Blaise 5 with RTI's Integrated Field Management System on Field Interviewer Laptops

Lilia Filippenko, Preethi Jayaram, Joe Nofziger, Brandon Peele, R. Suresh - RTI International

1. Abstract

For many years RTI's Integrated Field Management System (IFMS) is a standard system to use for CAPI projects on laptops for instruments developed in Blaise and other CAI software used by RTI. The laptop Case Management System (CMS) is configured to launch the specific CAI software as required by the project. Since Blaise 5 uses a different approach compared to Blaise 4 to install and launch Blaise instruments on laptops, we needed to adapt our CMS to support Blaise 5. We tweaked our CMS and upgraded the associated Manipula scripts, and now RTI's IFMS can support Blaise 4, Blaise 5 and other CAI software as needed. The paper will describe the process we followed and the challenges we encountered during this development.

2. Overview of Integrated Field Management System (IFMS)

IFMS is used for almost all field studies conducted by RTI and is a web-based application responsible for electronic assignment and transfer of cases to field staff, standard case status reports, data transmission, field monitoring, interviewer production, and laptop case management. IFMS is set up for every project regardless of the CAI software used on Field Interviewer (FI) laptops.

The general approach for field studies is as follows:

- A Master database for all cases of the study resides at RTI and is used to create a zip file for every case in the study with preloaded information
- Each case is assigned by a Field Supervisor to a Field Interviewer using the IFMS website, so cases are distributed among laptops
- When an FI laptop connects to RTI, cases assigned to him/her are transmitted from RTI and loaded in the laptop's database. It could be MySQL, Blaise, or SQLite database.
- If a case needs to be assigned to another interviewer, the Field Supervisor initiates an order on the IFMS website to transfer a case and its data files are transferred between laptops.
- When an interview for a case is completed, its data file is sent back to RTI for loading into the Master database.

Although many special functions are used by IFMS, just a few of them are specific to a software package and need adjustments, such as export/import case data from/to Master/Laptop databases; update status of the case on laptop set by the instrument or entered by FI; and launch the instrument.

3. Overview of Case Management System (CMS)

Before the start of data collection for a project, all necessary software and instrument files are loaded on laptops at RTI. At RTI International a .Net Case Management System (CMS) is used on laptops. The CMS application allows Field Interviewers to update case status, enter comments, launch instruments, and synchronize the status of cases with a centralized SQL server database. Various software packages or languages are used to develop instruments, including Blaise, Hatteras, and others. The type of software

package is stored in configuration files and is used by the CMS to create and invoke an appropriate CAI package object to manage work with the individual package. The CAI package is a software abstraction that hides the details of the language/software from the CMS, allowing it to perform essential operations on it without knowledge of the underlying implementation. Creating a new CAI Package class allows the CMS to work with new or custom survey administration tools with minimal change.

Methods implemented by the CAI package and applied for every case on the laptop include:

- **Import** (load a case into the laptop database)
- **Update event and status** codes for a case to the instrument database
- **Invoke** an instrument for a case
- **Export** (extract a case from the laptop database for transfer)

To work with Blaise 5, a new CAI package was added to the CMS. It was copied from the Blaise 4.8 CAI package and changes were made to pass modified or new parameters to accommodate requirements for Blaise 5. The goal for moving to Blaise 5 was, and still is, to maintain the same functionality as before with Blaise 4 while take advantage of many new Blaise 5 features. By maintaining the Blaise 4.8 CAI package alongside the new one for Blaise 5, the CMS continues to support both versions.

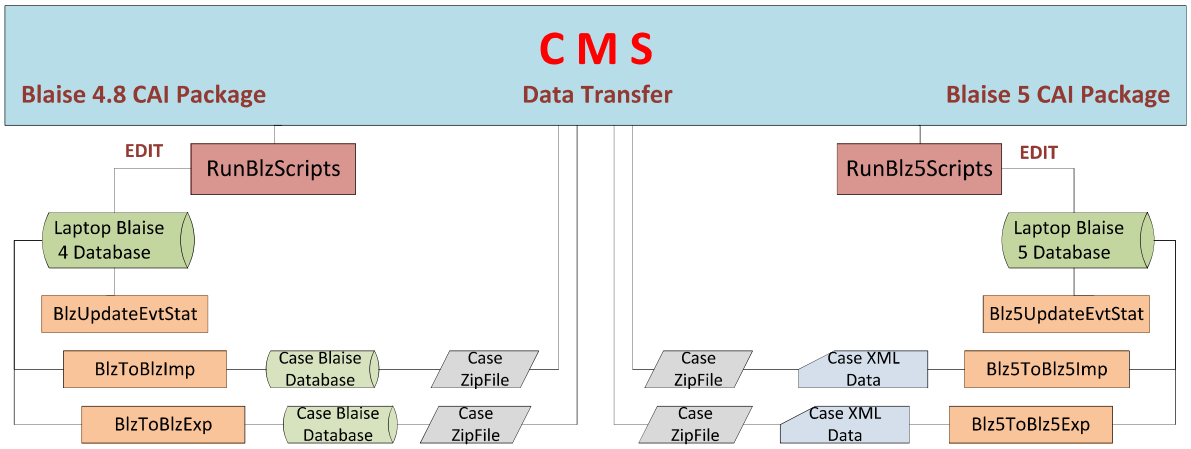
This paper will describe implementation of these methods with Blaise instruments and challenges with applying them to Blaise 5 instruments.

