nmit
Nelson Marlborough Institute of Technology
Te Whare Wānanga o Te Tauihu o Te Waka a Māui

Te Pūkenga

Bachelor of Information Technology

PRJ701

Legacy Weighbridge Software Cloud Integration Upgrade

Mitchell Trow

ID: [redacted]

PRJ701

Bachelor of Information Technology

Nelson Marlborough Institute of Technology

2022

In collaboration with:

tasman district council | Te Kaunihera o te tai o Aorere

nmit Nelson Marlborough Institute of Technology Te Whare Wānanga o Te Tauihu o Te Waka a Māui

Te Pūkenga

ACKNOWLEDGEMENTS

I would like to thank my colleagues and superiors at the Tasman District Council for giving me this project opportunity and being extremely accommodating, especially when it comes to the requirements for the NMIT assignment. Notably: [redacted]

I would also like to thank my NMIT tutor [redacted], and project supervisor [redacted] for being patient with my late project start date and project choice.

ABSTRACT

Exploration of an upgrade solution for a legacy IT software and infrastructure system that creates and manages data for multiple refuse weigh-bridge sites. The existing solution uses an outdated and unsecure version of SSH for file transfer of Microsoft Access database files. The software uses hard-coded, absolute paths for file retrieval and navigation; making solution location specific on the systems it is located on.

Planning for partial or full migration of said software and infrastructure system to a modern Microsoft Azure cloud-based solution, utilising SaaS, and IaaS for a more reliable and secure approach. Modification of Microsoft Access database files to make them location agnostic. Allowing the possibility of a previously fixed system to be located and accessed by Microsoft Azure cloud-based services and authenticated users. Changing the Access file communication method via the internet to a modern and secure method using said Azure solution.

The testing of modifications made to said Access database files and legacy Weighbridge software interactions with Azure cloud-based file storage. Discovering limitations with desired approach and modifications of said approach. Revised planning, testing, and eventual documentation of discovered alternative approaches that still fit with the original project goals.

Creation of a deployment plan, acknowledging potential issues that may arise and considerations made to possible backup solutions. Allocation of time for bug fixing and issue amendment before full deployment.

In the end, conclusions, and reflection of the progress on the project and what changes would be made to the approach given the findings discovered. Also, considerations for how the project could continue going forward to achieve more of the initial goals, given more time.

1. CONTENTS

2. INTRODUCTION

This report covers the planning, research, and work done for the Tasman District Council (TDC) regarding the upgrade of how data is communicated between Refuse Weighbridge sites and the main headquarters and/or remote TDC staff. Eventually, where the main data storage is located. The report will also explore the research undertaken around the different options considered and how/what options were deployed.

3. BACKGROUND

In the past, the TDC used a very centralized and local approach for their IT infrastructure architecture. More recently, there has been a change in upper management, and a "Digital Innovation Program" (DIP) has been created to help guide and streamline plans going forward. As a result, there has been a large upgrade effort to bring legacy hardware and software up to modern standards for reliability, redundancy, and security.

The DIP is an abstract plan with a much larger initiative for improving the technology experience of the entire region with new technologies, reinventing old ones, and expanding its use.

Figure 1: Digital Innovation Program Workstreams diagram (Tasman District Council, 2020)

The DIP is broken down into six workstreams as seen in Figure 1.

As a result of the implementation of the Digital Innovation Program, there has been a need to upgrade and reinvent the Refuse Weighbridge software's communication method and security.

I have been commissioned to investigate, report, propose ideas and then deploy a solution.

3.1. Microsoft Azure

The Tasman District Council makes use of many Microsoft services to manage and operate the organisation. The service that the Information Services department makes use of is called Microsoft Azure.

Azure is Microsoft's cloud computing service that allows customers to take advantage of the many datacentres located around the world. Within Azure itself, it offers hundreds of sub-services. To highlight a few of the most popular ones:

- Virtualized Containers
- Active Directory
- Storage
- Data
- IoT (Internet of Things)
- Power Automate
- Power BI
- Machine Learning

(Microsoft, n.d.-a)

I plan to utilize Azure services in as much capacity as possible with my solutions, as this fits with the TDC's Digital Innovation Program's goals.

3.2. Software as a Service (SaaS)

> Software as a service (or SaaS) is a way of delivering applications over the Internet—as a service. Instead of installing and maintaining software, you simply access it via the Internet, freeing yourself from complex software and hardware management.
>
> SaaS applications are sometimes called Web-based software, on-demand software, or hosted software. Whatever the name, SaaS applications run on a SaaS provider's servers. The provider manages access to the application, including security, availability, and performance.
>
> (Salesforce, n.d.)

The TDC is making more use of SaaS as a part of the Digital Innovation Program. Utilising Microsoft Azure, more of the tasks that would historically be run as proprietary software or local scripts are slowly being migrated to Azure services such as Power Automate and SharePoint.

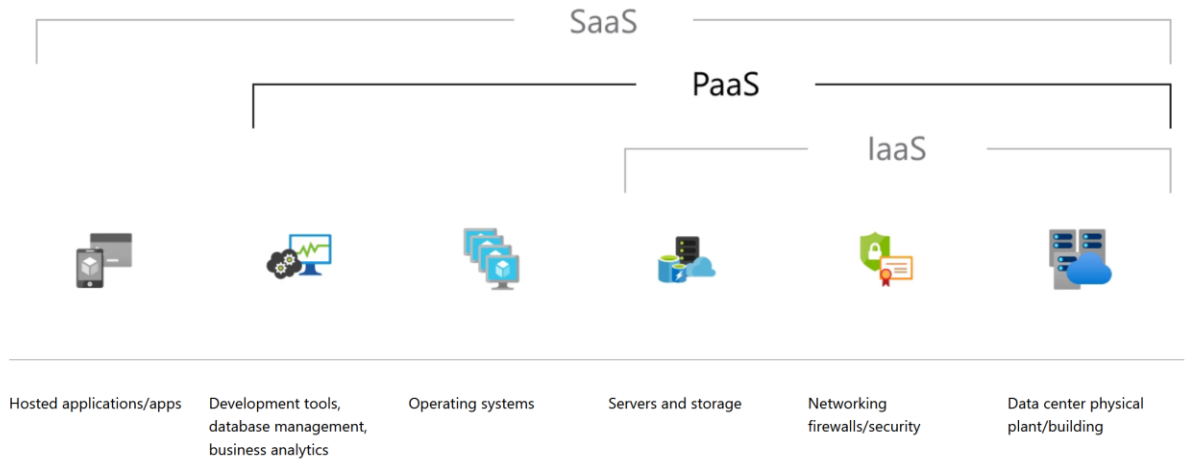


Figure 2: SaaS, PaaS, IaaS relationship diagram (Microsoft, n.d.-c)

3.3. Platform as a service (PaaS)

Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications. You purchase the resources you need from a cloud service provider on a pay-as-you-go basis and access them over a secure Internet connection.

(Microsoft, n.d.-c)

Expanding from SaaS, PaaS is being utilised at the TDC for moving legacy environments to Azure cloud-based services. PaaS is a subset of SaaS as seen in Figure 2, meaning that it uses elements of SaaS to allow integration.

3.4. Infrastructure as a service (IaaS)

Infrastructure as a service (IaaS) is a type of cloud computing service that offers essential compute, storage, and networking resources on demand, on a pay-as-you-go basis. IaaS is one of the four types of cloud services, along with software as a service (SaaS), platform as a service (PaaS), and serverless.

(Microsoft, n.d.-b)

Like PaaS, IaaS is a subset of both SaaS and PaaS (as seen in Figure 2). This is where the TDC is utilizing the hosting services offered by Azure. All the legacy infrastructure is being slowly migrated to Azure.

3.5. Secure Shell Protocol (SSH)

> The Secure Shell (SSH) Protocol is a protocol for secure remote login and other secure network services over an insecure network.
>
> (...)
>
> The SSH protocol consists of three major components: The Transport Layer Protocol provides server authentication, confidentiality, and integrity with perfect forward secrecy. The User Authentication Protocol authenticates the client to the server. The Connection Protocol multiplexes the encrypted tunnel into several logical channels.

(Lonvick & Ylonen, 2006)

3.6. SSH File Transfer Protocol (SFTP)

> SFTP (SSH File Transfer Protocol) is a secure file transfer protocol. It runs over the SSH protocol. It supports the full security and authentication functionality of SSH.
>
> (...)
>
> SFTP also protects against password sniffing and man-in-the-middle attacks. It protects the integrity of the data using encryption and cryptographic hash functions and authenticates both the server and the user.

(ssh.com, n.d.)

3.7. Secure Copy Protocol (SCP)

> scp copies files between hosts on a network. It uses ssh for data transfer, and uses the same authentication and provides the same security as ssh. Unlike rcp, scp will ask for passwords or passphrases if they are needed for authentication.
>
> File names may contain a user and host specification to indicate that the file is to be copied to/from that host. Local file names can be made explicit using absolute or relative pathnames to avoid scp treating file names containing ':' as host specifiers. Copies between two remote hosts are also permitted.
>
> (Rinne & Ylonen, 2013)

As of 2019, Cimpanu (2019) writes that SCP has been plagued with security issues relating to legacy code that cannot be changed due to compatibility issues in RCP.

In April of 2022, the developers of OpenSSH (2022), changed SCP so that it uses SFTP in the background by default.

3.8. Microsoft Access

> Microsoft Access is a well-known database management system produced by Microsoft and is part of the Microsoft 365 office suite. Microsoft Access combines Microsoft's relational Jet Database Engine with software development tools and a graphic user interface (GUI). It was first released in November 1992, so it's been around for a while. In the rapidly changing, fast-paced IT world, we can best describe a 30-year-old program as "venerable."

(Terra, 2022)

Microsoft Access has faced an uncertain future at multiple points in the recent past as Cooper (2019) writes:

> [Microsoft] announced in November 2017, close to the application's 25th birthday, that it intended to retire Microsoft Access from its online productivity suite.
>
> (...)
>
> The official shutdown date for Access Web Apps and Web Databases in Office 365 was set for April 2018. Despite declaring the removal of Access from Office 365, Microsoft quietly changed its mind. Updates continued to appear – the latest version of Access in Office 365 was released in September 2020.

3.9. Visual Basic (VB)

> Visual Basic is an object-oriented programming language developed by Microsoft. It makes it fast and easy to create type-safe .NET apps. Some common uses for Visual Basic are creating Windows-based applications, utilities to perform specific tasks, and adding functionality to existing applications.

(Carnes, 2022)

However Visual Basic is outdated and most implementations are considered "legacy" by Microsoft. "In 2008 Microsoft stopped support for VB and declared it a Legacy software" (Mabbutt, 2019).

3.10. Visual Basic for Applications (VBA)

> VBA is an abbreviation for Visual Basic for Application. VBA is a programming language that was developed by Microsoft Corp., and it is integrated into the major Microsoft Office applications, such as Word, Excel, and Access.
>
> The VBA programming language allows users to access functions beyond what is available in the MS Office applications. Users can also use VBA to customize applications to meet the specific needs of their business, such as creating user-defined functions, automating computer processes, and accessing Windows APIs.

(Schmidt, 2022)

3.11. Batch file (BAT)

A batch file is a script file that stores commands to be executed in a serial order. It helps automate routine tasks without requiring user input or intervention. Some common applications of batch files include loading programs, running multiple processes or performing repetitive actions in a sequence in the system.

The .bat extension also applies to batch file in the Disk Operating System (DOS). One of the best-known DOS batch files is Autoexec.bat that initializes DOS at system start-up.

(Awati, 2022)

3.12. PowerShell (posh) (ps1)

PowerShell is an object-oriented automation engine and scripting language. It is designed mainly for IT professionals and system administrators to control & automate the administration of Windows OS and other applications.

It combines the flexibility of scripting, command-line speed, and the power of a GUI-based admin tool. It allows you to solve problems efficiently by helping system admin to eliminate future manual labour hours.

(Thompson, 2020)

PowerShell has been created as a successor to the BAT script, as it can take advantage of many more features built-in by default. It generally takes much less lines of code to achieve the same result as a BAT script.

3.13. Microsoft Open Database Connectivity (ODBC)

The Microsoft Open Database Connectivity (ODBC) interface is a C programming language interface that makes it possible for applications to access data from a variety of database management systems (DBMSs). ODBC is a low-level, high-performance interface that is designed specifically for relational data stores.

The ODBC interface allows maximum interoperability-an application can access data in diverse DBMSs through a single interface. Moreover, that application will be independent of any DBMS from which it accesses data. Users of the application can add software components called drivers, which interface between an application and a specific DBMS.

(David et al., 2021)

3.14. Virtual Machines

A Virtual Machine (VM) is a compute resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual "guest" machines run on a physical "host" machine. Each virtual machine runs its own operating system and

functions separately from the other VMs, even when they are all running on the same host. This means that, for example, a virtual MacOS virtual machine can run on a physical PC.

(VMware, n.d.)

3.15. Regular Expression (regex)

A regular expression (also called regex or regexp) is a way to describe a pattern. It is used to locate or validate specific (...) patterns of text in a sentence, document, or any other character input.

(Walilko, 2022)

4. PLACEMENT PLAN

The project will take place between the 8th of August and the 4th of November. There is an expectation to work around 300 hours on the project in that timeframe.

During that time, I will follow the following plan:

- Investigation – Figure out how the system works
- Research – Look into alternatives that might work better that fit the scenario
- Planning – Document alternatives to outline the scope and identify workload required
- Testing – Test to see if alternatives work. If the alternative fails, go back to research
- Proposal – Propose changes to colleagues at TDC. If failure, go back to research, planning, or testing
- Documentation, Deployment & Testing Plan– Document changes and how to deploy and test them.
- Deployment – Deploy changes
- Monitoring – Monitor changes to catch any problems that occur and fix them

4.1. Placement Project Goal

The goal of this project is to try to improve and adapt the Weighbridge system to conform with the TDC's DIP goals. In theory, doing this will streamline the maintenance and upkeep of the system as it will be integrated with other systems. Most importantly, it will be somewhat easier to remotely manage.

4.2. Potential Limitations & Issues

Some aspects of the project cannot be changed in the time frame of the project. Majorly, the Weighbridge software itself cannot be changed for an entirely new solution without spending a large amount of time investigating different options and proposing that to the different departments. The behaviour of the Weighbridge software cannot be changed since there is no access to the original source code. All features that can be changed via configuration do not include any in the area that needs to be changed. In a sense, the goal is to change the way a black box behaves without changing the internals.

4.3. Potential Ethical Issues

The most prevalent ethical issue present is that of data privacy of those whose information is entered into the Weighbridge software system. Although the data isn't personally identifiable to any person specifically, it does include a transactional history for vehicles and businesses that have contracts with the TDC. Including, business names, dates & times, waste contents, vehicle registration numbers, and transactional history

5. PLACEMENT REPORT

5.1. Initial Investigation

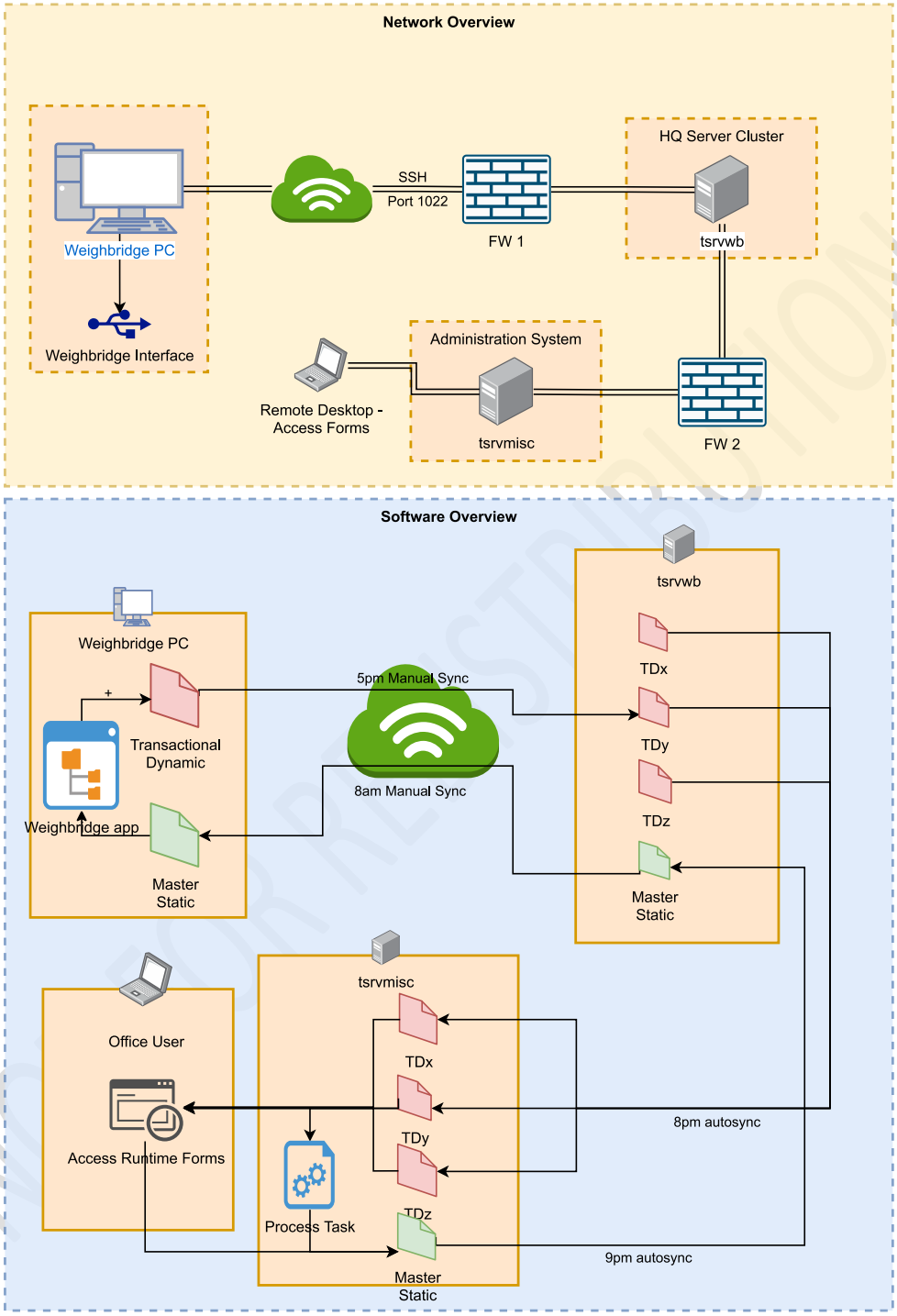


Figure 3: Network and Software diagram showing existing deployment

Through initial meetings with the Information Services staff at the TDC, I created a diagram (Figure 3) outlining what the infrastructure layout currently is and what software operates on it.

The Staff would like their existing Weighbridge software (the software that communicates with the physical weighbridge) to integrate in any capacity with Microsoft Azure services.

