Challenges and Lessons Learned Using Blaise IS with SQL Server

Max Malhotra and Jas Sokhal
Survey Research Center, University of Michigan

1 Introduction

The University of Michigan's Survey Research Center Survey Research Operations unit (SRC/SRO) is engaged in a collaborative Army Study to Assess Risk and Resilience in Service members (STARRS). This is the largest study of mental health risk and resilience ever conducted among military personnel with a goal to collect data from approximately 100,000 Soldiers over a 2-year period. The National Institute of Mental Health (NIMH) assembled a group of experts to carry out this research, including teams from the Uniformed Services University of Health Sciences (USUHS), University of Michigan, Harvard University, the University of California-San Diego, and NIMH. The Technical Services Group (TSG) at SRC was assigned the task of Systems Development for the survey data collection, the management of data, and the control of data.

Computer Assisted Interview (CAI) data collection had to be performed at the remote Army installations located around the United States with no access to the Army network infrastructure. This created unique challenges in terms of maintaining data, confidentiality, integrity and availability.

The requirements included:

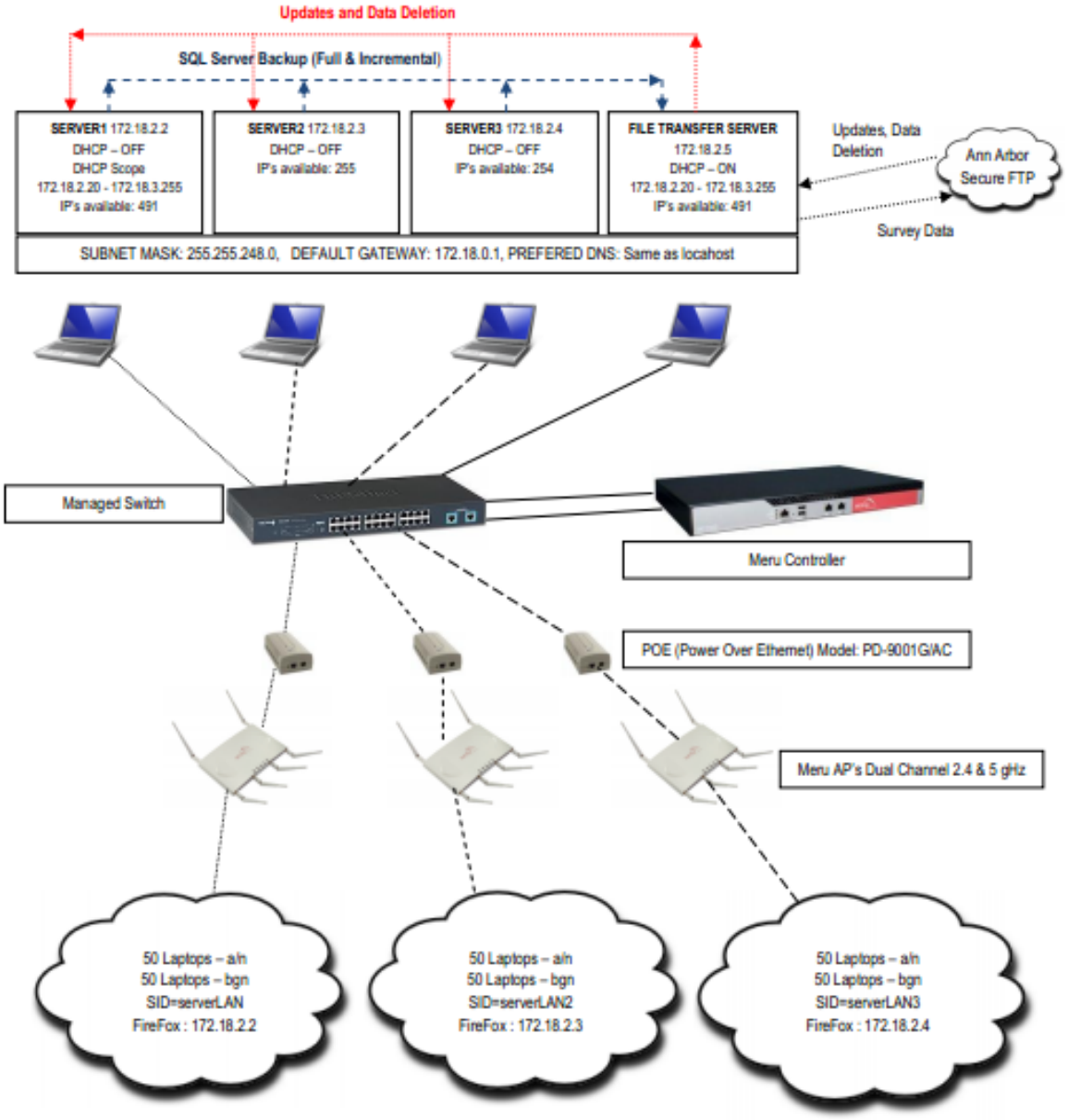
- Federal Information Processing Standards (FIPS) Compliance
- Bitlocker Encryption
- A secure relational database support Transparent Data Encryption (TDE)
- A Secure protocol for transferring data to and from the Data Management Center
- Data deletion methodology on remote servers
- The capability to integrate Neurocognitive testing

