framework for the development of applications in a java environment which can run in multiple android enabled devices, from mobile phones and tablets to smart TVs and smart mirrors.
The most widely used tool for developing Android applications is Intelij's Android Studio which is currently in its 3.0 version.
The Android OS is based on the Linux kernel and was initially developed to work on touchscreen devices. The source code is released under an open source license agreement, however most android devices ship with a mix of the open source OS and proprietary software depending on the manufacturer
The android Operating system receives regular updates. The latest release is the OREO release, Released on August 21 2017.
Application development is done using the Android software development kit primarily and can also be performed in conjunction with programming languages like C# and JAVA.

Figure 4 The Android Stack Source: en.wikipedia.org

Web Services

As the internet evolved, the idea of web services developed. The intent behind a web service is to use the internet as a transactional tool and not simply as a data visualization tool. By web service we mean any application to application interaction for which the data are transferred through the internet. In order to ensure universal conformity in communications, standards were developed to rule over the web transactions.
Some of these standards are:
Extensible Markup Language (XML)
Hypertext Transfer Protocol (HTTP)
SOAP
REST
Universal Description, Discovery, and Integration (UDDI)
Web Services Description Language (WSDL).
All the above standards are pieces of a web service implementation and solve many problems related to communications between systems.
In scope of this thesis, two if these standards of web service communication will be examined in more detail and used in the sample implementation. Simple Object Access Protocol or SOAP and Representational state transfer or REST.

SOAP

Simple Object Access Protocol or SOAP is a Protocol for exchanging structured, usually XML-based, information. It was first introduced in 1998 by Dave Winer, Don Box, Bob Atkinson, and Mohsen Al-Ghosein and has since been used extensively in the communication via web services. As defined by the World Wide Web Consortium (W3C) SOAP message consist of a document information item with exactly one child the SOAP envelope which contains,
A local name of envelope,
A namespace name,
Zero or more namespace-qualified attribute information items,
One or two child element information items, the header which is optional
and the body which is mandatory.

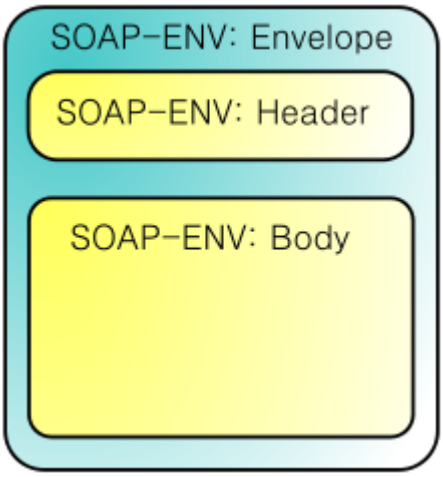

Figure 5A SOAP Message Source: en.wikipedia.org

SOAP Advantages:

- SOAP is independent of the platform.
- In SOAP encoding and communications protocol are independent to the environment they run on. A SOAP web service can be consumed and the source platform is irrelevant to the client implementation.
- As with the source platform, the programming language the web service is implemented on is irrelevant for the clientXML and WSDL are used for data transportation.
- Because the generation of an XML is an easy and thoroughly documented process for any language, the entry barrier is virtually nonexistent.
- SOAP uses the standard and extremely wide spread internet HTTP protocol.
- SOAP runs on HTTP, so there aren't any firewall problems. SOAP can be a protocol for exchanging information in an decentralized and distributed environment.
- SOAP is transport protocol independent.

SOAP Disadvantages:

- Because of the XML format is detailed, SOAP might slower than competing solutions technologies. This usually not an issue when small messages are sent because the differences are negligible.
- The HTTP formatting is lacking. This is because SOAP does not use HTTP methods like GET and POST to format the message
- When relying on HTTP as a transport protocol, if the firewall is designed to only allow web browsing, a more detailed breakdown of the received packages needs to be performed.

REST

While SOAP is a protocol of communication, REST actually is an architectural style for network-based software.
The REST acronym stands for Representational State Transfer, and as the name states, its resources are stateless. Stateless meaning that the server publishing the web service needs to be able to fully understand the request without any further context.
One further characteristic is that the request needs to be cacheable. Data marked as cacheable are able to be reused for a response if a subsequent request is the same.
All components of a REST request need to interact with the same interface. This means that changes in the implementation can be done independently.
Rest resources are represented by some media type. The media type may be XML, JSON, RDF and many others. The preferred protocol for REST services is over HTTP due to simplicity and easy mapping to RESTfull principles. When the HTTP protocol is used, standard HTTP operations e.g. GET, PUT, POST, DELETE are used to manipulate and request via standard interfaces.

Advantages of SOAP compared to REST

SOAP provides the following advantages compared to Rest.

- SOAP is transport independent while REST requires the use of HTTP Protocol as a transfer medium.
- SOAP works well in well in distributed enterprise environments while when using REST a point to point communication is required.
- Provides significant pre-build extensibility by using standards like WSDL

Advantages of REST compared to SOAP

- No expensive tools require to interact with the Web service
- Smaller learning curve
- Efficient (SOAP uses XML for all messages, REST can use smaller message formats)
- Fast (no extensive processing required)
- Closer to other Web technologies in design p
- hilosophy

Sample Application Implementation

In order to demonstrate the implementation and consumption of SAP ECC web services technically, an application was created.
Two different web services were created on a SAP ECC system and a single Android application to consume them. The Web Services are of a similar functionality one using REST and the other SOAP.
The applications functionality consists of the retrieval of the business partner number from a sap system given the Contract account.
In the case of the rest service in order to present the passing of multiple different values in a single JSON, the Conversion Flag was added.
If the conversion field is set the returned Business partner is in "External Format" meaning it is converted form the SAP internal format, which fills all empty characters in front of the business partner number with the character 0, to one without 0.

Android Implementation

The android app was built on Intelij's Android Studio 3.0.
Since the android app is to demonstrate the functionality of the communication between and android device and a SAP ECC system, a simple user interface was designed with a Text View element, two switches and a Button.
The TextView is so the user can input the Contract Account , the switches to choose which Communication method the app will use and the button to initiate the call to the SAP ECC system.
The web service response will appear on a Toast message as soon as it is received for both communication messages.
Since the switches ensure the communication method, only one switch can be enabled at one time

```
SOAP.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton SOAP, boolean isChecked) {
        if (isChecked){
            REST.setChecked(false);
        }
    }
});
REST.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton REST, boolean isChecked) {
        if (isChecked){
            SOAP.setChecked(false);
        }
    }
});
```

On the press of the Call Web service button the corresponding web service call is initiated taking the string input of the user as a parameter.

Implementation of a REST request

For the REST implementation, The Volley library was used. To include the volley library to the application the following was added to the applications gradle.

```
compile 'com.android.volley:volley:1.0.0'
```

A separate class was created to handle the request itself and another to build the JSON to be send.
The call to the web service is initiated by a call of method CallREST passing he main activities Context
.

```
try {
    CallResponce = RESTClass.callREST(context);
} catch (JSONException e) {
    e.printStackTrace();
} catch (AuthFailureError authFailureError) {
    authFailureError.printStackTrace();
}
```

A try catch implementation is necessary in order to catch possible authorization errors the request may receive.
First step of this implementation is to set up a request queue
.

```
RequestQueue queue = Volley.newRequestQueue(context);
```

Then setting up a StringRequest to add to the created queue. An OnResponse listener is also attached to the string request to capture the response JSON.
.

```
StringRequest stringRequest = new StringRequest(Request.Method.POST, url,
                    new Response.Listener<String>() {
                        @Override
                        public void onResponse(String response) {
                            answer = response;
                        }



//                          }
                    }, new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {

                }
            })
```

Now to implement the attachment of the JSON data to the request string.
The request string is attached in the body of the request so an Override of the getBody method is implemented.

```
@Override
public byte[] getBody() {
    JSONObject JSON = null;
    try {
        JSON = JSONCreate.CreateJSON(vkont,conversion);
```

```
    } catch (JSONException e) {
        e.printStackTrace();
    }
```

