WasteBasket

Version 200

Revision AAA

Users Guide

22. November 2011

greenHouse

Software & Consulting

Karl-Heinz Weber
Heinrichstraße 12
D-45711 Datteln/Horneburg

Please Comment

If you have questions or problems concerning the content of this document, please let me know! Send your comments to:

GreenHouse Software & Consulting
Karl-Heinz Weber
Heinrichstraße 12
D-45711 Datteln/Horneburg
Germany

Phone	+49 (0)2363 72566
Fax	+49 (0)2363 66106
Mobile	+49 (0)172 23 18248
E-Mail:	Info@GreenHouse.de
Internet:	www.GreenHouse.de
PGP fingerprint:	`3A32 D90A D125 5418`
	`1150 2484 6629 2DD2`

1. Table of Contents

2. Introduction

Wouldn't it be nice to easily recover purged files on a Tandem[1] system, instead of restoring them from the archive (you do have backups of your important files, do you?)?

Windows supports this function, while Tandem does not - until today: GreenHouse designed and developed a Tandem/GUARDIAN based WasteBasket, from which a purged file can easy and hassle-free be restored.

And what about purging an open files, e.g. to replace a program file with by a new one? This would make a program upgrade much easier by simply purging the existing file (while keeping its functionality intact) and moving the new file into place (making it available right away).
The ability to even purge open files is one of the features of WasteBasket.

The WasteBasket product is based on a library (WBLIBxxx) which has to be attached to programs, performing the PURGE function, such as FUP, TACL, or other - even user written - programs.

The WBLIB library intercepts these procedure calls:

- FILE_PURGE_
- PURGE

and saves the file(s) to be purged in a subvol named $vol.WASTEGHS.
For performance reasons the purged files are always saved on the original volume, while restoring files from the WasteBasket allows the definition of a target location which can be different from the purged files original location.

WBLIB as well intercepts the procedure call:

- User_Authenticate_

to find out if the User-ID of the process, to which WBLIB is attached, has changed (mainly intended to catch the logon in TACL).

WBDOG

To ensure that WasteBasket does not fill up the disk volumes with purged files, the supportive program WBDOG (WasteBasket Watch Dog), controlled by $ZZKRN, checks the files in WasteBasket once a day and deletes those, residing there for more than a defined number of days.
The execution time of WBDOG as well as the life time of files in the WasteBasket in days is configurable in the EDIT file WBCONFIG.

WBCOM

The command interpreter WBCOM allows the management of the WasteBasket, and the restoration of files from it.

WBSERV

The WBSERVB server process performs all the WasteBasket data base handling: This allows the best security settings on the WasteBasket database files, and keeps the WBLIB library as small as possible.

[1] After 33+ years on this platform it still is a Tandem, nothing else!

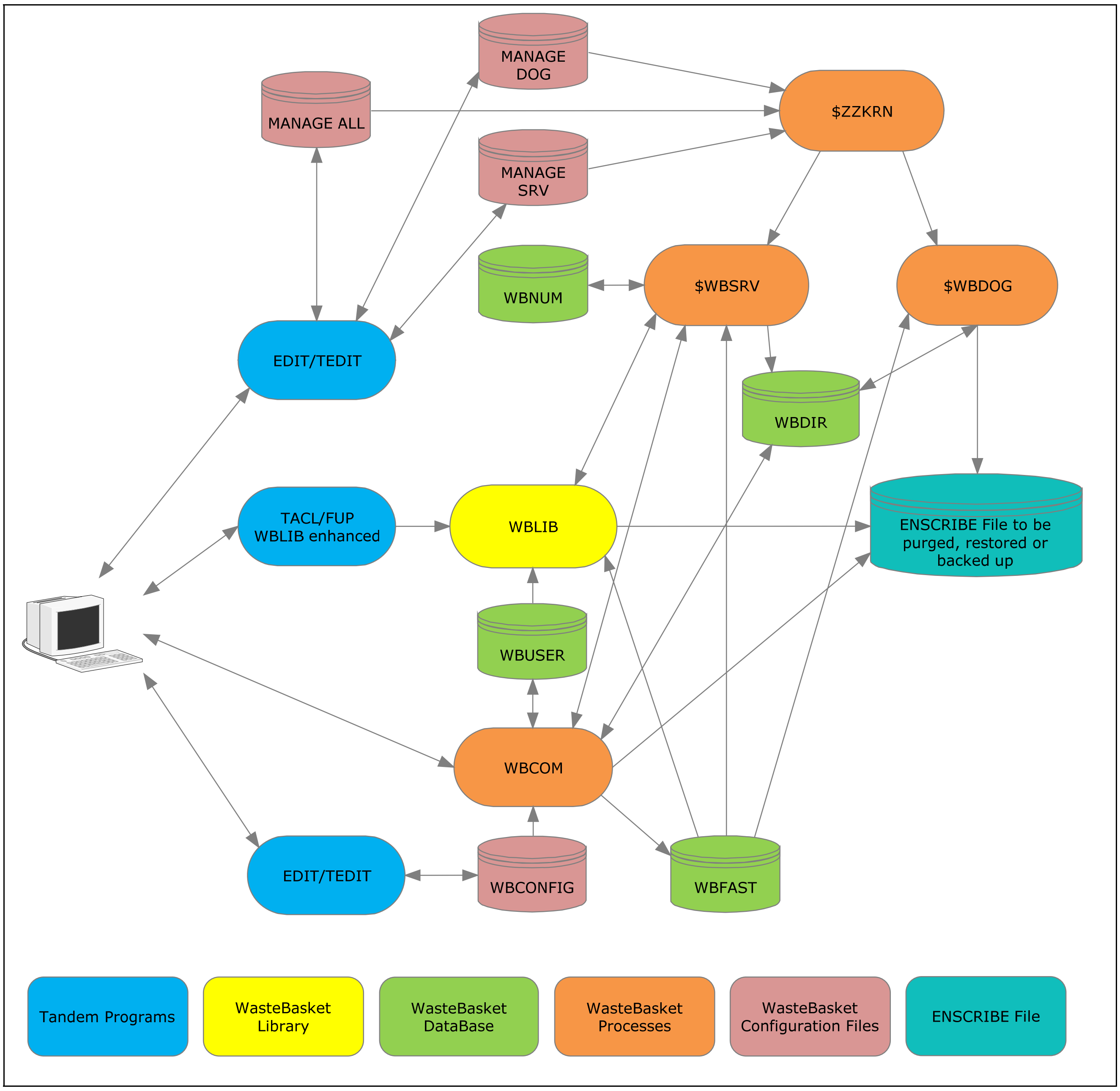
Security Settings

A user can only access files in the WasteBasket he as well has purged.

- GUARDIAN and Alias users are treated individually.
  e.g. the Alias user "CarlWeber" is treated as an individual user. Its relationship to the GUARDIAN user "GHS.CARL" is NOT used to qualify the access.
- Users can be configured for management access on files they did not purge.
- SUPER.SUPER is the only ID with access rights to ALL files in the WasteBasket.

Actually the WasteBasket works on ESCRIBE type files only.
It does **NOT** work an SQL and OSS type files.
SAFEGUARD ACL's are NOT saved.

3. Overview of WasteBasket components



MANAGE ALL|DOG|SRV

TACL macros to start, stop and display information of/from the $WBSRV and $WBDOG persistent processes.

$ZZKRN

Kernel process of the Tandem system.
Controls the $WMSRV server process as well as the $WBDOG clean-up process.

WBNUM

File used to produce monotone WasteBasket entry numbers.

