How To Run Index Jobs

Release 14.2

Last update: September 17, 2001
1. INTRODUCTION ............................................................................................................ 4

2. WHICH JOBS TO RUN WHEN: SEQUENCES AND DEPENDENCIES... 5
   2.1 p_manage_05, _07, _27, _12 ........................................................................... 5
   2.2 p_manage_01 ................................................................................................. 5
   2.3 p_manage_02 and Associated Jobs ..................................................................... 5
       • Main Headings jobs ........................................................................................ 5
       • Other Headings jobs ..................................................................................... 6
   2.4 Typical job sequences for each library ........................................................... 7
       • Typical job sequence for BIB library: .......................................................... 7
       • Typical job sequence for AUThority library: .............................................. 8
       • Typical job sequence for ADM library: ....................................................... 8
       • Typical job sequence for HOL library: ........................................................ 8
       • Typical job sequence for Course Reading library: ....................................... 8

3. TURNING OFF ARCHIVE LOGGING ........................................................................ 9

4. PROCESSES AND CYCLE SIZE ........................................................................ 10
   • How Many Processes? ..................................................................................... 10
   • What Cycle Size? ............................................................................................ 10

5. DISK SPACE AND FILE LOCATIONS .................................................................... 12
   Moving $TMPDIR, $data_scratch, and $data_files ........................................... 14

6. PREPARATION FOR INDEX JOBS ........................................................................ 15
   Clean temp/scratch directories ......................................................................... 15
   Check Oracle space ............................................................................................ 15
   Cancel jobs which might interfere ...................................................................... 15

7. UNLOCKING THE LIBRARY WHILE THE JOB IS RUNNING ............................ 16

8. MONITORING THE JOBS ....................................................................................... 17
   p_manage_01 ....................................................................................................... 17
   p_manage_02 ....................................................................................................... 17
   p_manage_05 ....................................................................................................... 18
   p_manage_07 ....................................................................................................... 18
   p_manage_08 ....................................................................................................... 19
   p_manage_12 ....................................................................................................... 19
   p_manage_17 ....................................................................................................... 20
9. TROUBLESHOOTING ........................................................................................................... 22
   • Job is Stuck ...................................................................................................................... 22
   • Restart ............................................................................................................................. 23
   • Job is Slow ...................................................................................................................... 23
   • Specific Error Messages ................................................................................................ 24
   • Problems using/searching indexes ................................................................................ 24

10. ESTIMATING RUN TIME .................................................................................................... 25

11. APPENDIX A: SAMPLE COMMANDS FOR RUNNING JOBS FROM COMMAND LINE .......................................................... 28

12. APPENDIX B. HOW MANY PROCESSORS DO YOU HAVE? ........ 30

13. APPENDIX C. DIFFERENCE BETWEEN Z95 AND Z950 ................. 31
1. Introduction

This document describes how to run index jobs such as Words and Headings. It touches upon such issues as turning off archive logging, number of processes, disk space and file locations, unlocking the library while a job is running, monitoring the jobs, troubleshooting, and estimation of the run time.

For easy reference, here is a list of the batch utilities and daemons mentioned in this document:

- p_manage_01 = Generate Words
- p_manage_02 = Generate Headings
- p_manage_05 = Generate Direct Index
- p_manage_07 = Create Short Doc records
- p_manage_08 = Organize Table (under “Manage Database Tables”)
- p_manage_12 = Create Links between records (under “Manage Database Tables”)
- p_manage_15 = Delete unused headings (*14.2 and higher only*)
- p_manage_16 = Re-sort Headings
- p_manage_17 = Alphabetize Long Headings
- p_manage_27 = Create Sort Index records
- p_manage_32 = Build Counters for Logical Bases (for Headings) (*14.2 and higher only*)
- p_manage_102 = Copy Headings from AUT library to BIB library (*14.2 and higher only*)
- p_manage_103 = Create Z07s for headings which are candidates for correction (*14.2+ only*)

- UE_01 = indexing daemon
- UE_08 = cross-referencing daemon

Generally, the jobs are submitted through the Web Staff Menu, Web Services.

Jobs can also be submitted by entering a csh command at the unix prompt. See Appendix A for sample scripts to use for running the job from the command line. (Currently the p_manage_102 and p_manage_01_z95_z950 are not available through Web Services and can be run *only* from the command line.)
2. Which Jobs to Run When: Sequences and Dependencies

In general, you run index jobs because

a) You have batch-loaded records (without selecting the “full indexing” option),
b) You have made table changes and want the fields in existing records to be indexed differently, or
c) There is a problem with the index which needs to be corrected.

2.1  p_manage_05, _07, _27, _12

The p_manage_01,  p_manage_05,  p_manage_07, and p_manage_27 utilities are “base” jobs; in other words, they are not dependent on any other jobs. They can be run in any order you wish.

The p_manage_12 utility is also independent, but the links it creates are required by the other jobs. Generally, you should not need to run p_manage_12. If you feel you do, please consult Ex Libris Customer Support.

2.2  p_manage_01

If you have “setenv ADJACENCY Y” specified in ../alephe/aleph_start_505, then p_manage_01 will build word pairs in the z97 dictionary. This greatly speeds up Adjacency searching. But it does require additional work space. (See the “Disk Space and File Locations” section below.) And it increases the size of the z9n Oracle Word tables: for a 1.7 million record file, the z97 dictionary is about four times larger; and the (much larger) z950 and z98 files are about 50% greater. (Note: in a small file, say 30,000 records, the z950 and z98 will be increased by as much as three times.)

p_manage_01 has an optional step, p_manage_01_z95_z950, which can be run after the regular p_manage_01 run. See Appendix C, “Difference Between z95 and z950” for more information.

2.3  p_manage_02 and Associated Jobs

Note: The Headings are basically usable immediately after the p_manage_02 is run. If the availability of the Headings is a consideration, consider running other steps with the online up or postponing until the next evening.

- Main Headings jobs

If you have an authority library and this is a complete run (indexing all the records), you should run p_manage_102 before running p_manage_02.

- p_manage_102 copies the headings from authority records into the BIB file Z01. This makes the long, complete run of UE_08, which matches headings with authority records, unnecessary. Only the run of manage_102 for the first authority library should specify “1” (“delete existing headings”). Runs for the
second, third, etc., authority library, and the run of manage_02, should specify "0" ("Update") (so the previous headings you've copied into the Z01 aren't deleted).

- **p_manage_02**

If p_manage_02 is preceded by p_manage_102, then you should run p_manage_02 with:
  - "Procedure to run" = Update ("0")
  - "setenv sw_force_chk Y" specified in aleph_start_505.
    (This is described in 14.2 rep_change 2642.)