As it stands as outlined in Figure 3, the current way data is communicated is via SCP over SSH. It is done manually by the staff at the Weighbridge site, each morning at 8am and each afternoon at 5pm. The files it sends are entire Microsoft Access database files that contain tables that are manipulated and modified by the Weighbridge software.

The files that are transmitted are named "Site Static" and "Site Dynamic". Site Static contains information that stays consistent such as login credentials and contractors with contracts with the TDC. Site Static is never modified by the Weighbridge software. Site Dynamic contains data that the Weighbridge software saves, such as log entries and financial transactions.

When the dynamic database files are sent at 5pm from the weighbridge sites, they are sent to a local server called "tsrvwb" using an outdated version of the software "Putty". Putty is an OpenSSH GUI implementation. "tsrvwb" is a local server running an outdated version of Ubuntu that exists just for file storage.

There is another computer running Windows Server called "tsrvmisc" that runs an automatic task of copying the files from "tsrvwb" each day at 8pm. The computer then runs another automatic task by calling a built-in function in an Access database file called "WFT3000 Admin". The admin database file contains VBA macros and modules which do different functionality depending on what parameters are given to the database file when it is opened. When the automatic task calls the admin database, it does so with the "autoimport" parameter. Doing this will cause a specific VBA module to run which imports all the data from the previously copied files.

The VBA module then processes the data and generates a new static database file (labelled "Process Task in Figure 3). The automatic task mentioned earlier then sends it to the "tsrvwb" file share server. Each Weighbridge software operator then pulls the new static database to their computer at 8am.

The Weighbridge software is deployed on several on-site PCs around the Tasman region. All of them have an internet connection; however, a couple of them have a connection that has a chance of failure for a "brief moment". The software is written in Visual Basic 6, and I estimate it was written in 1992 to run originally on Windows 3.1.

5.2. First round of Research

5.2.1. Potential Change 1

Because the Weighbridge software communicates with and uses Microsoft Access database files for its data transmission and storage method, there is no way to remove Access from the solution entirely. Since I cannot modify the Weighbridge software itself.

Microsoft Access has a feature in which tables can be in other DBMS solutions such as Microsoft SQL Server. In this way, Access can act as a frontend for other databases by utilising ODBC connections to the different data sources.

After Learning about this, I created a diagram to outline how I could implement SQL Server running in the Microsoft Azure cloud in the solution. See Figure 4 below:

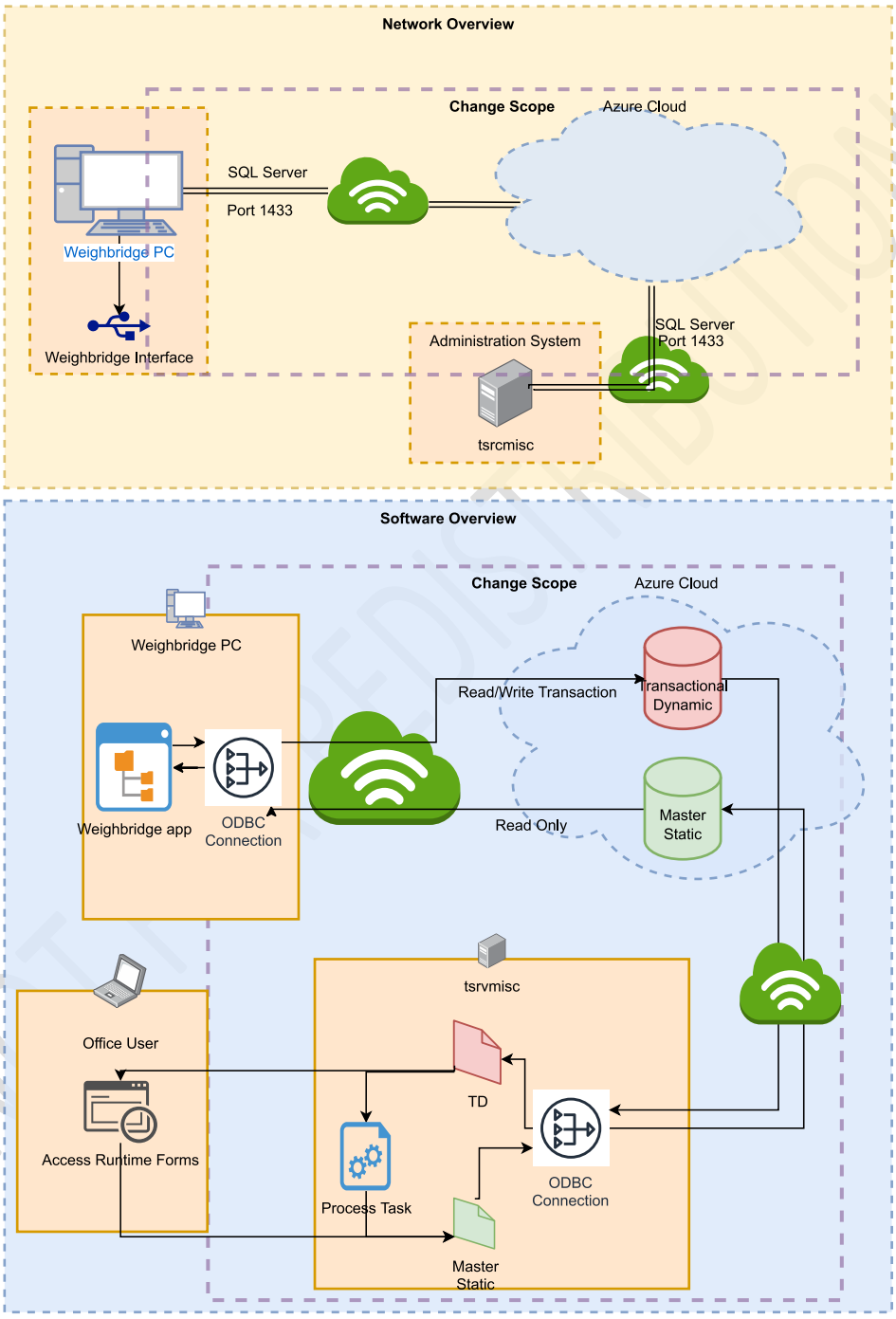


Figure 4: Network and software overview diagram showing how ODBC and SQL Server could be implemented

This fulfils the TDC's DIP goal as it is removing legacy hardware ("tsrvwb") and replacing it with an IaaS service. That being; databases stored in Azure cloud are no longer physical infrastructure, but rather being treated as a service.

5.2.2. Potential Change 2

I also created a backup plan that still utilises SQL server and ODBC in the case that the admin database is only able to communicate with database files that are split up into the different weighbridge sites. This is a hybrid of Potential Change 1 and the original architecture. See Figure 5 below:

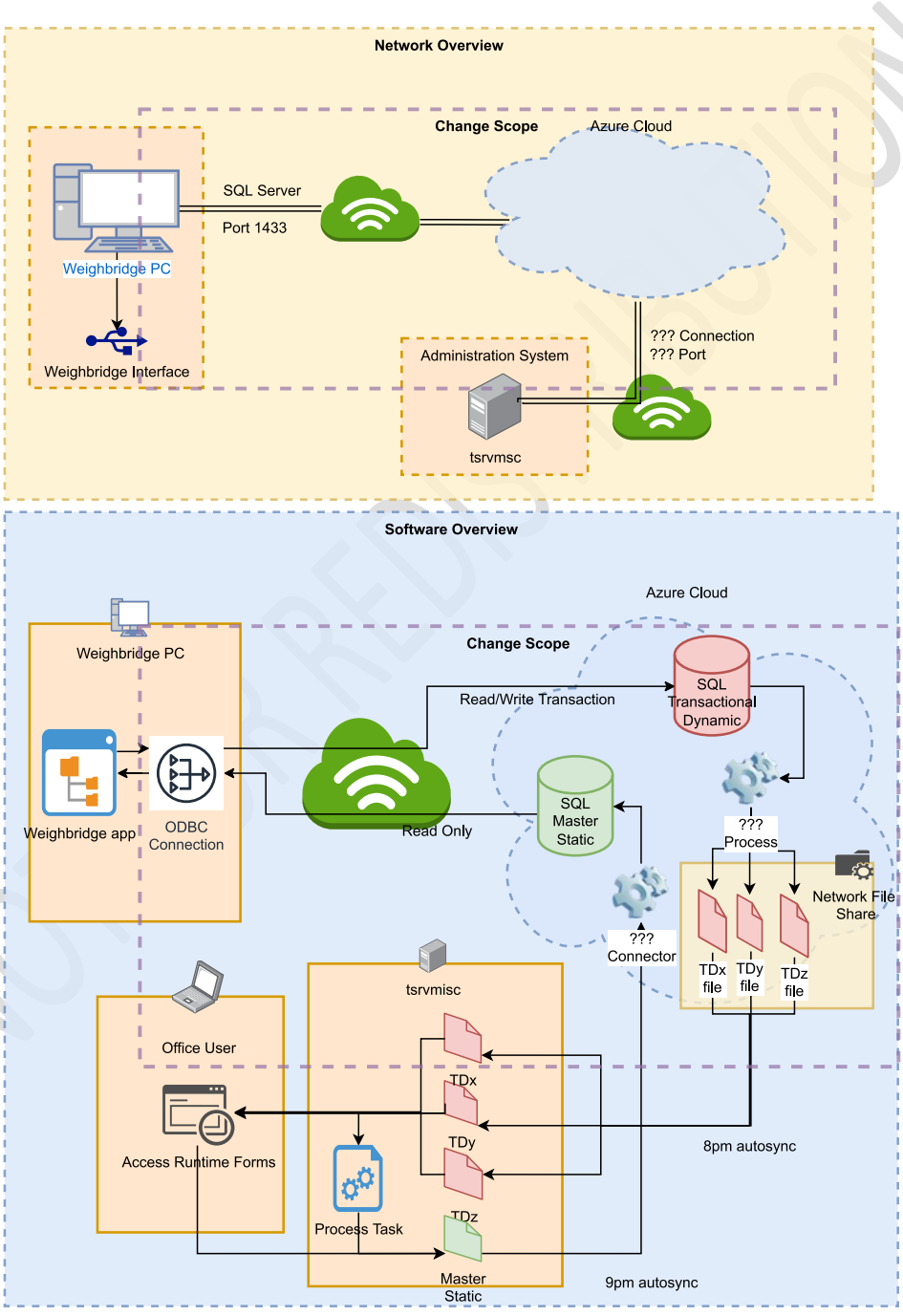


Figure 5: Network and Software overview diagram showing how ODBC and SQL Server could be implemented with legacy considerations

5.2.3. Potential Change 3

For backup plan considerations, I created this change that involves moving the "tsrvwb" file share section to the Azure cloud as an Azure File Storage share. This fulfils the minimal demands of the project in that the data is no longer dependent on outdated software and physical hardware. This could be as simple as changing the SCP "send to" address or changing the transmission method entirely. See Figure 6 below:

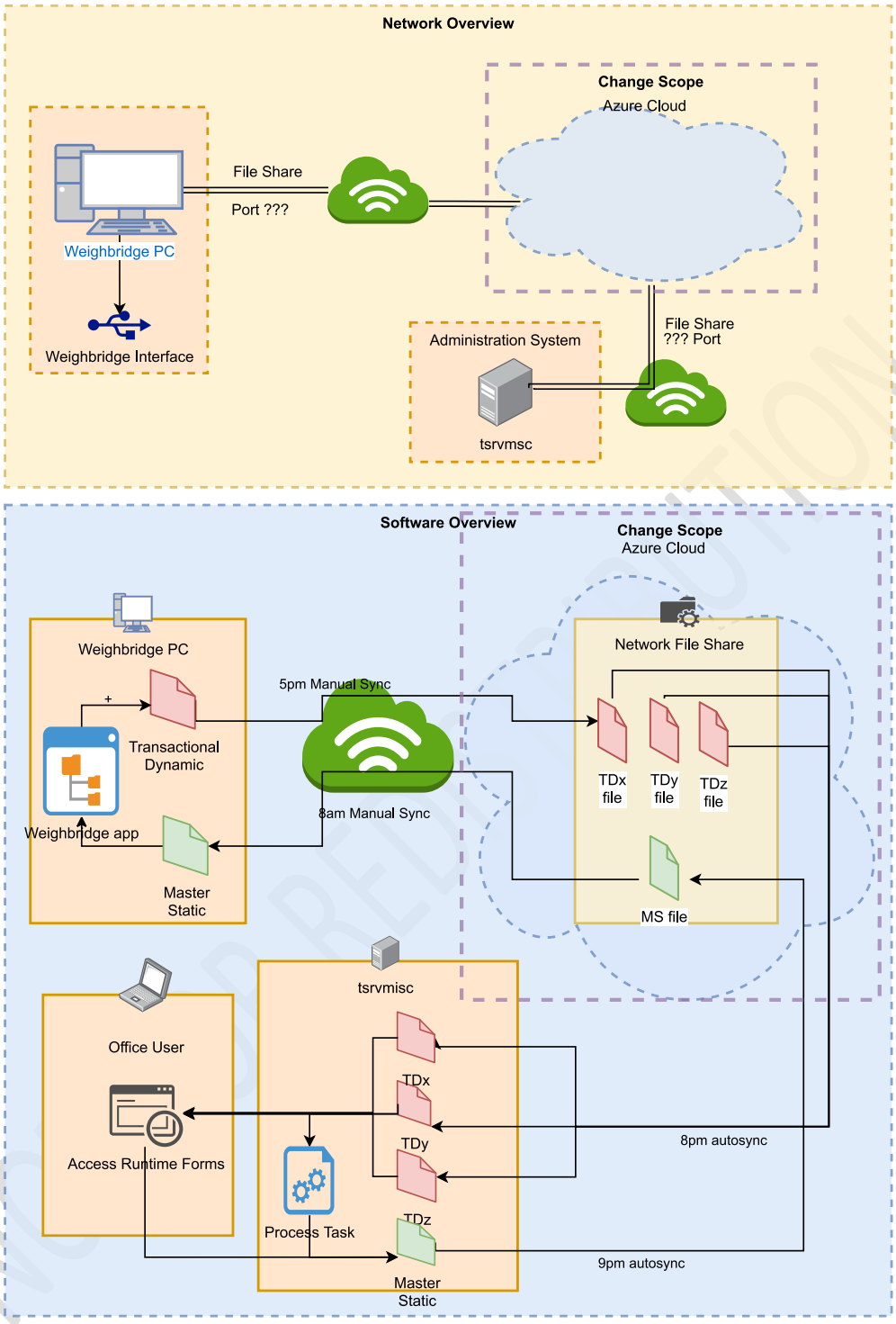


Figure 6: Network and Software overview diagram showing the replacement of "tsrvwb" with an Azure File Storage share

5.3. First round of Testing

I created an environment using Virtual Machines to simulate having a separate weighbridge PC, “tsrvmisc” PC, and a SQL server instance. To do this I used a software called VirtualBox

(Figure 7) in which I created a Windows 11 and a Windows Server 2022 virtual machine (Figure 8).

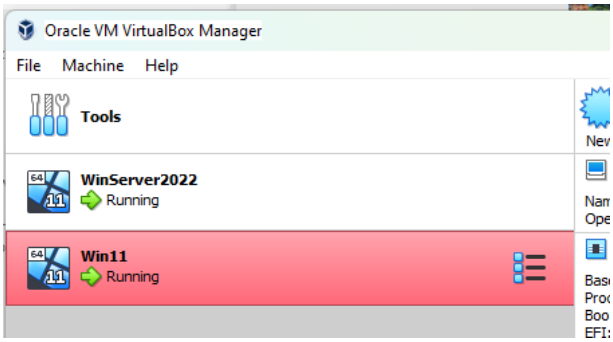


Figure 7: Screenshot of VirtualBox running Windows 11 and Windows Server 2022

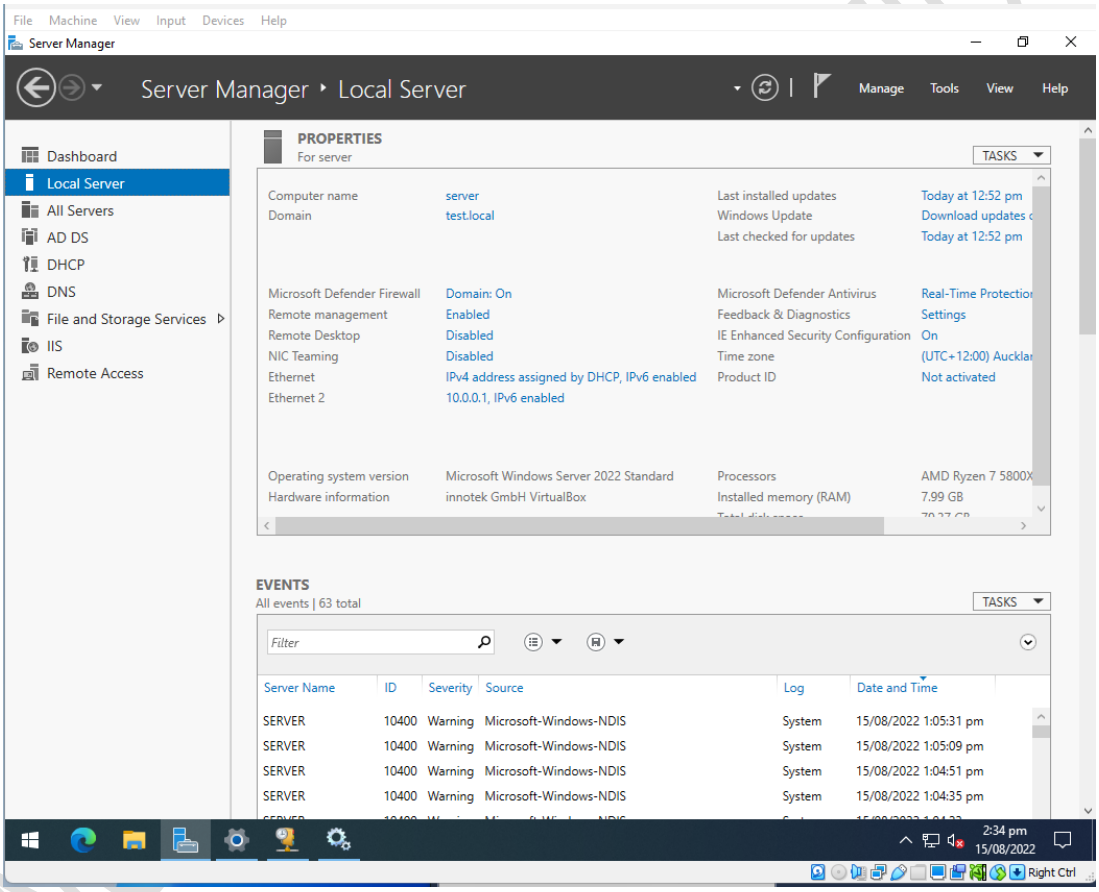


Figure 8: Screenshot of VirtualBox running Windows Server 2022, showing Management GUI

Both virtual machines are networked together on their independent, private network with Windows Server 2022 providing internet access. Windows Server is running SQL Server 2019 and Windows 11 is running the SQL Server Management Studio software (Figure 9).