$WBSRV
WasteBasket server process.
- handles the generation of monotone numbers to the base of 36 to name the files in the WasteBasket
-performs the update on the WBDIR directory file
- checks a file for the PURGE access right

$WBDOG
WasteBasket Watch Dog process: Performs the daily clean-up of the WasteBasket from over aged files.

WBDIR
Directory file, where all files in the WasteBasket have a record, can be identified, and restored.

TACL/FUP
WBLib enhanced program performing a PURGE operation.

WBLIB
WasteBasket library performing the PURGE action.
It has to be attached to programs, performing a WasteBasket covered PURGE operation.

WBUSER
WasteBasket file, holding user specific file names and file codes NOT to be taken into account.

ENSCRIBE File
ENSCRIBE type file to be
- purged by a WBLIB enhanced program such as TACL, FUP etc.
- backed up by WBCOM
- restored by WBCOM

OSS and SQL files are NOT supported!

WBCOM
WasteBasket command interpreter:
- produces the WBFAST file
- manages the WasteBasket systems
- restores files

WBFAST
Configuration file in binary format.
Is produced by WBCOM from the WBCONFIG file and used by:
- WBLIB library
- $WBDOG clean-up process
- $WBSRV server process

EDIT/TEDIT
Editor on the Tandem system. Used to edit WBCONFIG.

WBCONFIG
EDIT type configuration file.
Is changed by EDIT/TEDIT and processed by WBCOM to the WBFAST file.

4. WasteBasket Restrictions

WasteBasket does work only when:

- the WasteBasket enhanced program and
- the file to be purged

reside on the same EXPAND node.

e.g.:

FUP on \A purges file on \A	File is moved to the WasteBasket on \A
FUP on \A purges file on \B	File is NOT moved to the WasteBasket
FUP on \B purges file on \A	File is NOT moved to the WasteBasket

Only programs, using the system procedure calls:

- PURGE
- FILE_PURGE_

can be enhanced with the WasteBasket feature.

e.g. RESTORE cannot be enhanced, because it does not use the above mentioned procedure calls.

Only files, where the user has purge access rights, are taken into account! WBLIB does NOT bypass the GUARDIAN and SAFEGUARD security!

5. Naming conventions

The following names are hard coded:

$WBDOG	Process name of WasteBasket clean-up process.
$WBSRV	Process name of WasteBasket data base server.
$SYSTEM.WASTEGHS	Location of WasteBasket executables, configuration file, etc.
$vol.WASTEGHS	Location of the WasteBasket files.
$vol.WASTEGHS.Wxxxxxxxx	Name of file(s) in the WasteBasket. Range of xxxxxxx is: 0000000 up to 9ZZZZZZ, computed to the base of 36. When 9ZZZZZZ is reached, xxxxxxx is reset to 0000000.

6. Installation of WasteBasket

The WasteBasket system consists of a number of files.

All files have to be installed in location: $SYSTEM.WASTEGHS.
This is a hard coded location.

Purged files are saved by WBLIB and WBCOM to subvol WASTEGHS, which is used on ALL volumes.
This location is created automatically when the first file on a volume is purged by the WasteBasket product, and disappears when the last file from that location is deleted.

Perform the following steps to install the WasteBasket product and all its files:

1. Check the system for files in locations: $*.WASTEGHS
   Make sure these locations are empty, or at least do not contain files matching template:
   *.Wnnnnnnn where n is a value of 0000000..9ZZZZZZ, e.g. W00A1234

2. Upload the WasteBasket PAK type file named WB.1729 in **BINARY** mode into an empty subvol, e.g. $SYSTEM.GHS.
   Make sure the file code is 1729.

3. Upload the LicenseToken PAK type file TOKEN.1729 in **BINARY** mode into the same subvol.
   Make sure the file is named TOKEN, and the file code is 1729.

4. Upload the INSTALL TACL Macro INSTALL.101 in **ASCII** mode into the same location.

5. Logon to SUPER.SUPER
   This is necessary because the WBCOM command interpreter program file as well as the WBSERV server object uses PRIV code and needs to be licensed.
   SECOM[2] users can make use of a SUPERTACL instead of re-logging on.

6. Volume over to the location, to which you uploaded the files:
   ```
   VOLUME $SYSTEM.GHS
   ```

7. Install the WasteBasket files in $SYSTEM.WASTEGHS by executing the INSTALL TACL Macro using this command:
   ```
   [RUN] INSTALL  WB  $SYSTEM.WASTEGHS  <password>
   ```
   where

`<password>`	is the user specific password to UNPAK the TOKEN.1729 file

   e.g.:
   ```
   [RUN] INSTALL  WB  $SYSTEM.WASTEGHS  jajaduda23
   ```

   This unpacks the WasteBasket files the target location $SYSTEM.WASTEGHS, creates the data files, and sets the security attributes.

8. Volume over to $SYSTEM.WASTEGHS

[2] SECOM (Secure Command Manager) performs Single Sign on ON the system, ID hopping, session tracing, and much more.

9. EDIT the WBCONFIG file and configure these items:
   - the prompt for WBCOM
   - the default life time in days of the files in the WasteBasket; a good value is 6.
   - the DogTime; it defines, at what time $WBDOG runs the clean-up.
   - up to 45 file name templates defining those files NOT to be saved by WasteBasket
   - up to 45 file code templates defining those file codes NOT to be saved by WasteBasket
   - a flag causing $WBDOG to start DCOM
   - a flag causing WBLIB to de-allocate not used extents at PURGE time
   - the method of displaying files
   - up to 10 administrators
   - the default life time of backed up files
   - up to three names of the WBLIB object file
   - the name of the $WBDOG process
   - the name of the $WBSRV process
   - purge access on an existing ACL

   Detailed information for each item can be found in WBCONFIG.

10. Edit the STARTDOG TACL Macro and make sure, that the defined name (wbdog) is identical with the name defines in WBCONFIG.

11. Edit the STARTSRV TACL Macro and make sure, that the defined name (wbsrv) is identical with the name defines in WBCONFIG.

12. Start the $WBDOG process by executing the STARTDOG TACL Macro.
    To display information about $WBDOG, execute the INFODOG TACL Macro.
    In case you need to stop the $WBDOG process, execute the STOPDOG TACL Macro.
    Stopping $WBDOG does not have any effect on PURGE operations, but cause the disk volumes to run full!
    To successfully perform these steps, you have to be logged on to a user of the SUPER-group.

13. Start the $WBSRV process by executing the STARTSRV TACL Macro.
    To display information about $WBSRV, execute the INFOSRV TACL Macro.
    In case you need to stop the $WBSRV process, execute the STOPSRV TACL Macro.
    Stopping $WBSRV causes the WasteBasket system to stop saving purged files.
    To successfully do this, you have to be logged on to a user of the SUPER-group.

14. Execute the WBCOM command interpreter with the MAKEFAST command:
    ```
      [run] WBCOM MAKEFAST
    ```
    WBCOM checks the WBCONFIG file and produces the WBFAST file which is used by $WBDOG, $WBSRV and the WasteBasket library WBLIB.
    The MAKEFAST command has to be executed each time WBCONFIG is changed!
    Changes are automatically taken into account by $WBDOG and $WBSRV, no re-start is required!

15. Purge the uploaded files in $SYSTEM.GHS
    It helps to keep your system clean.

16. Logoff from SUPER.SUPER.

The security settings of the WasteBasket files are set during the installation by the XSECURE TACL Macro.

Please do NOT change the security settings of files in $vol.WASTEGHS.*.

The WasteBasket is now installed on your system, but it is NOT active yet!
To activate the WasteBasket functionality, WBLIB has to be attached to the programs, performing a file PURGE, such as FUP, TACL etc.
All NON enhanced programs work normal, that is: When they purge a file, it is NOT saved by the WasteBasket, but it is gone.