After an extensive evaluation of available data collection software and tools, TSG chose Blaise IS as the main data collection instrument front end with a MS-SQL Server back end. The data collection started in winter 2010, and as with any first Production launch, it had its fair share of complications, particularly with Blaise lockups using MS-SQL Server.

With the lack of traditional methods of communication and no access to a self-contained network infrastructure, addressing and remediating issues encountered in the field posed particularly unique challenges. This dictated that the Quality Assurance (QA) processes for the data collection equipment had to be flawless. In addition, a lab environment replicating the field had to be established for troubleshooting. We will explain in this paper the issues encountered during data collection, troubleshooting in the lab, collaboration with Statistics Netherlands resulting in software releases, and remediation plans.

2 Systems Architecture

The data collection system architecture for 300 concurrent participants, which was the maximum number per session, is illustrated in the diagram below. After extensive benchmark load testing it was ascertained that the optimal number of users connecting to a server in this architecture was 100 users. This was due to the fact that all participants started the survey at precisely the same time causing a peak load condition.



There are **12** different major pieces of equipment in use, not counting the printer, cables, power strips, or power cords! There are 24 points of connection among those 12 pieces of equipment, so that means that there are *at least* 36 different points of potential failure in the hardware setup. Each of these points of failure would need to be tested to eliminate the potential that this was where something had gone wrong. The technical problems are further compounded by the fact that each component in the network architecture plays a pivotal role in a large scale site administration, so that a failure of any one of these points could bring data collection to a halt for 100 survey participants. If this is not enough, all the equipment has to be assembled and then disassembled for each data collection session.

Each of the components had been designed with redundancy in mind so that each had a hot swap duplicate ready if a malfunction occurred.

The points of failure could include network architecture, server configuration, hardware failure, software malfunction, and other unknown variables. For the infrastructure to work correctly the components must follow an established path of communication, which implies that the controller and servers must connect to the switch, and the switch must be connected to the Power Over Ethernet

(POE) devices, and the POE must connect to the Access Point (AP). The Access points will then serve as the broadcast media to the clients and perform bi-directional communication to the servers.

Troubleshooting becomes compounded when there are 300 participants answering questions in a tightly limited time-constrained group survey session, particularly since there is no opportunity to re-take the survey. As such, we had to build in redundancy and countermeasures for every component in the infrastructure. System monitoring tools were utilized to observe the health of the controller and the computers connected to the network. The cabling was color-coded making it easier to swap out backup equipment at a moment's notice in case of any systems failure.

3 Security Requirements

This study required stringent security measures to preserve the integrity of the data. The following security measures were implemented on the data collection servers.

- **Federal Information Processing Standard (FIPS)**:
  System Cryptography: Use FIPS compliant algorithms Enable
  <machineKey validationKey="AutoGenerate,IsolateApps"
  decryptionKey="AutoGenerate,IsolateApps" validation="3DES"
  decryption="3DES"/>

- **Bitlocker**: With USB Key boot up.

- SQL Server **- Transparent Data Encryption (TDE)**

- **Secure File Transfer Protocol (SFTP)** to send/receive data

In addition to system security the equipment had to be supervised during data collection and all the equipment locked in a secure location while not in use at the Army base.
The file transfer server was the only piece of equipment that was authorized to be taken off site to perform a send / receive of the data.

4 Issues and Troubleshooting

Many issues were encountered initially and the problems were assigned to one of the following categories:

- System setup - connecting together all the components, location of Access Point (AP)
- Hardware failure - Server, Client Laptop, Controller, POE, Switch, AP
- Configuration - Server, Client Laptop, Controller, AP
- Software Failure - SQL Server, Blaise IS, IIS, Operating System, Firefox, etc.
- Other - Military signal scrambling, solar storms!!!