As shown the JSON to be attached is created by a method CreateJSON and takes as input the Vkont(Contract account) and whether we want a conversion of the format of the returned business partner or not

.

```
public static JSONObject CreateJSON(String vkont, String conversion) throws
JSONException {

    JSONObject requestJSON = new JSONObject();
    requestJSON.put("VKONT", vkont);
    requestJSON.put("CONVERSION", conversion);
    return requestJSON;
```

The above code adds the values passed to JSON nodes VKONT and CONVERSION.
getBody overwritten method then adds it to the string request in byte format.
The authorization is passed to the header of the request. For that method getHeader needs to be overwritten. The Username and password need to be the same as the SAP ECC User.
Basic authorization is passed to the request with the following code.

```
    @Override
    public Map<String, String> getHeaders() throws AuthFailureError {
        Map<String, String> headers = new HashMap<>();
        String credentials = "[redacted]:[redacted]";
        String auth = "Basic "
                + Base64.encodeToString(credentials.getBytes(), Base64.NO_WRAP);
        headers.put( k: "Content-Type", v: "application/json");
        headers.put( k: "Authorization", auth);
        return headers;
    }
```

The credentials need to be separated by ":" with the Username in front, being non case sensitive and the password after the : being, naturally, case sensitive.
Here the actual Username and password are being hidden for security purposes
All that remains is for the request to be added to the queue and for the queue to be started.
First the addition to the queue,

```
queue.add(stringRequest);
```

And then the queue is started.

```
queue.start();
```

Once the queue is started the onResponce listener takes over and whatever response the application gets from the SAP ECC server is returned.
The response from SAP ECC is a String in JSON format. In order to parse the data it contains first the string is cast to a JSON object. A try-catch implementation is necessary in case the string is not convertible to JSON format.

```
JSONObject JSONObj = null;
try {
     JSONObj = new JSONObject(CallResponce);
} catch (JSONException e) {
    e.printStackTrace();
}
```

The individual values of each node of the JSON are then retrieved by using

```
try {
    ls_gpart = JSONObj.getString("GPART");
} catch (JSONException e) {
    e.printStackTrace();
}
```

Again a try catch implementation is necessary in order to ensure that the nodes existence does not lead to an unhandled error.
The value of the desired node, in this example the Business Partner GPART is displayed to the User in a Toast message.

Implementation of a SOAP request

In order to implement the web service call using SOAP, library kSOAP was used.
kSOAP2 is made up of an XML parser, a de/serializer, and a transport layer.
To Import this library to the application the following code was added to the applications gradle.

```
compile 'com.google.code.ksoap2-android:ksoap2-android:3.6.1'
```

and the following repository

```
repositories {
    maven { url 'https://oss.sonatype.org/content/repositories/ksoap2-android-releases/' }
}
```

In order to handle the request call, two new classes were created, SOAPclass and HttpTransportBasicAuth.
SOAPclass has a single method witch both created the request and performs it.
Class HttpTransportBasicAuth extends to KSOAP2 Libraries HttpTransportSE and performs the addition of Basic authorization to the request encrypted in Base64.
The first step of the implementation is to create a new SOAPObject. In order to create this SOAPObject the NameSpace and the Method name need to be defined so that the object gets the request WSDL format.
This was done with the following,

```
SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
```

Setting the corresponding NAMESPACE and METHODNAME variables to the ones provided by the SAP ECC system.
The second Step is to fill the WSDL with the right properties. In our case the WSDL has only one parameter I_VKONT which stands for the Contract account.
With the following code, first the property info is created, then values are added and lastly the property info is added to the request which was created as a whole.

```
PropertyInfo wsdl = new PropertyInfo();
wsdl.setName("I_VKONT");
wsdl.setValue(Vkont);
wsdl.setType(PropertyInfo.STRING_CLASS);
request.addProperty(wsdl);
```

The final step of the creation of the request is to serialize the data in a format suitable for sending. This is done with thew following code,

```
SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
envelope.setOutputSoapObject(request);
```

Now that the request data is parsed and ready, the implementation of the request itself is the next step. Because the Http request needs to have BASIC Authorization the aforementioned class HttpTransportBasicAuth is implemented with the desired username and password passed.

```
HttpTransportBasicAuth aht= new HttpTransportBasicAuth(URL, username: "TALEPIS", password: "0071989tim");
```

And the correct credentials are passed to the call.

```
private String username;
    private String password;

    public HttpTransportBasicAuth(String url, String username, String
password) {
        super(url);
        this.username = username;
        this.password = password;
    }

    public ServiceConnection getServiceConnection() throws IOException {
        ServiceConnectionSE Connection = new ServiceConnectionSE(url);
        addBasicAuthentication(Connection);
        return Connection;
    }

    protected void addBasicAuthentication(ServiceConnection Connection) throws
IOException {
        if (username != null && password != null) {
            StringBuffer buf = new StringBuffer(username);
            buf.append(':').append(password);
            byte[] raw = buf.toString().getBytes();
            buf.setLength(0);
            buf.append("Basic ");
            org.kobjects.base64.Base64.encode(raw, 0, raw.length, buf);
            Connection.setRequestProperty("Authorization", buf.toString());
        }
} }
```

The web service is the called within a try catch statement. The response of the web service is received in a SOAPPrimitive format and then converted to a string.

```
try
    {
        aht.call(SOAP_ACTION, envelope);
        SoapPrimitive  resultsRequestSOAP = (SoapPrimitive)
envelope.getResponse();
        answer = resultsRequestSOAP.toString();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return answer;
} }
```

The web service response is then passed to the main activities variable and displayed as a Toast message.

SAP ECC Implementation

The SAP Internet Communication Framework is the framework were all communications between ABAP and the internet take place, provided those communications use HTTP, HTTPS or SMTP as protocols.
ABAP programs do not directly communicate with the Internet Communication Framework (ICF), ABAP programs communicate with wrappers like Web Dynpro ABAP, ABAP web services and recently ODATA Services to ensure that all necessary conversions are met.
In order to adapt ICF to consume and host REST web services interface and classes of the ABAP REST Library from package SREST are used.