4. CMS with Blaise Instruments on Laptop

4.1 Manipula Process and Setup Scripts

For Blaise instruments, the CAI package invokes a single main Manipula process script, passing a parameter to execute the requested method by calling corresponding Manipula scripts. Our Manipula-driven architecture is illustrated in Figure 1. The CMS handles transfer of encrypted case zip files to and from FI laptops. The Blaise 4.8 CAI package is shown on the left side and the Blaise 5 CAI package on the right. Each executes the appropriate main Manipula process script, which in turn calls other Manipula setups. The same set of Manipula scripts is used by the CAI packages, with small differences in parameters and type of input/output files.

Figure 1. CMS with Manipula Scripts on Laptop



The main Manipula process script is developed in a general way so that it can be prepared independently of the Blaise instrument with which it is used, and without changes to the CMS. This approach gives us the flexibility to adjust requirements for different studies as needed. An example of this is the ability to call an external application before conducting the interview or to spawn new cases after completion of the interview without returning to the CMS.

An abbreviated example from the main Blaise 4 Manipula process script used for current projects in the field is provided in Appendix A. It shows the calls to Manipula setups to load a case, extract a case, update event and status code in the Blaise database, and a direct call of the Edit function to start Blaise instrument. In the CMS a command line is constructed to start Manipula.exe with parameters where the type of the method to use is defined as the first parameter. The name of the data model and database is also passed as a parameter. Other parameters are the case ID, a study identifier, and a folder name to use.

4.2 Challenges in converting Manipula Scripts from Blaise 4 to Blaise 5

We successfully converted all our Blaise 4 Manipula scripts and just modified "EXPORT"/"IMPORT" Manipula setups to use file type "XML" for output and input respectively. We also did small changes to the list of parameters and how they were passed to Manipula scripts. To use the EDIT function a name of the package file was used instead of an instrument name, and run mode was specified as "ThickClient". But we discovered a few unexpected things and found that we needed to make more changes to Manipula scripts for Blaise 5. Some of challenges encountered are described below.

4.2.1 Additional Data Model Files

When we built a solution containing all our Manipula scripts, we noticed that for data models defined inside Manipula scripts in Blaise 5, prepared data models ending with "$$$.bmix" were created. For example, to create an ASCII file to log a result of different steps during the execution of RunBlz5Scripts, we use the data model "Info" defined in the RunBlz5Scripts.man as shown in Figure 2. After building RunBlz5Scripts.msu, we noticed that the file "runblz5scripts_Info$$$.bmix" was created. If the same datamodel was used in a few scripts, a new "$$$.bmix" was created each time.

Figure 2. Example of Data Model to Create Log File

```
DATAMODEL Info
FIELDS
      date : string[8]
      sp1 : string[1]
      time : timetype
      sp2 : string[1]
      name : string[50]
      sp3 : string[1]
      success : string[7]
ENDMODEL
```

As it turns out these files are needed on the laptop to successfully run Manipula scripts. We decided to create data model files and use them directly in Manipula scripts as shown in Figure 3. They are also added to the solution with Manipula scripts for installation on laptops.

