Självständigt arbete på grundnivå

Independent degree project - first cycle

Datateknik
Computer Engineering

Exploring IBM Integration Designer

Jonathan Ellström
Abstract

The interest for Business Process Management (BPM) is increasing in Sweden. Government agencies such as the Swedish Nation Board of Student Aid (CSN), the Swedish Companies Registration Office (Bolagsverket) and the Swedish Social Insurance Agency (Försäkringskassan) are implementing BPM into their organizations. Sogeti is an IT-consulting company that has employees that works at CSN with BPM, and one of the tools they use for integration is IBM Integration Designer. Since this technology is new and is gaining popularity, there is a need for increased knowledge about it. This thesis report explores the tool IBM Integration Designer, regarding its different ways to integrate with systems and compares the different options for exposing the integration solutions. By researching documentation from IBM, and by learning how to use the tool itself, knowledge could be acquired about IBM Integration Designer. The result was an implementation of the five different export bindings: SCA, HTTP, SOAP over HTTP, Enterprise JavaBeans (EJB) and Java Messaging Service (JMS), a comparison of these export bindings and finally a service in IBM Integration Designer that accesses a database and uses an external SMS API to send text messages. The result has been satisfactory to the purpose of this project in giving insight into IBM Integration Designer, one of the popular tools for integrating BMP.

Keywords: BPM, CSN, SCA, HTTP, SOAP, EJB, JMS, SMS, API.
# Table of Contents

**Abstract** .......................................................................................................................... iii  
**Terminology** ................................................................................................................... vi  
**Acronyms/Abbreviations** ............................................................................................... vi  
1 **Introduction** .................................................................................................................. 1  
   1.1 Background and problem motivation........................................................................ 1  
   1.2 Overall aim.................................................................................................................. 1  
   1.3 Scope........................................................................................................................... 1  
   1.4 Concrete and verifiable goals ..................................................................................... 2  
   1.5 Outline......................................................................................................................... 2  
2 **Theory** .......................................................................................................................... 3  
   2.1 Business Process Management.................................................................................. 3  
   2.2 IBM Integration Designer.......................................................................................... 3  
   2.3 Service-oriented Architecture..................................................................................... 4  
   2.4 Service Component Architecture................................................................................ 4  
   2.4.1 Modules................................................................................................................... 4  
   2.4.2 Imports and exports............................................................................................... 5  
   2.4.3 Service components............................................................................................... 5  
   2.4.3.1 BPEL processes................................................................................................. 5  
   2.5 Simple Object Access Protocol................................................................................... 6  
   2.6 Java Messaging Service.............................................................................................. 7  
   2.7 Enterprise JavaBeans................................................................................................. 7  
   2.8 Maven........................................................................................................................ 8  
   2.9 Twilio.......................................................................................................................... 8  
   2.10 Plivo............................................................................................................................ 9  
   2.11 BulkSMS.................................................................................................................. 9  
   2.12 MariaDB................................................................................................................... 9  
   2.13 Proof-of-concept....................................................................................................... 9  
   2.14 Advanced Rest Client.............................................................................................. 9  
   2.15 SoapUI..................................................................................................................... 10  
3 **Methodology** ................................................................................................................ 11  
   3.1 Theoretical background study................................................................................... 11  
   3.2 Tools.......................................................................................................................... 11  
   3.3 Comparison methodology......................................................................................... 11  
   3.4 Development methodology....................................................................................... 11  
   3.4.1 Sprints.................................................................................................................... 12  
   3.4.2 Kanban board......................................................................................................... 12  
   3.5 Result Verification Methodology.............................................................................. 13  
4 **Implementation** ............................................................................................................ 14  
   4.1 Export binding implementation................................................................................... 14  
   4.1.1 SCA export binding............................................................................................... 14  
   4.1.2 HTTP export binding........................................................................................... 15
## Terminology

This chapter presents the acronyms and abbreviations which are used in the report.

### Acronyms/Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BPEL</td>
<td>Business Process Execution Language</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise JavaBeans</td>
</tr>
<tr>
<td>HTTP</td>
<td>HyperText Transfer Protocol</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Messaging Service</td>
</tr>
<tr>
<td>JNDI</td>
<td>Java Naming Directory Interface</td>
</tr>
<tr>
<td>POC</td>
<td>Proof-of-concept</td>
</tr>
<tr>
<td>SCA</td>
<td>Service Component Architecture</td>
</tr>
<tr>
<td>SMS</td>
<td>Short Message Service</td>
</tr>
<tr>
<td>SOA</td>
<td>Service Oriented Architecture</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

This chapter gives an introduction to the project. The background and problem motivation are described first, followed by the overall aim, the scope, the concrete and verifiable goals and finally an overview of the different parts of the report.

