Blaise Source Code Editing System

Sheila Deskins and Danilo Gutierrez, University of Michigan

1. Introduction

This paper discusses a Blaise Source Code Editing System designed and developed by Health and Retirement Study (HRS) programmers. It covers the six major functional design components of a source editing system, as well as specifics relating to application design, development, and testing plans that allowed the creation of the current working system.

The Blaise Source Code Editing System or "Source Editor" is used to make several hundreds of updates to the Blaise Source files (.bla/.inc) automatically. In the past the process of updating information from electronic review systems had been incorporated into Blaise code by hand, which was a labor-intensive, tedious and error prone process.

The six major functional design components are: a file reader/parser to process the Blaise source code files, a maker of expanded Blaise statements, a translator/pre-processor to change electronic update information into a format the merger can process, a merger that combines the update information and the Blaise source code, a file writer to produce the revised files, and an interface to encompass the whole system.

This paper will go through the evolution, design and current state of the Source Editor System. Topics covered are: the motivation behind having such a system, key design concepts, the implementation of the modular design, the testing plan for systems development, and current and future uses of the system. An appendix, containing a sample program with details of tables used as I/O in key design components of the system, is included.

2. Background

The Health and Retirement Study (HRS) is a longitudinal study that began in 1992. For the 2002 data collection period, HRS's CAI[1] application switched to Blaise. Although the basic questionnaire content of a longitudinal study does not change over time, there are many changes in business practices. Changes such as implementing revised screen formatting guidelines and revisions which facilitate post data collection distribution are typical.

Changing needs such as these often translate into frequent large scale changes to HRS's CAI application. The number of lines of code requiring modification is usually counted in the hundreds or thousands. These voluminous changes are programmed into the CAI prior to each field period. The work is labor intensive and time consuming. Thus HRS wants to automate this work as much as possible.

Development of the source editor system began in 2003 when a programmer was given an electronic file with 2,700 descriptors to update. The time frame allotted for the task was based on doing the updates by hand. A source editor system was created which

[1] Computer Assisted Interview

consisted of a parsing application writing to a database and a series of ad hoc merging and writing routines. Descriptors were done using this early system. Later, a screen format change resulted in a few hundred fields needing updating. The updating was also done via this early source editor.

For the 2006 CAI application a few thousand descriptors were updated. The descriptor input file had a different format which required new ad hoc routines to be created. One routine was created to translate Blaise Data Entry Program (DEP) names to defined field and block names. Another routine identified duplicate update requests based on the mapping of the DEP name to a defined field and block name.

It was decided for the 2008 CAI to add a new language. This very large task prompted the programming group to develop the current system. The system needed to handle more than just field text and descriptor updates. With the complexities and the larger scale of the new language task, the system needed to become more robust and required additional functionality. This was the first large scale task identified for 2008 and more changes are planned.

3. Design Conceptualization

What is a source editor? A source editor is a set of related processes and applications that allow for the editing of Blaise source code files. Blaise source code files are files with .bla and .inc extensions that contain Blaise source code to compile and generate a .bdb, .bmi, and other files.

The source editor system preserves the original .bla and .inc file structure. The basic core components of the source editor are the parser, merger and writer. To edit the source code, we have to have the original source files available, a collection of planned changes (or updates) and a way to save and write the updated changes. The source editor system is designed to be a bulk editor. That is, to make several hundred changes at once.

When talking about editing Blaise source code files, what do we mean, exactly? Do we get to add new fields, new question text, new enumeration code names, new enumeration code labels, or new user defined data types? Do we get to add blank lines, whitespace, and comments? Can we modify what is there? How much can we delete? What assumptions are being made? Answering these questions helps to provide insight into the source editor system design and conceptualization.

Beginning with the first question, “Do we get to add new fields?” the answer is no. The goal is to edit or update existing Blaise source code, not to add new[2]. The Blaise source code consists of existing Blaise program statements. Our bulk updates deal with changes to text and languages. This begs the question “Is adding a new language something the source editor can do?” Adding a new language is not really adding something new. If you examine a Blaise statement for the field syntax, the statement allows for all languages defined by the languages statement.

[2] Adding new statements implies development of an authoring system, a much different concept than an editing system. HRS has a working authoring system which is not covered in this paper.

Syntax
FIELDS
Q [Q1, [...]] [(Tag)] [[Lid] "Text"] [...]
[/ [Lid] "Description"] [...] : T

All parts of the statement syntax may not be explicitly coded. However the statement does exist. It is desirable for the source editor to update all parts of the statement, especially since many of the bulk changes deal with languages. For example, if two languages are defined and the statement is coded as:

Fields

Q1 (Q1) "Are you ready to answer questions?":(y,n)

There is implicit room in the above statement for a second language text, two descriptors, and language identifiers for all the text. Table 1 below shows the statement as parsed, with explicit token types and corresponding tokens. Table 2 below shows the expanded statement with the implicit token types added.

Table 1 Statement as Parsed

LnNoBeg	ColBeg	LnNoEnd	ColEnd	Token[3]	TokenType
170	2	170	3	Q1	FName
170	4	170	4		WhiteSpaceBlank
170	5	170	8	(Q1)	FTag
170	9	170	9		WhiteSpaceBlank
170	10	170	45	"Are you ready to answer questions?"	Text
170	46	170	46	:	FDefnSep Separator :
170	47	170	47	(	EnumBeg Separator (
170	48	170	48	y	EnumCodeName
170	49	170	49	,	EnumSepSeparator ,
170	50	170	50	n	EnumCodeName
170	51	170	51	)	EnumEndSep Separator)

Table 2 Statement as Expanded

Line NumBeg	Column Num Beg	Line Num End	Column Num End	Token	TokenType	LangID	Code Value
170	2	170	3	Q1	FieldName		
170	4	170	4		WhiteSpaceBlank		
170	5	170	8	(Q1)	Tag		
170	9	170	9		WhiteSpaceBlank		
	9				FieldTextLangID	ENG	
170	10	170	45	"Are you ready to answer	FieldText	ENG	

[3]A token is part of a program statement consisting of characters identified as meaningful syntax.

Line NumBeg	Column Num Beg	Line Num End	Column Num End	Token	TokenType	LangID	Code Value
				questions?"			
	10				FieldTextLangID	SPN	
	10				FieldText	SPN	
	10				Separator/		
	10				DescTextLangID	ENG	
	10				FieldDescriptor	ENG	
	10				DescTextLangID	SPN	
	10				FieldDescriptor	SPN	
170	46	170	46	:	Separator:		
170	47	170	47	(	Separator(_Enum		
170	48	170	48	y	CodeName		y
	48				CodeTextLangID	ENG	y
	48				CodeText	ENG	y
	48				CodeTextLangID	SPN	y
	48				CodeText	SPN	y
170	49	170	49	,	Separator,		y
170	50	170	50	n	CodeName		n
	50				CodeTextLangID	ENG	n
	50				CodeText	ENG	n
	50				CodeTextLangID	SPN	n
	50				CodeText	SPN	n
170	51	170	51	)	Separator)_Enum		

In order to merge into a statement like this, the explicitly coded parts and the implicit parts need to be explicitly created in preparation for the merger. This concept of a Blaise statement being expanded so all the parts of the statement exist is being referred to as an expanded Blaise Data Object, or simply, Blaise Data Object (BDO). The BDO is a key concept in the source editor design. With the BDO we can modify the explicitly coded parts along with the implicit parts of a Blaise statement.

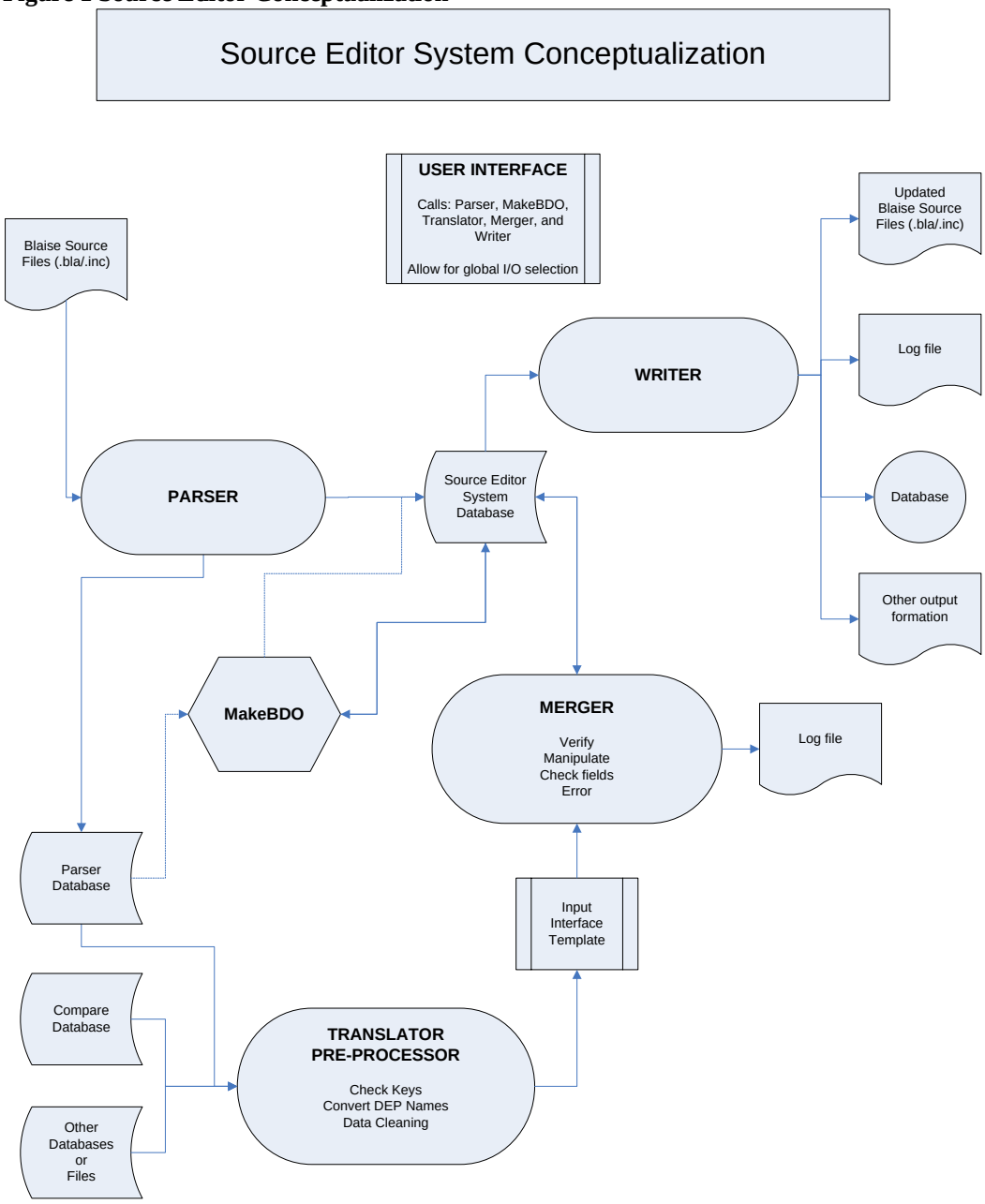
Whitespace and comments in Blaise are statements that stand alone. They're not associated with a field, a block, or a user defined data type. Comments and whitespace are not statements we can locate by using a field name or user defined data type name. They cannot be updated by the merger.

As for deletions, we can delete anything that exists for a Blaise statement, as long as the statement and parts of the statement can be uniquely identified.

As for the assumptions, a few are being made. One assumption is that the user data is in a structured electronic format. Another assumption is that a key, like field name, is provided with the update request. A third assumption is that more than one type of merge key is needed, and that it depends on the token type. The last assumption is that the merger will use an input format that will handle all requests. To see an example of the token types, merge keys, and the user input format, please refer to the appendix.

One question that came up often during the source editor design was "Why don't we use the .bmi?" The appeal of the .bmi is that Blaise has already done the parsing and tokenization of the statements from the source code files. It is in a form programmers working with the Blaise Application Program Interface (API) are familiar with. It is probably possible to create Blaise source code from the .bmi. However, the resulting source code files would lose the include file structure, comments and whitespace. For HRS' purposes it is important to retain the original include file structure, comments and whitespace. Due to these needs we cannot use the .bmi.

Figure 1 Source Editor Conceptualization



4. Design Implementation

Our general design implementation objective was to think of the core system components as modules, namely, a modular design approach. We did not want to create unnecessary or redundant functions across components. We wanted to be able to have reusable code, i.e., not so customized that we would have to open the source editor system code to change I/O specifications. More importantly, our design had to be able to handle the HRS CAI.

Our three main implementation objectives were: to create the BDO (Blaise Data Objects), merge the updates, and write out the new files. In order to create the BDO, we had to parse the Blaise source code files to know what is defined in the datamodel. We needed to add all of the potential space for the expanded statements. The merger would then take the user input data and merge it into the expanded statements in the BDO. Last, we would write out the new files with all of the updates.

In the design implementation, we programmed the basic components of the system separately. They are: the parser, MakeBDO[4], the pre-processor (translator), the merger and the writer. The parser and MakeBDO functions prepare the table for all possible expanded syntax. The pre-processor writes all user updates into one table (user input format) that is used by the merger to update the expanded syntax in the BDO. The writer is the last component in the system.

Taking into account the complexity of the tasks, the general design objective of a modular design, and the division of work tasks, we decided that trying to cram the statement expansion and language reordering into the parsing routines would be awkward, complex, confusing, and in any case not directly related to the parsing process. The language reordering and the statement expansion are not really part of the merging process either. So we decided we had to have an intermediate process of making a BDO. There were clear advantages of using separate components for the different functions in the source editor, including making debugging and application maintenance much easier.

