Running the DEP as Standalone

Roberto Picha and Mecene Desormice, U.S. Census Bureau

1. Abstract

For the last two decades, the U.S. Census Bureau has used Blaise 4 software to collect CATI and CAPI survey data in a "Standalone" environment, where the Blaise instruments interface with the Laptop and CATI case management systems. Our "standalone" (self-contained) environment involves the case management systems calling a Manipula transaction script, which in turn makes a call to the DEP via the Edit function. This script binds the metafile and local database containing a record or a single case. This process is highly integrated into the current survey systems that are currently being used at the Census Bureau. It is not an easy task to make drastic changes to this process.

With new software and hardware technologies emerging in the data collection world, a new paradigm is presented which involves more questions than answers. What is the ideal Blaise 5 environment for interviewing with different operating systems and on different devices? It is most likely a different environment than what is currently used for our CATI/CAPI data collection. Until we can figure out the best approach for this paradigm shift and while we are still running Blaise 4 production instruments, we decided to research what it will take to implement a Blaise 5 production survey in our existing environments (i.e., standalone). Can we continue to run our legacy control systems with a Blaise 5 instrument?

This paper covers our experiences in attempting to run a Blaise 5 instrument in our existing control systems environment (i.e., running the DEP as standalone). We began our research using Blaise 5.03 Build 810 and completed with Blaise 5.05 Build 975. We will share our lessons learned and experiences from this research, how the DEP can act as a service, and how to automate the process for installing of the minimum software required to deploy packages to this environment. We realize that this may not be the best approach for running Blaise 5 instruments, but it was important for us to determine what options we have for running Blaise 5 in production.

2. The use of Blaise for Data Collection

The U.S. Census Bureau CATI and CAPI data collection has been entirely built using Blaise 4 Enterprise; the software meets all the requirements to make it flexible and suitable with our systems. The Blaise 4 software is "distributed" to our Field Representative (FR) laptops and the phone center PCs. The software "distribution" primarily consists of the basic executables needed to bind the metafile and database running on the FR's laptop to collect data (Manipula.exe, Dataview.exe, and Hospital.exe).

It is important to mention that there are no special configurations or settings on the laptops in order to run Blaise 4. The data collection process is self-contained step and the instrument runs "as standalone" from within our case management system.

How Blaise 4 data collection is used by the Census Bureau most likely differs from how other agencies use Blaise 4 to collect data. Our current process was designed to be integrated with existing case management and control systems that could handle multiple data collection software applications. The Census Bureau is only using Blaise 4 to collect the data; it is not using Blaise to manage the assignments or case data. The "individual" case bdb concept was implemented so that cases could be easily re-assigned to other FR laptops or interviewer work stations. Tracking and re-assigning cases is handled by the control systems.

The goal of this research was to determine if it was possible to implement a production Blaise 5 survey with minimal changes to our control systems as it could take significant time, resources, and money to redesign the existing systems. It is essential that we keep it simple and manageable.

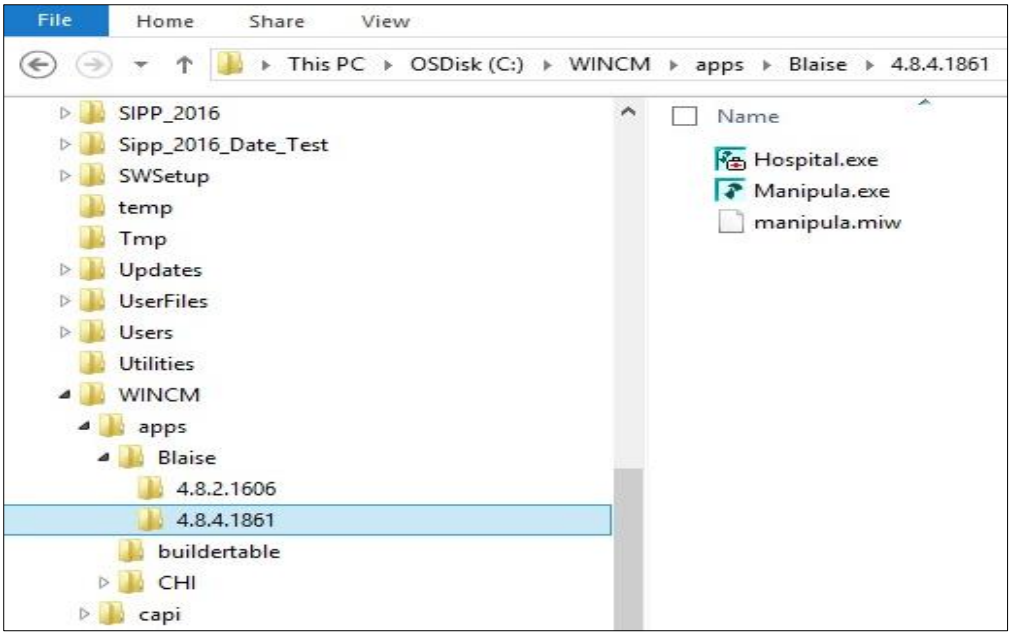
2.1 Current Process and Blaise software

Currently, the Census Bureau pushes down the minimum Blaise 4 software necessary to run our CAPI and CATI instruments. We are also required to run multiple version of the Blaise software as not all surveys are using the same version of Blaise 4.