1.1 Background and problem motivation

Some of the government agencies in Sweden such as the Swedish Nation Board of Student Aid (CSN), the Swedish Companies Registration Office (Bolagsverket) and the Swedish Social Insurance Agency (Försäkringskassan) are implementing Business Process Management (BPM) into their organizations.

CSN is using multiple of IBM’s tools to upgrade their old systems which are used by their case workers daily to manage task such as payments, loans, and discounts. For upgrading the front-end they are using the tool IBM Business Process Manager and for the integration, they are using the tool IBM Integration Designer.

Due to BPM being a technology with increasing popularity, there is a need for increasing the knowledge about it to meet the demand of the market. In Sundsvall, Sweden, the organizations need for competence within this area is so big that they hire consultants from India to travel and work with BPM. Sogeti [1] is an IT consulting company which have employees working as consultants with BPM at CSN. This thesis has been conducted at Sogeti with the assistance from consultants working at CSN.

1.2 Overall aim

The purpose of this thesis is to explore one of the tools for integrating BPM, IBM Integration Designer, regarding its different ways to integrate with systems and the different possibilities to expose the integration solutions.

1.3 Scope

Due to the time required to get used to IBM Integration Designer, and the time-limitation for the thesis, there has to be a scope, a focus, for this thesis. This focus will be put on comparing the different options to expose a Service Component Architecture (SCA) module in IBM Integration Designer by using different export bindings. An SCA module [2] determines which artifacts that are combined in an Enterprise Archive (EAR) file. In a module, there are different parts, such as imports, exports and service components. Examples for methods for exposing an SCA module is HTTP or SOAP. The focus will also be on creating a service in IBM Integration designer that queries a database and use an external API to send text messages.
1.4 Concrete and verifiable goals

For this project, there are three concrete and verifiable goals, where the first and second goals are closely connected:

1) Create a Service Component Architecture (SCA) module in the tool IBM Integration Designer, expose it and make a request to it using all of the following options:

- SCA export binding
- HTTP export binding
- SOAP over HTTP export binding
- Enterprise JavaBeans export binding
- JMS export binding

2) Make a comparison of the differences between the options to expose an SCA module regarding usage areas and the security options provided by IBM Integration Designer.

3) Create an SMS service as a proof-of-concept using IBM Integration Designer. This service will send a query to a database with an employee ID and receives a phone number as the response. Then use this phone number and a text to send a text message to the phone number using an external API.

1.5 Outline

Chapter 2 describes the theory had has been used for this thesis. Chapter 3 contains the information about the methods that have been used. Chapter 4 describes how the solution for the project has been implemented. Chapter 5 contains the results that have been accomplished and chapter 6 contains a discussion about the project and the conclusions that have been drawn.
2 Theory

This chapter gives an introduction to the theory that has been used during the project.

2.1 Business Process Management

Business Process Management (BPM) [3][4] is a discipline that functions as a process optimization process. BPM uses different services and software to provide the user with a total insight into the organization. This insight can help identify weaknesses and makes it easier to change the processes. This is done to increase the efficiency and reduce costs.

2.2 IBM Integration Designer

IBM Integration Designer [5][6] is one of the many software that can be used for implementing BPM. It uses the Service Component Architecture (SCA). IBM Integration Designer is used for the integration part and is used as middleware between raw data and other software. It is a development environment where the user can build and test automated services and processes. One example of this is an SCA module which is mentioned later in this theory chapter. To get an idea of what this software looks like, see figure 1 below.

Figure 1 – A view in IBM Integration Designer
2.3 **Service-oriented Architecture**

Service-Oriented architecture (SOA) [7] is a standard which has the purpose of separating implementation from integration logic. To do this, a middle layer is used, service components. These components can be created within IBM Integration designer. This design makes the components loosely coupled. See figure 2 below for a visualization of the three layers.

![Three layers of Service-Oriented Architecture (SOA)](image)

**Business integration logic**

**Service components**

**Implementation**

**Figure 2: The three layers of Service-Oriented Architecture (SOA)**

2.4 **Service Component Architecture**

Service Component Architecture (SCA) [8] is the backbone architecture of IBM Integration Designer and it contains a lot of different parts such as SCA modules, imports, and exports. It also makes it possible to have a service oriented architecture.