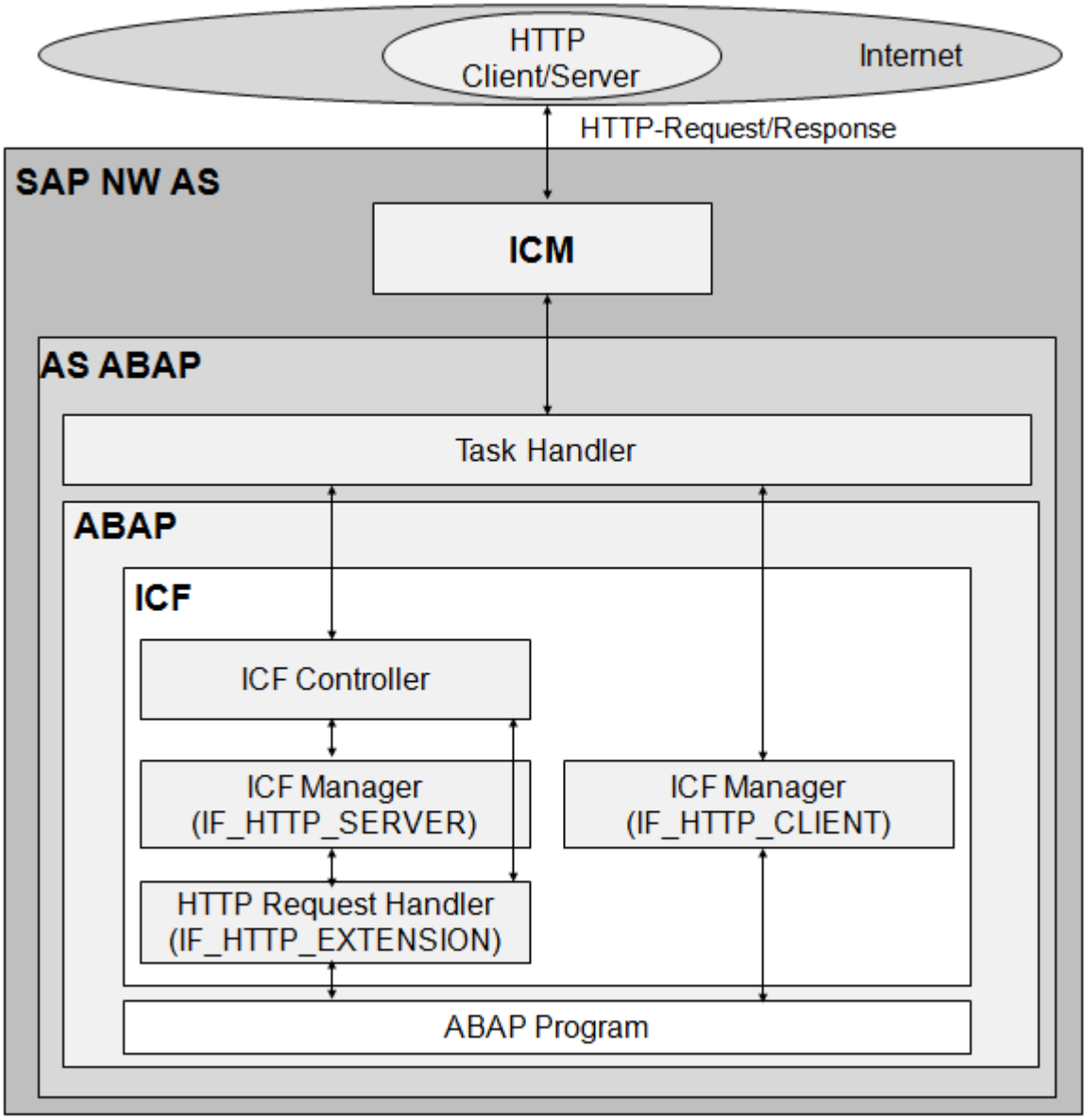

Figure 6 ICF Overview Source: help.sap.com

Each SAP ABAP web service using the ICF framework needs to be created as a node in the ICF tree. The initial node of the ICF represents the Application Servers web Address (the host and the port). Each further node describes the addition to the URL needed to access the web service.
Each service in the ICF tree needs to have at least one class assigned. This class needs to implement interface IF HTTP_EXTENSION. This implemented interface has only one method, the HANDLE method. HANDLE method contains input parameter server which is a reference to the server object IF_HTTP_SERVER. The attributes of this object make possible the manipulation of the request and the passing of values to the response whether it is in HTTP or XML format. Th attributes used to handle the request and response respectively are attributed REQUEST and RESPONSE.
Each HTTP/HTTPS request is handled by SAP application server as a separate ICF session.

Implementation of REST request handling

The first step to create a REST web service is to create the handler class. As discussed above, this class needs to implement the fundamental ICF interface IF_HTTP_EXTENSION.
The SAP ECC Class builder is accessible via SAP transaction se24.

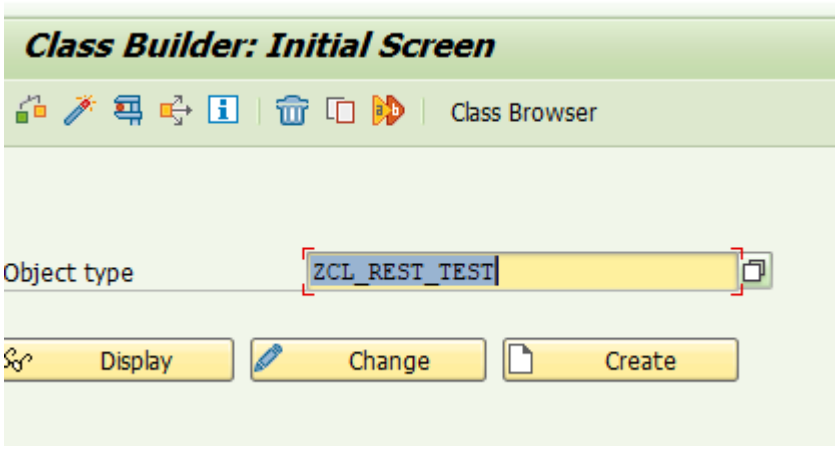

Figure 7SAP ECC Class Builder

We create our test class which will act as the handle of our web service. Here named ZCL_REST_TEST. Naming of ABAP objects is very important in SAP ABAP implementations for business continuity purposes. Every custom object created starts with the letter Z, and the name needs to be descriptive of the functionality performed.
Once within the class builder creation process the right customization is performed.
First customization Tab needs to be filled with informational data with the package being the most important one which will be discussed later.

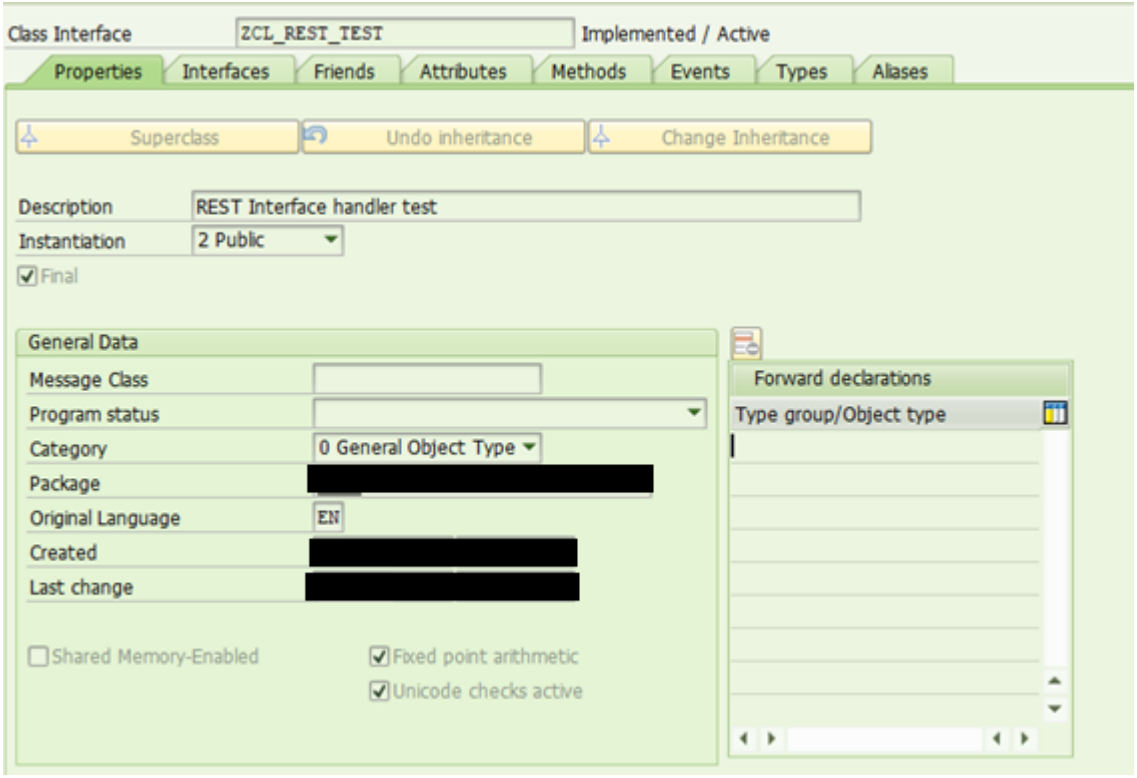

Figure 8Class Builder Properties

Secondly, as discussed previously the handler class needs to implement interface IF_HTTP_REQUEST. This is performed though the interfaces tab.