Given the scope of work that needed to be done, the time frame, and the resources available, we could not implement the whole design. We decided to do a phased-in implementation, where the programming effort would focus on core elements of the system. It was important for us to consider the addition of a new language in our application design and implementation. We called this Phase 1.

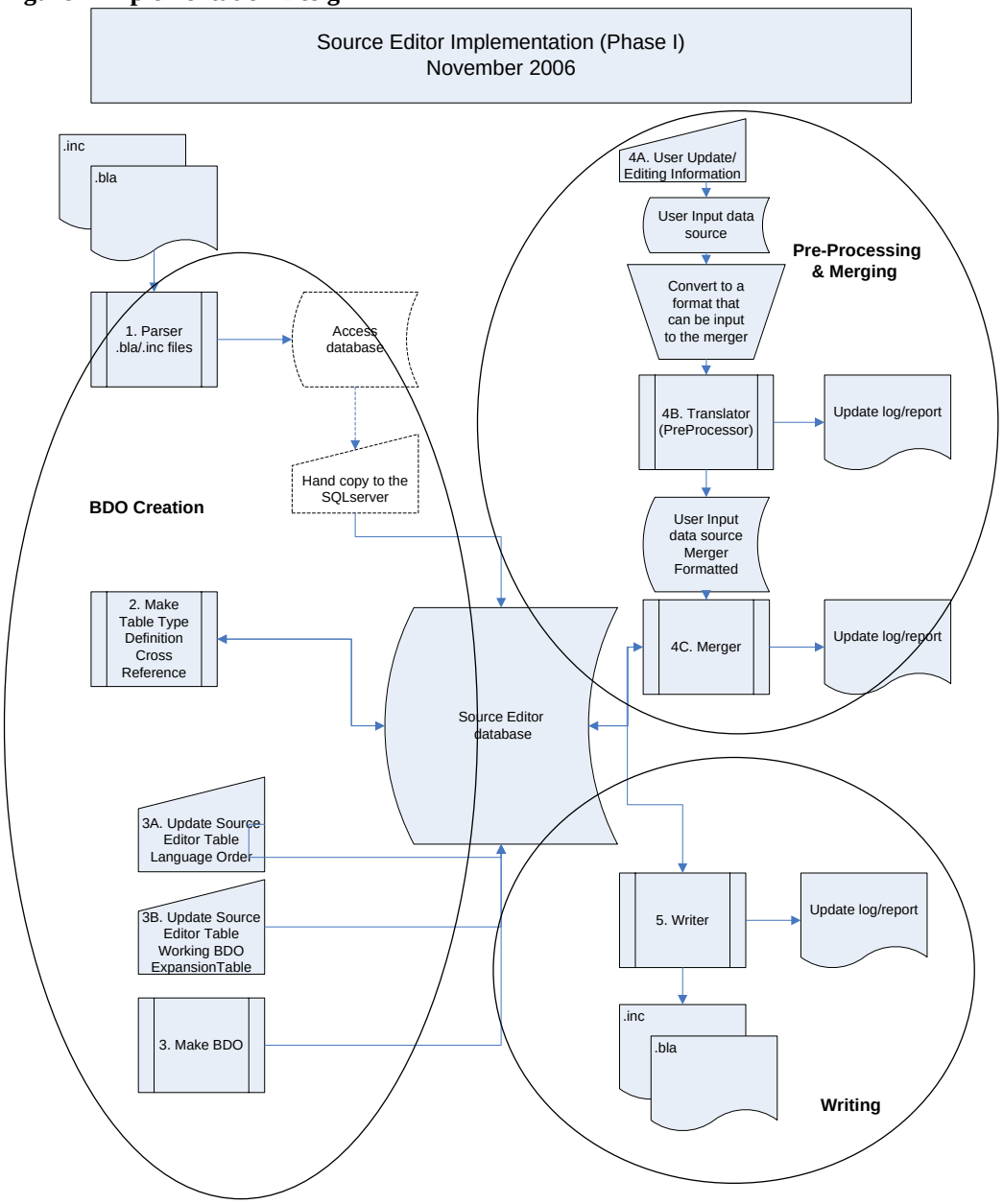
An implementation assumption we made is that the initial Blaise source files should compile. Thus, the parsing routine does not check for Blaise syntax errors. If the Blaise files don't compile after being run through the source editor, the user input data needs to be checked. Routines to check user input data will be taken into consideration when working on Phase 2 of the source editor system. Phase 2 developments will focus more on the translator, pre-processing routines, and the anticipated variability of user input files.

[4] MakeBDO is the application module that creates the BDO.

We currently have a working implementation of the source editor system's core functions. The next section covers the Phase 1 implementation modules of the source editor system.

5. Implementation Modules

Figure 2 Implementation Design



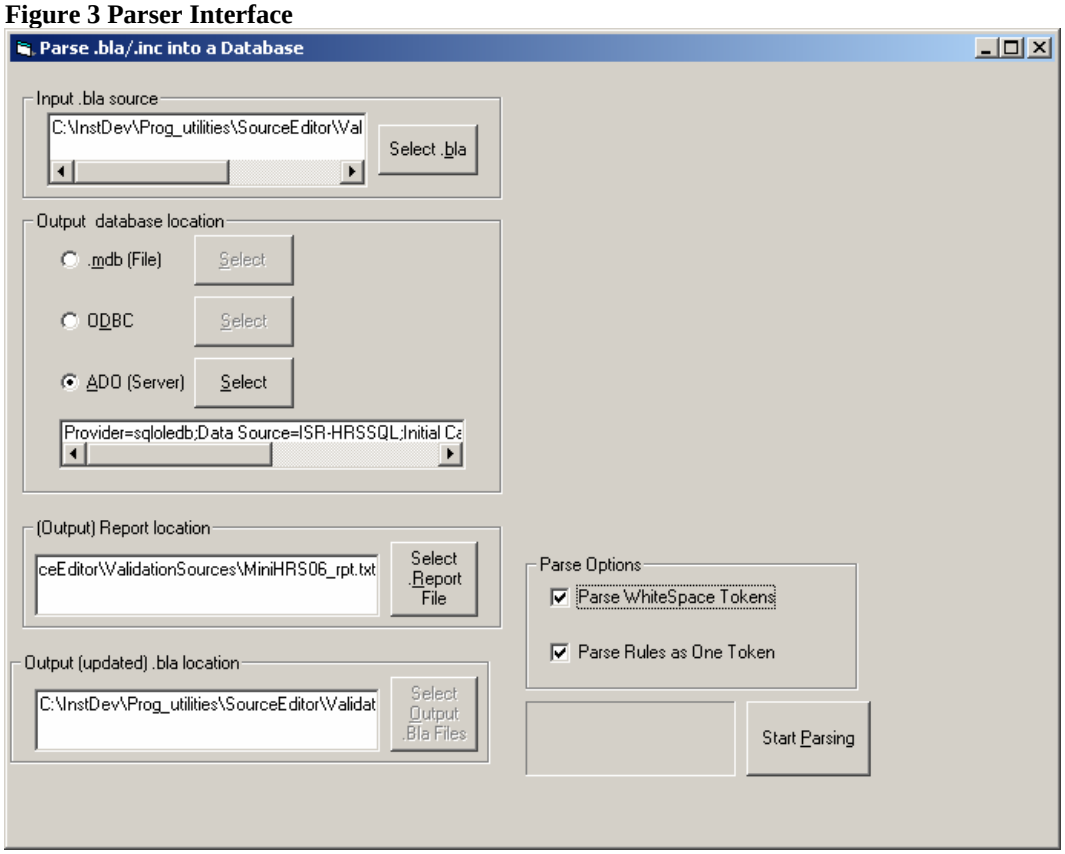
5.1 Parser

The first process in the source editor is parsing the .bla and .inc files into tokens. Tokens are defined parts of Blaise syntax statements. The parser can be used as a stand alone application to provide information about the block structures and logic structures. It has the ability to write to an MS Access database as well as to a SQLServer database. The

parser application is used to create the tables and information that is the base for the BDO table creation.

For the parser to generate output for the source editor system, the 'Parse WhiteSpace Tokens' option must be selected. 'Parse WhiteSpace Tokens' allows for the capture of tabs, spaces, carriage returns, etc. that are essential in order to preserve the whitespace in output files. This feature is optional and can remain unchecked if the parser database is to be used for analysis purposes only. 'Parse Rules as One Token' is also optional. Selecting it helps to reduce processing time and reduces the number of records generated when there are no anticipated updates to the code in the rules.

Figure 3 Parser Interface



The parser application is run -- after selecting an input .bla source, the output database location, and parse options -- using the 'Start Parsing' button. The parser indicates when it is done.

To see examples of tables populated by the parser, refer to the appendix.

5.2 Table Type Definition Cross Reference

In preparation for making the BDO ready for the merger, an application creates references to type definitions that are not included in the same location as the field. Some types are used in many places, but defined in only one place. The interface for the application is shown in Figure 4.

Figure 4 Cross Reference Interface

In the example below for field TP50 the enumeration type is defined at the field. For field TP60 the enumeration type is defined elsewhere. The cross reference table provides this information to the BDO. See Table 3 for an example of data in the cross reference table.

Fields

TP50 "What type of fish do you have?": set[3] of (n "none", f "fresh water", s "salt water")

TP60 "What type of mammals aside from dogs or cats do you have?": TMammals

Table 3 Type Definition Cross Reference

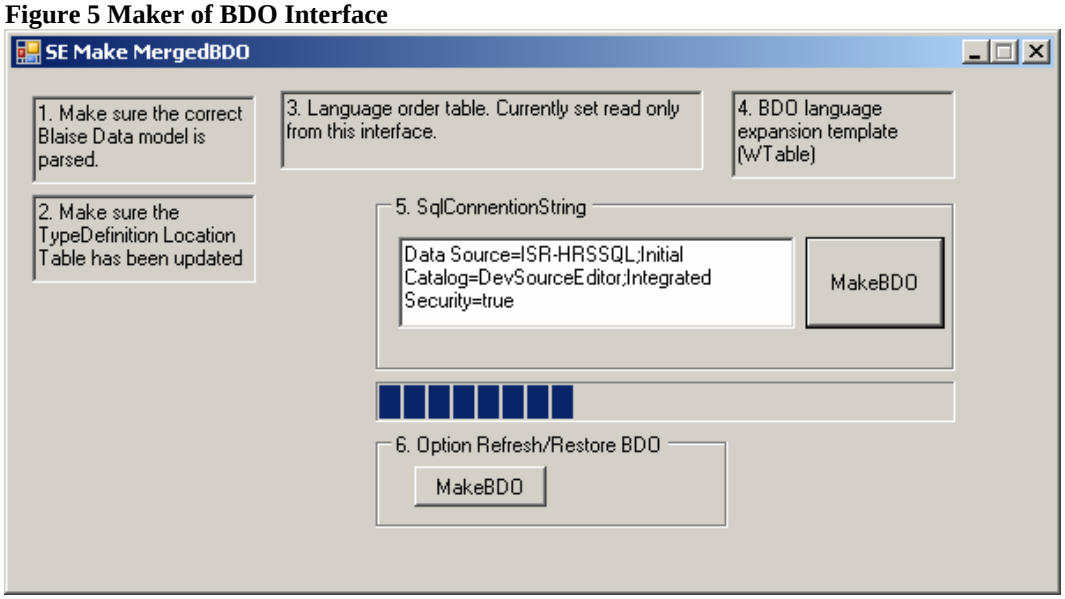
Blkname	Field Name	Type Name	Type No	Blk path	Blk No End	Blk NameEnd	Type NameEnd	TokenType
SE_illustration		tmammals		1..	1	SE_illustration	tmammals	TName
B2_Other	TP50		40					
B2_Other	TP60		41	8.1..	1	SE_illustration	tmammals	FTName

5.3 Creating the BDO (MakeBDO)

The MakeBDO application performs several functions. It creates anchors or spaces for Blaise statement tokens that are not explicit in the parsed code. It allows the user to indicate which implicit tokens to create. It has the capacity to reorder the languages. The MakeBDO application creates a table, which is used by the merger to write updates and by the writer as a data source.

Prior to running the application, in Figure 5, step 1 refers to tables made by the parser; step 2 refers to the cross reference table made by the stand alone Type Definition Cross Reference application; in step 3 the language order table is checked and adjusted if necessary for reordering languages. We knew we would get into volume, scale, and processing time issues, so in step 4, we provided the ability to indicate the number of languages[5] we want to allow space for in the expanded statement tokens for the BDO. All of these tables are used in the creation of the BDO.

Figure 5 Maker of BDO Interface



"Option refresh/restore BDO" is used to re-establish the BDO table before a merge. This shorter process is used for instances where a bad merge[6] was made. This is especially useful on a large datamodel because it saves time.

5.4 Translator / Pre-processor

The goal of the translator or pre-processor is to move data into a table that the merger uses to make data updates into the BDO table. In Phase 1, HRS used a customized pre-processor without a translator. A translator is needed to convert long DEP fieldnames into defined blocks and fieldnames. A translator is a major component planned for future development.