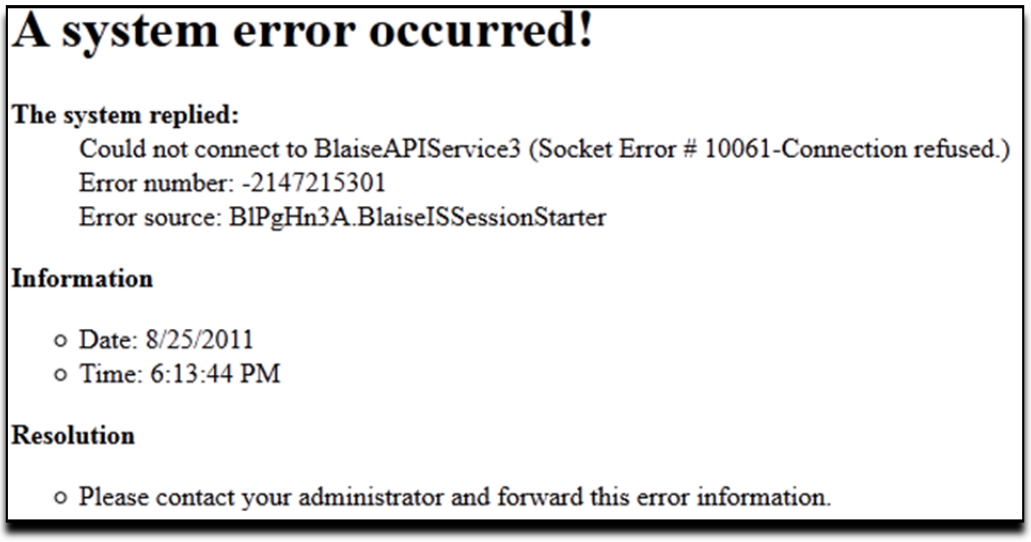
The focus in the following sections is primarily on the troubleshooting the Blaise issues and neurocognitive test.

4.1 BDB vs SQL Server

Concurrent benchmark testing was performed using both BDB's and MS-SQL server. This was conducted to isolate the issues related only to Blaise, and to verify that none of the other numerous components were inadvertently causing the Blaise lockups. It was observed that while using BDB's, no Blaise lockups were encountered due to the Data Link and performance was better than using SQL Server with Blaise.

4.2 Blaise API (Data Link) Issues

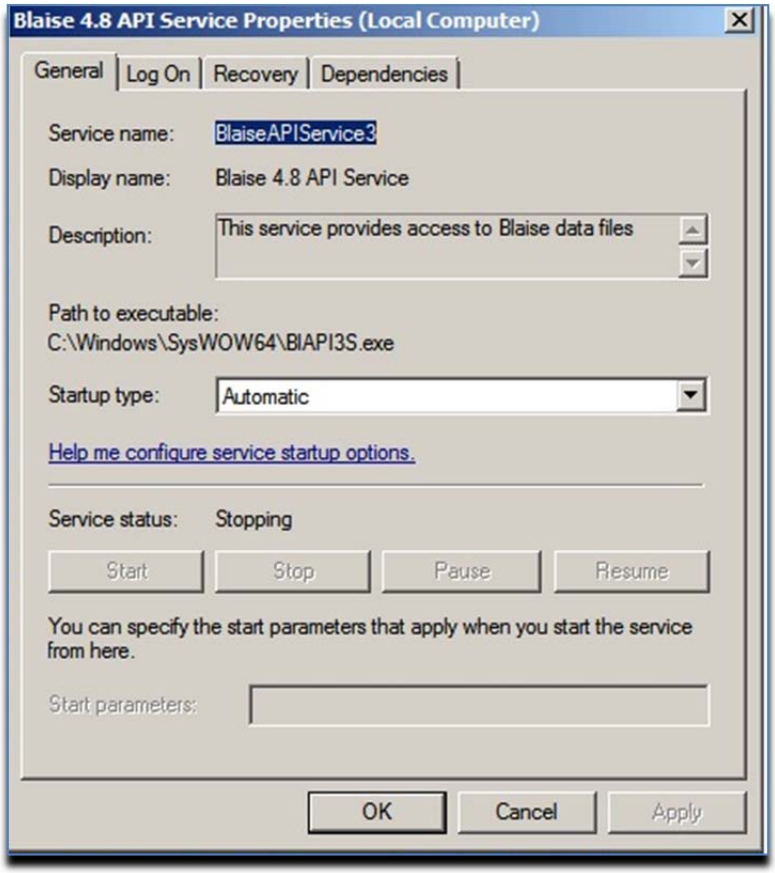
How did we discover it was Blaise API? On the client machine an error was received in the browser similar to the one shown below.