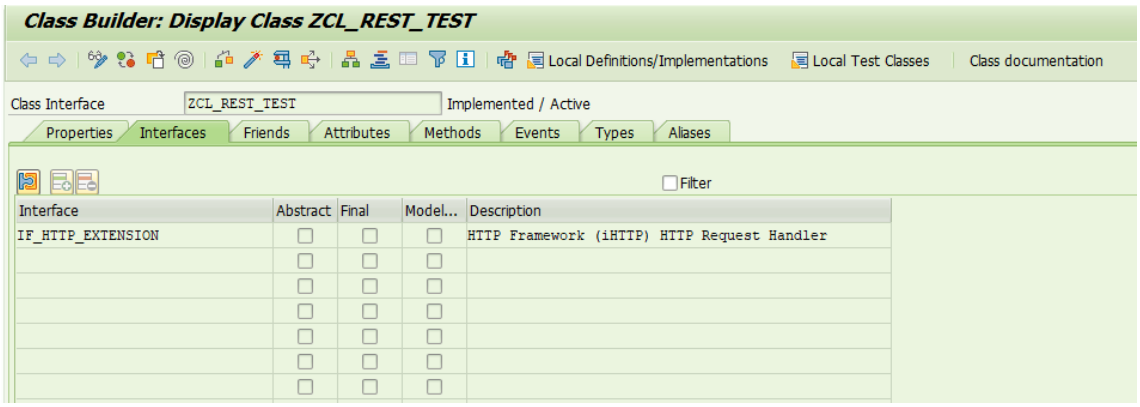

Figure 9Class Builder Interfaces

After this interface is added, the corresponding attributed and the only method are automatically populated in the corresponding tabs.

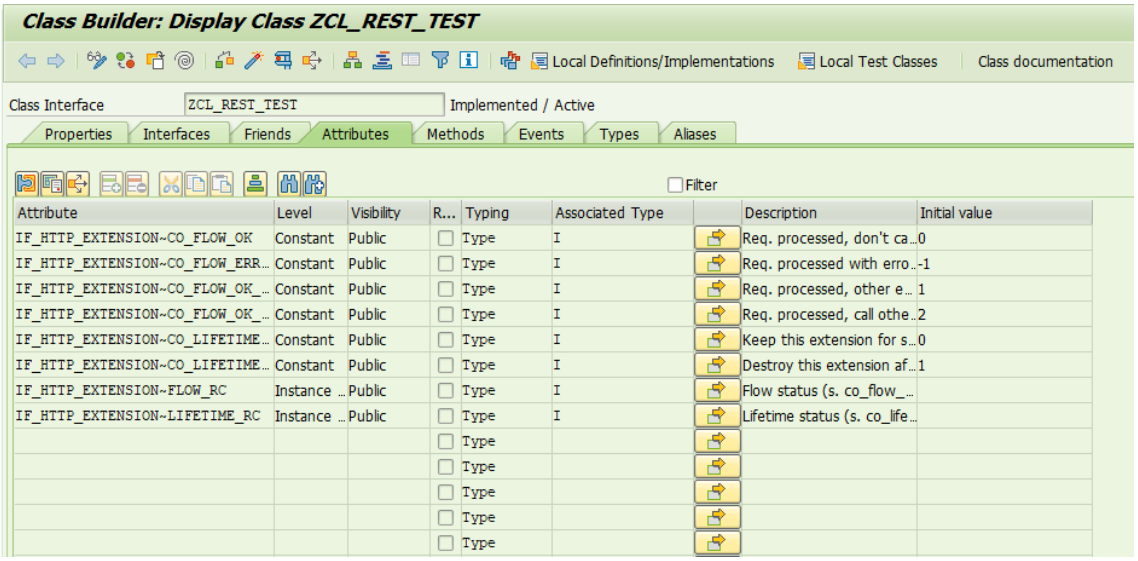

Figure 10Class Builder Attributes

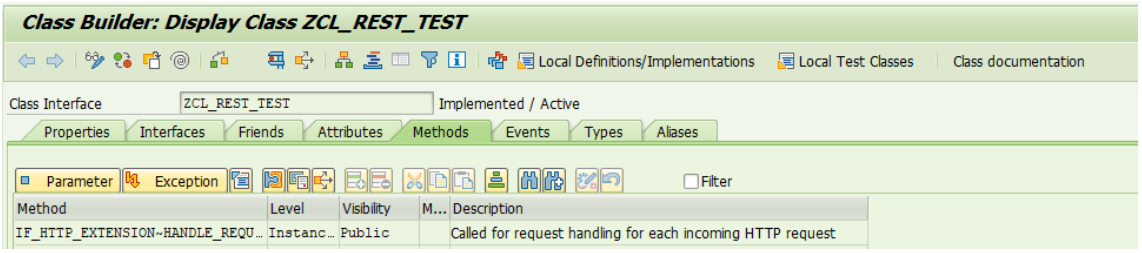

Figure 11Class Builder Methods

Now all customization of the class has been performed and the next step will be to implement the ABAP code with will be executed every time a REST request will reach the web service. The method needing implementation is the HANDLE_REQUEST method of class IF_HTTP_EXTENSION. The ABAP editor is available after double clicking the method in the Method Tab.

The first step in the ABAP implementation is the get the data the request has attached. For this the server attribute is used and the data is passed to a string variable lv_datac.

```
" Get request
  lv_datac = server->request->get_cdata( ).
```

The next step is to parse the request values to ABAP objects. Meaning ABAP structures and ABAP variables. This is performed by calling standard SAP ABAP statement CALL TRANSFOMATION.

```
CALL TRANSFORMATION id SOURCE XML lv_datac
                          RESULT VKONT = ls_JSON-vkont
                                 CONVERSION = ls_JSON-conversion.
```

This statement passes the values of JSON nodes VKONT and CONVERSION to the ABAP structure ls_JSON corresponding vkont and conversion fields.
These field as declared as strings so before performing any database selections with them we need to cast them to the correct internal ABAP types.
The cast is done by simply passing the values to variables declared in the correct type.
Here the correct variable type is data element vkont_kk.
A data element is a SAP ABAP data type and needs to be declared for every field.

In order to extract the Business partner, we could either perform the selection on the database directly or use a function module created for this reason. The preferred method is to create a function module for reusability and code structure purposes.

A simple function ZGET_GPART is created and called.
Creating is done via transaction se37.

The function module imports a vkont (Contract account Number)

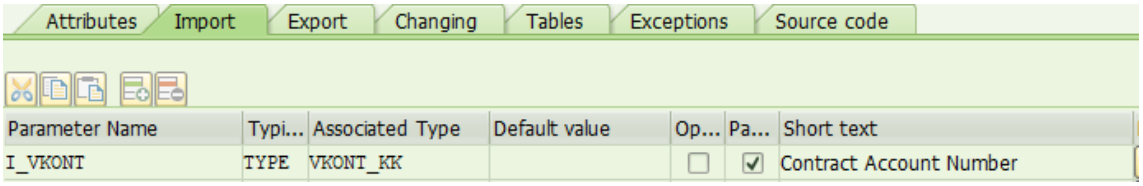

Figure 12Function Module Import Parameters

and returns a GPART (Business Partner)

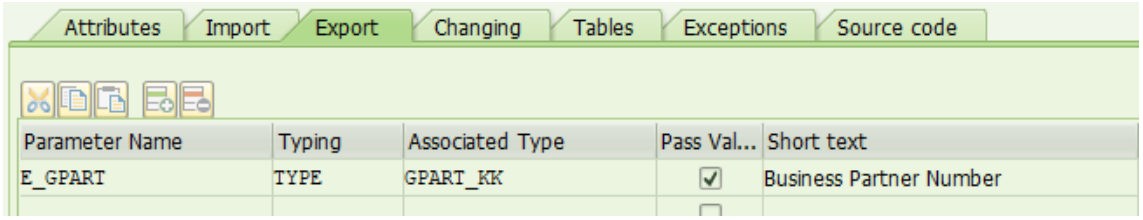

Figure 13Function Module Export Parameters

after performing the correct selection on Standard SAP table FKKVKP.

```
FUNCTION ZGET_GPART.
*"----------------------------------------------------------------------
--
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_VKONT) TYPE  VKONT_KK
*"  EXPORTING
*"     VALUE(E_GPART) TYPE  GPART_KK
*"----------------------------------------------------------------------

 SELECT SINGLE gpart
        INTO e_gpart
        FROM fkkvkp
       WHERE vkont = i_vkont.

ENDFUNCTION.
```

Table FKKVKP has been chosen because it contains the information we are trying to extract and because field vkont with which we perform the selection is the first field in the table key, hence the selection process is very quick.