Figure 3. Example of Code to Output Log File

```
USES
   MoveCase 'MoveCase'
   Info 'Info'
//---More code---
OUTPUTFILE file2:info('info.log',ascii)
   SETTINGS
       MAKENEWFILE=NO
//---More code---
PROCEDURE InfoFile
    PARAMETERS
        Import intReslt:integer
        Import strText:string
    INSTRUCTIONS
        IF intReslt<>0 THEN file2.success:='Failed '
        ELSE file2.success:='Success'
        ENDIF
        file2.date:=DATETOSTR(SYSDATE, 'MMddyyyy')
        file2.time:=sysTime
        file2.name:=strText
        file2.write
ENDPROCEDURE
//---More code---
    InfoFile(Reslt,'Run Interview for ' + strCaseId)
```

Starting with Blaise version 5.4 an additional file, Manipula runtime meta "_locals$$$.bmix", is created for a Manipula script that should be installed along with the ".msux" file on laptops. The number of files to install on laptops is therefore doubled, since for Blaise 4 only the ".msu" files are distributed on laptops to call Manipula scripts.

4.2.2 Blaise Data Interface File for ASCII and XML Files

During testing in the developer environment, we noticed that a Blaise Data Interface File (.bdix) was created in addition to the ASCII or XML file. It looked like it could be helpful to see the data especially for an XML case data file created by the "EXPORT" script and transferred to RTI. On the laptop all scripts were also working fine when we tested with version 5.3.0.1501.

When we started testing Manipula scripts upgraded to Blaise 5.4 on laptops, using version 5.4.2.1669, an error message was thrown that the "bdix" file cannot be created without a license. It was not clear why we needed a license with the new version and whether it was related only to this file. We contacted the Blaise support and received a suggestion to use special setting in our Manipula scripts to not create that file. As a result, all our scripts now have this setting: "CREATEBDIX=NO". It is good, though, to have that file for in-house review of XML case data files using the Blaise Data Viewer.

4.2.3 Production and Training Cases

With versions of Blaise before Blaise 5, RTI's standard practice was to distribute instruments on laptops into two folders – a folder where the production interview data was collected and a folder where FIs could be trained to work with the instrument. Therefore, special cases were prepared and loaded on laptop in "Train" folder along with a Manipula script used to reload those training cases if necessary. With Blaise 5 only one instance of the instrument can be deployed on the laptop, so we needed to make a change to require a special field present in all of the Blaise 5 instruments – "modeID". Manipula setups to load data

in the Master database at RTI have this field loaded as "4" (production mode), and Manipula scripts to load training cases use a value of "3". With Blaise 4 when FIs need to have all training cases reinitialized, the training database is simply deleted by the CMS and Manipula script to reload training cases is called. To reinitialize training cases with Blaise 5, we needed to modify Manipula script to remove only the training cases from the Blaise database and then load them again.

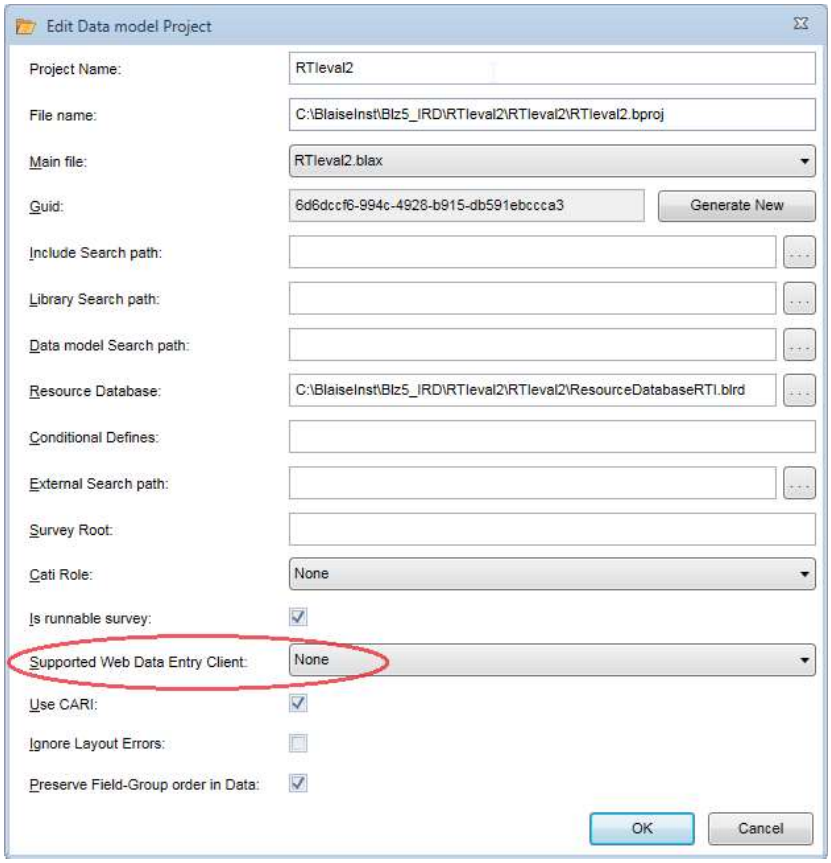
4.3 Blaise 5 Instrument Deployment as "Stand Alone"

When the "Edit" method to start a data entry session was added to Manipula in Blaise 5.4, we started using Manipula.exe to invoke Blaise instruments from a Manipula process script. Blaise is not installed on our laptops and only a few supporting files are needed. These additional files and settings to build a Blaise 5 instrument for running as "Stand Alone" on a laptop are described below.

