Portable BizTalk solutions

DANIEL ERIKSSON

Degree project in
Computer Science
Second cycle
Stockholm, Sweden 2013
Portable BizTalk solutions

Evaluating portable solutions to search for errors in BizTalk-platforms

Master Degree Project in Computer Science, DD221X, Spring 2013

Supervisor: Lars Arvestad
Examinator: Olle Bälter
Performed at: Integrationsbolaget

Daniel Eriksson
Åbackegatan 5A
702 32 Örebro
0766279279
<daniele2@kth.se>
Portable BizTalk solutions

Abstract
This report evaluates possible infrastructures to create portable BizTalk solutions. BizTalk is an integration based software mostly used at larger companies. Errors can occur in BizTalk and experts need an easy and portable solution to identify these. No such solution exists today, and this report focuses on how it could be performed. The results show that various tools need to be used to access information from BizTalk. Information about BizTalk must be protected by access rights, which are preferably controlled from a cloud portal. The cloud portal used in this project is Windows Azure, but other solutions have been considered. Azure has a specialized service to access secure locations, which other provider’s lack. Finally, a prototype application in Windows Phone 8 was developed. The solution has been shown to BizTalk experts, who were enthusiastic by the proposed solution and has proceeded with the project. They are currently analyzing what it would cost to develop a product and what could be charged for such a service.

Keywords: BizTalk, Mobile, Cloud computing, Windows Azure Windows Phone

Mobila lösningar för övervakning av BizTalk

Sammanfattning

Keywords: BizTalk, Molntjänster, Datormoln, Windows Azure Windows Phone
# Table of Contents

1 Introduction .................................................................................................................. 1
   1.1 Problem statement .................................................................................................. 1
   1.2 Purpose ................................................................................................................... 2
   1.3 Related work .......................................................................................................... 2
      1.3.1 Industry outlook ............................................................................................ 2
      1.3.2 Mobile cloud computing .............................................................................. 3
      1.3.3 Results of literature study ............................................................................. 3
2 Method ............................................................................................................................ 5
   2.1 Literature study ...................................................................................................... 5
   2.2 Part A - Access BizTalk ...................................................................................... 5
   2.3 Part B - Global communication .......................................................................... 5
   2.4 Part C - Graphical presentation ........................................................................... 6
3 Part A – Access BizTalk ............................................................................................... 7
   3.1 Orchestrations ...................................................................................................... 8
   3.2 Business Activity Monitoring .......................................................................... 8
   3.3 BizTalk Databases ........................................................................................... 9
   3.4 ESB toolkit ........................................................................................................ 10
   3.5 Windows Management Instrumentation (WMI) ........................................... 11
   3.6 ExplorerOM library ......................................................................................... 12
   3.7 Summary ........................................................................................................... 12
4 Part B - Global communication .................................................................................... 13
   4.1 Cloud Services ................................................................................................... 14
   4.2 Other cloud service providers ...................................................................... 16
   4.3 Evaluation methods .......................................................................................... 16
      4.3.1 Kahn’s criteria ............................................................................................. 17
      4.3.2 CIA Triangle .............................................................................................. 18
5 Part C - Graphical presentation .................................................................................... 19
   5.2 Survey ................................................................................................................ 20
   5.3 Heuristic evaluation ......................................................................................... 21
   5.4 Prototype application ......................................................................................... 22
6 Evaluation .................................................................................................................... 25
   6.1 Proposed solution ............................................................................................... 25
   6.2 Evaluation of Kahn’s criteria ........................................................................... 26
      6.2.1 Basic theory ................................................................................................. 26
      6.2.2 Data protection ............................................................................................ 26
      6.2.3 Data integrity .............................................................................................. 27
      6.2.4 Scalability .................................................................................................. 27
      6.2.5 Assumption ................................................................................................. 28
      6.2.6 Data access ............................................................................................... 28
6.2.7 Authentication

6.3 Evaluation of CIA-triangle

6.3.1 Confidentiality and Integrity

6.3.3 Availability

6.4 Evaluation of user interface

6.4.1 Visibility of system status

6.4.2 Match between system and the real world

6.4.3 User control and freedom

6.4.4 Consistency and standards

6.4.5 Error prevention

6.4.6 Recognition rather than recall

6.4.7 Flexibility and efficiency of use

6.4.8 Aesthetic and minimalist design

6.4.9 Help users recognize, diagnose, and recover from errors

6.4.10 Help and documentation

7 Discussion

7.1 General

7.2 Android and iOS

7.3 Security

7.2 Future work

7.3 Response from BizTalk experts

8 Conclusion

Bibliography

Appendix A – Survey
1 Introduction

BizTalk Server is an application developed by Microsoft to enhance the possibility of integrating dissimilar systems. With the help of different adapters, BizTalk can connect services which doesn’t understand each other and thus unable to communicate properly. By installing and configuring a BizTalk Server as a man in the middle (an interpreter), the services can exchange information with each other. BizTalk acts as an interpreter with business logic, which means that it can also process and alter messages which are interpreted.

As with most technical solutions, problems will occur, and companies need a fast and easy way of identifying these. People that work with integrations such as BizTalk are mobile and travel frequently, which makes a portable and fast monitoring solution desirable. By being notified about potential errors and identify these, they aim to achieve higher quality towards their customers. The project consists of evaluating how to construct a portable solution for BizTalk experts. Since no portable solution exists today, this requires an approach which has not been evaluated yet.

The project demands a portable solution independent of location and this type of solution has not been presented together with BizTalk earlier. This solution has been divided into three parts and these constitute the major checkpoints of the whole project. Part A discusses tools that can be used to access BizTalk and suitable for portable monitoring, Part B contains information about how the communication to the portable device should be handled and Part C involves the graphical user presentation on the device. Most of this report will focus on Microsoft-related products such as BizTalk, Windows Phone and Azure. Windows Azure is a cloud computing platform developed by Microsoft and constitutes a large part of this project, especially in Part B. The final solution on the other hand, should not be limited to Microsoft products, which is important to clarify.

1.1 Problem statement

The first step is to evaluate the requirements of a portable and flexible solution to identify errors in BizTalk systems. Because it needs to be portable, it requires involvement of either mobile, media tablet, or similar. The graphical presentation could then be solved with a native app or a webpage, which both have their strengths and weaknesses. In this case, a web solution would result in a couple of undesired problems, such as GUI performance, no notifications, and so on [1]. Because of this, a web solution will not be considered and is only mentioned shortly in the project.

For Android and iOS there exists nothing similar. Windows phone has an app which can connect to an older version of BizTalk (Server 2010) [2], but lack all of the substantial elements. It is a local solution which cannot be used on the field, the functions cannot be used to identify errors and the sensitive information that BizTalk exposes lacks security. In other words, this solution cannot be used as portable monitoring.

More specifically; the intention is to investigate how it is possible to create a portable solution to be able to monitor BizTalk Server 2013. This solution needs to work globally, outside the local network, and cannot compromise security. The last part of the project will consist of developing a prototype app for Windows Phone 8, in order to demonstrate the solution.
1.2 Purpose
The purpose of this report is to evaluate how to construct an infrastructure for mobile monitoring of the BizTalk environment. When such an infrastructure exists, companies that use BizTalk can achieve a higher quality of their solutions. It could also be presented as a unique selling point, because no company uses this type of monitoring yet. The purpose is not to demonstrate a working infrastructure that only can be accessed from Microsoft devices. Even though the prototype will consist of a Microsoft product, it must be clear that the infrastructure is not limited to this. The aim of the project is to create an infrastructure capable of globally communicate with all types of devices.

1.3 Related work
BizTalk is a fairly new product, with a first release in 2000. Since then, few research articles have consisted of discussing BizTalk and no article addresses the overall issue of this project. This could be because BizTalk targets few people, and mainly experts. Just installing BizTalk is difficult and an entire chapter has been dedicated to this process in Daniel Woolstons book about BizTalk [23]. Belle Selene Xia states that few cases have been devoted to the subject of efficient information systems such as BizTalk, partially due to lack of data [4]. Even though no article discusses the overall problem, most parts of the project have been discussed earlier.

1.3.1 Industry outlook
The future relevance of this project heavily depends on the industry outlook for BizTalk server. Xia discusses the future of BizTalk server in a case study made on BizTalk [4]. According to Xia, the need to use integration based software such as BizTalk does not seem to be decreasing. Xia’s reasoning seems compelling, but misses one concern regarding system integration, which is cloud computing. By moving systems to the cloud, the need for integration decreases since integrations could be handled automatically by the cloud. When building solutions locally, the need to integrate them are larger, though the possibility of communicating between each other are not standardized in any way. One of the main ideas of cloud computing is to standardize messy and complex IT solutions, and move them to a standardized cloud. The cloud contains advanced services which are more standardized and manages most of the integration within the cloud automatically. Cloud services are designed to communicate with each other without the need for additional software, such as BizTalk. By using local solutions, every company tend to reinvent the wheel each time. An example is Microsoft Azure Notification Hub, which is a cloud service that handles notifications systems. If a service needs to send notifications to iOS, Android, Windows Phone and Windows Store, the service needs to integrate this in four different and complex ways. The Notification Hub cloud service provides an easy way of integrating all these systems at once. When integrating all these systems by a standardized cloud service, the need for integration based software decreases.

Many larger companies are investing lots of money in joining the new trend towards cloud computing, and thus, demand on integration systems could possibly be decreasing. However, cloud computing cannot fully outsource the need for system integration, as integration will always be a part of software technology. This trend towards cloud computing could as well be a hype. According to one of the world’s most respectable research companies Gartner, cloud computing is currently in the state of a hype [24]. Gartner indicates that future expectations of cloud services are too high and that companies falsely believe in a future where cloud computing solves everything, including integration. Also, newer versions with BizTalk have evolved to be able to connect to the cloud more easily, which indicate that the development of BizTalk is aiming to adapt to the future. The industry outlook for BizTalk seems positive, and the industry needs an easy way of detecting errors on the field.
Even though cloud services could potentially outsource the need for integration in the future, this project heavily depend on these clouds, paradoxically. Many articles have been written about cloud services in combination with mobiles and security, and these are mentioned in Part B.

1.3.2 Mobile cloud computing
Cloud computing is not a product, but a service which provides shared resources, information and software to computers and devices like tablets and phones[32]. Resources are not available directly to the user, but are offered as a service in the cloud. As the smartphone market and mobile bandwidth continues to increase [14], mobile cloud computing is increasing in popularity. With increasing mobile bandwidth, smartphones can take more advantage of the cloud.