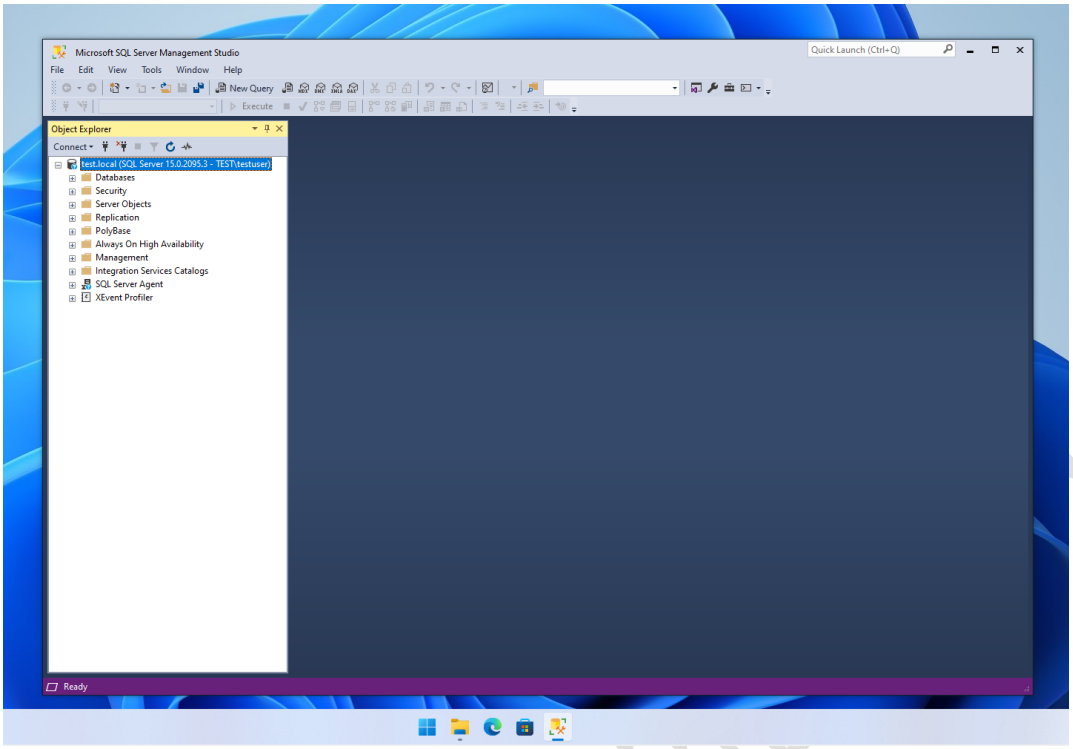


Figure 9: Screenshot of Windows 11 Running SQL Server Management Studio

Using software developed by Microsoft called "SQL Server Migration Assistant for Access" (Figure 10), I can easily migrate the data from both the Site Static and Site Dynamic database files to a SQL Server database instance (Figure 11).

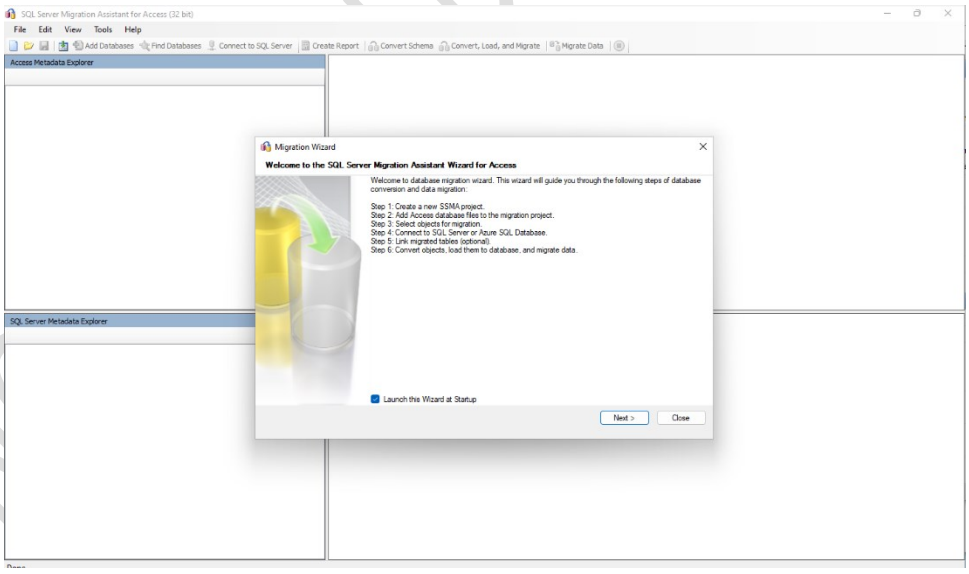


Figure 10: Screenshot of SQL Server Migration Assistant for Access showing Migration Wizard

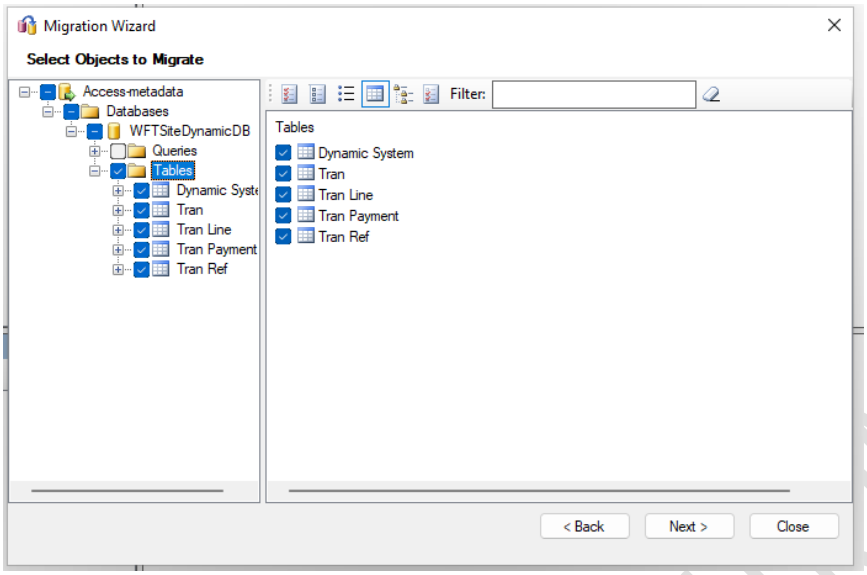


Figure 11: Screenshot of table selection for migrating to SQL Server in Migration Wizard

The Migration Wizard includes an option to automatically link the tables located inside the Access Database files to the SQL server counterparts (Figure 12), as to retain the functionality of the database file except for the data being located elsewhere.

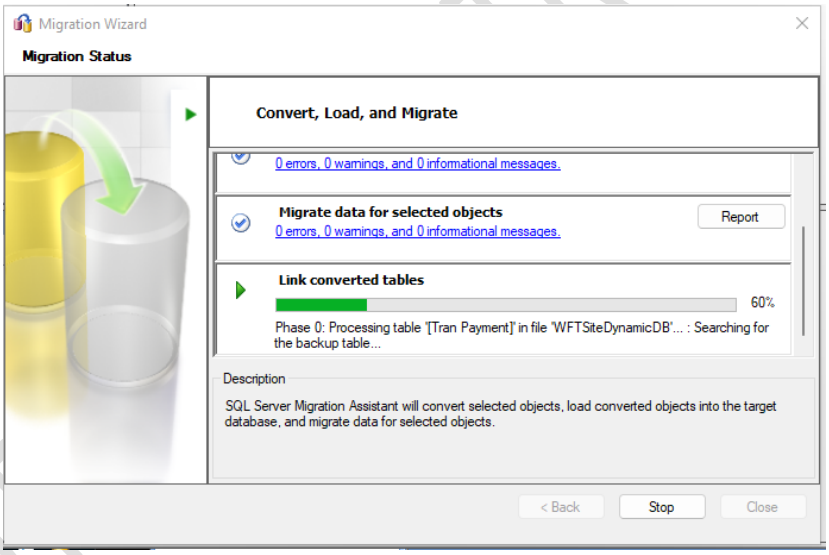


Figure 12: Screenshot of Migration Wizard automatically linking tables back to Access Database file

After the migration completes, the data is in SQL Server (Figure 13) and the tables in Access are linked to it (Figure 14).

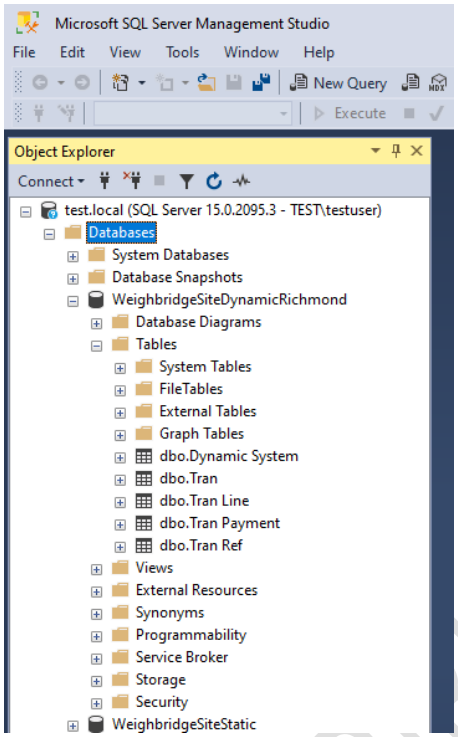


Figure 13: Screenshot of SQL Server Management studio showing Site Dynamic data located in SQL Server

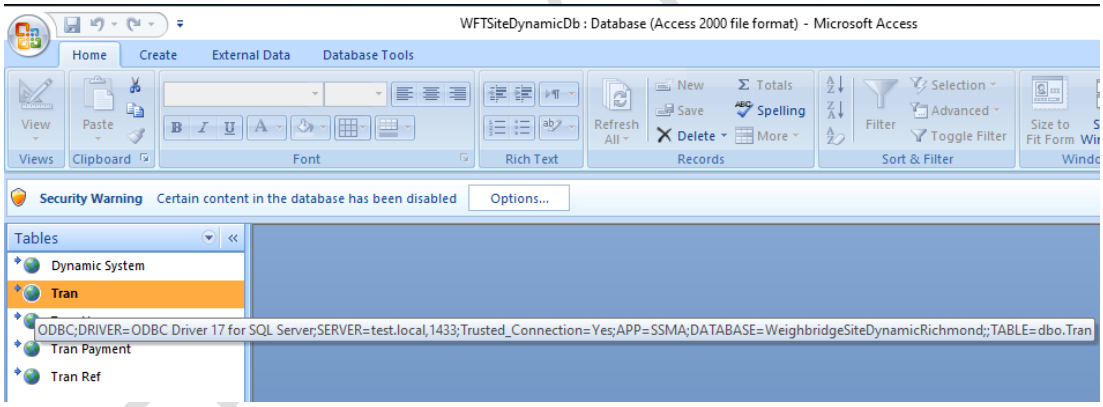


Figure 14: Screenshot of Site Dynamic database file open in Access, displaying tables linking to SQL Server

Placing the "Site Static' and "Site Dynamic" Access database files in the directory in which the Weighbridge software expects them, and then launching the Weighbridge software results in a successful launch (Figure 15).

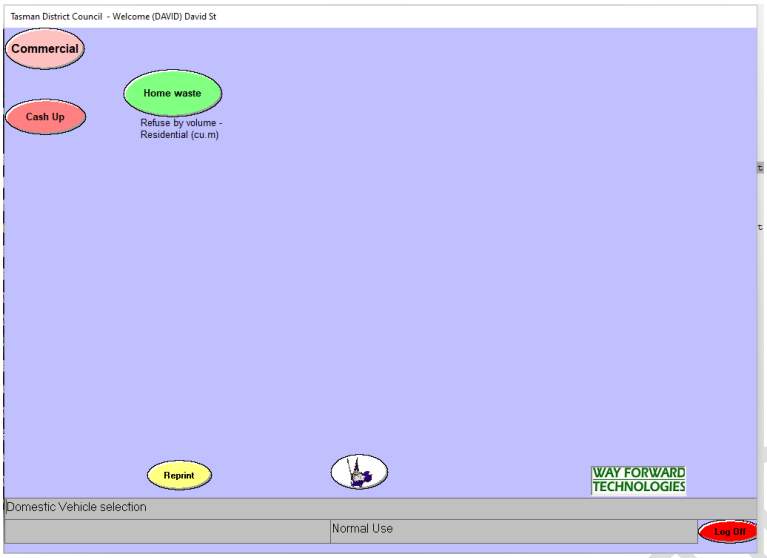


Figure 15: Weighbridge software screenshot showing it launching using linked tables and displaying the home screen

I know that the Weighbridge software is successfully communicating with the linked tables because upon launching because it appears with a login screen that checks the entered credentials against the "Site Static" database file (which is linked to SQL Server).

On further investigation, it appears that some functionality inside the Weighbridge software does not behave as expected. Most of the parts that retrieve and store information work properly, however, some of them do not. Critically the part of the software that displays the different types of refuse on the main screen does not appear to report all the functionality.

Figure 15 shows how the application behaves with linked tables, however, Figure 16 shows how it should behave.

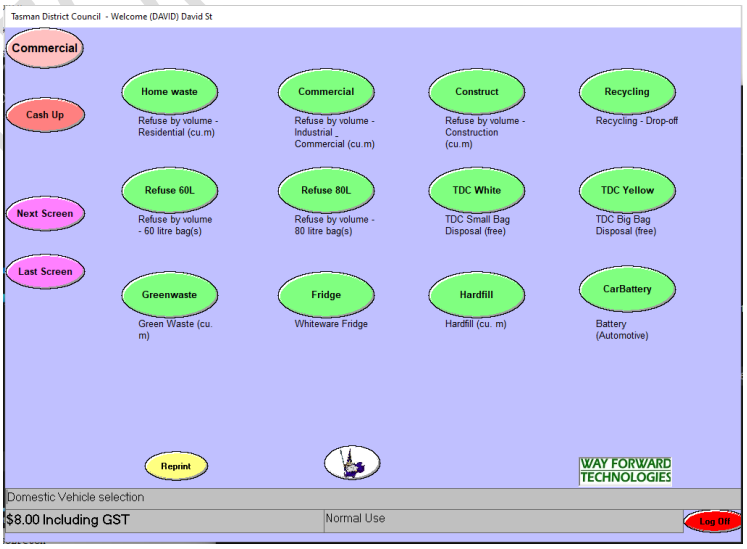


Figure 16: Weighbridge software screenshot showing the home screen with original Access database files

Assuming it was a problem with either the ODBC driver I was using or "SQL Server Migration Assistant for Access"; I have tried to manually link the tables in the Access databases to the remote tables in SQL server using different ODBC drivers.

I have tried the following ODBC drivers: "ODBC Driver 17 for SQL Server", "SQL Server" and "SQL Server Native Client 11.0". All of which yields the same problematic results.

Windows has a built-in ODBC driver admin GUI that contains a "Tracing" feature, in which all actions taken by ODBC drivers are logged to a text file (Figure 17).

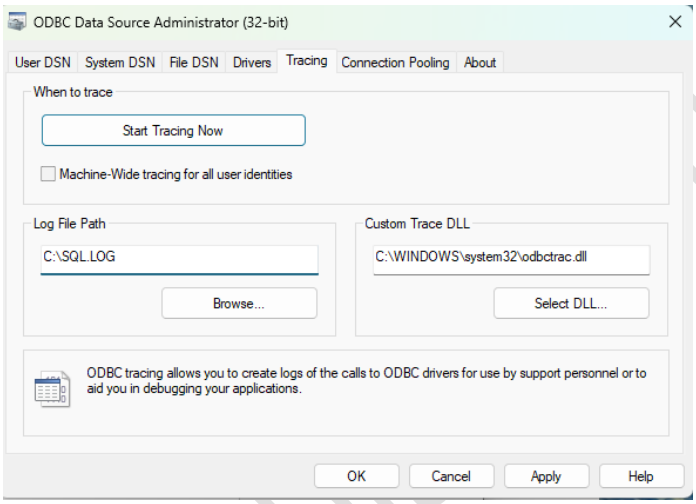


Figure 17: ODBC driver admin GUI screenshot

Using this feature, running the Weighbridge software with linked tables results in a log file that is tens of thousands of lines long. The log file contains all the requests the Weighbridge software makes to the ODBC driver and what data it returns. Around line 300, I can see the request the software makes to retrieve the results to be shown on the home screen and what data it returns (Figure 18).

```
288 WFT System 3000 3408-277c   ENTER SQLExecDirectW
289         HSTMT               0x09356EE8
290         WCHAR *             0x1C2C0FC8 [      -3] "SELECT "Product Id" ,"Product Name" ,"Product Group" ,"Status" ,"Category" ,"Sort Order"  FROM "dbo"."Product" WHERE ((
291         SDWORD                    -3
292
293 WFT System 3000 3408-277c   EXIT  SQLExecDirectW  with return code 0 (SQL_SUCCESS)
294         HSTMT               0x09356EE8
295         WCHAR *             0x1C2C0FC8 [      -3] "SELECT "Product Id" ,"Product Name" ,"Product Group" ,"Status" ,"Category" ,"Sort Order"  FROM "dbo"."Product" WHERE ((
296         SDWORD                    -3
297
298 WFT System 3000 3408-277c   ENTER SQLFetch
299         HSTMT               0x09356EE8
300
301 WFT System 3000 3408-277c   EXIT  SQLFetch  with return code 0 (SQL_SUCCESS)
302         HSTMT               0x09356EE8
303
304 WFT System 3000 3408-277c   ENTER SQLGetData
305         HSTMT               0x09356EE8
306         UWORD                     1
307         SWORD                    -8 <SQL_C_WCHAR>
308         PTR                 0x0019E7AC
309         SQLLEN                  512
310         SQLLEN *            0x0019E754
311
312 WFT System 3000 3408-277c   EXIT  SQLGetData  with return code 0 (SQL_SUCCESS)
313         HSTMT               0x09356EE8
314         UWORD                     1
315         SWORD                    -8 <SQL_C_WCHAR>
316         PTR                 0x0019E7AC [      20] "Home waste"
317         SQLLEN                  512
318         SQLLEN *            0x0019E754 (20)
319
320 WFT System 3000 3408-277c   ENTER SQLGetData
321         HSTMT               0x09356EE8
322         UWORD                     2
323         SWORD                    -8 <SQL_C_WCHAR>
324         PTR                 0x0019E7AC
325         SQLLEN                  512
326         SQLLEN *            0x0019E754
```

Figure 18: ODBC Log file containing requests made to the driver and what is returned

In the log file, it appears that only one row/entry is returned by the ODBC driver. That is named "Home Waste". This lines up with what is displayed on the home screen (Figure 15).

