Unduplication of Listing Data

Michael K. Mangiapane, U.S. Census Bureau

1. Introduction

The Survey of Construction Listing Instrument (SOC LI) is a CAPI instrument that collects data used to measure new residential home construction in the United States. For this survey, Census Bureau Field Representatives (FRs) visit Building Permit Offices (BPOs) and collect a table of up to 2400 building permits and their associated information. A running pre-sample of the permits is flagged by the instrument to collect extra information such as the location of the residence, the builder of the residence, the owner of the residence, and physical characteristics of the residence. When the listing is completed, the instrument generates a final sample of the permits, and new cases are created from that sample for follow-up using the SOC Questionnaire Instrument (SOC QI), which captures more detailed characteristics of the new residence being built.

Originally a PAPI survey, the instrument was first converted to CAPI in 1997 and used CA Clipper software for the data collection and sampling functions. When the U.S. Census Bureau started to convert its other surveys from CASES or Clipper to Blaise, research was conducted on converting the SOC LI as well. From June 2005 through May 2008 a major initiative was undertaken to convert the SOC LI. It resulted in a Blaise 4.7 instrument that was successfully put into production and is still in use today. One requirement of the instrument is to check a newly entered permit number to make sure it is not a duplicate of any previously entered permit numbers since they are supposed to be a unique identifier. This process is referred to as unduplication. The listing instrument should be able to check for duplicates in a reasonable amount of time, prompt the user that it is a duplicate, and give them the option to correct their entry. Since listing forms can be fairly large it would be relatively easy for an FR to lose track of where they are on a printout and start keying in a permit that was already entered in the case. The Clipper version of the SOC LI was able to perform unduplication immediately after entering a permit number. This functionality was requested for unduplication in the Blaise version since it was something the FR's were used to, and it made sense to check at that point.

1.1 Survey Design

The SOC LI started development in Blaise 4.7.0 b957 and was fielded in Blaise 4.7.1 b1168. It contains a listing table of 2400 lines, each one containing 43 fields. Each permit listed always has a permit number, the date the permit was issued, and the number of housing units. If a permit falls into the pre-sample or final sample, the FR will collect the physical location of the new residence, the builder or owner of the residence, and a respondent category. The FR has the option to collect additional information about a listed permit that falls into sample while they are listing permits during the pre-sampling or after the final sample is run.

The listing instrument also includes two additional "instruments" that were originally written in CASES software. The first is the Questionnaire of Building Permit Officials (QBPO) which collects information about how building permits are recorded in the BPO. The second is the Little Questionnaire Instrument (LQI) which is a subset of the SOC QI and collects physical characteristics of a new residence such as the size of the residence, size of the lot, number of bedrooms, bathrooms, and so on. These instruments are both parallel blocks in the listing instrument and are available as needed.

1.2 Hardware and Software Used

The hardware that the SOC LI currently runs on is a Dell Latitude D400 laptop with an Intel Pentium M processor running at 1.4 GHz and 512 MB of RAM. Microsoft Windows 2000 Service Pack 4 is the primary operating system. As mentioned before, Blaise 4.7.1 b1168 is the version of Blaise currently used in the production version of the SOC LI. All FR's in the field use this software and hardware configuration and all testing times noted in this paper are based on these specifications.

2. Objectives of the Paper

This paper will detail the approaches that were taken to meet the requirements of unduplication, the challenges that were faced within each approach, and the results of prototyping unduplication. The approaches included:

- **Using a procedure** within the listing table that is called after the permit number is entered to compare the current line to all of the lines above it using strings.

- **Using a Maniplus script** that would run after the user indicates they are done listing, using loops to compare a line with all of the lines below it.
- **Using the Blaise API** to call a Visual Basic application that is connected to the instrument via an alien router to perform unduplication using a hybrid of the procedure comparison and the loops of the Maniplus script.

There will also be a discussion of why we decided to use the Blaise API for unduplication and bundling it with a helper program for the SOC LI, the SONC Builder Table. This decision also allowed unduplication to be used in similar listing instruments that were also being converted to Blaise at the same time with minimal changes to the design of the instruments.