Abul Nasir Kahn discusses mobile cloud computing and security aspects in his paper about secure mobile cloud computing [6]. The paper by Kahn present evaluation criteria for data and application security. Kahn is mainly focusing on a cloud service which share mobile data, instead of a messaging service like this project. However, most of Kahns criteria are of importance for all type of mobile cloud computing. Kahn mentions seven security frameworks, and by applying these security criteria to portable monitoring by BizTalk experts, it is clear that security must be evaluated thoroughly. Information processed by BizTalk experts could be highly confidential, and must be protected. An unsecure solution would not be relevant for developers using BizTalk. The criteria presented by Kahn will be mentioned in section 5 and evaluated in section 6.

Sandeep K. Sood discusses the CIA security triangle in combination with cloud computing [7], and suggests a possible approach. Sood mention a more general approach than Kahn, and does not relate this to mobile cloud computing. Confidentiality, integrity and availability are the three building blocks of this triangle which is evaluated in section 6.3. Confidentiality refers to keeping the information secret, in the case of the project this is when secret error information is exposed from the on-premise service to the cloud and mobile. On-premise refers to the local network where the BizTalk server is installed. Information from this network must be kept secret. Integrity refers to ensure data integrity over the network, meaning the data should not be modified during transportation or in the cloud. Availability is the last part of the triad, and aims to ensure availability of the system when needed. Confidentiality and integrity are covered by the seven components in Kahn’s security model, but not availability. Even though Kahn’s model is considered more accurate for mobile cloud computing, a short evaluation of the CIA triangle will be performed in the evaluation section.

1.3.3 Results of literature study
The literature study was performed to understand the problems BizTalk experts face when searching for errors and thus help in designing a solution to simply the searching process. Problem statement and method have been constructed from the literature study performed in the beginning of the project.

During the literature study, it became clear that the solution needed to be classified using different criteria. Several frameworks consisting of important criteria were evaluated, and Kahn’s framework [6] was chosen for this project. Kahn presents a framework much similar to this project, were mobile applications needs secure connection through cloud services.

Part A, which analyses how to connect to BizTalk, requires preliminary requirements in order to be performed properly. The most important information to present on a portable device needs to
be confirmed before starting with Part A. Otherwise it would be unclear which information should be extracted from BizTalk. The preliminary requirements set during the literature study consist of:

- Grouping suspended (error) messages by application.
- Status information of host instances.
- Status information of BizTalk applications.
- A basic overview of the BizTalk platform in connection with the mobile.
- A mobile application which can access remote BizTalk servers and present relevant data.

These requirements are further evaluated in Part C, but serves as a starting point for the evaluation performed in Part A.

Results concluded from the literature analysis in Part C have resulted in a survey, heuristic evaluation and MVVM software pattern. A survey of 15-20 questions directed to BizTalk experts on different companies was chosen. The survey must be performed by independent companies in order to receive an unbiased review. By reading a paper of Wei-siong [34], heuristic evaluation was chosen as evaluation method. Part C describes in depth why heuristic evaluation was chosen and not user testing. During the literature study, software pattern for Part C was chosen to MVVM which is a modern pattern to minimize GUI coding.
2 Method

The project demands a portable solution independent of network location and this type of solution have not been presented together with BizTalk before. It can be divided into three parts and this constitutes the major checkpoints of the whole project. Part A discusses how to connect and retrieve information from BizTalk, Part B contains information about how the communication should be handled and Part C involves the graphical presentation. Before starting with each of the parts, a literature study was made.

2.1 Literature study

First of all, a literature study was made. By reading books and papers relevant to each part, the problem was easier to understand and a possible approach could be presented. The method and problem statement is largely based on the literature study performed at the beginning of the project. Within the literature study, preliminary requirements were set in order to evaluate the first parts more easily.

The preliminary requirements consisted of a mobile application which could access remote BizTalk servers and present relevant data. Relevant data consisted of grouping error/suspended message, host/application status and a basic overview. These requirements are further evaluated in Part C, but serves as a starting point for the project.

2.2 Part A - Access BizTalk

Part A is the most crucial part of the project and without it, nothing would work. If the project is not possible to carry through, it will be obvious in this part, which makes it important to analyze this part first. There are a couple of tools which could be used to access BizTalk and retrieve information. Some are easier and others present more flexible solutions, but all of these tools have to be evaluated against the preliminary requirements of Part C. Because of this, it is important to have performed all the literature studies before beginning with this part, otherwise false assumptions could be drawn. The literature study contains a preliminary requirement of Part C.

The technical challenge in this part is to analyze different tools to access relevant data from BizTalk and choose the tool which provides the best methods for mobile monitoring. Worst case scenario is that no tool will be good enough, and in that case, a connection directly to BizTalk’s databases is needed. These databases contain hundreds of tables and this type of evaluation would be time consuming.

2.3 Part B - Global communication

When Part A is finished it is clear to say whether the project can be completed. The next challenge is to expose the information from Part A to a mobile unit, e.g., media tablet, mobile or similar. Important to realize is that an administrator needs to access a client’s BizTalk Server. This is also what makes this part so unique, almost all mobile solutions have the opposite situation, where a few servers are configured to communicate with multiple clients. Part B requires a solution where a few administrators are in need of accessing secure locations on client computers with BizTalk installations. Problems that occur are that firewalls and address translation boxes (NATs) cannot be configured or changed to meet specifications of connectivity. An acceptable solution does not involve changing a client computer’s security
standard. Hopefully, this could be solved with some sort of encrypted cloud service. This contributes to the engineering challenge in this part.

Another problem in this part is how mobiles safely and easily can connect according to the model previously mentioned. Where should the security and authentication be located, in the mobile? In the cloud? On-premise server? Or a combination of these? The technical challenge in this part is that the authentication must occur according to the reversed model previously described. Part B is expected to be the most challenging and time consuming part in this project.

2.4 Part C - Graphical presentation

Part C requires an analysis of what to present to the user. If this analysis is not performed, it is possible that unimportant data is shown to the user, which results in a less interesting project. In this part it is possible to investigate graphical user interface, notifications, and other user related enhancements. In order to determine how and what to present, a survey will be performed. The purpose of the survey is to find out which parts of BizTalk are important to analyze, and could be referred to during the graphical evaluation. Part C is not as important as the other parts, but should not be underestimated. If underestimated, it could result in something totally unusable.

Previous work within the graphical part of this project consists of user testing and heuristic evaluation. Wei-siong Tan compares these methods to achieve high usability, and states that neither method can be replace by the other [34]. Both methods are used in software technology to identify usability problems in applications.

Wei-siong also states which method should be used when. Some methods are more likely to identify usability problems at an early stage of the project, and others could be done later. When choosing heuristic methods, Wei-siong suggests Nielsen and Mack [27], which has become a standard according to Wei. These types of evaluations will be presented in section 5 and evaluated in the subsequent section.
Part A – Access BizTalk

BizTalk Server presents a couple of ways to access information about BizTalk platforms and monitor them. Some tools can be added to a BizTalk Server in order to retrieve information from it and others are installed during the initial installation. None of these tools have been primarily designed for portable monitoring, but most of them can be used as such. There also exist different frameworks that can be used to access BizTalk directly from code. This part analyses these tools and frameworks with focus on mobile monitoring of errors. When focusing on mobile monitoring in this part, the idea is not to send this information directly to a mobile, but retrieve relevant information that is useful on a mobile and send it to the cloud, described in Part B. The application responsible for accessing BizTalk and expose it to the cloud will be called WCF service. Windows Communication Foundation (WCF) is a runtime to build server oriented application in the .NET framework. Throughout this project, WCF can be understood as a back-end application written in C#.

Today, this type of error searching is performed mainly through BizTalk Administration Console (see 5.2). The Administration Console helps BizTalk experts with server management of the platform responsible for the integration. Errors and all types of information can be shown here, including custom queries. This project focuses on error searching, but the Administration Console has many more features than searching for errors. Xia discusses the importance of this tool and how it is used in their company, which also consists of configuration issues in the server [4]. The paper mentions suspended message briefly, which is the same as a trapped error and can in most cases be corrected and resumed. During the initial literature study, suspended messages were prioritized as the most important feature for portable monitoring. These messages should somehow be presented on a portable device, both information about specific errors and statistics. As an illustration, Figure 1 shows two suspended messages which have been caught in the MsgBox Database. In the figure, messages have entered BizTalk where it says “IN” and exited at “OUT”, but two of them have been caught in the “MsgBox” component.

![Figure 1: WCF Service using different approaches (in green) to access data from BizTalk (in red). Orchestrations represent most of the logic in BizTalk.](image)
Figure 1 represents the problem associated with Part A of the project. Information about BizTalk can be retrieved from different tools, frameworks and BizTalk components, and the aim of this section is to present the best possible solution for mobile monitoring of BizTalk platforms.

3.1 Orchestrations

An orchestration is a powerful tool in BizTalk server which processes data between systems. Meaning, orchestrations are responsible for most of the logic in a BizTalk server configuration. Messages that are guided between different systems using BizTalk can be modified with code using an Orchestration. It is an implementation of a business process which can be modeled in Visual Studio (Microsoft’s IDE). These orchestrations are a huge part of BizTalk and are a part of the requirements, which is to be able to monitor orchestrations from a mobile. But the flexibility also allows them to subscribe to errors and forward their information. This information will be sent to Part B in order to globally expose them to a portable device.

In BizTalk Server 2006 the ability to generate an error report message when a message is suspended was added [23]. The report message could be routed to an orchestration and then be sent to Part B through all the different adapters that BizTalk can handle. However, to be able to generate this type of report, the message needs to be flagged as a routable error which means that changes need to be performed to the application that needs monitoring. One BizTalk platform usually has many applications, and modifications to all of them are not a smart solution. A solution involving this type of manipulation is not stand alone and should be avoided, though it requires extra work to flag application ports as routable. Stand-alone refers to a solution which can be deployed without modification in the BizTalk platform.

Another problem is that the messages are in fact routed to the subscribing orchestration, meaning they are routed from the message box to the orchestration. This gives rise to yet another problem, the message is disappearing from the message box and thus unable to find in the Administration Console. BizTalk experts using the console and not the mobile application will not be aware of the error, because it has disappeared from the message box and routed to the orchestration in charge of mobile monitoring.

This approach has too many drawbacks to be part of the solution. But orchestrations have some unique features as well, which is the subscription mechanism. When subscribing to an error, it is directly forwarded to the orchestration which fires exactly when the error is produced. This could prove to be an asset if mobile notifications are interesting. If not using a publish/subscribe mechanism to detect errors, errors need to be polled in certain intervals, causing a delay of the notification. All in all, this advantage cannot repay the disadvantage of using a solution where currently working applications need to be modified. The whole idea of the project is to create a stand-alone solution.