If p_manage_02 is not preceded by p_manage_102, then you should run p_manage_02 with:
  - "Procedure to run" = Rebuild entire ("1")
  - "setenv sw_force_chk N" in aleph_start_505 (note: this is the default)

- **p_manage_08** (specifying “Headings Index, Table Z01” as the table you are organizing). (This job is not absolutely necessary, but it speeds up the p_manage_17 and also helps online performance.)

- **p_manage_17** (There is a p_manage_17_turbo which greatly speeds up the complete run of p_manage_17; it should become the standard version for the complete run soon.)

The jobs should be run in the above order. But other, non-headings, jobs can come in between: If you like, you could run p_manage_02, then p_manage_07, then p_manage_08, and so on. p_manage_17 though, cannot be run before p_manage_08; p_manage_08 cannot be run before p_manage_02….

- **Other Headings jobs**

  - **p_manage_32**: Run if you have small bases specified in tab_base. (Running manage_32 for large bases can result in huge Z0102 Oracle tables. Use util-h-1-10 to check if your tab_base setup is reasonable.)
  - **p_manage_103**: Create Z07s for headings which are candidates for correction. It would only be run when the bib headings are not already synchronized with the authority headings. That is, when the bib headings have not had authority work done on them and a file of possibly matching authority records has been loaded. And when there are authority records with a UPD value of “Y”. (If all authority records have UPD “N”, then no updates would be made anyway.
  - **p_manage_15**: Deletes headings which don't have any titles (Z02 records) associated with them. We do not normally recommend running this job since it may delete unreferenced headings you want to keep. A new job (p_manage_105) which handles this better will be included in version 15.
  - **p_manage_16**: Re-sorts the existing Z01 headings in accordance with the current tab00.lng and tab_filing. It can be run in two modes: report mode or
update mode. Update mode actually updates the Z01-FILING-TEXT and Z01-FILING-SEQUENCE. The job does *not* need to be run as part of the manage_02 sequence. You would run it only when changes have been made to the filing procedures specified in tab00.eng, to tab_filing, or to the alephe/unicode/unicode_to_filing... character equivalents table, and you don’t want to bother with the complete run of p_manage_102 and p_manage_02.

**UE_08 (if you have authority records)**

You must wait until you have run both the BIB p_manage_02 and the AUTHority p_manage_02 utilities before starting the UE_08. UE_08 updates the BIB Z01 based on what it finds in the authority Z01.

If you have run the p_manage_102 (see above), you **would not**, and **must not**, run UE_08 with the “N” specification —you should **NEVER** need to use the “N” specification--; you would instead specify “C” (checking of new headings only).

### 2.4 Typical job sequences for each library

Note 1: The z103 should be in place for the BIB, ADM, and HOL libraries before any index jobs are run for these libraries. p_manage_12 generates the z103. The z103 is normally built as part of conversion/upgrade and should not normally need to be run. When p_manage_12 is re-run for one library (BIB, ADM, and HOL) it must be run for all of them. (Either the BIB should be run first or the BIB Z103 should be dropped/recreated before you start.) (See the Web Services p_manage_12 Help for additional information.)

Note 2: See Appendix A for sample scripts to use for running the job from the command line. (Currently the p_manage_102, and p_manage_01_z95_z950 are not available through Web Services and can be run *only* from the command line.)

Note 3: * = job is part of sequence of jobs dependent on each other.

- **Typical job sequence for BIB library:**

  p_manage_05  
p_manage_01  
p_manage_07  
p_manage_27  

  *Headings, method #1 (with authorities, large database)*

  p_manage_102* (for *each* authority library)  
p_manage_02* {with "Procedure to run" = Update ("0")}  
p_manage_32* (if you have small bases defined in tab_base)  
p_manage_08* (for Z01)  
p_manage_17* (system can be available, including Browse)
Headings, method #2 (no authorities or smaller database)

p_manage_02* {with "Procedure to run" = Rebuild entire ("1")}

p_manage_32* (if you have large database with small logical bases)

p_manage_08* (for Z01)

p_manage_17* (system can be available, including Browse)

Special jobs: p_manage_01_z95_z950, p_manage_103, p_manage_15, p_manage_16, p_manage_12 (see Note 1, above).

• Typical job sequence for AUTHority library:

  p_manage_05

  p_manage_02* {with "Procedure to run" = Rebuild entire ("1")}

  p_manage_08* (for Z01)

  p_manage_17* (system can be available, including Browse)

  p_manage_01

  p_manage_07

  p_manage_27

Special jobs: manage_16 and manage_12 (if you have set up relationships in the authority library .tab/tab07).

• Typical job sequence for ADM library:

  p_acq_04 (order index)

  p_manage_111 (for users, vendors, budgets, and reading room)

  p_manage_12 (see Note 1, above).

• Typical job sequence for HOL library:

  p_manage_12 (see Note 1, above).

• Typical job sequence for Course Reading library:

  p_manage_12 (should be run first if Z103s aren't already in place)

  p_manage_05

  p_manage_02* {with "Procedure to run" = Rebuild entire ("1")}

  p_manage_17* <may not be necessary, if few long headings>

  p_manage_01

  p_manage_07

  p_manage_27
3. Turning Off Archive Logging

[Note: Sites with relatively small databases (fewer than 300,000 titles) which can be processed in an overnight window need not bother with this section. Leave archive logging as is ]

Oracle archive logging can add 25-50% to the time required for a job to run. But since the running of an index job with archive logging off would normally require a complete, cold Oracle backup both before and after the job, the circumstances in which this is a good strategy are pretty limited. Generally, the only jobs which might even be candidates are the p_manage_01 and p_manage_02.

If you are running several big jobs at the same time, it may be justified.

Certainly during the original data conversion and indexing and during upgrade to a new version, you will want to have archive logging turned off.

For further information, please consult the document, “ALEPH Batch Jobs and Oracle Archive Logging”.
4. Processes and Cycle Size

[Note: Sites with relatively small databases (fewer than 300,000 titles) which can be processed in an overnight window need not bother with this section. Just specify "1" process.]

- **How Many Processes?**

Certain index jobs give you the option of specifying that multiple processes will be generated. This lets the job make use of more of your computer's processing capacity. The p_manage_01, p_manage_02, p_manage_05, p_manage_07, p_manage_12, p_manage_27, p_manage_32, and p_manage_102 jobs let you select a number of processes from 1 to 9. If you specify "5", 5 different processes will be started, each doing different sections of the file.

Each process requires its own sort space. It is possible that there could be sufficient sort space for the job when run with 5 processes, but that it fails when run with 9 processes. See the “Disk Space and File Locations” section immediately following.

Though it might seem strange, you may specify a number of processes greater than your number of processors. (This has to do with how the ALEPH processing is done.) (If you don't know how many processors you have, see Appendix B.)