3. Unduplication Using a Procedure

One approach for accomplishing unduplication as soon as a permit number is entered is to call a procedure in the rules and let it handle doing the comparison. Since data does not need to be stored from the comparison, and we were aiming to keep the listing rules as simple as possible, it made sense to put the comparison into a procedure. The search itself only looks at permit numbers that are listed on the lines above the newly entered permit number since it is expected that if a number is a duplicate, the incorrect number is the one that was just entered (or the "second" permit number). This also applies even if the FR backs up and edits a permit number. If a duplicate number is found, the FR is prompted with a hard edit check that a duplicate was found and they need to make a correction.

3.1 Procedure Design and Challenges

Since the SOC LI listing table is an array of blocks, one could structure the call to the procedure inside a loop to reach back to previously listed permit numbers. However, that would generate an internal parameter to connect to each line that the procedure has to look at. In this design the instrument would slow down immensely as additional permits are listed since internal parameters are inefficient compared to declared parameters. To work around this we decided to store permit numbers in a string at the table level and pass them into the individual block and then into the procedure. To store 2400 permit numbers of up to 24 characters long and also have the comparison list be comma-delimited, it takes a string of 60,000 characters. Blaise's current limit for a string is 32,767 characters so two strings are required to hold all of the permit numbers.

After a permit number is entered, a procedure is called to trim any leading or trailing spaces from the current permit number to make the comparison accurate. The unduplication procedure is called and the current line number, current permit number, and the strings of the previously entered permit numbers are passed in. The strings are put into an auxfield that uses the OPEN type so that the comparison only has to be done on one list rather than two separate lists. The length of the first permit number in the list is calculated based on finding the position of the first comma and a loop is started up to do the comparison and step through the list. Each listed permit number in the combined field that comes before the current permit number is checked against the current permit number. If it is not a duplicate, the procedure moves on to the next listed permit number by calculating new start and end positions in the list to pick up the next listed permit number. This continues until the procedure reaches the end of the loop (based on the line number passed in) or until a duplicate is found. The following Blaise code is a snippet of how the unduplication procedure searches for duplicates:

```
temp_Num := Line_Num - 1
temp_PermNum := ''
CombinedField := AllStrings + String2

IF temp_Num = 0 THEN
   BeginPos := 1  {Move on to the next permit}
   EndPos := LEN(CombinedField) {Find the next permit}
ELSE
   BeginPos := 1  {Default to the beginning of the string}
```

```
   {Find the end of the first permit number}
  EndPos := POSITION(',',CombinedField,BeginPos)


   FOR I := 1 TO temp_Num DO
      temp_PermNum := SUBSTRING(CombinedField,BeginPos,EndPos-BeginPos)
      IF Permit_Num = temp_PermNum THEN
         ERROR "@Zs@Z@LA permit number duplicating a previously
                entered permit number has been detected.
                @/Return to the listing and correct the first permit
                number or second permit number@L"
      ENDIF


      BeginPos := EndPos+1  {Move on to the next permit}
      EndPos := POSITION(',',CombinedField,BeginPos) {Find the next
                                                      permit}
   ENDDO {I := 1 TO Line_Num}
ENDIF {temp_Num = 1}
```

If a duplicate is found, a hard error is displayed to the FR informing them that the current permit number is a duplicate of a previous entry (See Figure 1).

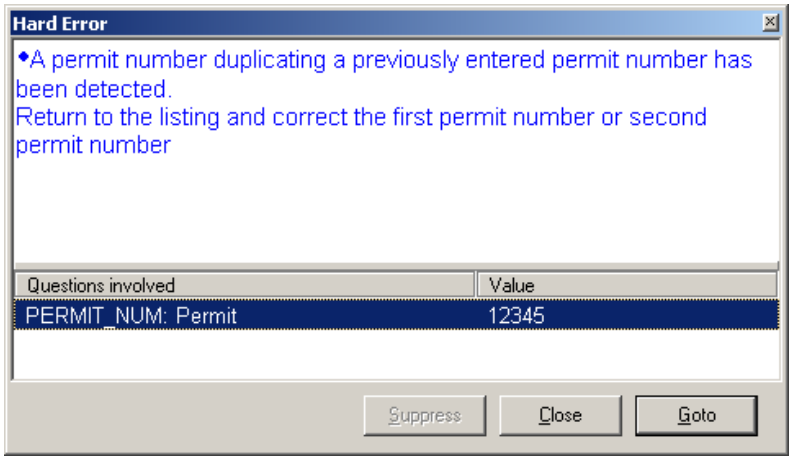

Figure 1: Unduplication error message in the Blaise procedure.