7. Test of WasteBasket with a copy of FUP

The installation procedure stored all relevant files onto your system and secured them properly, but the WasteBasket is NOT active yet!
To activate the WasteBasket, the WBLIB code has to be attached to programs, performing a file purge, such as FUP, TACL etc.

To become familiar with the WasteBasket and its functionality, we recommend testing it with a copy of FUP.

1. Logon to SUPER.SUPER
   This is necessary because FUP runs PRIV code and has to be re-licensed after copying it, and when the library is attached.

2. Volume over to an empty location.

3. Get a copy of the current FUP program file:
   ```
   FUP DUP $SYSTEM.SYSnn.FUP,*,SAVEALL
   ```

4. Bind the WBLIB100 library file to this FUP by executing the utility BINDLIB:
   ```
   [RUN] $SYSTEM.WASTEGHS.BINDLIB FUP WITH $SYSTEM.WASTEGHS.WBLIB100
   ```
   e.g.:
   ```
   $GHS1 WASTEB 414> bindlib fup with wblib100
   BINDLIB (400) - T7172H06 - (18Aug2010)   System \GINKGO, running NSK H06.19
   Copyright (c) GreenHouse Software & Consulting 2001-2004,2007,2009,2010
   Number of successful binds:  1
   $GHS1 WASTEB 415>
   ```
   BINDLIB is a GreenHouse FreeWare tool and part of the product delivery.
   It attaches/detaches the WasteBasket library WBLIB100 to/from the FUP object file.

5. To check if the bind was successful, use the SHOWLIB utility:
   ```
   [RUN] $SYSTEM.WASTEGHS.SHOWLIB FUP
   ```
   e.g.:
   ```
   $GHS1 WASTEB 415> showlib fup
   SHOWLIB (403) - T7172H06 - (21Sep2010)   System \GINKGO, running NSK H06.19
   Copyright (c) GreenHouse Software & Consulting 1999 .. 2010

   Used Program file template: \GINKGO.$GHS1.WASTEB.FUP
   Used LIB file template:     \*.$*.*.*

   $GHS1.WASTEB.FUP              ->  $GHS1.WASTEGHS.WBLIB100

   Number of object files in error:    0

   Checked files with code 100:        1
   Number of executables:              1
   Total executables with library:     1
   Checked files with code 800:        0
   Number of executables:              0
   Total executables with library:     0
   $GHS1 WASTEB 416>
   ```
   SHOWLIB is a GreenHouse FreeWare tool and part of the product delivery.

6. License this copy of FUP:
   ```
   [RUN] $SYSTEM.SYSnn.FUP LICENSE FUP
   ```

7. Logoff from SUPER.SUPER.

Now this copy of FUP is WasteBasket enhanced, and all files, purged by this FUP, end up in the WasteBasket system.

1. Create a TESTFILE:
   ```
   $GHS1 WASTEB 420> create testfile
   $GHS1 WASTEB 421>
   ```

2. Purge this file using the enhanced FUP:
   ```
   $GHS1 WASTEB 421> run fup purge testfile
                 CODE             EOF      LAST MODIF   OWNER RWEP   TYPE   REC BL
   $GHS1.WASTEB
    TESTFILE                        0           17:35 100,5   UUOO
   PURGE? Y
   $GHS1.WASTEB.TESTFILE PURGED.
   $GHS1 WASTEB 422>
   ```

3. Check that the file is gone:
   ```
   $GHS1 WASTEB 422> fileinfo testfile
   No files match given template(s)
   $GHS1 WASTEB 423>
   ```

4. Run WBCOM to find the file in the WasteBasket:
   ```
   $GHS1 WASTEB 423> wbcom
   WBCOM (100) - T7172H06 - (10Jan2011)   System \GINKGO, running NSK H06.19
   Copyright (c) GreenHouse Software & Consulting 2011
   WasteBasket system OK
   WB> files

   $GHS1.WASTEB

   TESTFILE
   WB>
   ```

5. Execute the INFO command to get more information of this file:
   ```
   WB> info

   $GHS1.WASTEB

                 Code             EOF Last Modification   Owner  RWEP   WBNum
   TESTFILE         0               0 20Jan2011 17:35:10 100,5   UUOO    1AZ
   WB>
   ```

   The file is now in the WasteBasket.

6. To restore is from there, execute the RESTORE command of WBCOM:
   ```
   WB> restore testfile
   $GHS1.WASTEGHS.A0000169 restored to $GHS1.WASTEB.TESTFILE
   WB>
   ```
   This restores the file to the original location, and deletes it from the WasteBasket.

8. Recommended Programs to Enhance

Only programs calling PURGE or FILE_PURGE_ can be enhanced with WasteBasket.

Recommended programs for enhancement are:

- TACL
- FUP
- GreenHouse FreeWare tools MYPURGE, ORPHANS and others.
- GreenHouse product SECOM, FTPSERV-E and others.

To check a program file for using one of the mentioned procedure calls, the delivered utility program WBCHECK can be used.

Simply start it with this command:

```
[run] WBCHECK <filename>
```

where

`<filename>`	defines the file(s) to be checked. Wildcards are supported.

It displays objects, using one of the procedure calls.

9. Wildcard Sort Rules

All entries, supporting wildcards, are sorted internally according to the MCO (Most COmplete) rules:

A file name of

- $A.B.C   is more complete than
- $A.B.?   which is more complete than
- $A.B.*

A file number of

- 100   is more complete than
- 10?   which is more complete than
- *

10. SYSnn change

When the system image is changed, the system programs, residing in $SYSTEM.SYSnn, or changed in $SYSTEM.SYSTEM, and enhanced with WBLIBxxx, have to be re-enhanced to activate the WasteBasket for these new program files.

11. Suggestions

It is highly recommended NOT to enhance the original program files, such as FUP or TACL, but to useance a copy of these files for the WasteBasket enhancement.