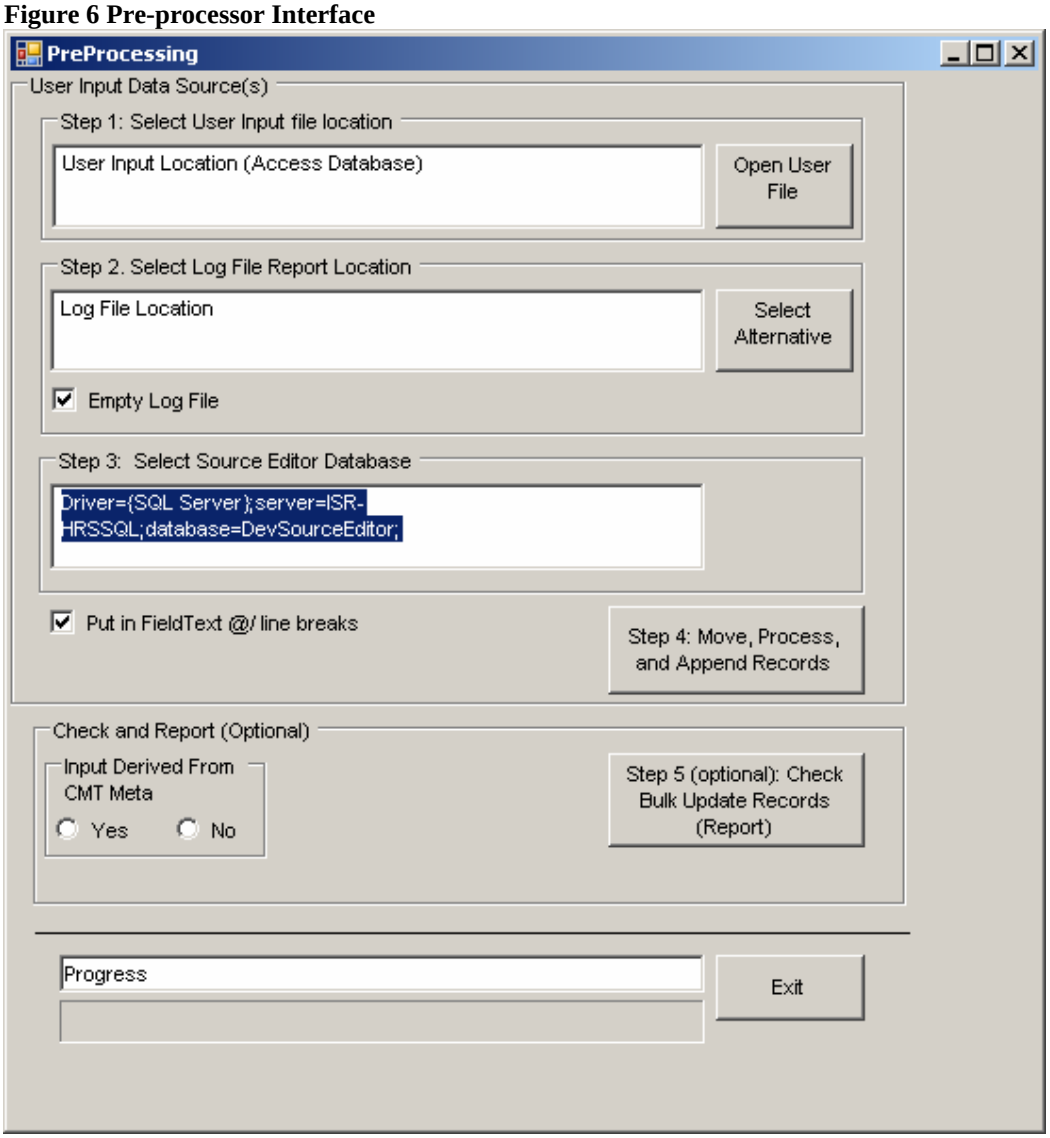
---

[5] HRS has seven languages.

[6] A bad merge would occur when the user update information is not specified correctly via bad data or bad format.

How did we get away without writing a translator in Phase 1? Our updates mainly consisted of adding a new language. For this task the user input data was a very clean data source with the Blaise block path and field names, so we didn't have to translate DEP fieldnames.

Figure 6 Pre-processor Interface



The option 'Put in FieldText @/ line breaks' is used to modify user input for question text. It converts '@/' to <linebreak>+ '@/' to have the line break in the typical manner for HRS.

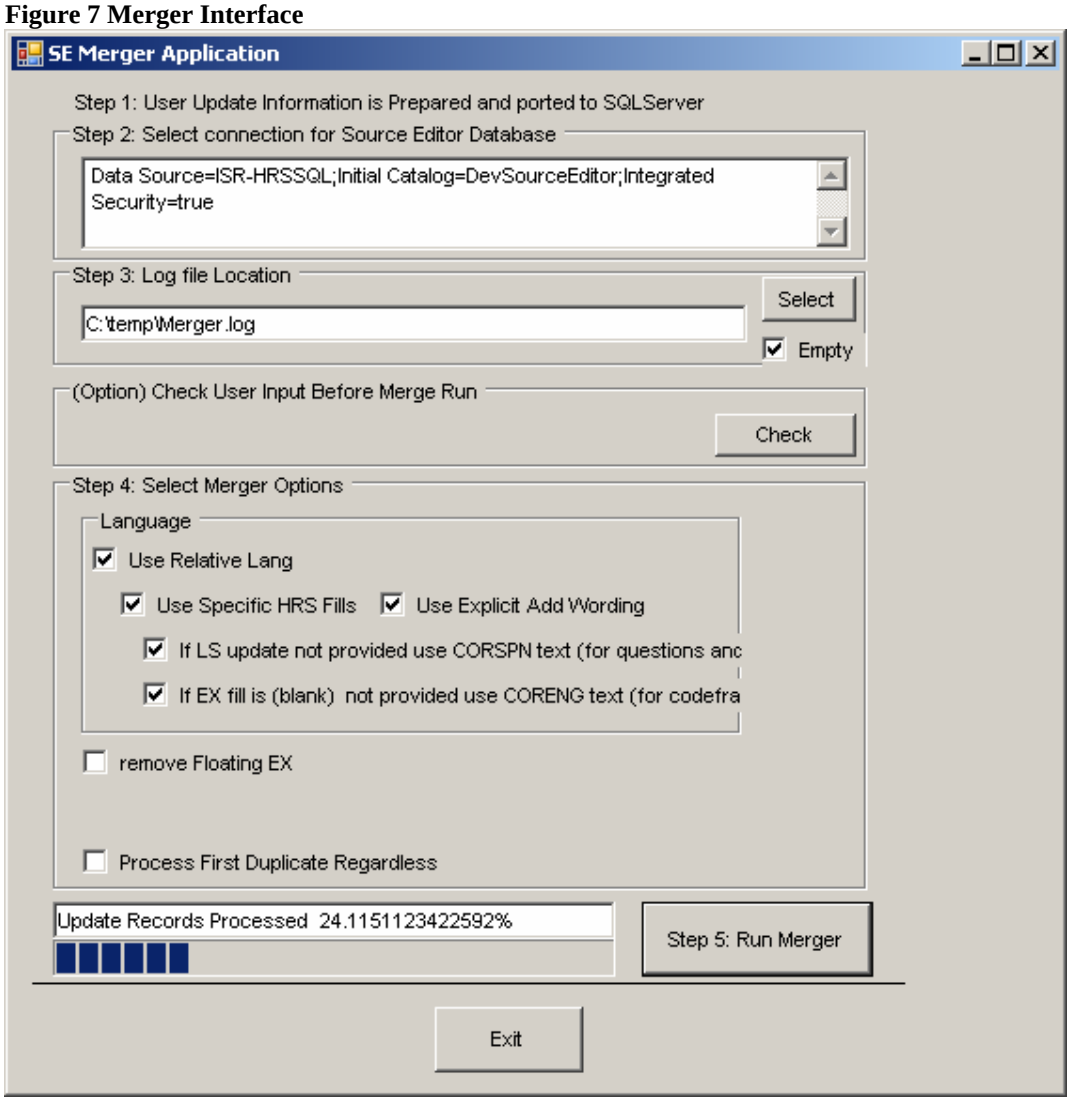
Some of the pre-processes to prepare user input data for merging are:

- placing quotes around text
- explicitly stating Language ID (LID)
- checking for duplicate requests
- checking for items referenced multiple times but only appearing once in code
- checking for duplicate block names
- checking for token type

5.5 The Merger

The merger application processes changes or requests and merges them into the BDO table.

Figure 7 Merger Interface



‘Use Relative Language’ and ‘Use Explicit Add Wording’ are generic options that can be used with any instrument. ‘Use Relative Language’ is selected when LID is not used in code. ‘Use Explicit Add Wording’ writes automated quoted text wherever it is needed in the expanded statements. All other options are HRS-specific.

The option “Check User Input Before Merge Run” is optional. This option writes a report with checks on language order; expansion statements, duplicate blocks and duplicate include files. To see an example of the report produced by this option, refer to the appendix.

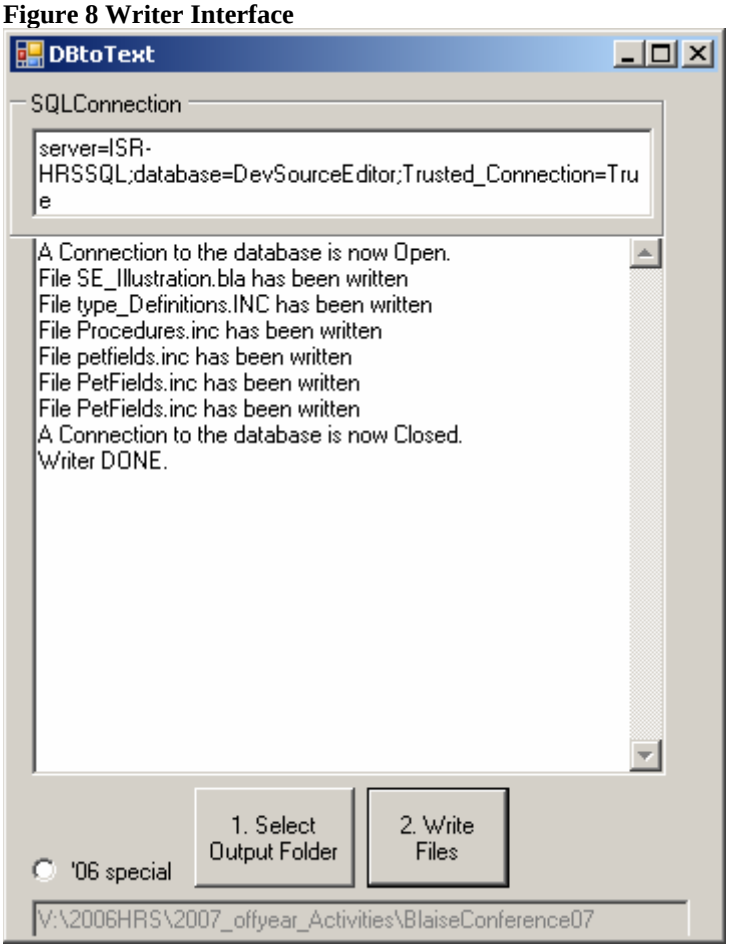
The option ‘Process First Duplicate Regardless’ allows the processing of duplicate user update requests.

The merger writes a log file with information about run time, number of token type records, and user input updates that were not processed. To see an example of the log produced by this process, refer to the appendix.

5.6 The Writer

The writer takes records from the BDO table and creates updated .bla and .inc files. If no updates and no language re-ordering are applied, the .bla and .inc files will write out unchanged[7].

Figure 8 Writer Interface



The writer produces a log indicating the name of each file produced and the corresponding number of updates, if any. This is a sample log:

> File SE_Illustration.bla has been written with 8 updates.
>
> File type_Definitions.INC has been written with 57 updates.
>
> File Procedures.inc has been written with 0 updates.
>
> File petfields.inc has been written with 18 updates.
>
> File PetFields.inc has been written with 18 updates.
>
> File PetFields.inc has been written with 18 updates.[8]

[7] As expected, initial source files run through the source editor without user updates will output with no changes.

6. System Interface

Future plans include creating a system interface that encompasses all system functions through one interface organized by tabs that will reference each component's interface, and include instructions on how to perform critical checks.

7. Testing

Good application development has a testing plan to ensure a quality product. We planned for three types of tests: the first test was to test each component or program module separately; each programmer did rudimentary component testing prior to using the validation datamodel. The second test was to integrate the components to test for interoperability and general system functioning. The final test dealt with size, scalability and throughput issues.

As input to the testing process, we created validation files containing Blaise source code designed to model all the functionality that our programs were supposed to have. Validation files are preferable to user inputs that either do not exercise all the functions of the application or produce multiple errors for functions that are used. Our last step was to run large volume test. Since these tests take time and resources, they should not be done until after the basic functionality testing for each piece is complete.

The validation source files for the source editor system are too large to include in this paper. We have small representative Blaise Program Source Code files covering key concepts in the appendix of this paper.

8. Is It Worth It? Current Use and Future Plans

Is it worth it? The main reason we created a successful application is because it meets the goal of doing large scale updates faster and more accurately than doing them by hand. Our rough estimates of the time differences between doing tasks by hand and doing them using an automated method such as the source editor indicate that it's worth it.

The example of changing descriptors by hand involves the steps of typing in the field name, searching the code for the field, copying the new descriptor from the user update file, deleting the existing descriptor, and pasting the descriptor into the source code. Doing these highly repetitive steps 2,700 times increases the likelihood of repetitive use injuries (a non-trivial issue), has more potential for errors due to typing and searching techniques used, and would take about 165 hours.

The system is even more useful in dealing with large scale changes. As noted above, during the most recent development effort a new language was added to the instrument. The addition of a new language affects all fields and all enumerated types. This is a much larger task than changing descriptors. The HRS CAI application is a very large application. It has 175,600 lines of code covering 56 files and 5,600 uniquely defined fields (excluding auxfields, locals, and parameter fields). To parse the whole application,

---

[8] Since the file was included three times, it is written three times by the writer. This can be problematic if different changes are requested for blocks and fields that map into the same include file.

make 6,500 updates, and write the updated files took about 24 hours using the source editor.