Taking this approach meant that unduplication would run immediately after the permit number was entered. This satisfies the requirement that the FR is prompted as soon as they enter a duplicate permit number. They would be able to correct the duplicate entry and move on without an issue.

3.2 Prototyping and Testing

While testing the SOC LI prototype with this procedure an issue developed as more permits were listed. If a user quit a case before it was complete and then re-entered the case to finish it, the load time to start the DEP would increase depending on the number of permits that had already been listed. Jumping into some parallel blocks and back to the main path of the SOC LI also increased, making it look like the instrument had "frozen." While this was not noticeable when the number of permits was from 1-200, the lag time became unacceptable for longer listings.

Number of Permits	Instrument Load Time	Table Load Time
100	2 seconds	0 seconds
200	3 seconds	3 seconds
500	12 seconds	10 seconds
1000	56 seconds	54 seconds
2399	9 minutes 40 seconds	9 minutes 46 seconds

It was expected that there would be some lag with longer listings, but we did not expect the lag to become as long as it did within a small number of permits. Surprisingly, unduplication itself was still instantaneous within the table. However, it just took far too long to be able to open up the instrument and get to the listing table and this would have been unacceptable to the FRs. Unduplication using a procedure would be appropriate if the SOC LI was limited to a few hundred permits, but that is not possible with the current design and limits of the survey.

Advantages of using a procedure:

- Checks for duplicates as permits entered.
- "Instantaneous" check in listing table.
- Easy implementation.

Disadvantages of using a procedure:

- Huge instrument load lag for large listings.
- Lag introduced when navigating parallel blocks.

4. Unduplication Using a Maniplus Script

Another approach we tried is to take unduplication outside of the instrument and put it into a Maniplus script. Using Maniplus reduces the size of the instrument and makes the instrument more efficient since the instrument does not have to hold a list of the permit numbers and constantly run unduplication on previous lines when the rules are re-executed. Unlike the procedure, the unduplication script checks all of the permit numbers ahead of the permit number that is being examined, though it would be simple to reverse the process to make it more like the procedure.

4.1 Design and Challenges

The main challenge with this approach is that the script had to be associated with an "action" so it would not automatically run after the permit number was entered. Instead, this script ran after the FR exited the listing table via a parallel block. If the FR indicates they are done with listing, the unduplication script is called as one of the actions performed when leaving the parallel block. After determining the last listed line of the table, nested loops are executed. The outer loop is a pointer to the permit number currently being examined, and the inner loop is a pointer that steps through all of the permit numbers below the number in the outer loop to search for duplicates. If no duplicates are found or any duplicate number is marked as "Listed Per HQ", the instrument moves on to allow the FR to answer some final questions and finish the listing. If a duplicate is found, the script prompts the FR that a duplicate has been found and must be corrected before the listing can be wrapped up as a completed case. The following Blaise code shows the loops inside the Maniplus script and the IF statement that would signal a duplicate permit.

```
FOR I := 1 TO EndPointer DO
   FOR J := I+1 TO EndPointer DO
      IF (Listing.PermitList[I].PERMIT_NUM =
          Listing.PermitList[J].PERMIT_NUM) AND
         (I <> J) AND ((Listing.PermitList[I].PERMIT_REM < 2) OR
         (Listing.PermitList[J].PERMIT_REM < 2))  THEN
```

This comparison is simple and straightforward enough, but the challenge is in how to display and edit duplicate permits. Information about the duplicates must be displayed to the FRs in an easy to read format so that they can look at each permit and determine which permit number is correct and which one needs to be corrected. Another challenge in this approach is being able to view and correct the permit number quickly without requiring the FR to scroll through several screens to get to the listing line they need. The SOC LI is designed to show 15 permits per screen, if one permit is on line 112 and the other permit is on line 182, they have to press Page Down or Page Up to move ahead or back 4 screens to look at the permits in question. That is a very inefficient way to look at the permits and make the necessary correction.