4.2.1 Blaise API Service

The error above would appear simultaneously for all users, usually 100, connected to a particular server. By reviewing the client machine, we could determine very quickly which server had the issue. The first step was to look to see if the Blaise API service was still running on the server. If it was running, then the assumption was that the Blaise API service had an internal problem and was hanging. After restarting the service we noticed that this resolved the issue in most cases. However, in some instances the Blaise API service failed to restart and the message below was displayed:

Following this error, the ability to Start or Stop the service is no longer available. The solution was to open the task manager and end the process for BIAPI3S.exe and then restart the Blaise API service.

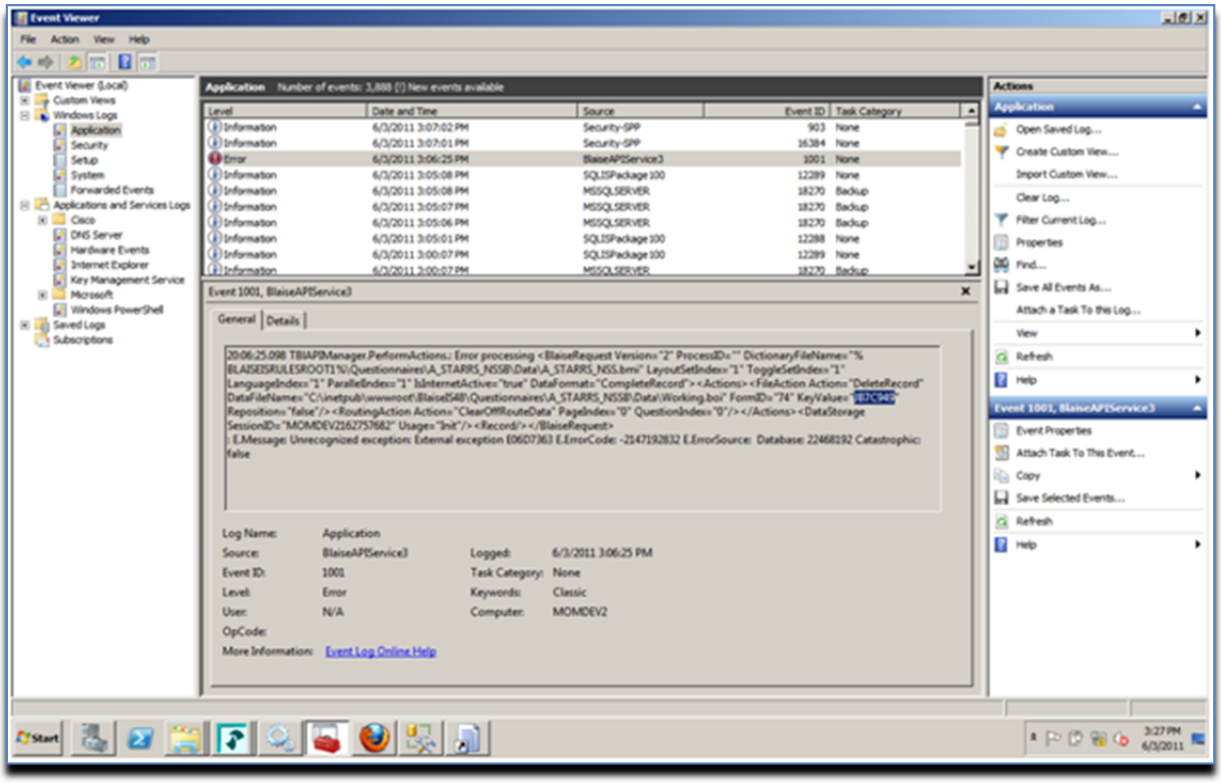


NOTE: By restarting the Blaise API service the session data disappears, but the browser still has a valid asp session. By pressing F5 the new interview page is determined by the API service with an empty session so it returns to the first page. This new empty session has an empty key value. Therefore the save actions to the working database fail.

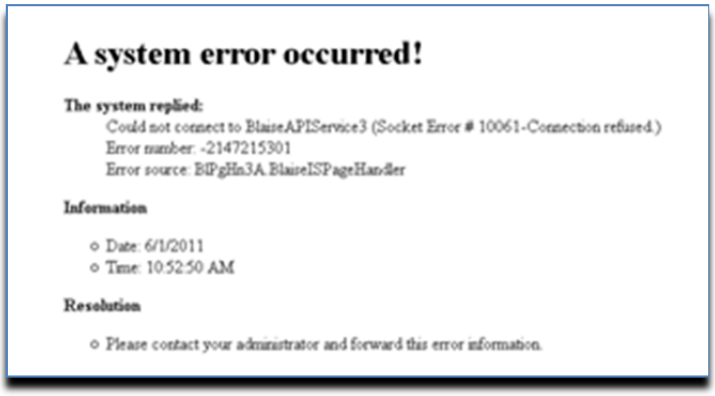
Most errors related to the Blaise API service were similar to the one above and multiple case studies and corresponding Windows Logs and Blaise Garbage logs were shared with Statistics Netherland.