4.3.1 Building a Package to Deploy on Laptop

To distribute a Blaise 5 instrument on Field Interviewer laptops, a package (.bpkg) should be built in the Control Center and the project settings must be changed so only needed files are included in the package. This is done by selecting "Edit Project" from the dropdown menu in the Solution Explorer and the "Edit Data Model Project" dialog will open. The value of the "Supported Web Data Entry Client" should be set to "None" as shown in Figure 4 to have only basic instrument files included in the package.

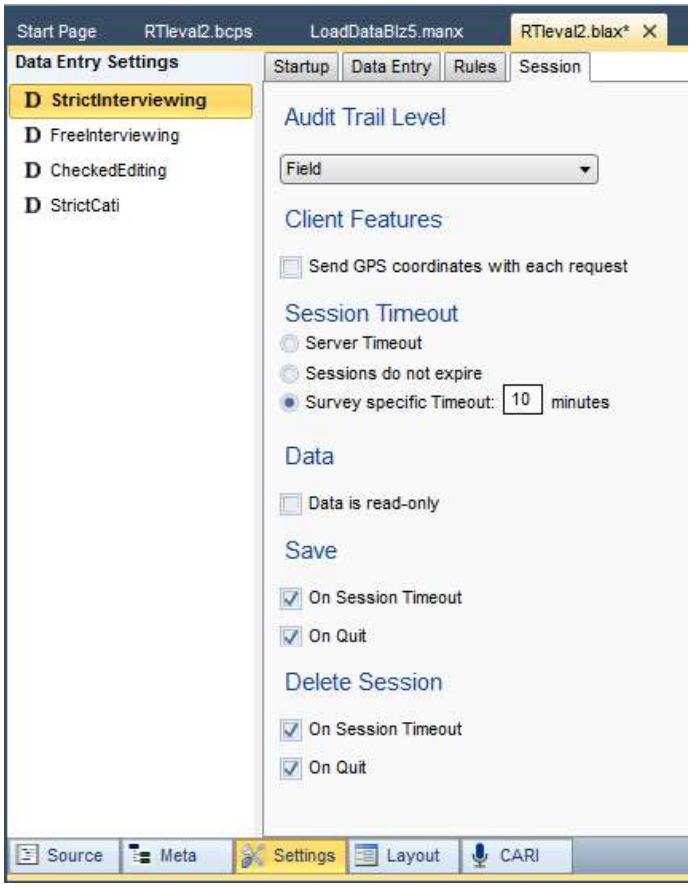
Figure 4. "Supported Web Data Entry Client" Value in Data Model Project



Although it was recommended to set the Session Timeout in the session tab of the applied Data Entry Settings to "Sessions do not expire", we decided to use "Survey specific Timeout" as shown in Figure 5.

With these settings, collected interview data is saved in the Blaise database and deleted from the Session database regardless of the interview status - completed or partial. This is critical if partial case need to be transferred to a different FI and our Manipula scripts use the Blaise database to export a partial case.

Figure 5. Setting for Session Timeout



4.3.2 Programs needed on Laptop

During the CMS installation on laptop, folders are created as specified in configuration settings for the project. For all projects a folder is created for programs that are needed for the project, and it is used later by the CMS to run any specific project's program from that folder. The following programs and components were copied into this folder on the laptop:

- Manipula.exe
- Manipula.exe.config – Configuration file with deploy folder value as "C:\Blaise5\StandAlone"
- System.Data.SQLite.dll – SQLite library
- Msvcr100.dll – Microsoft Visual C++ Runtime library
- Survey package file (.bpkg) build as described above in 4.3.1
- Batch file to start a Manipula script to install the package
- Manipula process script to select the package and install it

The first four files were copied from the Blaise installation folder '...\StatNeth\Blaise5\Bin' on the developer's computer. The .NET Framework (4.5.1 or higher) is also required for running Blaise 5 instruments and is already installed on RTI laptops, as it is needed by the CMS.