Back in the web service handler method the newly created function module is called with the correct parameters.

```
CALL FUNCTION 'ZGET_GPART'
    EXPORTING
      i_vkont = lv_vkont
    IMPORTING
      e_gpart = lv_gpart.
```

Now that the response data have been extracted from the database, the next steps are to add them to a JSON and pass the JSON itself to the web service responce.
The JSON is created in byte format directly again by using the Standard SAP statement CALL TRANSACTION but the input needs to be in String format.
The output of CALL TRANSACTION needs to be passed into a string writer.
For this reason a new string writer is created referencing class cl_sxml_string_writer.

```
DATA writer TYPE REF TO cl_sxml_string_writer.
```

So the returned Business Partner number is cast to string and passed to the statement.

```
  lv_gpartc = lv_gpart.
  writer = cl_sxml_string_writer=>create( type = if_sxml=>co_xt_JSON )
.
  CALL TRANSFORMATION id SOURCE gpart = lv_gpartc
                         RESULT XML writer.
```

As displayed the writer is set to create a string in JSON format.
Now the xstring, the string in byte format, needs to be extracted from the writer.

```
lv_responcex = writer->get_output( ).
```

The JSON creating is complete and all that remains is to pass it into the response.
First the content type is set .

```
server->response->set_content_type( 'application/JSON' ).
```

And then the JSON is added to the response.

```
server->response->set_data( lv_responcex ).
```

The response will be send as soon as the "end method". statement is reached.

Implementation of SOAP request Handling

The first step in implementing a SOAP web service is implementing the actual code which will be executed when the web service is called.
In SAP ECC we will create a function module for this purpose.
Function module creation is performed through SAP transaction SE37.

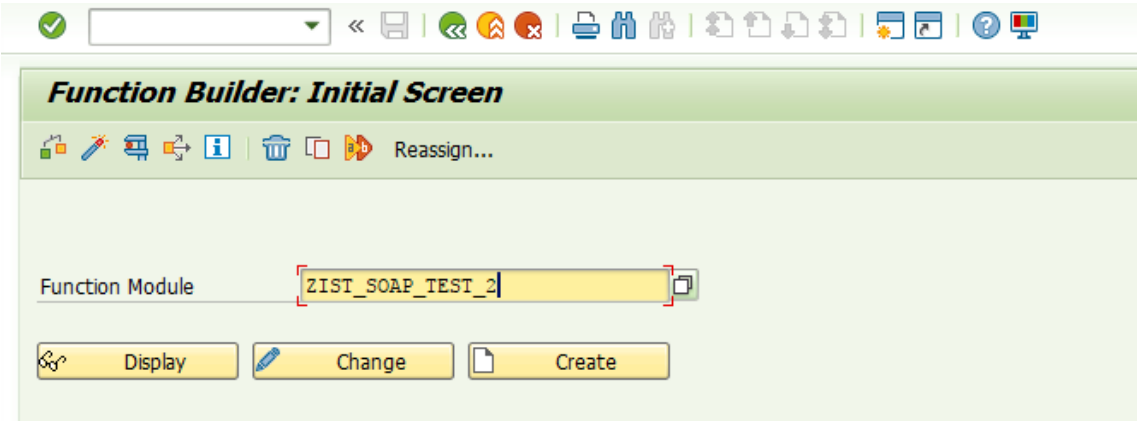

Figure 14Function Builder se37

This function module will import the Contract Account (VKONT) and export the Business Partner (GPART).

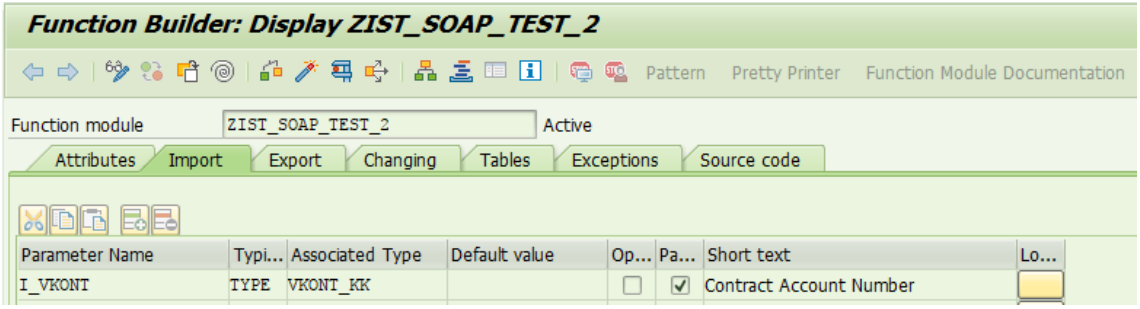

Figure 15Function Builder Import Parameters

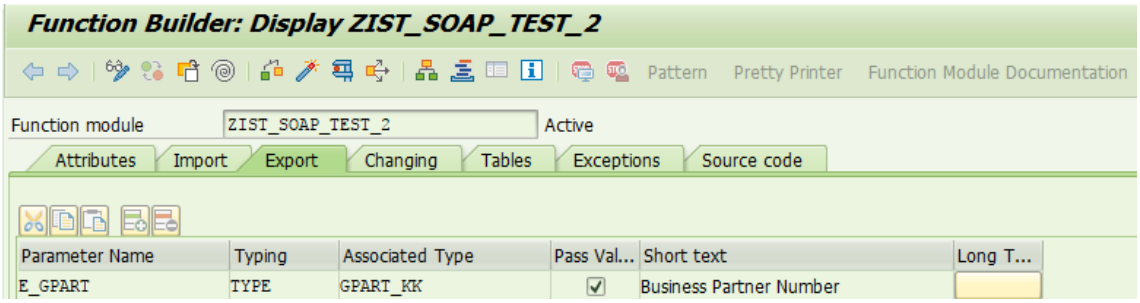

Figure 16Function Builder Export Parameters

The selection of the GPART happens with a simple select on table FKKVKP.

```
FUNCTION ZIST_SOAP_TEST_2.
*"----------------------------------------------------------------------
--
*"*"Local Interface:
*"  IMPORTING
*"     VALUE(I_VKONT) TYPE  VKONT_KK
*"  EXPORTING
*"     VALUE(E_GPART) TYPE  GPART_KK
*"----------------------------------------------------------------------
--

 SELECT SINGLE gpart
        INTO e_gpart
        FROM fkkvkp
       WHERE vkont = i_vkont.

ENDFUNCTION.
```

Now the next step is to publish this function module as a web service.
This is done via a SAP wizard accessible as shown in the bellow depiction.