Looking at the table that is stored on SQL Server, shows that it contains 20+ results that should be showing on the home screen like in Figure 16. Namely, shows that "Home Waste" is one of them and it has a "Sort Order" value of 21 (Figure 19). The reason "Sort Order" is important is that it is the first result returned by the query that is called in Figure 18 on line 290 with the ending clause: "**`ORDER BY "dbo"."Product"."Sort Order"`**"

100 %

Results | Messages

	Product Id	Product Name	Product Group	Price per Unit	Charge Type	Minimum Price	Maximum Price	Kg per m3	Dry Weight	Product Notes	Status	Category	Sort Order
1	Commercial	Refuse by volume - Industrial & Commercial (cu.m)	Incoming	82.61	Per Item	13.0435	NULL	0	0	NULL	Active	Both	22
2	Construct	Refuse by volume - Construction (cu.m)	Incoming	82.61	Per Item	13.0435	NULL	0	0	NULL	Active	Both	23
3	Home waste	Refuse by volume - Residential (cu.m)	Incoming	82.61	Per Item	5.22	NULL	0	0	NULL	Active	Both	21

Figure 19: Screenshot of entries in the Products table that is displayed on the Weighbridge software home screen

The SQL Server Management Studio comes with a utility called SQL Server Profiler, which shows the logs and performance metrics of requests made to SQL Server (Figure 20).

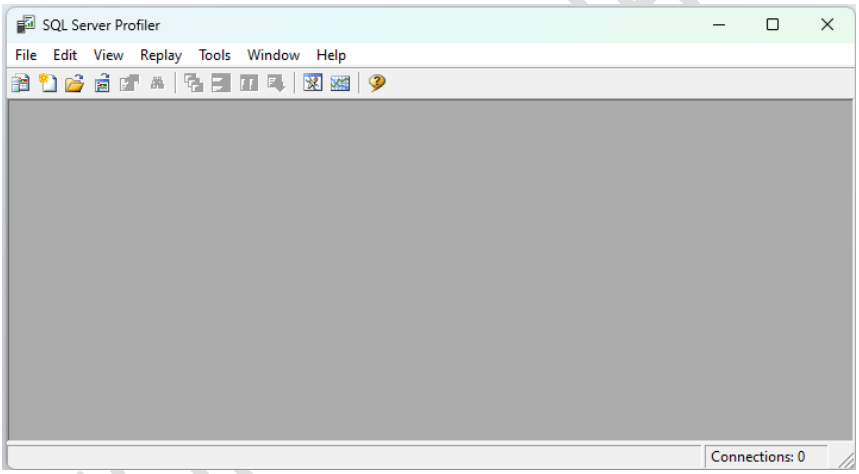


Figure 20: Screenshot of SQL Server profiler

Although Figure 18 shows the query that the Weighbridge Software is asking of the ODBC driver, the ODBC then translates that into a different query that is then sent to the SQL Server. Using the SQL Server profiler, I can see the request made to SQL Server by the ODBC driver (Figure 21).

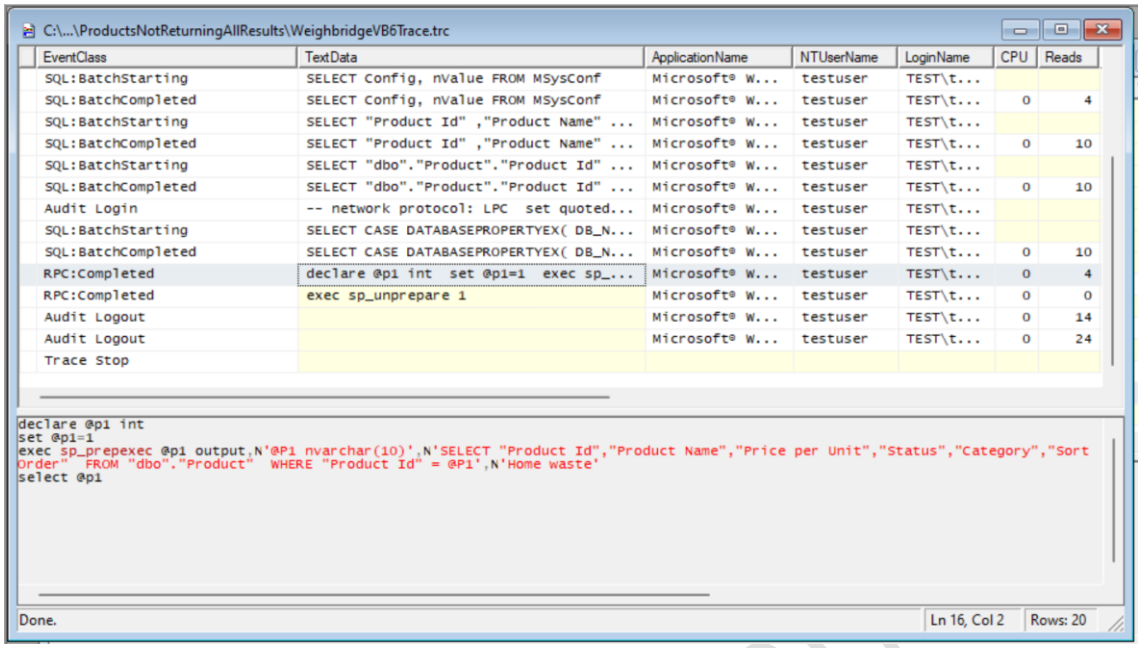


Figure 21: Screenshot of SQL Server profiler showing query sent to SQL Server from the Weighbridge software

As shown in Figure 21, the ODBC driver is creating what is called a "Prepared Statement". As The PHP Group (n.d.) writes:

> Many of the more mature databases support the concept of prepared statements. (...) The query only needs to be [prepared] once, but can be executed multiple times with the same or different parameters. When the query is prepared, the database will analyze, compile and optimize its plan for executing the query. (...) This means that prepared statements use fewer resources and thus run faster.

The prepared statement it creates is only called once, with the criteria of the "Product Id" being "Home Waste". This means only rows containing the "Product Id" being "Home Waste" is returned and since only one row contains that ID, only "Home Waste" is returned.

At this point in time, I am not sure what the problem is. Whether it is a problem with the ODBC driver or the Weighbridge software. Ideally, the Weighbridge software would have no idea it is communicating with SQL Server since it is behind both the ODBC and Access abstraction layer. Meaning, that Access should behave as normal. To test this, I will execute the statement I found in the ODBC log (Figure 18, line 290) inside Microsoft Access itself, using the database file that is linked to SQL Server.

Running the query found in the ODBC log in Microsoft Access (Figure 22), I found that it returns all the expected results as it should. This is also reflected in the SQL Server Profiler log (Figure 23).

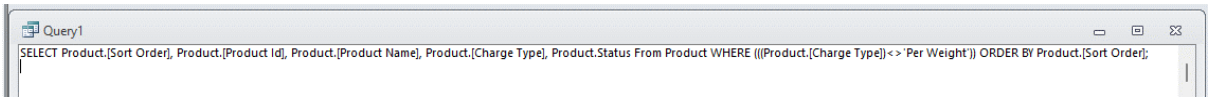


Figure 22: Screenshot of query found in ODBC log inside a Microsoft Access query window

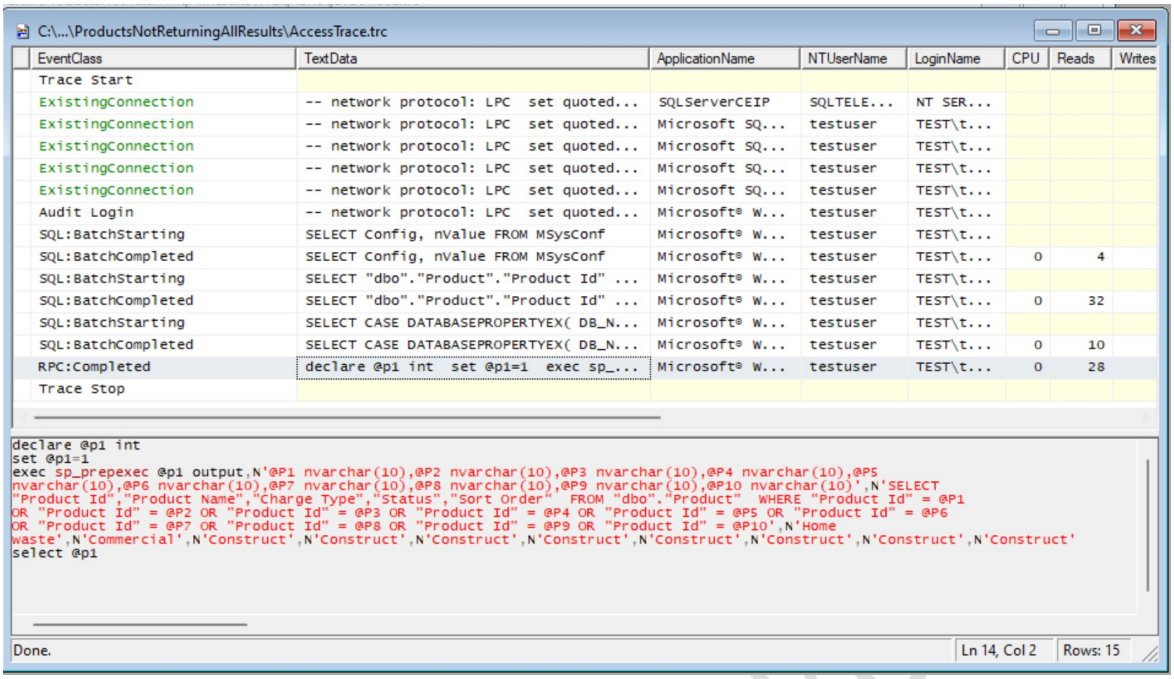


Figure 23: Screenshot of SQL Server profiler showing the query sent to SQL Server from Microsoft Access

As opposed to the query seen in Figure 21, the query in Figure 23 shows that the "Product Id" parameter is set to include all the possible entries in that column. This means that the criteria clause is invalidated and that all the results are allowed to return.

Because Microsoft Access going through a linked table using ODBC results in the correct results being returned, this rules out these two as being the culprit of the issue.

Knowing that I have two backup options, plenty of time left, and a determination to find the issue. I decided to try and find the issue, hoping that it was a simple parameter change I could do in Access, ODBC admin, or SQL Server. To find out how the Weighbridge software was behaving exactly, I decided to decompile the software.

As Black (2022) writes:

> To decompile software means to take that software and change the language it is written in to something that is more understandable to humans. It does this by taking the language of the original program and changing it into a source code that can be better understood. If done correctly, when a person attempts to decompile software, he or she can create a program that acts very similar, if not exactly like, the original software.

I found a piece of software called "VB Decompiler" (Figure 24) which allows me to decompile the Weighbridge software and get an understanding of how exactly behaves.

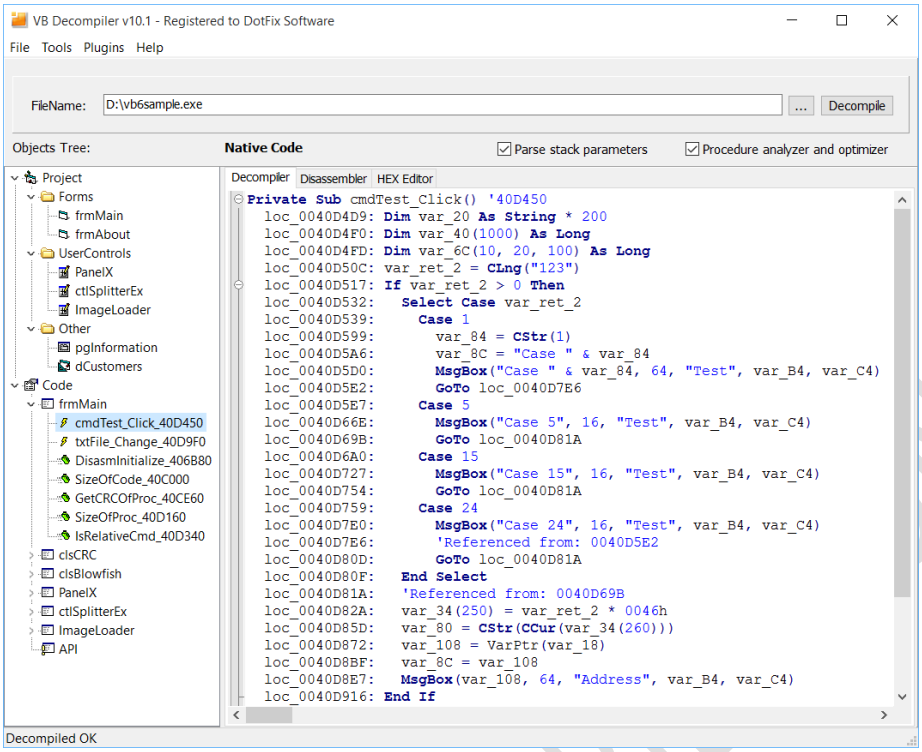


Figure 24: Screenshot of VB Decompiler software showing a decompiled piece of Visual Basic code (*VB Decompiler - Products - Visual Basic 6.0 Decompiler and .NET Disassembler*, n.d.)

Although you can decompile code back to the original language it was created in, it isn't very nice to read for humans. Many of the human-readable elements are missing as the machine does not need to include them. So, to save space, all the elements that help humans read code is removed. As Jaffe et al. (2018) explains:

> When code is compiled, information is lost, including some of the structure of the original source code as well as local identifier names. Existing decompilers can reconstruct much of the original source code, but typically use meaningless placeholder variables for identifier names. Using variable names which are more natural in the given context can make the code much easier to interpret, despite the fact that variable names have no effect on the execution of the program. In theory, it is impossible to recover the original identifier names since that information has been lost

After decompiling and digging inside the decompiled Weighbridge software, I found the location of the SQL query found in Figure 22 inside the code (Figure 25).

```
Public Sub Proc_6_32_5A6F0B
  Dim var_90 As Me
  loc_005A6F4A: call __vbaStrR8(global_700158, ecx, " Commercial:cmdAdd:",
edi, esi, ebx)
  loc_005A6F54: var_34 = __vbaStrR8(global_700158, ecx, "
Commercial:cmdAdd:", edi, esi, ebx)
  loc_005A6F64: var_38 =  &
  loc_005A6F73: var_eax = Proc_2_1_59ECB5(30, 0, global_00434328)
  loc_005A6FAB: If (var_64 = "All") = 0 Then GoTo loc_005A6FDC
  loc_005A6FC1: var_30 = "SELECT Product.[Sort Order], Product.[Product
Id], Product.[Product Name], Product.[Charge Type], Product.Status From
Product WHERE (((Product.[Charge Type])<>'Per Weight')) ORDER BY
Product.[Sort Order];"
  loc_005A6FC6: GoTo loc_005A706E
  loc_005A6FDC: var_30 = "SELECT Product.[Sort Order], Product.[Product
Id], Product.[Product Name], Product.Category, Product.Status, [Product
Group].[Product Group], [Product Group].[Product Type] FROM [Product
Group] INNER JOIN Product ON [Product Group].[Product Group] =
Product.[Product Group] WHERE (((Product.Category)='Commercial') AND
((Product.Status)='Active') AND (([Product Group].[Product
Type])='NonWeighed')) ORDER BY Product.[Sort Order];"
  loc_005A706E: var_90 = CStr(var_30)
  loc_005A70F7: var_90 = (Me.RecordCount > False)
  loc_005A710F: If var_90 = 0 Then GoTo loc_005A71E4
  loc_005A719D: var_eax = frmNonWeigh.Show 1, var_68
  loc_005A71A5: var_94 = frmNonWeigh.Show 1, var_68
  loc_005A71DE: var_eax = Proc_63_4_6C3EA5(var_64, var_60, var_7C)
  loc_005A71E4: Method_8964F04D
  loc_005A7217: Exit Sub
End Sub
```

Figure 25: Code snippet from decompiled Weighbridge software showing where the home screen query is located

This snippet of code isn't very easy to read. However, through cross-referencing the code between different locations where it is pointing to and getting advice from a tutor who used to teach Visual Basic; I was able to determine what exactly is happening here. The query outlined in Figure 22 is chosen as the statement to be used, it then asks Access how many rows (records) will be returned. It then goes into a loop in which it gets the corresponding row and decrements the number of rows left to retrieve. Once there are no rows left to retrieve, it sends the data to the Home Screen code to display it as buttons.

After doing some more research into the issue, I found there is an undocumented difference in behaviour between Microsoft Access and ODBC. According to geekgirlau (2006) on the "vbaexpress.com" forums, when requesting the number of rows using SQL Server, you must move the database cursor to the end before doing it. If you are using Access, it does not seem this requirement is needed.

Wenzel (2021) Explains what a database cursor is:

> A database cursor can be thought of as a pointer to a specific row within a query result. The pointer can be moved from one row to the next. Depending on the type of cursor, you may be even able to move it to the previous row.

According to the Microsoft documentation and geekgirlau (2006), you move the cursor to the end of the table using the command "MoveLast" before requesting the record count. The Microsoft documentation includes a warning about this:

> Note:
>
> *The value of the RecordCount property equals the number of records that have actually been accessed. For example, when you first create a dynaset or snapshot, you have accessed (or visited) only one record. If you check the RecordCount property immediately after creating the dynaset or snapshot (assuming it has at least one record), the value is 1. To visit all the records, use the MoveLast method immediately after opening the Recordset, and then use MoveFirst to return to the first record. This is not done automatically because it may be slow, especially for large result sets.*

(Microsoft, 2021b)

As a result of this finding, there is no way to use the Weighbridge software with an ODBC driver without modifying the original software's source code (which is not possible). This rules out the use of Potential Change 1 (5.2.1) and Potential Change 2 (5.2.2) as they both involve the use of the ODBC driver with the Weighbridge software.

5.4. Second Round of Research

While exploring Potential Change 1 (5.2.1) with Microsoft Access linking tables to SQL server tables, I noticed another way of linking tables in Access; that is to link them to another Access database file.

Expanding on from the Access linked tables idea, I discovered a private peer-to-peer file synchronization software. It is called "Syncthing". Crawford (2019) explains:

> Syncthing is a secure decentralized peer-to-peer (P2P) file synchronization program that can sync files between devices on a local network or over the internet. (...) [Syncthing] is completely free and open source (FOSS).
>
> Syncthing allows you securely to backup data without the need to trust a third-party cloud provider. (...)
>
> This is referred to in techy circles as a "BYO (Cloud) model", where you provide the hardware instead of a third-party commercial vendor. The encryption used is also fully end-to-end, as you encrypt it on your device, and only you can decrypt it. No-one else holds the encryption keys.