4.3 More Troubleshooting with the Blaise API Service

We observed that after stopping the Blaise API service during the course of a test, the Neurocognitive (NC) tests continued. In the Event viewer we noticed that there was a BlaiseAPIServie3 error that occurred.



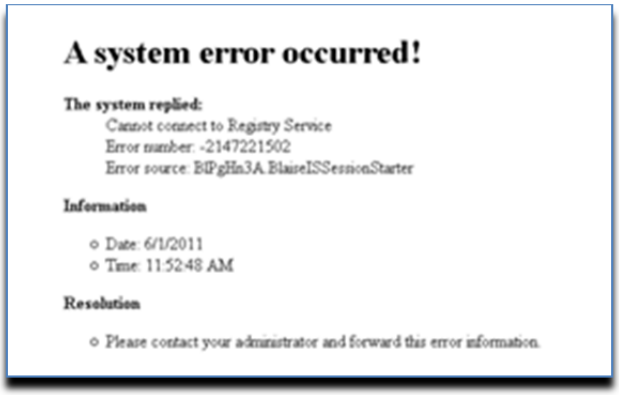
Also the following screen was displayed in the browser:



The Blaise API Service was restarted on the server followed by pressing the F5 refresh key on the client machine. This resulted in another error shown below. We had to shutdown Firefox and restart it and log back in, which allowed the survey to continue.

Next we stopped the Blaise 4.8 Internet Survey Manager Service and we saw the same error as when we stopped the Blaise 4.8 API Service. In addition, stop the Internet Survey Manager Service, causing the Blaise API service to stop as well. No errors were generated by stopping this service in the event log. However, the screen below was displayed on the client machine.

Next we stopped the Blaise 4.8 Registry Service this again caused the Blaise API service to stop; additionally this caused the Blaise Internet Survey Manager Service to stop as well. The following errors were displayed on the client machines.



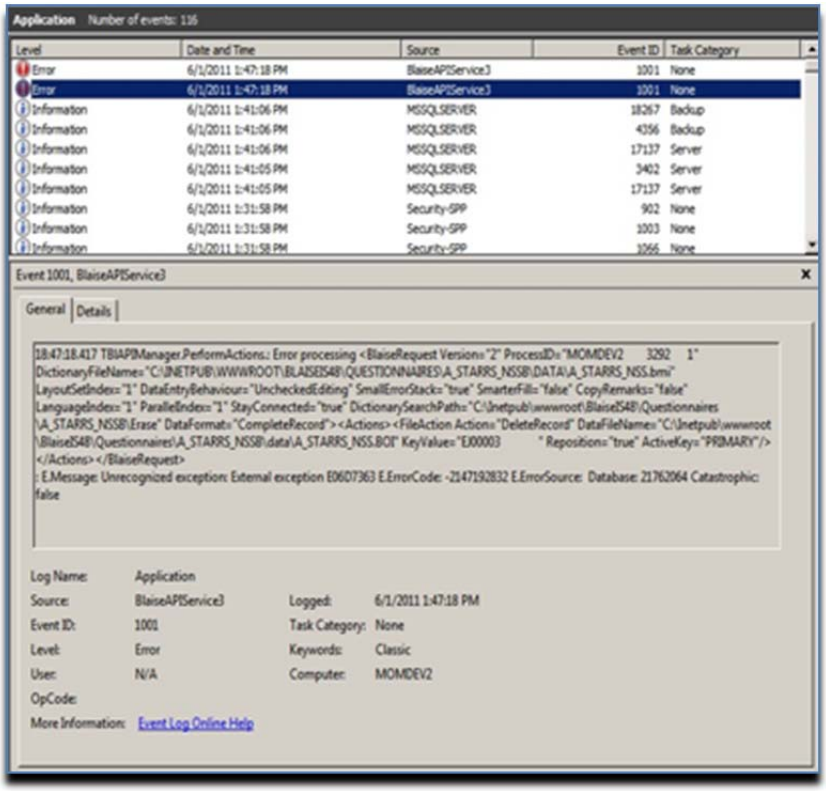
We saw the following error as well on the client machine.

After services were restarted and survey participants were back in the survey, we shut down the Portal Service, Server Pack Service, and the Service Monitor Service which had no adverse effect on the survey.

