ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

Bank Audi

Internship report

Internship in web services and middleware

Bank Audi – IT department

Form 22-06-2015 to 21-08-2015

Supervisor Name	Charbel Haboub
Tutor Name	Chadi Morad
Tutor Name	Ghassan Boutros
Intern Name	Karim Assaad

Acknowledgment

First of all I would like to express my gratefulness for my school ENSIIE and especially for Mr. Julien Forest and Mme. Marie Szafranski who are responsible of the internship program.

I want to give a special thanks to Mr. Charbel Haboub for consenting to be my supervisor of the Internship Program. I also express my full gratitude to Mr. Chadi Morad for providing me with all the theoretical knowledge I learned and for his guidance, plus I would like to thank Mr. Ghassan Boutros for helping to achieve my goals during the internship and for giving the tools and the technical knowledge to proceed in my project.

In the end, I express my overall gratitude to the middle wear team, and to Bank Audi in general for providing me such a scope.

Contents

I. Introduction:

Middleware is the software that connects software components or enterprise applications. It's the software layer that lies between the operating system and the applications on each side of a distributed computer network. Typically, it supports complex, distributed business software applications. In particular, it allows applications to communicate using web services. Business services allow application to communicate with the middle ware, and proxy services allow communication between business services in the middle ware, plus it plays the role of orchestrator to convert the language and the structure of applications.
In this century, a software engineer must have a minimal knowledge about the middleware.
The internship's at Bank Audi in the middle ware team (in the DEV environment) provide the needed skills and knowledge to use the middleware and web services in order to create communication between applications.

II. Bank Audi

1. History

Bank Audi was founded in 1830. It's a universal bank with a presence in 12 countries. It operates principally in Lebanon, the MENA region (Jordan, Egypt, Syria, Saudi Arabia, Qatar, Sudan, Abu Dhabi) and in Monaco, Switzerland, France and Turkey. It offers a full range of products and services that covers commercial and corporate banking, retail and individual banking and private banking, and investment banking and on-line brokerage. In addition of being ranked first among Lebanese banks in terms of total assets, shareholder's equity, customer's deposits, loans and advances and net profits, Bank Audi has:

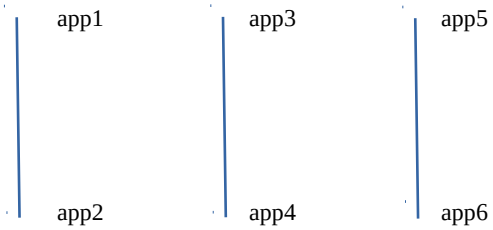
- One of the largest branch networks in Lebanon, with 80 branches
- Undertaken significant regional expansion and has the fourth largest coverage among the top 15 Arab banking institutions in the MENA region.

2. The IT department in Bank Audi

The IT department helped a great deal Bank Audi to become a leader on Lebanese banks. With its modern technologies the IT department provided costumers with all the banking services and the online applications. And since Bank Audi was growing very fast, a middle ware team was founded to integrate a service bus in the bank system. "This team is organized into 3 pillars: Design, Execution, and Operations." And since then Bank Audi is transforming the IT into a quality service provider, and thus making the department a business partner within the Bank.

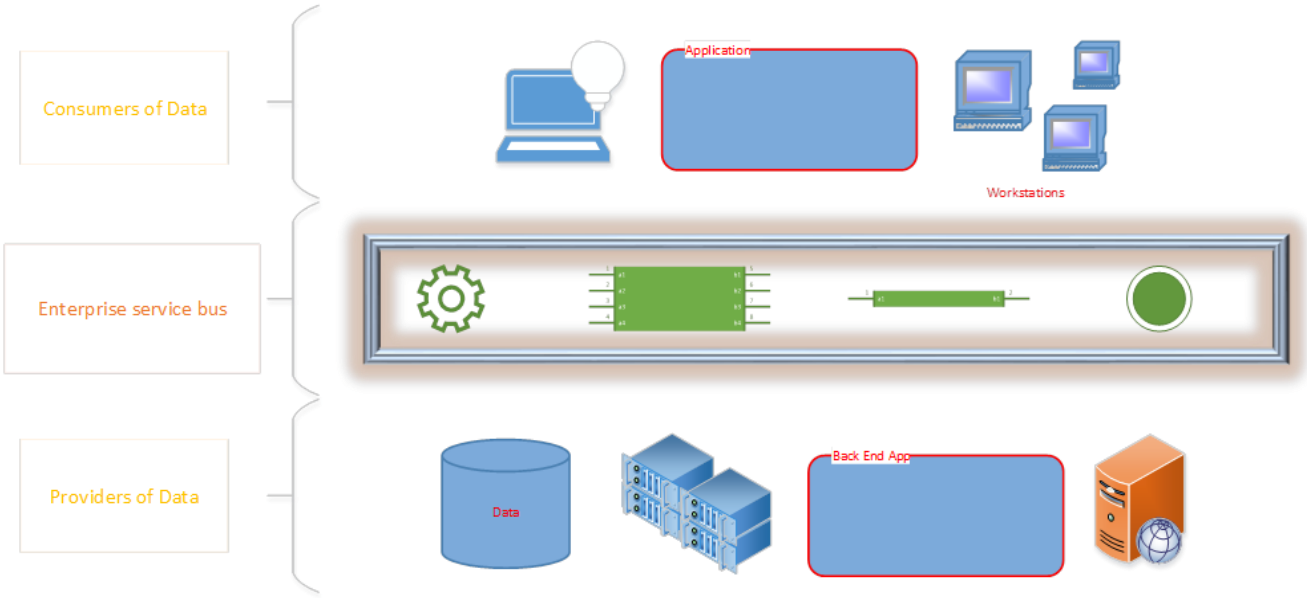
3. Middleware in Bank Audi

In the department of Group Information Technology, Cobol 400 was used to code the core which was initially using a point to point topology to connect applications, as shown in the image below.



This topology was effective on a small scale, but when Bank Audi decided to pursue modern technologies and open its door for services other than banking such as retail and lending, a new topology was required that is able to add new technologies without having to reconstruct the whole core. So it had two choices: to buy a whole integrated core with all the baking applications and be dependent of a seller company, or to create its own middleware and let the information department integrate applications into it. Because of the high qualification of the IT department, Bank Audi was able to go on with the second choice. And so was assured an independency of any IT company.

To create the middle ware, the team had to split the core into many independent applications in addition of a small fragment of the core witch is not yet decomposed. And in the center they created the middle ware.



The picture below shows the bank software architecture as a high level.

III. Methodology, Tools & Strategy

1. Methodology

The middle ware team uses a scrum methodology for managing a product development.

Scrum is an agile methodology that has replaced the waterfall methodology. It is an agile methodology therefore you must finish the giving tasks in a giving time. In the scrum, the time giving is always two weeks.
The Scrum creates a cross functional team which is divided into:

- Project Owner: He is the responsible of the project
- Developers: They are responsible of the development
- Operations: They are responsible of the deployment
- Scrum Master: he manages the process for how information is exchanged.

The procedure used in these two weeks is called sprint, and it is divided into:

- Planning: It occurs in the beginning of the sprint.
- Stand up: It's a 15 minutes stand up meeting done on daily basis to improve collaboration. In a stand up each person must give an answer on three questions:
  - What did I do yesterday
  - What I will do today
  - What are the problems I am facing

And the rest of the team can share their opinions on those matter.

- Grooming: It's a meeting that discuss what the team must cover in the next sprint.
- Demo: The demo is the presentation of each story and their result, the demo is the last step of the sprint.
- Retrospective: It occur in the same day as the demo and its purpose is to improve the quality of the team.

2. Tools

a. Languages

The languages used by the middle ware team are:

- XML: EXtensible Markup Language
- XSD: It's a schema of an XML, it present the types and the structures
- XPATH: Its used in XQuery and XSLT to call variables
- XQUERY and XSLT: Languages used to transform and orchestrate. Plus it can integrate a DVM
- WSDL: it's used to connect
- DVM: it's an xml architecture in an Xquery or XSLT file that plays the role of a database

b. Software

The software used by the middle ware team are:

- SVM: it's a hosting service for projects. It is used to commit, update …
- SoapUI: It's used for testing SOA
- Eclipse and Jdevelopper 11g: it's used to code services and implement them.
- Web logic: It's used to test the services locally.
- Middle Ware

3. Strategy

a) Architecture

In the web logic, the department uses two oracle architecture: OSB & SOA

- OSB: (Oracal Service Bus)
  - It's a product of oracle that has an ESB[1] architecture.
  - It uses Xquery for transformation.
  - It provides message delivery services.
  - It is used to make simple connections.
- SOA: (Service-Oriented Architecture)
  - It's an architecture in which application components provide services to other components via a communications protocol, typically over a network. The principles of service-orientation are independent of any vendor, product or technology.
  - It's an architecture that uses XSLT for transformation.
  - It's simpler and faster than OSB and support more.
  - It is uses to make complex connections.

b) Development process (Overview)