The software distributed to the FR's laptops consists of the basic executables for data collection along with a few tools that are used for assisting the FR and/or Technical Assistant Center (TAC) for production related issues.

a) **Manipula** – Case Management will use Manipula to launch the instrument scripts that control the actions taken. These scripts make updates to the Blaise data, execute the Blaise interview using DEP, and output updates to the control systems.
b) **Data browser** – Used for analyzing the contents of the database when issues arise. Only the latest version of this executable is pushed to the laptops, therefore it is not parked in the same location as Manipula.
c) **Hospital** – Used in the event that something goes wrong with the case or between the instrument and case management.

Figure 1. Example of the folder structure where Blaise 4 software resides



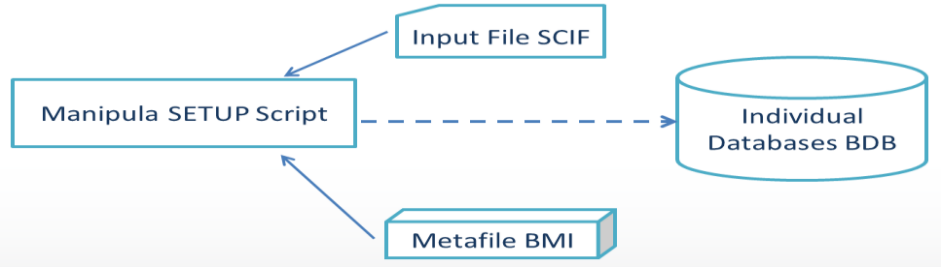
The Blaise 4 software is distributed to the laptops via AFARIA that uses a channel communication port. The executables are then parked to a fixed location on the laptop, the location is part of the OS path and therefore, no other configuration is needed. This action is only performed once for each version of Blaise approved for production.

Since our survey instruments are coded in different versions of Blaise, it is essential that the laptop environment is compartmentalized by the software version allowing us to support and run several versions of Blaise in the same OS.

2.1.2 Survey Instruments in the current process

The survey instrument is packaged as a compressed file and then delivered to our Master Control System (MCS). MCS runs a Manipula Setup script binding the instrument metafile and the SCIF input file (ASCCII file) creating single Blaise databases for each case. These single databases are then placed inside a blob and provided to ROSCO, where individual case assignments are made to the FRs. MCS also delivers the instrument compressed file (package) to a server from where software testing performs testing and then copies the survey instrument to the production server. From there, the instrument is transmitted to the FRs once they connect to the server. There is a communication channel process that will push down the instrument to FR's laptops and unzip the contents in the appropriate survey interview period folder. The instrument package contains the metafiles, i.e., instrument and any external files used by the main metafile.

Figure 2. MCS runs the setup script to create individual case BDB files



Our Case Management System (LCM) and Instrument talk to each other via ASCII files to share & update information when the FR opens the case.

LCM makes the call to our Manipula transaction script as follows:

C:\Wincm\apps\Blaise\4.8.4.1681\Manipula.exe C:\Wincm\database\studies\<surevyid>\e-inst\capi_trans.msu

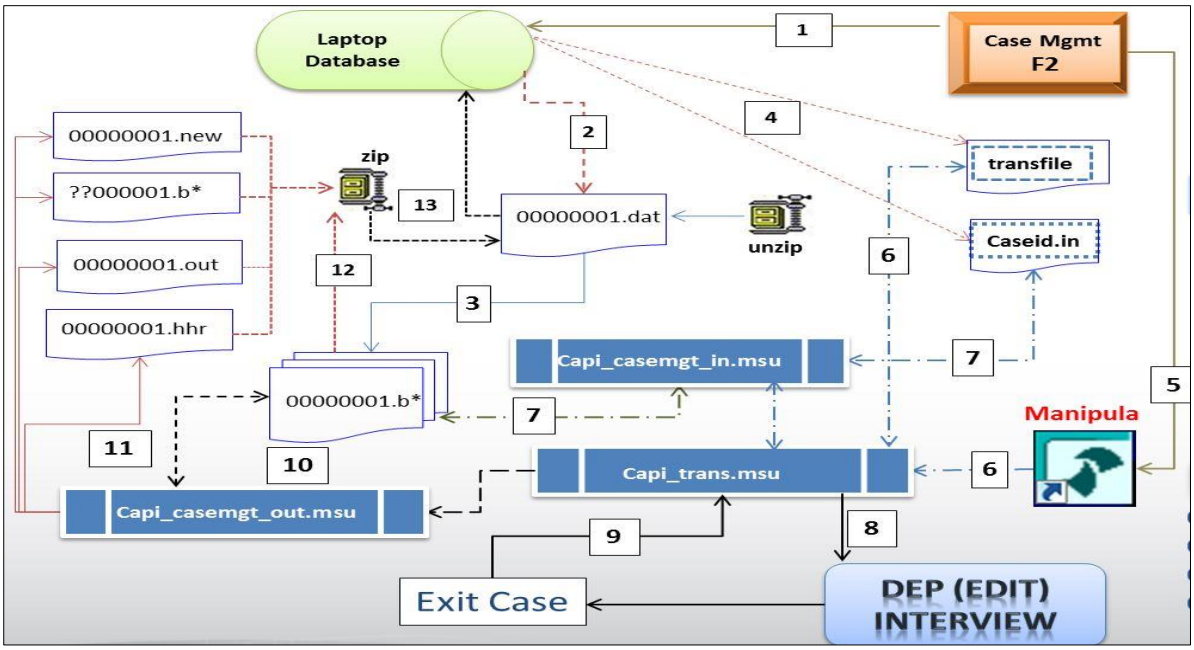
Then the capi_trans internally makes the call to the DEP as in figure 3:

Figure 3. Manipula calls the DEP as EDIT function

```
Reslt2:= EDIT(THEINSTRUMENT +  ' /M..\config\dews.bmf' +
                               ' /H..\config;..\externals /E..\config;..\externals' +
                               ' /F' + thecase +
                               ' /K' + thecase +
                               ' /G /X')
```

Overall, the interaction between the instrument and LCM must be flawless. The interaction not only calls the instrument, it makes updates to the case data via another Manipula script that is called from within our capi_trans.msu. Upon completing the case, updates (and possibly spawned cases) are made available to LCM to update, spawn and writing back information to LCM at times, other process are expected. The flow is described in figure (4).

Figure 4. Case Management and Blaise 4 Survey Instrument Interface



2.2 Blaise 5 Software and the new Process

Again, the desire is to maintain our current process as much as possible. There were concerns with how to blend Blaise 5 within our systems and expect the same results.

2.2.1 Installing Blaise 5 software to laptops.

Initial research suggested a full installation of Blaise 5 for all our laptops. However, this may not be permissible due to the configuration of our current laptops. The complexity with this approach is that we would not able to run different versions of the software simultaneously on the FR's laptop. There may be ways to address the issue, but at the time of this research we decided to explore ways of placing a subset of executables and run small scale testing.

2.2.2 Installing the Instrument or .bpgk file to the laptop.

We learned that through the Control Center and via the solution option, a package can be created containing all the pieces necessary to launch a Blaise 5 instrument. In observing the package contents, we noticed that all metafiles plus the regular defaulting accompanying files are included. Even though the Manipula scripts are part of the main solution, ideally we would like to have the scripts as part of the package. We can manually add the script as temporary solution.

In order for an instrument to run on our laptops, the instrument needs to be installed via the Server Manager. Since we do not want the FR to perform this step, our case management system would need to be adjusted to add this step for each new/update production instrument. The alternative would be to do it programmatically. While all this is doable, the consensus is to avoid this step and stick to our current process.

3. What We Require vs. What We Need

As this research was conducted, it became obvious that Blaise services and Server Manager could do all the steps to install and run an instrument in Blaise 5. While this is probably the best way to simplify our process, the Bureau is not yet on board with such approach given the constraints explained earlier.

Since Blaise 5 is not a simple copy of software process as in its predecessor Blaise 4, some initial steps must be conducted on the FR laptop prior to launching a survey. The Blaise services and server manager must be available one way or another. This imposes a challenge in identifying all dependencies. More importantly, would this new process allow laptops to run different version of the software without issues?

3.1 What do we required?

a) **Blaise server**
   Blaise 5 services are required to launch an instrument, whether this survey is launched locally (laptop) or over the network or internet. The server will handle all the transactions and databases for various surveys.

b) **Server Manager**
   The server manager is very useful and required to accomplish important tasks. With our early research of Blaise 5, we used the server manager to install/uninstall surveys. We learned that in order to use this tool, we pretty much had to install the entire enterprise to the laptops. This was not the most ideal situation, yet it was doable at the small scale test. We also found some early issues with the server manager that were later corrected by Statistics Netherlands.

While the server manager proves to be of use, we found that it has extremely large dependencies that needed to be pushed to the FR's laptops, which is impractical for our purpose.

3.2 What do we needed?

During the IBUC conference in China (2014), Statistics Netherlands indicated that the DEP could literally run as service, that is, the DEP contains an embedded service component. This service allows us to remove the need of the server manager and any extra dependencies.

The following syntax was all that was needed to launch a survey: **DEP.exe inst.bpkg**

At glance, this seemed simple and straight forward. However, we learned that there are a lot of things going on internally with the DEP and those findings are discussed below.

3.3 DEP to maintain our current process

Since the DEP can perform as a service, we decided to explore the mechanisms that trigger the events plus identify our needs and apply them accordingly. In the process, we learned that the DEP can do what is needed with minimum requirements.

Step 1: DEP Sets up the Environment

From using the syntax shared above, the DEP looks at the environment in which it is going to be run. It uses the information indicated by the **dep.exe.config** file to:

- Identify the location of the surveys
- Identify the location of the database configuration
- Identify the location where the package should reside

If the DEP does not find the environment properly configured, it will create one on the fly with default settings. Figure 5 shows an example of the DEP configuration file.

Figure 5. Example of DEP Configuration File

```
<?xml version="1.0"?>
<configuration>
        <appSettings>
                <add key="ExternalCommunicationAddress" value="http://localhost:8033"/>
                <add key="DeployFolder" value="C:\temp\Standalone\"/>
                <add key="ConfigurationDatabase" value="RuntimeConfiguration.db"/>
                <add key="ManagementCommunicationPort" value="8031"/>
                <add key="InternalCommunicationPort" value="8032"/>
                <add key="ExternalCommunicationPort" value="8033"/>
                <add key="ServerManagerDatabase" value="ServerManagerDatabase.db"/>
                <!--<add key="CertificateStoreName" value="My" />
    <add key="CertificateStoreLocation" value="LocalMachine" />
    <add key="CertificateThumbprint" value="7F94601C4FC2EE71C8AAC02947607F2E2CEEE4D2" />-->
        </appSettings>
        <!--<startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.1"/></startup>-->
</configuration>
```

We can also create our own configuration file, specifying the appropriate parameters as seen above.