3.2 Business Activity Monitoring

Business Activity Monitoring (BAM) extends BizTalk Server in a range of different ways. Similar to ESB toolkit (discussed in 3.4), it can be accessed through web services that expose information about BizTalk. However, this is not the typical use of BAM, which is mostly used to allow monitoring of the data itself and present to non-technical people, such as amount of data and type of data fitting a special criterion. A typical example is to put benchmarks against data passing through BizTalk and export this to an Excel file. The problem with BAM is that the data to be monitored must be explicitly declared and the web services contain no functions for simple monitoring of applications or errors. Analyzing of data has to be configured manually, and information about the server itself cannot be retrieved this way. Altogether BAM is a non-
flexible solution to send error information with mobile destinations, and all relevant the information cannot be retrieved.

BAM is large product with many features, and can be used for most statistical queries. For portable monitoring it cannot be used in any simple and acceptable way. Also, BAM is not installed on all machines and thus have to be installed and configured on each server that needs portable monitoring. Therefore, this is not a desirable solution.

### 3.3 BizTalk Databases

BizTalk installs several databases in the SQL server environment, the number of databases installed during the installation depends on the installation itself. BizTalk server may use as much as 10-20 databases depending on the server functionality; four of these databases consist of the fundamental operations required for BizTalk [3]. The MessageBox database and the Management database are two of these four databases, which consist of information about errors and server state. These databases are the only relevant ones in this project. Together, just the two databases contain approximately 220 tables, 20 views and 700 stored procedures. These numbers could be slightly different between different versions of BizTalk, but no significant change. BizTalk Server uses databases heavily, and almost all information about the state of the Server can be retrieved from here. In order to retrieve relevant information from this enormous data pool, a study of this data had to be made.

To match the preliminary requirements of user presentation, an investigation of the databases was started. It quickly became clear that BizTalk is a huge application, constructed by a larger group of people, which add even more complexity and redundancy. Initially, the structure of the database could not be understood, due to all of the different relations between the 200+ tables. An idea was to investigate this using the SQL Server profiler, which logs all calls to SQL server and thereby understand the database schema. Using this profiler, it is possible to monitor certain processes and their activity against the databases.

A use case from the preliminary requirements of the functionality was selected, namely grouping suspended messages by application. This query was performed in the Administration Console and then monitored using SQL Server profiler. The hypothesis was that the Administration Console could call a few easier stored procedures, which then could be used as a part of the solution, instead of deriving the whole complex database. A stored procedure is a subroutine stored in the database, which can be used to centralize most of the logic to the database. This means that the stored procedures could be called, and the true logic behind the tables could be ignored.

The experiment consisted of setting up the profiler to only monitor the process ID of BizTalk Administration Console and then updating suspended messages grouped by application. After completion of the profiler, the hypothesis turned out to be wrong. A total of 15 stored procedures were called, where some calls contained up to 1000 characters (only the procedure call). By looking into these stored procedures in the BizTalk database, it is estimated that the heaviest procedures consisted to approximately 600 lines of SQL code. All in all, this resulted in a couple of thousand lines SQL code to analyze, much of it with high redundancy. This is probably the downside of having many persons work on a single project, but since BizTalk is such a large project, this is probably the only alternative. All this code was not analyzed, though it would be an ugly and time consuming solution. The underlying data model of the database had to be understood to proceed with this solution. Another discovery was that some information could not be retrieved from the database. By monitoring using the profiler and changing status of host instances, no information was recorded, thus status of host instances cannot be located in the database.
After some time, when the underlying data model of the database finally was understood, lots of problems had been detected. The two different databases that were of importance had different naming conventions. The MessageBox deviates from the naming convention used in the Management database, Administration Console and everywhere else. Not only does the MessageBox have a hundred tables, it also has confusing names. It is as if the people working with the MessageBox had their own world when constructing it, but surely they must have a good explanation.

In the MessageBox, the applications table consists of the different host instances, which can also be found in the Management database, but named as host. The applications are in a table called modules in MessageBox, but in the Management database it’s called applications. All this redundancy and different naming conventions made it difficult to find and derive what was searched for. Information about elements in these tables was not found directly and had to be linked with a couple of tables first, also when there are so many tables with different names it easily becomes a mess. Status of BizTalk applications and Host Instances (part of the requirement) cannot be retrieved directly from the table.

BizTalk applications can enter four different application statuses, and these are determined by the underlying components of the application [31]. In order to retrieve status about BizTalk applications through this approach, all components (e.g. Orchestrations/Send Ports) associated with the application has to be iterated through and analyzed. Manual calculation and iteration through underlying components, is an unwise and error prone solution. Information about host instances cannot be retrieved at all, and is explained in section 3.5.

Using the database is not user friendly, but has not constructed to be so. Even though it’s possible to use the database manually to retrieve and add information, Microsoft warns users of manipulating the database. The warning mainly addresses changing database tables, though this could potentially crash a BizTalk server permanently. The only real advantage of using the BizTalk database is that it’s always accessible and almost all information is stored there.

### 3.4 ESB toolkit

ESB toolkit 2.2 extends the functionality of BizTalk and is fully integrated into BizTalk Server 2013, which makes it possible to install it directly from the installation media. Unlike many other tools that need to be installed separately, ESB toolkit provides an easy installation process. This is new in BizTalk Server 2013, other versions of BizTalk forces a more complicated installation. The toolkit extends BizTalk Server in a range of different ways; one of them is fault management and reporting. All of the other capabilities are not relevant in this project.

According to Microsoft, the toolkit focuses on extending BizTalk to provide new capabilities such as building robust, connected and service-oriented applications [5]. What this means is that the main focus of the toolkit is to extend functionality of BizTalk and not fault management. The toolkit exposes a range of web services to accomplish this where one of them is Operation Web Service, which is the only interesting service in this project. Operation Web Service exposes 17 functions which access BizTalk’s databases and can be used to retrieve information about the Server. These functions consist of retrieving application status, suspended message, hosts, system status and other information that could be relevant on a portable device.

ESB toolkit would be the easiest and most future safe alternative to safely access and retrieve information from BizTalk Server 2013. This is because Operation Web Service is a stand-alone web service which can be easily installed to a BizTalk platform and then accessed through other
services described in Part B. If Microsoft decides to change the layout of BizTalk with newer versions, this would likely affect Orchestrations and the Databases. Since the toolkit exposes a stand-alone web service, also the toolkit would be updated when the next version of BizTalk arrives. This would require minimal changes to adapt the solution to a newer version of BizTalk. It is also backward compatible with BizTalk Server 2006 and 2009, which uses ESB toolkit 1.0 and 2.1.

In order to use ESB toolkit as a part in this project, it must fit the requirements of the use case scenario performed in Part C. As described earlier, a preliminary scenario was performed already in the literature studies and this will be used to validate the possibility of using ESB toolkit in part of a portable mobile solution. One of the early user requirements consisted of grouping suspended message per application. Even though it is easy to acquire information about applications, hosts and suspended messages, these are unfortunately not linked together. Information about suspended messages are more relevant when it is possible to see which application caused the suspension, which is not possible through the newest version of ESB toolkit (version 2.2). Host status and application status cannot be retrieved either. By looking even further into the problem, only 17 predefined methods provide a very basic solution to a complex problem. The relevant databases have over 200 tables, and to summarize this using 17 predefined methods is simply too basic.

### 3.5 Windows Management Instrumentation (WMI)

Windows Management Instrumentation (WMI) provides management features through components which can access information and notifications. It can be used in a framework to provide information about BizTalk. By using the ManagementObject library in C#, WMI objects can access information contain from the BizTalk platform. Important objects that can be accessed through this framework are host instances, orchestrations and send/receive ports. Application information and grouped error messages cannot be retrieved through objects in this list. All of these objects and features are part of the requirement and need to be retrieved in another way.

Almost all information about BizTalk is stored in its databases, but not status of host instances. This fact was confirmed by monitoring SQL traffic to the database when starting a host instance. By starting a host instance, its status changes but SQL profile monitor recorded no database activity when performing this test. Host instances are windows services, and status information needs to be retrieved from a tool which access Windows services, WMI is such a tool. Status of host instances cannot be retrieved easily through any of the other tools presented, thus WMI is needed.

Another advantage of using WMI is the installation process. In most other solutions, a database connection, or installations of tools are needed in order to receive information. But in the case of WMI, no such thing is needed. WMI operates directly on components associated with the server is is running on, and only requires access rights to fully function.

WMI is the simplest and easiest solution for accessing information from BizTalk. It presents an easy installation process and contains information about most of the features interesting for portable monitoring of BizTalk. What it cannot be used for, is grouping error messages, retrieving status about applications and more advanced features.
3.6 ExplorerOM library

ExplorerOM is a library used in a similar way as WMI, but offers other functionalities. This library is specifically dedicated to retrieve information from BizTalk and has a list of classes which is used for this purpose. Among others, it contains classes with information about Applications, Orchestrations and Send/Receive Ports [30], which is important for portable monitoring. Other tools and frameworks cover all of these features except applications. Information about applications cannot be retrieved directly through the database and has to be manually calculated (section 3.3), but ExplorerOM is a wiser solution. As described in section 3.3, using the database to access status of applications is an unwise and error prone solution.

ExplorerOM does require a database connection, to enable the functions contained in the framework and thus is slightly more complex than WMI. ExplorerOM library is similar to WMI, and both offer similar functions. Because WMI presents a slightly simpler solution, WMI should be preferred when possible.

3.7 Summary

This section provides a summary of which tools could be used in a portable solution. BizTalk is a powerful product which present many ways of retrieving and manipulating data within a BizTalk platform. The tools presented all have their advantages and this has been evaluated in relation to portable monitoring. WMI presents the simplest solution, and should be used for most of the operations, including information about Host Instances, Orchestrations, Send Ports and Recieve Locatations. Information that cannot be handled by this framework will be used with ExplorerOM library instead. The only relevant extension that this library present, is a simple framework for Application information. Last but not least, advanced features such as grouping error messages can only be done through direct database access. As of today, there exist no solution which can be used to retrieve all information, thus multiple solutions have to be used.