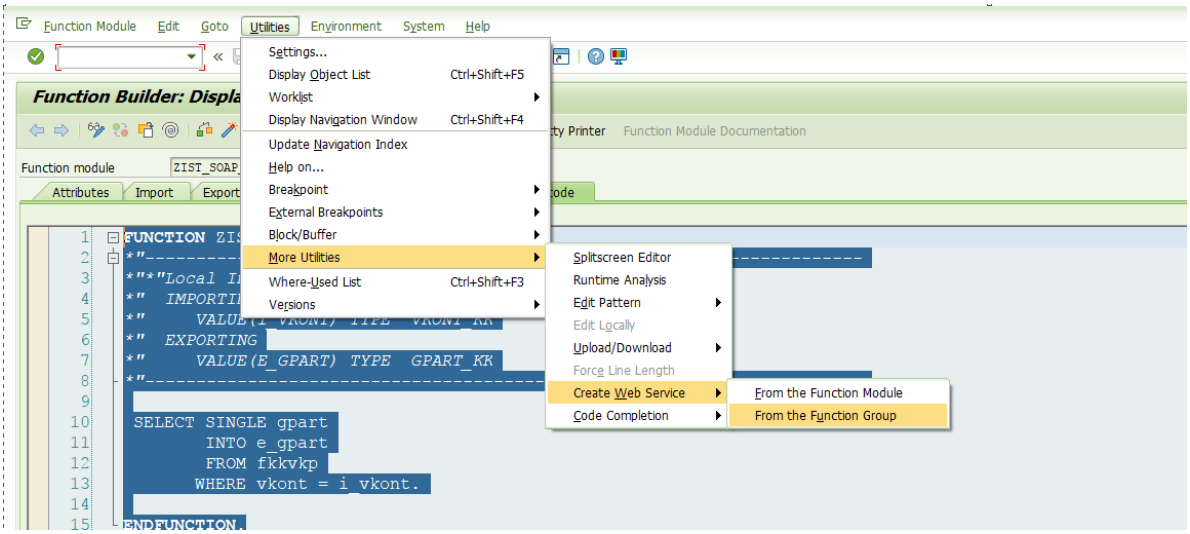

Figure 17Publish Function Module Menu

And the setup is straight forward as shown in the bellow images of the wizard

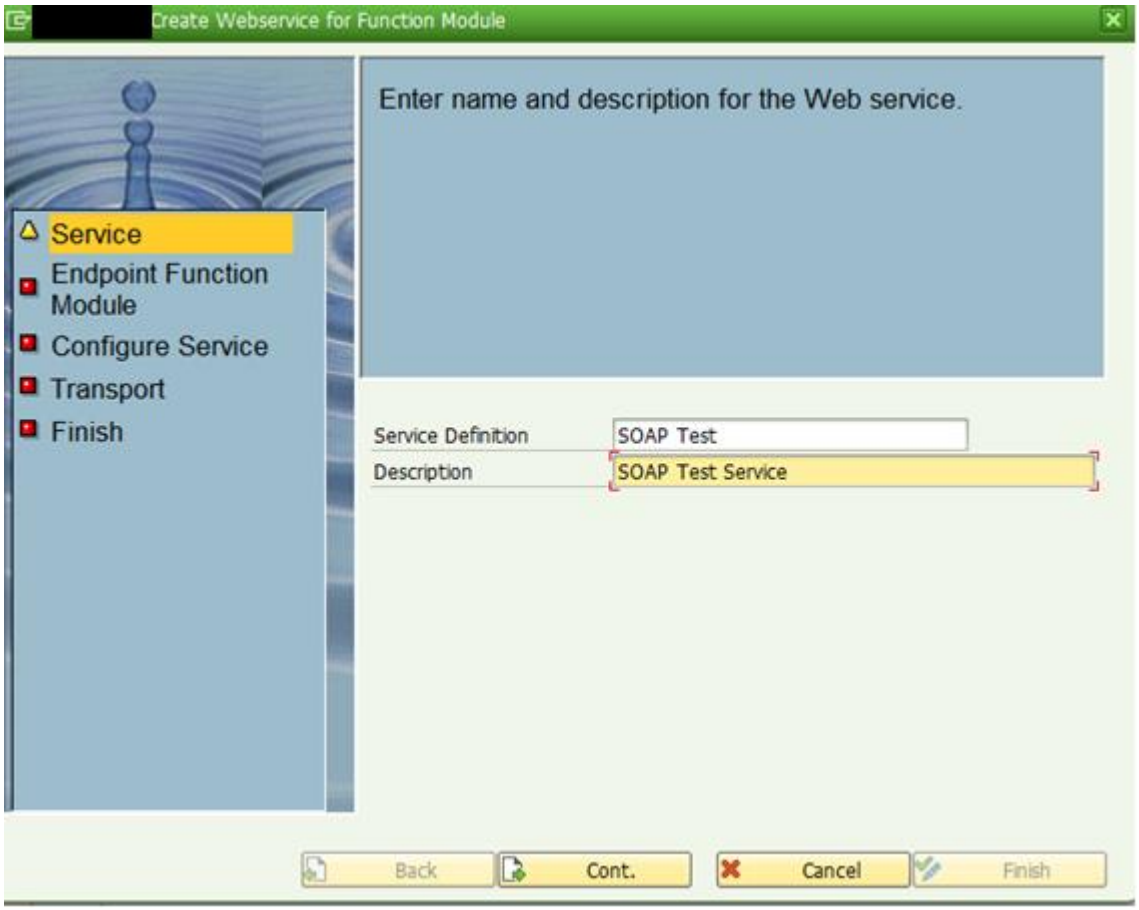

Figure 18Create Web Service for Function Module Wizard

Figure 19Create Web Service From Function Module Wizard Security Profile

As shown an Authentication process with Username and Password is chosen. This username and password is The SAP username and password. This way it can be ensured that from a security perspective only application which have a unique user in the SAP system can access the web service and the build in credential manager will handle the access and the authorization of each request. It could be implemented for example that the user SAP user we create for the web service to use has access only to this function module.

After we finish with the creation of the web service infrastructure we need to proceed with the publication of it. The publication of the web service is handled by the SAP SOA manager, reached by the transaction of the same name.

SOA Manager is a WebDynpro ABAP application falls under the ICF tree and handles all SOAP Web Services. For this reason, its ICF node needs to be active.

As a WebDynpro the SOA manager opens in a browser.

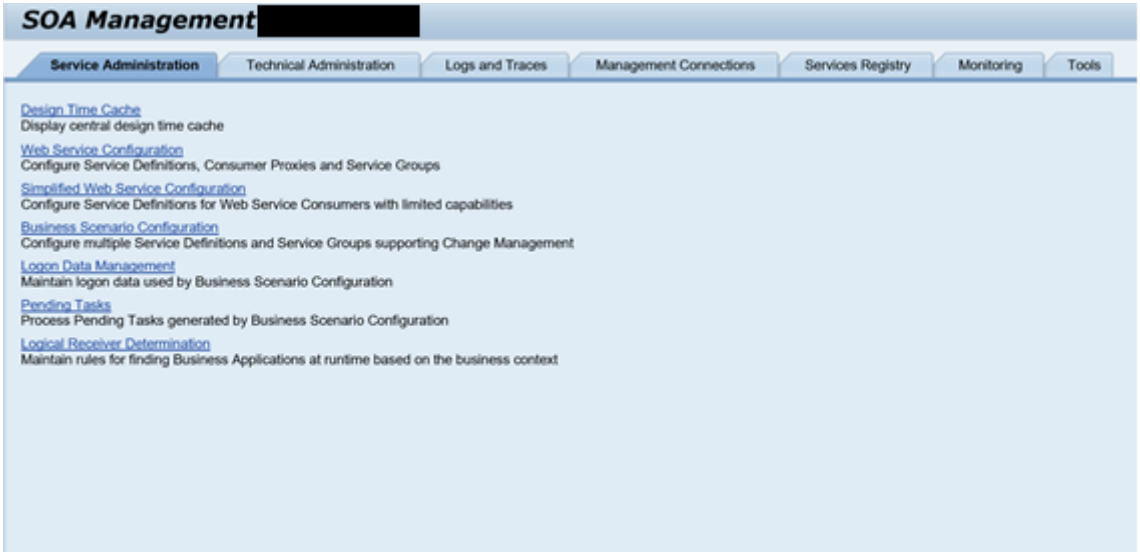

Figure 20SOAManager

The service is created in the first screen and then we proceed to its configuration.