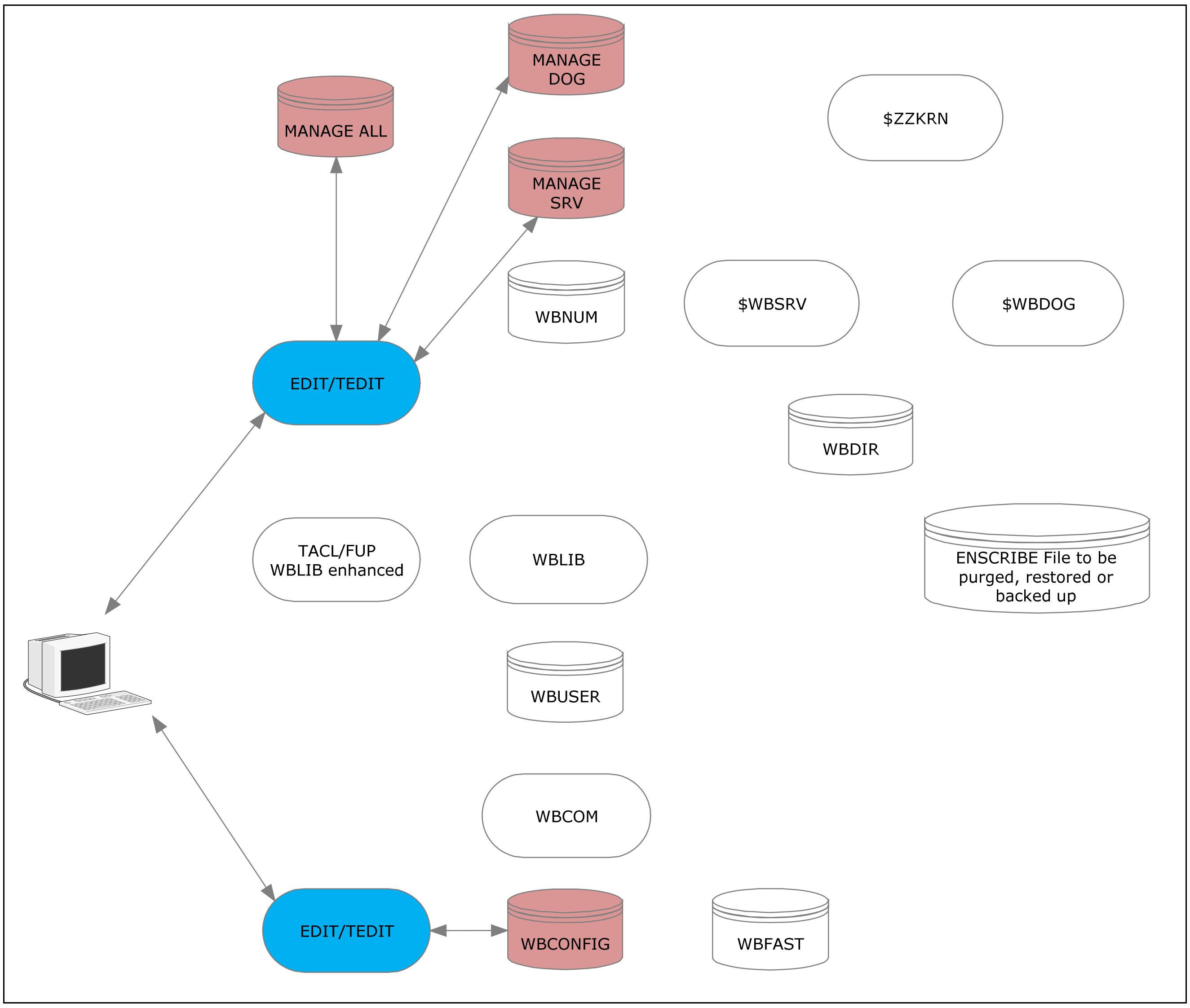
TACL

- Make a copy of TACL in $SYSTEM.SYSnn, and name it e.g. TACLW (TACL WasteBasket).
  Make sure the security settings are identical with the ones of the original TACL.
- Bind WBLIB to TACLW, and keep the original TACL untouched.
- Make TACLW a new service in TELNET, and direct the 6530 emulator to use TACLW instead of TACL.
- Do NOT use TACLW on the console!

FUP

- Make a copy of FUP in $SYSTEM.SYSnn, and name it e.g. FUPO (original FUP). Make sure its security settings are identical with the ones of FUP, and that it is licensed.
- Bind WBLIB to FUP, and keep FUPO untouched.
  This keeps an original FUP as FUPO available in case WBLIB on FUP fails, and causes it to misbehave.

12. Overview Edit Configuration Files



Use the EDIT/TEDIT program to change the WasteBasket configuration files:

- MANAGEALL consists of
  - INFOALL
  - STARTALL
  - STOPALL

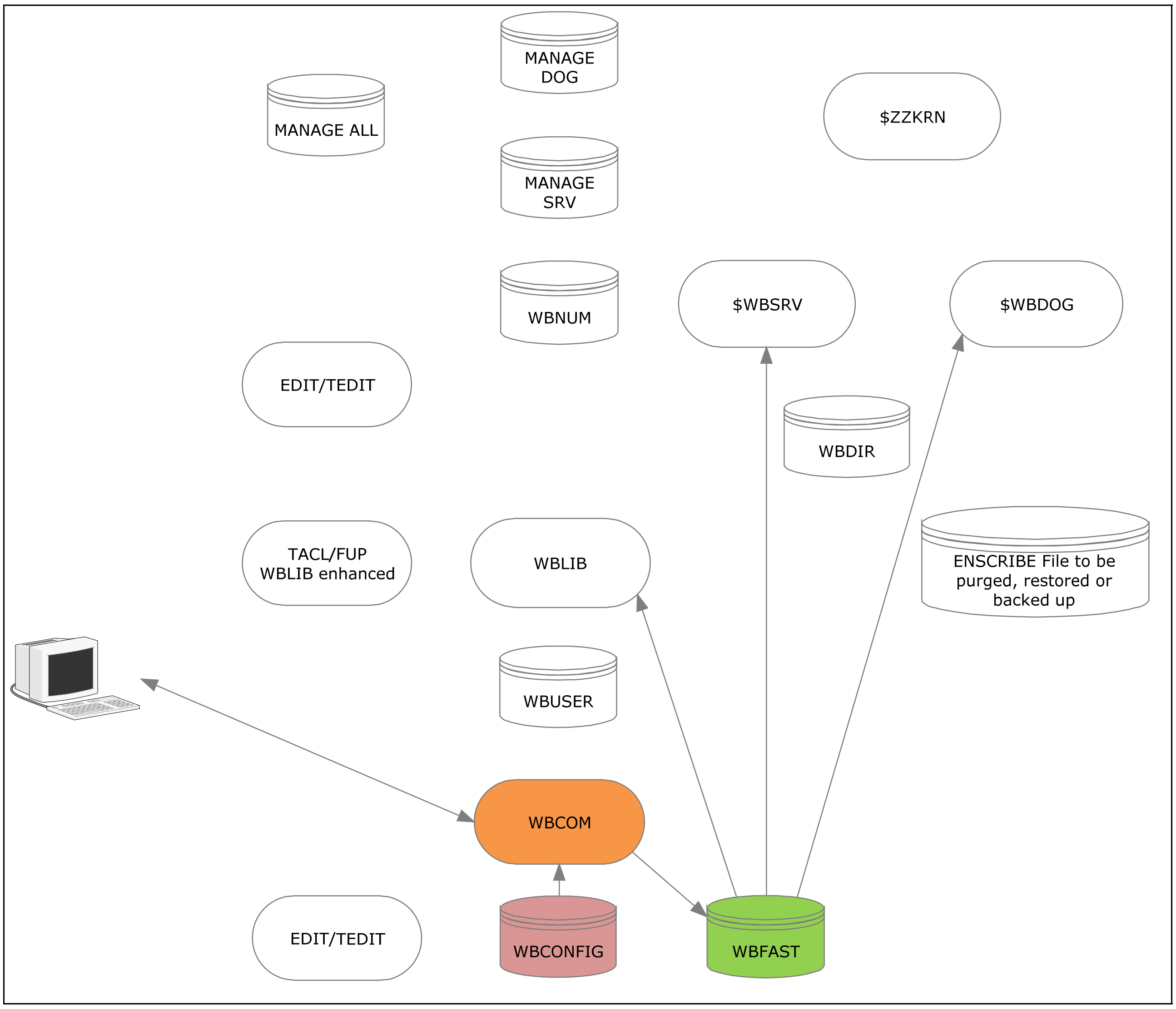
- MANAGEDOG consists of
  - INFODOG
  - STARTDOG
  - STOPDOG

- MANAGESRV consists of
  - INFOSRV
  - STARTSRV
  - STOPSRV

- WBCONFIG
  When WBCONFIG is changed, execute the WBCOM MAKEFAST command to update the WBFAST configuration file which is used by:
  - WBLIB
  - $WBSRV
  - $WBDOG

  WBCONFIG is used by WBCOM.