Though it might seem that you would want to have as many processes as possible, this is not always the case. For jobs (such as p_manage_01 and p_manage_02) where multi-threaded extraction steps are followed by single-threaded steps, having too many (extraction) processes running at the same time will cause the extraction step to "run ahead" -- at the expense of the subsequent single-threaded steps. In general, the number of processes should be twice the number of processors in the machine but not more than 6. For most jobs, 4 processes will keep the extraction step in balance with the following step(s). (One exception is the p_manage_32 where 8 processes usually works best.)

- **What Cycle Size?**

If you have the space (see the next section, "Disk Space...") and want to make the job run faster instead of increasing the number of processes (beyond 4) you should increase the "cycle size". The cycle size is specified in the xxx01/prof_library loop_length parameters: for
  - manage_01, it's the word_loop_length;
  - manage_02, the acc_loop_length;
  - manage_05, the index_loop_length;
  - manage_12, the lkr_loop_length;
  - manage_27, the sort_loop_length;
  - manage_32, the z0102_loop_length;
  - manage_102, the acc_loop_length.
These are our general recommendations:

<table>
<thead>
<tr>
<th>Number of records</th>
<th>Cycle size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 100,000</td>
<td>20000*</td>
</tr>
<tr>
<td>Less than 1 million</td>
<td>50000</td>
</tr>
<tr>
<td>1-4 million</td>
<td>100000</td>
</tr>
<tr>
<td>More than 4 million</td>
<td>200000</td>
</tr>
</tbody>
</table>

(* If the number of records is fewer than 100,000, then a cycle size of $\frac{1}{4}$ the number of records will be optimal. But with relatively few records the job will run fairly quickly and take relatively little space even with a larger cycle size—as long as the cycle size isn’t too *low*: you should *never* have a cycle size like 1000….)

(Note: For p_manage_32 (z0102_loop_length), which is using the z01 as its input file, the numbers you see above should be doubled.)

Specific example: For doc files in the 1-2 million range, having a word_loop_length of 100000 with 4 processes will give you a better result that a word_loop_length of 50000 with 9 processes.
5. Disk Space and File Locations

[Note: Sites with relatively small databases (fewer than 300,000 titles) which can be processed in an overnight window should just make sure that the xxx01 $data_scratch, the xxx01 $data_files, and the $TMPDIR directories are cleaned up before running p_manage_01 or _02 and need not otherwise be concerned with this section.]

Most indexing jobs do not require any great amount of work space, but the p_manage_02 and, especially, the p_manage_01, do.

The sort space ($TMPDIR space) required is a function of the number of processes specified and the “cycle size” which, in turn, depends on the word_loop_length (for manage_01) and acc_loop_length (for manage_02), specified in the xxxnn/prof_library file. (The greater the number of processes and the larger the loop_length, the more $TMPDIR space required – but the faster the job runs – see “Estimating Run Time”, section 10, below.).

The $data_scratch space is a function of the number of document-words/document-headings (how many documents there are and how many fields in each are indexed), and, to a lesser extent, the cycle size and the number of processes. (See preceding section 4.)

The peak use of $data_scratch space by indexing jobs normally occurs in the p_manage_01_e step of the p_manage_01 job. With typical MARC bibliographic records (averaging about 1K per record), a typical number of Word indexes (20-40), an appropriate word_loop_length (see “What cycle size?” in preceding section), and “setenv ADJACENCY Y”**, the $data_scratch will vary from 10K per doc record to 15K per doc record.

{** Running p_manage_01 with “setenv ADJACENCY Y” (see Section 2.2 above) increases the $data_scratch, $data_files, and (assuming no change in word_loop_length) $TMPDIR space requirements by about 100% for a 30,000-record file and 50% for a 1.5-million-record file.}

Reducing the number of processes can result in a slight reduction in the $data_scratch. As noted above, it can result in a great reduction in $TMPDIR space.

The location of $data_scratch and $data_files is specified in the ./xxxnn/prof_library and can be temporarily changed, if necessary. (See the “Moving $TMPDIR, $data_scratch, and $data_files” section immediately below.)

If you are not using a RAID array (with a single file system, with its own internal distribution mechanism), it is highly desirable from a performance standpoint for each of these to be in a different file system. (We suggest that you temporarily move the directory in order to achieve this.) The most important concern is that the $TMPDIR be in a different file system than the others. (See the “Moving $TMPDIR, $data_scratch, and $data_files” section immediately below.)
If archive logging is on (see Section 3, above), you need to make sure that your archive log is big enough to accommodate the large number of log records which will be written.

In addition, if you have added fields/indexes to tab11 since the last run of the p_manage_01 or p_manage_02 jobs, consider the possibility that the Oracle tables, which the job produces, may need to have their extents increased or the disk(s) they write to may need to be increased.

We have found (version 12.4) that for 1.6 million BIB records with 5 processes you need at least:
- 16 gigabytes free space in the xxxnn library’s $data_scratch (e.g., exu01/scratch) directory (for storing temporary files);
- 4 gigabytes free space in the xxxnn library’s $data_files directory (e.g., exu01/files) (for temporarily storing the sequential index files prior to their loading into Oracle);
- 8 gigabytes free space in the $TMPDIR (for sort space).

In another case (version 12.4), p_manage_01 for 1.7 million BIB records with 8 processes failed when there was only 7.5 gigabytes of space for $TMPDIR but was successful with an xxxnn $data_scratch with 20 gigabytes free space and $TMPDIR with 10 gigabytes free space.

In a third case, under version 14.2, ("Site B" in Section 10, below), the p_manage_01 ran successfully for 1.7 million BIB records, with 19 Word indexes, with "setenv ADJACENCY Y", with 9 processes, with word_loop_length 50000, with:

21 gig. $data_scratch
14 gig. $TMPDIR
3 gig. $data_files ,

taking 20 hours. [As noted in the previous section, this would probably have been better with 4 processes and word_loop_length 100000.]