Consumers of Data is connected to the middle ware through proxy services.
Providers of Data is connected to the middle ware through business services.
The developer begins by

1) Collecting the XSDs and the WSDLs from the provider if they are needed (if they are not provided he creates create them himself).
2) Creating the XSDs and the WSDLs that will be used on the proxy service
3) Creating the architecture of the proxy using Routing, Replace, Assign, if else, Stage, Callout...
4) Creating the Xquery files needed in the proxy service.
5) Implementing the Xquery files in the proxy service.
6) Testing the services using the Web Logic
7) Commit

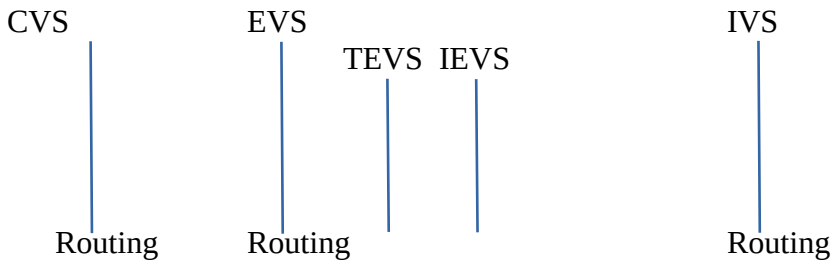
---

[1] *ESB (enterprise service bus): It's a software architecture model used for designing and implementing communication between mutually interacting software applications in a service-oriented architecture (SOA)*

c) Canonical Model:

- Definition: A canonical model is a design pattern used to enable communication between different data format.

The bank uses canonical model in the middle ware to assure that it is generic and not link to any other system. The canonical model is shown in the image below:



CVS: transform the consumer to canonical (vice versa)
EVS: security
TEVS: use http protocol to transport data
IEVS: load balancer
IVS: transform the canonical to provider (vice versa)

So every time a service is connected to the middle-ware, the team create an Xquery to transform it into canonical.

d) Environments:

The middle ware in Bank Audi is divided into 4 environments:

Environment	Users	Purpose	Support
DEV	Developers	Development & testing	----------
QA	Testing team	Testing	Developers
UAT	Business team	Demand and verify	Testing team & developers
LIVE	Operations	Deployment	Developers

IV. Tasks and Project

1. Tasks:

The first week of the internship was dedicated to learn XML and web services technologies. (Xquery, XSD, WSDL, Proxy services, Business Services, OSB, DVM)
In the second week began the small tasks:

- First task (simple task): create WSDLs and XSDs based on giving XML samples.
- Second task (simple task): Create a web service to "Bankmate" to ensure the validation of a costumer account using simple transformations with Xquery. This task required creation of a proxy service and an Xquery file in addition of two WSDLs for the proxy and the business service that communicate with Bankmate[2]. (All the XSDs and and the WSDLs related to the business service were provided in addition the URL of bankmate/validateAccount.

- Third task (complex task): Create a web service to "bankmate" to get account situations after the validation of thia account. This task required creation of a proxy service and an Xquery file in addition of the WSDL for the proxy and a buisness service that communicate with the application "bankmate" and a CallOut functions toward the second task. (All the xsds and the wsdl related to the buisness service were provided in addition the the URL of "bankmate/getAccountSituations").

  A screenshot that represent the message flow of the proxy service created in the third task can be found in the annex: it validates the account then gets its details.

To make the message flow, the Xquerys are implemented in the stages "Replace" and "Assign", The business service is called in the Routing. And The callOut is used to call another service (in this example it call itself to verify the account). The stage Response is used to send a message in case of an error.

2. Project:

The second month was dedicated to a committed project.

- ### Background:

The middle ware team are working on a massive project which conduct to implement all the operations of Bank Audi into the middle ware. This massive project is divided to smaller projects, and one of those project is the branch Network project.

- ### Objective:
  - Create a web service that have two operations:

[2] *Bankmate : it's an application for Bank Audi that*

- getBranchList

Input: Branch name
Branch address
Coordinate
Status
Regional
Branch type

Output: branch code
Branch name
Address
Coordinate

- getBranchDetails:
  Input: Branch code
  Output: Branch EBO[3]

- Create a DVM on OSB then implement it in SOA

- Bonus:

Create a mobile interface specific for the employees that uses the middle-ware to search for branches and give its details of the selected branch.

- Procedure:
  - Create a new capability in the system which is related to branches and is called chn.Brn
  - Create a DVM on OSB which play the role of a database for all the branches information (name, location, latitude, longitude, region…). And later on the middleware team will add an event listener (EDA) on this DVM allowing it to be the center for the public data related to the branches. And once it is modified on this DVM it will change automatically in the applications, in the database and in other DVM in other departments.
  - Update the Branch EBO
  - Create an XSD and a WSDL for the evs and an XSD and a WSDL for the ivs.
  - Create four proxy, for the evs, isv, tevs and ievs
  - Create Xquery files
  - Commiting the OSB project with SVM
  - Implement the DVM on SOA

[3] *EBO: An XSD that contains everything that comes to mind on a specific subject (It is very optimized)*

- Technical details:

The Proxy architecture of the three first stages in the ivs and the evs are similar
Since the web service contain more than one operation, an operational node was used to carpet the procedures.
In the first stage of each operation, variables are declared.
The second stages' objective is to create two reports, one for the request and another for the response. The reports are used to simplify troubleshooting in case of an error at the testing phase.
The third stage of each operation is the validation of request as well as the response. To validate that the data sent or the data received are well the data expected, the element of sending the data in the XSD file is compared to the body of the data sent and the element of receiving the data in the XSD file is compared to the body of the data received.

In the ivs, the fourth stage of each operation is used to transform the data using the DVM.

In the evs, the fourth stage and fifth stage are stages for mapping. The fourth stage map the data received, and the fifth map the data sent.
The sixth stage is used for routing. The route to the ivs.

The ievs and the tevs have only one stage which route to the evs. And both have http protocols.

The evs, ivs, tevs and ievs have an error handler stage which manage the errors and import a DVM called "ErrorDVM" from the core of the middle ware to map the errors and give its details and descriptions.
So if we presume to have an error in a certain stage, we will obtain an error nest which give us all the errors which resulted the creation of this error on this certain stage.

The two function are implemented in the evs as well as the ivs.
In the evs the proxy send the output to the ivs which will map the data using the DVM and return the result to the evs.
The DVM is implemented on SOA because in the operations department they use SOA and they will be responsible of updating the data.

GetBranchList is a smart search function. It takes the branch name or the branch type or both as input and it returns all the branches that are coherent with the input.

The images below represent the architecture of the ivs, evs, tevs and ievs respectively

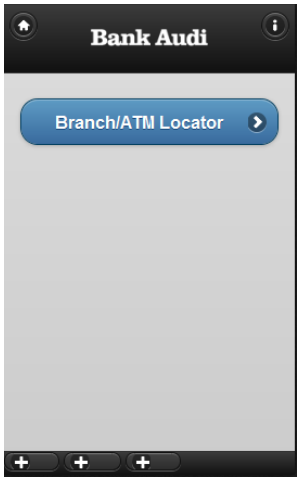

OperationalBranch
getBranchList
getBranchDetails
Default
Request Pipeline
Response Pipeline
Set_ebmCID
Assign
Set_operationName
Set_requestKey
Set_userHdr
LogRequest
LogResponse
Report
ValidateRequest
ValidateResponse
Validate
mapRequest
CheckErrors
If Then
Else
Raise Error
Replace
Fault Handler
ServiceFaultHandler
Insert
Reply

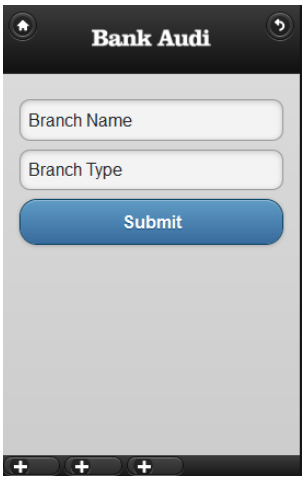
TEVS *IEVS*

- Difficulties:
  - Because the middle ware team uses oracle service bus 11g (rather than oracle service bus 12c) it was very difficult to call the DVM from SOA. And although we managed to find a way to call it, in the testing phase we noticed that after updating a field in the DVM on SOA, OSB update the data automatically after one day which is not an optimal solution. So the team decided to take responsibility of updating the DVM on OSB until they update to oracle service bus 12c.
  - To implement the two operations in a mobile app, an API must be created in addition to a JASON section in the HTML. In the middle ware team neither the eclipse nor JDeveloper are suited to create an API because certain libraries are missing specifically the libraries related to “REST binding”. To install them the team must send a request to the installment team. This process takes time and thus making this task incompatible with the duration of the internship.

- Bonus:

The mobile interface was created as shown below:





When clicking on the Branch/ATM Locator the application must show two search fields that helps the users to locate the branch needed. Once the user click on a branch name, the app must show its details.