Table 4 Source Editor Scale and Time Estimate

Scale of Task	Approx time for task
Parsing 175,600 lines of code (56 files)	8 hours
Type Cross Reference	2 hours
Create BDO	4 to 6 hours
Merge (approx 6500 updates)	6 hours
Write files	3 minutes
Total Time	24 hours

The time difference is a few days versus a few weeks. Since most of the work is being done by machine, updates are more accurate, the likelihood of repetitive use injuries is reduced, and time is saved, by not having to replicate information that is already in a structured electronic format. If large scale tasks are done by hand a few times every other year (as in HRS), the time invested in developing a source editing system is justified.

9. Conclusion

The source editor system design conceptualization, development, and implementation encompassed design features such as a parser, a maker of expanded Blaise statements, a merger, a user data translator/pre-processor and a writer. The elements work together to make a system that edits Blaise source code files, processing updates on a very large scale.

Although we plan to add functionality to the system in the next development phase, the source editor can handle all of the following tasks without further development of the core functions:

- Strip out obsolete or dated comments from prior years.
- Update tags. Modify tags to be more descriptive.
- Update descriptors. Modify labels for data out.
- Update data types. Modify field size, field ranges, etc.
- Update language text. Text provided by another system such as a product from the HRS translation group.

As currently implemented, the Source Editor System benefits are:

- Time saving, resulting in faster turn-around of tasks.
- Hundreds of changes can be made at one time.
- More accurate placement of updates and therefore better quality.
- May reduce repetitive-use injury.
- Robust enough to handle applications as large as HRS.
- Generic enough to handle other non-HRS Blaise CAI applications.

- The application can add or re-order languages.
- The application has features to help handle scale issues.

Along with all these benefits, the Source Editor updates the .bla and .inc files automatically, while preserving comments, whitespace, and the .bla and .inc file structure.

10. Appendix

This appendix is not for the casual reader. The main document gives the overview of the design concepts. The material in the appendix gives a cryptic view of implementation design details that most readers would find, uninteresting, too detailed, or undocumented to make sense of. This information would make sense to someone that can look at tables and ascertain primary and secondary key relations determined by various combinations of columns.

10.1 Sample Blaise Program Source Code Files

Four files make up the sample illustrative program. These files will be used to show how tables are populated. The tables will be included. Records in the tables may be excluded to save space.

These files are used to illustrate things as:

- How the parser reads in lines and files
- Duplicate block names
- Types defined at the field versus types defined elsewhere
- Relative language order for text
- Explicit language order by using LID
- HRS conventions of using special comments and to denote language by relative position.

10.1.1 FileName: SE_Illustration.bla

```
DATAMODEL SE_illustration "Illustration of Source Editor Ideas"

LANGUAGES =
 Eng    "English",
 SPN    "Spanish",
 AltEng "Alternative English",
 AltSpn "alternative spanish"  {to be used in the future}

ATTRIBUTES= nodk, norf, noempty

INCLUDE "type_Definitions.INC"

INCLUDE "Procedures.inc"  {See what happens when a program line with include has
another statement in it.}

TYPE tmammals=(n "none",     {This is where the that needs to be looked up is found --
see TP60}
            h "hamster",
            f "ferret",
            o "other")
```

```
BLOCK B1_pet "Type of pet"

   INCLUDE "PetFields.inc" {Note: This file is included several times under different
blocks to illustrate the 'duplicate' file concept}

RULES
          tp1
          IF tp1=S or tp1=y or tp1=dk then
            tp2
          endif
          tp3
          if tp3<>no or tp3=dk then
            tp5
          endif
ENDBLOCK

BLOCK B2_Other "OtherType of pet"
   PARAMETERS IMPORT pipt:string

INCLUDE "PetFields.inc" {Note: This file is included several times under different
blocks to illustrate the 'duplicate' file concept}

FIELDS
TP40 (T4) {This is an example of relative language positioning.}
    {eng} "This is a special question about your other ^pipt:
          @/@/Why do you have this pet?  "
    {spn} "Esta es una pregunta especial sobre sus otros  ^pipt:
          @/@/Why do you have this pet? "
    {alteng} "This is a special question about your other ^pipt:
          @/@/Why do you and your family have this pet? "

/"Why this pet":
open

{mentioned in the paper to show an enumeration defined at a field and
 and enumeration defined elsewhere that needs to be looked up}
TP50 "What type of fish do you have?": set[3] of (n "none", f "fresh water", s "salt water"
)
TP60 "What type of mammals aside from dogs or cats do you have?": TMammals

ENDBLOCK

Fields
          Q1 (Q1) "Are you ready to answer questions?":(y,n) {mentioned in the paper to
show a statement}

          Q2 (Q2) "Do you have any pets?" :TYesNoV2


          Q3 eng "How many pets do you have?" /"Number of pets"
          :SERange1, dk
```

```
DogPets:B1_pet
CatPets:B1_pet
RodentPets:B1_pet

otherPets:B2_Other
Q4 (E1) {eng}"Thank you" {spn} " " {alteng} "THANK YOU"
:(continue)

LOCALS pettype, p:string
RULES
Q1
IF q1=y or q1=dk then
Q2
          if q2=y or q1=dk then
            q3
            p:='dogs'
            TXT_Pettype(p,pettype ) {HRS uses procedures to fill text versus putting it in
the rules}
            DogPets
            pettype:='cats'
            catPets
            pettype:='rats'
            RodentPets
            pettype:='not mentioned'
            otherPets (  pettype)
          endif
ENDIF
ENDMODEL
```

10.1.2 FileName: type_Definitions.INC

```
type
      {V1 has 'gaps' in the statement elements which need to be filled in by the BDO}
          TYesNoV1 =( Y (2) "yes",
                    N (3) "no" )

          {V2 has all statement element completely filled in}
          TYesNoV2 = (Y (1) eng "yes" spn "si" alteng "YES",
                    N (5) eng "no" spn "no" alteng "NO"), dk, norf, empty

          SERange1= 1..15, rf

Tmonth =( jan (1) "january",
        feb (2) "february",
        mar (3) "march",
        apr (4) "april",
        may (5) "may",
        jun "june",
        jul "july",
        aug "august",
        sep "september",
```

```
        oct "october",
        nov "november",
        dec "december")

Tinteger = -999..999
T2005ToPresent =(t2005,
t2006, t2007)

{just to show some of the Spanish diacriticals}
Tspnspc =( s "Sección",
         e "él" ,
         u "en los últimos dos años"), dk
```

10.1.3 FileName: Procedures.inc

```
PROCEDURE TXT_Pettype
parameters
inword:string
export outword:string
{translates the fill text between english and spanish}
RULES

IF   inword='dogs' then
   if activelanguage <> eng and activelanguage <>alteng then
      outword:='perros'
   ELSE
       outword:=inword
   endif
ELSEIF INWORD='cats'
       then   outword:='gatos'
else
 outword:=inword
ENDIF
endprocedure

Block BA3
{This block is not used. It's just to show an include file being called by an include file}
include "petfields.inc"
endblock

           BLOCK BB "BB version 2" {an example of a duplicate block name}
            fields bb1 "Pick a number between 1 and 3 ":1..3, empty, dk, rf
           ENDBLOCK
Table BC
fields
           BC "put some words in . . .":open
           Block BB "BB version 1" {an example of a duplicate block name}
           fields bb1 "Pick a number between 1 and 5 ":1..5, empty

           endblock
```

ENDTABLE {Blaise is not strict on the end token for Blocks, Procedures, Datamodel, and Table}

10.2 Overview of Tables and Procedures used by the Source Editor System

This table is a list of procedures and tables used by Source Editor System programs, with a brief description of the table or procedure and a column listing which application module uses it. The column on the far left indicates whether a table or procedures appears in the appendix.

In-cluded in the appendix	Tbl or Proc	Name	Brief description of table or procedure	PARSER	LOOK-UP	MAKEBDO	Pre-PROCESS	MERGER	WRITER
y	T	Blocks	Blocks, Prodecures, Tables, and DataModel defined	x	x	x			
y	T	FieldType	Fields, Auxfields, Parameter Fields, Locals and the parsed type number associated with the field	x		x			
y	T	FileLines	Lines read from the .bla and .inc files while parsing through the data model	x		x			
y	T	Files	the file(s) that started the parsing and all included files	x		x			
n	T	LgcBlocks	the logic blocking structure	x					
y	T	Token	The tokenized Blaise Statements	x	x	x			
n	T	TypeCode	Enumerated code name and implicit/explicit code number	x					
n	T	TypeLEQD_language	All language text for fields, descriptors and enumerations	x					
n	T	TypeNCDKRFE	The dk/nodk, rf/norf, empty/noempty associated with a type	x					
n	T	TypeOf	Parsed type number and description of user defined or system defined datatype	x		x			
n	T	TypeSet	Parsed type number and description of type defined as an array or set	x					
n	P	SEB00	empties table ITTemplate		x				
n	T	ITTemplate	Table structure template use to write data to		x				
y	P	SEB01	Puts together information from tables and populating the table with information needed to do the look-up		x				
n	P	SEBFT00	empties table FT		x				
n	T	FT			x				

y	P	SEBFT01	Selects the records of interest need for the look-up process and puts them in FT		x				
y	T	ftRefTbl	Resultant look-up table after program does the work using the FT table		x	x			
y	T	Wtable	Work table specifying how to expand/make BDO space for languages and language identifiers			x			
y	T	BDOVariantsUniverse	Contains the blaise statement token order			x			
y	T	LanguageOrder	Contains the langauge order in the parsed datamodel and the language order to be used when 'writing' /creating the MergedBDO table			x			
y	T	SE_DatabaseDictionary	Contains the Tokentype definitions and descriptions used in the MergedBDO Table			x			
y	P	MakeTokeBDOPrep	Puts together information from tables and makes a view used by the program which makes the mergedBDO			x			
n	T	BDOVariantsUniverseRoll	A rolled-up, collapse view of information from the table BDOVariantsUniverse			x			
n	T	dt40	a temp table used by the program making the MergedBDO to reorder language			x			
n	T	TokenDBOTest	a temp/test debug/back up copy of BDO records prior to having columns renamed and tokentypes created to match the SE_DatabaseDictionary tokentypes			x			
n	T	tta	a temp table			x			
y	P	MakeMergedBDO	puts information together from tables to make the mergedBDO table			x			
n	T	DupFilesB	Table with Duplicate Block Names			x			
n	T	DupFilesA10	Table with Duplicate Include file use			x			
n	P	SE0_makeFromTest	Puts tables together to make table TTA			x			
n	P	SE5_1_S0	empties column TokenTypeBDO in TTA			x			
n	P	SE5_1_S1	Puts in the SE Dictionary token type name			x			
n	P	SE5_2_S2	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE5_3_S	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE5_4_S	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE6_1_F	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE6_2_FA	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE6_3_FL	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE6_4_FP	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE6_5_t	Puts in the SE Dictionary token type name			x			

			using another criteria						
n	P	SE7	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE8_1_clean	Cleaning up data in a column left over from the processing program			x			
n	P	SE91	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SE92	Puts in the SE Dictionary token type name using another criteria			x			
n	P	SETypeNoEndZero	Cleaning up data in a column left over from the processing program			x			
n	P	SEMergerBDOKey	the table needs to have keys defined			x			
n	P	MakeftRefTblRollup	The procedure to make the table			x			
n	T	ftRefTblRollup	A rolled-up, collapse view of information from the table ftRefTbl			x			
y	T	MergedBDO	The table to merge updates into and to write revised files from			x		x	x
n	T	AllCodeMergInfo	Special Input for LS language update				x		
n	T	HRS06_cod_LANGUAGES	Special Input for LS language update				x		
y	T	UserInputFormat	User/Bulk updates to be source files				x	x	
n	T	UserInputProcessed	UserInputFormat table with processing updates to match the merge log of records not processed					tbd	
n	T	UserInputFormat_Enum	preprocessor input for PS generated var file				tbd		
n	T	UserInputFormat_Qtext	preprocessor input for PS generated cod file				tbd		

10.3 Parser Tables[9]

10.3.1 Table: Files

This table keeps track of the files used or included in the datamodel. Each time a file is referenced or included a unique file reference number, FLRefNo, is generated. An include file used several times will have more than one file reference number associated with a file name, see File Name 'petfields.inc'.

This table also has information as to which file called the include file, FlNestPath[10]; how deep or nested the call is, FlNest; and the number of lines in the file, FlLnCnt.

[9] The input sample source files were modified for better readability after the tables were generated. The data in the tables will differ slightly from source files.

[10] The path read left to right moving from the lowest level to the highest. For example file reference number 4, was include from file reference number 3, Procedures.inc, which was included from file reference number 1, SE_Illustration.bla.

Aside: If you want to rename the source editor update files, change the file name in this table prior to making the MergedBDO or change the filename in the MergedBDO table.

FlRefNo	FIName	FIPath	FIDate	FILnCnt	FINest	FINestPath	FLBegTime	FLEndTime	FLTimeElapsed
2	type_Definitions.INC	V:\2006HRS\2007_offyear_Activities\BlaiseConference07\SourceEditorPaper\BlaiseExample\	07/23/2007 7:49:56 PM	34	2	2.1.	589977546	589983218	5672
4	petfields.inc	V:\2006HRS\2007_offyear_Activities\BlaiseConference07\SourceEditorPaper\BlaiseExample\	07/23/2007 3:45:30 PM	12	3	4.3.1.	589984281	589986468	2187
3	Procedures.inc	V:\2006HRS\2007_offyear_Activities\BlaiseConference07\SourceEditorPaper\BlaiseExample\	07/24/2007 3:47:18 PM	41	2	3.1.	589984281	589988125	3844
5	PetFields.inc	V:\2006HRS\2007_offyear_Activities\BlaiseConference07\SourceEditorPaper\BlaiseExample\	07/23/2007 3:45:30 PM	12	2	5.1.	589989093	589991015	1922
6	PetFields.inc	V:\2006HRS\2007_offyear_Activities\BlaiseConference07\SourceEditorPaper\BlaiseExample\	07/23/2007 3:45:30 PM	12	2	6.1.	589991718	589993640	1922
1	SE_Illustration.bla	V:\2006HRS\2007_offyear_Activities\BlaiseConference07\SourceEditorPaper\BlaiseExample\	07/26/2007 2:59:08 PM	99	1	1.	589991718	589997812	6094

10.3.2 Table: FileLines

This table contains the file lines as read by the Blaise program. The line number, LnNo, is a unique program line identifier. This is necessary because an include file can be referenced more than once. The line content, of an included file, can have a different meaning depending on how it's included. (see petfields and field TP1 as an example. In one case field TP1 is defined under block B2_Other. In another case field TP1 is defined under block BA3.