The resulting Oracle tables were using these amounts of space:

<table>
<thead>
<tr>
<th>Table</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z97</td>
<td>900 meg</td>
</tr>
<tr>
<td>Z97_ID</td>
<td>400 meg</td>
</tr>
<tr>
<td>Z97_ID1</td>
<td>400 meg</td>
</tr>
<tr>
<td>Z97_ID1</td>
<td>200 meg</td>
</tr>
<tr>
<td>Z950</td>
<td>6.2 gig</td>
</tr>
<tr>
<td>Z950_ID</td>
<td>50 meg</td>
</tr>
<tr>
<td>Z98</td>
<td>1.8 gig</td>
</tr>
<tr>
<td>Z98_ID</td>
<td>400 meg</td>
</tr>
</tbody>
</table>
Moving $TMPDIR, $data_scratch, and $data_files

- To move $TMPDIR: Create a new file and then change the $TMPDIR entry in ../alephe/aleph_start_505. Example: Define a /aleph15/tmp and then change:

    # setenv TMPDIR          ${ALEPH_MOUNT}/a5${VERSION_NUMBER}_5/tmp
    setenv    TMPDIR             /aleph15/tmp

- To move data_scratch: Create a new file and then change the $data_scratch entry in ../xxxnn/prof_library. Example: Define a /aleph2/xxx01_scratch and then change:

    # setenv data_scratch       $xxx01_dev/xxx01/scratch
    setenv    data_scratch       /aleph2/xxx01_scratch

    And create a symbolic link to the new data_scratch (first make sure there’s nothing you want in the old scratch):

    aleph-M525=XXX01>>rm -R scratch
    aleph-M525=XXX01>>ln -s /aleph2/xxx01_scratch scratch

- To move data_files: Create a new file and then change the $data_files entry in ../xxxnn/prof_library. Example: Define a /aleph2/xxx01_files and then change:

    # setenv data_files         $xxx01_dev/xxx01/files
    setenv    data_files         /aleph2/xxx01_files
6. Preparation for Index Jobs

When p_manage_01 (and, to a lesser extent, p_manage_02) are run against large databases, they can require a great deal of space and run for a long time. You should:

**Clean temp/scratch directories**

Make sure that the xxx01 $data_scratch, the xxx01 $data_files, and the $TMPDIR directories are cleaned of any extraneous, temporary files. If you think, based on calculations in the preceding disk space section, that the disks the $data_scratch, $data_files, or $TMPDIR reside on may not have enough space, consider moving them temporarily to a different space.

**Check Oracle space**

You can use util a/17/11 (Check space utilization of Oracle tables) to make sure that the relevant files (z950/z97/z98 for Words, z01/z02 for Headings, and z11 for Direct index) are not near their maximum number of extents (505, as delivered by Ex Libris) or maximum tablespace sizes. If they are, the extent sizes specified in the xxx01/file_list should be increased prior to running the job or the tablespace(s) should be increased.

**Cancel jobs which might interfere.**

The manage_nn index jobs may need to run when backup jobs or other jobs normally run. You need to make sure that jobs which would interfere with the index job are canceled. The mechanisms for automatic running are the ALEPH job daemon and the Unix cron. The ALEPH job daemon can be canceled via util e/14 or individual jobs can be commented out via util e/15. Consult your Unix systems administrator in regard to cron jobs which might interfere.
7. Unlocking the Library While the Job Is Running

[Note: Sites with relatively small databases (fewer than 300,000 titles) which can be processed in an overnight window should just leave the library locked (in those cases where the job locks it) -- and not worry about this section.]

- p_manage_17 and UE_08 do not lock the library and do not interfere with normal use of the system (except, perhaps, slowing things down slightly).

- p_manage_07 locks the library in 12.2 and 12.3, but you can unlock it after about 5 minutes (after the drop/recreate of the Z13 has occurred).

- p_manage_12 does not lock the library but note that when you specify DELETE=YES, the OPAC becomes unusable because you do not have the links to the items and HOLDings, that is, no locations and no call numbers.

- p_manage_01, 02, 05, 08, and 27 also all lock the library. Generally, the best rule is to leave the library locked while these jobs are running. If online updates were to occur to the BIB (or authority) file while the p_manage_05, 08, or 27 were running, these updates (and perhaps the job) would fail. Most of these jobs run quickly enough to fit in an overnight window.

At some sites the p_manage_01 and p_manage_02 run for a long time (see the “Job Run Time Estimates” section below) and you may need to unlock the library. This can be done if you make sure that no online index updates are performed. How do you do this? You may think that making sure that no one is doing Cataloging is enough, but you also need to consider the Fast Cat function in Circulation and the possibility that there is a backlog of Z07 updates (produced, for instance, by UE_08) waiting to be processed. Basically, you need to make sure that the UE_01 indexing daemon is not running. (Note that when you unlock the library, the UE_01 process is restarted immediately and automatically!)

To ensure that no online index updates are attempted:

1. From the Web Services menu, select the xxxnn library and submit the job.
2. On the server: run “util c/l” to verify that the job is running.
3. Enter:
   ```
   "dlib xxxnn"
   "touch $data_scratch/util_e_stop"
   "ds"
   ```
4. Verify that a "util_e_stop" file has been created in the xxxnn data_scratch directory.
5. Run “UTIL c”, then “6” to unlock the library.
6. After the index job is done, enter:
   ```
   "dlib xxxnn"
   "rm $data_scratch/util_e_stop"
   "util e"
   ```
7. Restart the UE_01, UE_08, and any other UE processes which normally run in this library.
8. Run "util c 1" to verify that the UE processes are running.
8. Monitoring the Jobs

After you submit the job you may want to verify that it is still running and to check its progress. This is done differently depending on the job.

**p_manage_01**

The Oracle files involved are:
- **doc** - the document table which is being indexed (in the case of a BIB library, this is the BIB file)
- **Z970** - word synonyms (optional)
- **Z97** - word dictionary
- **Z98** - bitmap
- **Z980** - cache of bitmap updates (not written to by manage_01; updated online only)
- **Z950** - words in document (for discussion of z95 vs. z950 see Appendix C, below)

(Section 5, above, discusses the disk space required for manage_01 work files and also includes an example of the space occupied by these Oracle tables.)

The “central” log file for the job is `xxxnn_p_manage_01.nnnnn`. If the job is submitted through the Web Services, this log is in the alephe/scratch directory.

There are six steps to the job:
1. (when "Rebuild entire") build z97 records from z970 synonym records *(optional)*
2. **p_manage_01_a**: read doc records, extract words
3. **p_manage_01_b**: sort words
4. **p_manage_01_c**: build/write z97 record
5. **p_manage_01_d**: build/write z950 record
6. **p_manage_01_e**: merge + build/write z98

These steps write work files and their own “sub-logs” (example: `p_manage_01_a1.log`) to the `xxxnn/scratch` directory. In addition, the `p_manage_01_a` step, which does parallel retrieval of the records from the input file, and the `p_manage_01_e` step, which does parallel merging and building of the z98, produce `p_manage_01_a.cycles` and `p_manage_01_e.cycles` files, respectively. These files have a column for each sub-step of the step and rows showing the different sections into which the input file has been divided for processing. The four columns in `p_manage_01_a.cycles` represent the _a, _c, _d, and _d1 steps, respectively. As a sub-step is performed for a particular section of the file, it is marked with a “+”. The final _e step cannot start until all of the a - d steps are completed.

