SMASER

REBIF

TRIPsystem
Product Documentation

End User License Agreement

All rights to this software, its documentation and logotypes of the TRIP product family and software (altogether “Software”) supplied by Smaser AG (Smaser) are exclusively owned by Smaser.

The transfer of this Software, solutions or parts thereof requires the prior written agreement of Smaser. Furthermore, the customer has the right to use licensed Software and / or process solutions supplied by Smaser to the extent specified in his contract with Smaser.

The free-to-use non-commercial version doesn’t require a prior written agreement with Smaser but such customers, organizations and/or third parties agree by using the software and / or solution of Smaser to be strongly obliged to keep all rights to this software, documentation and logotypes of the TRIP product family absolutely uninfringed and protected.

About this document

The following document describes the prompts that you will get from Rebif, and how to respond to them to get differing kinds of output. The purpose of the exercise is to help a customer support representative to be more effective in the optimization of TRIP system performance, particularly in large customer sites, or when a database is to be written to non-modifiable media (such as CD-ROM).

Conventions used in this document

`Xyzzy (x)`	is a prompt from REBIF to the user (where 'x' is the default response provided by REBIF and taken if the user simply hits the RETURN key)

What is REBIF?

REBIF is a utility aimed at the tuning of large databases where search performance can be increased dramatically by sensibly tuning the layout of the primary tables within the index files. It allows the specification of certain parameters particular to this layout, *which last only for the duration of the REBIF run*, i.e. further updates to the database will use the system default values rather than those chosen by the user during the REBIF run.

In order to make sensible use of the REBIF utility you must fully understand how to use the EXIF utility described in document TRIP-EXIF. You should also have a good understanding of how the index files are constructed, as described in document TRIP-IF.

The REBIF utility program is located in the directory with the same path that is defined as TDBS_EXE in the TRIPsystem configuration file tdbs.conf. Run REBIF from the command line:

```
<TDBS_EXE>/rebif
```

Prompts from REBIF

REBIF will prompt the user for the information it needs. The values prompted for can also be specified as options to the REBIF program on the command line. See section "Command line options" below for more information.

Old BIF or VIF file name :
Specify here the actual filename of an existing index file (of either type). You cannot specify a database name here.

New BIF or VIF file name :
Specify here the actual filename of the index file to be created. This file cannot already exist as REBIF will not overwrite existing files. This therefore implies that probably at least twice the amount of storage required for the old index file is required to run REBIF.

Entry width (current entry width) :
This prompt allows you to specify the size of the primary entry table within the index file itself. The default for the prompt is the width of the existing table, which you can adjust upwards or downwards. Note that setting bit 3 of the database flags (no automatic size adjustments during index) has no effect here, and if you specify an entry width which is insufficiently large, you will incur a reorganization of the index which will, of course, result in a less well performing index file.

Note that although REBIF will allow you to specify a value between 1 and 24 for the entry width of a BIF.

Entry overflow percentage (30) :
When the system is deciding when to reorganize an index file, the primary motivating factor is the overflow within the entry table itself. Each block within the entry table may link to other blocks within the secondary entry storage area, which may in turn link to further blocks within the secondary entry storage area. The entry overflow percentage governs the threshold at which an automatic reorganization is performed. The default, 30%, means that the file will reorganize when the size of the secondary entry storage area is greater than 30% of the size of the entry table itself.

Obviously, when building an index for a slow media, such as CD-ROM, the best performance will be gained when the secondary entry storage area does not exist, i.e. 0% entry overflow.

If you are concerned about the size of an index file on fast media, such as magnetic disk, you can decrease the entry width of the file and increase the entry overflow percentage in an attempt to keep the size of the primary entry table down. This is not always guaranteed to work and should only be performed in conjunction with use of the EXIF utility.

Data sub-item size limit (50) :

When a term is entered to the index, the term itself as well as the postings of that term has to be recorded within the entry block chosen by TRIP for that term.

If the entry block in question contains insufficient space for the term or posting to be added, TRIP will search the entry block for existing terms to have their posting information copied to the term overflow area of the index file thus freeing up space within the entry block itself. Postings copied in this manner are pointed to from the entry block by term overflow pointers (see next section).

The *data sub-item size limit* governs the threshold used by TRIP when searching for candidate terms in this fashion. Terms which have fewer than this number of postings physically stored within the entry block are left intact, whilst terms with greater than this number of postings have these postings copied to the term overflow area of the index. As mentioned before, postings thus copied are replaced in the entry block with a single term overflow pointer which tells TRIP where in the term overflow area to find the actual postings.