The Manipula process script (Figure 6) and a batch file to run it can be used to install any survey used by the study.

Figure 6. Manipula Process Script to Install Survey

```
PROCESS InstallSurvey "Install Survey on Laptop"
AUXFIELDS
  Survey: STRING[100]
MANIPULATE
  Survey:= SELECTFILE('Select survey','', 'Blaise Package (*.bpkg)|*.bpkg')
  INSTALLPACKAGE(Survey)
END
```

The batch file has the following code:

"Manipula.exe InstallSurvey.msux".

After a successful run, instrument files are installed in "C:\Blaise5\StandAlone\Surveys\<Survey name>".

4.4 Installation of Manipula Scripts

Our Manipula scripts to run with the CMS are standard; even the name of the instrument in them is the same. With Blaise 4 we use a special program to prepare an installation package with instrument files and Manipula scripts to install on the laptop by running single batch file. As a first step that program uses "b4cpars.exe" to prepare Manipula scripts for a data model passed as parameter. Unfortunately, there is no way right now to use the same approach with Blaise 5. We need to use the GET method in some Manipula setups, so we cannot use the reserved word VAR to pass name of the data model at run time. Although Instrument Builder can be used to prepare all Manipula scripts in the solution at once, we still need to manually change the name of the data model in such Manipula setups.

After Manipula scripts and the data models used in them are prepared, they are copied to the folder where the instrument files were installed by the "InstallSurvey" Manipula script. We hope that in the future we will have something like with Blaise 4 when scripts could be prepared without any changes.

4.5 Audit data to extract

With Blaise 4 we have a configuration file on the laptop to create an audit file for every case and it is transmitted to RTI along with the interview data for a case. In Blaise 5, Audit data is saved in the SQLite database for all cases. Our goal now is to have a Manipula process script that will extract audit data for a case when the case is exported after it is completed or is partial and needs transfer to a different FI.

5. Conclusion

Most of our work is done and we are ready to use Blaise 5 for CAPI studies. Still ahead is to modify some ancillary tools we have developed for working with Blaise 4. These tools include defining procedures during production to do (1) changes to a data model that are harmless to data and therefore suitable for automatic adjustment and (2) changes which cause data file incompatibility. We plan to modify an existing program to apply upgrades to production Blaise 5 databases. It will use the same approach as for Blaise 4 databases, creating backup folders and Manipula scripts to upgrade the Master database and then

the database on the laptop. We will also evaluate tools in Blaise 5 to prepare SAS datasets for delivery, or upgrade our "BlaiseToSAS" application to use with SQLite and SQL Server databases.

With some reading and experimentation, we have learned several differences in laptop installation, script invocation, and storage between Blaise 4 and Blaise 5. Our existing infrastructure and componentization lessen the impact on us. We hope the implementation changes we have documented here will help others to adjust their processes.

6. References

Blaise 5 Help Reference Manual, Statistics Netherlands
http://help.blaise.com/

Blaise 5 Tutorials, Statistics Netherlands, "Blaise 5 Deploying DEP Stand Alone", 2016
"Libraries\Documents\MyDocuments\Blaise5\Tutorials\Data Entry\Deploying_DEP_Stand_Alone.pdf"