Restart of this final step (should it fail) is described in the Troubleshooting (Section 9) heading “Restart of p_manage_01_e” below.

**p_manage_02**

The Oracle files involved are:
- **doc** - the document table which is being indexed (in the case of a BIB library, this is the BIB file)
Z01 - headings file
Z02 - ACCDOC – which documents contain each heading
Z114 – Pinyin Chinese dictionary (optional)

The “central” log file for the job is xxxnn_p_manage_02.nnnnn (where “xxxnn” is the library). If the job is submitted through the Web Services, this log will be in the alephe/scratch directory.

There are four steps to the job: p_manage_02_a, _c, _d, and _d1.

These steps write work files and their own “sub-logs” (example: p_manage_02_a_1.log) to the xxxnn/scratch directory. In addition, the p_manage_02_a step, which does parallel retrieval of the records from the input file produces a p_manage_02.cycles file. This file has a column for each sub-step of the step and rows showing the different sections into which the input file has been divided for processing. As a sub-step is performed for a particular section of the file it is marked with a “+”.

**p_manage_05**

The Oracle files involved are:
- doc - the document table which is being indexed (in the case of a BIB library, this is the BIB file)
- Z11 - the direct index

The “central” log file for the job is xxxnn_p_manage_05.nnnnn (where “xxxnn” is the library). If the job is submitted through the Web Services, this log is in the alephe/scratch directory.

There are two steps to the job: p_manage_05_a and p_manage_05_c.

These steps write work files and their own “sub-logs” (example: p_manage_05_a_1.log) to the xxxnn/scratch directory. In addition, the p_manage_05_a step, which does parallel retrieval of the records from the input file produces a p_manage_05.cycles file. This file has a column for each sub-step of the job and rows showing the different sections into which the input file has been divided for processing. As a sub-step is performed for a particular section of the file it is marked with a “+”.

**p_manage_07**

The Oracle files involved are the
- doc - the document table which is being indexed (in the case of a BIB library, this is the BIB file)
- Z13 - short doc file

The central log file for the job is xxxnn_p_manage_07.nnnnn (where “xxxnn” is the library). If the job is submitted through the Web Services, this log is in the alephe/scratch directory.

There is one step to the job: p_manage_07_a.
This job does not write any work files; it updates the Z13 file directly. The job writes a sub-log for each process to the $xxxnn$/scratch. In addition, there’s a $p_{\text{manage\_07}}$.cycles file which shows the different sections into which the input file has been divided for processing. You can also monitor its progress by seeing how many records have been written to the Z13. (The assumption is that one Z13 record is written for each BIB record.) This SQL command shows you how many there are:

```
SQL-EXU01> select count (*) from z13;
```

**p_{\text{manage\_08}}**
The Oracle files involved is the Z01 which is exported and then re-imported.

The log file for the job is $xxxnn_{\text{p\_manage\_08}}.nnnnn$ (where “xxxnn” is the library). If the job is submitted through the Web Services, this log is in the alephe/scratch directory.

The $p_{\text{manage\_08}}$ invokes two subprocedures, $p_{\text{file\_03}}$ and $p_{\text{file\_04}}$. The $p_{\text{file\_03}}$ procedure writes the Z01 sequential file to the $xxxnn/files$ directory. The Z01 is dropped/recreated. And then the $p_{\text{file\_04}}$ proc reads the file back in from the $xxxnn/files$ directory.

**p_{\text{manage\_12}}**
The Oracle files involved are the Z103s. When the job is run for the BIB library the, BIB Z103 is dropped/recreated. When it is run on the ADM, the ADM Z103 is dropped/recreated and the BIB Z103 is updated. When it is run on the HOL, the HOL Z103 is dropped/recreated and the BIB Z103 is updated.

When the job is run for the AUTHority library the AUTHority Z103 is dropped/recreated. Likewise, when the job is run for a Course Reading library, just the $xxx30 Z103$ is dropped/recreated.

The “central” log file for the job is $xxxnn_{\text{p\_manage\_12}}.nnnnn$ (where “xxxnn” is the library). If the job is submitted through the Web Services, this log will be in the alephe/scratch directory.

There is one step to the job: $p_{\text{manage\_12\_a}}$.

This step writes work files and “sub-logs” (example: $p_{\text{manage\_12\_a\_1}}.log$) to the $xxxnn$/scratch directory. In addition, there’s a $p_{\text{manage\_12}}$.cycles file which shows the different sections into which the input file has been divided for processing. As a sub-step is performed for a particular section of the file it is marked with a “+”.
**p_manage_17**

The Oracle file involved is the Z01. The job updates the z01_filing_sequence of headings whose z01_acc_code, z01_alpha, and z01_filing_text are the same.

The log file for the job is \texttt{xxxnn\_p\_manage\_17.nnnnn} (where “\texttt{xxxnn}” is the library). If the job is submitted through the Web Services, this log is in the alephe/scratch directory.

This job does not write any work files; it updates the Z01 file directly. To monitor its progress, run “\texttt{UTIL g 2}” and look at the “last-long-acc-number” value.

**p_manage_27**

The Oracle files involved are

- \texttt{doc} - the document table which is being indexed (in the case of a BIB library, this is the BIB file)
- \texttt{Z101} - sort file

The log file for the job is \texttt{xxxnn\_p\_manage\_27.nnnnn} (where “\texttt{xxxnn}” is the library). If the job is submitted through the Web Services, this log is in the alephe/scratch directory.

This job writes a \texttt{z101.seq} file to $\texttt{data\_files}$. It writes a sub-log for each process to the \texttt{xxxnn/scratch}. In addition, there’s a \texttt{p\_manage\_27.cycles} file which shows the different sections into which the input file has been divided for processing. You can also monitor its progress by seeing how many records have been written to the \texttt{z101} file. This SQL command shows you how many there are:

\begin{verbatim}
SQL-EXU01> select count (*) from z101;
\end{verbatim}

**p_manage_32**

The Oracle files involved are:

- \texttt{Z01} - headings file
- \texttt{Z02} - ACCDOC - which documents contain each heading
- \texttt{Z0102} - Counters for Logical Bases -- which headings belong to which logical bases

[Previously, the OPAC programs needed to read the word indexes (bitmaps) of the Z02s for each Z01 in order to determine if the holdings should be displayed for a particular base. For a small base, it could take many reads to find the records to fill the screen. The Z0102 makes this unnecessary; greatly improving performance for small bases.]

The “central” log file for the job is \texttt{xxxnn\_p\_manage\_32.nnnnn} (where “\texttt{xxxnn}” is the library). If the job is submitted through the Web Services, this log will be in the alephe/scratch directory.