You can specify values between 20 and 100 for this parameter. Generally, you should never need to modify the default case and if you do modify it significantly, time must be spent modeling the performance both before and after the modification to ensure that said modification has indeed had a beneficial effect.

Pointer sub-item size limit (100) :

If copying actual postings to the term overflow area of the index does not free sufficient space within the entry block to accommodate the new term or posting, TRIP will then examine term overflow pointers within the entry block.

Normally, terms within the index occur many times within the data. Each time a new occurrence is indexed a new posting is added to the entry block until the postings need to be copied to the term overflow area. This copying will occur many times during the life of the database and each time TRIP does this it adds a term overflow pointer to that term's posting list within the entry block.

The *pointer sub-item size limit* governs the threshold used by TRIP when searching for candidate terms to have their term overflow pointers copied to the term overflow area. In such cases, a single pointer is then used to reference the list of pointers copied from the entry to the term overflow area. Obviously, the pointers thus copied still point to whatever they pointed to before being copied (which could be actual postings or further pointer lists).

If sufficient space is still not forthcoming within the required entry block, TRIP will create an entry overflow block within the entry overflow area of the index and "double" the existing block so that the terms from the original entry block are now split across two blocks.

Of course, if in the act of creating a new entry overflow block, the index file crosses the entry overflow threshold set above, TRIP will force an index file reorganization to take place.

You can specify values between 50 and 200 for this parameter. Generally, you should never need to modify the default case and if you do modify it significantly, time must be spent modeling the performance both before and after the modification to ensure that said modification has indeed had a beneficial effect.

Size guideline for new segments (50) :

Within the postings for any particular term in the index, TRIP creates what are called segments. These are used for optimizing Boolean searches so that groups of postings which are not relevant to a search are not read from the index.

In order to make this system work, TRIP requires a guideline for the number of postings to be stored together within a single segment. The system default is 50, i.e. 50 postings will normally be stored together within a single segment (this can be seen by running EXIF against the ALICE.BIF as shown in example 2 of the document TRIP-EXIF). This default value will be exceeded, or lessened, in certain cases when the addition, or removal, of a particular posting will help the compression of a particular segment.

Given sufficient modeling, this parameter will yield to modification. Tests within R&D have shown that in particularly large controlled vocabulary databases, such as newspapers, segments of up to 75 or 80 can produce significant performance gains.

This parameter should never be set to less than 50 and cannot be set to more than 100. In most cases, settings of greater than 66 will be detrimental to the CPU usage of TRIP but as stated previously this does not always mean a slower search.

BUT file to be produced (N) :

This prompt will only appear if the file being rebuilt is a BIF. This can be useful when, for instance, the VIF has become corrupt or an index job terminated abnormally when updating the VIF and has left it unusable. There is no other mechanism to produce a BUT file for input to the VIF scan phase of SCIFFEX (as documented in TRIP-INDEX) which does not involve scanning the BAF and can therefore save significant time in recovery of such a VIF.

Output from REBIF

When you run REBIF, you will see a summary output produced showing the layout of free space within the index file. You should ignore this output as it is only relevant to the *old* index file and not to the new.

In order to actually examine the layout of the new index file, you must use the EXIF utility described in the document TRIP-EXIF.

“Real world” examples of using REBIF

Let us suppose that you are preparing a database for mastering to a CD-ROM. In this case, you want as few I/O operations to occur during an index as possible. This ensures that the read head does as few seeks as possible, and therefore speeds up the search greatly (a seek on a CD-ROM can take as long as 0.3 seconds in extreme cases).