Step 2: DEP Installs the Instrument

Once the environment is configured, the DEP will un-package and install the instrument the first time the syntax is invoked. The DEP runs the following steps:

- Places a copy of the package under the "Upload" folder
- Unzips the package and places it under the "Survey" folder
- Reads the main metafile and extracts The following two items about the instrument:
    - **Instrument ID** - unique identifier. This items can also be obtained from the Control Center
    - **Checksum**. This is important for data integrity. After compilation, in the output window log, the checksum is available and apparently this contained in the main metafile

Step 3: DEP populates Databases and Tables

The DEP then proceeds to populate (at minimum) two databases and some tables:

- The configuration database and table, see figure 7.
- The server manager database and tables, see figures 9, 10 & 11

Once these three steps are completed, the DEP renders the instrument - binding the Metafile with the database - just as it is done currently with Blaise 4

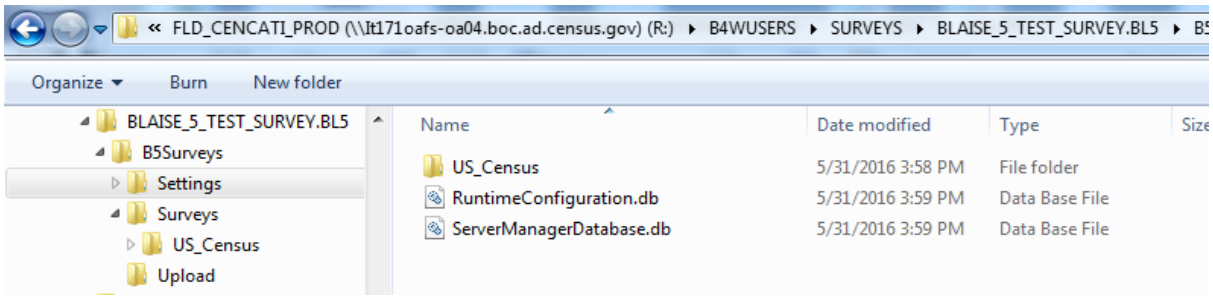
Note: If the instrument is called for subsequent interviews of the same case or any other cases for that survey instrument, the DEP acknowledges the environment and just proceeds to read the tables from the databases.

3.4 Database Configuration and Management

While the convenience of the DEP doing all steps automatically was great for us, we found that some management and configuration could enhance our capabilities for a quick turnaround. We decided to automate the process and manage it properly for our purposes. Note that we currently do not use the DEP to launch our instruments; rather we use Manipula that in turn calls the DEP as an EDIT function.

Figure 6 shows an example of the folder structure and the survey folder from where the instrument is launched as a standalone environment over the network.

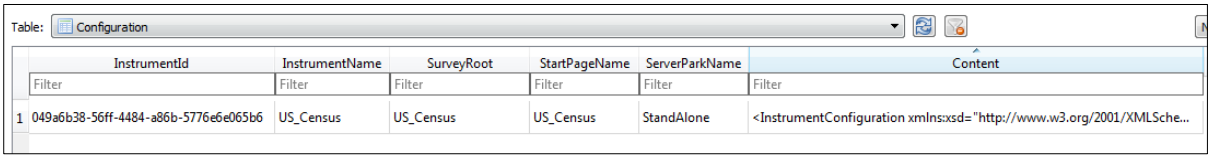
Figure 6 – Example of Blaise 5 Folder with 2 basic databases to run standalone



3.4.1 Runtime Configuration Database

This database has only one table - the Configuration table. This table stores data such as Instrument ID, Instrument Name, Survey Root (locations used to find the metafile and database), SeverPark Name, and Content. Figure 7 shows an example of the information stored in this table.

Figure 7. Example of Configuration Database Information



The "C**ontent**" is a string of XML text that contains the configuration of the instrument. Figure 8 shows an example of the data listed in the content column of the Configuration Table.

Figure 8. Example of Data contained in the "Content Column" of the Configuration Table


```
<InstrumentConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Version>1</Version><InstrumentName>inst</InstrumentName><InstrumentId>e0701098-4772-4d91-b2dc-281be9c22130</
InstrumentId><OldMetaFileName /><MetaFileName>C:\wincm\B5Surveys\Surveys\inst\inst.bmix</MetaFileName><ResourceFileName>C:
\wincm\B5Surveys\Surveys\inst\inst.blrd</ResourceFileName><DataFileName>C:\wincm\B5Surveys\Surveys\inst\inst.bdix</
DataFileName><InitialLayoutSetGroupName>CASI</InitialLayoutSetGroupName><InitialLayoutSetName /
><InitialDataEntrySettingsName>StrictInterviewing</InitialDataEntrySettingsName><SurveyRoot>inst</SurveyRoot><StartPageFileName>C:
\wincm\B5Surveys\Surveys\inst\inst.aspx</StartPageFileName><ServerParkName>StandAlone</ServerParkName><Status>Active</
Status><InstallDate>2016-08-10T14:01:54.000202</InstallDate><DataChecksum>4278984200.3387305133.650760389</
DataChecksum><CatiRole /><CatiSpecificationFileName /></InstrumentConfiguration>
```


3.4.2 Server Manager Database

The Server Manager Database contains seven tables that store critical information that is required to run the Survey. The tables store data related to:

- Deployment location
- Server name
- ServerPark name
- Port Number
- Master address
- Run Mode
- IP address
- Server Roles

When deploying a package through the Server Manager tool, the tool automatically fills the database tables with the required data. However, due to the challenge we encountered in running the Blaise 5 package outside the Control Center and the Server Manager application, we have deployed a utility to

perform the task of server Manager and alleviate the burden of moving several files to several devices. The utility or tool helps us to save time and resources during our test. Some of the important tables are discussed below.