Note: The 34+12+26+93 lines of code from the source files translate into 34 + (12x3) + 26 + 93 lines of code due to multiple including of the file petfields.inc

Note: The file reference number, the line number within the include file does not necessarily follow the sequence of the program line number, LnNo.

Note: Some records have been removed from the table records for brevity.

FlRefNo	LnNo	LnLength	FILnCnt	FILn
1	1	63	1	datamodel SE_illustration "Illustration of Source Editor Ideas"
1	2	0	2	
1	3	25	3	languages =Eng "English",
1	4	16	4	SPN "Spanish",
1	5	34	5	AltEng "Alternative English" ,

FlRefNo	LnNo	LnLength	FlLnCnt	FlLn
1	6	0	6	
1	7	61	7	altspn "alternative spaniah" {to be uswd in the future}
1	8	0	8	
1	9	31	9	{Add AltSpn language later}
1	10	31	10	attributes= nodk, norf, noempty
1	11	30	11	include "type_Definitions.INC"
2	12	0	1	
2	13	4	2	type
2	14	87	3	{V1 has 'gaps' in the statement elements which need to be filled in by the BDO}
2	15	25	4	TYesNoV1 =(Y (2) "yes",
2	16	25	5	N (3) "no")
2	17	0	6	
2	18	51	7	{V2 has all statement element completly filled in}
2	19	51	8	TYesNoV2 = (Y (1) eng "yes" spn "si" alteng "YES",
2	20	66	9	N (5) eng "no" spn "no" alteng "NO"), dk, norf, empty
2	41	47	30	{just to show some of the spanish diacriticals}
2	42	23	31	Tspnspc =(s "Sección",
2	43	19	32	e "él" ,
2	44	43	33	u "en los últimos dos años"), dk
2	45	11	34	
1	46	0	12	
1	47	44	13	include "Procedures.inc" {it's an odd place}
3	48	21	1	procedure TXT_Pettype
3	49	10	2	parameters
3	50	13	3	inword:string
3	67	60	20	endprocedure {blaise complier isn't too pick about its ends}
3	68	0	21	
3	69	0	22	
3	70	9	23	Block BA3
3	71	85	24	{This block isn't used. It's just to show an include being called by an include file}
3	72	23	25	include "petfields.inc"
4	73	26	1	{field for the pet blocks}
4	74	32	2	type TYesNoV2 = (Y (11) "yes",
4	75	21	3	N "no",
4	76	29	4	s "Several"), dk
4	77	0	5	
4	78	50	6	fields TP1 eng "For this type of pet, ^pettype, @/
4	79	62	7	@/Do you have any? " alteng"For this type of pet,

FlRefNo	LnNo	LnLength	FlLnCnt	FlLn
				^pettype, @/
4	80	30	8	@/Do you have any? ":TYesNoV2
4	81	44	9	TP2 "How many ^pettype do you have?": 0..100
4	82	74	10	tp3 (Pet3) "Is there anything special you want to mention about ^pettype?"
4	83	27	11	: (Yes,Maybe, No) , dk, rf
4	84	45	12	TP5 (PT4) "If so what you like to say?": open
3	85	8	26	endblock
3	86	0	27	
3	87	62	28	Block BB "BB version 2" {an example of a duplicate block name}
3	88	64	29	fields bb1 "Pick a number between 1 and 3 ":1..3, empty, dk, rf
3	89	8	30	endblock
3	90	0	31	
3	91	8	32	table BC
3	92	0	33	
3	93	6	34	fields
3	94	34	35	BC "put some words in . . .":open
3	95	1	36	
3	96	63	37	Block BB "BB version 1" {an example of a duplicate block name}
3	97	56	38	fields bb1 "Pick a number between 1 and 5 ":1..5, empty
3	98	0	39	
3	99	9	40	endblock
3	100	8	41	endtable
1	101	44	13	include "Procedures.inc" {it's an odd place}
1	102	0	14	
1	103	24	15	type tmammals=(n "none",
1	104	26	16	h "hamster",
1	105	25	17	f "ferret",
1	106	24	18	o "other")
1	107	0	19	
1	108	26	20	block B1_pet "Type of pet"
1	109	0	21	
1	110	23	22	include "PetFields.inc"
5	111	26	1	{field for the pet blocks}
5	112	32	2	type TYesNoV2 = (Y (11) "yes",
5	113	21	3	N "no",
5	114	29	4	s "Several"), dk
5	115	0	5	
5	116	50	6	fields TP1 eng "For this type of pet, ^pettype, @/

FlRefNo	LnNo	LnLength	FlLnCnt	FlLn
5	117	62	7	@/Do you have any? " alteng"For this type of pet, ^pettype, @/
5	118	30	8	@/Do you have any? ":TYesNoV2
5	119	44	9	TP2 "How many ^pettype do you have?": 0..100
5	120	74	10	tp3 (Pet3) "Is there anything special you want to mention about ^pettype?"
5	121	27	11	: (Yes,Maybe, No) , dk, rf
5	122	45	12	TP5 (PT4) "If so what you like to say?": open
1	123	6	23	rules
1	124	3	24	tp1
1	125	38	25	IF tp1=S or tp1=y or tp1=dk then
1	135	33	35	block B2_Other "OtherType of pet"
1	136	32	36	parameters import pipt:string
1	137	0	37	
1	138	23	38	include "PetFields.inc"
6	139	26	1	{field for the pet blocks}
6	140	32	2	type TYesNoV2 = (Y (11) "yes",
6	141	21	3	N "no",
6	150	45	12	TP5 (PT4) "If so what you like to say?": open
1	151	0	39	
1	152	6	40	fields
1	153	9	41	TP40 (T4)
1	154	61	42	{eng} "This is a special question about your other ^pipt:
1	155	43	43	@/@/Why do you have this pet? "
1	156	60	44	Esta es una pregunta especial sobre sus otros ^pipt:
1	157	42	45	@/@/Why do you have this pet? "
1	158	64	46	{alteng} "This is a special question about your other ^pipt:
1	159	58	47	@/@/Why do you and your family have this pet? "
1	160	0	48	
1	161	16	49	/"Why this pet":
1	162	4	50	open
1	163	0	51	
1	164	93	52	TP50 "What type of fish do you have?": set[3] of (n "none", f "fresh water", s "salt water")
1	165	74	53	TP60 "What type of mammals aside from dogs or cats do you have?": TMammals
1	169	6	57	Fields
1	170	51	58	Q1 (Q1) "Are you ready to answer questions?":(y,n)

FlRefNo	LnNo	LnLength	FlLnCnt	FlLn
1	210	8	98	endmodel
1	211	0	99	

10.3.3 Table: Blocks

This table contains one record for each block, table, datamodel, or procedure defined. A unique number identifies each block structure.

Note: There are instances where the same block name is defined under a different path. In cases like this the block number must be used to distinguish the block uniquely. See Block name “BB”.

FlRefNo	BlkNo	Blktype	Blkname	Blkdescriptor	Blkpath	Blknest
1	1	D	SE_illustration	"Illustration of Source Editor Ideas"	1..	1
3	2	P	TXT_Pettype		2.1..	2
3	3	B	BA3		3.1..	2
3	4	B	BB	"BB version 2"	4.1..	2
3	5	T	BC		5.1..	2
3	6	B	BB	"BB version 1"	6.5.1..	3
1	7	B	B1_pet	"Type of pet"	7.1..	2
1	8	B	B2_Other	"OtherType of pet"	8.1..	2

10.3.4 Table: FieldType

For each field, auxfield, local field, or parameter field, the field name is listed, and given a unique field number, FieldNo. Each list of field(s) is assigned a unique type. Most lists of fields have only one element. See how this statement “locals pettype, p:string” appears in the table below for the fields “pettype” and “p”.

FlRefNo	BlkNo	FieldNo	FieldType	FieldName	TypeNo
3	3	0	B	BA3	13
3	4	0	B	BB	19
3	6	0	B	BB	23
1	7	0	B	B1_pet	26
1	8	0	B	B2_Other	32
1	1	0	D	SE_illustration	1
3	2	0	P	TXT_Pettype	10
3	5	0	T	BC	21
3	2	1	PI	inword	11
3	2	2	PE	outword	12
4	3	3	F	TP1	15
4	3	4	F	TP2	16
4	3	5	F	tp3	17
3	3	6	F	TP5	18

FlRefNo	BlkNo	FieldNo	FieldType	FieldName	TypeNo
3	4	7	F	bb1	20
3	5	8	F	BC	22
3	6	9	F	bb1	24
5	7	10	F	TP1	28
5	7	11	F	TP2	29
5	7	12	F	tp3	30
1	7	13	F	TP5	31
1	8	14	PI	pipt	33
6	8	15	F	TP1	35
6	8	16	F	TP2	36
6	8	17	F	tp3	37
1	8	18	F	TP5	38
1	8	19	F	TP40	39
1	8	20	F	TP50	40

FlRefNo	BlkNo	FieldNo	FieldType	FieldName	TypeNo
1	8	21	F	TP60	41
1	1	22	F	Q1	42
1	1	23	F	Q2	43
1	1	24	F	Q3	44
1	1	25	F	DogPets	45
1	1	26	F	CatPets	46

FlRefNo	BlkNo	FieldNo	FieldType	FieldName	TypeNo
1	1	27	F	RodentPets	47
1	1	28	F	otherPets	48
1	1	29	F	Q4	49
1	1	30	L	pettype	50
1	1	31	L	p	50

10.3.5 Table: Token

This table contains one record per token. A token is uniquely identified by the beginning program line number and line column of the token. The token type and other columns provide secondary keys and ancillary information needed to identify certain statements token types.

Note: Only a few records have been included to show how a field statement is parsed into tokens.

FlRefNo	LnNoBeg	ColBeg	LnNoEnd	ColEnd	Token	TokenNew	TokenUpdate	TokenType	LngNo	BlkNo	FldNo	TypeNo	Keyword	KeywordOption	CCode	CVal	LnglD
1	170	2	170	3	Q1			FName	0	1	22	41	FIELDS	Question text		0	
1	170	4	170	4				WhiteSpaceBlank	0	1	22	41	FIELDS	Question text		0	
1	170	5	170	8	(Q1)			FTag	0	1	22	42	FIELDS	Question text		0	
1	170	9	170	9				WhiteSpaceBlank	0	1	22	42	FIELDS	Question text		0	
1	170	10	170	45	"Are you ready to answer questions?"			Text	1	1	22	42	FIELDS	Question text		0	
1	170	46	170	46	:			FDefnSep Separator :	0	1	22	42	FIELDS	Question text		0	
1	170	47	170	47	(			EnumBeg Separator (	0	1	22	42	FIELDS	Question text		0	
1	170	48	170	48	y			EnumCodeName	0	1	22	42	FIELDS	ENUMERATED	y	0	
1	170	49	170	49	,			EnumSep Separator ,	0	1	22	42	FIELDS	ENUMERATED	y	1	
1	170	50	170	50	n			EnumCod	0	1	22	42	FIEL	ENUME	n	1	

FlRefNo	LnNoBeg	ColBeg	LnNoEnd	ColEnd	Token	TokenNew	TokenUpdate	TokenType	LngNo	BlkNo	FldNo	TypeNo	Keyword	KeywordOption	CCode	CVal	LnglD
								eName					DS	RATED			
1	170	51	170	51	)			EnumEndSep Separator)	0	1	22	42	FIELDS	ENUMERATED		0	

10.4 BDO Tables

Along with tables from the parser, other tables are used in the creation of the MergedBDO table. These are examples of table generated by the type definition cross reference application, the language order table, the work table language statement expansion, the table of the universe of all statement elements, and the dictionary of token types.

10.4.1 Table: ftRefTbl - Type Definition Cross Reference

The Type Definition Cross Reference table is made by the Itable application. There is one record per type number, indicating where the type is actually defined, in the end type number column. If the type number is the same as the end type number, it's a system defined data type, or the data type defined directly at the field. If the type number and end type number are different, it's a user defined data type that is defined at the type number listed in the column of type number end, i.e. the type is defined at a remote location.

Note: Types names have the same issues as fieldnames. There can be duplicate names defined in different path.