13. Overview MAKEFAST



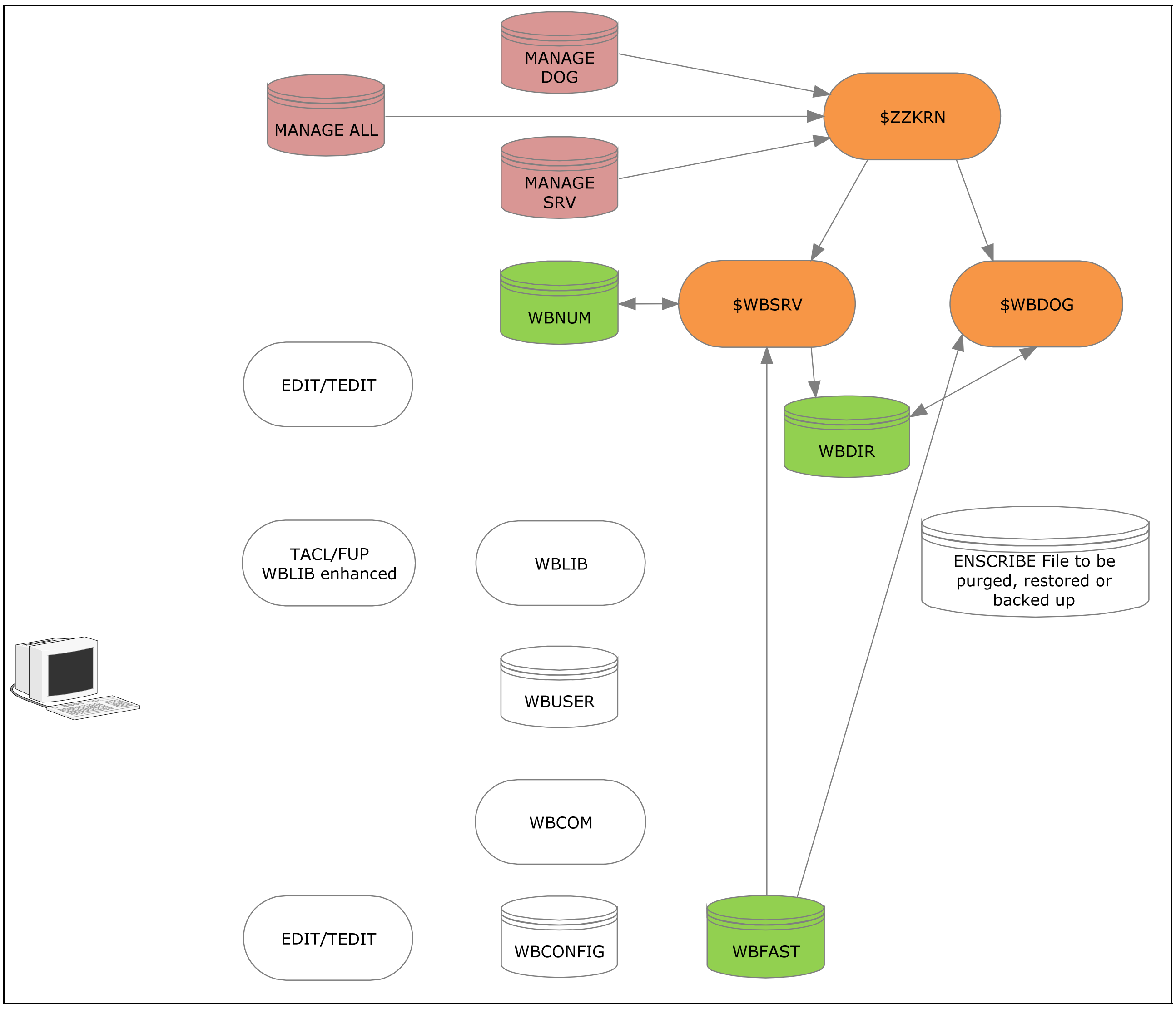
When WBCONFIG is changed, it has to be processed by the MAKEFAST command of WBCOM:
This produces the WBFAST file which holds all relevant configuration attributes.

WBFAST is used by:

- $WBDOG
- $WBSRV
- WBLIB

A change if WBCONFIG and the execution of the MAKEFAST command can be done any time: All depending programs recognize a change, and automatically take it into account. No re-start is needed.

14. Overview start servers



Some of the functions of the WasteBasket are handled by permanent server processes.

$WBDOG

The $WBDOG process keeps track of the files in the WasteBasket system and purges those, which have reached their life time.

$WBSRV

Some of the WasteBasket actions require to be executed by SUPER.SUPER, which is performed in the $WBSRV process. This makes the WBLIB library code a NON PRIV product, and allows all WasteBasket data files to be owned by SUPER.SUPER and secured as tight as possible.

To start, stop and retrieve information from these processes, TACL macros are supplied. To successfully execute them, the user has to be logged on to a SUPER-group member.

$WBDOG reads the WBFAST configuration file, and opens the WBDIR file.

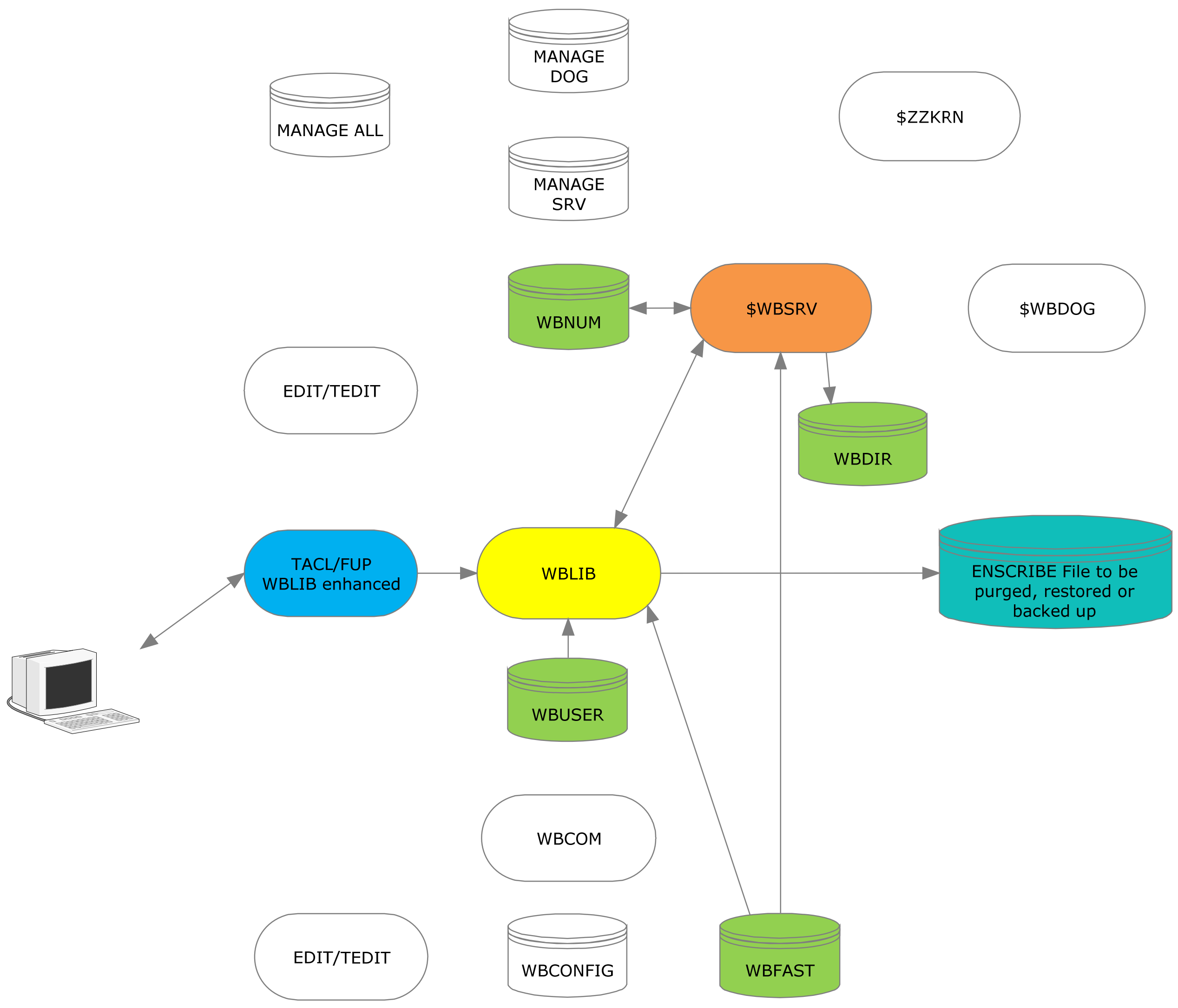
$WBSRV reads the WBFAST configuration file, and opens WBDIT as well as WBNUM.