2.4.1 **Modules**

An SCA module [2] is a combination of parts such as processes, exports or imports. The modules result in service applications that can be tested and deployed to a process server. It has many similarities to a project, in the fact that it contains different aspects and can be run as an application.
2.4.2 Imports and exports

Imports and exports [9] are access points of a module. You can import services outside a module so that they can be called from within the module. You can expose the module to external components using exports.

There are different options for creating these imports and exports. Some of them are the following:

- Simple Object Access Protocol (SOAP)
- Hypertext transfer Protocol (HTTP)
- Java Message Service (JMS)
- Web service (SOAP over HTTP)
- Enterprise JavaBeans (EJB)

2.4.3 Service components

Service components [10] are created in IBM Integration Designer and they are the middleware in Service-Oriented Architecture (SOA). These components is the part in a SCA module that can be exported or imported. These services are combined in SCA modules to result in service applications that can be tested and deployed to the process server.

2.4.3.1 BPEL processes

Business Process Execution Language (BPEL) [11] is a standard industry language. A BPEL process is a service component that implements a business process. It can contain simple code and handle requests and responses.
2.5 Simple Object Access Protocol

Simple Object Access Protocol (SOAP) [12] is a protocol that uses XML to communicate. As seen in figure 3 on the next page, a soap body is wrapped and sent to the recipient. The response is also a soap body wrapped up.

![Figure 3 – SOAP request and response](image-url)
2.6 Java Messaging Service

Java Messaging Service (JMS) [13] is a messaging service which is asynchronous. It uses a connection factory to create a connection which creates a session that can create a message producer and consumer. See figure 4 below for a visualization.

![JMS Programming Model](image)

**Figure 4 – JMS Programming Model**

2.7 Enterprise JavaBeans

Enterprise JavaBeans (EJB) [14] is an API that is based on Java EE. It allows communication between Java EE business logic.
2.8 **Maven**

Maven [15] is an automated build-system which is developed by Apache. It is able to simplify the build process by creating components which can be included in the build path of an application.

2.9 **Twilio**

Twilio [16] is an API that can be used to exchange text messages. It has support for multiple programming languages and has code samples for PHP, Node, Python, Ruby, Java and .NET. See figure 5 below for viewing a Twilio API request that can be sent via their online interface and figure 6 for the received text message.

![Figure 5 – Twilio API request](image)

![Figure 6 – SMS from Twilio](image)
2.10 **Plivo**

Plivo [17] is an API that can be used to exchange text messages. This API supports unicode UTF-8 so it can send text messages in any language such as Arabic and Chinese.

2.11 **BulkSMS**

BulkSMS [18] is an API that can be used to exchange text messages. This API also allows scheduling of SMS messages to be sent at a specific time and date.

2.12 **MariaDB**

MariaDB [19] is a relational database using Structured Query Language (SQL). It is an open source version of SQL and it is developed by the same developers of MySQL.

2.13 **Proof-of-concept**

A proof of concept (POC) [20] is concrete evidence of the potential of a service or product.

2.14 **Advanced Rest Client**

Advanced Rest Client [21] is an application which allows the user to test existing APIs. The user can select HTTP Methods such as GET, POST, PUT or DELETE in the application and send requests. See figure 7 below for the interface of the application.

![Advanced Rest Client interface](image)

**Figure 7 – Advanced Rest Client interface**
2.15 SoapUI

SoapUI [22] is an application which is open-source. It can be used for testing SOAP and REST APIs. See figure 8 below for the interface for an SOAP UI request.

![Figure 8 – SOAP UI request interface](image)
3 Methodology

This chapter presents the methods that have been used. First off the method for the theoretical background study is described, followed by the tools used. After that the development methods used in the project and finally the method for verifying the results.

3.1 Theoretical background study

To gain the knowledge required to be able to use IBM Integration Designer for the purpose of this project, there had to be a background study based on IBM’s documentation. Getting a basic understanding of terms like Service Oriented Architecture, Service Component Architecture and modules was a prerequisite for being able to work with IBM Integration Designer. After a basic understanding was acquired, it was possible to move forward with experimenting with IBM Integration Designer and to work towards the result.