V. Applying university skills

The knowledge in XML, XSD and DTD helped a great deal in understanding the concept of the WSDLS.
The course of Database and Web Development at ENSIIE gave me an advantage when I worked of the DVMs.
Plus information languages that I learned at ENSIIE helped me to learn a new programing language easily (XQuery).

VI. Conclusion

The internship was an excellent opportunity for me to gain some hands on experience in the middle ware field. It has been an excellent and rewarding experience. I have learned almost up to 80% of everything I need to know about the usage of the middle-ware and developed my knowledge in XML technologies. And I have learned a new programming language, a new concept in the modern information technologies and the relation between business and information. In addition I have been able to meet many people that might be able to help me with opportunities in the future.
One main thing that I have learned in this internship is time management skills especially which is related to the scrum methodology.
In this century, the middle ware became a very important asset in the modern technologies, and software engineering post needed in all fields (business, economic, banking …).

VII. Annex

Project:

- Xquery Code for transforming getBranchDetails from evs to ivs

```
declare namespace ns2 = "http://www.banqueaudi.com/ebo/util.cmn.EBMHeader";
declare namespace ns1 = "http://www.banqueaudi.com/ebo/util.cmn.Basic";
declare namespace ns4 = "http://www.banqueaudi.com/ebo/svl.cmn.Identifiers";
declare namespace ns3 = "http://www.banqueaudi.com/chn.brn.ivs.Branch";
declare namespace ns0 = "http://www.banqueaudi.com/chn.brn.evs.Branch";
declare namespace xf =
"http://tempuri.org/chn.brn/Chn.brn.evs.Branch_1.0/xquery/getBranchDetails/getBra
nchDetails_to_getBranchDetails/";
declare function xf:getBranchDetails_to_getBranchDetails($header1 as
element(ns2:header),
    $getBranchDetails1 as element(ns0:getBranchDetails))
    as element(ns3:getBranchDetails) {
        <ns3:getBranchDetails>
            <ns2:header>{ $header1/@* , $header1/node() }</ns2:header>
            <ns3:body>
                <ns4:branchId>{ data($getBranchDetails1/ns0:body/ns4:branchId)
}</ns4:branchId>
            </ns3:body>
        </ns3:getBranchDetails>
};
declare variable $header1 as element(ns2:header) external;
declare variable $getBranchDetails1 as element(ns0:getBranchDetails) external;
xf:getBranchDetails_to_getBranchDetails($header1,
    $getBranchDetails1)
```

- Xquery Code for transforming getBranchDetails from ivs to evs

```
declare namespace ns2 = "http://www.banqueaudi.com/ebo/util.cmn.EBMHeader";
declare namespace ns1 = "http://www.banqueaudi.com/ebo/util.cmn.Basic";
declare namespace ns4 = "http://www.banqueaudi.com/chn.brn.ivs.Branch";
declare namespace ns3 = "http://www.banqueaudi.com/ebo/svl.cmn.Identifiers";
declare namespace ns0 = "http://www.banqueaudi.com/chn.brn.evs.Branch";
declare namespace ns5 = "http://www.banqueaudi.com/ebo/svl.brn.Branch";
declare namespace ns6 = "http://www.banqueaudi.com/ebo/pty.party.ContactDetails";
declare namespace xf =
"http://tempuri.org/chn.brn/Chn.brn.evs.Branch_1.0/xquery/getBranchDetails/getBra
nchDetailsResponse_to_getBranchDetailsResponse/";
declare function xf:getBranchDetailsResponse_to_getBranchDetailsResponse($header1
as element(ns2:header),
    $getBranchDetailsResponse1 as element(ns4:getBranchDetailsResponse))
    as element(ns0:getBranchDetailsResponse) {
        <ns0:getBranchDetailsResponse>
            <ns2:header>{ $header1/@* , $header1/node() }</ns2:header>
            <ns0:body>
                <ns5:branch>{ $getBranchDetailsResponse1/ns4:body/ns5:branch/@* ,
$getBranchDetailsResponse1/ns4:body/ns5:branch/node() }</ns5:branch>
            </ns0:body>
        </ns0:getBranchDetailsResponse>
};
declare variable $header1 as element(ns2:header) external;
declare variable $getBranchDetailsResponse1 as
```

```
element(ns4:getBranchDetailsResponse) external;
xf:getBranchDetailsResponse_to_getBranchDetailsResponse($header1,
    $getBranchDetailsResponse1)
```

➢ Xquery Code for transforming getBranchList from evs to ivs

```
declare namespace ns2 = "http://www.banqueaudi.com/ebo/util.cmn.EBMHeader";
declare namespace ns1 = "http://www.banqueaudi.com/ebo/util.cmn.Basic";
declare namespace ns4 = "http://www.banqueaudi.com/ebo/svl.brn.Branch";
declare namespace ns3 = "http://www.banqueaudi.com/chn.brn.ivs.Branch";
declare namespace ns0 = "http://www.banqueaudi.com/chn.brn.evs.Branch";
declare namespace ns5 = "http://www.banqueaudi.com/ebo/pty.party.ContactDetails";
declare namespace xf =
"http://tempuri.org/chn.brn/Chn.brn.evs.Branch_1.0/xquery/getBranchList/getBranch
List_to_getBranchList/";
declare function xf:getBranchList_to_getBranchList($header1 as
element(ns2:header),
    $getBranchList1 as element(ns0:getBranchList))
    as element(ns3:getBranchList) {
        <ns3:getBranchList>
            <ns2:header>{ $header1/@* , $header1/node() }</ns2:header>
            <ns3:body>
                <ns4:branchName>{ data($getBranchList1/ns0:body/ns4:branchName)
}</ns4:branchName>

   <ns3:status>{data($getBranchList1/ns0:body/ns0:status)}</ns3:status>
                {
                if (exists($getBranchList1/ns0:body/ns4:branchType))
                then <ns4:branchType>{
data($getBranchList1/ns0:body/ns4:branchType) }</ns4:branchType>
                else ""
                }
                <ns4:branchAddress>{
$getBranchList1/ns0:body/ns4:branchAddress/@* ,
$getBranchList1/ns0:body/ns4:branchAddress/node() }</ns4:branchAddress>
                <ns4:regionalOffice>{
$getBranchList1/ns0:body/ns4:regionalOffice/@* ,
$getBranchList1/ns0:body/ns4:regionalOffice/node() }</ns4:regionalOffice>
            </ns3:body>
        </ns3:getBranchList>
};
declare variable $header1 as element(ns2:header) external;
declare variable $getBranchList1 as element(ns0:getBranchList) external;
xf:getBranchList_to_getBranchList($header1,
    $getBranchList1)
```

➢ Xquery Code for transforming getBranchList from ivs to evs

```
declare namespace ns2 = "http://www.banqueaudi.com/ebo/util.cmn.EBMHeader";
declare namespace ns1 = "http://www.banqueaudi.com/ebo/util.cmn.Basic";
declare namespace ns4 = "http://www.banqueaudi.com/chn.brn.ivs.Branch";
declare namespace ns3 = "http://www.banqueaudi.com/ebo/svl.cmn.Identifiers";
declare namespace ns0 = "http://www.banqueaudi.com/chn.brn.evs.Branch";
declare namespace ns5 = "http://www.banqueaudi.com/ebo/svl.brn.Branch";
declare namespace ns6 = "http://www.banqueaudi.com/ebo/pty.party.ContactDetails";
declare namespace xf =
```

```
"http://tempuri.org/chn.brn/Chn.brn.evs.Branch_1.0/xquery/getBranchList/getBranch
ListResponse_to_getBranchListResponse/";
declare function xf:getBranchListResponse_to_getBranchListResponse($header1 as
element(ns2:header),
    $getBranchListResponse1 as element(ns4:getBranchListResponse))
    as element(ns0:getBranchListResponse) {
        <ns0:getBranchListResponse>
            <ns2:header>{ $header1/@* , $header1/node() }</ns2:header>
            <ns0:body>
                <ns5:branchSummarySet>{
$getBranchListResponse1/ns4:body/ns5:branchSummarySet/@* ,
$getBranchListResponse1/ns4:body/ns5:branchSummarySet/node()
}</ns5:branchSummarySet>
            </ns0:body>
        </ns0:getBranchListResponse>
};
declare variable $header1 as element(ns2:header) external;
declare variable $getBranchListResponse1 as element(ns4:getBranchListResponse)
external;
xf:getBranchListResponse_to_getBranchListResponse($header1,
    $getBranchListResponse1)
```