There are three steps to the job: \texttt{p\_manage\_32\_a, _c, and _d}. 
The p_manage_32_a reads the Z01 headings, sorts them, and matches their Z02s with each base; the p_manage_32_c moves the file to $data_files/z0102.seq and loads them into Oracle; the p_manage_32_d updates the Z01-UPDATE-Z0102 flag to “N”. These steps write their own “sub-logs” (example: p_manage_32_a_1.log) to the xxxn/scratch directory and produce a p_manage_32.cycles file. This file has a column for each sub-step and rows showing the different sections into which the input file has been divided for processing. As a sub-step is performed for a particular section of the file it is marked with a “+”.

**p_manage_102**

The Oracle files involved are the doc - the **authority** document table
Z01 - the bib headings file

The log file for the job is xxxnn.p_manage_102.nnnnn (where “xxxnn” is the library). *Currently, this job is not in Web Services. It would need to be submitted from the command line. See the sample command in Appendix A.*

There are two steps to the job: p_manage_102_a and p_manage_102_c.

The p_manage_102_a reads the headings from the authority record and sorts them; the p_manage_102_c builds the $data_files/z01.seq and loads it to the Oracle table. The job writes a sub-log for each process to the xxxn $data_scratch. In addition, there’s a p_manage_102.cycles file which shows the different sections into which the input file has been divided for processing.

You can also monitor the progress of the job by seeing how many records have been written to the Z01. (The assumption is that one Z01 record is written for each heading in the authority file.) This SQL command shows you how many have been written:

```
SQL-EXU01> select count (*) from z01;
```

**UE_08**

The Oracle files involved are the BIB Z01 and the authority library Z01.
The log appears as “run_e_08.nnnnn” in the xxxnn/scratch directory (where “xxxnn” is the library. See Section 2.3, above, for more information on ue_08.

© Ex Libris Ltd., 2001
How To Run Index Jobs
Release 14.2
Updated: September 17, 2001
9. Troubleshooting

In general, you should let the job run to completion and then examine the log(s) as described in the previous sections to make sure that it was successful.

How can you tell if it was successful?

- Did it leave the Library locked? (This usually indicates a problem.)
- Search the log (the basic $alephe_scratch log) for these strings: 'error ' (with one blank after the "r"), 'space', 'fail', 'duplicate', 'extend'.
  Search the logs (if any) in the xxxnn/scratch for the same strings.
- Run util f/4 on the file it was supposed to create: Do you see records? Are they in order?
- Do util a/17/14 --are the Oracle indexes for the table(s) present and valid?
- Do an sql record count: "select count(*) from znn" --is the count what you expect?

- Job is Stuck

If you find, however, that the job is not progressing (it is writing out the same record over and over again, or not writing out any records), you should cancel the job. Please note that in the Oracle load step some jobs may go for long periods without writing anything to the log file. This is normal. You can use the

SQL-xxx01> select count(*) from znn;

query to monitor progress in writing to the actual Oracle table.

You may see in the log of p_manage_01_a, or p_manage_02_a, that the same loop of 20000 (for example) repeats itself because of a crashing expand on a particular document. Please contact Ex Libris Customer Support in such cases.

Or you may see that certain p_manage_01_a cycles are stuck. You see this in the p_manage_01.cycles file (where certain cycles are still in step 1 while cycles after them have all steps done). And the logs for these cycles haven't been written to for a long time. This is probably a problem in trying to extract a particular bib record. To see what record that is, look at the end of the manage_01_1_0 file (where "n" is the process number). For instance, if the p_manage_01_a_6.log is the one which hasn't been written to, then check the end of the manage_01_1_6 file.

To cancel the job, you enter:

ps -ef | grep manage_nn (where "nn" is the job number) and
ps -ef | grep z0 (for z01/z02) or grep z9 (for z95/97/98).

And then

kill -1 nnnnn (where "nnnnn" is the process number, the first number) for each of the processes you found in the first steps (the "ps -ef | grep"s).

Then enter the two grep commands above again. If some of the processes are still active after several minutes, then repeat the kill, using “kill –9 nnnnn” this time.
• Restart

With p_manage_01, _02, and other jobs with “.cycles” files, there is a potential to use the "cycles" files (see the previous "Monitoring the Jobs" section) to make the job resume at the point where it left off when it is resubmitted. Among other things, you should edit the .cycles file itself so that sections that were in the middle of being processed will be redone.

Restart procedures are described in the "How To Run Index Jobs (Advanced Supplement)" (not yet published as of this writing). This should be done only in coordination with Ex Libris Customer Support. Generally, it is better to start the job over from scratch.

Restart of p_manage_01_e

If the last step of the long p_manage_01 job (p_manage_01_e) fails, you can save quite a bit of time by rerunning just this step. It’s handled differently than the failure of the a-d1 steps (where you would modify the cycles file). p_manage_01_e must be rerun as a whole, for all cycles:

• Drop/recreate the z98 table (using util a/17/1).
• Drop the z98_id (using util a/17/4).
• Uncompress the xxx01/scratch/manage_01_4.*.del.Z files by entering the command: `uncompress *del.Z`. (Note that the uncompressed files take 4-5 times the space of the uncompressed files.)
• Remove the “.del” suffix from each manage_01_4.*.del file.
• Run the p_manage_01 with the same parameters as before, except, instead of specifying “p_manage_01” as the proc to be executed, specify “p_manage_01_e”.
• Build Oracle indexes
  o Drop z97_id1, z97_id2, z97_id3 (if they exist) (util a/17/4)
  o Create z97_id1, z97_id2, and z97_id3 (util a/17/2) (Note: z97_id3 is optional; only used if you have z970 Word synonyms)
  o Create z98_id (util a/17/2)

• Job is Slow

If you find that a job is taking much longer than it should (based on previous experience or on Section 10 of this document), you can use this to diagnose:

Has something changed? (More records? Different expands being used? Expands which read additional records can add greatly to i/o and total time.)

Or, does one of the files which it’s reading not have a proper Oracle index built on it? {Tables lacking indexes *can* still be used (--Oracle doesn’t give you any error message--), it’s just that reading of the records is extremely slow.}

You can use util a/17/14 ("List existing indexes for a table") to check what Oracles indexes
exist and if they are valid.

If you are working with a new version, consider that possibility that a new index may have been added for the file but is not actually defined in your Oracle.

Are you specifying the same number of processes as was specified previously?

Has the cycle size (acc_loop_length, word_loop_length, etc.) been greatly reduced? (It should never be less than 1% of the number of doc records; 50000 works well for anything under 2 million doc records.) See discussion in Section 5 (Disk Space).

Was Oracle Archive Logging turned off when you ran the job before, and it’s now turned on? This can make a 20-50% difference.

[Note: There are other entries in the WebPRB relating to the slowness of specific jobs: search by job number.]