3.2 Tools

IBM Integration Designer has been the primary tool during this project. For the database that was used for the SMS service, XAMPP and MariaDB has been used. This database was chosen because the it is one of the most simple databases to use which was useful since the database used did not contain a lot of information. Previous experience with the MariaDB database was also a factor of the decision. The service that has been used for implementing the texting service is Twilio API. This was the first SMS API that was tested and it was easy to use and it served its purpose well.

3.3 Comparison methodology

As mentioned in the introduction chapter 1.4, the second concrete and verifiable goal was to compare the differences between the options to expose an SCA module regarding usage areas and options for security. This will be done in two major ways. The first one being reading IBM’s documentation and take note of the differences. The second way is by experimenting with IBM Integration Designer and manually explore the differences with usage areas and the options for enabling security for connecting to the export binding itself.

3.4 Development methodology

Sprints and a Kanban board has been methods used to aid the workflow and focus during the project. These methods are described in chapter 3.4.1 and in chapter 3.4.2. In the study “A survey study of critical success factors in agile software projects” [23], there are three critical factors for success. The first one is to have a correct delivery strategy. The second factor is to have agile-style software engineering techniques and the third one is to have a high-caliber team
capability. If these aspects are applied to this project, the first factor was to have a correct delivery strategy. The attributes of this according to the study is to have a regular delivery of software, and to deliver the most important features first. The main purpose of the sprints and kanban board is to achieve this goal. The sprints and kanban board also had the purpose of fulfilling the second factor of success mentioned in the study, agile software engineering techniques. One important attribute of this second success factor is to pursue simple design. This will be achieved by using methods such as simplifying the database and the code being used for the export bindings as much as possible. The third aspect, to have a high-caliber team capability, can be overlooked since this was not a team project.

3.4.1 Sprints

This project has been divided into three sprints which lasted three weeks each. After each sprint, there was a presentation which shed some light on the progress, the problems and what the way to move forward was. The sprints have been used to increase the efficiency of the development during the project by dividing the time-pressure from the final deadline into the different sprint deadlines. If something was not complete as expected at the end of a sprint, the time-pressure became more apparent which gave an insight that was useful for managing the remaining time.

3.4.2 Kanban board

Figure 9: Trello

A Kanban board, Trello, can be seen in figure 9. This Kanban board has been used to visualize and organize the tasks left to be done and the task can be divided into sub-tasks for an easier overview. The different tasks was sorted in order of importance for a prioritized to-do list. Trello has been used via their online website.
3.5 Result Verification Methodology

The chosen method to verify the results was to closely compare the result to the concrete and verifiable goals in chapter 1.4 of this report, that is, to see if the goals have been fulfilled and if the knowledge about the different capabilities and export bindings in IBM Integration Designer has been increased to a satisfactory level.
4 Implementation

This chapter presents the implementation that has been used. Initially, there is a description of the implementation of the SMS Service. Following the implementation for the export bindings are described.

4.1 Export binding implementation

This sub-chapter describes the implementation of the different export bindings.

4.1.1 SCA export binding

Since the SCA export binding can only work within IBM Integration Designer, the implementation made was for communication between two SCA modules. The BPEL process component of the SCA module that is exporting takes a string:name as input and replies with “Hello <name>!”. See figure 10 below for a visualization of the implementation in IBM Integration Designer.

![SCA export binding implementation diagram]

Figure 10 – visualization of the SCA export binding implementation
4.1.2 HTTP export binding

The HTTP export binding allows the components to have an external communication, unlike the SCA export binding. This implementation has a Java component that takes a string:name as input and replies with “Hello <name>”. The java component has a HTTP export binding connected to it which the application Advanced Rest Client sends its requests to. See figure 11 below for a visualization of the implementation in IBM Integration Designer.

In the properties for the HTTP export binding, it was required to navigate to Binding→Advanced Configuration → HTTP Method Settings and add the HTTP Method Type POST.

![Figure 11 – visualization of the HTTP export binding implementation](image)
4.1.3 SOAP over HTTP export binding

The SOAP over HTTP export binding allows the components to communicate externally via SOAP over HTTP. Just as the HTTP implementation, this implementation has a Java component that takes a string:name as input and replies with “Hello <name>”. The java component has a SOAP over HTTP export binding connected to it which the application SOAP UI sends its requests to. See figure 12 below for a visualization of the implementation in IBM Integration Designer.