- Xquery Code for mapping getBranchDetails with the DVM

```
declare namespace ns2 = "http://www.banqueaudi.com/ebo/svl.cmn.Identifiers";
declare namespace ns1 = "http://www.banqueaudi.com/ebo/util.cmn.EBMHeader";
declare namespace ns4 = "http://www.banqueaudi.com/ebo/svl.brn.Branch";
declare namespace ns3 = "http://www.banqueaudi.com/chn.brn.ivs.Branch";
declare namespace ns0 = "http://www.banqueaudi.com/ebo/util.cmn.Basic";
declare namespace ns5 = "http://www.banqueaudi.com/ebo/pty.party.ContactDetails";
declare namespace xf =
"http://tempuri.org/chn.brn/Chn.brn.ivs.Branch_1.0/xquery/getBranchDetails/getBra
nchDetails_to_getBranchDetailsResponse_DVM/";
declare namespace dvm = "http://xmlns.oracle.com/dvm";
declare function xf:getBranchDetails_to_getBranchDetailsResponse_DVM($header1 as
element(ns1:header),
    $getBranchDetails1 as element(ns3:getBranchDetails),
    $BranchesNetworkDVM as element(*))
    as element(ns3:getBranchDetailsResponse) {
        <ns3:getBranchDetailsResponse>
            <ns1:header>{ $header1/@* , $header1/node() }</ns1:header>
            <ns3:body>
            {
                if ( exists(data($BranchesNetworkDVM/dvm:rows/dvm:row[
dvm:cell[1] = data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[1])) )
then
                                    <ns4:branch>
                                        <ns2:branchId>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[1]) }</ns2:branchId>
                                        <ns4:branchName>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[2]) }</ns4:branchName>
                                        <ns4:branchType>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
```

```
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[15]) }</ns4:branchType>
                                    <ns4:manager>
                                        <ns4:name>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[12]) }</ns4:name>
                                        <ns5:emailAddressContact>
                                            <ns5:email>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[13]) }</ns5:email>
                                        </ns5:emailAddressContact>
                                    </ns4:manager>
                                    <ns4:branchAddress>
                                        <ns5:addressContact>
                                            <ns5:address>
                                                <ns5:line1>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[3]) }</ns5:line1>
                                                <ns5:state>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[14]) }</ns5:state>
                                                <ns5:phoneContactSet>
                                                    <ns5:phoneContact
type="Landline">
                                                        <ns5:phone>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[9]) }</ns5:phone>
                                                    </ns5:phoneContact>
                                                    <ns5:phoneContact
type="Fax">
                                                        <ns5:phone>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[10]) }</ns5:phone>
                                                    </ns5:phoneContact>
                                                </ns5:phoneContactSet>
                                            </ns5:address>
                                        </ns5:addressContact>
                                        <ns4:geographicLocation>
                                            <ns4:geographicCoordinates>
                                                <ns4:longitude>{
fn:replace(data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[5]), " ", "")
}</ns4:longitude>
                                                <ns4:latitude>{
fn:replace(data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[4]), " ", "")
}</ns4:latitude>
                                            </ns4:geographicCoordinates>
                                        </ns4:geographicLocation>
                                    </ns4:branchAddress>
                                    <ns4:serviceSet>
                                        <ns4:service>
                                            <ns4:name>ATM</ns4:name>
                                            <ns4:count>{
                                                if
(data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
```

```
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[7])="")
                                                        then 0
                                                        else
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[7])

                                                        }</ns4:count>
                                                </ns4:service>
                                                <ns4:service>
                                                      <ns4:name>ITM</ns4:name>
                                                      <ns4:count>{
                                                        if
(data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[8])="")
                                                        then 0
                                                        else
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($getBranchDetails1/ns3:body/ns2:branchId) ]/dvm:cell[8])

                                                        }</ns4:count>
                                                </ns4:service>
                                          </ns4:serviceSet>
                                    </ns4:branch>

                  else(
                              <ns4:branch/>
                        )

                        }
                </ns3:body>
        </ns3:getBranchDetailsResponse>
};
declare variable $header1 as element(ns1:header) external;
declare variable $getBranchDetails1 as element(ns3:getBranchDetails) external;
declare variable $BranchesNetworkDVM as element(*) external;
xf:getBranchDetails_to_getBranchDetailsResponse_DVM($header1,
    $getBranchDetails1,
    $BranchesNetworkDVM)
```

- Xquery Code for mapping getBranchList with the DVM

```
declare namespace ns2 = "http://www.banqueaudi.com/ebo/svl.cmn.Identifiers";
declare namespace ns1 = "http://www.banqueaudi.com/ebo/util.cmn.EBMHeader";
declare namespace ns4 = "http://www.banqueaudi.com/ebo/svl.brn.Branch";
declare namespace ns3 = "http://www.banqueaudi.com/chn.brn.ivs.Branch";
declare namespace ns0 = "http://www.banqueaudi.com/ebo/util.cmn.Basic";
declare namespace ns5 = "http://www.banqueaudi.com/ebo/pty.party.ContactDetails";
declare namespace xf =
"http://tempuri.org/chn.brn/Chn.brn.ivs.Branch_1.0/xquery/getBranchList/getBranch
List_to_getBranchListResponse_DVM/";
declare namespace dvm = "http://xmlns.oracle.com/dvm";
declare function xf:getBranchList_to_getBranchListResponse_DVM($header1 as
element(ns1:header),
    $getBranchList1 as element(ns3:getBranchList), $BranchesNetworkDVM as
element(*))
```

```
    as element(ns3:getBranchListResponse) {
        <ns3:getBranchListResponse>
            <ns1:header>{ $header1/@* , $header1/node() }</ns1:header>
            <ns3:body>
            {   let $branchName :=  concat(upper-
case(fn:substring(data($getBranchList1/ns3:body/ns4:branchName), 1, 1)) , lower-
case(fn:substring(data($getBranchList1/ns3:body/ns4:branchName), 2)))
                return
                if ( data($branchName) != "" and
data($getBranchList1/ns3:body/ns4:branchType) !="" ) then
                    <ns4:branchSummarySet>
                    {

                        for $branchId in $BranchesNetworkDVM/dvm:rows/dvm:row[
contains(dvm:cell[2], data($branchName)) and dvm:cell[15] =
data($getBranchList1/ns3:body/ns4:branchType)]/dvm:cell[1]
                                return
                          (
                            <ns4:branchSummary>
                                <ns2:branchId> {
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[1])
}</ns2:branchId>
                                <ns4:branchName>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[2])
}</ns4:branchName>
                                <ns4:branchAddress>
                                    <ns5:addressContact>
                                        <ns5:address>
                                            <ns5:line1>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[3])
}</ns5:line1>
                                            <ns5:state>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId
]/dvm:cell[14]) }</ns5:state>
                                            <ns5:phoneContactSet>
                                                <ns5:phoneContact
type="Landline">
                                                    <ns5:phone>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[9])
}</ns5:phone>
                                                </ns5:phoneContact>
                                                <ns5:phoneContact
type="Fax">
                                                    <ns5:phone>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId
]/dvm:cell[10]) }</ns5:phone>
                                                </ns5:phoneContact>
                                            </ns5:phoneContactSet>
                                        </ns5:address>
                                    </ns5:addressContact>
                                    <ns4:geographicLocation>
                                        <ns4:geographicCoordinates>
                                        <ns4:longitude>{
fn:replace(data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId
]/dvm:cell[5]) , " ", "") }</ns4:longitude>
```

```
                                                    <ns4:latitude>{
fn:replace(data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId
]/dvm:cell[4]) , " ", "") }</ns4:latitude>
                                                        </ns4:geographicCoordinates>
                                                    </ns4:geographicLocation>

                                                </ns4:branchAddress>

                                            </ns4:branchSummary>
                                        )
                                    }
                                </ns4:branchSummarySet>
                            else if ( data($branchName) != "" ) then
                                    <ns4:branchSummarySet>
                                {
                                    for $branchId in
$BranchesNetworkDVM/dvm:rows/dvm:row[ contains(dvm:cell[2],
data($branchName))]/dvm:cell[1]
                                                    return
                                    (

                                                <ns4:branchSummary>
                                                    <ns2:branchId> {
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[1])
}</ns2:branchId>
                                                    <ns4:branchName>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[2])
}</ns4:branchName>
                                                    <ns4:branchAddress>
                                                        <ns5:addressContact>
                                                            <ns5:address>
                                                                <ns5:line1>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[3])
}</ns5:line1>
                                                                <ns5:state>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId
]/dvm:cell[14]) }</ns5:state>

   <ns5:phoneContactSet>
   <ns5:phoneContact type="Landline">
   <ns5:phone>{ data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
$branchId ]/dvm:cell[9]) }</ns5:phone>
   </ns5:phoneContact>
   <ns5:phoneContact type="Fax">
   <ns5:phone>{ data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
$branchId ]/dvm:cell[10]) }</ns5:phone>
   </ns5:phoneContact>
   </ns5:phoneContactSet>
</ns5:address>
</ns5:addressContact>
<ns4:geographicLocation>
   <ns4:geographicCoordinates>
         <ns4:longitude>{ fn:replace(data($BranchesNetworkDVM/dvm:rows/dvm:row[
dvm:cell[1] = $branchId ]/dvm:cell[5]) , " ", "") }</ns4:longitude>
         <ns4:latitude>{ fn:replace(data($BranchesNetworkDVM/dvm:rows/dvm:row[
```

```
dvm:cell[1] = $branchId ]/dvm:cell[4]) , " ", "") }</ns4:latitude>
    </ns4:geographicCoordinates>
</ns4:geographicLocation>
</ns4:branchAddress>
</ns4:branchSummary>
                                    )
                                  }
                                </ns4:branchSummarySet>
                                else if (
data($getBranchList1/ns3:body/ns4:branchType) !="" ) then
                                   <ns4:branchSummarySet>
                                {

                                    for $branchId in
$BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[15] =
data($getBranchList1/ns3:body/ns4:branchType)]/dvm:cell[1]
                                              return
                                     (

                                          <ns4:branchSummary>
                                              <ns2:branchId> {
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[1])
}</ns2:branchId>
                                              <ns4:branchName>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[2])
}</ns4:branchName>
                                              <ns4:branchAddress>
                                                    <ns5:addressContact>
                                                          <ns5:address>
                                                                <ns5:line1>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId ]/dvm:cell[3])
}</ns5:line1>
                                                                <ns5:state>{
data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId
]/dvm:cell[14]) }</ns5:state>

    <ns5:phoneContactSet>

    <ns5:phoneContact type="Landline">

    <ns5:phone>{ data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
$branchId ]/dvm:cell[9]) }</ns5:phone>
    </ns5:phoneContact>
    <ns5:phoneContact type="Fax">
    <ns5:phone>{ data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] =
$branchId ]/dvm:cell[10]) }</ns5:phone>
    </ns5:phoneContact>
    </ns5:phoneContactSet>
    </ns5:address>
    </ns5:addressContact>
    <ns4:geographicLocation>
          <ns4:geographicCoordinates>
                <ns4:longitude>{
fn:replace(data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId
]/dvm:cell[5]) , " ", "") }</ns4:longitude>
```

```
                <ns4:latitude>{
fn:replace(data($BranchesNetworkDVM/dvm:rows/dvm:row[ dvm:cell[1] = $branchId
]/dvm:cell[4]) , " ", "") }</ns4:latitude>
            </ns4:geographicCoordinates>
      </ns4:geographicLocation>
</ns4:branchAddress>
</ns4:branchSummary>
                                    )
                                }
                            </ns4:branchSummarySet>
                    else(
                            <ns4:branchSummarySet/>
                    )
                    }
                </ns3:body>
            </ns3:getBranchListResponse>
};
declare variable $header1 as element(ns1:header) external;
declare variable $getBranchList1 as element(ns3:getBranchList) external;
declare variable $BranchesNetworkDVM as element(*) external;
xf:getBranchList_to_getBranchListResponse_DVM($header1,
    $getBranchList1, $BranchesNetworkDVM)
```