Firstly, you would run the EXIF utility to examine the current state of the index file before determining what strategy to adopt with REBIF.

```
*** TRIP System Utility EXIF - BIF/VIF examination ***
     Version 6.0-0   10-DEC-2008 11:52:17.53

BIF or VIF file name                  : /trip/v600/demo/alice.bif

Time-stamp limit?                (N) : n
Normal run?                      (Y) : y
Storage statistics?              (Y) : y
Summary statistics?              (Y) : y
Minimal summary?                 (Y) : y
One entry block?                 (Y) : n
First entry block number?        (2) : 2
Last entry block number?        (65) : 65
```

```
EB: 2-65         Type    Ocnt  Rcnt/F    Sspace  Pcnt     L0     L1     L2   L>2
--------------    -- ------- ------- --------- ----- ------ ------ ------ -----

   3524 terms      2   62665            188942   259   3294   1070      -     -
      1 terms     10     475               337     1      -     10      -     -
    125 terms     13    1844              2879     1    132      2      -     -

-------------------------------------------------------------------------------
   3584 terms          64984   10279    192158   261   3426   1082      -     -
-------------------------------------------------------------------------------

Entry overflow level  0:    64


                          Entry      Term
  Free Class     Entry  Overflow  Overflow     Total
  ----------  --------  --------  --------  --------
           0        13         0        19        32
           1        20         0        14        34
           2        13         0        10        23
           3         6         0         6        12
           4         5         0         3         8
           5         3         0         2         5
           6         2         0         4         6
           7         0         0         5         5
           8         1         0         0         1
          11         1         0         0         1
          16         0         0         2         2
          18         0         0         1         1
          24         0         0         1         1
          25         0         0         3         3
          26         0         0         1         1
          31         0         0         1         1
------------  --------  --------  --------  --------
Unused bytes     10279         0     23792     34071
Unused part     0.0790    0.0000    0.1623    0.1221


Elapsed: 00:00:15
```

This shows us that although the primary entry table is flat (the reported “entry overflow level 0” is equal to the size of the entry table itself - 64 blocks or 2^6), the term overflow is well used - 1082 terms have their postings recorded in the term overflow area. This is almost 30% of the terms in the database and will cause performance degradation on slow media.

To change this layout for better performance we shall attempt to flatten the term overflow area thus getting more terms stored in level 0 (L0 in the above table) and removing as many terms as possible from level 1 (L1 - term overflow).

```
**** TRIP System Utility REBIF - Rebuild BIF/VIF file ****
     Version 6.0-0    10-DEC-2008 12:25:33.94

Old BIF or VIF file name              : trip$demo:alice.bif
New BIF or VIF file name              : []alice.bif
Entry width                     (6) : 7
Entry overflow percentage      (30) : 30
Data sub-item size limit       (50) : 50
Pointer sub-item size limit   (100) : 100
Size guideline for new segments (50) : 60
BUT file to be produced?         (N) : N
```

As stated previously, the output from REBIF can be ignored, and so is not reproduced here.

From here you can see that we have increased the width of the entry table from 6 to 7 and increased the segment size guideline from 50 to 60. The increase in segment size is actually detrimental if the entry table is not also increased, but a smaller performance gain can be seen without increasing the segment size. Note that such increases should never be performed without judicious performance modeling before and after changes.

In order to examine the new layout of the file, we again use the EXIF utility.

```
*** TRIP System Utility EXIF - BIF/VIF examination ***
     Version 6.0-0    10-DEC-2008 12:25:54.09

BIF or VIF file name                  : alice.bif

Time-stamp limit?                (N) : n
Normal run?                      (Y) : y
Storage statistics?              (Y) : y
Summary statistics?              (Y) : y
Minimal summary?                 (Y) : y
One entry block?                 (Y) : n
First entry block number?        (2) : 2
Last entry block number?       (129) : 129


EB: 2-129       Type   Ocnt  Rcnt/F    Sspace  Pcnt    L0    L1    L2   L>2
--------------   -- ------- ------- --------- ----- ----- ----- ----- -----

   3524 terms     2   62665            189183    60  3707   487     -     -
      1 terms    10     475               319     0     8     -     -     -
    125 terms    13    1844              2894     0   129     -     -     -

-------------------------------------------------------------------------------
   3584 terms         64984   89185    192396    60  3844   487     -     -
-------------------------------------------------------------------------------

Entry overflow level  0:   128


                         Entry      Term
  Free Class     Entry  Overflow  Overflow     Total
  ----------  --------  --------  --------  --------
           0         7         0         4        11
           1         5         0        12        17
           2         8         0         5        13
           3         4         0         2         6
           4         4         0         5         9
           5         3         0         2         5
           6         5         0         1         6
           7         5         0         0         5
           8         8         0         2        10
           9         9         0         1        10
          10         8         0         0         8
          11         5         0         0         5
          12         4         0         0         4
          13         8         0         2        10
          14         3         0         1         4
          15         8         0         0         8
          16         9         0         0         9
          17         8         0         0         8
          18         6         0         1         7
          19         4         0         1         5
          20         2         0         0         2
          21         3         0         0         3
          22         1         0         0         1
          23         1         0         0         1
          24         0         0         1         1
          31         0         0         1         1
------------  --------  --------  --------  --------
Unused bytes     89185         0     15421    104606
Unused part     0.3429    0.0000    0.1847    0.3022


Elapsed: 00:00:13
```

What we can see here is that the entry table is now twice the size (as expected), all INteger and PHrase values are now stored at level 0 and the number of word values stored at level 1 has decreased from 1070 to 487 so that the percentage of level 1 terms is reduced to 13.5%. Of course, the physical size of the file is increased in this case from 274K to 340K although this increase is not always incurred, particularly in very large files.