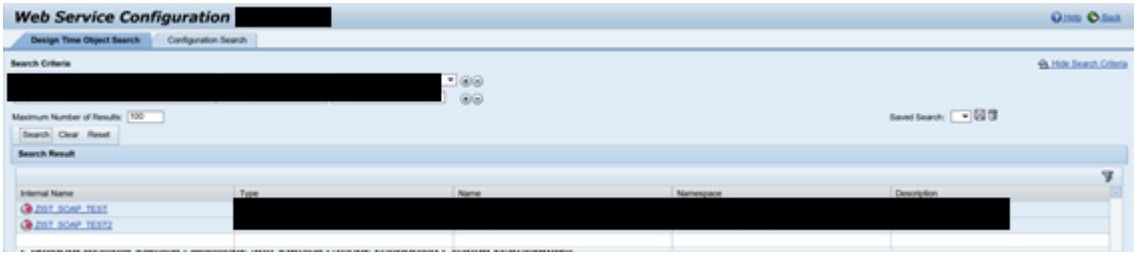

Figure 21Web Service Configuration

The first tab concerns security of the web service published and is usually done by a SAP basis consultant. Here the sample security features of the example web service.

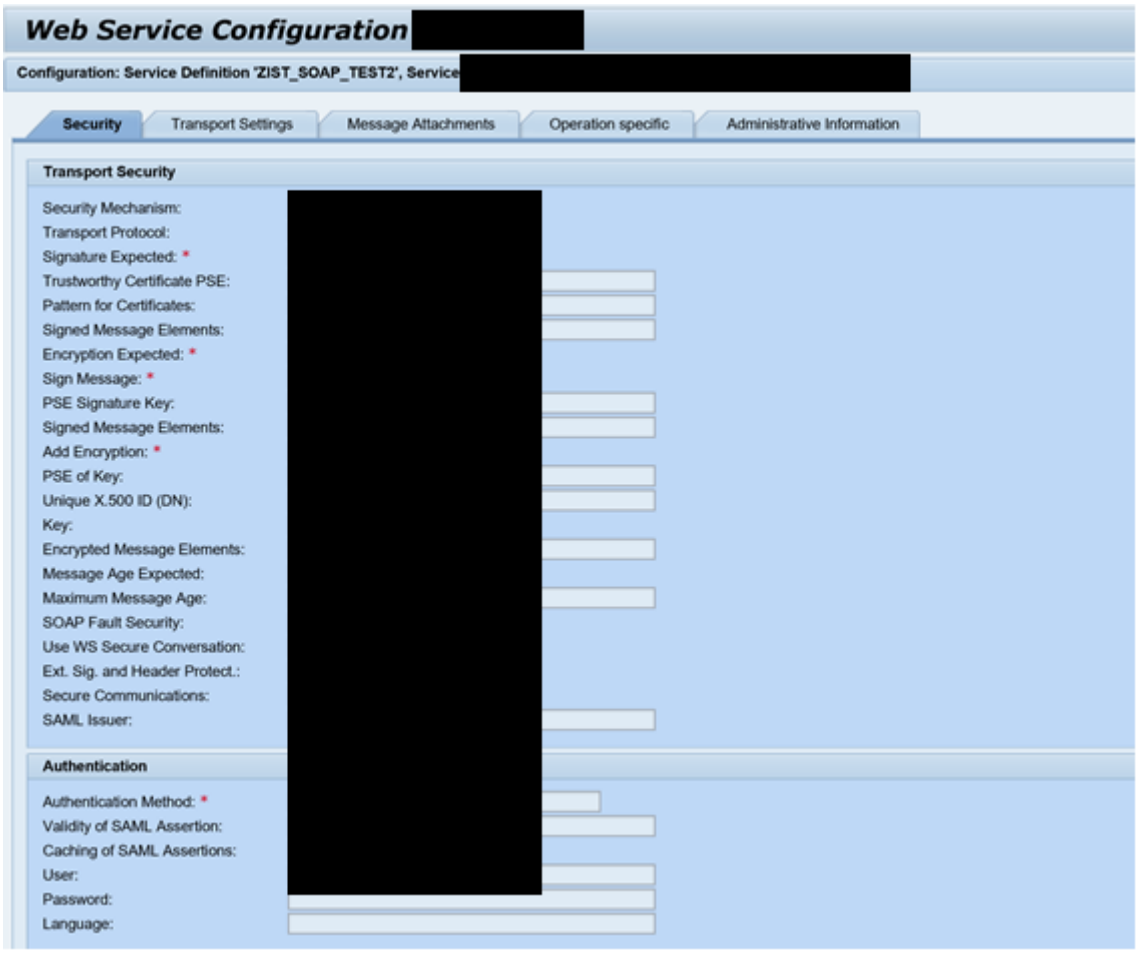

Figure 22Web Service Security Configuration

In the next tab the transport settings are set. These include the transport Protocol and the URL address the web service will be accessible from.

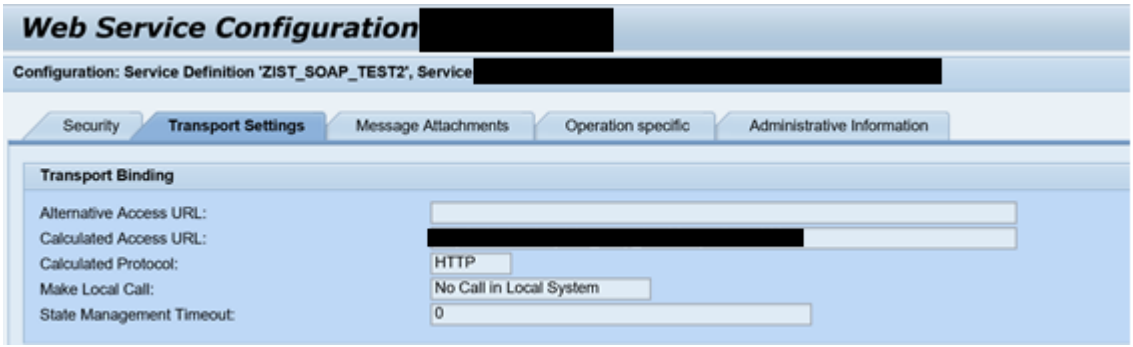

Figure 23Web Service Transport Settings Configuration

In this scenario we define that this web service will not have any message attachments.

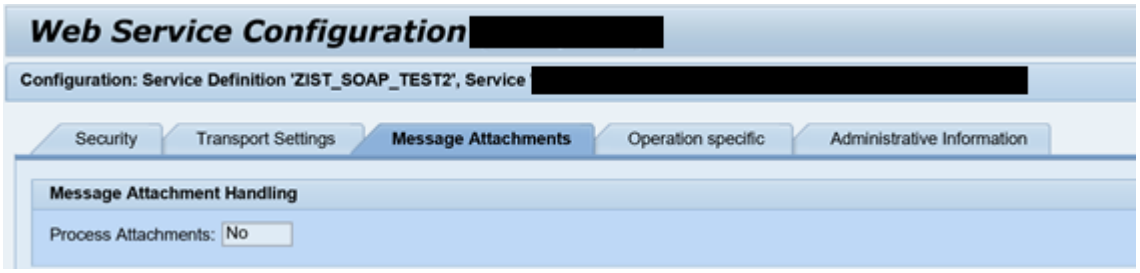

Figure 24Web Service Configuration Message Attachments

After the customization is complete, the WSDL of the web service needs to be produced.

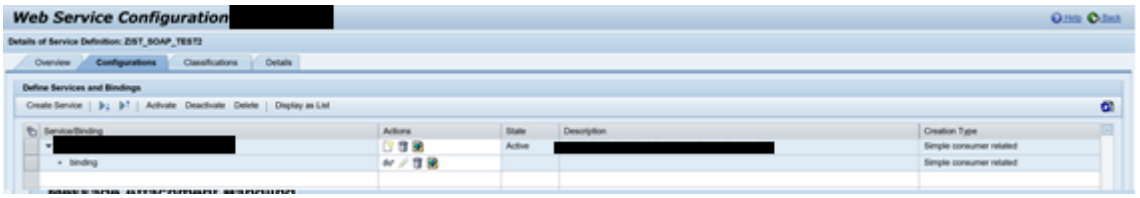

Figure 25Web Service Configuration Bindings

The WSDL generation depends on the client platform we want to connect.
In this case we chose the following. On the bottom most section of this WebDynpro we can see the URL were the WSDL resides and this is the URL the client app will have visibility to.