![Figure 12 – visualization of the SOAP over HTTP export binding implementation](image-url)
4.1.4 Enterprise JavaBeans export binding

The Enterprise JavaBeans export binding allows components to communicate with Java EE business logic. The implementation has a BPEL process that takes a string:name as input and replies with “Hello <name>!” . The BPEL process has an EJB export binding connected to it which a web application sends its requests to. See figure 13 for a visualization of the implementation in IBM Integration Designer.

![Diagram](image)

**Figure 13 – visualization of the EJB export binding implementation**
4.1.5 JMS export binding

The JMS export binding allows components to communicate via JMS. The implementation has a BPEL process that takes a string:name as input and replies with “Hello <name>”. The BPEL process has a JMS export binding connected to it which a web application sends its requests to. To make the JMS export functional, some configurations had to be made in the Websphere Admin Console for the process server that was running it. Configurations also had to be made to the JMS export to match the Java Naming and Directory Interface (JNDI) with the JNDI for the console.

The configurations needed to be was to add Destinations, JMS queues, a JMS queue connection factory and a JMS activation specification. See figure 14 below for a visualization of these configurations and figure 15 for a visualization of the implementation in IBM Integration Designer.

<table>
<thead>
<tr>
<th>Resource type</th>
<th>Resource name</th>
<th>JNDI name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>JMSSample</td>
<td>jms/JMSSample</td>
</tr>
<tr>
<td>Destination</td>
<td>JMSSampleReply</td>
<td>jms/JMSSampleReply</td>
</tr>
<tr>
<td>Destination</td>
<td>JMSSampleCorrelation</td>
<td>jms/JMSSampleCorrelation</td>
</tr>
<tr>
<td>JMS queue</td>
<td>JMSSample</td>
<td>jms/JMSSample</td>
</tr>
<tr>
<td>JMS queue</td>
<td>JMSSampleReply</td>
<td>jms/JMSSampleReply</td>
</tr>
<tr>
<td>JMS queue</td>
<td>JMSSampleCorrelation</td>
<td>jms/JMSSampleCorrelation</td>
</tr>
<tr>
<td>JMS queue connection factory</td>
<td>JMSSampleReplyQCF</td>
<td>jms/JMSSampleReplyQCF</td>
</tr>
<tr>
<td>JMS activation specification</td>
<td>JMSSampleAS</td>
<td>jms/JMSSampleAS</td>
</tr>
</tbody>
</table>

Figure 14 – the resources added in the Websphere Admin Console.

Figure 15 – visualization of the JMS export binding implementation
4.2 SMS Service implementation

This sub-chapter describes the implementation of the SMS service.

This implementation uses an SCA module in IBM Integration Designer as its core component. By accessing the module externally with an HTTP POST containing a string:message and an integer:employee_ID, the module sends a query with the employee ID to a database, which in return replies with a phone number. With this phone number and the string:message, the SCA module uses the texting API, Twilio, that sends the text message to the phone number. See figure 16 below for a flowchart of this process.

---

Figure 16 – SMS Service implementation
4.2.1 MariaDB implementation

The database is a simple SQL database which is containing one table, employee. This table has three columns: FULLNAME, EMPLOYEE_ID and PHONE_NUMBER. See figure 17 below for a visualization of the content in the database. To be able to access the database in IBM Integration Designer, the mysql-java connector jar had to be added to the build path for the SCA module. See figure 18 for the code in IBM Integration Designer that is using the database.

Figure 17- Query showing the content of the database.

```
// tries to connect and query the Mysql database.
try {
    Class.forName("com.mysql.jdbc.Driver");
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/smsdb", "root", "*");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("select * from employee where employee_id =" + employee_id); // query

    if (rs.next()) {
        System.out.println(rs.getString(1) + " + " + rs.getString(2) + " + " + rs.getString(3));
        recipientPhoneNr = rs.getString(3); // Phone number acquired from the database
    } else {
        System.out.println("No matching phone number");
    }
} catch (ClassNotFoundException e) {
    System.err.println("ClassNotFoundException caught: " + e);
} catch (SQLException e) {
    System.err.println("SQLException caught: " + e);
}
```

Figure 18 – Code for sending a query to the database.
4.2.2 Twilio implementation