Appendix A. Example of Blaise 4 Manipula Script "RunBlzScripts.man"

```
strScriptName := uppercase(PARAMETER(1))

CASE strScriptName OF

  'IMPORT':
     strWorkF := PARAMETER(2)
     strCaseId := PARAMETER(3)
     strOutFile := PARAMETER(4)
     strInstName := PARAMETER(4)
     strProjectID := PARAMETER(5)

     LogFile('BlzToBlzImp BEFORE ' + PARAMETER(3))
     strRun:= 'BlzToBlzImp /W' + strWorkF + ' /I' + strCaseID + ' /O' + strOutFile + ' /P' + PARAMETER(3) + ' /Q'
     Reslt:= CALL(strRun)
     LogFile('BlzToBlzImp AFTER  ' + PARAMETER(3))

     IF Reslt<>0 THEN
        file1.success:='Failed '
     else
         file1.success:='Success'
     ENDIF
     file1.date:=sysDate
     file1.time:=sysTime
     file1.zrid:=parameter(3)
     file1.imp:='I'
     file1.write

  'EXPORT':
     strWorkF := PARAMETER(2)
     strInstName := PARAMETER(3)
     strOutFile := PARAMETER(4)
     strCaseId := PARAMETER(4)
     strXfer := PARAMETER(5)
     strProjectID := PARAMETER(6)

     strParam :='/P' + strCaseID + ';' + strXfer
     strRun:= 'BlzToBlzExp /W' + strWorkF + ' /I' + strInstName + ' /O' + strOutFile + ' ' + strParam + ' /Q'
     LogFile('BlzToBlzExp BEFORE ' + PARAMETER(4))
     Reslt:= CALL(strRun)
     LogFile('BlzToBlzExp AFTER  ' + PARAMETER(4))

     IF Reslt<>0 THEN
        file1.success:='Failed '
     ELSE
        file1.success:='Success'
     ENDIF
```

```
      file1.date:=sysDate
      file1.time:=sysTime
      file1.zrid:=parameter(4)
      file1.imp:='E'
      file1.write

   'UPDATE':
      strWorkF := PARAMETER(2)
      strInstName := PARAMETER(3)
      strCaseId := PARAMETER(4)
      strProjectID := PARAMETER(9)
      strParam := '"' + '/P' + strCaseId + ';' + PARAMETER(5) + ';' + PARAMETER(6) + ';' + PARAMETER(7) + ';' + PARAMETER(8) + '"'
      strRun:= 'BlzUpdateEvtStat /W' + strWorkF + ' /I' + strInstName + ' ' + strParam + ' /Q'
      Reslt:= CALL(strRun)
      InfoFile(Reslt,'BlzUpdateEvtStat for ' + strCaseId)

   'INVOKE':  {do not need special script}
      strWorkF := PARAMETER(2)
      strInstName := PARAMETER(3)
      strProjectID := PARAMETER(5)
      IF PARAMETER(4)<>'' THEN
         strCaseId := ' /K' + PARAMETER(4)
      ELSE
         strCaseID :=''
      ENDIF
      {Custom code to create lookup table with CW names who completed OC section}
      IF (Uppercase(strInstName)='CSWRKR') AND (strCaseId<>'') AND (strProjectID='14780') THEN
         strRunTemp :='C:\NSCAW\PGMS\CreateCWnamesDB.exe'
         IF FILEEXISTS(strRunTemp) THEN
             {get value of PSUID for the case}
             strTemp := PARAMETER(4)
             strPSUID := SUBSTRING(strTemp,2,3)
             strRunTemp := strRunTemp + ' /PSU=' + strPSUID + ' /' +strTrainProd
            Reslt:= run(strRunTemp, WAIT)
            InfoFile(Reslt,'CreateCWnamesDB')
                  ENDIF
      ENDIF
      strRun:= strWorkF +'\' + strInstName + ' /G /X /C'+ strInstName +'.diw @' + strInstName + '.bcf ' + strCaseId
      IF strInstName<>'' THEN

          Reslt:= EDIT(strRun)  {Start Interview}

           InfoFile(Reslt,'Run Interview for ' + PARAMETER(4))
           {Custom program to create cases for RECS - 14615 }
           IF ((Uppercase(strInstName)='RX2015') AND (strProjectID='14615')) THEN
               strTemp:= GETVALUE(strInstName,strInstName,PARAMETER(4), 'main_case.sum_stat')
               IF (strTemp = '2690') THEN
                   strRun:= 'C:\RECS\pgms\CreateCasesNet.exe'
                   IF FILEEXISTS(strRun) THEN
                       DISPLAY('Wait one moment please...' )
                       {Create RECS cases}
                       strRun:= 'C:\RECS\pgms\CreateCasesNet.exe /ID=' + PARAMETER(4) + ' /' + strTrainProd
                        Reslt:= run(strRun, WAIT)
                       InfoFile(Reslt,'Run Create Cases for ' + PARAMETER(4))
                   ENDIF
               ENDIF
           ENDIF
              ELSE
           InfoFile(1, 'Run Interview for empty InstName')
      ENDIF

   'RESETT':
      strWorkF := PARAMETER(2)
      strInstName := PARAMETER(3)
      strOutputF := PARAMETER(2)
      strProjectID := PARAMETER(5)
      {PARAMETER(4) is a script name do load data in training DB}
      IF FILEEXISTS(PARAMETER(4)) THEN
          strRun:= PARAMETER(4) +' /W' + strWorkF + ' /O' + strInstName + ' /T' + strOutputF + ' /Q'
          Reslt:= CALL(strRun)
          InfoFile(Reslt,'Reset Train Cases')
      ENDIF

ENDCASE
```