I will explore the potential use of Syncthing in one or more of the alternatives.

Because Potential Change 1 (5.2.1) and Potential Change 2 (5.2.2) can no longer work, I will explore sub-options within Potential Change 3 (5.2.3).

5.4.1. Potential Change 3a

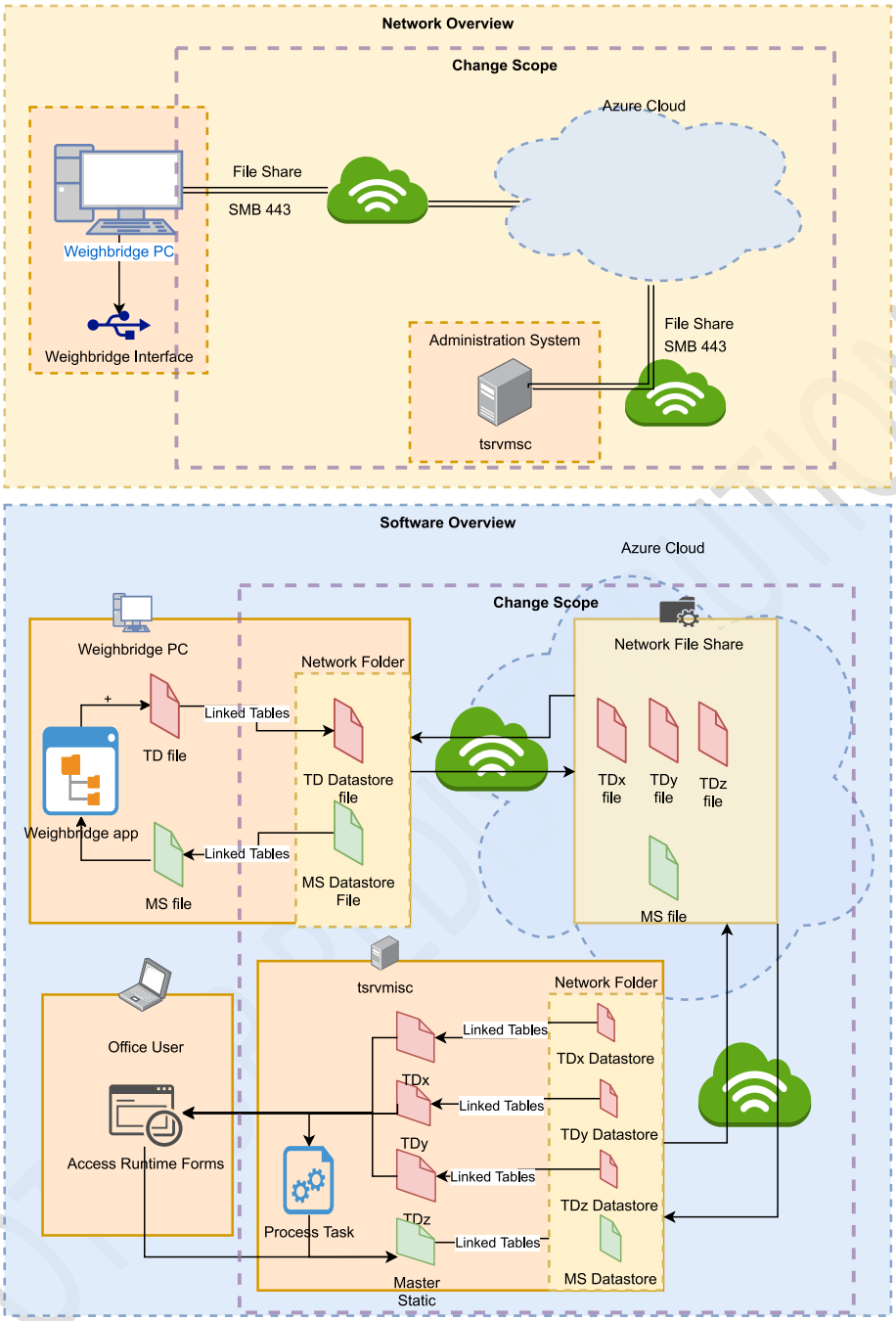


Figure 26: Network and software diagram of using Access linked tables on an Azure File Storage share

This proposal will utilize an Azure File Storage share and mount them as network drives and/or folders in the machines that interact with them. On said share will be Access database files that contain only the tables with data in them. The weighbridge PCs and the "tsrvmisc" PC will have their local Access database tables removed and added back as a link to the data version stored on the Azure share. I will call the remote data database files the "Datastore" to distinguish them as different from those that are local.

The problem with this proposal is that if the internet connection drops even momentarily, the Weighbridge software will stop functioning correctly. Since the data it requires to operate, and store is located on a remote Azure server.

One solution for this is to use Offline Files. It's a feature built-into Windows since Windows 2000. As Brink (2020) writes:

> Offline Files is a feature of Sync Center that makes network files available to a user, even if the network connection to the server is unavailable.
>
> Users can use offline files (if enabled) to make their network files always available offline to keep a copy of the files stored on the network on your computer. This allows users to work with them even when they are not connected to the network or a server is unavailable. The next time the user connects to the network or the server is available, their offline files on your computer will automatically sync to the network files on the server to update.

5.4.2. Potential Change 3b

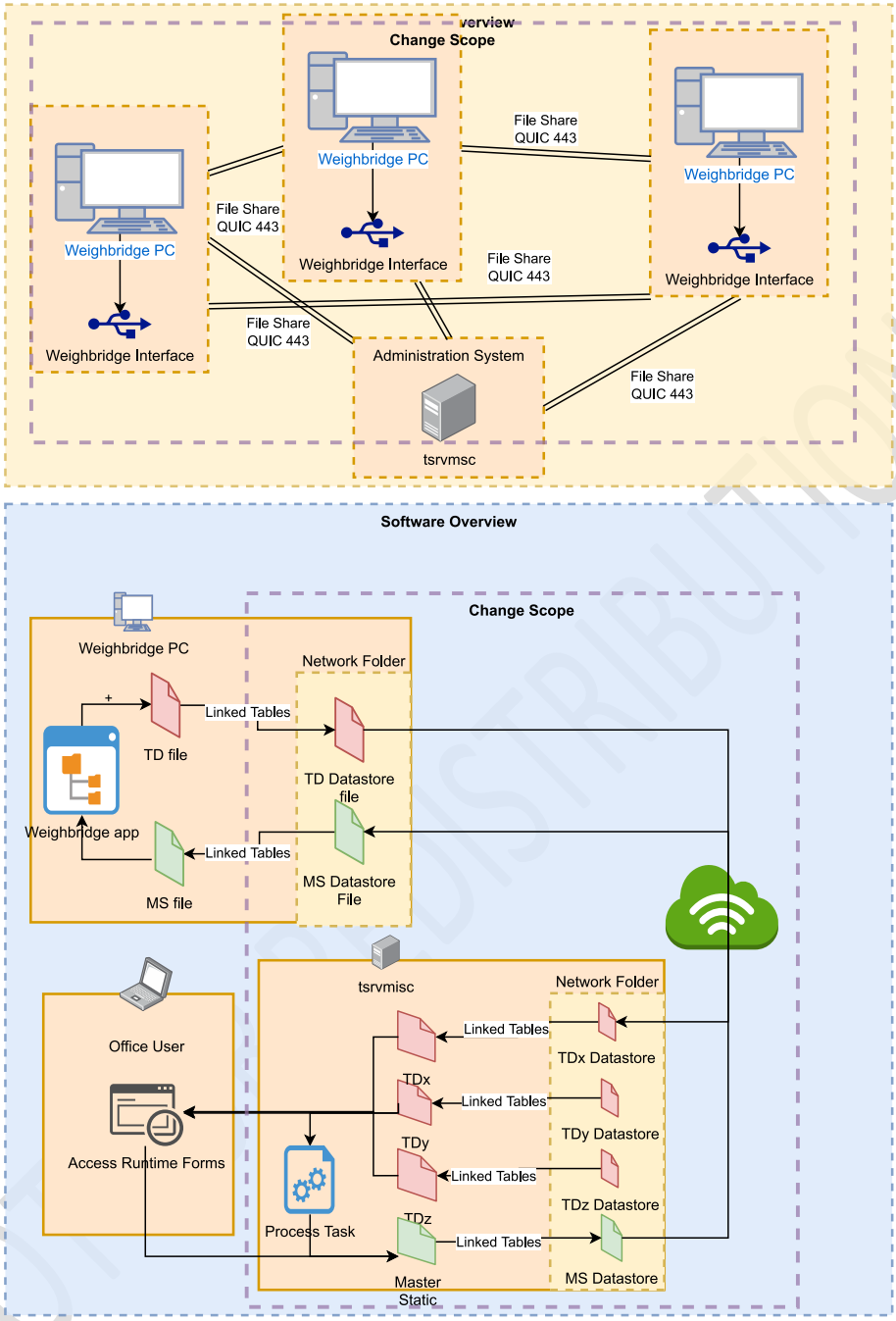


Figure 27: Network and software diagram of using Access linked tables, synchronised by P2P software "Syncthing"

This solution is like 3a (5.4.1) except; it does not involve an Azure share at all. Instead, Syncthing synchronizes the "datastore" files between all the weighbridge PCs and the "tsrvmisc" computer. This solution does not suffer from internet connectivity issues like Potential Change 3a (5.4.1), as the files are located on the local machine and synchronized when connectivity resumes.

5.5. Second round of Testing

5.5.1. Change 3a

I have been given an Azure File Storage network share by my colleagues at the TDC for testing. I have created the Access linked tables setup as described in Potential Change 3a (5.4.1).

With the included functionality of the Offline Files feature inside Windows, I can communicate with the "datastore" files even when the internet connection cuts out. The Offline Files feature is set up on the network share (Figure 28 & Figure 29) I described above, and it is set up to synchronize automatically every 5 minutes.

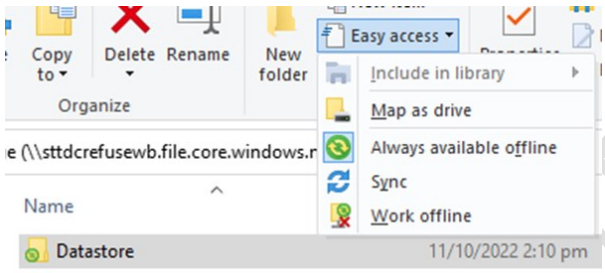


Figure 28: Screenshot showing Offline Files enabled on the Datastore folder

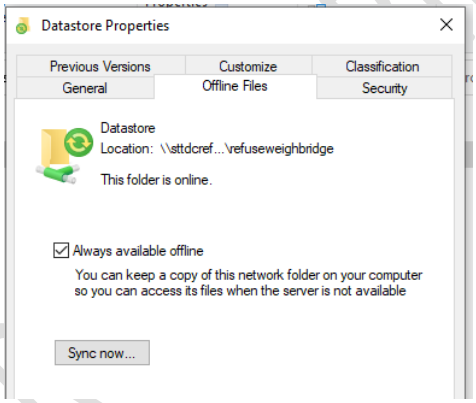


Figure 29: Screenshot showing Offline Files properties window

After setting this up, I create the "Datastore" databases by opening both the "Site Static" and "Site Dynamic" database files in Access. Using the "Split" functionality inside Access; for both databases, I break out the data tables inside them and place them each in a new Access file I end with the suffix "_DS" (DataStore). This is just to help distinguish them from the original database files. The datastore versions are placed in the shared folder attached to windows.

Using this method, the Weighbridge software behaves correctly, displaying entries as they should. As seen in Figure 16.

To test the reliability of the connection and ensure there is no data corruption or errors with the Weighbridge software, I used a utility called clumsy (Figure 30).

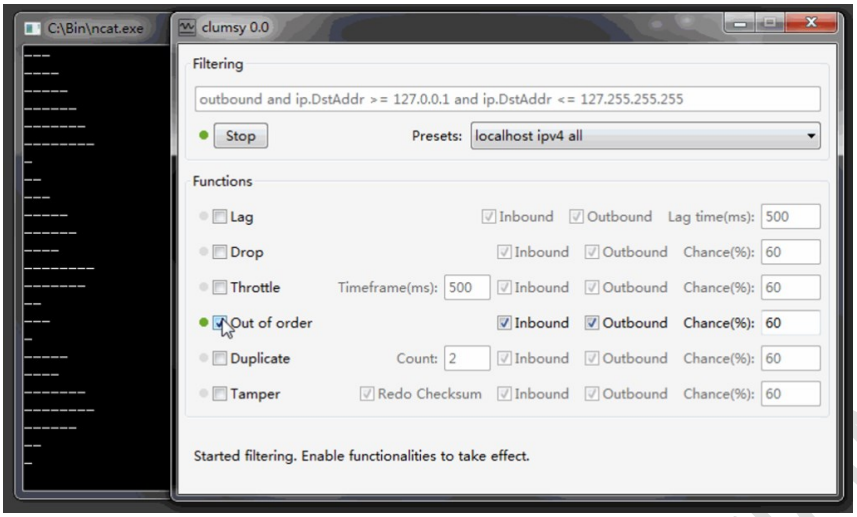


Figure 30: Screenshot of Clumsy (Tao, 2013/2022)

clumsy makes your network condition on Windows significantly worse, but in a managed and interactive manner.

(Tao, 2013/2022)

Using Clumsy, I can tamper and cause errors in the network connection. This in theory will affect the reliability of communicating with the Azure File Storage shared folder, and in turn, affect the Weighbridge software's connection to the datastore.

After testing the Weighbridge software with clumsy running, I am still able to access the web, albeit slowly and with security errors. However, the Weighbridge software is working completely fine. This is due to the Offline files serving the software the cached version, which is stored locally.

5.5.2. Change 3b

Using the same database split and "Datastore" method outlined in 3a (5.5.1); I will test the effectiveness of Syncthing as a solution. Using the same Virtual Machine setup seen in Figure 7, I set up Syncthing on both machines (Figure 31).

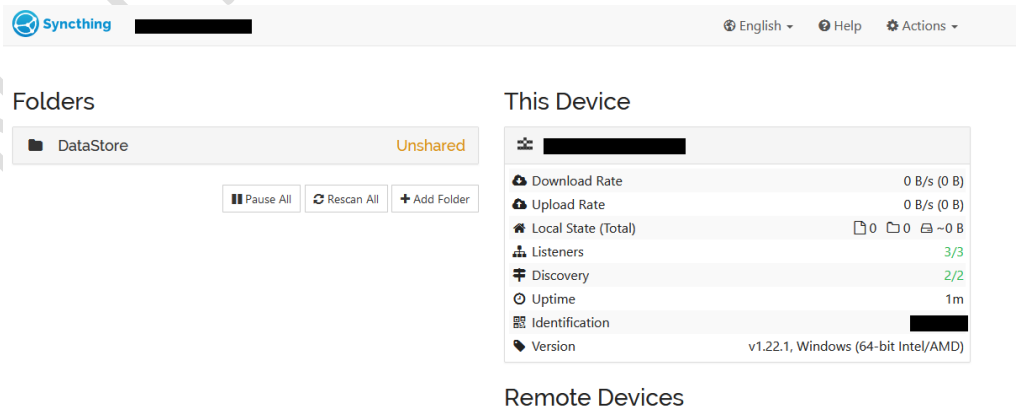


Figure 31: Screenshot of Syncthing web GUI

Syncthing does not work on a schedule like Offline Files; but rather monitors a function called "FindFirstChangeNotification" in the Windows programming API (application programming interface) (Microsoft, 2021a). This lets the program know when a file is modified or saved. So Syncthing only sends out changes and receives changes when the file is modified. Microsoft Access only triggers this when the application closes. Because the files are stored locally and synchronised, there is no point in testing the network using clumsy to see if the Weighbridge software behaves as expected.

5.6. First proposal

I presented options 3a (5.5.1 & 5.4.1) and 3b (5.5.2 & 5.4.2) to my colleagues at the TDC and demonstrated them in operation. They were very pleased with option 3a (5.5.1 & 5.4.1) however they were not as enthusiastic as option 3b (5.5.2 & 5.4.2). They did not like it because it does not align with their infrastructure goals and existing servicing standards.

They pointed out that I still need to make sure that the administration Access database file works with my proposed changes.

They also suggested that perhaps given enough time left, I could modify option 3a (5.4.1) to remove the "tsrvmisc" PC entirely, instead, have it be in a virtual machine on the Azure cloud service. This fulfils more of the TDC's DIP goals, as the "tsrvmisc" machine is replaced by an IaaS solution.

Potentially the machine could be scheduled to start up, run the "Process Task" on the database site files, and then terminate itself again. Removing the need for a machine to be running all the time. Thus, saving money.

Below is a diagram of the hypothetical implementation (Figure 32), known as "Option 3ab":

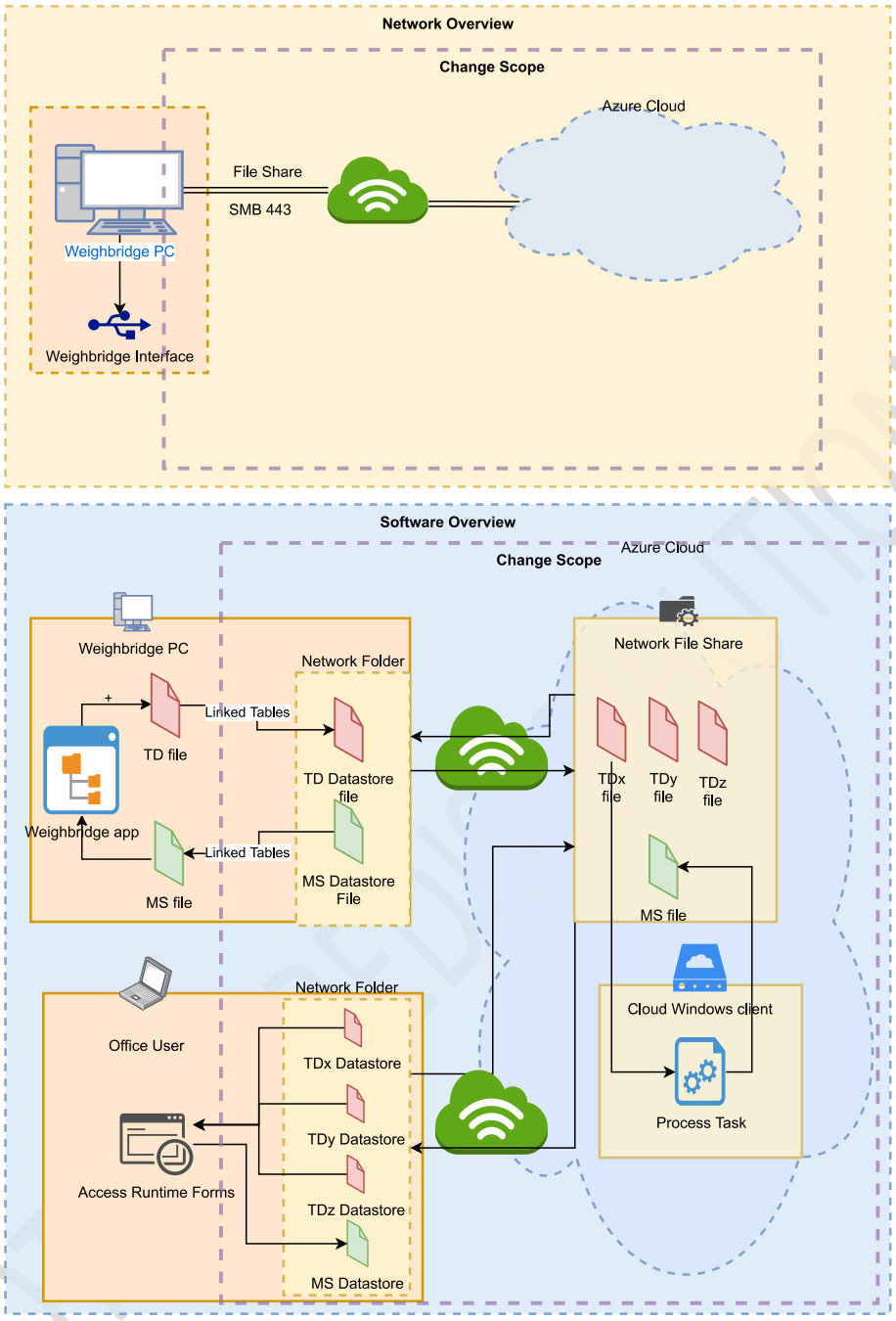


Figure 32: Network and software diagram showing implementation with a cloud-based virtual machine PC

5.7. Third round of Testing

I obtained a copy of the Admin Access database file known as "WFT Sys3Admin". This is the database file that collates all the data from the different weighbridge locations and stores it in one location for analysis and administration. As briefly described in the Initial Investigation (5.1).