- **LogicalRoot** Table - The critical "LogicalRoot" table contains the **location path** as well as the **instrument ID** that is used to map the instrument and location (See Figure 9).

Figure 9. Example of the Logical Root Table

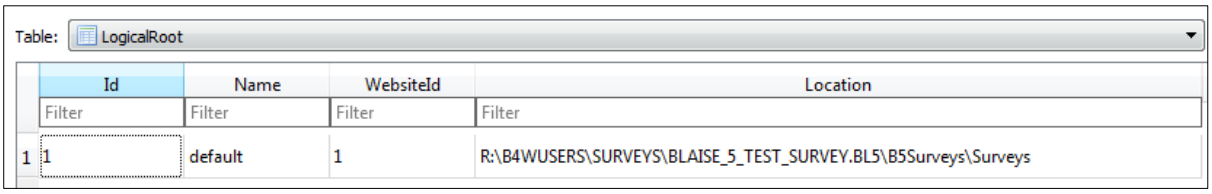


Table: LogicalRoot

	Id	Name	WebsiteId	Location
	Filter	Filter	Filter	Filter
1	1	default	1	R:\B4WUSERS\SURVEYS\BLAISE_5_TEST_SURVEY.BL5\B5Surveys\Surveys

- **Park Table** - The Park table is used to set some individual settings for the instrument. The **ID** is used to map to the right instrument and server name (See figure 10).

Figure 10. Example of the Park Table

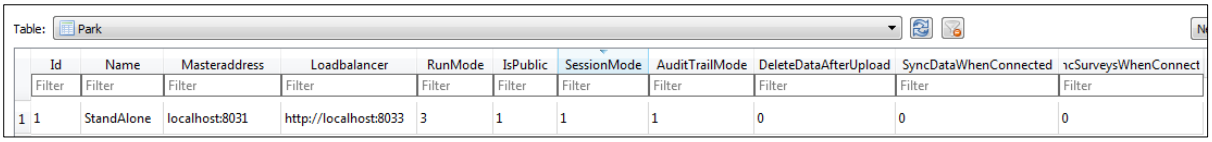


Table: Park

	Id	Name	Masteraddress	Loadbalancer	RunMode	IsPublic	SessionMode	AuditTrailMode	DeleteDataAfterUpload	SyncDataWhenConnected	ıcSurveysWhenConnect
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	StandAlone	localhost:8031	http://localhost:8033	3	1	1	1	0	0	0

- **Server Table** - The Server table maps the instrument via the **ParkID** with specific roles (See figure 11)

Figure 11. Example of the Server Table

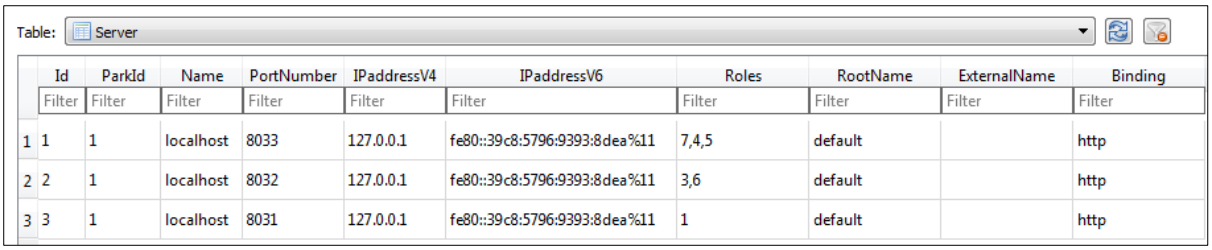
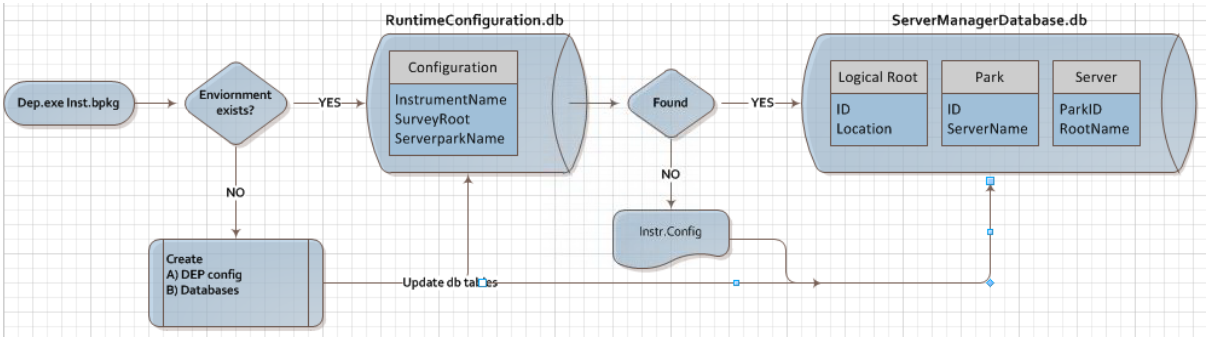