Many similar battery tests were conducted and the results were shared with Statistics Netherland.

4.4 Data Deletion Roadblock

While troubleshooting Blaise issues we stumbled upon survey data in the windows event log written by the Blaise API service similar to the one shown in the diagram below.



The problem appeared to be happening while data were moving from the working to the main database.

This led us to believe that there was an issue with the Blaise API service deleting records in the working database. To test this theory we added 10,000 rows of data to a test table and wrote a utility using the Blaise API service in manipula to loop through the 10,000 rows and perform consecutive 'ReadNext', 'Delete' statements. We consistently observed that the API service crashed between 400 - 470 rows.

We successfully ran the same utility using BDB without any issues.
1. The import was much faster
2. The delete was faster
3. 10200 cases were deleted without any errors in the event log.
4. Even if the API service is stopped, the BOI deleting script is run just as fast.

From this, we concluded that the Blaise API service had a problem with the BOI database.

This observation was shared with Statistics Netherland and the utility forwarded to them. They were able to replicate the error in their environment and discovered a bug in the DBLink component resulting in a Blaise release.

4.5 BOI File Corruption

On several occasions the Working.BOI file gets corrupted and Blaise is no longer able to access tables for the Working.BOI database. The temporary solution to this issue is to open the main BOI file for that database and rename the tables to WORKING… Subsequently we do a "Save As" and save it as

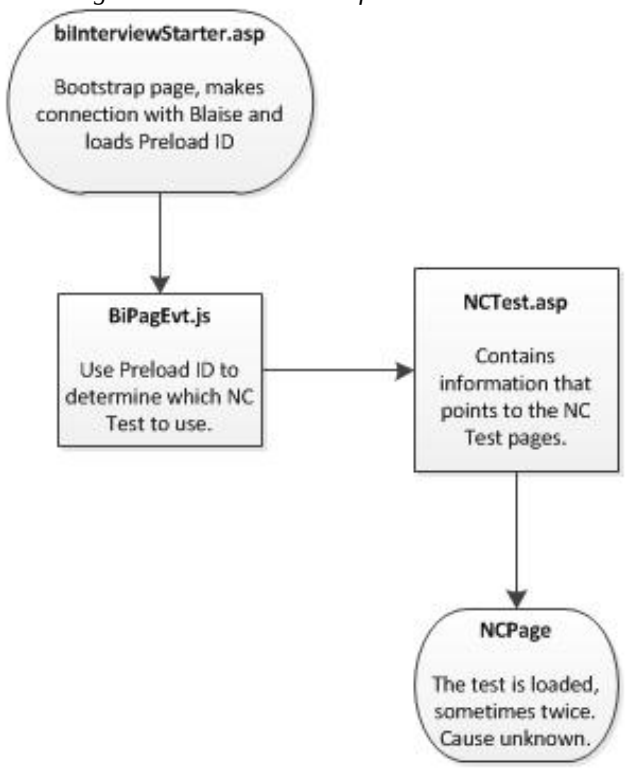
the Working.BOI file. The cause has yet to be determined. One theory is that an old version of the .BOI-file is restored by the Registry Service.

4.6 Duplicate Neurocongnitive Tests

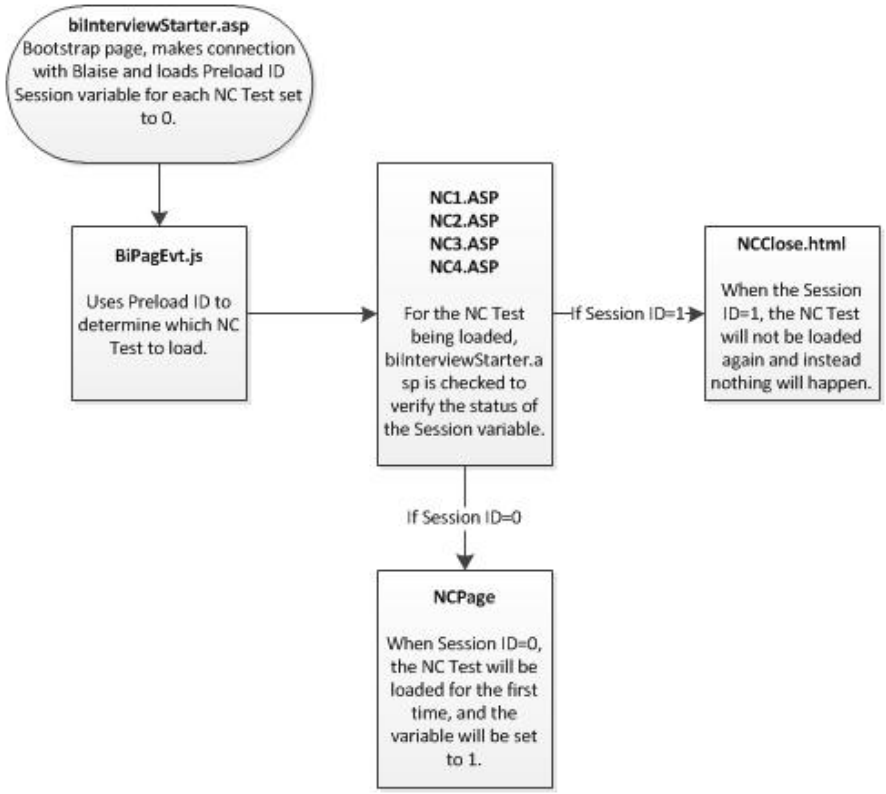
During the course of the survey neurocognitive (NC) tests are administered to the soldier, written in Flash and launched from Blaise IS survey. These test are designed to gather baseline thinking/reasoning/reaction measures. We noticed that, in some cases, two instances of the same NC test would launch when a soldier started the first test. When the first one was completed the second would come to the foreground, and the soldier, naturally, would do the test a second time. This error occurred with an alarming frequency of about 20%.