WMI, ExplorerOM and the database connection should be implemented using a WCF service on-premise. In other words, an application located locally on the server running BizTalk, with privileges to access information. Tools discussed in this part have consisted mainly of Microsoft products, since BizTalk environment is a Microsoft product which offer tools that can be used for further connections.
4 Part B - Global communication

Part B is the main part of the project and also the most challenging one. Data received from BizTalk in Part A need to be exposed to a portable device. Even though the project consists of developing a prototype app for Windows Phone in step 3, the global communication should not be limited to Windows Phone or mobile devices. Smartphones, media tablets and portable computers are examples of devices which should be able to connect to a solution presented in this part. The problem is illustrated in Figure 2.

![Figure 2: Two portable devices accessing four BizTalk platforms through cloud services. The blue rectangles represent the local network.](image)

In the classical client-server model, all devices will be assigned to either a client or a server. A server shares its resources and a client is a device which consumes these resources, such as Data, CPUs and printers [8]. In this classical pattern one or a few servers on the same network are configured to serve the needs of other users on the same or another network [8]. This is by far the most common scenario when discussing client and servers, see Figure 3a. However, in this project, the servers are already dedicated for another purpose. The servers are located in different networks and configured to provide a powerful but safe integration platform in BizTalk. These platforms are often large projects located at companies such as Fortum, Vattenfall, Clas Ohlson and similar [10]. Firewalls and server configurations at companies like Fortum are not likely to weaken security standards because of a project like this.
The most common case, as in figure 3a, is to configure firewalls and NATs to accept incoming connections from the clients. This is fairly easy when dealing with a few servers on the same network, but in this case, clients are supposed to connect to many server on completely different networks. As illustrated in figure 3b, these firewalls and NATs (red lines) are not configured the same way and has to be manually modified for each environment. Manually changing every server configuration and firewall is a error prone and dangerous road to travel. In combination with the statement that companies are not likely to be willing to change the configuration, this leads to one thing only. Changes in servers, firewalls and NATs should be none or minimal. This requirement deviates from the classical client-server idea.

![Figure 3a: Multiple clients accessing servers on the same network](image1)

![Figure 3b: A few clients accessing servers on different networks](image2)

When requests come from inside the local network (on-premise) where the server is located, the firewalls and NATs won't block requests. If a mobile phone should reside within the local network it should be fine, but, the whole idea of the project is to present a portable solution independent of location.

In order to solve this problem, the data from the server has to be publically exposed through the firewall. By installing a custom WCF service that extracts data from BizTalk (Part A) and then expose this to a cloud service, the problem can be approached.

### 4.1 Cloud Services

When using a cloud service, the portable device will always be able to reach the cloud. However, the cloud needs to be able to access the local service located on-premise (locally) on servers behind firewalls. Exposing an on-premise service publically is difficult, but there exists solutions which connect endpoints located behind firewalls to the cloud.

Azure Service Bus Relay is a cloud service dedicated for this purpose. Endpoints of services can be located in places which otherwise would be difficult or impossible to reach[11]. The reason why they are difficult to reach is because companies set up firewalls and NATs to block most of the traffic due to security reasons. The Service Bus Relay exposes an on-premise service securely to the cloud, through firewalls and NATs. Microsoft advertise this service as a product that can be used to expose a local service securely through the cloud even though firewalls and address translation boxes are involved[11]. This is true in most cases, but its not magic, the Relay has its limits. If a firewall is blocking all type of requests, the Relay will certainly not work. However, the Service Bus Relay uses a few techniques to try getting through the firewall and into the local server. By trying different protocols and ports that firewalls usually pass through such as TCP and HTTP, the Relay can hopefully connect even though behind a firewall.
The people developing this product have done their best to go through the firewall. If the Relay cannot connect to the cloud, the probability of being able to connect at all is low.

Service Bus Relay offers multiple ways to set up the server and its binding. Bindings that can be used consist of sessionful connections layered on top of TCP, and session less connections such as raw HTTP. With sessionful connections, clients have an established connection during the entire session (e.g., phone calls). Without sessions, each request is treated as a new request (e.g., SMS service). In [13], Thanapal point out that it is important to save battery life in mobile cloud computing. The paper implies that, if possible, most of the computation should be done remotely to save battery power. In Thanapals case, when using the cloud for computational offload, mobile bandwidth is also an important aspect. When sending computational tasks to the cloud, it could result in bandwidth issues if the dataset is large. In this project, the data is already available to the cloud, thus less mobile bandwidth needed. And since mobile bandwidth continues to expand rapidly [14], bandwidth should be of less importance for this project. When setting up this relay, according to Thanapals principle, computationally intensive operations should be performed in the cloud or on-premise, but not in the mobile. In [15], computationally intensive tasks performed remotely could save as much as 45% of battery life. An example of a mathematical task that could be implemented in this project is statistics from suspended messages. These statistics could prove useful to BizTalk experts interested in analyzing errors in their BizTalk platform.

Azure offers another cloud service especially for portable devices called Windows Azure Mobile Services and is still in preview mode [12]. Preview means that all of the expected functionality haven't been implemented, or isn't working properly yet. This service represents a very easy way of connecting mobile devices to the cloud securely. By the literature studies that have been made on this service, currently it is too simple and thereby lack important integrations needed for this project. Specifically, it is very easy to connect to the cloud, but not to an on-premise service. An on-premise service is located exactly as it sounds, locally on the premise. These services are often located behind firewalls and network address translation boxes which make it difficult to reach them. In order to connect to an on-premise service securely, the Service Bus Relay is the way to go.

Azure Service Bus Topics is yet another cloud service offered by Microsoft. Initially, this service was supposed to handle the bindings between an on-premise service and a portable device. Topics can be sent by the portable device in order to identify which BizTalk platform to connect to. By sending a topic from the mobile, the service bus identify which topic should be connected to which BizTalk platform. But Topics only supports publish/subscribe message model [16], resulting in an ineffective communication model. With publish/subscribe mechanism it is not possible to request a remote resource, but only to send a message. This message has to be manually retrieved and then reconstructed to form a response, which needs to be manually retrieved on the portable device. More code and configurations yields higher complexity and thus more error prone. Also, Topics does not offer the same connection to on-premise services as Service Bus Relay does. All in all, only using topics would not work and adding topics to the solution would only result in higher complexity and more likely to develop errors.

Another reason of using cloud services is related to access rights, which users should be able to access which BizTalk platforms. Access controls can be controlled either on the local server, or in the cloud. Since the portable device is supposed to access multiple servers in different environments, a local server solution would require changes in the server when modifying its access rights. If the access control service is located in the cloud instead, all access rights could be monitored and controlled from one source. Since the connection needs to be stateless (see 6.2.1), sessions cannot be established, resulting in a server that will have difficulties identifying
and authenticate users. If using server side implementation this could be solved with cookies or similar techniques. Also, if the server should be responsible for managing accounts, some sort of administrative tool needs to be developed. Otherwise it would be difficult to add and change accounts. When using cloud access control system, the system is centralized to the cloud and none of these problems occur. Azure Access Control Service present administrative tools to accomplish this, and can be linked together with Service Bus Relay [32]. Authentication can also be extended to support Facebook, Microsoft, Google and Yahoo account authentication [32]. Conclusions are, Azure ACS should be used for client authentication, and Service Bus Relay to reach the on-premise service.

4.2 Other cloud service providers

As stated earlier, the results presented in this report should not be limited to Microsoft products. This does not mean that Microsoft products cannot be a part of the proposed solution, but that the result must be useful for other implementations such as iOS and Android. The fact that BizTalk is a Microsoft product does give Microsoft the upper hand when evaluating a back-end. With that said, other solutions have been considered.

Amazon was an early adopter and one of the first to offer cloud computing as a service. Today it is one of the larger cloud service companies and have a huge portfolio of services [17]. As many other cloud computing companies, Amazon is mostly focusing on computational offload and storage options. Customers like Instagram state that even though they have 100 million users, Amazon EC2 enables them to effectively maintaining its infrastructure with just three employees [18]. The portfolio of services that Amazon offers does not present any cloud service that could be used to expose on-premise web-services through firewalls and NATs [17].

Hewlett Packard is another service provider which competes in the cloud computing business, and offer products similar to components in Azure Service Bus [20]. HP offers load balancers and messaging systems similar to the load balancers and Service Bus Messaging systems in Windows Azure. Even though some component from the Service Bus also exists in HP cloud, a service similar to Service Bus Relay does not exist.

Oracle Cloud also provides dynamic messaging capabilities in the cloud [21]. This message system is still in preview and is similar to both HP cloud messaging and Azure Service Bus Messaging. Currently, it does not provide an easy way of connecting to an on-premise service like Service Bus Relay does.

Google Cloud Platform is yet another platform with a number of services to offer, but it does not consist of a huge portfolio like Amazon [19]. Their portfolio of products only consists of a handful of services, and no service similar to Azure Service Bus Relay. Rackspace is also one of the larger participants in the cloud, but does not offer anything similar to the Service Bus Relay[22].

These are the larger providers of cloud computing services, and only one of them provides a product which offers a simple request/response messaging system to an on-premise web-service. Windows Azure is the only cloud provider among these to offer such as service.

4.3 Evaluation methods

Abul Nasir Kahn present a paper about secure mobile cloud computing and provides evaluation criteria for data and application security [6]. Kahn is mainly focusing on a cloud service which share mobile data, instead of messaging data, but the criteria are valid for this project too. Kahn
mentions seven different security criteria, and by applying these security criteria to the situation of BizTalk experts that are interested in portable monitoring, it is clear that security must be evaluated. Evaluation will be performed according to the following criteria.

4.3.1 Kahn’s criteria
Firstly, Kahn mentions basic theory which includes cryptographic principles and computational load on mobile devices. This is important though the energy parameter must be considered when using cryptography on a mobile device, thus its battery life has to be evaluated.

Secondly, Kahn mentions data protection which refers to the protection of the data on the mobile and in the cloud. With data protection, unauthorized users should not be able to access other person’s secret information.

The third criterion defines the data integrity parameter and this definition contributes to maintain the integrity of the data stored and passing through the cloud. Kahn clarifies that the data itself should be protected from modifications in this criterion, which is also an important security framework for the portable solution aiming at BizTalk experts.

Another criteria proposed by Kahn is scalability which focus on that the security framework must be highly scalable if users of the mobile app increases rapidly. This is considered less relevant but will still be discussed in the proposed solution.

The next criterion defines the underlying assumptions of the security framework and suggests that these assumptions must be weaker than the framework itself. Components can be classified as fully trusted, semi-trusted, or distrusted. If the assumption is that the component can be distrusted, the security framework can compromise the security of this component. Semi-trusted means that some functions can be compromised while others needs to be performed perfectly, for example integrity can be compromised while data protection is done properly.