Table: Server

	Id	ParkId	Name	PortNumber	IPaddressV4	IPaddressV6	Roles	RootName	ExternalName	Binding
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	localhost	8033	127.0.0.1	fe80::39c8:5796:9393:8dea%11	7,4,5	default		http
2	2	1	localhost	8032	127.0.0.1	fe80::39c8:5796:9393:8dea%11	3,6	default		http
3	3	1	localhost	8031	127.0.0.1	fe80::39c8:5796:9393:8dea%11	1	default		http

The flow of this process as we see it happening is depicted in Figure 12. We may be off in the links across tables; however, this is pretty close to what we have observed when calling the DEP as standalone.

Figure 12. Example of the DEP flow to configure the environment as standalone



4. Putting all together

At the time of the writing of this paper, Manipula in Blaise 5 is still unable to call the DEP as a function. So, was decided that we should work with a hybrid approach of Manipula and DEP.

a) DEP is used to install the instrument (initial stage)
b) Manipula is used to synchronize data between our case management and the Blaise 5 case (as we currently do in Blaise 4)
c) DEP is used to launch the case once the update is complete

As a testing platform, this process worked. We used one single database with all the records (cases) loaded. The only inconvenience was to launch the DEP as initial stage in order to install the instrument. Subsequent calls were made properly by passing parameters such as the primary Key.

Introducing an external tool was necessary; the - "QLite_DbCreator" – its main mission is to create (QLite) databases needed by the DEP to run an instrument. The tool is responsible for creating the tables and filling them with data. This tools only dependency is the "System.Data.SQLite" library, which is the same library used by the DEP as well as the server manager.

4.1 Server Manager Tool

This section focuses on deploying a Blaise package as Standalone on a device with or without Blaise 5 installed. By default, the Server Manager is the primary tool used to deploy and administer Blaise 5 surveys. The Server Manager performs three major functions:

- Configure and manage the Server Park Servers
- Deploy surveys to Server Park
- Create and manage access to surveys

Like all the Blaise 5 components, the Server Manager is part of the Blaise installation. It is accessible through the Blaise 5 Control Center and can be accessed independently by opening the "ServerManager.exe" application from the Blaise 5 Bin folder.

To run Server Manager outside the Control Center or Blaise 5 Bin folder, all the reference libraries must be copied to the new location. Because the Server Manager depends on a range of libraries, that creates a challenge when copying or moving them into a new device or location. The challenge was to identify and copy the specific libraries to run the application. Identifying the reference library is time consuming. Copying all the libraries invoke resource and permission issues. The amount of files to copy range between 300 to 400 MB in size.

Not being able to identify the minimum dependencies to run the SeverManager, we decided to develop our own "ServerManager tool" to overcome this challenge.

4.2 QLite_DBCreator Tool

QLite_DBCreator.exe is a small C #.Net application that has the task of creating the required Blaise 5 SQLite databases and creating, updating and storing data in the database tables. Unlike the SeverManager tool, this application references only SQLite.dll as a dependent library and receives a list of parameters as input to produce the databases. The list of parameters includes a number following the task you want to achieve.

- To create a database,
  - a **[1]** <space> **[database name]** is entered as parameter.
- To create a table in selected database,
  - a **[2]** <space> **[database name]** <space> **[Table name]** is entered.

The application has the capability to create multiple db databases or tables simultaneously. QLite_DBCreator can be used manually and at the DOS prompt, to perform these tasks but this method was cumbersome. The ideal approach was to automate the process by including the application in the standalone installation package, and then call it through the VBScript.

Beside configuring and setting the enviornment, the VBScript would open a file called "instr.config" and extract information such as InstrumentName, InstrumentId, CheckSum, ServerParkName, and ParkId. The information is used to generate external files and update in the db database tables.

The VBScript calls the C# application and provides the required parameter using some internal procedure calls. The C# application proceeds in creating the databases and the tables, reads the external files, and loads data to the designated database tables.

Figure 13. QLite_DBCreator Tool code Example

```
createBlaiseDB db = new createBlaiseDB();
createTable tb = new createTable();
if (st[0] == "1" )
{
    /*Create one or more than database:
     * enter [1] <space>[database name]<space>[database name] <space>[…]
     */

    for (int i = 2; i < st.Length; i++)
    {
        databaseName = getDatabaseName(st[i].ToLower());
        if (databaseName != "")
        {
            databaseName = strPath + @"\"+ databaseName + ".db";
            db.CreateDb(databaseName);
        }
    }
```

```
//CreateDb gets one parameter<database name and fullpath>
//Checks if path and db exist. If path and db don't exist, it will it

public void CreateDb(string db)
{
    string directory = Path.GetDirectoryName(db);
    try
    {
        if (!File.Exists(db))
        {
            // Creates directory if it doesn't already exist:
            Directory.CreateDirectory(directory);

            // Creates file if it doesn't already exist:
            SQLiteConnection.CreateFile(db);
        }
        else
            Console.WriteLine(db + " already exists!");
    }
    catch (FileNotFoundException e)
    {
        if (e.Source != null)
            Console.WriteLine("IOException source: {0}", e.Source);
    }
```