The code was redesigned to circumvent this error, as diagramed below.

Neurocognitive Test Process - Before



Neurocognitive Test Process –After



4.7 Missing Data

We have cases where the survey data are missing but the paradata are available. Paradata of a survey consists of data about the process by which the survey data is collected.

The survey data can be reconstructed from the paradata. Since the Blaise API service was being utilized to write the paradata from the client to the database, this posed a high risk of complete data loss for some cases due to Blaise API lockup frequency.

To mitigate this risk a paradata backup table, named Paradata_Alternate, was created. There was an addition to the front-end code to also write the paradata to the Alternate table. However, instead of using the Blaise API to write the data, the native Microsoft ActiveX Data Objects (ADO) component was used to write directly to the database. The clear advantage to this approach is that if a Blaise IS service crashes the alternate table will not be affected.

One of our major concerns was what impact this change may have on performance.
Even though we test it in the lab, it's difficult to replicate the exact production environment.
To mitigate the risk we tested this in a small scale field test. We also built in a toggle function to allow the feature to be turned on and off on specific servers. Currently, we still have cases where the survey data are not written by Blaise to the Working or Main database even though the Blaise API service appears to work fine for other users on the same server. We are researching this issue and trying to replicate it in the lab.

5 Quality Assurance

Quality Assurance (QA) is the systematic process of checking to see whether the system being developed is meeting the requirements. Since remote access for troubleshooting in the field was next to none, airtight QA processes were established. This was accomplished by several daily reviews of

the processes and continual refinement. The equipment was configured and rigorously tested in the QA lab before deployment.

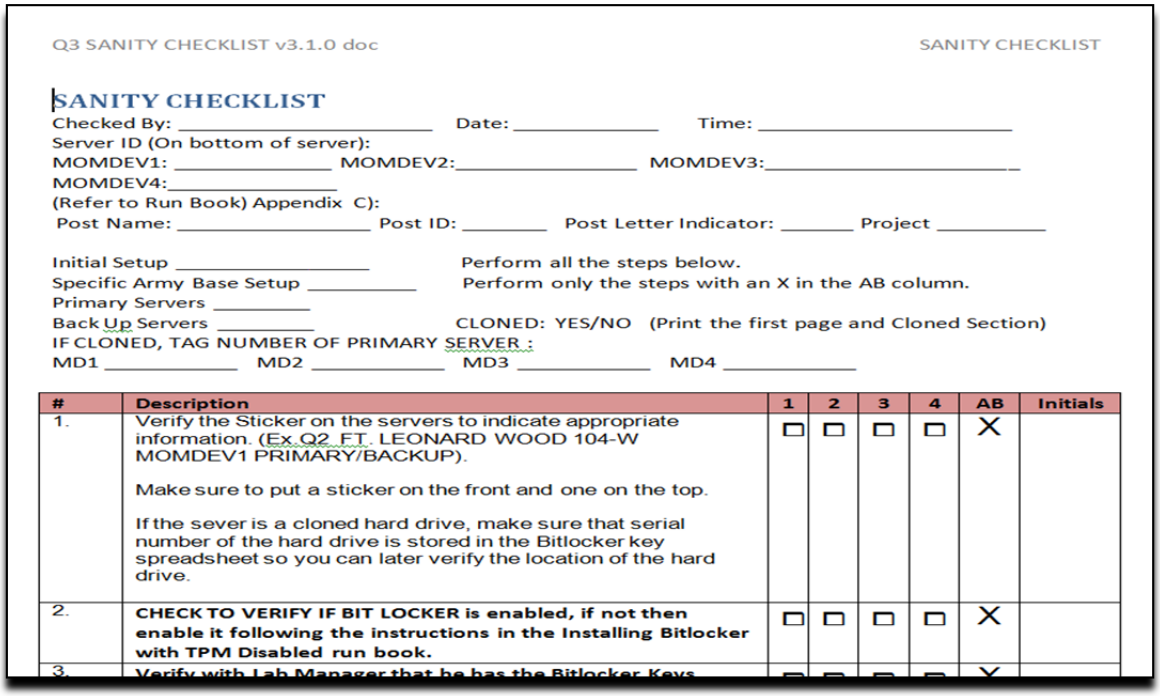
In particular, the servers running Blaise were setup using a very specific set of instructions in order to maintain consistency throughout the project. These installations are then tested by the Business Team, Programmers, Data Managers, and the Lab Technical Support staff. We also used Visual Studio for load testing.