To be able to use the Twilio API, it was required to add the Twilio jar file in to the build path of the SCA module in IBM Integration Designer. Maven was used to to create the jar file. See figure 19 below for the code in IBM Integration Designer that is using the Twilio API. Two other SMS APIs was also tested, Plivo and BulkSMS. These APIs were tested due to issues covered in result chapter 5.3. Plivo was tested but quickly discarded due to the free trial account running of out credits almost instantly. To try an another free alternative, BulkSMS was tested. Unfortunately similar issues that occurred with Twilio occurred with this API as well, so the final implementation was using Twilio anyway.

```java
final string ACCOUNT_SID = "AC38af021c99a466e767ff058ec2f6b763a"; //User id for twilio
final string AUTH_TOKEN = "293a65d21c86d3b4c81f336b2276bc"; //Authentication token - https://www.twilio.com/console
final string senderPhoneNumber = "+46769448907"; //Twilio phone number.

twilio.Init(ACCOUNT_SID, AUTH_TOKEN); //Connect to Twilio account

//Create and send a message to a phone number using Twilio
if(recipientPhoneNumber != null){
    phoneNumber to = new PhoneNumber(recipientPhoneNumber);
    phoneNumber from = new PhoneNumber(senderPhoneNumber);
    string messageToSend = "hello from testjavaproject - Main";
    Message messageObject = Message.creator(to, from, messageToSend).create(); //Sends a message
}
```

Figure 19 – Code for sending a text message using Twilio API.
5 Results

This chapter presents the results that have been achieved during the project. First off the result of implementing the export bindings is described, followed by the result for the comparison of the export bindings, and finally the result for the SMS service.

5.1 Proof-of-concept export bindings

This sub-chapter describes the result of the creating the five different export bindings.

5.1.1 SCA export binding

As mentioned in the implementation chapter 4.2.1, the communication for SCA was between two different SCA modules. The communication between the two modules was successful and an invocation with the parameter string “Jonathan” got the successful response of “Hello Jonathan!”. This was done by accessing the BPEL process called SimpleProcess in the other module via communication to the SCA export. See figure 20 below which visualizes the result.

![Integration Test Client: SCAClientModule_Test](image)

Figure 20 – result of SCA export binding invocation
5.1.2 HTTP export binding

Using the application Advanced Rest Client, a HTTP POST could be sent as a request to the HTTP export binding in IBM Integration Designer and receive a successful response. The request contained a string “Jonathan” and the response carried the result-string “Hello Jonathan”. See figure 21 below for the HTTP request and figure 22 for the HTTP response.

Figure 21 – HTTP request

Figure 22 – HTTP response
5.1.3 SOAP over HTTP export binding

Using the application SoapUI, a SOAP request could be sent to the SOAP over HTTP export binding in IBM Integration Designer and receive a successful response. The request contained a string “Jonathan” and the response carried the result-string “Hello Jonathan”. See figure 23 below for the request and response.

Figure 23 – SOAP over HTTP request and response
5.1.4 Enterprise JavaBeans export binding

By using the web application created, a request containing the string “Jonathan” could be sent to the EJB export in IBM Integration Designer. This export was connected to the BPEL process. This request received a successful response. The response carried the result-string “Hello Jonathan!” See figure 24 below for the request and figure 25 for the response.

Figure 24 – EJB request

Figure 25 – EJB response
5.1.5 JMS export binding

By using the web application created, a request containing the string “Jonathan” could be sent to the JMS export in IBM Integration Designer. This export was connected to the BPEL process. This request received a successful response. The response carried the result-string “Hello Jonathan!”. See figure 26 below for the request and figure 27 for the response.

Figure 26 – JMS request

Figure 27 – JMS response
5.2 Comparison export bindings

This sub-chapter describes the result of the comparison of the five different export bindings.

5.2.1 SCA export binding

The SCA export binding [24][25] has no options for external communication. It can be useful when all the services are within the same server in IBM Integration Designer. It can be used to easily link separated modules. It is the simplest and fastest option of the bindings. There are no options for enabling security for the SCA export binding.

5.2.2 HTTP export binding

The HTTP export binding [24][26] is useful when the services are communicating over the internet with external services that use HTTP Methods such as GET, POST, or DELETE. It uses synchronous communication. This export binding has no options for enabling security. To get this export binding to work with HTTP methods beyond GET, it was required to configure it and add those methods in the HTTP method settings.

5.2.3 SOAP over HTTP export binding

The SOAP over HTTP export binding [24][27] is useful when the services are communicating over the internet with external services that use SOAP. It uses synchronous communication.

For the SOAP over HTTP export binding, there is an option to choose a default policy set. The policy sets contain a set of defined security options for the export binding. The service that sends a request to this export binding should have the same policy. See figure 28 below for a visualization of the different policy sets available for the SOAP over HTTP export binding.