Data access is another criterion and also an important part of the framework. It can be divided into automated and semi-automated access. Automated is when user does not need to retrieve new information such as a new password when accessing new or updated data. Semi-automated is when the user must provide more security information to access new information. In the case of BizTalk, this could be when the mobile user will be granted access to a new domain, in other words, granted access to another platform on a new server.

Last but not least, Kahn mention security authentication, meaning there should be some mechanism to verify users and data. In the case of server authentication, this could be if a user requests error information from a server located on a specific domain, there should be some mechanism to verify that the data is in fact delivered from the exact same server. In the case of client authentication, servers should be able to verify if the request comes from a trusted user.

The information that these experts will monitor could be highly confidential, and it is not acceptable to publicly expose this data. All criteria are interesting and are evaluated in section 6, however, some criteria are less important, such as scalability. Scalability will probably not be used though few people will be involved, and authentication is not that important since false data will not mislead any BizTalk expert. The worst thing that could happen is that they receive information about an error which has not happened. This is not probable and will be detected almost immediately when investigating the error further with the mobile or other tools such as BizTalk Administration Console.
4.3.2 CIA Triangle

Sandeep K. Sood discusses the CIA security triangle in combination with cloud computing [7], and this has been considered when designing a security solution for this project. When evaluating the security model for this project in section 6, the CIA triangle will be evaluated in relation to the portable solution.

Confidentiality, integrity and availability are the three building blocks of this triangle. Confidentiality refers to keeping the information secret, in the case of this project, it is when secret error information are exposed from the local on-premise service to the cloud and mobile. This information must be kept secret.

Integrity constitutes to ensure data integrity over the network, meaning the data should not be modified during transportation or in the cloud. In this project, this part is less relevant than confidentiality, tough this has already been mentioned. Availability is the last part of the triad, which aims to ensure that the system must be available when it is needed.

Sood discusses the CIA triad in relation to the cloud in two phases. The first phase deals with transmitting and storing data in the cloud, and the second about retrieving it. In phase one, Sood uses an algorithm consisting of evaluating the CIA elements in relation to the security requirements. The output of the algorithm classifies where the data should be located in the cloud, and on which security level. If a more manual solution of the project is constructed, this would be interesting to evaluate, otherwise, no important information such as usernames or error messages will be stored manually in the cloud. Most of the sensitive information will only be passing through, to the on-premise secure server. The second phase deals with retrieving the data in the same manual manner, by manual registration, security questions and so on. Kahn presents a more flexible solution, where Sood has presented a manual way of storing and retrieving data. Kahn’s security framework is more comprehensive, and is the preferred framework for this project.
Part C focuses on graphical presentation of the information received from Part B. This part aims to evaluate how the presentation should be done, and what to present. The problem associated with this part is illustrated in Figure 4.

Figure 4: Two mobile applications presenting information from the cloud

Literature studies made in this part have consisted of studying which software patterns to use and how the user interface should be evaluated. It is important to make a qualified analysis of what to present to the user, otherwise users will be confused. This resulted in a closer look at heuristic evaluation and user testing.

Heuristic evaluation is a method in software technology aiming to identify usability problems in the user interface. This type of evaluation requires an expert to evaluate problems with the interface. Even though most of the heuristics recommend more than one expert to evaluate the interface, only one expert is acceptable. A heuristic evaluation requires a lot of time from an expert evaluator, which makes it difficult to use expert evaluators for this project. Users cannot be used as evaluators, though this method is called user testing. User testing is when users of the prototype evaluates the interface, usually by navigation in a prototype application.

These methods have been discussed for decades, but Wei-siong Tan gives a relatively new and comprehensive evaluation of these methods in relation to each other [34]. The paper states that neither of the methods can be replaced by each other, but that heuristic evaluation is the most important source of detecting usability errors. Heuristic analysis should be implemented at early
stages to cover more high-level structural problems and address some of the root causes of these problems, while user testing should be conducted at a later stage[34]. Since the main purpose of this part is to create a user friendly prototype, the user testing has a much lower priority and has been neglected.

This project is unique in the sense that the target audience are all experts, in other words, the user interface should be presented so experts understand and can use it. The main problem regarding user interface is often that an expert designs an interface which should be used by non-experts. It is common sense that trying to put yourself in another person's position is a difficult and error prone way. Experts often falsely value cool functionality higher than usability, which is an example of failing to put yourself in the users position. The resulting product is rich in functionality, but difficult to use in an efficient way. In this project, it has been easier to evaluate usability since the user is also an expert. This is another reason why user testing will not be included in this project, even though heuristic evaluation can never truly replace user testing [34].

A short analysis was made directly during the literature studies, to be able to determine which key components Part A (Access BizTalk) needs to handle. The result of this initial literature study resulted in a few basic requirements. These requirements consisted of grouping suspended messages per application, retrieving status from applications/hosts, and a basic overview. In each of the parts, these requirements have been accounted for and discussed. The final functional requirements were to perform a heuristic evaluation of the user interface and a survey directed to people searching for errors in BizTalk. Section 5.2 and 5.3 discusses these problems which contribute to a more comprehensive analysis of the user interface.

5.2 Survey

The purpose of the survey was to find out which parts of BizTalk that was important to analyze, which could be referred to during the heuristic evaluation. Twenty questions were directed to people using BizTalk and searching for error/suspended messages. This section will only present the overall result, which is useful for the heuristic evaluation in 5.3 and the full survey can be located at the end in Appendix A.

The survey consisted of 20 questions and was directed to people working in different environments, in order to get an unbiased evaluation. Most of the BizTalk experts are consultants, and thus work in different environments even though they are employed within the same company. Nevertheless, the survey was handed to multiple companies. Mostly to avoid the risk of a biased evaluation, but also to receive as much answers as possible. Even though most of the companies working with BizTalk in Sweden were asked, only 14 people choose to participate. In general, few people work with integration and BizTalk which could be a reason to the low interest. Those who were asked were often positive, but did not have the time to complete a survey.

From the survey, almost all of the developers used BizTalk every working day. Questions that consisted of a grading criteria, is presented as an average score in percent. A 90% score represent an average of 90% importance, in other words, the component/object is of 90% importance to the average person. When searching for errors, which is the primary target in this project, BizTalk Administration Console scored the most important tool for error searching. With a 98% score (Question 2, Appendix A), BizTalk experts consider this application the most important in relation to competitive tools. Other answers consisted of the Windows Event Log, the ESB Portal and other tools. Even though the survey clearly shows that some of the tools were unknown to the developer, the Administration Console will be the main source of information when developing the application. Constructing an application which is familiar to
the real world is an important criterion discussed by Nielsen [28], and is presented as an evaluation criterion in 5.3.

In BizTalk Administration Console, Group Overview received a score of ~80% important and could be represented as a basic overview in the mobile application (Question 7, Appendix A). As for more specific information, suspended items, grouped suspended Instances, Send/Receive Ports received almost 100% all of them (Question 9-10, Appendix A). If grouping should be done when representing these, grouping by application would be the most important factor.

The survey indicates that statistic is not the only important information, but specific information of each error or application also is an important feature (Question 13-14, Appendix A). If notifications should be implemented, status changes of host instances should be an important feature, according to question 16 in the survey. The solution presented in Part A consists of request/response, and not publish/subscribe, which does result in database polling if notifications will be used. Polling is an operation where the resource is asked continuously at given intervals, in order to check for new information. According to the survey, BizTalk experts are interested in a notification delay no longer than 5 minutes (Question 18, Appendix A).

5.3 Heuristic evaluation

The heuristic evaluation will ease the task of designing a user interface that is easy to use and which BizTalk experts can relate to. What this means is to present information in the same structured way they are receiving information today and analyze which tools that are useful today. Heuristic evaluation will be performed according to Nielsen and Mack [27], which have more or less become a standard for this type of evaluation and are still used frequently [9]. Nielsen and Mack imply that all types of evaluations should be done with more than one person. This is not possible in the scope of this project, and the evaluation will only be performed by the author. Nielsen and Mack mention ten different criteria which will be used in section 6.4 Evaluation of Nielsen’s criteria. These are the criteria that will be evaluated, and was published 1994 by Nielsen in his book Usability Engineering [28].

Visibility of system status:
Users should always be informed by the system about what is going on, through relevant feedback within reasonable time.

Match between system and the real world:
The system should follow real-world convention and speak the same language as the user. Words and phrases should be familiar to the user, rather than the system-oriented terms familiar to the developer.

User control and freedom:
Users need a clearly marked exit path, if the users should have chosen the wrong function by mistake. This should be possible without having to go through a number of unwanted pages.

Consistency and standards:
Words, situations and actions should mean the same thing throughout the application. Follow platform conventions so the user doesn't have to wonder what different things mean.

Error prevention:
The design should be carefully evaluated to prevent problems from occurring in the first place. This could be achieved by eliminating error-prone conditions directly or by checking for them.
**Recognition rather than recall:**
Users' memory load should be minimized by making actions and objects visible. Information from one page to another should be visible, so the user easily can remember the choices. Minimize the user's memory load by making objects, actions, and options visible. Usage of the system should be visible if not easily understandable.

**Flexibility and efficiency of use:**
Expert features unseen by the novice users can be used by an expert to speed up the navigation process. In this way, the system is efficient for both experienced and inexperienced users.

**Aesthetic and minimalist design:**
Pages should not contain irrelevant information which is rarely needed. All objects on a page compete with the relevant information and decrease their perceived visibility.

**Help users recognize, diagnose, and recover from errors:**
Error messages should be expressed without codes in order to indicate the problem, and help users recognize and recover from the error.

**Help and documentation:**
If the system cannot be used without proper documentation, this should be easy to search and focus on the user. Simplicity over complexity.

### 5.4 Prototype application

In order to demonstrate the solution, a prototype application had to be developed. The solution should work with all portable devices within the smartphone or tablet segment. Windows Phone 8 was selected as the operating system for the prototype, but it could have been Android or iOS as well.

Microsoft does not recommend classical MVC software pattern for Windows Phone, instead they recommend the slightly different MVVM pattern for both Windows Phone and Windows 8 apps[36]. It is possible to build applications for Windows Phone using traditional patterns also, but programming in MVVM brings quite a few advantages [37].

The MVVM pattern result in minimal to none GUI programming, and thus be able to focus on core functionality and design. The key benefit is enabling true separation between the two main components, the model and the view. What this means is that when the model needs to change, it can, without the view needing to and vice-versa. MVVM is a pattern largely based on the traditional MVC pattern, but targeted at modern UI development which supports event-driven programming such as Silverlight, HTML 5 and WPF. Silverlight development is almost exactly the same as in Windows Phone.