Both processes are kept available by the $ZZKRN system.

In case $WBDOG is NOT available, the WasteBasket system still functions, but it may happen, that disk volumes runs full.

In case $WBSRV is NOT available, the WBLIB code is no longer able to save files in the WasteBasket: A purged file is gone as it would be without the WasteBasket.

15. Overview PURGE



Purging a file - or set of files - from the system and saving it by the WasteBasket is a more complex task.

The PURGE action ends up in the WBLIB library.

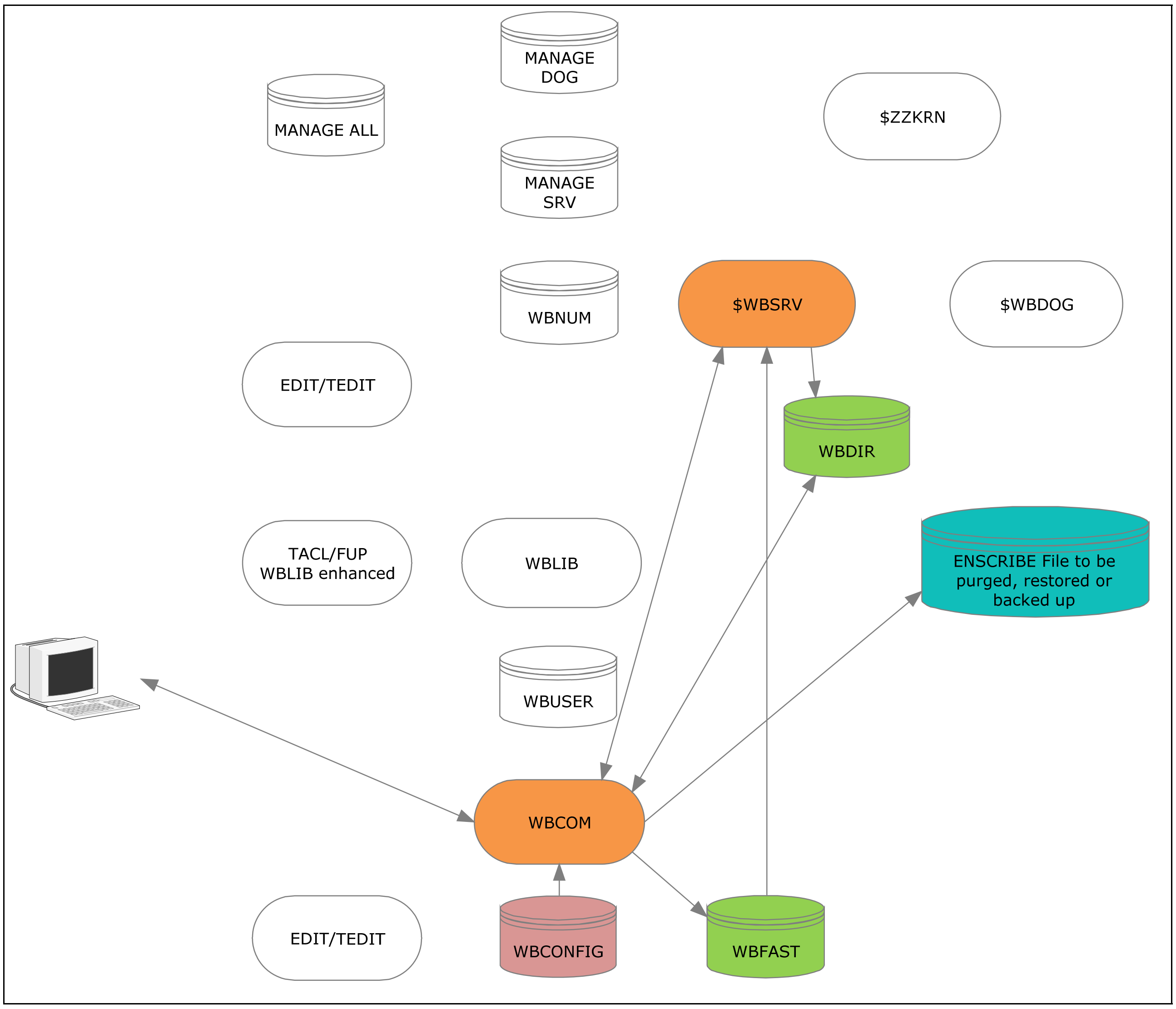
It first checks, if the file in question has to end up in the WasteBasket, or if it has to be purged right away. This is done by consulting the WBUSER file.

When the file is OK to be saved it has to be checked, if the user does have purge access on it. This is done by talking to the $WBSRV process.

When an OK is received from the server, WBLIB asks it for a WasteBasket unique number.

When it is received from the $WBSRV server, WBLIB saves all file attributes, performs the rename, and sends a message to the $WBSRV server to update the WBDIT directory file.

16. Overview RESTORE



Restoring a file - or set of files - from the WasteBasket is done with the WBCOM utility, and the RESTORE command.