![Figure 28 – Policy sets for the SOAP over HTTP export binding.](image)
5.2.4 Enterprise JavaBeans export binding

The Enterprise JavaBeans export binding [24][28] allows Java EE applications to externally invoke the components in IBM Integration Designer. The export itself is a stateless bean. The EJB export uses synchronous communication. See figure 29 below for a visualization of the functionality behind the EJB Export. When using an Enterprise JavaBeans export binding, there is no options for enabling security.

Figure 29 – Java EE application making a request to a EJB Export.
5.2.5 JMS export binding

The JMS Export binding [24][29] allows external JMS applications to externally invoke the components in IBM Integration Designer. The JMS export binding is useful when access to messaging systems is needed. It uses asynchronous communication. To get the JMS export binding to work in IBM Integration Designer, there has to be configurations made in both the export itself, as well as in the Websphere Admin Console. See figure 30 below for a visualization of the functionality behind the JMS export.

When using a JMS export binding, it is possible to specify a Java Authentication and Authorization Services (JAAS) alias. This alias, which by default was “DeAdminAlias”, can be used for the Java 2 connector (J2C) security.

Figure 30 – External application making a request to a JMS Export that is connected to a Service Integration Bus.
5.2.6 **Comparison export bindings summary**

See figure 31 below for a summary of certain properties. Furthermore, despite the limitations of the SCA export binding, it is the simplest and fastest export binding and is useful for communicating between SCA modules in IBM Integration Designer. The HTTP binding is most useful when the communications are using HTTP methods. The SOAP export binding is most useful when communicating with SOAP and it has certain security policy sets as options for enabling security. The EJB export binding is most useful when communicating with Java 2 EE business logic. The JMS export binding is the only export binding that has support for asynchronous communication, and while it is required to configure it to get it working, it has support for security in the form of Java Authentication and Authorization (JAAS) alias.

![Comparison of the export bindings](image)

**Figure 31 – Comparison of the export bindings**
5.3 Proof-of-concept SMS service

As mentioned in the implementation chapter 4.2, this service should begin with a HTTP POST request to the SCA module in IBM Integration Designer. This module would then query a database and finally use the Twilio API to send an SMS. This full service was unable to be completely functional due to an exception caused when running the service on the process server in IBM Integration Designer. The root cause was a component, org.apache.httpcomponents in the process server that was deprecated, and this component caused the same component in the Twilio JAR to be overwritten. This resulted in the service being unable to run and failed with the following exception:

```
“java.lang.NoSuchFieldError:
org/apache/http/conn/ssl/AllowAllHostnameVerifier.INSTANCE”
```

However, if the service was running without the HTTP POST and not on the process server, the service could run successfully. The service sent a query to the database and got a phone number as a response which was used with the Twilio API to send a text message. See figure 32 below for the console in IBM Integration Designer when running the service and figure 33 on the next page for the SMS being received by the Twilio API successfully.

![Figure 32 – Console in IBM Integration Designer when running SMS service.](image-url)
Figure 33 – SMS received by the Twilio API.
6 Conclusions

This chapter covers the conclusions that evaluate the results of the project.

After this project has been completed the result is that the majority of the goals have been fulfilled. If you recall the concrete and verifiable goals from the introduction chapter 1.4, the first goal was to create an SCA module in IBM Integration Designer and expose it with the five different export bindings. This has been done successfully as you can see in the result chapter 5.1.

The second goal was to make a comparison of the differences between these export bindings regarding the different usage areas and the options for enabling security provided by IBM Integration Designer. This has also been done successfully as seen in the result chapter 5.2. Most of the export bindings were very similar, with JMS being the one that stood out the most since it had so many configurations needed to work properly. Interestingly enough, as seen in figure 31 in the result chapter 5.2.6, it was only the JMS and HTTP over SOAP export bindings that had options to enable security for the export itself.

For the third goal, to create the SMS service, this has been partially fulfilled. The issues with the compatibility with the process server in IBM Integration Designer and external JAR-files has been a great issue during this project. Adding external dependencies to this process server is a big weakness since the dependencies are rarely compatible with it. A great deal of other SMS API’s has been tried out to get this to work. This was done without success. Examples of these API’s are Plivo and BulkSMS. To get this service to work correctly, one would have to find out how to make the process server prioritize a package over another. The final version as seen in the result chapter 5.3 works successfully with the database and the Twilio API, without the integration with the SCA module in IBM Integration Designer.