A solution we developed to overcome these challenges is an approach to unduplication that was created in the National Crime Victimization Survey (NCVS) and detailed in a 2007 IBUC paper on the challenges of converting that survey to Blaise. In this solution, a separate table in the instrument is created that has two lines with the line number, permit number, sample flag, and remark field from the original listing table. The form pane displays the full permit information for both permits in question in a side-by-side format so that the FR can look at the two permits together on one screen to make the comparison (See Figure 2). The unduplication script fills the table with the permit information so that the FR just has make corrections to the displayed permit(s).

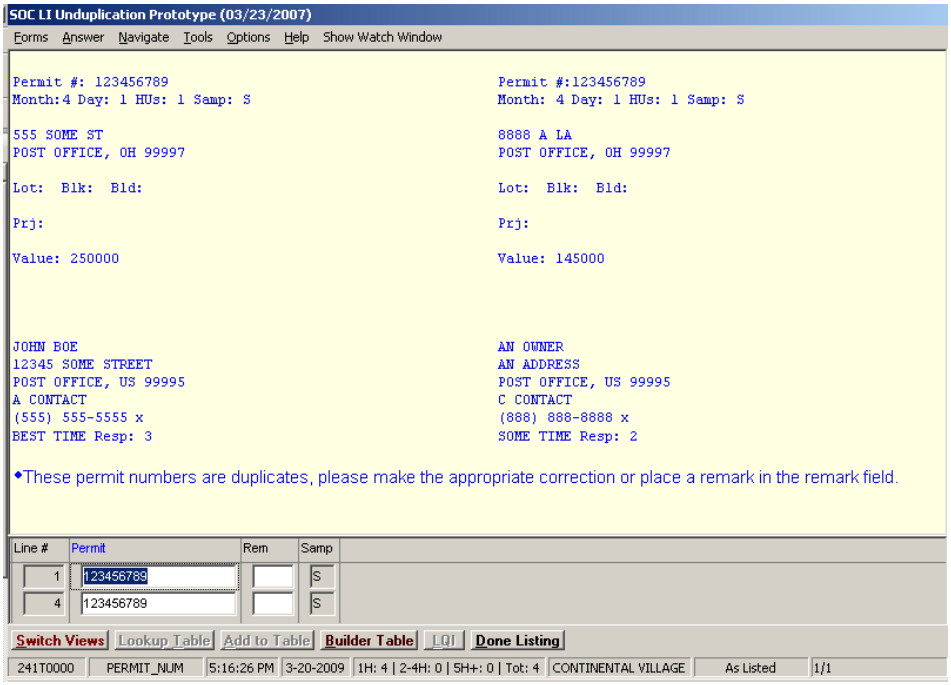

Figure 2: Unduplication screen to make corrections.

Once corrections are made, the unduplication script is called again. The script transfers the data from the holding table back into the listing table, and brings the FR back to the wrap-up question for the case. The FR presses Enter and unduplication runs again, repeating this entire process until no more duplicate permits are found in the table.

At the time this approach was not preferred because unduplication did not run until after the listing was done. If there are multiple duplicates an FR has to repeat this search multiple times until all of the duplicates are found, which is less user-friendly than reporting that a permit number is a duplicate right away. This unduplication approach was still a somewhat viable option because it did not slow the instrument down like the procedure.

4.2 Prototyping and Stress Testing

Unlike the procedure-based method, using Maniplus did not impact the load time of the instrument; however, the search for duplicates did take longer as more permits were listed. Since unduplication only runs once at the end of listing rather than multiple times because of rules execution in Blaise, it did not slow the instrument down during listing. We normally run our Maniplus scripts in quiet mode in the instrument, so during unduplication it

looks like the instrument is "frozen" while the script is executing. A message was added advising the FRs that it would take time to check for duplicates (See Figure 3).

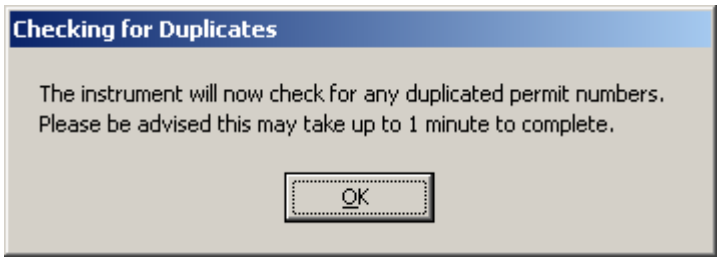

Figure 3: Advisory message to the FRs from the Maniplus script.