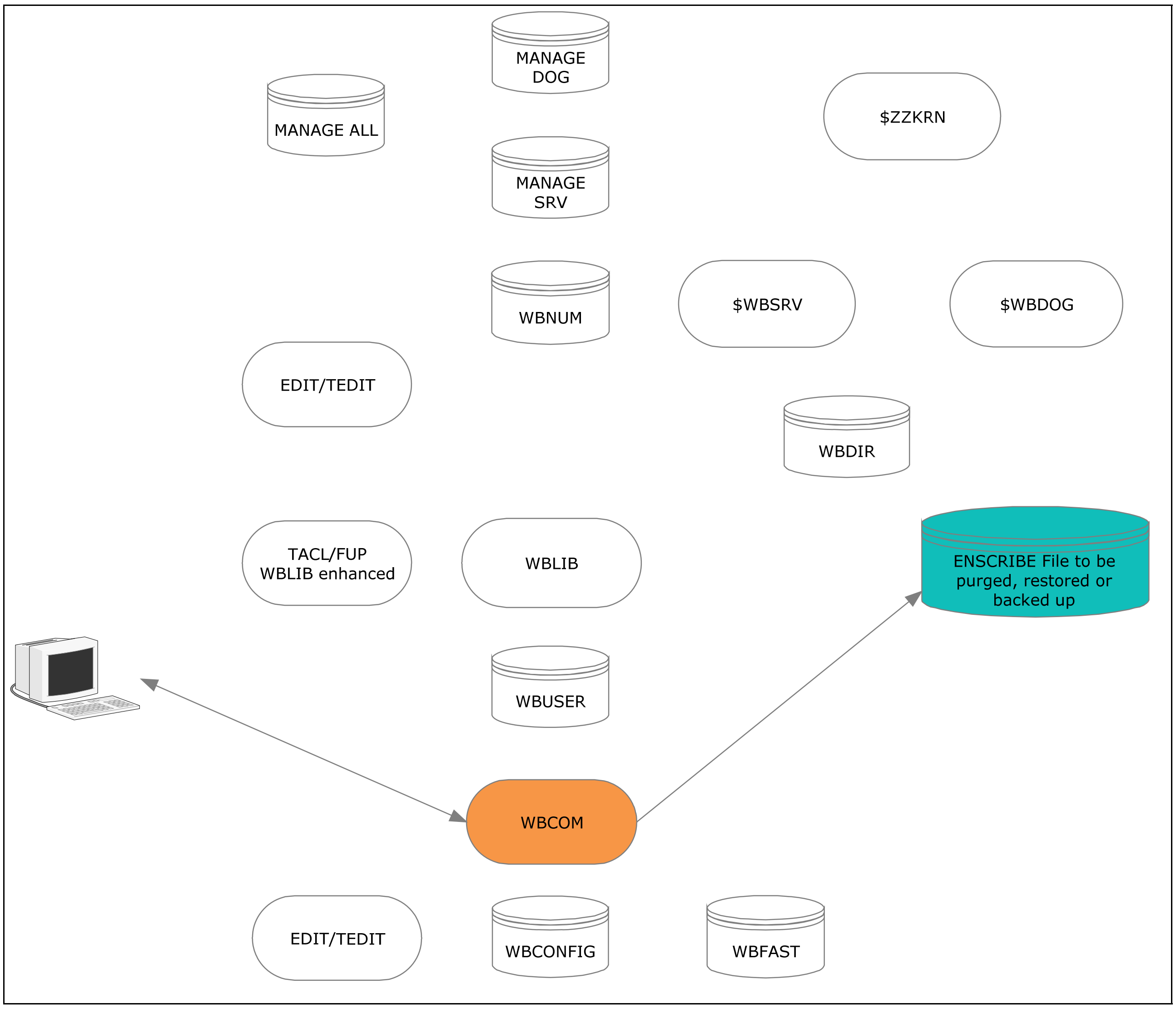
When WBCOM is started, it checks the WBCONFIG files for changes. In case there any, the file WBFAST is checked and re-built.

The first task of WBCOM is to check, if the requested file to be restored is:
- in the Wastebasket
- owned by the WBCOM user or
- managed by the WBCOM user

In case these checks are password OK, WBCOM renames the file from the WasteBasket back to its original place, and restores all file attributes.

Duplicate files in the WasteBasket are handled correctly.

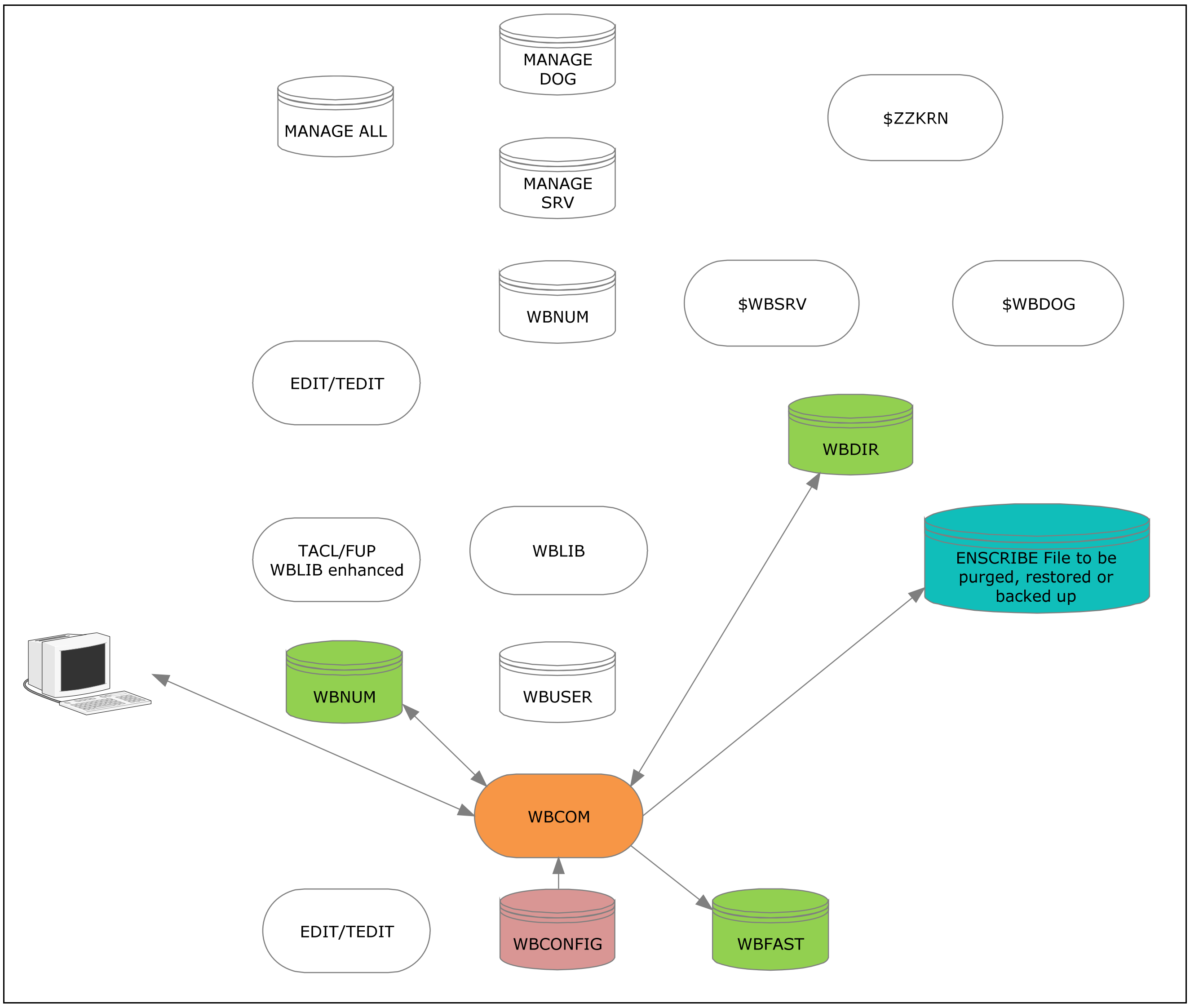
17. Overview REALPURGE



The WBCOM program provides the ability to purge a file from the system WITHOUT saving it in the WasteBasket.

The REALPURGE|REALDELETE commands do this. A file, purged by these commands, is gone.

18. Overview REALPURGE OPEN



The REALPURGE command of WBCOM allows it to even purge an already open file. This is a meaningful option when e.g. replacing an existing program being used with a new version.