4.3 The New Process

What we learned from researching the Server Manager and the DEP, enhanced our view and allowed us to refine our instalation instrument package and software distribution via a vbs instalation script. The installation package we use for the deployment of Blaise 5 instruments contains the following:

- **Blaise 5 Software** (Containing the distribution software in a zip file)**:**
  - Manipula.exe
  - Dep.exe
  - Dep.Resources.dll
  - System.Data.SQLite.dll
  - Two empty Blaise 5 databases:
    - RuntimeConfiguration.db
    - ServerManagerDatabase.db
  - Qlite_DbCreator.exe – C# software use for creating/managing databases.
- **InstallBlaise5StandAlone.vbs** - VBScript that prepares the enviornment; configuring, extracting software from zip, and instaling Blaise 5 survey instruments.
- **Inst.Config** - Configuration file for the survey that is being installed.
- **Inst.bpkg** - Survey package (instrument) .

Figure 14. Example of the Blaise 5 Instrument Installation Package files

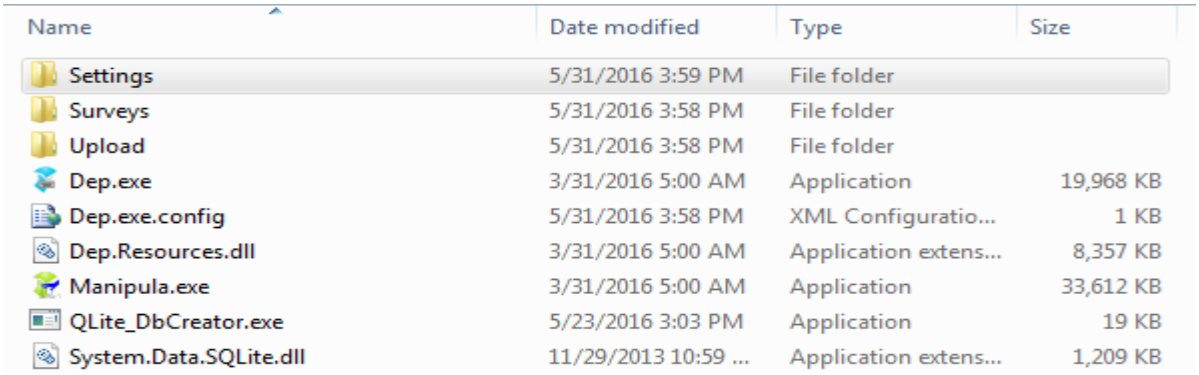
Name	Date modified	Type	Size
B_5.0.5.950_StandAlone.zip	5/23/2016 3:04 PM	WinZip File	29,034 KB
inst.bpkg	1/13/2016 12:04 PM	BPKG File	36,863 KB
InstallBlaise5StandAlone.vbs	6/2/2016 11:25 AM	VBScript Script File	26 KB
Instr.config	8/8/2015 10:16 AM	XML Configuratio...	61 KB

4.3.1 The Standalone Blaise 5 Installation VBScript

The Standalone Blaise 5 installation VBScript is the core element to setup the enviornment and Blaise instrument package. Its main responsibility is to create and configure the survey deployment location to install the package.

Listed below is the deployment folder structure created by the installation package. The VBScript unzips the file that contains the "Settings", "Surveys", and "Upload" folders and then copies all the required files (to run the survey instrument as standalone) and places them at the root level of the deployment folder

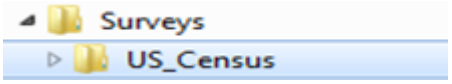
Figure 15. Example of Blaise 5 Standalone Folder Structure



This folder structure contains:

- **Settings** Folder - contains a list of databases
- **Surveys** Folder - holds subfolders for individual surveys. The subfolders can follow our current naming convention (i.e., use survey ID) to align with our current process (Fig 16).