BlkNo	Typeno	Blkpath	FName	BlkNoEnd	TypeNoEnd	TokenType
1	45	1..	B1_pet	0	0	FTName
1	46	1..	B1_pet	0	0	FTName
1	47	1..	B1_pet	0	0	FTName
1	48	1..	B2_Other	0	0	FTName
8	38	8.1..	open	8	38	FTName
8	39	8.1..	open	8	39	FTName
7	31	7.1..	open	7	31	FTName
5	22	5.1..	open	5	22	FTName
3	18	3.1..	open	3	18	FTName
1	5	1..	SERange1	1	5	TName
1	44	1..	SERange1	1	5	FTName
8	33	8.1..	string	8	33	FTName
2	11	2.1..	string	2	11	FTName
2	12	2.1..	string	2	12	FTName
1	50	1..	string	1	50	FTName
1	8	1..	T2005ToPresent	1	8	TName

BlkNo	Typeno	Blkpath	FName	BlkNoEnd	TypeNoEnd	TokenType
1	7	1..	Tinteger	1	7	TName
8	41	8.1..	TMammals	1	25	FTName
1	25	1..	tmammals	1	25	TName
1	6	1..	Tmonth	1	6	TName
1	9	1..	Tspnspc	1	9	TName
1	3	1..	TYesNoV1	1	3	TName
8	34	8.1..	TYesNoV2	8	34	TName
8	35	8.1..	TYesNoV2	8	34	FTName
7	27	7.1..	TYesNoV2	7	27	TName
7	28	7.1..	TYesNoV2	7	27	FTName
3	14	3.1..	TYesNoV2	3	14	TName
3	15	3.1..	TYesNoV2	3	14	FTName
1	4	1..	TYesNoV2	1	4	TName
1	43	1..	TYesNoV2	1	4	FTName

10.4.2 Table: LanguageOrder

This table is used to specify the language order of the parsed datamodel, and the desired language order of the updated output files. Language reordering is rarely done. This table is updated by hand as needed. The language identifier, language order defined (parsed), and language order new definition (output) are the main columns of interest.

LngID	LngOrderDefined	LngOrderNewDefn	TextEnum	TextQuestion	TextDescriptor	TextFillEnum	TextFillQuestion	TextFillDescriptor
ENG	1	4	True	True	True	" pgm enum fill coreng"	" pgm question fill coreng"	" pgm descriptor fill coreng"
SPN	2	2	True	True	False	" pgm enum fill corspn"	" pgm question fill corspn"	" pgm descriptor fill corspn"
ALTENG	3	3	True	True	False	" pgm enum fill exteng"	" pgm question fill exteng"	" pgm descriptor fill exteng"
ALTSPN	4	1	True	True	False	" pgm enum fill extspn"	" pgm question fill extspn"	" pgm descriptor fill extspn"

10.4.3 Table: WTable – statement expansion for language tokens

This table is updated by hand. The update information in the expansion of statements, particularly in the language area, can be limited. This is done to deal with issues of scale and volume which can slow down the processing.

Note: The order_general column codes 141 is the code for the LID for question text, 142 for the question text, 161 for the LID for descriptor text, and 162 the descriptor text[11][12].

indexOrder	order_general	Ingno	BegMidEnd
0	130	0	1
4	142	2	0
15	150	0	2
22	161	4	0
23	162	4	0
30	170	0	3
31	310	0	0
35	331	1	0
36	332	1	0
37	331	2	0
38	332	2	0
39	331	3	0
40	332	3	0
49	340	0	4

[11] See the BDOVariantsUniverse Table for other code meanings.
[12] In the table the implicit space will be added for question text for the second descriptor LID and text for the fourth language, and enumeration LID and text for first three languages. The records in the table are determine by the programmer, space, and nature of the input for the update run.

10.4.4 Table: BDOVariantsUniverse – Blaise Statement token order

This table outlines all the elements and order of Blaise Statements tokens. This table should not be updated unless new statements or elements of statements have been discovered.

I've included the entirety of this table because it is a key function and analysis piece. The primary and secondary merge keys are indicated for the various statement tokens. This information was essential in writing the code and algorithms used in the merger program.

Note: The general statement order column here is the same column used in the Wtable.

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
10															mergeKeys													
20	D	B	T	P	F	A	L	PF	Types=		TokenType abstract Description	(Parsed) TokenType	TokenBDO TokenType		DBTP (Block Num)	Blk / Fld Type	Type Num	Field Num	Type or Field Name	Lang Ident (LID)	Code Name	Attrib Type	Defn Order	TokenType	main stack group	split stack grouping	default W/Grp	
30	y	y	y	y						*	Keyword DataModel/Block/Table/Procedure structureBegKeyword	StrctBeg	StrctBeg	21	x	s	or x	s	s					x	10	1	1	
40	y	y	y	y							name	StrctName	StrctName	22	x	s	or x	s	s					x	10	1	2	
50	y	y	y	n							desc	StrctDesc	StrctDesc	23	x	s	or x	s	s					x	10	1	3	
60	y	y	y	y							structureEndKeyword	StrctEnd	StrctEnd	900	x	s	or x	s	s					x	10	2	4	
100					y	y	y	y	y	*	Keyword	FTBeg	FTBeg	100	x	s		x	s					x	19	1		

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
											FIELDS/AUXFIELDS/LOCALS/PARAMETERS/TYPES																	
110								y	n	**	Keyword Import/Export/Transit	FType	FType	110	x	s		x	s					x	19	1		
120					y	y	y	y	n		FieldName	FName	FName	121	x	s	s	x1	s				x	x	19	1		
130					y	y	y	y	n		FieldName Sep ,	FNameSeparator ,	FNameSep	122	x	s	s	x	s				x	x	19	1		
140					y	y	n	n	n		TypeTag	FTag	FTag	130	x	s	x	s	s2					x	20	1		
150					y	y	n	n	y	l	TextLID	TextLID	FTTextLID	141	x	s	x	s	s2	x			s	x	20	1		
160					y	y	n	n	y	l	Text	Text	FTText	142	x	s	x	s	s2	x			s	x	20	1		
170					y	y	n	n	n		DescSep /	DescSeparator /	FDescSep	150	x	s	x	s	s2					x	20	1		
180					y	y	n	n	n	l	DescTextLID	DescTextLID	FDescTextLID	161	x	s	x	s	s2	x			s	x	20	1		
190					y	y	n	n	n	l	DescText	DescText	FDescText	162	x	s	x	s	s2	x			s	x	20	1		
200					y	y	y	y	n		Type Sep :	FDefnSep Separator :	FDefnSep	170	x	s	x	s	s2					x	20	1		
210	y				n	n	n	n	n		Languages	LANGUAGES	FTBeg	25	x	s	x		s2					x				
220	y_Tlng				n	n	n	n	y		TypeNameDef	Tname	TName	26	x	s	x	s	s2					x	20	1		
230	y				n	n	n	n	y		Type =	TNameSep Separator =	TNameSep	28	x	s	x	s	s2					x	20	1		
240	y				n	n	n	n	y			LANGUAGESSeparator		27	x	s	x	s	s2					x	20	1		

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
												=																
250																												
260					y	y	n	y	y	*	Keyword Array/Set	ASBeg	ASBeg	200	x	s	x	s	s2					x	20	1	B	
270					y	y	n	y	y		[	ASRngBegSeparator [	ASRngBeg	210	x	s	x	s	s2					x	20	1	B	
280					y	y	n	y	y		min	RngValBeg	ASRngValBeg	220	x	s	x	s	s2				x	x	20	1	B	
290					y	y	n	y	y	*	..	Separator ..	ASRngValSep	230	x	s	x	s	s2				x	x	20	1	B	B2
300					y	y	n	y	y		max	RngValEnd	ASRngValEnd	240	x	s	x	s	s2				x	x	20	1	B	B2
310					y	y	n	y	y		sep ,	Separator ,	ASRngValSetSep	241	x	s	x	s	s2				x	x	20	1	B	B2
320					y	y	n	y	y		]	ASRngEndSeparator]	ASRngEnd	250	x	s	x	s	s2					x	20	2	B	B2
330					y	y	n	y	y		of	ASSepOF	ASSep	260	x	s	x	s	s2					x	20	1	B	
340																												
350																												
360					y	y	n	y	y	*	EnumBeg(	EnumBeg Separator (	EnumBeg	300	x	s	x	s	s2					x	20	1	C	1
370					y	y	n	y	y		EnumCodeName	EnumCodeName	EnumCodeName	310	x	s	x	s	s2		x		s	x	21	1	C	2
380					y	y	n	y	y		EnumCodeNumBeg(	EnumCodeNumBegSepSeparator (	EnumCodeNumBeg	321	x	s	x	s	s2		x		s	x	21	1	C	2
390					y	y	n	y	y		EnumCodeNum	EnumCodeNum	EnumCodeNum	322	x	s	x	s	s2		x		s	x	21	1	C	2
400					y	y	n	y	y		EnumCodeNumEnd)	EnumCodeNumEndSepSeparator)	EnumCodeNumEnd	323	x	s	x	s	s2		x		s	x	21	1	C	2
410																												

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
420					y	y	n	y	y	l	EnumDesTextLID	EnumDescTextLID	EnumDesTextLID	331	x	s	x	s	s2	x	x		s	x	21	1	C	2
430					y	y	n	y	y	l	EnumDesText	EnumDescText	EnumDesText	332	x	s	x	s	s2	x	x		s	x	21	1	C	2
440																												
450					y	y	n	y	y		EnumCodeSep,	EnumSepSeparator ,	EnumCodeSep	340	x	s	x	s	s2		x		s	x	21	1	C	2
460					y	y	n	y	y		EnumEnd)	EnumEndSep Separator)	EnumEnd	350	x	s	x	s	s2					x	20	2	C	1
470																												
480																												
490					y	y	y	y	y	*	TypeNameRef (system or user defn)	FTName	FTName	400	x	s	x	s	s2					x	20	1	D	1
500					y	y	n	y	y		[	Separator [	FTRngBeg	410	x	s	x	s	s2					x	20	1	D	2
510					y	y	n	y	y		min	RngValBeg	FTRngValBeg	420	x	s	x	s	s2					x	20	1	D	2
520					y	y	n	y	y	*	..	Separator ..	FTRngValSep	430	x	s	x	s	s2					x	20	1	D	3
530					y	y	n	y	y		max	RngValEnd	FTRngValEnd	440	x	s	x	s	s2					x	20	1	D	3
540					y	y	n	y	y		sep ,	Separator ,	FTRngValSetSep	441	x	s	x	s	s2				x	x	20	1	B	B2
550					y	y	n	y	y		]	Separator]	FTRngEnd	450	x	s	x	s	s2					x	20	2	D	2
560																												
570																												
580	y	n	n	n	y	y	n	y	y	**	Keyword ATTRIBUTES	ATTR	ATTR	600	x	s							x	x		1		0
590	y	n	n	n	y	y	n	y	y		ATTRIBUTES =	ATTRSepSeparator =	ATTRSep	601	x	s							x	x		1		0
600	y	n	n	n	y	y	n	y	y	*	sep ,	ATTRIBDRESepSeparato		602	x	s		s				x	x	x		1		1

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
												r ,																
610	y	n	n	n	y	y	n	y	y		toggle type(DK/NDK or RF/RF or Empty or non-Empty)			610	x	s	x	s	s2			x	x	x		1		1
620																													
630	y	n	n	n	y	y	n	y	y		attrib sep ,	ATTRIBDRESepSeparator ,	ATTRDSep	611		s							x					
640	y	n	n	n	y	y	n	y	y		DKNDK sep ,	ATTRIBDRESepSeparator ,	ATTRDSep	612		s							x					
650	y	n	n	n	y	y	n	y	y		DKNDK	ATTRDK	ATTRD	613		s							x					
670	y	n	n	n	y	y	n	y	y		RFNRF sep ,	ATTRIBDRESepSeparator ,	ATTRRSep	622		s							x					
680	y	n	n	n	y	y	n	y	y		RFNRF	ATTRNRF	ATTRR	623		s							x					
700	y	n	n	n	y	y	n	y	y		EMPTYNEMPTY sep ,	ATTRIBDRESepSeparator ,	ATTRESep	632		s							x					
710	y	n	n	n	y	y	n	y	y		EMPTYNEMPTY	ATTRE	ATTRE	633		s							x					
660	y	n	n	n	y	y	n	y	y		NDK	ATTRNDK	ATTRNDK	614		s							x					
690	y	n	n	n	y	y	n	y	y		NRF	ATTRRF	ATTRNRF	624		s							x					
720	y	n	n	n	y	y	n	y	y		EMPTYNEMPTY	ATTRNE	ATTRNE	634		s							x					
760											Comment	COMMENT	Comment	11	x								x	x	30	1		
770											WhiteSpaceBlank	WhiteSpaceBlank	WhiteSpaceBlank	12	x								x	x	80	1		
780											WhiteSpaceTab	WhiteSpace	WhiteSpaceTa	13	x								x	x	90	1		

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
												Tab	b															
790																												
800										*	Keyword Router	ROUTER		40											40	1		
810											Name			41											40	1		
820										*	Keyword Alien	ALIEN		45	x									x	40	2		
830											(	Alien Separator (		46											40	2		
860											MethodName | DLLFileName	Alien second		49											40	2		
880											DLLProcName | DLLProcNumber	Alien third		51											40	2		
900																												
910																												
920																												
930										*	Keyword Include (key word)	INCLUDE		31	x								x	x	50	1		
940											Include file name	File Name	File Name	32	x								x	x	50	1		
950																												
960										*	Keyword Parallel	PARALLEL		71	x								x	x	60	1		
970											name			72	x			x					x	x	60	1		
980											=			73	x			x					x	x	60	1		
990											Field			74	x			x					x	x	60	1		
1000																												
1010										*	Keyword Primary/Second	PRIMARY		61	x			x						x	70	1		

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
											ary																	
1020											field name	Name		62	x			x1					x	x	70	1		
1030											comma	Separator ,		63	x			x					x	x	70	1		
1040										*	Keyword Primary/Secondary	SECONDARY		65														
1050											field name	Name		66														
1060											comma	Separator ,		67														
1070																												
1080	y	y	y	y							Keyword Rules			801														
1090	y	y	y	y							Rules Text	RulesBlock		800														
1104												text-double quote		4														
1110												text-single quote		3														
1120												Nonwhite-delim		2														
1140												Separator ;		10														
1130												Nonwhite		1														
1150												KeyWord		0														
1160																												
1170									*		- triggers/controls BDO records																	
1180									x		-primary keys																	
1190									s		- seconary keys																	
1200	D	B	T	P	F	A	L	PF	Typ		Keyword values from Token Region																	