MVVM represents the model (M) and the view(V) as in the classical MVC pattern. What is new is the view-model (VM) which binds the model to the view. It converts the contents from the model and binds it to the view which minimizes GUI coding. The view-model is a “model of the view”, in other words, it serves as an abstraction of the view that is in charge of exposing the real model to the view. This view-model changes model information into view information which can be accessed from the view and in a sense it could be perceived as a controller, but the main difference is that it binds model data to the view. When the model changes, the view is updated automatically, without any code acting as a controller. Since this project will not focus on programming, this pattern is an excellent solution where design and functionality can be prioritized.
The prototype application constitutes to the last part of the project and binds all parts together. Development of the application consisted of following Microsoft’s pattern guidelines, the survey in 5.2, the heuristics presented in 5.3 and earlier requirements set in Part A and Part B. According to the survey in 5.2, BizTalk administration console was the most important tool to search for errors, with a score of 98% importance. The heuristics by Nielsen stated clearly that the relation between the real world and the application must be solid, thus has the development of the Windows Phone application been concentrated around BizTalk administration console. Group overview is an important view in the administration console, scoring 80% valuable in 5.2, and will be the root of the smartphone app. This overview have been translated into a native phone app. Figure 5 is a view of the Group Overview in BizTalk administration console.

Group overview has been converted into this phone application, with a minimalistic design and touch friendly user interface.
The large squares, also referred to as tiles, are clickable and present a list of information and status about the underlying item. Each row of tiles represent an underlying pivot view with specific information about the tile, and swipeable to see information about next tile. These three images represent the first row, Applications, Host Instances, Orchestrations and so forth.

Development of this application consisted of following MVVM software pattern, survey performed in 5.2, and heuristics mentioned in 5.3. Only a few of the heuristics by Nielsen were presented in this section, but all of them were considered during development. Section 6.4 ‘Evaluation of user interface’ present a deeper look into the heuristic evaluation performed when constructing this application.
6 Evaluation

6.1 Proposed solution
Tools to access BizTalk, discussed in Part A, consist of WMI, ExplorerOM framework and direct database access. The WCF service is responsible of controlling these, and exposes an endpoint which can be exposed to the cloud. Service bus relay connects WCF endpoints with Access Control System for security reasons. Communication is handled using HTTP and REST, a technique which can be used with other portable devices such as iOS and Android. Each square under the service bus relay represent a unique URI which is used to connect the portable device with the on-premise WCF service using REST.

Figure 8: Proposed solution where two mobiles are connected to four local BizTalk platforms through cloud services.

Once the infrastructure has been set up it is easy to add more clients and servers. Adding a BizTalk platform to the cloud is performed by installing the WCF service at the local BizTalk server, and configures a unique endpoint which will be exposed to the service bus relay. In order for a client to connect to this platform, the URI must be added to the user in the Access Control System. Adding a user is even easier, and performed using Azure ACS portal. This portal provides a graphical user interface where all the access rights can be maintained from one
place. Access rights are controlled by the unique URI that serves as an endpoint to a BizTalk platform.

6.2 Evaluation of Kahn’s criteria

6.2.1 Basic theory
This criterion discusses the cryptographic principles and computational requirement on the portable device. Computational requirements are an important part of mobile security solutions and cannot be neglected. Mobiles have limited battery power and cannot have a security framework which drains the battery. Thanapal also discusses this problem, and states that this issue needs to be evaluated [13].

A synchronous solution where the communication is handled by an active connection is not desirable. In this case, it means that the mobile starts a new active session when issuing a connection to the cloud, and this session won't end until the user closes the app. Sessions enable an easy way of connecting devices and detecting which requests comes from which user, and its a common solution to a problem like this. However, this solution is not suitable for mobile devices due to multiple reasons. An active connection would prevent the device to enter standby mode, and thus constantly draining power. Since the phone needs to be in constant contact with the cloud, it could also result in a data overhead, meaning data is sent though no new information have been requested. Users with mobile data connection are often concerned about the amount of data since their telephone operator often offers a limited amount of data per month.

An asynchronous solution is a better approach when developing mobile solutions. This means that the mobile only needs to access the internet when issuing a request. By applying this, all requests are considered to be new requests and enable a stateless implementation. This implementation makes it difficult to separate clients from each other, but can be solved with cookies or session variables that are saved on the client. The proposed solution uses a web client in the mobile app to issue all connections over HTTPS. This web client also have the ability to cache identical requests, resulting in a smoother user experience and less power consumption. Since the cloud needs some sort of verification of the client, all requests contain custom headers with authentication tokens. This result in a small but needed overhead in order to block unauthorized clients. The response is also asynchronous, meaning the current thread is not blocked by waiting for the response. Asynchronous response enables the user to navigate through the user interface while waiting for a reply. When the response is pushed to the mobile, an interrupt occurs and the corresponding method is resumed.

6.2.2 Data protection
Data protection contributes with protection security on data transported between the on-premise web-service and the portable device. Unauthorized users should not be able to interpret information sent over the internet. BizTalk is an expensive product mostly aiming at larger companies. Large companies (and some small) companies often have valuable data, which must be protected. BizTalk handles low level communication within the company and can expose this type of secure information. Therefore the data protection criteria is the most important criteria throughout this project.

Protection of data can be achieved in multiple ways. As stated in 6.1, the proposed solution consists of server certificates located at Windows Azure. Client certificate are not used, though the identity of the client is not important in the protection layer, this is discussed further in
Authentication (6.2.7). The server certificates currently uses TLS 1.0 with 128-bit AES encryption, and is managed and controlled by Windows Azure solely.

Advantages of using server certificates controlled by an authority like Microsoft are many. Certificates do not need to be created and managed in the solution, as this service is offered by Microsoft. The certificate is automatically trusted by root authorities, and does not produce warnings. If a fully trusted certificate is needed, this is usually bought at companies like VeriSign [25]. Currently the connection is issued through TLS 1.0, but as technology develops, the Azure team can change the certificate, if more efficient or secure solutions appear in the future.

Disadvantages consist of less control, since the certificates are managed by Azure they could be changed or expire at any time. Windows Azure has already suffered from this twice in the past 2 years[26]. This issue is described further in 6.3.3 which discusses availability problems that is a part of the CIA-triangle.

6.2.3 Data integrity

Integrity refers to the accuracy and consistency of data during its entire life-cycle. Data passing through the cloud and to local servers must be accurate and reliable for further processing. This criterion is emphasized when developing solutions sending secret and important information. If the validity and correctness of the data cannot be fully trusted, modified data could mislead users and systems, which could cause serious damage.

In this project, integrity is desirable but not a requirement. Even though highly sensitive information could be sent, the modification itself is not an issue. If integrity is compromised, BizTalk experts using the portable solution could receive false errors. This would only happen if the network is attacked, and the result of this compromise would not be catastrophically. When BizTalk experts continue to evaluate the false error with the portable solution or administration console, they will quickly realize that a false alarm has been raised. Not being notified when an error occurs is also a problem if integrity is compromised. This is not desirable and should be avoided if possible, but does not contribute to a critical security problem in this project.

The proposed solution uses a HTTPS connection with server certificate to connect to the cloud. This server certificate is controlled and managed by Windows Azure, including security mechanisms and algorithms. In the connection that has been set up for this project, Azure uses TLS 1.0 with 128 bit encryption as mentioned earlier. With this type of encryption, integrity between these connections is preserved. As a result of the data protection requirement (6.2.2), integrity is automatically achieved.

6.2.4 Scalability

If users increase, the system should be stable and still support a secure connection. This could be if other users than BizTalk expert want to take advantage of the system and/or if the system is implemented on more servers. Since all of the data is channeled through the service bus relay and access control system, this could become a bottleneck. An advantage of using a cloud service like Azure is that it’s highly scalable and supposed to handle big data. Even though all calls pass through this relay, it should not limit the connection. If it does, it won't affect the security in Azure Access Control System.

Conclusions drawn from this project does not regard scalability. It is believed that scalability should not be a problem in this solution, but since this feature is not needed, no further investigation has been made.
6.2.5 Assumption
The framework itself must be stronger than the assumptions of it. Assumptions of the components of this project are that most of the criteria must be taken into account, but not all of them. Therefore, the requirement is regarded as semi-trusted, where scalability and integrity can be compromised, even though integrity isn't compromised in the solution.

6.2.6 Data access
Kahn discusses data access in combination with sharing resources, while this paper will focus on accessing multiple resources (BizTalk platforms). Data access is when accessing a remote BizTalk server and can be divided into automated and semi-automated access. In the case of this project, automated means that data access can be controlled within the system without the involvement of other means, e.g. email, SMS, or call. Semi-automated is when the data access can be controlled by sending secret information (e.g. password, secret key) through other means.

The proposed solution provides data access control in Azure Access Control System, thus can be controlled within the system. If a mobile user needs to access another BizTalk platform, this could be changed in Access Control System at runtime, without involvement of physical means (e.g. email, SMS), and thereby be able to access another platform with the same account. Once the user account has been created, the user will never have to share secret information to access new or updated data.

6.2.7 Authentication
Authentication can be divided in both server- and client-side authentication. With server authentication, mobile users should be able to verify which server the user is connected to. Even though server authentication is a positive aspect it’s not a critical part. Worst case scenario is that a client trusts a hacked server, and false information is presented to the user. Crucial information is not leaked, and motivations for an attack like this do not exist.

Authentication of the client is a more important criterion, though critical information can be leaked. If a BizTalk server does not know which device it is connected to, information about the server can be leaked. By verifying the client through Azure Access Control System (ACS), this problem can be solved.

The proposed solution uses Secure Socket Layer (SSL) with server certificate. With server certificate, the cloud server (Azure Cloud) issues a secure connection between a BizTalk server and the mobile application. This connection is not issued if the credentials does not pass client authentication which is performed in ACS. Since the connection is stateless (hence 6.2.1), each request contains SSL encrypted username and password, and is validated through ACS.

6.3 Evaluation of CIA-triangle

6.3.1 Confidentiality and Integrity
These two components of the CIA triangle are covered by Kahn’s criteria and have already been discussed. Data integrity is discussed in 6.2.3, and confidentiality is covered by Kahn’s criteria about protection (6.2.2).

6.3.3 Availability
Availability is often an important aspect of security, and information systems that are not available when needed is almost as bad as none at all. A disadvantage of relying on other
systems (such as cloud services), is loss of control. Windows Azure Cloud is relatively new and has already suffered from several critical downtime events. In February 2012 Azure Cloud Services went offline for 12 hours because of a leap-year bug with a security certificate [26]. Almost a year later, in February 2013 it went offline again, this time due to an expired certificate [26].