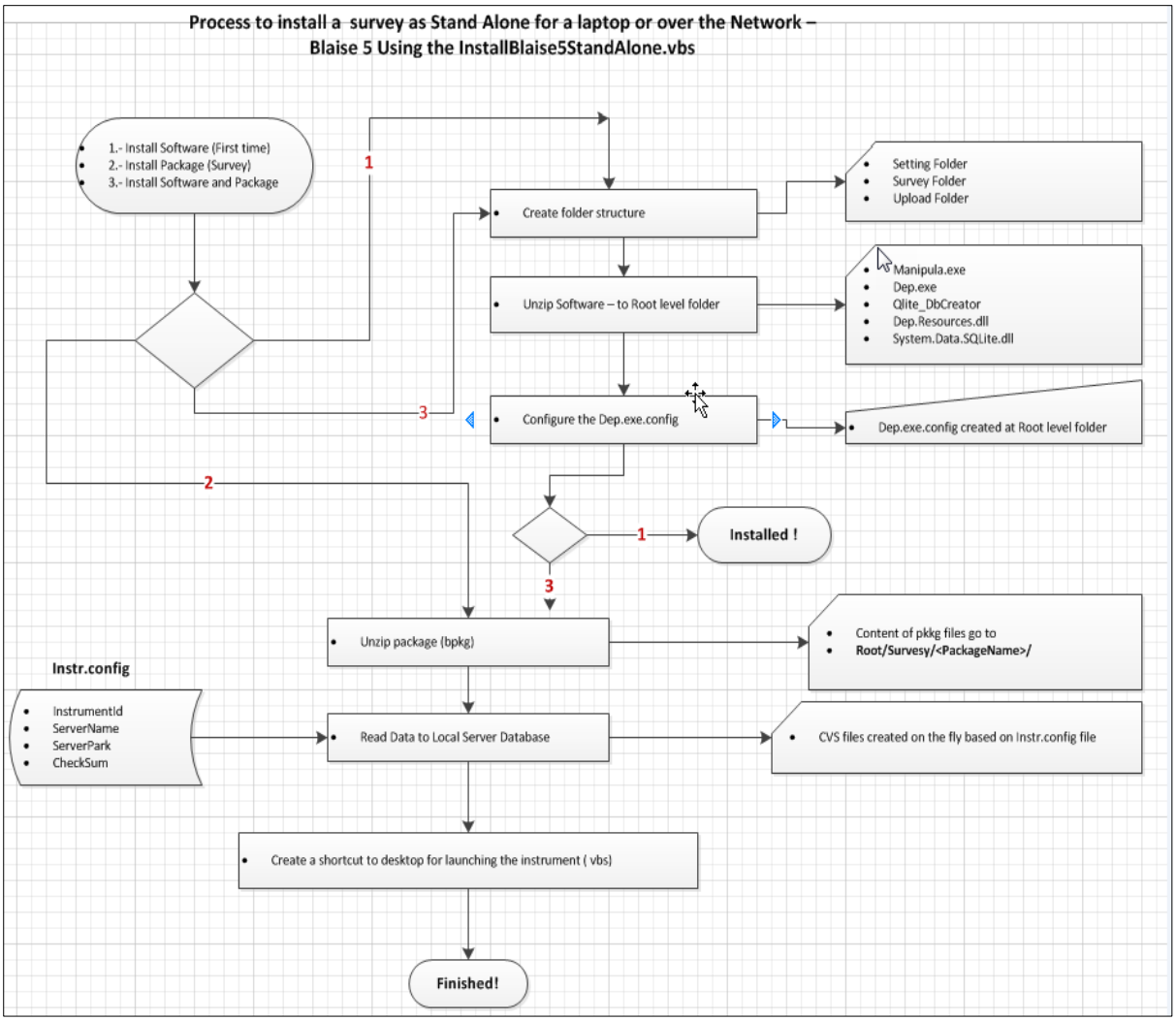
Figure 16. Example of the US_Census survey under the folder Surveys



- **Upload** Folder - stores a copy the package (.bpkg file)This is the folder where the packages would be placed.

The installation creates the "**inst.bpkg.vbs**" script. This script runs the survey as standalone when it is invoked. See Figure 17 for the flow of the VBScript instrument installation process.

Figure 17. Process describing how the VBScript installs and runs a standalone instrument



4.3.1 The Settings Folder

By default, Blaise 5 uses SQLite as the Database management System to store its database services. The SQLite Databases are part of the Blaise 5 installation and reside in the Settings folder.

There are a total of six databases in the Settings folder:

- RuntimeConfiguration
- ServerManagerDatabase
- RuntimeSessionData2
- Credentials
- ReportingCacheDatabase
- AuditTrailData

In our test, we use only two of six databases:

- RuntimeConfiguration
- ServerManagerDatabase

The two databases are critical to run the Blaise survey package. They store information related to the survey configuration, InstrumentID, InstrumentName, deployment location, Machine or Server name,

IP address, serverPark, and so on. They contain all that is necessary for the DEP to lauch a Blaise 5 survey effectively.

5. Summary

We were able to successfully install and run a Blaise 5 instrument as a standalone application that could possibly run under our current interviewing environment. If for some reason we must move a survey to Blaise 5 in our existing interviewing environment, we feel that this is something we may be able to handle. This process would also allow us to continue to support multiple versions of the software on the FR's laptop.

However, there are still a few remaining challenges that we hope to resolve with future Blaise 5 releases. We are currently unable to call a Blaise 5 instrument running an individual case database, as we do for Blaise 4. Also, we are not able to run a Blaise 5 instrument from Manipula.

Recommendations for future enhacements:

a) We would like the ability to extract the instrument ID and Checksum from the metafile by Manipula. This is currently done manually via the control center
   - The instrument ID is part of the project configuration setting
   - The Checksum is calculated when compiling/building the instrument

b) We think it would be beneficial to include any msux file (Manipula script) as part of the Blaise 5 package.

c) We would like to be able to bind the main metafile (bmix) with the database (bdbx) that is not the defaulting name. For intance, the F toggle in Blaise 5 means input file path indicating the input folder location. Whereas in Blaise 4, it indicates the name of the Blaise database. Currently there is no corresponding toggle to indicate a database (bdbx) name in the command line.

d) Perhaps reducing the number of dependencies of the ServerManager can help. The ServerManager is a great tool that can be run at the commnad line prompt, allowing us to automate our process.

6. References

Peter Segel, Mangal Subramarian, Richard Frey, Ray Snowden (Westat) "Blaise 5 Server configuration for Web Surveys", Proceedings of the 16th International Blaise User Conference, April 2015.

Blaise 5 Help (version 5.0.5.950). (2016).

7. Acknowledgements

The author would like to acknowledge the work and knowledge of Mecen Desormice for his assistance with research project.

The views expressed in this paper are those of the authors and not necessarily those of the U.S. Census Bureau.