It takes less than 1 minute to run unduplication unless all of the listing lines are filled. The following times are based on creating a duplicate permit of the last listed line so that it takes the maximum amount of time to reach a duplicate:

Number of Permits	Unduplication Time
100	2 seconds
200	3 seconds
500	6 seconds
1000	14 seconds
2399	55 seconds

These are better times than what we had observed with using a procedure, satisfying the requirement that unduplication checks for duplicates in a reasonable amount of time. However, using the script meant the instrument did not catch a duplicate permit number as soon as it is entered. This is not what we want from unduplication, but there was a distinct possibility that we would have to use this approach unless another way could be found that was both instantaneous did not slow down the instrument.

Advantages of using a Maniplus script:

- No lag time with instrument load, in listing table, or navigating parallel blocks.
- Fairly easy implementation.

Disadvantages of using a Maniplus script:

- Does not provide the functionality requested – duplicates are not identified until after listing is "complete." This could lead lots of extra work cleaning up the listing.
- Convoluted way in which FRs had to deal with duplicate permit numbers.
- A one-time lag of up to 1 minute for large listings.

5. Unduplication Using Blaise API to Call a Visual Basic Application

The Maniplus approach showed promise, but had a major drawback. As a result, we decided to see if we could run unduplication outside of the instrument and run it immediately after a permit number was entered. Another application was already being developed in Visual Basic 6 using the Blaise API to work with the SOC LI. We decided to see if an unduplication function could run using Visual Basic and the Blaise API through an alien router without slowing the instrument down.

5.1 Design and Challenges

The unduplication function for Visual Basic 6 takes elements of both the unduplication procedure and the Maniplus script to create a hybrid of each of their approaches to search for duplicates. Since the function runs independently of the instrument via an alien router, it only needs to take the permit number from the current line and use a loop to compare it to all of the permit numbers listed before the current number. The function determines the line number of the current permit number to set the loop counter and stores the value of current permit number. The function enters the loop and checks the current permit number against each permit number entered above it. The Visual Basic code below details the loops and if statements that locate a duplicate permit.

```
If DS.AlienRouterStatus = blrsPostEdit Then
     If ActiveField.Required Then
      For i = 1 To Index - 1
         If WhichSurvey = "SOC" Or (WhichSurvey = "FullView" And
            Mid(db.Field("INTPER").Text, 7, 2) = "SL") Then
            If UCase(Trim(db.Field("Listing.PermitList[" & i &
               "].Undup.PERMIT_NUM").Text)) =
               UCase(Trim(ActiveField.Text)) And _
               db.Field("Listing.PermitList[" & i &
               "].PERMIT_REM").Value < 2 Then
                  retval = True ' match found should not be itself
                  Exit For
            End If
         End If  'WhichSurvey = "SOC"
      Next i
   End If   'ActiveField.Required
End If   'DS.AlienRouterStatus = blrsPostEdit
```

The alien router status has to be set to blrsPostEdit, otherwise unduplication runs before the FR enters a permit number. To make sure the permit number comparison is accurate, the two permit numbers are made uppercase during comparison since some permit offices use letters as a part of a permit number. If a duplicate permit number is found, a message is immediately displayed to the FR that a duplicate has been found (and where). Otherwise the cursor just moved on to the next field in the list (See Figure 4).

Figure 4: Unduplication error message from Visual Basic for the SOC LI.