This database file contains most of its logic within VBA modules and forms with VBA code. As seen in Figure 33 below:

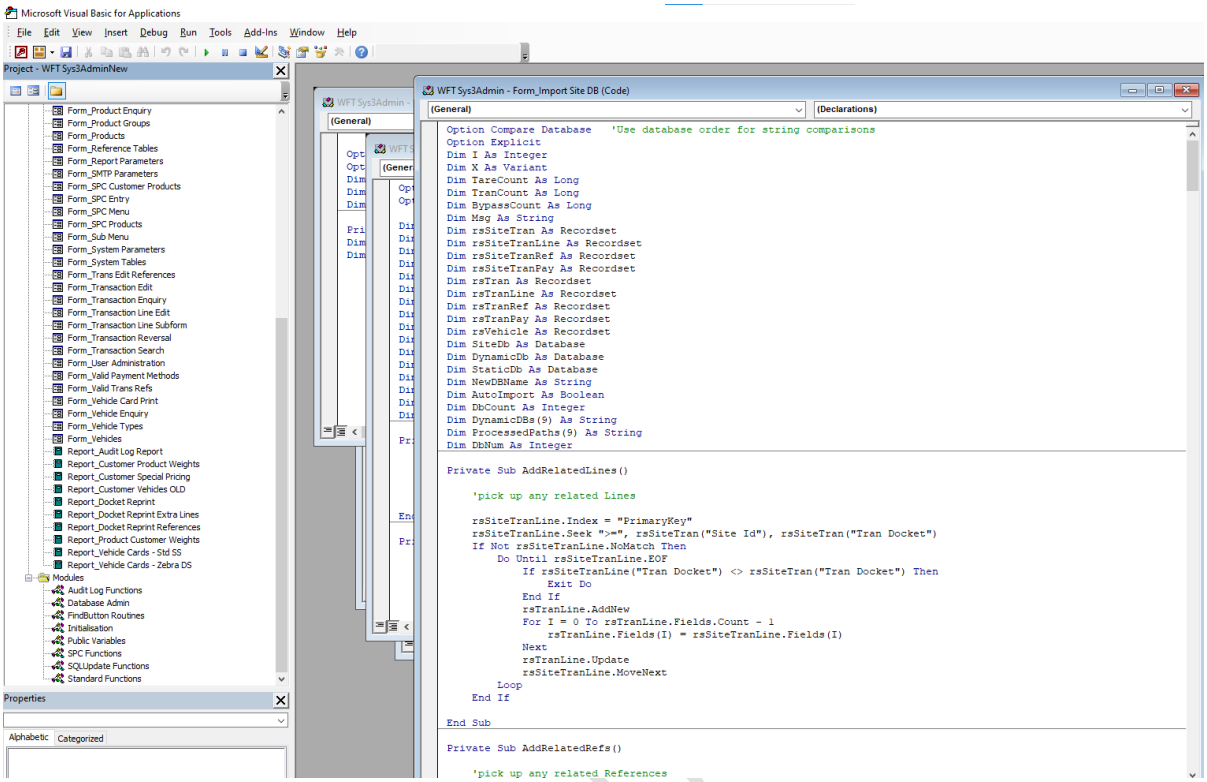


Figure 33: Screenshot of the VBA editor, showing forms, reports, modules, and a section of code for the form "Import Site Db"

The way the admin database file imports the different site data is by scanning a folder in the relative directory called "Import". It then processes the data and generates a new Site Static database, placed in another folder called "Export". After the imported database files are processed, they are moved to a folder inside "Import" called "Processed".

To try to integrate my current solution with the admin database, I simply placed the site database files in the "Import" directory that has their tables linked back to the datastore. To import the data from those, I simply launch the admin database with the command-line argument of "AutoImport". In theory, that should import the data that is linked to the datastore databases. However, it causes the admin database to crash.

Because VBA code is open to be read by anyone with permission to open the admin database file, I decided to have a look at the code to determine how the "AutoImport" function works.

I found that when the database loads, a VBA module named "Initialisation" runs, which reads all the command line arguments associated with the instance. If it detects the "AutoImport" argument is present, it opens a form named "Import Site Db" (Figure 33). In the code for that form, I can see the way it imports the data is that it first links the tables it wants to import to the admin database, imports the data using a SQL query, and then unlinks them.

The problem with this approach is that the tables it is trying to import from are in turn linked to another database file. From my testing with the Access linked tables features earlier, I discovered that a second-generation link will not work.

To remedy this, I decided to take it upon myself to modify the VBA code of the admin database file to import directly from the datastore. This way it will still only be a first-generation link and will not cause it to crash.

5.7.1. Modification of Admin Database

5.7.1.1. Adding Regex processing for command line arguments

The current way the Initialization module detects which command line arguments are given, it just checks if it contains specific words, without considering their context. As for the "DbPath" argument (which is where you give it the path to the directory the admin file is in), it must be placed at the end since the way it parses the path is by trimming the string to the end of the word "DbPath". This isn't ideal, since most command line arguments should be able to be passed in any order.

To remedy this, I am going to use a regular expression (see 3.15) to split the command line arguments into their respective elements and store them in a list. Then check each element in said list for the keywords, paths, etc. To do this I first need to add the VBA expressions library to the VBA project (Figure 34).

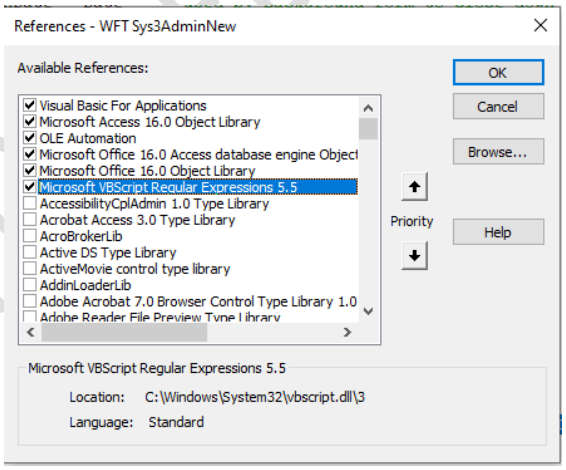


Figure 34: Screenshot of the VBA project library references window

I then execute the regular expression on the command line arguments and set global variables which are used by other parts of the code later. As seen in Figure 35 below:

5.7.1.2. Adding new command line arguments

```
Set regexObject = New RegExp
With regexObject
    .Pattern = "(?:[^/]|//)+"
    .Global = True
    .Multiline = True
End With
' Use old variables if no argument passed
DbPath = DbPath1
ImportPath = DbPath & LegacyLocalImportFolder
ExportPath = DbPath & LegacyExportFolder
DbSuffix = LegacyDbSuffix
' find what arguments are in argument list
For Each Match In regexObject.Execute(Command$)
    If InStr(Match.Value, "DbPATH=") <> 0 Then
        DbPath = Trim(Mid(Match.Value, InStr(Match.Value, "DbPATH=") + 7))
    End If
    If InStr(Match.Value, "ImportPATH=") <> 0 Then
        ImportPath = Trim(Mid(Match.Value, InStr(Match.Value, "ImportPATH=") +
11))
    End If
    If InStr(Match.Value, "ExportPATH=") <> 0 Then
        ExportPath = Trim(Mid(Match.Value, InStr(Match.Value, "ExportPATH=") +
11))
    End If
    If InStr(Match.Value, "DbSuffix=") <> 0 Then
        DbSuffix = Trim(Mid(Match.Value, InStr(Match.Value, "DbSuffix=") + 9))
    End If
Next Match
```

Figure 35: Code snippet from the Initialization module showing the parsing of command-line arguments using a regex pattern

Partially seen in Figure 35 above, I created several new command line arguments to make the admin database more portable and easier to use with a datastore:

- `/importpath=<path>`
  - The path that the database will search for database files to import
- `/exportpath=<path>`
  - The path that the database will export finished database files to
- `/DbSuffix=<suffix>`
  - Import databases only containing this suffix. E.g., "_DS" for datastore
- `/legacylocalimport`
  - Switch to enable legacy behavior (i.e., look and export to old locations)
- `/noprocessmove`
  - Switch to disable old behavior of moving databases to a folder called "Processed" after import

All these options change how the admin database looks for and interacts with different files. I then modified all the parts of the code where there are hard-coded values for the paths I am looking to change. For example, where the database specifically looks for a folder called "Import", I replace that instance with the path that is passed via a variable.

I have done this because some of the features of the admin database are not compatible with having a datastore. Particularly, the datastore cannot be in the "Import" directory in the same directory as the admin database. With these new options like "noprocessmove", the database will no longer use the old behavior and move the database files after processing them. However, you can revert to the old behavior for any of these arguments simply by not including them at startup. Using the "legacylocalimport" option will use the old behavior as well as the new behavior. This is in the cases where some of the weighbridge sites are using the new datastore method and some are not.

After testing and debugging these modifications with my previously established datastore solution, I am happy to say that everything about the auto import process appears to be working correctly. Because the changes made to the database code aren't major, I am not too concerned about data security or corruption issues. The changes made simply just modify where the database imports and exports data to and from.

5.7.1.3. Enabling the database to be launched from a remote file share

Because the changes I have made modify how the database interacts with the file system, it is now possible to place the entire admin database file itself on a network share. This is possible because all the hard-coded locations are now able to be edited via command line arguments. This outlines one of the desired outcomes by my colleagues at the TDC as described in Figure 32 (5.6).

The issue is that the way the database has been launched in the past is by passing the arguments using a Windows Shortcut. Windows shortcuts are not relative, so including one in the same directory as the admin database file will not work. You can't tell the shortcut to "open the file right next to me", it needs to know exactly how to find it from the root of the drive. It also must be modified to specifically target versions of Microsoft Office which are installed.

The original setup had several shortcuts, each with its own separate arguments.

To remedy this, I have created a PowerShell script that launches the admin database and is completely agnostic of the location and which version of Microsoft Office is installed (it always launches the newest one). The script itself has launch arguments to limit the number of shortcuts that need to be created that point to the script file. See Figure 36 below:

```powershell
param (
    [switch]$autoimport, #import with new behavior
    [switch]$legacylocalimport, #import from new folder with new behavior
(MUST BE USED WITH AUTOIMPORT)
    [switch]$legacymode, #Old Behavior Only
    [switch]$legacymodeimport #Old Behavior Only AND import like old behavior
);

#Get path to most recent Access version installed on PC
$ACCESSPATH = get-childitem 'C:\Program Files*\Microsoft Office' -recurse |
Where-Object {$_.Fullname -notlike '*Updates*'} | Where-Object {$_.Name -eq
'msaccess.exe'} | Sort-Object -Descending | Select-Object -first 1 | ForEach-
Object { $_.FullName }

#Get path to import and export paths defined in text files
$IMPORTPATH = Get-Content ImportPath.txt  -First 1
$EXPORTPATH = Get-Content ExportPath.txt  -First 1
$curDir = (Get-Item .).FullName

$noPrint = Test-Path "$curDir\ImportNoPrint*" -PathType Leaf
$DBARGS = """$curDir\WFT Sys3Admin.mdb"" /wrkgrp ""$curDir\WFT Sys3.mdw""
/user [redacted] /pwd [redacted] /cmd /dbpath=$curDir"

#If any file named ImportNoPrint exists, set no print arugment
if($noPrint){
    $DBARGS += " /ImportNoPrint"
}
if(!$legacymode){
    $DBARGS += " /DbSuffix=_DS /importpath=$IMPORTPATH /exportpath=$EXPORTPATH
/NoProcessedMove"
}
if($autoimport -and !$legacymode){
    $DBARGS += " /AutoImport"
    if($legacylocalimport){
        $DBARGS += " /legacylocalimport"
    }
}
elseif($legacymode -and $legacylocalimport){
    $DBARGS += " /AutoImport"
}
Start-Process $ACCESSPATH $DBARGS
```

Figure 36: Code snippet of PowerShell script which launches admin database file

To recreate the original shortcuts using the PowerShell script, I created the following shortcuts accompanying the admin database file. See Table 1 below:

Shortcut Name	Target
Run Admin	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1`
Run Admin & AutoImport	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1 -autoimport`
Run Admin & AutoImport & Legacy Import	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1 -autoimport -legacylocalimport`
Run Admin in Legacy Mode	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1 -legacymode`
Run Admin in Legacy Mode & Legacy Import	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1 -legacymode -legacymodeimport`

Table 1: Table containing shortcut names and targets for original shortcut recreations on administration pc/shared folder

5.7.1.4. Modification of existing system to replace SSH file transfer with Azure File Storage share

Because of issues with SCP noted in 3.7, the existing solution needs to be changed as well. This is a simple change that just involves copying the files to an Azure File Storage share instead of sending them to a remote SSH server using SCP.

Below in Figure 37 is a PowerShell script that sets up the share on the weighbridge share and creates the required files.

```powershell
$sharename = "sttdcrefusewb.file.core.windows.net"
$sharefolder = "refuseweighbridge"
$sharepath = "\\$sharename\$sharefolder"
$user = "localhost\sttdcrefusewb"
$password = Read-Host -Prompt "Enter password for $user on $sharename"
$connectTestResult = Test-NetConnection -ComputerName $sharename -Port 445
if ($connectTestResult.TcpTestSucceeded) {
    # Save the password so the drive will persist on reboot
    cmd.exe /C "cmdkey /add:`"$sharename`" /user:`"$user`" /pass: `"$password`""

    # Save network folder to explorer

    Set-Variable -Name desktopIniContent -Option ReadOnly -value
([string]"[.ShellClassInfo]`r`nCLSID2={0AFACED1-E828-11D1-9187-B532F1E9575D}`r`nFlags=2")

    $networkLocationPath = "$env:APPDATA\Microsoft\Windows\Network Shortcuts"

    [void]$(New-Item -Path "$networkLocationPath\$sharefolder" -ItemType Directory -
ErrorAction Stop -Force)

    Set-ItemProperty -Path "$networkLocationPath\$sharefolder" -Name Attributes -Value
([System.IO.FileAttributes]::System) -ErrorAction Stop

    [object]$desktopIni = New-Item -Path "$networkLocationPath\$sharefolder\desktop.ini" -
ItemType File -Force

    Add-Content -Path $desktopIni.FullName -Value $desktopIniContent

    $WshShell = New-Object -ComObject WScript.Shell

    $Shortcut = $WshShell.CreateShortcut("$networkLocationPath\$sharefolder\target.lnk")
    $Shortcut.TargetPath = "$sharepath"
    $Shortcut.Save()

    # write sync task files
    $sitecode = Read-Host -Prompt "Enter Sitecode. E.g. Rich01, Mari01, Murc01, Coll01,
Taka01"
    $sitecode | Out-File -FilePath .\SiteCode.txt
    "$sharepath\Admin\Import\Processed" | Out-File -FilePath .\ReceivePath.txt
    "$sharepath\Admin\Import" | Out-File -FilePath .\SendPath.txt
} else {
    Write-Error -Message "Unable to reach $sharename via port 445."
}
```

Figure 37: Code snippet of PowerShell script that sets up Azure File Storage share on target PC and creates required files used by transfer script

Below (Figure 38) is the PowerShell script I created that replaces the existing daily sync task script that used SCP.

```
param (
    [switch]$send, #for sending files to remote server
    [switch]$receive #for receiving files from remote server
);
#get remote server paths and sitecode
$SendPATH = Get-Content SendPath.txt  -First 1
$ReceivePATH = Get-Content ReceivePath.txt  -First 1
$SiteCode = Get-Content SiteCode.txt -First 1

if($send){
    Copy-Item "C:\Weighbridge\WFTSiteDynamicDB.mdb" -Destination
"$SendPath\$SiteCodeWFTSiteDynamicDB.mdb" -Force
}
if($receive){
    Copy-Item "$ReceivePath\WFTSiteDynamicDB.mdb" -Destination
"C:\Weighbridge\WFTSiteDynamicDB.mdb" -Force
}
```

Figure 38: Code snippet of PowerShell script that sends/receives database files

To make the change easier on the Weighbridge software operators, I recreated the original shortcuts (Table 2), that instead call the PowerShell script.

Shortcut Name	Location	Target
Afternoon Task before you go Home	Desktop	`powershell.exe -noprofile -executionpolicy bypass -file C:\Weighbridge\TDC_Scripts\SyncScript.ps1 -send`
Morning Task When You Start	Desktop	`powershell.exe -noprofile -executionpolicy bypass -file C:\Weighbridge\TDC_Scripts\SyncScript.ps1 -receive`

Table 2: Table containing shortcut names and targets for original shortcut recreations on weighbridge sites

5.8. Second Proposal

I presented my findings for the third round of testing (5.7) to my colleagues at the TDC and demonstrated my changes (5.7.1) to them. They were very pleased with the opportunities this modification opens and noted that it fulfils more of their goals with the DIP. More so than the original project goal.