- XSD of the ivs (XSD of the evs is similar)

```
<xsd:schema targetNamespace="http://www.banqueaudi.com/chn.brn.ivs.Branch"
    elementFormDefault="qualified" version="1.0"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.banqueaudi.com/chn.brn.ivs.Branch"
    xmlns:hdr="http://www.banqueaudi.com/ebo/util.cmn.EBMHeader"
    xmlns:condet="http://www.banqueaudi.com/ebo/pty.party.ContactDetails"
    xmlns:brn="http://www.banqueaudi.com/ebo/svl.brn.Branch"
xmlns:id="http://www.banqueaudi.com/ebo/svl.cmn.Identifiers"
    xmlns:bsc="http://www.banqueaudi.com/ebo/util.cmn.Basic">
    <xsd:import namespace="http://www.banqueaudi.com/ebo/util.cmn.EBMHeader"
          schemaLocation="../../core/ebo/util.cmn.EBMHeader_1.0.xsd" />
    <xsd:import namespace="http://www.banqueaudi.com/ebo/pty.party.ContactDetails"
          schemaLocation="../../core/ebo/pty.party.ContactDetails_1.0.xsd" />
    <xsd:import namespace="http://www.banqueaudi.com/ebo/svl.brn.Branch"
          schemaLocation="../../core/ebo/svl.brn.Branch_1.0.xsd" />
    <xsd:import namespace="http://www.banqueaudi.com/ebo/svl.cmn.Identifiers"
          schemaLocation="../../core/ebo/svl.cmn.Identifiers_1.0.xsd" />
    <xsd:import namespace="http://www.banqueaudi.com/ebo/util.cmn.Basic"
          schemaLocation="../../core/ebo/util.cmn.Basic_1.0.xsd" />
    <xsd:complexType name="GetBranchDetailsType">
          <xsd:sequence>
                <xsd:element ref="hdr:header" />
                <xsd:element name="body">
                      <xsd:complexType>
                            <xsd:sequence>
                                  <xsd:element ref="id:branchId" />
                            </xsd:sequence>
                      </xsd:complexType>
                </xsd:element>
          </xsd:sequence>
    </xsd:complexType>
```

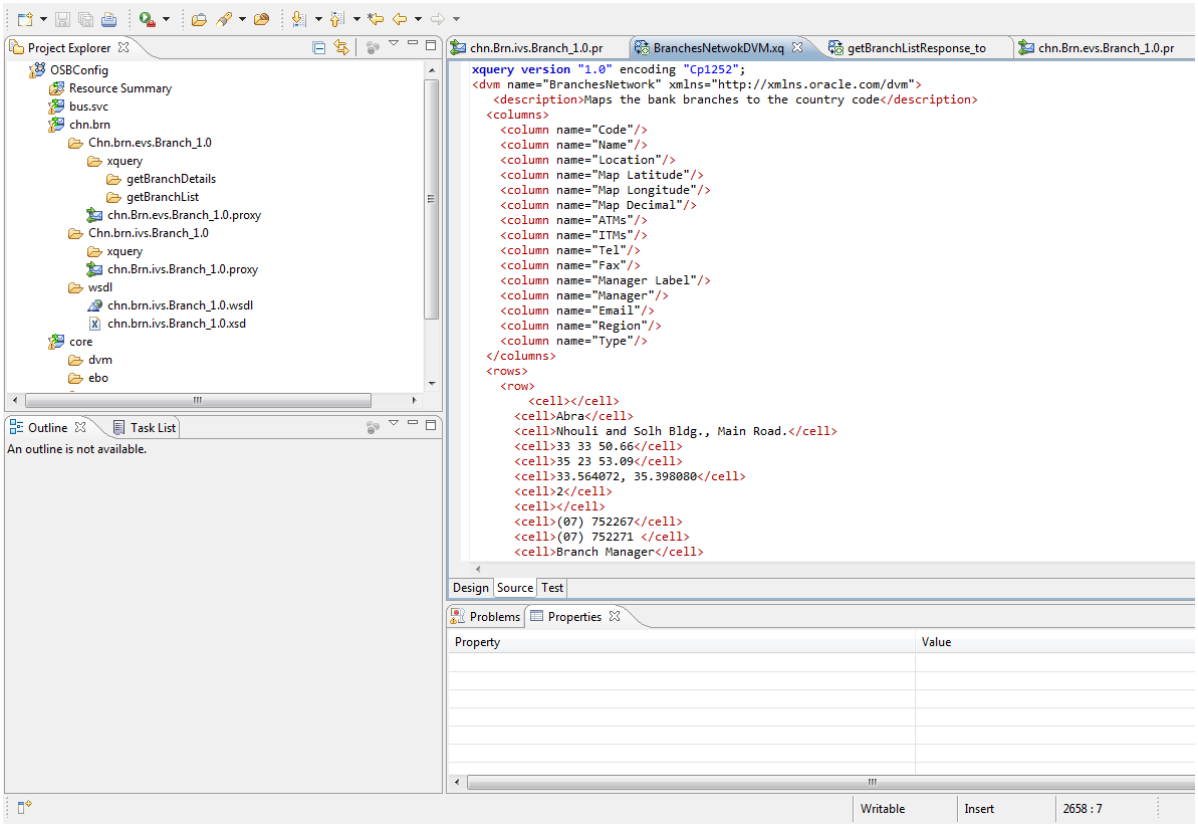
```
    <xsd:element name="getBranchDetails" type="GetBranchDetailsType" />
    <xsd:complexType name="GetBranchDetailsResponseType">
          <xsd:sequence>
                <xsd:element ref="hdr:header" />
                <xsd:element name="body">
                      <xsd:complexType>
                            <xsd:sequence>
                                  <xsd:element ref="brn:branch" />
                            </xsd:sequence>
                      </xsd:complexType>
                </xsd:element>
          </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="getBranchDetailsResponse"
type="GetBranchDetailsResponseType" />
    <xsd:simpleType name="StatusType">
          <xsd:restriction base="xsd:string">
                <xsd:enumeration value="Physical"/>
                <xsd:enumeration value="ITM"/>
          </xsd:restriction>
    </xsd:simpleType>
    <xsd:complexType name="GetBranchListType">
          <xsd:sequence>
                <xsd:element ref="hdr:header" />
                <xsd:element name="body">
                      <xsd:complexType>
                            <xsd:sequence>
                                  <xsd:element ref="brn:branchName"
minOccurs="0"/>
                                  <xsd:element name="status"
type="bsc:StatusType" minOccurs="0"/>
                                  <xsd:element ref="brn:branchType"
minOccurs="0"/>
                                  <xsd:element ref="brn:branchAddress"
minOccurs="0"/>
                                  <xsd:element ref="brn:regionalOffice"
minOccurs="0"/>
                            </xsd:sequence>
                      </xsd:complexType>
                </xsd:element>
          </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="getBranchList" type="GetBranchListType" />
    <xsd:complexType name="GetBranchListResponseType">
          <xsd:sequence>
                <xsd:element ref="hdr:header" />
                <xsd:element name="body">
                      <xsd:complexType>
                            <xsd:sequence>
                                  <xsd:element ref="brn:branchSummarySet"/>
                            </xsd:sequence>
                      </xsd:complexType>
                </xsd:element>
          </xsd:sequence>
    </xsd:complexType>
```

```
    <xsd:element name="getBranchListResponse" type="GetBranchListResponseType" />
</xsd:schema>
```