In order to call unduplication from within the instrument, an embedded block is in the listing line. This block contains only the permit number and an alien router to call unduplication.

```
EMBEDDED BLOCK blkUnduplication
   FIELDS
      PERMIT_NUM (PERMIT_NUM)
      ENG "@L?@L·@W[F1]@W
           @/^PermitFill
           @/@LEnter the permit number issued:@L"
      SAS "PERMNUM"
      HLP "H_PNumber"
                 / "Permit" : TString24, NODK, NORF

   ROUTER Unduplicate ALIEN('BuilderTableDLL.BT','DupEntry')
ENDBLOCK
```

In the field section of the listing line:

```
Undup : blkUnduplication
```

In the rules of the listing line:

```
Undup.Unduplicate
```

Unduplication inside Visual Basic works perfectly in that the FR instantly gets feedback that a permit number is a duplicate. However, an immediate challenge surfaced when this function was implemented. Since unduplication runs as the cursor is leaving the permit number field and runs successfully despite the "duplicate permits" message, Blaise treats that successful run as an indicator that everything is fine so the cursor moves to the next field. The "duplicate permits" message does not come up again unless the FR backs up to the permit number field on that line. If the FR does not back up, then the duplicate permit number is still in the listing. We tried to keep the cursor in place by changing the status of the alien router or even clearing the keyboard buffer in Visual Basic, but it seemed like nothing would work. The solution to this challenge came by doing something most experts say you are not supposed to do when you make assignments to variables in a program:

```
If DS.KeyBuffer = "" = "" Then
End If
```

Setting the keyboard buffer to empty twice on the same line did not create a syntax error when the function was compiled, but during run-time unduplication would fail and exit immediately without completely finishing the program. Since unduplication exits and returns the result that the program failed, the cursor stays on the permit number field rather than move forward to the next field. A bonus of using this solution is that even if the FR tries to press a function key or click the mouse to leave the permit number field, they cannot leave the field if the permit number is a duplicate, they have to fix it first.

Another challenge that surfaced with this implementation of unduplication was some odd cursor behavior in the instrument when the FR clicked a parallel block tab. If the cursor is on the permit number field, the parallel block tab would briefly have focus, but then the focus would immediately return to the main path and the cursor would go back to the very first question of the SOC LI. If the FR clicks the tab again, the focus goes to the selected parallel block. This does not happen if a parallel block is accessed from a function key, and it only happens if the cursor is on the permit number field. We suspect that it has to do with how the alien router handles focus and have attempted to find a workaround. However, the only workaround at this point would be to remove the tabs from the instrument. This solution would affect functionality in the instrument for the FRs and could be considered worse than the problem it fixes. As a result, it was decided to leave this alone and live with the occasional jump to the beginning of the survey.

5.2 Prototyping and Stress Testing

Since the Visual Basic function is very quick at running the loops and the Blaise API is fast to switch back and forth between unduplication and the DEP, lag is not as noticeable when unduplication is running. When a listing is 1000 permits or more, a lag can be observed. However, this lag occurs only when the user re-enters a previous listing case and enters a new permit number, which runs unduplication for the very first time in that session. The error message or cursor movement is nearly instantaneous after the initial run. All times below are for an initial run of unduplication.

Number of Permits	Unduplication Time
100	<1 second
200	1 second
500	1 second
1000	4 seconds
2399	7 seconds

Advantages of using Blaise API to call a VB application:

- Very small lag time with first run of unduplication (after reloading instrument).
- No lag when navigating parallel blocks.
- Checks for duplicates as permits entered – functionality requested.
- "Instantaneous" check in listing table (no lag).

Disadvantages of using Blaise API to call a VB application:

- More challenging implementation – must install DLL on laptops.
- Focus issue when using the mouse to change parallel blocks from the permit number field.

6. Implementing Unduplication

Based on internal testing of our prototypes, we decided to use Visual Basic and the Blaise API for unduplication in the SOC LI. It does the duplicate check and gives feedback to the FR almost instantaneously so they can correct any problems immediately. It is similar to unduplication in the Clipper version of the SOC LI so the FR does not have to adjust to a new way of being alerted to a duplicate permit. Another program was already being developed for the SOC LI in Visual Basic, the SONC (Surveys of New Construction) Builder Table. Rather than make the instrument keep track of two different files, the unduplication function is coded inside the Builder Table since every laptop that runs a SONC survey has the Builder Table installed as well.

6.1 PAL and NCE

As part of the initiative to convert the SOC LI from Clipper to Blaise, other listing surveys were being converted to Blaise as well. These instruments included the Permit Address Listing (PAL) and Nonresidential Coverage Evaluation (NCE) listing instruments. One aspect of their design is to have them look and feel similar to the SOC LI so it will be easier for an FR to learn how to navigate the instruments. As part of that requirement, unduplication was implemented in PAL and NCE. It makes sense to use the same method for unduplication across all of the surveys since it means minimal changes to the survey design.