The development during this project has been done in three separate sprints which greatly assisted in keeping the right prioritization. The sprints helped structuring the project into smaller and smaller sub-goals. A Kanban board in the form of Trello was used to keep a good overview of all the tasks that was left to be done. This made it possible to quickly get into the work when resuming the work-session. If you recall the study “A survey study of critical success factors in agile software projects” mentioned in the methodology chapter 3.4, there were three critical aspects for success. The success factor “Correct delivery strategy” had the attributes to have a regular delivery of software, and to deliver the most important features first. Another success factor was “Agile software engineering techniques”. Both of these success factors have been achieved using the sprints and kanban board. For the success factor “Agile software engineering techniques”, one attribute was to pursue simple design. This this has
been achieved in many ways, such as simplifying the database and the code used for the export bindings as much as possible.

Regarding ethical aspects for this project, BPM and IBM Integration Designer is mostly used for government agencies which manage a great deal of sensitive data. Because of this, it is important to consider the security options when working with IBM Integration Designer and whilst the export bindings themselves are on the low side of security, there are a lot of options for the server that is available by accessing the Websphere Admin Console.

The learning curve for IBM Integration Designer was quite steep. The installation package was not working properly so a partial manual installation was required. The errors given did not give an accurate description the cause of the issue at a satisfactory level, and a basic knowledge of the program often came short in solving issues. When working with BPM, it is worth to consider if using IBM Integration Designer is even worth the effort. This project has been challenging, educational and given a great deal of insight into BPM and especially IBM Integration Designer.
References

[1] Sogeti, “Who are we?”,
https://www.sogeti.se/om-oss/vilka-ar-vi/
Retrieved: 2017-06-16

Retrieved: 2017-06-19

https://www-01.ibm.com/software/info/bpm/what-is-bpm/
Retrieved: 2017-06-21

https://www.capgemini.com/blog/capgemini-oracle-blog/2011/06/bpm-or-bpel-that-is-the-question
Retrieved: 2017-06-21

[5] IBM Knowledge Center, “Getting started with IBM Integration De-
signer”,
Retrieved: 2017-06-21

[6] IBM Knowledge Center, “Video transcript: Getting started with IBM In-
tegration Designer”,
Retrieved: 2017-06-21

Retrieved: 2017-06-21

Retrieved: 2017-06-21

[9] IBM Knowledge Center, “Imports and exports”,
Retrieved: 2017-06-22
[10] IBM Knowledge Center, “Service components”,
Retrieved: 2017-06-21

Retrieved: 2017-06-21

http://www.techrepublic.com/article/an-introduction-to-the-simple-object-access-protocol-soap/
Retrieved: 2017-06-22

http://docs.oracle.com/javaee/6/tutorial/doc/bncdr.html
Retrieved: 2017-06-22

[14] IBM Knowledge Center, “Enterprise JavaBeans (EJB) 3.0 specification”,
Retrieved: 2017-06-22

https://maven.apache.org/what-is-maven.html
Retrieved: 2017-06-22

[16] Twilio, “Programmable SMS”,
https://www.twilio.com/
Retrieved: 2017-06-23

[17] Plivo, “SMS API”,
https://www.plivo.com/sms-api/
Retrieved: 2017-08-24

[18] BulkSMS, “BulkSMS text messenger”,
Retrieved: 2017-08-24

[19] MariaDB, “About MariaDB”,
https://mariadb.org/about/
Retrieved: 2017-06-23

http://searchcio.techtarget.com/definition/proof-of-concept-POC
Retrieved: 2017-06-23
https://chrome.google.com/webstore/detail/advanced-rest-client/hgm-loofdfffnphfgcellkdfbfbjeloo
Retrieved: 2017-06-23

[22] SoapUI, “What is SoapUI”,
https://www.soapui.org/open-source.html
Retrieved: 2017-06-23


[24] IBM Knowledge Center, “Selecting appropriate bindings”,
Retrieved: 2017-06-23

Retrieved: 2017-06-23

[26] IBM Knowledge Center, “HTTP bindings”,
Retrieved: 2017-06-23

[27] IBM Knowledge Center, “Web service bindings”,
Retrieved: 2017-06-23

[28] IBM Knowledge Center, “EJB bindings”,
Retrieved: 2017-06-23

[29] IBM Knowledge Center, “JMS bindings”,
Retrieved: 2017-06-23