They suggested I should document my changes and create a deployment & testing plan. This is with the goal of allowing another staff member to deploy the changes in case I do not have enough time to deploy the changes myself.

5.9. Documentation, Deployment & Testing Plan

5.9.1. Deployment Plan

The problem with having to modify the admin database file is that it is no longer easy to test a partial deployment. A partial deployment in this case meaning, only modifying one of the weighbridge sites to have the new datastore version. The reason for this is that the admin database file must process all the sites. If you have two versions of the admin database, one processing the old solution data and the other being the new version processing the new data, the admin database becomes out of sync. It is then extremely difficult to then synchronize them back up.

I thought of this scenario; therefore I added the launch argument of "legacylocalimport" (5.7.1.2). This enables the admin database to process the new datastore files and the legacy files.

Another problem is that the original solution involved SCP, as explained in 3.7, SCP is no longer secure. As a result, I will need to modify the original solution to make it more secure first. This is outlined above in 5.7.1.4.

5.9.2. Testing Plan

The plan to test and ensure that the new system operates correctly, I plan to test the datastore version on one site initially for a couple of weeks. This is to pick up on any missed bugs and issues that may arise. Therefore, I can fix said issues before the full deployment to all the sites.

I will initially test the new version on the weighbridge site that gets the least amount of traffic. This is if there is a complete failure of the new datastore system, the Weighbridge software operator can manually record the entries of the day on paper. Those entries can then be manually entered into the admin database via the forms that it has built-in.

This is to ensure data and transaction history is not lost, even if the system is no longer operational.

5.9.3. Documentation

To aid in deployment, I have created two instructional documents for either myself or other TDC staff to follow. One document outlines how you set up the datastore version on the weighbridge site and how the new admin database, startup script, and launch shortcuts are to be deployed. This is seen at: 13.1 Appendix A: Remote datastore upgrade instructions.

The other outlines how to modify the original solution deployed on the weighbridge sites to use the more secure Azure File Storage share instead of the SCP command. This can be seen in 13.2 Appendix B: Weighbridge file share security upgrade instructions.

Because the admin database includes a lot of VBA code. I also created a document outlining the code changes I have made. This is in the scenario that another TDC employee needs to modify the source code, fix a bug I have created, add new features, etc. This can be seen in 13.3 Appendix C: Admin database changes

6. DEPLOYMENT & MONITORING

At the time of writing this report, I have yet to deploy the solution. However, if not me, then another one of my colleagues should be capable of doing it by following the documentation provided.

7. PLACEMENT EVALUATION & REFLECTION

7.1. Personal Reflection

For this project, I discovered that working with legacy software (that has little or no documentation) is very frustrating. However, I enjoyed finding the limitations in the approaches I could take and then figuring out how to work around them.

I do not regret the choice to do a work placement at all. I found there is something satisfying about doing a project that has practical implications and not a typical theoretical implementation (like a typical project assignment). Although working for a larger organisation, especially a government body, I found, does tend to be very bureaucratic. Although I would justify it is for a good reason.

For the most part, I still enjoyed this project, I just wished I could've done a full-stack programming project. That's what I tend to enjoy doing the most. Having to dig around in someone else's undocumented, legacy code isn't exactly my idea of fun. Though, I do understand that is just the nature of most typical IT-related jobs.

I thought the TDC's DIP plan was a good idea. It is still quite vague, but I understand the point of it. It's meant to be specific technology independent. Plus, it sets the goals for the IT teams as a whole to work towards.

7.2. Future considerations

If I were to do the project again, I would commission the original developer of the Weighbridge software to update it and fix the bug noted in 5.5, Second round of Testing. That way, the project could take advantage of some of the newer, more advanced Azure services available. Such as Azure "DataVerse" and Power BI. Also, the code contained inside the admin database could be migrated to a SaaS solution.

If there were no time considerations for this project, I would look into possible replacements for the entire Weighbridge solution replacements. Considering that the software is 30 years old and utilises multiple deprecated technologies.

8. CONCLUSION

In conclusion, the new methods implemented, documented, and tested should (in theory) make the solution more reliable, secure, and manageable. The software may still be outdated but not to the detriment of the security of the system, having wrapped a more modern and secure communication approach around it.

The modifications made allow the solution to be more future proof for hardware and software changes. Utilising more modern methods of data handling and storage, allowing the solution to be more flexible in how it can be accessed, and agnostic of the systems it runs on.

Although that I could not get some of the desired technology ideas to work with the existing software, I am happy with the progress that was achieved.

I think the Weighbridge software will need replacing sometime in the near future evident by the antiquity of its appearance (Figure 16), and because of the more modern feature-rich options possibly available. I feel that in the meantime, the changes achieved are complementary to the TDC's DIP goals and should make the solution more compatible with the future technologies and ideas that go along with that.

9. REFERENCES

Awati, R. (2022, January). *What is batch file?* SearchWindowsServer. https://www.techtarget.com/searchwindowsserver/definition/batch-file

Black, K. (2022, October 18). *What does It Mean to Decompile a Software Program?* EasyTechJunkie. http://www.easytechjunkie.com/what-does-it-mean-to-decompile-a-software-program.htm

Brink, S. (2020, July 24). *How to Add or Remove Always Available Offline Context Menu in Windows.* https://www.tenforums.com/tutorials/163151-how-add-remove-always-available-offline-context-menu-windows.html

Carnes, B. (2022, June 17). *Learn Visual Basic (.NET) – Full Course.* FreeCodeCamp.Org. https://www.freecodecamp.org/news/learn-visual-basic-net-full-course/

Cimpanu, C. (2019, January 14). *SCP implementations impacted by 36-years-old security flaws.* ZDNET. https://www.zdnet.com/article/scp-implementations-impacted-by-36-years-old-security-flaws/

Cooper, S. (2019, October 28). Microsoft Access: Is it still relevant? *Comparitech.* https://www.comparitech.com/net-admin/microsoft-access/

Crawford, D. (2019, December 13). *Syncthing Review | Is it worth it?* ProPrivacy.Com. https://proprivacy.com/cloud/review/syncthing

David, E., Roth, J., Guyer, C., & Microsoft. (2021, November 3). *Microsoft Open Database Connectivity (ODBC)—Open Database Connectivity (ODBC).* https://learn.microsoft.com/en-us/sql/odbc/microsoft-open-database-connectivity-odbc

geekgirlau. (2006, February 26). *Recordset who only returns one record [Archive]—VBA Express Forum* [Question]. Vbaexpress.Com. https://www.vbaexpress.com/forum/archive/index.php/t-7229.html

Jaffe, A., Lacomis, J., Schwartz, E. J., Goues, C. L., & Vasilescu, B. (2018). Meaningful variable names for decompiled code: A machine translation approach. *Proceedings of the 26th Conference on Program Comprehension*, 20–30. https://doi.org/10.1145/3196321.3196330

Lonvick, C. M., & Ylonen, T. (2006). *The Secure Shell (SSH) Protocol Architecture* (Request for Comments RFC 4251). Internet Engineering Task Force. https://doi.org/10.17487/RFC4251

Mabbutt, D. (2019, March 6). *What is Visual Basic?* ThoughtCo. https://www.thoughtco.com/what-is-visual-basic-3423998

Microsoft. (n.d.-a). *Cloud Computing Services | Microsoft Azure*. Retrieved November 2, 2022, from https://azure.microsoft.com/en-us/

Microsoft. (n.d.-b). *What is IaaS? Infrastructure as a Service | Microsoft Azure*. Retrieved November 3, 2022, from https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-iaas

Microsoft. (n.d.-c). *What is PaaS? Platform as a Service | Microsoft Azure*. Retrieved November 3, 2022, from https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-paas

Microsoft. (2021a, August 1). *Obtaining Directory Change Notifications—Win32 apps*. https://learn.microsoft.com/en-us/windows/win32/fileio/obtaining-directory-change-notifications

Microsoft. (2021b, September 14). *Count the number of records in a DAO Recordset*. https://learn.microsoft.com/en-us/office/vba/access/concepts/data-access-objects/count-the-number-of-records-in-a-dao-recordset

OpenSSH. (2022, August 4). *OpenSSH 9.0 Release Notes*. OpenSSH 9.0 Release Notes. https://www.openssh.com/txt/release-9.0

Rinne, T., & Ylonen, T. (2013, April 14). *Scp: Secure copy—Linux man page*. https://linux.die.net/man/1/scp

Salesforce. (n.d.). *What is Software as a Service (SaaS): A Beginner's Guide*. Salesforce.Com. Retrieved November 3, 2022, from https://www.salesforce.com/in/saas/

Schmidt, J. (2022, October 31). *Excel VBA*. Corporate Finance Institute. https://corporatefinanceinstitute.com/resources/excel/excel-vba/

ssh.com. (n.d.). *SFTP protocol, clients, servers etc. Page by the original author of SFTP.* Retrieved November 3, 2022, from https://www.ssh.com/academy/ssh/sftp

Tao, C. (2022). *Clumsy* [C]. https://github.com/jagt/clumsy/blob/d7c5fcfe22543925a62e05a994e63a2204862070/README.md (Original work published 2013)

Tasman District Council. (2020). *Digital Innovation Program Workstreams* [Internal].

Terra, J. (2022, May 24). *What is Microsoft Access? An Introductory Guide | Simplilearn.* Simplilearn.Com. https://www.simplilearn.com/what-is-microsoft-access-article

The PHP Group. (n.d.). *PHP: Prepared statements and stored procedures—Manual*. Retrieved November 4, 2022, from https://www.php.net/manual/en/pdo.prepared-statements.php

Thompson, B. (2020, April 5). *Powershell Tutorial for Beginners: Learn Powershell Scripting.* https://www.guru99.com/powershell-tutorial.html

VB Decompiler—Products—Visual Basic 6.0 decompiler and .NET disassembler. (n.d.). Retrieved October 12, 2022, from https://www.vb-decompiler.org/products.htm

VMware. (n.d.). *What is a Virtual Machine? | VMware Glossary.* VMware. Retrieved November 5, 2022, from https://www.vmware.com/topics/glossary/content/virtual-machine.html

Walilko, A. (2022, January 19). *What are Regular Expressions?* Liquid Web. https://www.liquidweb.com/kb/what-are-regular-expressions/

Wenzel, K. (2021, October 22). *What is a Database Cursor?* Essential SQL. https://essentialsql.com/database-cursor/

10. TABLE OF FIGURES

11. LIST OF TABLES

12. GLOSSARY OF TERMS

Term	Definition
NMIT	Nelson Marlborough Institute of Technology
TDC	Tasman District Council
DIP	Digital innovation Program
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
SSH	Secure Shell Protocol
SFTP	SSH File Transfer Protocol
SCP	Secure Copy Protocol
RCP	Remote Copy Protocol
GUI	Graphical User Interface
VB	Visual Basic
VBA	Visual Basic for Applications
MS	Microsoft
API	Application Programming Interface
DOS	Disk Operating System
Exec	Execute
Posh, PS, ps1	PowerShell
OS	Operating System
BAT	Batch file
ODBC	Open Database Connectivity
DBMS	Database Management System
VM	Virtual Machine
Regex	Regular Expression
TDx, TDy, TDz	Transactional Dynamic 1, 2, 3
MS	Master Static (diagrams only)
SQL	Structured Query Language
PC	Personal Computer
Dbo	Database Object
FOSS	Free and Open-Source Software
BYO	Bring Your Own
P2P	Peer-to-Peer
WFT	Way Forward Technologies
Sys	System
Db	Database
DS	DataStore
IoT	Internet of Things

13. APPENDICIES

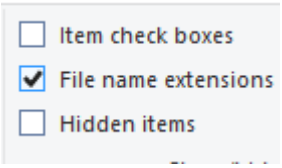
13.1. Appendix A: Remote datastore upgrade instructions

Add remote share folder to Windows and enable Offline Files on Weighbridge PC

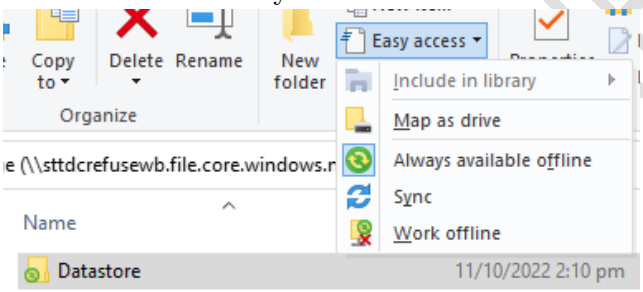
1. Ask [redacted] ([redacted]@[redacted]) for Azure Weighbridge share password for the user "[redacted]"
2. Save the following as AddShare.txt on the desktop

```
$sharename = "[redacted]"
$sharefolder = "refuseweighbridge"
$sharepath = "\\$sharename\$sharefolder"
$user = "localhost\sttdcrefusewb"
$password = Read-Host -Prompt "Enter password for $user on $sharename"
$connectTestResult = Test-NetConnection -ComputerName $sharename -Port 445
if ($connectTestResult.TcpTestSucceeded) {
    # Save the password so the drive will persist on reboot
    cmd.exe /C "cmdkey /add:`"$sharename`" /user:`"$user`" /pass:
`"$password`""

    # Save network folder to explorer

    Set-Variable -Name desktopIniContent -Option ReadOnly -value
([string]"[.ShellClassInfo]`r`nCLSID2={0AFACED1-E828-11D1-9187-
B532F1E9575D}`r`nFlags=2")

    $networkLocationPath = "$env:APPDATA\Microsoft\Windows\Network Shortcuts"

    [void]$(New-Item -Path "$networkLocationPath\$sharefolder" -ItemType
Directory -ErrorAction Stop -Force)

    Set-ItemProperty -Path "$networkLocationPath\$sharefolder" -Name
Attributes -Value ([System.IO.FileAttributes]::System) -ErrorAction Stop

    [object]$desktopIni = New-Item -Path
"$networkLocationPath\$sharefolder\desktop.ini" -ItemType File -Force

    Add-Content -Path $desktopIni.FullName -Value $desktopIniContent

    $WshShell = New-Object -ComObject WScript.Shell

    $Shortcut =
$WshShell.CreateShortcut("$networkLocationPath\$sharefolder\target.lnk")
    $Shortcut.TargetPath = "$sharepath"
    $Shortcut.Save()

} else {
    Write-Error -Message "Unable to reach $sharename via port 445."
}
```

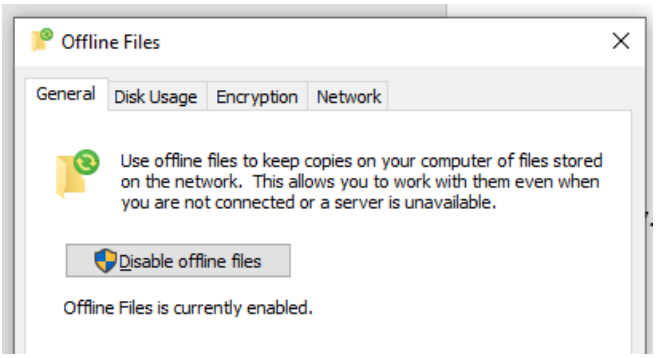
3. In explorer, navigate to the View tab and enable the checkbox “File name extensions”



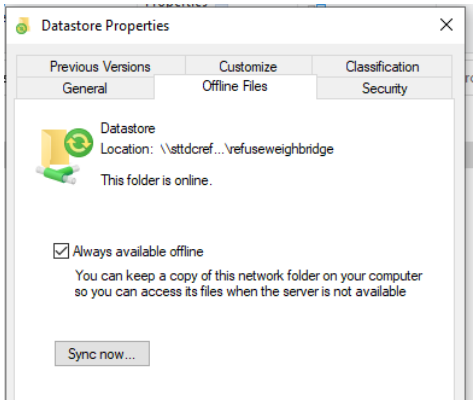
4. Rename the file you created from “AddShare.txt” to “AddShare.ps1”
5. Execute the script by holding shift, right clicking and clicking “Open PowerShell Here” in the folder where you saved the file.
6. In the command window that appears, run following command:
   `powershell.exe -noprofile -executionpolicy bypass -file .\AddShare.ps1`
7. When prompted, paste in the password you obtained by right clicking on the window and hitting the enter key.
8. Ensure that script succeeded, the drive appears in “This PC” explorer page and you can open it. You should see at least one folder named “Datastore”
9. In explorer, click on the folder named “Datastore”, expand Home tab, open easy access menu and click “Always Available Offline”



- If running windows 11 or newer, Open control panel and enter “W:\” in address bar.
- If easy access menu items are greyed out:
  i. In window search for “Manage Offline Files”, ensure that in the dialog that appears, it says “Offline Files are currently enabled. Otherwise click “Enable offline files”



  ii. Sign out of windows and reboot pc. Do this on repeat until the option is no longer greyed out.

10. Opening the properties tab for the “Datastore” folder should display the “Offline Files” tab. 

11. Search for “Sync center” in windows search
12. In the window that appears, double click on “Offline Files”
13. Click on refuseweighbridge and click the schedule button.
14. In the dialog that appears, ensure that the refuseweighbridge share is selected and click next
15. Click “At a scheduled time”
16. Change “Repeat every” to 10 minutes and click next.
17. Name it “Weighbridge schedule” and click save.
18. Press Win+R to open the Run dialog and enter: `taskschd.msc`
19. In the left panel, expand: Task Schedule Library -> Microsoft -> Windows -> Sync Centre
20. There should be a folder and a subfolder with GUID names, expand each one until you find the one with the task inside it named “Weighbridge schedule”.
21. Double click on “Weighbridge schedule” and click the triggers tab.
22. Click the new button and on the dialog that appears change the drop down to “At logon”.
23. Change settings to Any user and click Ok.
24. Click the new button again and on the dialog that appears, change the drop down to “On disconnect from user session”.
25. Change the settings to Any user, “Connection to local computer” and click Ok.
26. Click Ok on properties dialog.