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
									es=		Fields(FALPFTypes)Rules																	
1210											ALIEN																	
1220											ATTRIBUTES																	
1230						x					AUXFIELDS																	
1240		x									BLOCK																	
1250	x										DATAMODEL																	
1260		x	1	1							ENDBLOCK																	
1270	x										ENDMODEL																	
1280		1	1	x							ENDPROCEDURE																	
1290		1	x	1							ENDTABLE																	
1300					x						FIELDS																	
1310											INCLUDE																	
1320									1		LANGUAGES																	
1330							x				LOCALS																	
1340								x			PARAMETERS																	
1350								x			PARAMETERS EXPORT																	
1360								x			PARAMETERS IMPORT																	
1370								x			PARAMETERS TRANSIT																	
1380											PRIMARY																	
1390				x							PROCEDURE																	
1400											ROUTER																	
1410											RULES																	

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
1420											SECONDARY																	
1430			x								TABLE																	
1440									x		TYPE																	
1450																												
1460																												
1470											PARALLEL																	
1480											USES																	
1490											EXTERNAL	EXTERNALS Meta Information																
1500												EXTERNALS reference name																
1510																												
1520																												
1530																												
1540																												
730																												
740																												
750																												
840											ProgID	ProgID		47											40	2		
850											,	Alien secondSeparator ,		48											40	2		
890											)	Alien Separator)		52											40	2		
870											,	Alien thirdSeparator ,		50											40	2		

ord	SD	SB	ST	SP	FF	FA	FL	FP	FT	KeyLng	TT_Abstract	TT_Parsed	TT_TBDO	Order_General	KeyDBTP	KeySOrFType	KeyTNum	KeyFNum	KeyTorFName	KeyLID	KeyCodeName	KeyAttrib	KeyDefnOrder	KeyTT	GrpMainStack	GrpStackSplit	Grp	GrpSplit
1100																												
1102												KeyWordLogic		5														

10.4.5 Table: SE_DatabaseDictionary – A Dictionary of Token Types

This table is used to determine the names of the token types used in the MergedBDO. It is through these names that the token types have more meaning than the token types described by the parser.

Note: TokenTypeID corresponds to Order_General in the table BDOVariantsUniverse. In this dictionary these same abstracted token types would have one or more token types depending on what type of structure it is in.

Note: Records in this table are reduced.

TokenTypeID	D	B	TBL	P	Fld	Aux	Local	ParmFld	TYP	TokenType	Definition	Restrictions	Mandatory	Keyword
11	-	-	-	-	-	-	-	-	-	Comment	Text enclosed by braces not evaluated by the compiler includes { }		False	False
21	y				-	-	-	-	-	Datamodel	Declares and names a data model.		True	True
21		y			-	-	-	-	-	Block	Defines a type as a block: a user defined sub model of related fields and rules		True	True
22		y			-	-	-	-	-	BlockName	Name of block		True	False
100	-	-	-	-		y				Auxfields	Defines auxiliary fields to store temporary information, only available during the application.		False	True
100	-	-	-	-	y					Fields	indicates the beginning of a fields section. In this section you define one or more fields in the current block.		False	True
142	-	-	-	-	y	y	n	n	y	FieldText	Text related to a Field		False	False
310	-	-	-	-	y	y	n	y	y	CodeName	Defines the domain of a type as a discrete range in which each category is specified as an identifier. Each category identifier may have a code number and a text		False	False
141	-	-	-	-	y	y	n	n	y	FieldTextLangID			False	False
161	-	-	-	-	y	y	n	n	n	DescTextLangID			False	False

10.5 Procedures: Showing Key Relationships between BDO Tables

Putting the SQL of procedures is a concise and cryptic way to share primary and secondary keys used in the table for creating the BDO, and how these pieces get put together. The SQL statements provide clues to entity relationships between tables. If a table or procedure name referenced by the procedure does not appear here, the chart at the beginning of the appendix will provide a brief description of it.

10.5.1 Procedure: SEb01 – populates ITTemplate with records

```
INSERT INTO dbo.ITTemplate
                (BlkNo, TypeNo, Blkpath, Name, TokenType, FldNo)
SELECT     dbo.Token.BlkNo, dbo.Token.TypeNo, dbo.Blocks.Blkpath,
dbo.Token.Token AS Name, dbo.Token.TokenType, dbo.Token.FldNo
FROM        dbo.Token LEFT OUTER JOIN
                dbo.Blocks ON dbo.Token.BlkNo = dbo.Blocks.BlkNo
WHERE     (dbo.Token.TokenType = N'FTName') OR
                (dbo.Token.TokenType = N'TName') OR
                (dbo.Token.TokenType = N'StrctName')
```

10.5.2 Procedure: SEbft01 – populates FT with records from ITTemplate

```
INSERT INTO dbo.ft
                (BlkNo, TypeNo, FldNo, TName)
SELECT     BlkNo, TypeNo, FldNo, Name AS TName
FROM        dbo.ITTemplate
WHERE     (TokenType = N'TName') OR
                (TokenType = N'StrctName')
ORDER BY BlkNo DESC, TypeNo DESC
```

10.5.3 Procedure: MakeTokenBDOPrep – Makes a combine view of several tables

```
SELECT     TOP 100 PERCENT dbo.Token.LnNoBeg, dbo.Token.ColBeg,
dbo.Token.LnNoEnd, dbo.Token.ColEnd, dbo.Token.BlkNo, dbo.Token.FldNo,
                dbo.Token.TypeNo, dbo.Token.CCode, dbo.Token.LngID,
dbo.Token.TokenType, dbo.Token.Keyword, dbo.Token.Token, dbo.Token.TokenNew,
                dbo.Token.TokenUpdate, dbo.Token.KeywordOption, dbo.Token.LngNo,
dbo.Token.CVal, dbo.Token.FlRefNo, dbo.FieldType.FieldType,
                dbo.Blocks.Blktype, dbo.Token.FldNo AS FldNoRng,
dbo.BDOVariantsUniverseRoll.Order_General, dbo.Token.TokenType AS
TokenTypeBDO
FROM        dbo.Token LEFT OUTER JOIN
                dbo.BDOVariantsUniverseRoll ON dbo.Token.TokenType =
dbo.BDOVariantsUniverseRoll.TT_Parsed LEFT OUTER JOIN
                dbo.FieldType ON dbo.Token.FldNo = dbo.FieldType.FieldNo AND
dbo.Token.BlkNo = dbo.FieldType.BlkNo LEFT OUTER JOIN
                dbo.Blocks ON dbo.Token.BlkNo = dbo.Blocks.BlkNo
ORDER BY dbo.Token.LnNoBeg, dbo.Token.ColBeg
```

10.5.4 Procedure: make table tta – backup table where token types get revised

```
SELECT     TOP 100 PERCENT dbo.TokenBDOTest.TokenID,
dbo.TokenBDOTest.LnNoBeg, dbo.TokenBDOTest.ColBeg,
dbo.TokenBDOTest.LnNoEnd,
                dbo.TokenBDOTest.ColEnd, dbo.TokenBDOTest.BlkNo,
dbo.TokenBDOTest.FldNo, dbo.TokenBDOTest.TypeNo, dbo.TokenBDOTest.CCode,
                dbo.TokenBDOTest.Keyword, dbo.TokenBDOTest.Token,
dbo.TokenBDOTest.TokenNew, dbo.TokenBDOTest.TokenUpdate,
                dbo.TokenBDOTest.KeywordOption, dbo.TokenBDOTest.LngNo,
dbo.TokenBDOTest.CVal, dbo.TokenBDOTest.FlRefNo,
                dbo.TokenBDOTest.FldNo AS FldNoRng,
dbo.TokenBDOTest.Order_General, dbo.BDOVariantsUniverseRoll.TT_Parsed AS
TokenType,
                dbo.LanguageOrder.LngID, dbo.TokenBDOTest.Blktype,
dbo.TokenBDOTest.TokenTypeBDO, dbo.TokenBDOTest.FieldType,
                dbo.TokenBDOTest.DupFiles
INTO            dbo.tta
FROM            dbo.TokenBDOTest INNER JOIN
                dbo.BDOVariantsUniverseRoll ON dbo.TokenBDOTest.Order_General =
dbo.BDOVariantsUniverseRoll.Order_General LEFT OUTER JOIN
                dbo.LanguageOrder ON dbo.TokenBDOTest.LngOrderNewDefn =
dbo.LanguageOrder.LngOrderNewDefn
ORDER BY dbo.TokenBDOTest.TokenID, dbo.TokenBDOTest.LnNoBeg,
dbo.TokenBDOTest.ColBeg
```

10.5.5 Procedure: MakeMergedBDO – Makes the table used by the merger

```
SELECT     TOP 100 PERCENT dbo.tta.TokenID, dbo.tta.TokenTypeBDO AS
TokenType, dbo.tta.Token, dbo.tta.BlkNo AS BlockNum,
                dbo.Blocks.Blkname AS BlockName, dbo.tta.FldNo AS FieldNum,
dbo.FieldType.FieldName, dbo.tta.TypeNo AS TypeNum, dbo.tta.LngID AS LangID,
                dbo.tta.CCode AS CodeValue, dbo.tta.LnNoBeg AS LineNumBeg,
dbo.tta.ColBeg AS ColumnNumBeg, dbo.tta.LnNoEnd AS LineNumEnd,
                dbo.tta.ColEnd AS ColumnNumEnd, dbo.tta.FlRefNo AS FileRefNum,
dbo.Files.FlName AS FileName, dbo.tta.TokenUpdate,
                dbo.Blocks.Blkpath AS BlockPath, dbo.FileLines.FlLnCnt AS
FileLineCountBeg, FileLines_1.FlLnCnt AS FileLineCountEnd,
dbo.ftRefTblRollup.BlkNoEnd,
                dbo.ftRefTblRollup.TypeNoEnd, dbo.TypeOF.TypeName,
dbo.DupFilesB.FlName, dbo.DupFilesA10.[count] AS FileUseCount,
                dbo.Files.FlLnCnt AS FileLineNumMax
INTO            dbo.MergedBDO
FROM            dbo.FileLines RIGHT OUTER JOIN
                dbo.DupFilesB INNER JOIN
                dbo.DupFilesA10 ON dbo.DupFilesB.FlName = dbo.DupFilesA10.FlName
RIGHT OUTER JOIN
                dbo.Files ON dbo.DupFilesB.FlRefNo = dbo.Files.FlRefNo RIGHT
OUTER JOIN
                dbo.ftRefTblRollup LEFT OUTER JOIN
```

dbo.TypeOF ON dbo.ftRefTblRollup.TypeNoEnd = dbo.TypeOF.TypeNo RIGHT OUTER JOIN
dbo.tta ON dbo.ftRefTblRollup.Typeno = dbo.tta.TypeNo AND dbo.ftRefTblRollup.BlkNo = dbo.tta.BlkNo LEFT OUTER JOIN
dbo.FieldType ON dbo.tta.BlkNo = dbo.FieldType.BlkNo AND dbo.tta.FldNo = dbo.FieldType.FieldNo LEFT OUTER JOIN
dbo.FileLines FileLines_1 ON dbo.tta.FlRefNo = FileLines_1.FlRefNo AND dbo.tta.LnNoEnd = FileLines_1.LnNo ON
dbo.Files.FlRefNo = dbo.tta.FlRefNo LEFT OUTER JOIN
dbo.Blocks ON dbo.tta.BlkNo = dbo.Blocks.BlkNo ON dbo.FileLines.FlRefNo = dbo.tta.FlRefNo AND dbo.FileLines.LnNo = dbo.tta.LnNoBeg
ORDER BY dbo.tta.TokenID, dbo.tta.LnNoBeg, dbo.tta.ColBeg, dbo.tta.LnNoEnd, dbo.tta.ColEnd, dbo.tta.LngID

10.5.6 Procedure: SE0_makeFromTest

SELECT TOP 100 PERCENT dbo.TokenBDOTest.TokenID, dbo.TokenBDOTest.LnNoBeg, dbo.TokenBDOTest.ColBeg, dbo.TokenBDOTest.LnNoEnd,
dbo.TokenBDOTest.ColEnd, dbo.TokenBDOTest.BlkNo, dbo.TokenBDOTest.FldNo, dbo.TokenBDOTest.TypeNo, dbo.TokenBDOTest.CCode,
dbo.TokenBDOTest.Keyword, dbo.TokenBDOTest.Token, dbo.TokenBDOTest.TokenNew, dbo.TokenBDOTest.TokenUpdate,
dbo.TokenBDOTest.KeywordOption, dbo.TokenBDOTest.LngNo, dbo.TokenBDOTest.CVal, dbo.TokenBDOTest.FlRefNo,
dbo.TokenBDOTest.FldNo AS FldNoRng, dbo.TokenBDOTest.Order_General, dbo.BDOVariantsUniverseRoll.TT_Parsed AS TokenType,
dbo.LanguageOrder.LngID, dbo.TokenBDOTest.Blktype, dbo.TokenBDOTest.TokenTypeBDO, dbo.TokenBDOTest.FieldType,
dbo.TokenBDOTest.DupFiles
INTO dbo.tta
FROM dbo.TokenBDOTest INNER JOIN
dbo.BDOVariantsUniverseRoll ON dbo.TokenBDOTest.Order_General = dbo.BDOVariantsUniverseRoll.Order_General LEFT OUTER JOIN
dbo.LanguageOrder ON dbo.TokenBDOTest.LngOrderNewDefn = dbo.LanguageOrder.LngOrderNewDefn
ORDER BY dbo.TokenBDOTest.TokenID, dbo.TokenBDOTest.LnNoBeg, dbo.TokenBDOTest.ColBeg

10.6 Merger Tables

10.6.1 User Input Format

This is table used by the merger to get update records from. The translator/preprocessor writes all user input record to this one table for processing by the merger. Different token types can have different merge keys. Certain columns will be populated or unpopulated based on the token type. This table was mentioned in the paper. However all of the columns are not, nor do the column heading give a clear clue as to what or how it's used for. Since this is an important table in the design of the merger application I'm listing

tersely what each column or theme of columns is supposed to do instead of showing an example of table records.