Loss of availability could be a serious issue in some situations, but not in this project. The portable solution is provided as a complementary way of searching for errors. A few hours downtime per year does not justify choosing another solution. Another loss of availability is a deliberate attack which could result in longer downtime. However, such an attack must have a clear objective, and usage of an application like this does not have any such a motive.

6.4 Evaluation of user interface
The criteria evaluated in this section are performed according to Nielsen and Mack [27]. Nielsen and Mack mention ten different criteria for evaluation of user interface. These criteria have more or less become a standard, and aim to achieve higher usability [9].

6.4.1 Visibility of system status
Application status should be clearly visible within the the mobile app, to inform users of ongoing activities. Group overview described in 5.4 represents the status of a BizTalk platform and is easily accessed from all views in the application. This overview provides the user with a basic overview of system status. If status changes, it will be visible within this overview. The overview represents all elements controlled by the application, in order to provide a basic information to the user.

6.4.2 Match between system and the real world
The application speaks the user’s language, with words and phrases similar to BizTalk Administration Console. Since the Administration Console is the most useful tool (hence 5.2), the application use the same words and phrases BizTalk experts are used to. Other naming conventions, exemplified in 3.3 are not familiar to BizTalk experts, and will not be used. By applying this, the application follow real-world conventions, thus making information appear in a natural and logical order. The structure of the application follows the same pattern as BizTalk Administration Console by the implementation of Group Overview and navigation to error messages.

6.4.3 User control and freedom
The application does not contain unwanted states, where the user’s needs to go through an extended dialogue as a result of choosing the wrong functions by mistake. The back button is a required hardware button for windows phone devices and serves as an "emergency exit" to leave the unwanted states. The mobile app presented in 5.4 does not contain an extensive amount of navigation, and has been constructed to present only the most relevant data according to BizTalk experts, based on the survey and heuristics presented in 5.2 and 5.3.

6.4.4 Consistency and standards
In order to follow platform conventions, the application has been developed according to Microsoft’s guidelines, with elements similar to other windows phone applications, such as tiles, panoramic views and pivot elements. Each row of tiles, described in 5.4, represent an underlying pivot element. A pivot view represents multiple views/objects, and the views are changed by swiping left and right. By using consistency, the user is aware of application state, thus can be used in a more efficient way.
6.4.5 Error prevention

Error prevention is something every developer aim for, and is a huge topic. Design paths that have been chosen to prevent errors consist of following. The prototype has been developed according to the MVVM model to minimizing coding. Its common knowledge that more code results in a higher probability of errors. Even though there are exceptions, pointed out by Neville-Neil [35], these situations rarely occur. Error-prone conditions are checked in the application and presented to the user when occurring. Some errors are presented with a sad smiley, similar to the new Blue Screen of Death (BSOD) in Windows 8 (consistency and standards 6.4.4).

6.4.6 Recognition rather than recall

The user does not have to remember information from one page to another; otherwise the user could question the results or forget what was chosen. When navigating to the underlying pivot view from group overview, name and value from the overview is consistent with the view in the pivot page, demonstrated in 5.4. The user does not need to remember which BizTalk platform is connected, though this information can easily been seen from any view. More detailed information about the server can be viewed in the overview.

6.4.7 Flexibility and efficiency of use

Accelerators, used by expert users to speed up interaction process, can make the system efficient for both inexperienced and experienced users. The target audience of this application does not differ much in experience, though most of them are already experts. And because this is only a prototype, this criterion has not been evaluated.

6.4.8 Aesthetic and minimalist design

Minimalistic design is an important criterion, though every extra unit of information competes with the relevant units of information and lowers their relative visibility. The group overview has been designed in the most minimalistic possible way. By only displaying the most relevant information (e.g. amount of errors), the overview cover most of the important information from the survey. If more information is desired from the overview, the clickable tiles provide this.

6.4.9 Help users recognize, diagnose, and recover from errors

Currently, most error messages are caught in the application, and presented to the user in plain text, as suggested in this criteria. This is possibly the best idea, but has to be evaluated further in a production environment, not accessible through the scope of this project. It is possible, that these types of errors are confidential to the mobile user, and should not be presented.

6.4.10 Help and documentation

Currently, the prototype application is only a demonstration of what could be done, and only provides the most important features. This result in a minimalistic application, and an easy navigation process. The application in its current state does not need help and documentation, and if this is desirable, section 5.4 describes the most basic steps. As for larger applications with complex functions, help and documentation is a more crucial criterion.
7 Discussion

7.1 General
The proposed solution described in this project is only a first step in developing a product. Focus has been on infrastructure and theory, and not on a larger product with advanced functions. Basic functions which have proven most useful according to the survey (5.2), are currently the only ones implemented. The motivation for each criterion has been based on the prototype, and not a fully developed product. Criteria such as “6.4.10 Help and documentation” are an excellent example of a criterion which must be evaluated further before launching a commercial product.

7.2 Android and iOS
At multiple times, the use of “Microsoft only” software has been discussed. Since BizTalk is a Microsoft product, it has been difficult to provide better tools from other companies than Microsoft. Windows Azure is a cloud platform not only for Microsoft devices and provide excellent support for Android and iOS systems. It is clear that Azure has a humble approach towards other companies, an example of that is the Azure Notification Hub which was developed for Apple’s iOS before Windows Phone. All in all, Azure is designed to provide features that the market is demanding, independent of company. The proposed solution is using Azure and REST, and is by no means limited to Microsoft clients. If an Android or iOS prototype had been the final checkpoint of this project, the solution would still have been the same.

7.3 Security
Security has been an important part of the project, since most BizTalk platforms contain secret information. When designing security solutions, it is important to rely on frameworks to understand which areas to focus on. All systems are more or less possible to hack, though the question is how difficult should it be. Some criteria such as 6.2.2 Data protection have been given more attention in this paper, because motive for this type of attack exists. However, even for this criterion there exist limitations. German scientists have proven to retrieve information from the memory of the phone by freezing it and perform a custom boot [38]. By this method, secret information could potentially be leaked from client phones. Even though the framework presented in this paper is strong, it can be hacked using the most advanced techniques. It is unlikely that such a high motivation will ever exist towards BizTalk platforms.

7.2 Future work
This project has consisted of evaluating how to monitor BizTalk from a portable device and a prototype in Windows Phone 8. If this solution should be part of a saleable product, Android and iOS applications must also be developed. This will result in a larger product which will take time and money to construct. The next step is to analyze what could be charged of a product like this, and if the industry is willing to pay such a price. According to the survey (5.2), BizTalk experts are interested in a portable solution. However, it is unclear how much experts are willing to pay for such a solution.

A complete product will involve the Android and iOS platforms as well. Part A and Part B is fully compatible with other platforms, since the solution uses HTTP REST calls. All authentication and back-end management will be controlled from the cloud, independent of
client platform. Most of the evaluation performed in Part C can be reused when developing for another platform. The survey will still be an important asset, and evaluation can be performed according to evaluation criteria in Part C.

7.3 Response from BizTalk experts

During the end of the project, the proposed solution was presented to BizTalk experts. The solution was interesting and much welcome by the experts that attended the presentation. All of them could see possible use cases where this type of portable solution was desirable. The simple management of authentication and highly flexible back-end was one of the most positive aspects. Theory and back-end was the goal of this project, and the mobile prototype was constructed only to demonstrate the solution.

At the presentation, a salesman from Microsoft was invited. The salesman was clearly interested in the results, but did also offer some critique. Critique was mainly focused on the prototype and its functions. In order to produce a product, the prototype must be refined and polished to include more complex functions. At the presentation, notifications were only handled at a basic stage. Windows Phone does not offer a notification center like Android and iOS, thus information about the notification disappears directly. The salesman from Microsoft suggested a solution with live tiles instead. Live tiles are large squares located at the home screen (similar to the Group overview in Figure 6a, 6b), and is automatically updated by notifications. This would help BizTalk users to see how many errors have occurred or the how many host instances have been changed, directly from the home screen of their phone.

The salesman was also asked if Microsoft could be part of promoting the continuation of a project like this. Even though the answer was vague, it wasn’t a definite “no”. It is not impossible that Microsoft is willing to promote a mobile solution as presented in this project. Currently (until 2013-06-30), Microsoft is offering developers money just to build apps for Windows 8 and Windows Phone 8 [40]. Before asking Microsoft for promotion, the prototype must be refined and polished. The infrastructure that has been evaluated throughout this project is highly usable and shouldn’t need much modification. Future work should be focused on mobile apps.

Shortly after the presentation, the BizTalk experts decided to perform the first steps described in 7.2 Future Work. These first steps consist of evaluating whether selling a portable solution like this is profitable.
8 Conclusion

There exists no tool to successfully manage all communication with BizTalk in Part A. WMI and ExplorerOM was most suitable for common operations on BizTalk, and more advanced features could be controlled through database access. Evaluations performed in Part B suggest that computational intensive tasks should be located in the cloud or on-premise, due to limited battery power in mobiles. Authentication of clients is performed in the cloud, in order to control all access rights from one place. According to the survey performed in Part C and 7.3 Response from BizTalk experts, a portable solution is highly desirable. The next step is to evaluate whether this is a service that can be sold. If so, the prototype needs to be developed into a product for Android and iOS.

Evaluation criteria proposed by Kahn [6] have been the most important part of this project. By applying these criteria, solutions could be more easily classified and prioritized. The criteria have been used mostly in Part B, but have also been effective in the other parts. According to section 7, BizTalk experts are enthusiastic by this solution and will hopefully continue the project.


24. Gartner, Aug 2012, Figure 1: Hype cycle for emerging technologies 2012, (as of July 2012)
35. Neville-neil, December 2012 George, Can more code mean fewer bugs, V. Communications of the ACM, Vol.55 (12), pp.31-32
38. Tilo Muller, Michael Spreitzenbarth, and Felix C. October 2012, Freiling Frost Forensic Recovery of Scrambled Telephones, Department of Computer Science, Friedrich-Alexander University of Erlangen-Nuremberg
Appendix A – Survey
1) Hur ofta använder du BizTalk

<table>
<thead>
<tr>
<th></th>
<th>Varje arbetsdag</th>
<th>Varje vecka</th>
<th>Varje månad</th>
<th>Aldrig (Fortsätt inte med enkäten)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9 (64.3%)</td>
<td>2 (14.3%)</td>
<td>1 (7.1%)</td>
<td>2 (14.3%)</td>
</tr>
</tbody>
</table>

2) Hur betydande är dessa verktyg när du felsöker din BizTalk-platform

<table>
<thead>
<tr>
<th>Verktyg</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>Mycket viktig</th>
<th>Responses</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventloggen i Windows</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>50.0%</td>
<td>12</td>
<td>4.33 / 5</td>
</tr>
<tr>
<td>BizTalk Administration Console</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>11</td>
<td>91.67%</td>
<td>12</td>
<td>4.92 / 5</td>
</tr>
<tr>
<td>ESB Portal/Management Console</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>10</td>
<td>2.10 / 5</td>
</tr>
<tr>
<td>Felsökning på annat sätt</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2.50 / 5</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

3) Om Annat. Hur felsöker du?