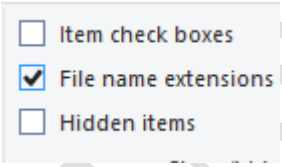
Copy Site Dynamic and Site Static database files to remote share folder

1. Open TDC file share named \\tsrvmisc\ and open the folder Weighbridge
2. Open the folder named Export
3. Copy the file named “WFTSiteStaticDb.mdb” and place it the “Export” inside the Datastore folder on the refuseweighbrige share.
4. Rename the copy and added the suffix “_DS”. E.g., “WFTSiteStaticDb_DS.mdb”

5. Go back to the share named \\tsrvmisc\, open the folder Weighbridge, open the folders Import and then "Processed".
6. Copy all the WFTSiteDynamicDb databases with the prefix of the site you want and place them in the "Import" folder inside the Datastore folder on the refuseweighbridge share.
7. Rename the copied databases to include the suffix "_DS". E.g., "Rich01WFTSiteDynamicDb_DS.mdb"

Link Site Dynamic and Site Static databases to the remote datastore tables on the Weighbridge PC

1. Open TDC file share named \\tsrvmisc\ and open the folder Weighbridge
2. Open the folder named Export
3. Copy the file named "WFTSiteStaticDb.mdb" and place it inside the folder C:\Weighbridge
4. Go back to the share named \\tsrvmisc\, open the folder Weighbridge, open the folders Import and then "Processed".
5. Copy the database WFTSiteDynamicDb that includes the prefix for the site on which the Weighbridge PC you are operating on resides. E.g., Richmond site is "Rich01WFTSiteDynamicDb_DS.mdb"
6. Place it inside the folder C:\Weighbridge and rename it to remove the site Prefix. E.g., "Rich01WFTSiteDynamicDb.mdb" -> "WFTSiteDynamicDb.mdb"
7. In explorer, navigate to the View tab and enable the checkbox "File name extensions"

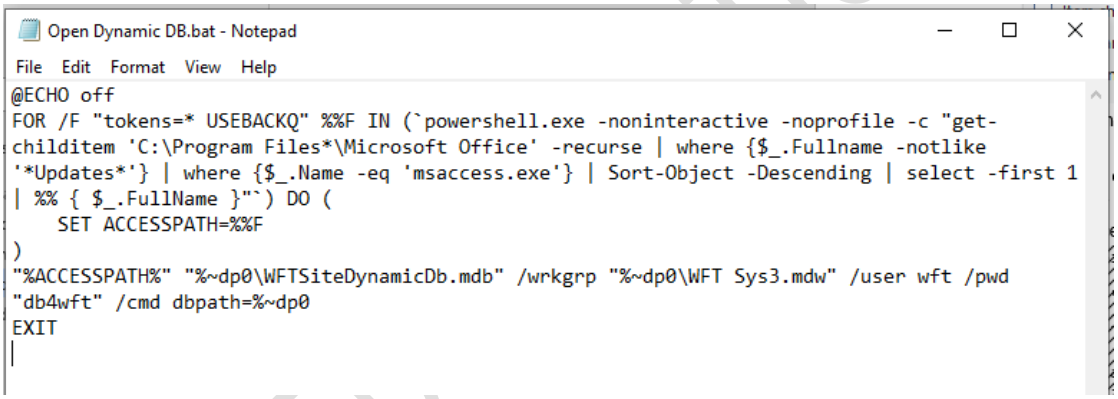


8. Create a Text document and rename it from "New Text Document.txt" to "Open Site Dynamic DB.bat"
9. Right click on "Open Site Dynamic DB.bat" and click edit.
10. In the text box that appears, paste the target for Microsoft access.

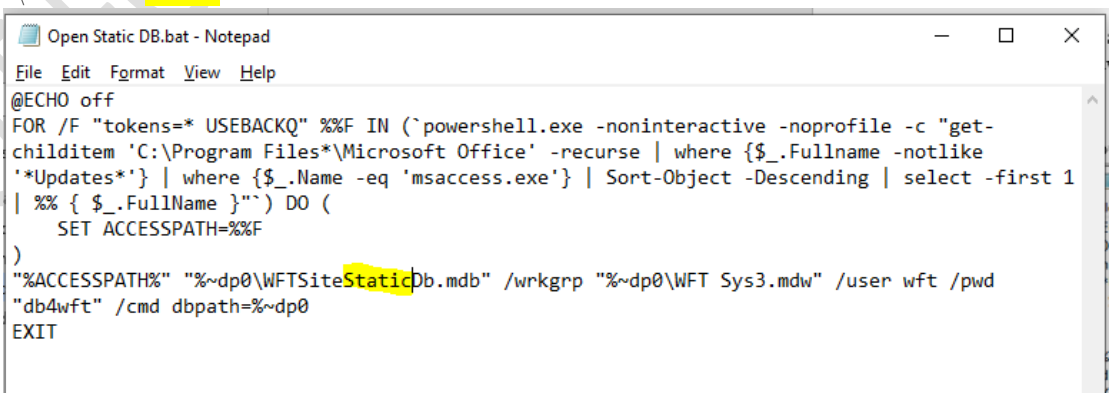
11. Press space at least once and copy and paste the following text:

```
@ECHO off
FOR /F "tokens=* USEBACKQ" %%F IN (`powershell.exe -noninteractive -noprofile
-c "get-childitem 'C:\Program Files*\Microsoft Office' -recurse | where
{$_.Fullname -notlike '*Updates*'} | where {$_.Name -eq 'msaccess.exe'} |
Sort-Object -Descending | select -first 1 | %% { $_.FullName }"`) DO (
    SET ACCESSPATH=%%F
)
"%ACCESSPATH%" "%~dp0\WFTSiteDynamicDb.mdb" /wrkgrp "%~dp0\WFT Sys3.mdw" /user
wft /pwd "db4wft" /cmd dbpath=%~dp0
EXIT
```

- Your notepad window should now look like this:

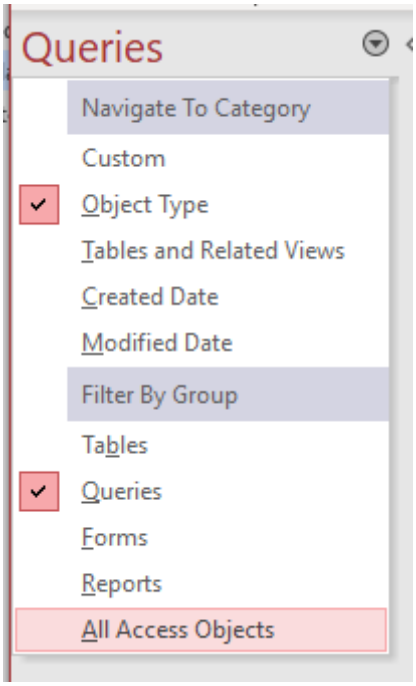


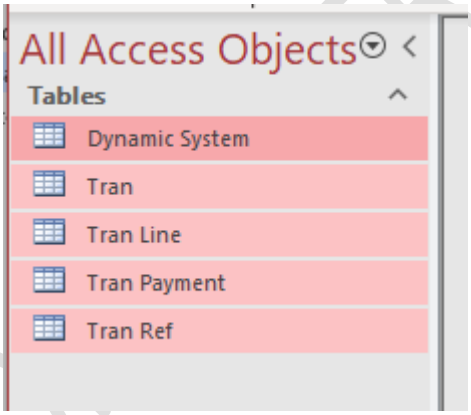
12. Click file and then click save.
13. Make a copy of the file “Open Site Dynamic DB.bat” in the same directory and rename it to “Open Site Static DB.bat”
14. Right click on “Open Site Static DB.bat” and click edit.
15. Modify the statement ... "%~dp0\WFTSiteDynamicDb.mdb"... to be ... "%~dp0\WFTSiteStaticDb.mdb"...

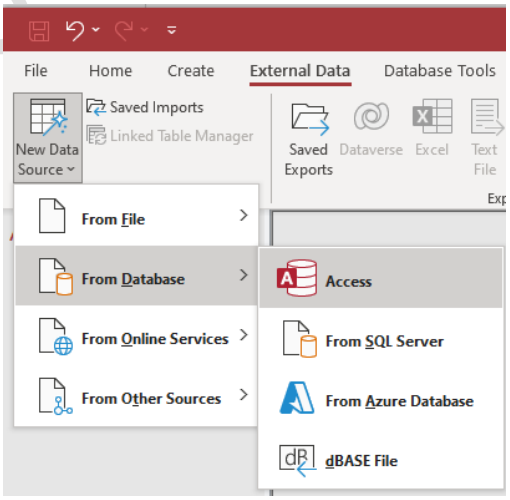


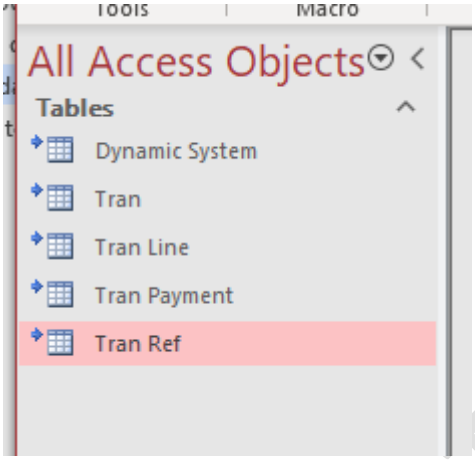
-

16. Double click to open the file “Open Site Dynamic DB.bat”.

17. In the Microsoft Access window that appears, change the navigation view to All Objects 

18. Click on the first table and then hold shift and click on the last table in the navigation menu. 

19. Right click on any table and then click delete. Press "Yes" on every dialog box that appears. Your navigation view should now have no tables listed.
20. Navigate to the external data tab, click "New Data Source", From Database and then Access Database.
    - 

21. In the dialog that appears, browse the refuseweighbridge share, Datastore Folder and then Import. Select the WFTSiteDynamicDb_DS file that includes the prefix for the site of the computer you are operating on. E.g., Rich01WFTSiteDynamicDb_DS for Richmond.
22. Change the option to "Link to the data source by creating a linked table" and click Ok.
23. In the list dialog that appears, click "Select All" and then click Ok. There should now be linked tables in the navigation view.
    - 

24. Navigate to the "Database Tools" tab and click the button "Compact and Repair Database"
25. Once it is done, close the database.
26. Execute the file "Open Site Static DB.bat".
27. Repeat steps 21-29 for static database, except on step 25, locate WFTSiteStaticDB_DS in Export folder.

Change WFT Admin database file to the 7.0.X version and update launch shortcuts on tsrvmisc PC

1. Complete steps 1-4 under "Add remote share folder to Windows and enable Offline Files on Weighbridge PC"
2. Create backup of entire Weighbridge Folder on tsrvmisc PC.
3. Copy the 7.x.x version of the file to "WFT Sys3Admin.mdb" to the Weighbridge Folder.
4. Create a new text file and rename it from "New Text Document.txt" to "RunAdminScript.ps1"

5. Open it in notepad and paste the following content:

```
param (
    [switch]$autoimport, #import with new behavior
    [switch]$legacylocalimport, #import from new folder with new behavior
(MUST BE USED WITH AUTOIMPORT)
    [switch]$legacymode, #Old Behavior Only
    [switch]$legacymodeimport #Old Behavior Only AND import like old behavior
);

#Get path to most recent Access version installed on PC
$ACCESSPATH = get-childitem 'C:\Program Files*\Microsoft Office' -recurse |
Where-Object {$_.Fullname -notlike '*Updates*'} | Where-Object {$_.Name -eq
'msaccess.exe'} | Sort-Object -Descending | Select-Object -first 1 | ForEach-
Object { $_.FullName }

#Get path to import and export paths defined in text files
$IMPORTPATH = Get-Content ImportPath.txt  -First 1
$EXPORTPATH = Get-Content ExportPath.txt  -First 1
$curDir = (Get-Item .).FullName

$noPrint = Test-Path "$curDir\ImportNoPrint*" -PathType Leaf
$DBARGS = """$curDir\WFT Sys3Admin.mdb"" /wrkgrp ""$curDir\WFT Sys3.mdw""
/user WFT /pwd db4wft /cmd /dbpath=$curDir"

#If any file named ImportNoPrint exists, set no print arugment
if($noPrint){
    $DBARGS += " /ImportNoPrint"
}
if(!$legacymode){
    $DBARGS += " /DbSuffix=_DS /importpath=$IMPORTPATH /exportpath=$EXPORTPATH
/NoProcessedMove"
}
if($autoimport -and !$legacymode){
    $DBARGS += " /AutoImport"
    if($legacylocalimport){
        $DBARGS += " /legacylocalimport"
    }
}
elseif($legacymode -and $legacylocalimport){
    $DBARGS += " /AutoImport"
}
Start-Process $ACCESSPATH $DBARGS
```

6. Save and close the script
7. Create a new Text Document and name it “ImportPath.txt”
8. Open the folder called import inside the datastore and copy the path.
9. Open the text document ImportPath.txt, paste the path.
10. Repeat steps 7-9 except for the export folder and ExportPath.txt
11. Remove all shortcuts containing the prefix “System 3000” in the Weighbridge folder

12. Create the following shortcuts with the following paths

Shortcut Name	Target
Run Admin	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1`
Run Admin & AutoImport	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1 -autoimport`
Run Admin & AutoImport & Legacy Import	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1 -autoimport -legacylocalimport`
Run Admin in Legacy Mode	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1 -legacymode`
Run Admin in Legacy Mode & Legacy Import	`%windir%\System32\WindowsPowershell\v1.0\powershell.exe -noprofile -executionpolicy bypass -file .\AdminRunScript.ps1 -legacymode -legacymodeimport`

13. In each shortcut you created, open the properties and ensure that the field "Start In:" is empty.
    - Optional: Create text document called ImportNoPrint.txt to prevent autoimport from printing report.

13.2. Appendix B: Weighbridge file share security upgrade instructions

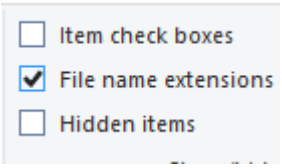
Add remote share folder to Windows Explorer

1) Ask [redacted] ([redacted]@[redacted]) for Azure Weighbridge share password for the user "[redacted]"
2) Copy the password to the clipboard
3) Navigate in Windows Explorer to "C:\Windows\Weighbridge\TDC_Scripts"
4) Delete the contents of the folder.

5) Save the following as ShareSetup.txt

```
$sharename = "[redacted]"
$sharefolder = "refuseweighbridge"
$sharepath = "\\$sharename\$sharefolder"
$user = "localhost\sttdcrefusewb"
$password = Read-Host -Prompt "Enter password for $user on $sharename"
$connectTestResult = Test-NetConnection -ComputerName $sharename -Port 445
if ($connectTestResult.TcpTestSucceeded) {
    # Save the password so the drive will persist on reboot
    cmd.exe /C "cmdkey /add:`"$sharename`" /user:`"$user`" /pass: `"$password`""

    # Save network folder to explorer

    Set-Variable -Name desktopIniContent -Option ReadOnly -value
([string]"[.ShellClassInfo]`r`nCLSID2={0AFACED1-E828-11D1-9187-B532F1E9575D}`r`nFlags=2")

    $networkLocationPath = "$env:APPDATA\Microsoft\Windows\Network Shortcuts"

    [void]$(New-Item -Path "$networkLocationPath\$sharefolder" -ItemType Directory -
ErrorAction Stop -Force)

    Set-ItemProperty -Path "$networkLocationPath\$sharefolder" -Name Attributes -Value
([System.IO.FileAttributes]::System) -ErrorAction Stop

    [object]$desktopIni = New-Item -Path "$networkLocationPath\$sharefolder\desktop.ini" -
ItemType File -Force

    Add-Content -Path $desktopIni.FullName -Value $desktopIniContent

    $WshShell = New-Object -ComObject WScript.Shell

    $Shortcut = $WshShell.CreateShortcut("$networkLocationPath\$sharefolder\target.lnk")
    $Shortcut.TargetPath = "$sharepath"
    $Shortcut.Save()

    # write sync task files
    $sitecode = Read-Host -Prompt "Enter Sitecode. E.g. Rich01, Mari01, Murc01, Coll01,
Taka01"
    $sitecode | Out-File -FilePath .\SiteCode.txt
    "$sharepath\Admin\Import\Processed" | Out-File -FilePath .\ReceivePath.txt
    "$sharepath\Admin\Import" | Out-File -FilePath .\SendPath.txt
} else {
    Write-Error -Message "Unable to reach $sharename via port 445."
}
```

6) In explorer, navigate to the View tab and enable the checkbox “File name extensions”



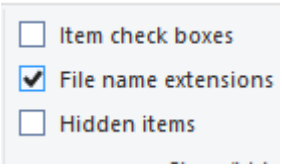
7) Rename the script from “ShareSetup.txt” to “ShareSetup.ps1”
8) Execute the script by holding shift, right clicking and clicking “Open PowerShell Here” in the folder where you saved the file.
9) In the command window that appears, run following command:
`powershell.exe -noprofile -executionpolicy bypass -file .\ShareSetup.ps1`
10) When prompted, paste in the password you obtained by right clicking on the window and hitting the enter key.
11) Also, when prompted, enter the sitename in which the PC is located.
12) Ensure that script succeeded, the drive appears in “This PC” explorer page and you can open it. You should see at least one folder named “Datastore”

Create new Sync Script.

1. Navigate in Windows Explorer to “C:\Windows\Weighbridge\TDC_Scripts”
2. Save the following in the folder as “SyncScript.txt”

```
param (
    [switch]$send, #for sending files to remote server
    [switch]$receive #for receiving files from remote server
);
#get remote server paths and sitecode
$SendPATH = Get-Content SendPath.txt  -First 1
$ReceivePATH = Get-Content ReceivePath.txt  -First 1
$SiteCode = Get-Content SiteCode.txt -First 1

if($send){
    Copy-Item "C:\Weighbridge\WFTSiteDynamicDB.mdb" -Destination
"$SendPath\$SiteCodeWFTSiteDynamicDB.mdb" -Force
}
if($receive){
    Copy-Item "$ReceivePath\WFTSiteDynamicDB.mdb" -Destination
"C:\Weighbridge\WFTSiteDynamicDB.mdb" -Force
}
```

3. In explorer, navigate to the View tab and enable the checkbox “File name extensions”



4. Rename the script from “ShareSetup.txt” to “ShareSetup.ps1”

5. Create the following shortcuts in the same folder:

Shortcut Name	Target
Afternoon Task before you go Home	`powershell.exe -noprofile -executionpolicy bypass -file C:\Weighbridge\TDC_Scripts\SyncScript.ps1 -send`
Morning Task When you Start	`powershell.exe -noprofile -executionpolicy bypass -file C:\Weighbridge\TDC_Scripts\SyncScript.ps1 -receive`

6. Copy both shortcuts you created to the desktop

13.3. Appendix C: Admin database changes

WFT System 3000 Admin 7.0.X Changes

Added the following launch arguments:

- `/DbSuffix=<suffix>`
  - This is the ending part of the filename in which access will search for and export. E.g., WFTSiteStaticDb_DS.mdb will be created when "_DS" is the suffix. So the argument would be **"/DbSuffix=_DS"**
- `/ImportPath=<path>`
  - This is the file path that access will search for the incoming database files from the weighbridge sites. E.g., \\fileshare\Datastore\Import\ would be **"/ImportPath=\\fileshare\Datastore\Import\"**
- /ExportPath=<path>
  - This is the file path that access will export the static database to. E.g., \\fileshare\Datastore\Export\ would be **"/ImportPath=\\fileshare\Datastore\Export\"**
- `/NoProcessedMove`
  - This tells access not to move the imported database files to a folder called "Processed" inside the
- `/legacylocalimport`
  - This is to be used alongside the existing /AutoImport option. When used, auto import will use the new behavior defined Dbsuffix, importpath, etc. but will also carry out legacy behavior as a secondary task. This is so access can support importing the new datastore databases but also the legacy ones too.