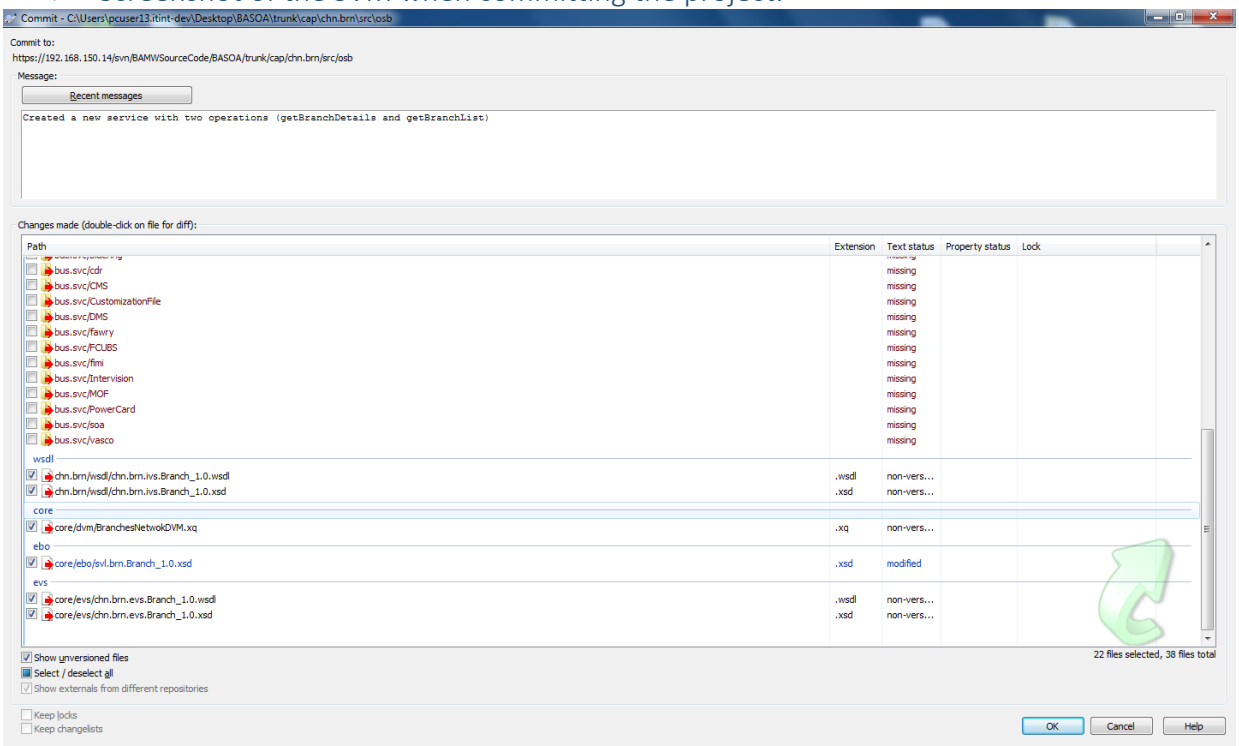
- WSDL of the ivs (WSDL of the evs is similar)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://www.banqueaudi.com/chn.brn.ivs.Branch"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  name="chn.brn.ivs.Branch"
  targetNamespace="http://www.banqueaudi.com/chn.brn.ivs.Branch">
       <wsdl:types><schema xmlns="http://www.w3.org/2001/XMLSchema">
                    <import
namespace="http://www.banqueaudi.com/chn.brn.ivs.Branch"
schemaLocation="chn.brn.ivs.Branch_1.0.xsd"/></schema></wsdl:types>
  <wsdl:message name="getBranchDetails">
    <wsdl:part name="payload" element="tns:getBranchDetails"/></wsdl:message>
  <wsdl:message name="getBranchDetailsResponse">
    <wsdl:part name="payload" element="tns:getBranchDetailsResponse"/>
  </wsdl:message>
  <wsdl:message name="getBranchList">
    <wsdl:part name="payload" element="tns:getBranchList"/></wsdl:message>
  <wsdl:message name="getBranchListResponse">
    <wsdl:part name="payload" element="tns:getBranchListResponse"/>
  </wsdl:message>
  <wsdl:portType name="chn.brn.ivs.Branch">
    <wsdl:operation name="getBranchDetails">
      <wsdl:input message="tns:getBranchDetails"/>
      <wsdl:output message="tns:getBranchDetailsResponse"/>
    </wsdl:operation>
    <wsdl:operation name="getBranchList">
      <wsdl:input message="tns:getBranchList"/>
      <wsdl:output message="tns:getBranchListResponse"/>
    </wsdl:operation></wsdl:portType>
  <wsdl:binding name="chn.brn.ivs.Branch.Binding"
type="tns:chn.brn.ivs.Branch">
    <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getBranchDetails">
      <soap:operation soapAction="getBranchDetails" style="document"/>
      <wsdl:input><soap:body use="literal"/></wsdl:input>
      <wsdl:output><soap:body use="literal"/></wsdl:output></wsdl:operation>
    <wsdl:operation name="getBranchList">
      <soap:operation soapAction="getBranchList" style="document"/>
      <wsdl:input><soap:body use="literal"/></wsdl:input>
      <wsdl:output><soap:body use="literal"/></wsdl:output>
    </wsdl:operation></wsdl:binding></wsdl:definitions>
```

➢ Screenshot of the DVM on OSB



➢ Screenshot of the SVM when committing the project:



➢ Code of the Mobile app interface:

```
<!DOCTYPE html>
<html><head>
    <meta charset="utf-8" />
    <title> Bank Audi </title>
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1" />
    <link rel="stylesheet" href="jquery-mobile/jquery.mobile.css" />
    <script src="jquery-mobile/jquery.js"></script>
    <script src="jquery-mobile/jquery.mobile.js"></script>
    <link rel="stylesheet" href="css/style.css" /></head>
  <body>
    <div id="home" class="background" data-role="page" data-title="View: Home" data-theme="b">
      <div data-role="header"><h1> <img class="logo" src="images/audi.png" /></h1>
          <a href="#" data-transition="flip" data-icon="home" data-iconpos="notext"> Home</a>
        <a href="#info" data-icon="info" data-rel="dialog" data-iconpos="notext" class="menuButton ui-btn-right"> info</a>
      </div>
      <div data-role="content">
<a href="#branch" data-role="button"data-transition="none" data-icon="arrow-r" data-iconpos="right"> Branch/ATM Locator </a>
      </div>
      <div data-role="footer" data-position="fixed">
          <a href="#" data-role="button"  data-icon="gred"></a><a href="#" data-role="button"  data-icon="gred"></a>
          <a href="#" data-role="button"  data-icon="gred"></a></div></div>
<div id="branch" class="background" data-role="page" data-theme="b">
          <div data-role="header">
                  <h1><img class="logo" src="images/audi.png" /></h1>
                  <a href="#home" data-transition="none" data-icon="home" data-iconpos="notext"> Home</a>
                  <a href="javascript:history.back()" data-transition="none" data-icon="back" data-iconpos="notext">Back </a>
          </div>
          <div data-role="content">
                  <form action="BranchList.php">
                          <input type="text" name="BranchName" placeholder="Branch Name">
                          <input type="text" name="BranchType" placeholder="Branch Type">
                          <input type="submit" value="Submit">
                  </form></div></div>
          <div id="info" class="background" data-role="page" data-theme="b">
                          <div data-role="header">
                                  <h1><img class="logo" src="images/audi.png" /></h1>
                          </div>
                          <div data-role="content" >
                                  <p>this application is for bank audi</p>
                          </div></div>
    <script>$.mobile.defaultPageTransition = "pop"; </script></body></html>
```

BranchList.PHP:

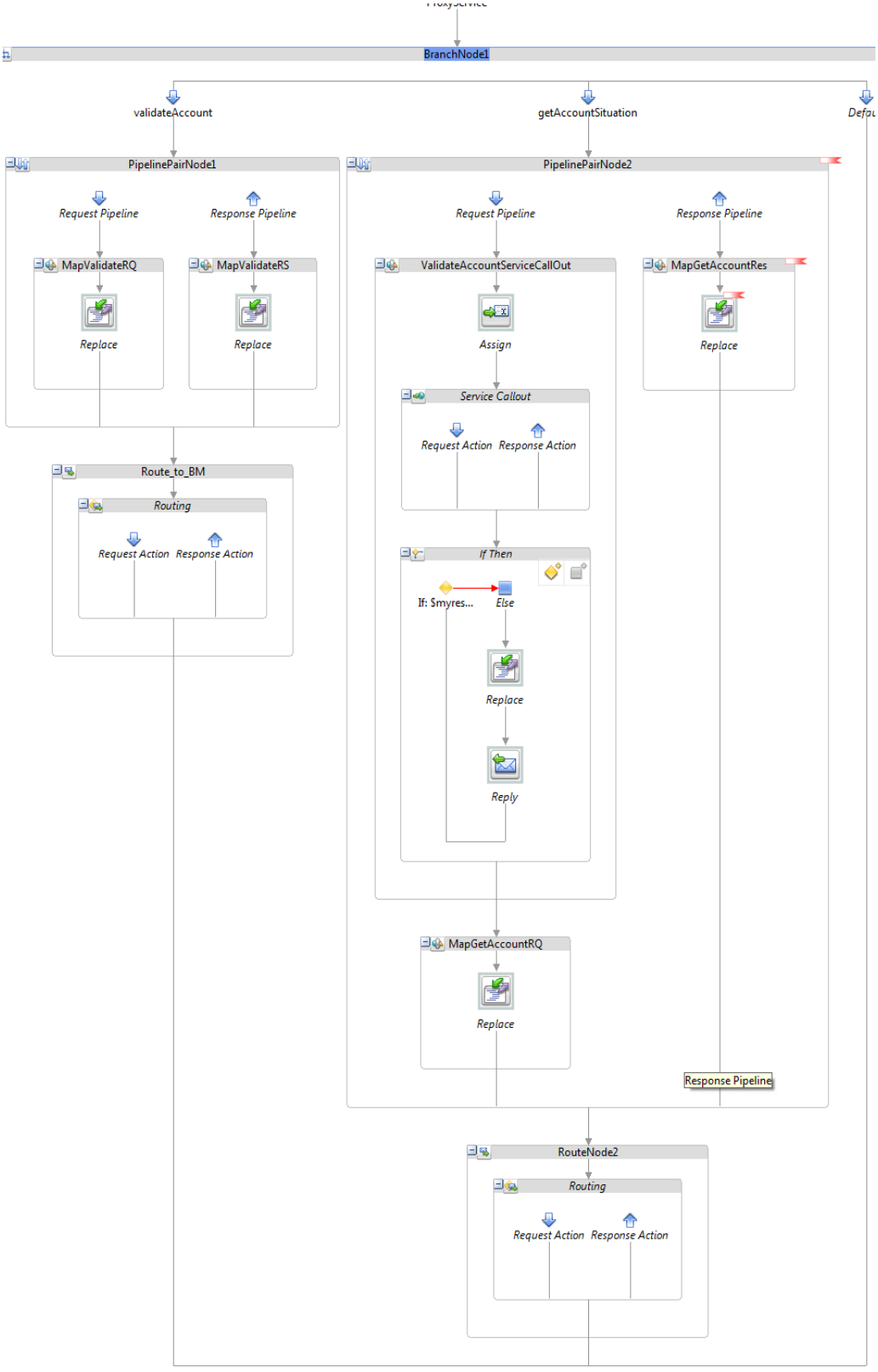
```
<?php
require_once 'lib/nusoap.php';
require 'functions.php';
$BranchName = $_GET['BranchName'];
$BranchType = $_GET['BranchType'];
$wsdl = "http://www.banqueaudi.com/chn.brn.evs.Branch";
$output = $client->__soapcall('functions.service', array($BranchName, $BranchType));
$client = new soapclient($wsdl);
echo json_encode($output);
?>
```

BranchDetails.php:

```
<?php
require_once 'lib/nusoap.php';
require 'functions.php';
$BranchName = $_GET['BranchId'];
$wsdl = "http://www.banqueaudi.com/chn.brn.evs.Branch";
$output = $client->__soapcall('functions.service', array($BranchId));
$client = new soapclient($wsdl);
echo json_encode($output);
?>
```

Tasks (2 sample):

- Screenshot of the architecture of the third task:



Xquery Code for transforming from BankMate to proxy

```
(:: pragma bea:global-element-parameter parameter="$mwaccsitResponse1" element="ns13:mwaccsitResponse"
location="../Business/wsdl/schema/MWACCSITServicesService_schema1.xsd" ::)
(:: pragma bea:global-element-return element="ns0:getAccountSituationResponse" location="../pp.acc.ivs.Account_1.1.xsd" ::)
declare namespace ns14 = "http://www.banqueaudi.com/ebo/pty.org.Organisation";
declare namespace ns15 = "http://www.banqueaudi.com/ebo/pty.party.ContactDetails";
declare namespace ns9 = "http://www.banqueaudi.com/ebo/pty.person.Person";
declare namespace ns5 = "http://www.banqueaudi.com/ebo/pty.cust.Customer";
declare namespace ns12 = "http://www.banqueaudi.com/ebo/util.cmn.EBMHeader";
declare namespace ns13 = "http://mwaccsit.wsbeans.iseries/";
declare namespace ns6 = "http://www.banqueaudi.com/ebo/psa.acc.Account";
declare namespace ns7 = "http://www.banqueaudi.com/ebo/psa.cmn.Identifiers";
declare namespace ns10 = "http://www.banqueaudi.com/ebo/ent.cmn.Identifiers";
declare namespace ns8 = "http://www.banqueaudi.com/ebo/svl.brn.Branch";
declare namespace ns11 = "http://www.banqueaudi.com/ebo/pty.cmn.Identifiers";
declare namespace ns2 = "http://www.banqueaudi.com/ebo/util.cmn.Basic";
declare namespace ns1 = "http://www.banqueaudi.com/ebo/psa.accdet.Account";
declare namespace ns4 = "http://www.banqueaudi.com/ebo/svl.cmn.Identifiers";
declare namespace ns3 = "http://www.banqueaudi.com/ebo/pty.party.Party";
declare namespace ns0 = "http://www.banqueaudi.com/ivs/pp.acc.Account";
declare namespace xf = "http://tempuri.org/Project%201/xquery/getAccountBM_to_MW/";
declare namespace dvm = "http://xmlns.oracle.com/dvm";
declare namespace local="http://local.home";
declare function local:absolute($parameter as xs:decimal)
as xs:decimal?
{
          if ($parameter < 0)
          then
                    (
                    let $numb := $parameter * (-1)
                    return $numb
                    )
          else
                    (
                    let $numb := $parameter
                    return $numb
                    )
};
declare function xf:getAccountBM_to_MW($mwaccsitResponse1 as element(ns13:mwaccsitResponse), $CurrencyCodesDVM as element(*),
$BlockedForDVM as element(*))
    as element(ns0:getAccountSituationResponse) {
        <ns0:getAccountSituationResponse>
           <ns12:header>
                    <ns12:ebmCID>"Karim"</ns12:ebmCID>
                    <ns12:ebmMID>{ fn-bea:uuid() }</ns12:ebmMID>
                    <ns12:ebmRTID>{ fn-bea:uuid() }</ns12:ebmRTID>
                    <ns12:ebmPID>{ fn-bea:uuid() }</ns12:ebmPID>
                    <ns12:ebmSID>{ "112" }</ns12:ebmSID>
                    <ns12:ebmTimestamp>{ fn:current-dateTime() }</ns12:ebmTimestamp>
                    <ns12:propertySet>
          </ns12:propertySet>
          </ns12:header>

           <ns0:body>
               <ns6:accountSet>
               {
                for $acc in $mwaccsitResponse1/return/responseBody/account
            (:   for $acc in (1 to $mwaccsitResponse1/return/responseBody/accountCount ) :)
                    return
                (
                  <ns6:account>
                      <ns7:accountIdSet>
                          <ns7:accountId domiciliation = "Local"
                                        format = "{
                                                            if (data($acc/accountType) = "CUSTOMER")
                                                            then "Local"
                                                            else data($acc/accountType)
                                                            }">
                                    { data($acc/accountNumber) }
                          </ns7:accountId>
                      </ns7:accountIdSet>
                      <ns6:status>
                              <ns2:status>
                                        {data($acc/accountStatus)}
                              </ns2:status>
                                                  <ns2:description>
                                                  {
                                                            if (data($acc/accountStatus)="0")
                                                            then "Active"
                                                            else if (data($acc/accountStatus)="9")
                                                            then "Dormant"
                                                            else "Blocked"
                                                  }
                                                  </ns2:description>