Once all survey and security tests are passed, the servers are imaged to a cloning server to quickly replicate the setup. The efficiency gained by this process to build a QA certified server, was to achieve this in several hours as opposed to 2-3 days. Images of servers deployed in the field were stored in the QA lab and an exact duplicate of the server could be re-created in the lab for troubleshooting.

Sanity check lists were created with sign off from all the teams mentioned above, which established accountability for every piece of equipment that left the lab.

A sanity check is a process that systematically verifies all equipment, configuration, and data components, such as server builds. It encompasses software and hardware design and specifications.

For example, the sanity checklist document for the data collection server consists of approximately 50 steps with each step checked and signed off on for every server ready for shipping to base (see sample below). In this way, a quality control paper audit trail is maintained.



Q3 SANITY CHECKLIST v3.1.0 doc SANITY CHECKLIST

SANITY CHECKLIST

Checked By: _____ Date: _____ Time: _____
Server ID (On bottom of server):
MOMDEV1: _____ MOMDEV2:_____ MOMDEV3:_____
MOMDEV4:_____
(Refer to Run Book) Appendix C):
Post Name: _____ Post ID: _____ Post Letter Indicator: _____ Project _____

Initial Setup _____ Perform all the steps below.
Specific Army Base Setup _____ Perform only the steps with an X in the AB column.
Primary Servers _____
Back Up Servers _____ CLONED: YES/NO (Print the first page and Cloned Section)
IF CLONED, TAG NUMBER OF PRIMARY SERVER :
MD1 _____ MD2 _____ MD3 _____ MD4 _____

#	Description	1	2	3	4	AB	Initials
1.	Verify the Sticker on the servers to indicate appropriate information. (Ex Q2 FT, LEONARD WOOD 104-W MOMDEV1 PRIMARY/BACKUP). Make sure to put a sticker on the front and one on the top. If the sever is a cloned hard drive, make sure that serial number of the hard drive is stored in the Bitlocker key spreadsheet so you can later verify the location of the hard drive.	☐	☐	☐	☐	X	
2.	**CHECK TO VERIFY IF BIT LOCKER is enabled, if not then enable it following the instructions in the Installing Bitlocker with TPM Disabled run book.**	☐	☐	☐	☐	X	
3.	**Verify with Lab Manager that he has the Bitlocker Keys**	☐	☐	☐	☐	X	

6 Summary

Implementing Blaise IS with MS-SQL Server has not been without challenges, but the journey has been extremely fulfilling. The key to success was to establish QA best practices early, create a lab environment replicating production, engaging Statistics Netherland early in the game, and keeping the technical team continuously engaged at University of Michigan.

The following releases have been implemented, some of which were a direct result of the issues encountered.

1. Release 4.8.2.1589
2. Release 4.8.2.1606
3. Release 4.8.2.1618
4. Release 4.8.2.1639
5. Release 4.8.2.1649
6. Release 4.8.2.1653
7. Release 4.8.2.1656
8. Release 4.8.2.1700
9. Release 4.8.3.1717
10. Release 4.8.3.1735 ← The most stable so far
11. Release 4.8.4.1737

Acknowledgments/References

[1]Supported by NIMH U01 MH87981, funds provided by the Department of the Army with supplementary funding by NIMH.

[2]O'Reilly, Jim. Paradata and Blaise: A Review of Recent Applications and Research *International Blaise User Conference 2009, Riga, Latvia*