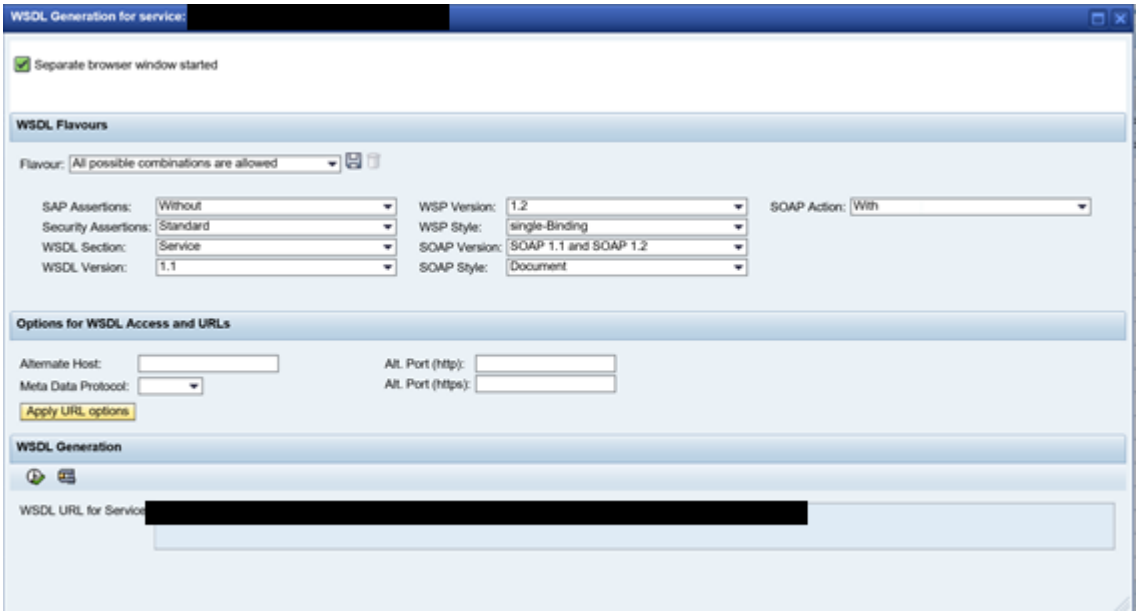

Figure 26WSDL Generation

With all these steps completed the produced WSDL looks like the following. This WSDL is accessible via the URL and Namespace of the SAP system and all properly configured clients can call the web service.

The WSDL produced is the following.

```
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions targetNamespace=" XXXXXXX "
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns=" XXXXXXX "
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsoap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <wsdl:documentation>
                <sidl:sidl xmlns:sidl=" XXXXXXX "/>
        </wsdl:documentation>
        <wsdl:types>
                <xsd:schema attributeFormDefault="qualified" targetNamespace="
XXXXXXX ">
                        <xsd:simpleType name="char10">
                                <xsd:restriction base="xsd:string">
                                        <xsd:maxLength value="10"/>
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:simpleType name="char12">
                                <xsd:restriction base="xsd:string">
                                        <xsd:maxLength value="12"/>
                                </xsd:restriction>
                        </xsd:simpleType>
                        <xsd:element name="ZIST_SOAP_TEST_2">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="I_VKONT"
type="tns:char12"/>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="ZIST_SOAP_TEST_2Response">
                                <xsd:complexType>
                                        <xsd:sequence>
                                                <xsd:element name="E_GPART"
type="tns:char10"/>
                                        </xsd:sequence>
                                </xsd:complexType>
                        </xsd:element>
                </xsd:schema>
        </wsdl:types>
        <wsdl:message name="ZIST_SOAP_TEST_2">
```

```
            <wsdl:part element="tns:ZIST_SOAP_TEST_2" name="parameters"/>
        </wsdl:message>
        <wsdl:message name="ZIST_SOAP_TEST_2Response">
            <wsdl:part element="tns:ZIST_SOAP_TEST_2Response"
name="parameter"/>
        </wsdl:message>
        <wsdl:portType name="zist_soap_test2">
            <wsdl:operation name="ZIST_SOAP_TEST_2">
                <wsdl:input message="tns:ZIST_SOAP_TEST_2"/>
                <wsdl:output message="tns:ZIST_SOAP_TEST_2Response"/>
            </wsdl:operation>
        </wsdl:portType>
        <wsdl:binding messageEncoding="Text" name="binding" textEncoding="utf-8"
type="tns:zist_soap_test2">
            <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
            <wsdl:operation name="ZIST_SOAP_TEST_2">
                <soap:operation
soapAction="XXXXXXX:zist_soap_test2:ZIST_SOAP_TEST_2Request"
style="document"/>
                <wsdl:input>
                    <soap:body use="literal"/>
                </wsdl:input>
                <wsdl:output>
                    <soap:body use="literal"/>
                </wsdl:output>
            </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="service">
            <wsdl:port binding="tns:binding" name="binding">
                <soap:address location="xxxxxxxxxxxxxx"/>
            </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

As shown the importing parameters and the exporting parameters are nodes on the WSDL . Also all relevant information a client app would need , such as the url (soap:address location=, Here changed to xxxxxxxx for security purposes), the function and the data elements of the values passed and received.

Conclusions

The integration of android and SAP has started to be an increasingly sought-after skill for SAP consultants, ABAP developers and web developers worldwide. Especially with the rapid rise of IoT enabled devices running android the communication between SAP systems and Android becomes crucial. For a big volume of requests and services SAP recommends the purpose build middleware SAP PO (Process Orchestration) Middleware. However, many companies choose to not implement this middleware because of the cost both in hardware and in implementation. SAP ECC systems can handle a variety of web services as demonstrated and with logging and authorization procedures already in place a custom implementation can work as well for companies with a small amount of web services needed.

In the new era of cloud and In memory computing in the SAP ecosystem, web services from SAP systems will become more mainstream and android application integration even more in accordance to the rapid rise of mobile applications.

Acknowledgements

I would like to thank my advisors on this thesis but most importantly Panagiodoros Gardikis and Vangelis Kalaitzoglou for their valuable insight and deep knowledge they were willing to share on relatively obscure aspects of a SAP system and the many hours they spend helping me debug the processes.

References

https://www.sap.com/greece/index.html.

https://en.wikipedia.org/wiki/ABAP

http://www.xyzws.com/scdjws/SGS22/8

https://www.w3.org/TR/2000/NOTE-SOAP-20000508/

https://blogs.sap.com/2013/03/08/create-an-sap-web-service-call-web-service-from-net-application-passing-user-credentials/

https://blogs.sap.com/2014/05/20/step-by-step-to-create-consume-and-trace-web-service-in-abap-system/

https://help.sap.com/doc/abapdocu_751_index_htm/7.51/en-US/abenicf.htm

https://www.quora.com/What-is-web-services-and-why-do-we-use-this

https://bytes.com/topic/java/answers/879415-advantages-disadvantages-soap-over-http

user.it.uu.se/~hsander/Courses/DistributedSystems/Reports/soap_report_2.pdf

www.stackoverflow.com

Dotnet.sys-con.com

https://www.w3.org/TR/soap12-part1/#soapenv

http://www.ws-i.org/

https://developer.android.com/training/volley/index.html

http://simpligility.github.io/ksoap2-android/getting-started

https://stackoverflow.com/questions/6331276/ksoap2-webservice-help

https://stackoverflow.com/questions/6817685/accessing-http-authentificated-soap-webservice-via-ksoap2-httptransportbasicauth

https://blogs.sap.com/2013/01/24/developing-a-rest-api-in-abap/

https://blogs.sap.com/2010/12/07/android-and-restful-web-service-instead-of-soap/

https://dalanzg.github.io/tips-tutorials/sap/2016/09/25/how-to-publish-rest-webservice-in-sap/

https://blogs.sap.com/2013/01/07/abap-and-JSON/