Note that further increasing the entry width to attempt to reduce the number of level 1 terms to 0 is not guaranteed to succeed and will probably result in an undesirably large index file.

As an example of how not to tune an index, we turn to the problem of attempting to reduce the size of the entry table for smaller files on fast media. Again using ALICE :

```
**** TRIP System Utility REBIF - Rebuild BIF/VIF file ****
     Version 6.0-0    10-DEC-2008 12:52:56.20

Old BIF or VIF file name              : trip$demo:alice.bif
New BIF or VIF file name              : []alice.bif
Entry width                      (6) : 5
Entry overflow percentage       (30) : 100
Data sub-item size limit        (50) :
Pointer sub-item size limit    (100) :
Size guideline for new segments  (50) :
BUT file to be produced?          (N) : n

---- Reorganizing the inverted file ----
---- New entry block width: 6 ----
```

As can be seen here, the reduction of the entry width did not actually work as REBIF forced an increase back to the original size. As can be seen from below, we have also badly affected the layout of the index, increasing the level 1 terms from 1070 to 1265 :

```
*** TRIP System Utility EXIF - BIF/VIF examination ***
     Version 6.0-0    10-DEC-2008 12:53:21.45

BIF or VIF file name                  : alice.bif

Time-stamp limit?                (N) :
Normal run?                      (Y) :
Storage statistics?              (Y) :
Summary statistics?              (Y) :
Minimal summary?                 (Y) :
One entry block?                 (Y) : n
First entry block number?        (2) :
Last entry block number?        (65) :


EB: 2-65        Type   Ocnt  Rcnt/F    Sspace  Pcnt    L0    L1    L2   L>2
--------------   -- ------- ------- --------- ----- ----- ----- ----- -----

   3524 terms     2   62665            189072   414  3130  1236     -     -
      1 terms    10     475               333     1     -    10     -     -
    125 terms    13    1844              2879    11   115    19     -     -

-------------------------------------------------------------------------------
   3584 terms         64984   24425    192284   426  3245  1265     -     -
-------------------------------------------------------------------------------

Entry overflow level  1:     6
Entry overflow level  0:    58


                         Entry      Term
  Free Class     Entry  Overflow  Overflow     Total
  ----------  --------  --------  --------  --------
           0         5         0        33        38
           1        10         0        16        26
           2         2         0         3         5
           3         6         0         2         8
           4         4         0         1         5
           5         4         0         7        11
           6         5         0         4         9
           7         9         0         2        11
           8         1         0         0         1
           9         8         0         0         8
          10         4         0         0         4
          11         4         0         1         5
```

```
            12          1          0          0          1
            14          1          0          0          1
            15          0          0          1          1
            16          0          0          2          2
            19          0          0          1          1
            20          0          0          1          1
            21          0          0          1          1
            22          0          0          1          1
            24          0          1          0          1
            25          0          1          4          5
            29          0          1          0          1
            30          0          2          0          2
            31          0          1          3          4
------------  ---------  ---------  ---------  ---------
Unused bytes      24425      11002      30039      65466
Unused part      0.1878     0.9024     0.1778     0.2088


Elapsed: 00:00:13
```

Following this “tuning”, we have an index file which now uses an entry flow area (for six of the primary entry blocks) and has a larger term overflow usage than that with which we started, both of which factors will produce performance degradation.

Command line options

REBIF takes the following options on the command line (see under "Prompts from REBIF" above for more detailed explanation of each option):

-f <filename>	Old BIFor VIF file name
-o <newfile>	New name for BIF or VIF file
-B <filename>	BUT file name
-w <entrywidth>	Entry width (prefix with +/- to modify)
-e <value>	Entry overflow limit percentage (0-500)
-d <value>	Data sub-item size limit (20-100)
-p <value>	Pointer sub-item size limit (50-200)
-g <value>	Size guideline for new segments (10-75)
--[no-]but	Use BUT file? (implicit if -B is specified)

Note that option names are case sensitive. Options specified on the command line will not be prompted for unless the associated specified value is invalid.

In addition, REBIF takes the following option that assigns reasonable defaults to options whenever possible:

--defaults	Use defaults for options when possible