Column Name Description
TokenType - as described by the dictionary
TokenTypeParser- the token type as described by the parser (This is a place holder for merging directly in to the token table if a BDO expansion isn't needed.)

Token - the revise/update data for the token
TokenUpdate - delete or update the token

BlockName - The defined block name
BlockNameOriginal – one application we use will append "_2" to a duplicate block name. This is a space holder to preserve this variant of the name.
BlockNum - When a block name appears more than once, you need to put in the actual block number to allow for a unique merge.
OvrUseBlkNum- When needing to use the BlockNum to process an update, you need to let the merger know, by putting data in this column.

FieldNameOriginal (same theme as duplicate blocks)
FieldName
FieldNum

TypeNameOriginal (same theme as duplicate blocks)
TypeName
TypeNum
OvrUseTypNum

FieldType
CodeValue

LangID
OvrUseAllLanguages – you want to use the same token to update all languages in the same manner for a token type

TokenProcessed - feedback from the merger saying if the token was found, or updated
TokenAcceptRejectCode
TokenAcceptRejectText

FileRefNum (these are planned to be used when you have the same include file used more than once)
OvrUseFileNum

Cheapkey – a cheap key added to uniquely identify a user update request record
BulkID – a duplicate of cheap key
BulkSource – since update records can come from several difference sources, the pre-processor will put something indicating the group or file the record came from.

10.6.2 Merged BDO

These are some records from the merge BDO table used by the merger and writer.

Note: Empty line information does not appear in this table. The writing routine compares line numbers between records and puts in the blank lines as needed.

Token ID	TokenType	Token	BlockNum	BlockName	FieldNum	FieldName	TypeNum	LangID	CodeValue	LineNumBeg	ColumnNumBeg	LineNumEnd	ColumnNumEnd	FileRefNum	FileName	TokenUpdate	BlockPath	FileLineCountBeg	FileLineCountEnd	BlkNoEnd	TypeNoEnd	TypeName	FIName	FileUseCount	FileLineNumMax	chepkey
1246	FieldName	Q1	1	SE_illustration	22	Q1	41			170	2	170	3	1	SE_Illustration.bla		1..	58	58		0				99	1246
1247	WhiteSpaceBlank		1	SE_illustration	22	Q1	41			170	4	170	4	1	SE_Illustration.bla		1..	58	58		0				99	1247
1248	Tag	(Q1.1)	1	SE_illustration	22	Q1	42			170	5	170	8	1	SE_Illustration.bla	Update	1..	58	58		0				99	1248
1249	WhiteSpaceBlank		1	SE_illustration	22	Q1	42			170	9	170	9	1	SE_Illustration.bla		1..	58	58		0				99	1249
1250	FieldTextLangID		1	SE_illustration	22	Q1	42	ENG			9			1	SE_Illustration.bla		1..				0				99	1250
1251	FieldText	"Are you ready to answer questions?"	1	SE_illustration	22	Q1	42	ENG		170	10	170	45	1	SE_Illustration.bla		1..	58	58		0				99	1251
1252	FieldTextLangID		1	SE_illustration	22	Q1	42	SPN			10			1	SE_Illustration.bla		1..				0				99	1252

MergedBDO																										
TokenID	TokenType	Token	BlockNum	BlockName	FieldNum	FieldName	TypeNum	LangID	CodeValue	LineNumBeg	ColumnNumBeg	LineNumEnd	ColumnNumEnd	FileRefNum	FileName	TokenUpdate	BlockPath	FileLineCountBeg	FileLineCountEnd	BlkNoEnd	TypeNoEnd	TypeName	FlName	FileUseCount	FileLineNumMax	cheapkey
1258	Separator/		1	SE_illustration	22	Q1	42				10			1	SE_Illustration.bla		1..				0				99	1258
1259	DescTextLangID		1	SE_illustration	22	Q1	42	ALTSPN			10			1	SE_Illustration.bla		1..				0				99	1259
1260	FieldDescriptor		1	SE_illustration	22	Q1	42	ALTSPN			10			1	SE_Illustration.bla		1..				0				99	1260
1261	Separator:	:	1	SE_illustration	22	Q1	42			170	46	170	46	1	SE_Illustration.bla		1..	58	58		0				99	1261
1262	Separator(_Enum	(	1	SE_illustration	22	Q1	42			170	47	170	47	1	SE_Illustration.bla		1..	58	58		0				99	1262
1263	CodeName	y	1	SE_illustration	22	Q1	42		y	170	48	170	48	1	SE_Illustration.bla		1..	58	58		0				99	1263
1264	CodeTextLangID		1	SE_illustration	22	Q1	42	ENG	y		48			1	SE_Illustration.bla		1..				0				99	1264

MergedBDO																										
TokenID	TokenType	Token	BlockNum	BlockName	FieldNum	FieldName	TypeNum	LangID	CodeValue	LineNumBeg	ColumnNumBeg	LineNumEnd	ColumnNumEnd	FileRefNum	FileName	TokenUpdate	BlockPath	FileLineCountBeg	FileLineCountEnd	BlkNoEnd	TypeNoEnd	TypeName	FlName	FileUseCount	FileLineNumMax	cheapkey
47	TYPE	type	1	SE_illustration	0	SE_illustration	2			13	1	13	4	2	type_Definitions.INC		1..	2	2		0				34	47
48	WhiteSpaceBlank		1	SE_illustration	0	SE_illustration	2			14	1	14	8	2	type_Definitions.INC		1..	3	3		0				34	48
49	Comment	{V1..}	1	SE_illustration	0	SE_illustration	2			14	9	14	87	2	type_Definitions.INC		1..	3	3		0				34	49
50	WhiteSpaceTab		1	SE_illustration	0	SE_illustration	2			15	1	15	1	2	type_Definitions.INC		1..	4	4		0				34	50
51	TYPEName	TYesNoV1	1	SE_illustration	0	SE_illustration	3			15	2	15	9	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	51
52	WhiteSpaceBlank		1	SE_illustration	0	SE_illustration	3			15	10	15	10	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	52
53	Separator=	=	1	SE_illustration	0	SE_illustration	3			15	11	15	11	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	53
54	Separator(_Enum	(	1	SE_illustration	0	SE_illustration	3			15	12	15	12	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	54
55	WhiteSpaceBlank		1	SE_illustration	0	SE_illustration	3			15	13	15	13	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	55
56	CodeName	Y	1	SE_illustration	0	SE_illustration	3		Y	15	14	15	14	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	56
57	WhiteSpaceBlank		1	SE_illustration	0	SE_illustration	3		Y	15	15	15	15	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	57

MergedBDO																											
TokenID	TokenType	Token	BlockNum	BlockName	FieldNum	FieldName	TypeNum	LangID	CodeValue	LineNumBeg	ColumnNumBeg	LineNumEnd	ColumnNumEnd	FileRefNum	FileName	TokenUpdate	BlockPath	FileLineCountBeg	FileLineCountEnd	BlkNoEnd	TypeNoEnd	TypeName	FlName	FileUseCount	FileLineNumMax	checkpkey
58	Separator(_codenum	(	1	SE_illustration	0	SE_illustration	3		Y	15	16	15	16	2	type_Definitions.INC	Delete	1..	4	4	1	3	TYesNoV1			34	58
59	CodeNumber	2	1	SE_illustration	0	SE_illustration	3		Y	15	17	15	17	2	type_Definitions.INC	Delete	1..	4	4	1	3	TYesNoV1			34	59
60	Separator)_codenum	)	1	SE_illustration	0	SE_illustration	3		Y	15	18	15	18	2	type_Definitions.INC	Delete	1..	4	4	1	3	TYesNoV1			34	60
61	WhiteSpaceBlank		1	SE_illustration	0	SE_illustration	3		Y	15	19	15	19	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	61
62	CodeTextLangID		1	SE_illustration	0	SE_illustration	3	ENG	Y		19			2	type_Definitions.INC		1..			1	3	TYesNoV1			34	62
63	CodeText	"yes"	1	SE_illustration	0	SE_illustration	3	ENG	Y	15	20	15	24	2	type_Definitions.INC		1..	4	4	1	3	TYesNoV1			34	63

10.7 Merger Log Information

10.7.1 The Check Option Report

```
SE Merger Application
---------------------
    Run 07/23/2007 6:39:45 PM

Check User Input Versus Blaise DataModel Structure
--------------------------------------------------

Language Defined and Order
--------------------------
    Run 07/23/2007 6:39:45 PM
      1.CORENG (old order 1)
      2.CORSPN (old order 2)
      3.PRXENG (old order 3)
      4.PRXSPN (old order 4)
      5.EXTENG (old order 5)
      6.EXTSPN (old order 6)
      7.MEDIA (old order 7)

BDO language Expansion Anchors
----------------------------------------------
    Run 07/23/2007 6:39:45 PM

>>>>>  For Field Text LIDs, there are no Merged BDO anchors.
ok --  For Field Text, all of the possible 7 Merged BDO anchors.

>>>>>  For Field Descriptor Text LIDs, there are no Merged BDO anchors.
>>>>>  For Field Descriptor Text, there are only 1 of the possible 7 Merged BDO anchors.

>>>>>  For CodeFrame Text LIDs, there are no Merged BDO anchors.
ok --  For CodeFrame Text, all of the possible 7 Merged BDO anchors.

Duplicate Blocks and Include file usage
-----------------------------------------------------------
    Run 07/23/2007 6:39:45 PM

Duplicate Blocks Information
----------------------------
>>>>> The following are 'duplicate' block name.
>>>>> The user input file will need to use Block numbers to uniquely distingusih these blocks.

     The Block Name BA_WThree appears 2 times.
     The Block Name BF_DECISIONS appears 2 times.

Duplicate (re-used)Include File Information
-------------------------------------------
>>>>> The following are 'duplicate' include file name.
>>>>> The user input updates to these file reference numbers - may conflict with each other or not be
updated.

     The File Name HRS06_W1.INC appears 2 times.
```

10.7.2 The Merger Log

```
SE Merger Application
---------------------
    Run 07/23/2007 6:44:07 PM
Checking Source Editor Database Connection
------------------------------------------
    07/23/2007 6:44:07 PM
    Server Connection Okay 07/23/2007 6:44:07 PM

Statistic of user input table
-----------------------------
    07/23/2007 6:44:07 PM
    Getting User Input 07/23/2007 6:44:07 PM
    There are N=6498 User Input TokenType update request records.

331E-Code Frame Enumeration Language Dependent Structure  Keys: TokenType, Block Name, Type
Name, CodeValue, and LangID.
>>>>> (51)>>>>> The user input record was not in the DataModel BDO being updated.
       TokenType: CodeText
       BlockName: B_EMPLOYERINFO
       FieldName: W168_VerifyEmpName
       TypeName : TW168_VerifyEmpName
       CodeValue: DENIESWRKPW
       LangID   : prxspn

331F-Code Frame Field Language Dependent Structure  Keys: TokenType, Block Name, Field Name,
CodeValue, and LangID.
>>>>> (51)>>>>> The user input record was not in the DataModel BDO being updated.
       TokenType: CodeText
       BlockName: BM_SSDI_APPACCEPTED
       FieldName: W256A
       TypeName :
       CodeValue: RETURNEDWORK
       LangID   : prxspn

141/161-Field Language Dependent Structure  Keys: TokenType, Block Name, Field Name, and LangID.
>>>>> (51)>>>>> The user input record was not in the DataModel BDO being updated.
       TokenType: FieldText
       BlockName:
       FieldName:
       TypeName :
       CodeValue:
       LangID   : PRXSPN

141/161-Field Language Dependent Structure  Keys: TokenType, Block Name, Field Name, and LangID.
>>>>> (51)>>>>> The user input record was not in the DataModel BDO being updated.
       TokenType: FieldText
       BlockName: B_EMPLOYERINFO
       FieldName: W158_CurrEmpName
       TypeName :
       CodeValue:
       LangID   : PRXSPN

... now processing TokenTypes of CodeText. with record 2967 of a total of 6498.
    45.6602031394275% processed. 07/23/2007 6:44:58 PM
```

10.8 A Sample Output file

The output has several changes due to relative positing and the language re-order done for this run.

10.8.1 FileName: PetFields.inc

```
{field for the pet blocks}
type TYesNoV2 =         ( Y (11) {SE Added text} "" {SE Added text} "" {SE Added text} "" "yes",
                    N  {SE Added text} "" {SE Added text} "" {SE Added text} "" "no",
                    s {SE Added text} "" {SE Added text} "" {SE Added text} "" "Several"), dk

fields TP1
    {SE Added text}""

{SE Added text}""

 {SE Added text}""
eng "For this type of pet, ^pettype, @/
@/Do you have any? " alteng"For this type of pet, ^pettype, @/
@/Do you have any? ":TYesNoV2
TP2 "How many ^pettype do you have?": 0..100
tp3 (Pet3)
    {SE Added text}""

{SE Added text}""

 {SE Added text}""
"Is there anything special you want to mention about ^pettype?"
: (Yes,Maybe, No)  , dk, rf
TP5 (PT4)
     {SE Added text}""

 {SE Added text}""

  {SE Added text}""
"If so what you like to say?": open
```