                                                  {
                                                            if (data($acc/accountStatus)="0")
                                                            then
                                                            (          if ($BlockedForDVM/dvm:rows/dvm:row[ dvm:cell[6] =
"None" ]/dvm:cell[6])
                                                                                then ()
```

```
                                                                                else
<ns6:blockedFor>{$BlockedForDVM/dvm:rows/dvm:row/dvm:cell[6]}</ns6:blockedFor>
                                                                    )
                                                                    else if (data($acc/accountStatus)="3")
                                                                    then
                                                                    (
                                                                                if (data($acc/accountClosingReason)="01")
                                                                                then <ns6:blockedFor>"DR"</ns6:blockedFor>
                                                                                else if (data($acc/accountClosingReason)="02")
                                                                                then <ns6:blockedFor>"CR"</ns6:blockedFor>
                                                                                else <ns6:blockedFor>"Error"</ns6:blockedFor>
                                                                    )
                                                                    else <ns6:blockedFor>"Both"</ns6:blockedFor>
                                                   }

                        </ns6:status>
                        {
                                if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/accountCurrencyCode)
]/dvm:cell[2])
                                then
                                          (
                                <ns6:currency>
                                          <ns2:code>
                                                    {data($acc/accountCurrencyCode)}
                                          </ns2:code>
                                          <ns2:number>
                                                              {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($acc/accountCurrencyCode) ]/dvm:cell[2])}
                                                    </ns2:number>
                                          </ns6:currency>
                                          )
                       else
                                ""
                       }
                        <ns6:balances>
                            <ns6:book>
                                      <ns6:value
                                      debitOrCredit=
                                          "{        if ($acc/bookBalance <0)
                                                    then "DR"
                                                    else "CR"
                                          }"
                                                    >
                                <ns2:amount>{local:absolute($acc/bookBalance)}</ns2:amount>
                                                    {
                                                                      if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($acc/accountCurrencyCode) ]/dvm:cell[2])
                                                                      then
                                                                                (

                                                                      <ns2:currency>

                                                                                <ns2:code>
                                                                                          {data($acc/accountCurrencyCode)}
                                                                                </ns2:code>
                                                                                <ns2:number>

          {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/accountCurrencyCode) ]/dvm:cell[2])}
                                                                                          </ns2:number>
                                                                                </ns2:currency>

                                                                                )
                                                             else
                                                                      ""
                                                             }
                                          </ns6:value>
                                          <ns6:counterValue
                                          debitOrCredit=
                                          "{        if ($acc/cvBookBalance <0)
                                                    then "DR"
                                                    else "CR"
                                          }"
                                          >
                                                    <ns2:amount>{local:absolute($acc/cvBookBalance)}</ns2:amount>



                                                    {
                                                                      if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($mwaccsitResponse1/return/responseBody/account[1]/consolidationCurrency) ]/dvm:cell[2])
                                                                      then
                                                                                (

                                                                      <ns2:currency>

                                                                                <ns2:code>

          {data($acc/consolidationCurrency)}
                                                                                </ns2:code>
                                                                                <ns2:number>

          {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/consolidationCurrency) ]/dvm:cell[2])}
```

```
                                                                                                </ns2:number>
                                                                                    </ns2:currency>

                                                                                    )
                                                                        else
                                                                                ""
                                                                        }
                                                </ns6:counterValue>
                                </ns6:book>
                                <ns6:available>
                                            <ns6:value
                                            debitOrCredit=
                                                "{          if ($acc/availableBalance <0)
                                                            then "DR"
                                                            else "CR"
                                                }"
                                                            >
                <ns2:amount>{local:absolute($acc/availableBalance)}</ns2:amount>

                                                            {
                                    if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/accountCurrencyCode)
]/dvm:cell[2])
                                    then
                                                (

                                    <ns2:currency>

                                                <ns2:code>
                                                            {data($acc/accountCurrencyCode)}
                                                </ns2:code>
                                                <ns2:number>
                                                                        {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($acc/accountCurrencyCode) ]/dvm:cell[2])}
                                                            </ns2:number>
                                                </ns2:currency>

                                                )
                        else
                                    ""
                        }

                                                </ns6:value>
                                                <ns6:counterValue
                                                debitOrCredit=
                                                "{          if ($acc/cvAvailableBalance <0)
                                                            then "DR"
                                                            else "CR"
                                                }"
                                                            >
                                                            <ns2:amount>{local:absolute($acc/cvAvailableBalance)}</ns2:amount>

                                                {
                                                                                if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($acc/consolidationCurrency) ]/dvm:cell[2])
                                                                                then
                                                                                            (

                                                                                <ns2:currency>

                                                                                            <ns2:code>

            {data($acc/consolidationCurrency)}
                                                                                            </ns2:code>
                                                                                            <ns2:number>

            {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/consolidationCurrency) ]/dvm:cell[2])}
                                                                                                        </ns2:number>
                                                                                            </ns2:currency>

                                                                                            )
                                                                        else
                                                                                ""
                                                                        }

                                                </ns6:counterValue>
                                </ns6:available>
                                <ns6:valueDate>
                                            <ns6:value
                                            debitOrCredit=
                                                "{          if ($acc/onlineBookBalance <0)
                                                            then "DR"
                                                            else "CR"
                                                }"
                                                            >
                <ns2:amount>{local:absolute($acc/onlineBookBalance)}</ns2:amount

                                                            {
                                    if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/accountCurrencyCode)
]/dvm:cell[2])
                                    then
```

```
                                        (
                                    <ns2:currency>
                                        <ns2:code>
                                            {data($acc/accountCurrencyCode)}
                                        </ns2:code>
                                        <ns2:number>
                                                {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($acc/accountCurrencyCode) ]/dvm:cell[2])}
                                            </ns2:number>
                                        </ns2:currency>
                                        )
                            else
                                    ""
                            }
                                        </ns6:value>
                                        <ns6:counterValue
                                        debitOrCredit=
                                        "{          if ($acc/cvOnlineBookBalance <0)
                                                    then "DR"
                                                    else "CR"
                                        }"
                                                    >
                                                    <ns2:amount>{local:absolute($acc/cvOnlineBookBalance)}</ns2:amount>

                                                    {
                                                                                    if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($acc/consolidationCurrency) ]/dvm:cell[2])
                                                                                    then
                                                                                        (
                                                                                    <ns2:currency>
                                                                                        <ns2:code>
            {data($acc/consolidationCurrency)}
                                                                                        </ns2:code>
                                                                                        <ns2:number>
            {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/consolidationCurrency) ]/dvm:cell[2])}
                                                                                                    </ns2:number>
                                                                                        </ns2:currency>
                                                                                        )
                                                                                else
                                                                                    ""
                                                                                }

                                        </ns6:counterValue>
                                </ns6:valueDate>
                                <ns6:netBalance>
                                            <ns6:value
                                            debitOrCredit=
                                                "{          if ($acc/onlineAvailableBalance <0)
                                                            then "DR"
                                                            else "CR"
                                                }"
                                                            >
            <ns2:amount>{local:absolute($acc/onlineAvailableBalance)}</ns2:amount>
                                                            {
                                    if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/accountCurrencyCode)
]/dvm:cell[2])
                                    then
                                        (
                                    <ns2:currency>
                                        <ns2:code>
                                            {data($acc/accountCurrencyCode)}
                                        </ns2:code>
                                        <ns2:number>
                                                {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($acc/accountCurrencyCode) ]/dvm:cell[2])}
                                            </ns2:number>
                                        </ns2:currency>
                                        )
                            else
                                    ""
                            }
                                        </ns6:value>
                                        <ns6:counterValue
                                                    debitOrCredit=
                                        "{          if ($acc/cvOnlineAvailableBalance <0)
                                                    then "DR"
```

```
                                                    else "CR"
                                                }"
                                                >
            <ns2:amount>{local:absolute($acc/cvOnlineAvailableBalance)}</ns2:amount>
                                                        {
                                                                if ($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] =
data($acc/consolidationCurrency) ]/dvm:cell[2])
                                                                then
                                                                        (
                                                                <ns2:currency>
                                                                        <ns2:code>
            {data($acc/consolidationCurrency)}
                                                                        </ns2:code>
                                                                        <ns2:number>
            {data($CurrencyCodesDVM/dvm:rows/dvm:row[ dvm:cell[1] = data($acc/consolidationCurrency) ]/dvm:cell[2])}
                                                                        </ns2:number>
                                                                </ns2:currency>
                                                                        )
                                                                else
                                                                        ""
                                                                }
                                                    </ns6:counterValue>
                                        </ns6:netBalance>
                                    </ns6:balances>
                                </ns6:account>
                                )}
                            </ns6:accountSet>
                        </ns0:body>
                    </ns0:getAccountSituationResponse>
};
declare variable $mwaccsitResponse1 as element(ns13:mwaccsitResponse) external;
declare variable $CurrencyCodesDVM as element(*) external;
declare variable $BlockedForDVM as element(*) external;
xf:getAccountBM_to_MW($mwaccsitResponse1, $CurrencyCodesDVM, $BlockedForDVM)
```