Much like the SOC-LI, PAL collects information about newly constructed residential homes in the United States. The major difference between the two surveys is that PAL only collects address information about the new residence and does not sample addresses for follow-up and further data collection. PAL has some extra requirements for unduplication because the sponsoring division wants to allow duplicate permit numbers in the instrument. However, the sponsor also wants to have those permits flagged for review by the FR during listing, or the sponsors during post-processing. Some BPOs for PAL are the same BPOs for the SOC LI, so those cases come to the FR pre-filled with permit numbers from a SOC LI case that had been collected previously. Sometimes permit details change between the time the permit is collected in the SOC LI and when it is being listed in PAL. PAL is designed to keep the old permit from the SOC LI and just add the new one as a line below the old permit. The instrument uses a Maniplus script to insert a line in the listing table and put a flag on that line to indicate that the permit number is a duplicate, and that some other permit information is different from the original. These permits are ignored during unduplication. For brand new PAL cases that do not have permit information from a SOC LI case, the FR is prompted with a message asking if a permit is a duplicate. If they answer yes, the instrument flags that permit line and the instrument moves on. If they answer no, the permit number has to be corrected.

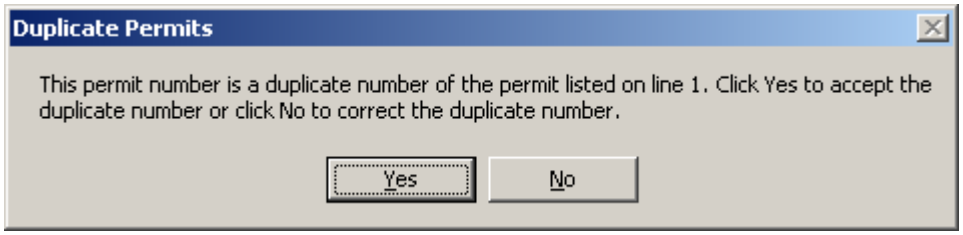

Figure 5: Unduplication error message from Visual Basic for PAL.

NCE is a listing survey that measures new commercial building construction in the United States. The instrument is very similar to the SOC LI so it carries a similar unduplication message. A difference between NCE and SOC LI is that a permit can be flagged for deletion in the NCE instrument. By allowing this flag in NCE, any permit that is flagged for deletion is ignored by unduplication.

7. Conclusion

The SOC LI is a survey that has run continuously in the field for many years and it was our goal during the Blaise conversion that it have a similar look and feel to the Clipper based instrument where possible. Unduplication is extremely important for a survey like the SOC LI because good data quality is absolutely critical in producing data on housing construction in the United States. Unduplication presented a unique challenge in that for a survey as large as the SOC LI it had to run quickly and with minimal impact on the instrument. Thanks to the flexibility of Blaise, there are a number of approaches that can be taken to accomplish unduplication, each with their own unique methods, challenges, and uses inside the instrument. We learned the following lessons about how to use each approach:

- A procedure would be beneficial for a smaller survey instrument that does not need the Blaise API.
- Maniplus scripting would work well if unduplication did not need to be performed immediately after a permit was keyed.
- Using the Blaise API is the best approach for larger instruments that require heavy lifting.

Since deploying the Blaise version of the SOC LI into the field, there have been no required adjustments made to unduplication. In the future we may consider modifying the duplicate permit number search to include any permits entered after the one on the current line. This would cover situations where the FR decides to back up and change a permit number in the middle of their listing.

8. References

Carter, C., Picha, R., Mangiapane, M., Arrington, A., Moshinsky, D.; U.S. Census Bureau, USA (2007): Challenges in Converting the National Crime Victimization Survey to Blaise, Presented at the 11th International Blaise Users Conference 2007, Annapolis, MD, 2007.

9. Acknowledgements

The author would like to acknowledge the following people for input into this paper and input on some of the technical requirements for unduplication.

Erica Filipek, U.S. Census Bureau

Roberto Picha, U.S. Census Bureau

Karen Bagwell, U.S. Census Bureau

Tom Spaulding, U.S. Census Bureau

Malcolm Robert Wallace, U.S. Census Bureau