- **Specific Error Messages**

  For suggestions on how to handle other specific error messages/error conditions, please consult the WebPRB OPAC or call/email Ex Libris Customer Support.

- **Problems using/searching indexes**

  If the problem you are seeing is in using/searching indexes, then please consult the “Diagnosing Index Problems” document.
10. Estimating Run Time

[Note: Sites with relatively small databases (fewer than 300,000 titles) which can be processed in an overnight window will probably want to ignore this section.]

The run time depends on

- The number of BIB (or authority) records to be processed
- The number of processes specified (the greater the number of processes, the less the run time)
- The “cycle size” (the value of word_loop_length or acc_loop_length in the xxxnn/prof_library; the greater the cycle size the less the run time)
- Whether or not archive logging is on,
- What expand routines are specified in xxx01/tab/tab_expand for
  - WORD (in the case of p_manage_01),
  - ACC (in the case of p_manage_02), and
  - INDEX (in the case of p_manage_05)
  - the more expands specified and the more other records they read (items, HOL, etc.), the longer the job will take; and
- (For p_manage_01, 02, or 05) how many different index codes you have specified in tab11 and how many indexed fields (on average) the BIB (authority) records contain which are included in these indexes

Here are some examples of how long particular jobs have taken (in hours) at particular sites (A, B, C, and D).

Site A (version 14.2) 1.7 million BIB records; 600,000 authority
BIB indexes: 14 IND; 46 ACC; 39 Word (13,800,000 ACC headings)
Authority: 13 IND; 3 ACC; 16 Word (4,000,000 ACC headings)
manage_01, 02, 05, 07 run with 9 processes; archive logging off;
word_loop_length = 50000; acc_loop_length = 50000
[As described in Section 4, word_loop_length 100000 with 4 processes would probably be better for this size file.]

Site B (version 14.2) 1.7 million BIB records; 800,000 authority
BIB indexes: 18 IND; 13 ACC; 19 Word (12,000,000 ACC headings)
Authority: 1 IND; 6 ACC; 2 Word (5,000,000 ACC headings)
manage_01, 02, 05, 07 run with 9 processes; archive logging off;
word_loop_length = 50000; acc_loop_length = 50000
[As described in Section 4, word_loop_length 100000 with 4 processes would probably be better for this size file.]

Site C (version 12.4) 2.2 million BIB records; 750,000 authority
BIB indexes: 17 IND; 27 ACC; 41 Word (50,000,000 ACC headings)
Authority: 2 IND; 14 ACC; 6 Word (4,000,000 ACC headings)
manage_02 with 5 processes;
acc_loop_length = 30000; [word_loop_length = 50000]
Site D (version 12.4) 1.6 million BIB records; 780,000 authority
   BIB indexes: 13 IND; 19 ACC; 37 Word (7,500,000 ACC headings)
   Authority: 10 IND; 10 ACC; 17 Word (2,300,000 ACC headings)
   manage_01, 02, 05 run with 7 processes; archive logging off;
   word_loop_length = 20000; acc_loop_length = 20000

**Average, 8 others**: This is an average of the number of records processed per hour
in runs at 8 other sites (under 12.4). These are all for the BIBliographic library.
Almost all before archive logging was turned on.

**Warning**: Unless otherwise noted, these times are with archive logging off. Running
the job with archive logging on will add 25-50% to the run time.

<table>
<thead>
<tr>
<th>p_manage_01:</th>
<th>BIB (hours)</th>
<th>Authority (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site A (14.2)*:</td>
<td>21 hrs.</td>
<td>4.5 hrs.</td>
</tr>
<tr>
<td>Site B (14.2)*:</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>Site C (12.4):</td>
<td>22</td>
<td>4.5</td>
</tr>
</tbody>
</table>

* = Under 14.2, with "setenv ADJACENCY Y" specified in alephe/aleph_start_505.
"setenv ADJACENCY Y" increases the runtime for p_manage_01 by 50%.

Average, 8 others, 12.4: 120,000 recs/hr (*varying from 50,000/hr – 200,000/hr*)

<table>
<thead>
<tr>
<th>p_manage_02:</th>
<th>BIB (hours)</th>
<th>Authority (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site A (14.2):</td>
<td>9.5 hrs.</td>
<td>2 hrs.</td>
</tr>
<tr>
<td>Site B (14.2):</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>Site C (12.4):</td>
<td>17 (arch.log on)</td>
<td></td>
</tr>
<tr>
<td>Site D (12.4):</td>
<td>7</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Average, 8 others, 12.4: 160,000 recs/hr (*varying from 100,000/hr – 250,000/hr*)
   (*130,000/hr with 3 processes; 180,000/hr with 7*)

<table>
<thead>
<tr>
<th>p_manage_05:</th>
<th>BIB (hours)</th>
<th>Authority (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site A (14.2):</td>
<td>3 hrs.</td>
<td>0.25 hrs.</td>
</tr>
<tr>
<td>Site B (14.2):</td>
<td>2.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Site D (12.4):</td>
<td>3</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Average, 8 others, 12.4: 700,000 recs/hr (*varying from 200,000/hr – 1,000,000/hr*)

<table>
<thead>
<tr>
<th>p_manage_07:</th>
<th>BIB (hours)</th>
<th>Authority (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site A (14.2):</td>
<td>8.5 hrs.</td>
<td>1 hr.</td>
</tr>
<tr>
<td>Site B (14.2):</td>
<td>2.5</td>
<td>1.3</td>
</tr>
<tr>
<td>Site D (12.4):</td>
<td>1.5</td>
<td></td>
</tr>
</tbody>
</table>

Average, 8 others, 12.4: 1,000,000 recs/hr (*varying from 200,000/hr – 1,000,000/hr*)

| p_manage_08 (for Z01): | |
|--------------------------|
Site A (14.2):  2.5 hrs.    0.25 hrs
Site B (14.2):  3            0.5
Site C (12.4):                      4
Site D (12.4):                      2

<table>
<thead>
<tr>
<th>BIB (hours)</th>
<th>Authority (hours)</th>
</tr>
</thead>
</table>

**p_manager_12**:  
Site A (14.2):  0.5 hrs (xxx01)  
4.5 (xxx50)  
5.0 (xxx60)  

* Normally needs to be run **only** as part of conversion.