- If principle varje dag
- Genom vårt egenutvecklade verktyg "Zupend Monitor". Som prenumererar på unika suspmsg situationer
- Queries i databasen
- Det beror på integrationen och om den har ytterligare felsökningsmöjligheter, tex egna loggar eller liknande
- Attach to process i VS för att kunna stega sig igenom delar av en integration tex pipelinekomponenten
- Orchestration debugger
- Spårbarhet via egen DB-storage alt BAM

4) Har du använt ESB Portal/Management Console för felsökning?

<table>
<thead>
<tr>
<th>Ja</th>
<th>3 (25.0%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nej</td>
<td>9 (75.0%)</td>
</tr>
<tr>
<td>Vet ej</td>
<td>0 (0.0%)</td>
</tr>
</tbody>
</table>

5) En central del i presentation av statistisk och felinformation är antalet element. Försök att uppskatta antalet instaser av
dessa i den platform du arbetar mest med.

<table>
<thead>
<tr>
<th></th>
<th>1-4</th>
<th>5-14</th>
<th>15-49</th>
<th>50-199</th>
<th>200+</th>
<th>vill ej svara</th>
<th>Responses</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Totalt antal applikationer</td>
<td>1 (8.33%)</td>
<td>4 (33.33%)</td>
<td>6 (50.00%)</td>
<td>1 (8.33%)</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.58 / 5 (51.60%)</td>
</tr>
<tr>
<td>Totalt antal host-instanser</td>
<td>1 (8.33%)</td>
<td>10 (83.33%)</td>
<td>0 (0.00%)</td>
<td>1 (8.33%)</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.08 / 5 (41.60%)</td>
</tr>
<tr>
<td>Orkestreringar per applikation</td>
<td>5 (41.67%)</td>
<td>4 (33.33%)</td>
<td>2 (16.67%)</td>
<td>1 (8.33%)</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>1.92 / 5 (38.40%)</td>
</tr>
<tr>
<td>Send Ports per applikation</td>
<td>2 (16.67%)</td>
<td>6 (50.00%)</td>
<td>3 (25.00%)</td>
<td>1 (8.33%)</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.25 / 5 (45.00%)</td>
</tr>
<tr>
<td>Reveice Locations per applikation</td>
<td>3 (25.00%)</td>
<td>5 (41.67%)</td>
<td>3 (25.00%)</td>
<td>1 (8.33%)</td>
<td>0 (0.00%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.17 / 5 (43.40%)</td>
</tr>
</tbody>
</table>

6) I BizTalk Server Administration kan man skapa olika grupper (eller servrar) som innehåller flera Applikationer och Host Instanser. Hur många grupper är typiskt för den platform du arbetar med?

<table>
<thead>
<tr>
<th>1</th>
<th>2-9</th>
<th>10+</th>
<th>vet ej</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9 (75.0%)</td>
<td>3 (25.0%)</td>
<td>0 (0.0%)</td>
</tr>
</tbody>
</table>

7) Genom att klicka på en grupp i BizTalk administration console kan man se information för hela gruppen (Group Overview), så som antalet suspenderade meddelanden, host instanser, pågående aktiviter, osv. Hur viktig är denna överblick ur ett felsökningsperspektiv?

<table>
<thead>
<tr>
<th>Välj</th>
<th>1 (8.33%)</th>
<th>1 (8.33%)</th>
<th>1 (8.33%)</th>
<th>4 (33.33%)</th>
<th>5 (41.67%)</th>
<th>Responses</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Välj</td>
<td>12</td>
<td>3.92 / 5 (78.40%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8) Är det något specifikt du känner att du saknar i den vyn?

- Gruppering som visar suspmsg baserat på dess unika feltyp
- Dependecymanagement mellan resurser och artefakter, men den funktionaliteten kommer i BT2013

10) Hur intresserad är du av att se detaljerad information om följande typer, dvs namn, status, egenskaper osv, för varje instans av den här typen:

<table>
<thead>
<tr>
<th>Typ</th>
<th>1 Inte intresserad</th>
<th>2 Lagom</th>
<th>3 Mycket intresserad</th>
<th>4 Vet ej</th>
<th>Responses</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Overview</td>
<td>2 (18.18%)</td>
<td>5 (45.45%)</td>
<td>4 (36.36%)</td>
<td>0 (0.00%)</td>
<td>11</td>
<td>2.18 / 3(72.67%)</td>
</tr>
<tr>
<td>Work in Progress</td>
<td>1 (8.33%)</td>
<td>10 (83.33%)</td>
<td>5 (41.67%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.00 / 3(66.67%)</td>
</tr>
<tr>
<td>Suspended Items</td>
<td>0 (0.00%)</td>
<td>1 (8.33%)</td>
<td>11 (91.67%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.92 / 3(97.33%)</td>
</tr>
<tr>
<td>Grouped Suspended Service Instances</td>
<td>0 (0.00%)</td>
<td>5 (41.67%)</td>
<td>7 (58.33%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.58 / 3(86.00%)</td>
</tr>
<tr>
<td>Tracked Service Instances</td>
<td>2 (16.67%)</td>
<td>5 (41.67%)</td>
<td>5 (41.67%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.25 / 3(75.00%)</td>
</tr>
<tr>
<td>Tracked Message Events</td>
<td>4 (33.33%)</td>
<td>2 (16.67%)</td>
<td>6 (50.00%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>2.17 / 3(72.33%)</td>
</tr>
<tr>
<td>EDI/EDINT Status</td>
<td>7 (58.33%)</td>
<td>3 (25.00%)</td>
<td>0 (0.00%)</td>
<td>2 (16.67%)</td>
<td>12</td>
<td>1.30 / 3(43.33%)</td>
</tr>
</tbody>
</table>

2.22 / 3 (74.08%) |

11) Om du fick gruppera orkestreringar, send ports, receive ports osv för att underlätta felsökning. Hur vill du helst ha den informationen grupperad? Det går att kryssa i flera

<table>
<thead>
<tr>
<th>Grupperat per Applikation</th>
<th>11 (64.7%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grupperat per Host Instans</td>
<td>4 (23.5%)</td>
</tr>
<tr>
<td>Annan gruppering</td>
<td>2 (11.8%)</td>
</tr>
</tbody>
</table>

12) Om annan gruppering, vad för gruppering?

- Grupperat per typ av felmeddelande
- Grupperat per adapter
13) Är det intressant att se utförlig beskrivning för ett specifikt fel, dvs namn, status, errorID för ett suspenderat meddelande

<table>
<thead>
<tr>
<th>Svarsval</th>
<th>Antal</th>
<th>Procent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ja, mycket viktigt</td>
<td>8</td>
<td>66.7%</td>
</tr>
<tr>
<td>Ja, lite grann</td>
<td>4</td>
<td>33.3%</td>
</tr>
<tr>
<td>Nej</td>
<td>0</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

14) Om du är tvungen att välja, vilket anser du skulle vara viktigast att presentera gällande suspenderade felmeddelanden.

<table>
<thead>
<tr>
<th>Inte Viktigt</th>
<th>Lite Viktigt</th>
<th>Viktigt</th>
<th>Grouped Antal</th>
<th>Likaviktiga</th>
<th>Information om specific fel</th>
<th>Vet Ej</th>
<th>Responses</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (8.33%)</td>
<td>0 (0.00%)</td>
<td>5 (41.67%)</td>
<td>0 (0.00%)</td>
<td>6 (50.00%)</td>
<td>0 (0.00%)</td>
<td>12</td>
<td>3.83 / 5 (76.60%)</td>
<td></td>
</tr>
</tbody>
</table>

15) Om felmeddelanden skulle grupperas. Hur viktigt skulle det vara att kunna gruppera på:

<table>
<thead>
<tr>
<th>Inte Viktigt</th>
<th>Lite Viktigt</th>
<th>Viktigt</th>
<th>Responses</th>
<th>Average Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ogrupperat</td>
<td>5 (50.00%)</td>
<td>5 (50.00%)</td>
<td>0 (0.00%)</td>
<td>10</td>
</tr>
<tr>
<td>applikationer</td>
<td>1 (8.33%)</td>
<td>4 (33.33%)</td>
<td>7 (58.33%)</td>
<td>12</td>
</tr>
<tr>
<td>host instanser</td>
<td>2 (16.67%)</td>
<td>9 (75.00%)</td>
<td>1 (8.33%)</td>
<td>12</td>
</tr>
<tr>
<td>feltyp</td>
<td>0 (0.00%)</td>
<td>7 (58.33%)</td>
<td>5 (41.67%)</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Ett fel har inträffat</th>
<th>Ny feltyp har registrerats</th>
<th>Förändring av status på host-instanser</th>
<th>Annat</th>
<th>Inte intressant med notiser</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 (20.0%)</td>
<td>4 (16.0%)</td>
<td>9 (36.0%)</td>
<td>6 (24.0%)</td>
<td>1 (4.0%)</td>
</tr>
</tbody>
</table>

17) Om annat. Vad för information kan vara intressant att få notiser om?
18) Baserad på din arbetssituation, var går gränsen för hur snabbt du vill ha ny information pushad till din mobil(notiser), exempelvis om felmeddelanden? Anledningen till att jag frågar detta, är för att BizTalk måste pollas med dessa intervall.

- < 1 sek: 0 (0.0%)
- < 30 sek: 0 (0.0%)
- < 5 min: 9 (75.0%)
- < 30 min: 3 (25.0%)
- < 2 h: 0 (0.0%)
- Förstår inte frågan: 0 (0.0%)

19) Finns det något du känner att du vill tillägga, något som enkäten missat.

- Arbets erfarenhet, hur många år man jobbat osv
- Iom svaret på fråga 17 skulle svaret på fråga 18 bli att pollning sker när vissa kriterier är uppfyllda
- Bra enkät