**p_manager_17**:  
Site A (14.2):  9 hrs. (turbo)    2 hrs. (turbo)
Site B (14.2):  14 hrs. (turbo)    35 (non) 4.5 (turbo)
Site C (12.4):                      15 hrs. (arch.log on)  2
Site D (12.4):                      15 hrs. (arch.log on)

**p_manager_27**:  
Site A (14.2):  8 hrs.    0.5 hrs.
Site B (14.2):  8.5     0.3
Site C (12.4):  12 (arch.log off)
Site D (12.4):  0.5
Average, 8 others, 12.4:  700,000 recs/hr (varying from 200,000/hr – 1,500,000/hr)

**p_manager_102 (14.2 and higher only)**:  
Site A (14.2):  1.25 hrs.    <not applicable>
Site B (14.2):  4        <not applicable>

**UE_08**: When run in N (New) mode, it takes roughly 1 second for every 50 ACC headings plus 1 second for each ACC heading which is linked to an authority record. For a typical site this might be 10 days (3 days plus 7 days).

Note: with 14.2, where p_manager_102 is run, you should **never** run ue_08 in N (New) mode.
11. Appendix A: Sample Commands for Running Jobs from Command Line

Note: These are intended as samples only. The parameters may not be appropriate for your particular site. If you have a question as to what the parameters in the command should be, we suggest submitting the job from the Web Services and seeing (in the main log in $alephe_scratch) what parameter string is generated.

Note: If you paste a command from a Word document like this onto the unix command line, you can do only one line at a time. Pasting both lines of a two-line command will not work.

Bib library:

csh -f $aleph_proc/p_manage_05 EXU01,1,000000000,999999999,9,0,00, > & $alephe_scratch/exu01_p_manage_05.log &
csh -f $aleph_proc/p_manage_01 EXU01,1,000000000,999999999,,7,0,00, > & $alephe_scratch/exu01_p_manage_01.log &
csh -f $aleph_proc/p_manage_07 EXU01,1,000000000,999999999,9,0,00, > & $alephe_scratch/exu01_p_manage_07.log &
csh -f $aleph_proc/p_manage_27 EXU01,9,0,00 > & $alephe_scratch/exu01_p_manage_27.log &

Headings, Method #1 (Authorities, large database):

csh -f $aleph_proc/p_manage_102 EXU01,EXU10,1,000000000,999999999,? > & $alephe_scratch/exu01/10_p_manage_102.log &
csh -f $aleph_proc/p_manage_02 EXU01,0,000000000,999999999,,9,0,00 > & $alephe_scratch/exu01_p_manage_02 &
csh -f $aleph_proc/p_manage_08 EXU01,z01,go > & $alephe_scratch/exu01_p_manage_08.log &
csh -f $aleph_proc/p_manage_17 EXU01,Y,0,00 > & $alephe_scratch/exu01_p_manage_17.log &

Headings, Method #2 (no Authorities or smaller database):

csh -f $aleph_proc/p_manage_02 EXU01,1,000000000,999999999,,9,0,00 > & $alephe_scratch/exu01_p_manage_02 &
csh -f $aleph_proc/p_manage_08 EXU01,z01,go > & $alephe_scratch/exu01_p_manage_08.log &
csh -f $aleph_proc/p_manage_17 EXU01,Y,0,00 > & $alephe_scratch/exu01_p_manage_17.log &
Authority library:

csh -f $aleph_proc/p_manage_05 EXU10,1,000000000,999999999,9,0,00, > & $alephe_scratch/exu10_p_manage_05.log &
csh -f $aleph_proc/p_manage_01 EXU10,1,000000000,999999999,,7,0,00, > & $alephe_scratch/exu10_p_manage_01.log &
csh -f $aleph_proc/p_manage_07 EXU10,1,000000000,999999999,9,0,00, > & $alephe_scratch/exu10_p_manage_07.log &
csh -f $aleph_proc/p_manage_27 EXU10,9,0,00 > & $alephe_scratch/exu10_p_manage_27.log &
csh -f $aleph_proc/p_manage_02 EXU10,1,000000000,999999999,,9,0,00 > & $alephe_scratch/exu10_p_manage_02 &
csh -f $aleph_proc/p_manage_08 EXU10,z01,go > & $alephe_scratch/exu10_p_manage_08.log &
csh -f $aleph_proc/p_manage_17 EXU10,Y,0,00 > & $alephe_scratch/exu10_p_manage_17.log &

ADM library:

csh -f $aleph_proc/p_acq_04 EXU50 > & $alephe_scratch/exu50_p_acq_04.log &
csh -f $aleph_proc/p_manage_111 EXU50,z70,Y,0,00 > & $alephe_scratch/exu50_p_manage_111.z70.log &
csh -f $aleph_proc/p_manage_111 EXU50,z76,N,0,00 > & $alephe_scratch/exu50_p_manage_111.z76.log &
csh -f $aleph_proc/p_manage_111 EXU50,z303,N,0,00 > & $alephe_scratch/exu50_p_manage_111.z303.log &

Other:

csh -f $aleph_proc/p_manage_12 EXU30,Y,Y,000000000,999999999,9,0,00 > & $alephe_scratch/p_manage_12_exu50.log &
csh -f $aleph_proc/p_manage_12 EXU60,Y,Y,000000000,999999999,9,0,00 > & /aleph/scratch/p_manage_12_exu60.log.0622 &
12. **Appendix B. How Many Processors Do You Have?**

For Sun, use the command:

"uname -X" (the numCPU value)

or, as root:

"psrinfo"

...

For AIX, Digital, or Compaq use the command:

"lsdev -C | grep proc"

(there will be a line for each processor in the resulting display)

or, as root,

"lcsfg -v | grep proc"
13. Appendix C. Difference between Z95 and Z950

The z95 is a grouping of multiple z950s into a single record. Both the z95 and z950 are used only for Word proximity searches. (The search first looks for a z95 for this document and, if it doesn't find it, looks for z950s.) If a search does not involve the "!" or "%" proximity operators, then it doesn't use either the z95 or the z950. (Prior to 14.2, the "!0" (zero-distance proximity) was used for Word adjacency searches. When "setenv ADJACENCY Y" is set in 14.2, the z95 and z950 are not used for adjacency –only for proximity.)

The p_manage_01 Words job produces z950s. The ue_01 online indexing process writes z95 records.

There is a batch procedure p_manage_01_z95_z950 which will read through the z950 and mimic this online function, grouping the multiple records into the single z95. (Words can continue to be used while it is running but it runs for a long time --several days.) Prior to "setenv ADJACENCY", this improved performance for adjacency searches. With "setenv ADJACENCY Y" it improves the performance of proximity searches only.

If your site doesn't use of proximity searching much and you have "setenv ADJACENCY Y", then running p_manage_01_z95_z950 is probably not justified. You can define a z95 which equal to the percentage of records you expect to be added/updated online between runs of p_manage_01. (If you expect 20% of the file to be added/updated, then the z95 should be 20% of the z950.)

Note: converting z950s into z95s does *not* save any space; the resulting z95 takes exactly as much space as the z950s it is created from.