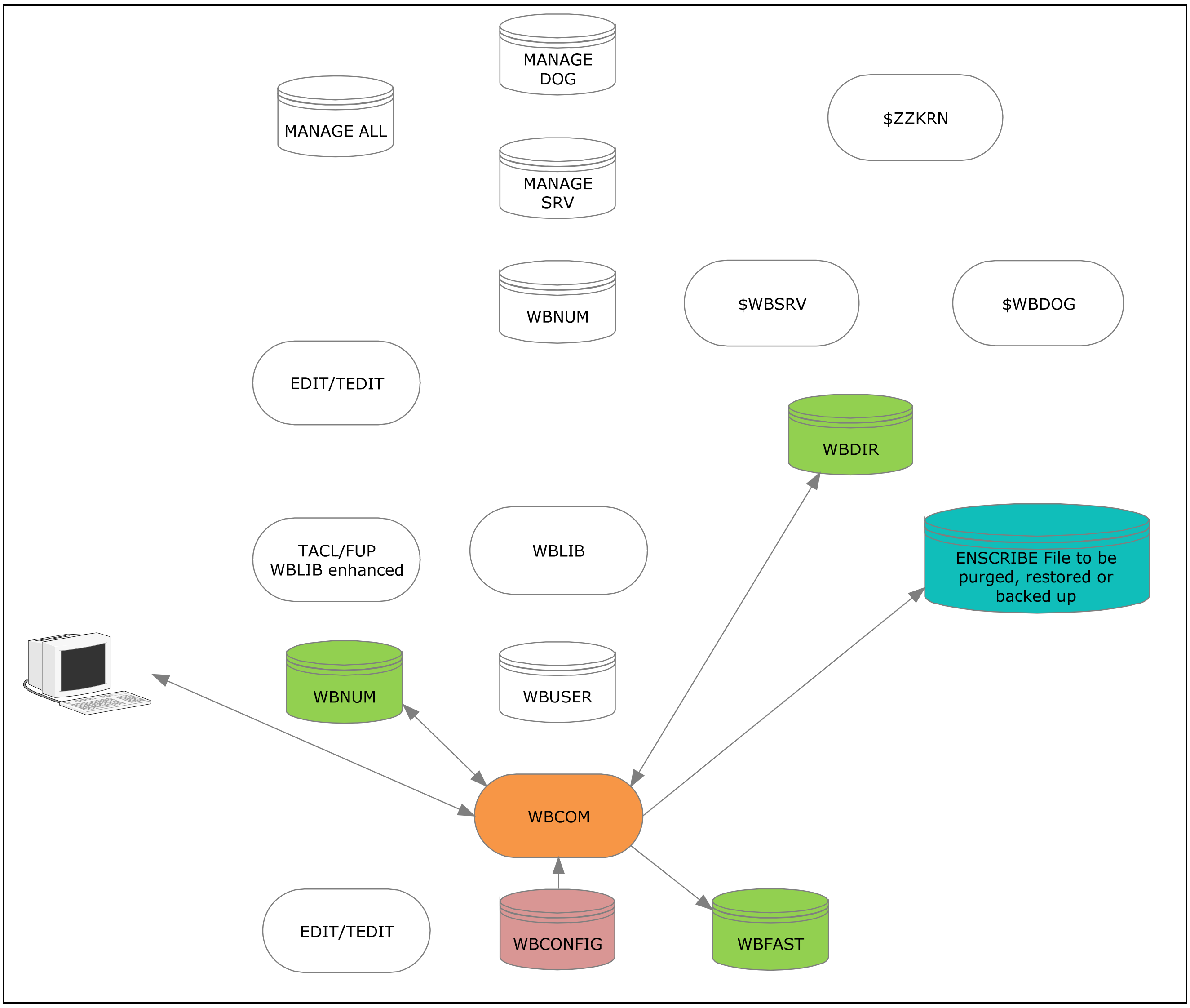
When WBCOM is started, it checks the WBCONFIG files for changes. In case there any, the file WBFAST is checked and re-built.

WBCOM then behaves like the WBLIB library: It collects all file attributes, builds a WBDIR record, enames the open file into the WasteBasket and writes a record into the WasteBasket directory file WBDIR.

This file can NOT be restored. It is automatically purged with the next WBCOM run when it is closed.

The number of open files in the WasteBasket is limited by the volume size.

19. Overview BACKUP



WBCOM allows the user to backup a file into the WasteBasket by keeping the original file intact.
This feature is meaningful e.g. to save a copy of a file being edited before it is changed.

The BACKUP command causes WBCOM to create a copy of the file to be backed up, and to perform a WasteBasket type purge on this copy.

Multiple restores of the same file are supported.

The number of duplicates is limited by the volume size.

20. WBCOM

The WBCOM (WasteBasket Command Interpreter) is the interactive interface to the WasteBasket and allows the user to get access to his purged files.
The owner of a file in the WasteBasket is the user, who purged the file, NOT the original owner of the file!

The following commands are available:

▪ ACCESS	Displays the users PURGE access right on a file/set of files.
▪ ADDFCIGNORE	Adds a File Code to the user specific ignore list.
▪ ADDFNIGNORE	Adds a File Name to the user specific ignore list.
▪ ALTER	Alters the life time of a file in the WasteBasket.
▪ BACKUP	Sends a file to the WasteBasket without purging it.
▪ CLEANUP	Checks and corrects the WasteBasket data base and files.
▪ CLS	Clears the WBCOM 6530 screen.
▪ COMMANDS	Displays all commands; identical with HELP
▪ CONFIG	Displays the current WBCONFIG attributes.
▪ DELETE	Deletes a file from the WasteBasket; identical with PURGE.
▪ DELFCIGNORE	Deletes a File Code from the users ignore list.
▪ DELFNIGNORE	Deletes a File Name from the users ignore list.
▪ EXIT	Terminates the WBCOM session; identical with Ctrl-Y.
▪ FC	The good old Fix Command.
▪ FILEINFO	Displays file information; identical with INFO.
▪ FILES	Displays files from the WasteBasket.
▪ GIVE	Gives a file to a new file owner.
▪ HELP	Displays help: identical with COMMANDS.
▪ INFO	Displays file information; identical with FILEINFO.
▪ LISTOPEN	Lists the open files from the WasteBasket data base.
▪ MAKEFAST	Reads the WBCONFIG configuration file and produces the WBFAST file.
▪ PURGE	Deletes a file from the WasteBasket; identical with DELETE.
▪ REALDELETE	Purges a file from the file system WITHOUT saving is in the WasteBasket.
▪ REALPURGE	Identical with REALDELETE.
▪ REMOVEACL	Removes an ACL from a file in the WasteBasket.
▪ RENAME	Renames a file in the WasteBasket.
▪ RESTORE	Restores a file from the WasteBasket.
▪ STATISTICS	Displays statistics.
▪ SUBVOLS	Displays known subvols.
▪ USERS	Lists all users with files in the WasteBasket
▪ VOLS	Displays available volumes.
▪ VOLUME	Changes the default location.
▪ WHO	Displays the WBCOM user and the current default location.

- All commands can be abbreviated down to the smallest string that makes it a unique abbreviation. e.g. C is not unique, while CLE or CO are.
- Commands and their attributes are NOT case sensitive.
- Command options can be given in any order.

Besides the commands described on the previous page, the following abbreviated commands are hard coded:

F	Identical with FILES
FI	Identical with INFO
V	Identical with VOLUME

21. ACCESS

Checks the PURGE access right of the WBCOM user on a file or set of files.
This command is available to administrators only.

Command syntax is:

```
ACCESS [<file-set>]
```

where

`<file-set>`	defines the GUARDIAN file(s) to be checked. When missing, all files of the current user's location are checked.

e.g.

```
ADDCCESS  A??B
```

22. ADDFCIGNORE

Adds a file code to the list of user specific file codes to be skipped: Files, matching the defined file code, are NOT saved in the WasteBasket.

Command syntax is:

```
ADDFCIGNORE <filecode-template> [[,] <LiT>]
```

where

`<filecode-template>`	is the file code to be individually processed. Valid entries are 0 .. 65535. Wildcards are supported, e.g. 18??8
`<LiT>`	defines the LifeTime of the file. Valid entries are: - IMMEDTATELY = the file is purged right away. This is the default. - DOGTIME = the file gets the default LiT set. - INFINITE = the file is never purged. - 0 .. 16000 = number of days a file should reside in the WasteBasket. When missing, IMMEDIATELY is assumed.

e.g.

```
ADDFCIGNOTE 100, immediately
```

23. ADDFNIGNORE

Adds a file name to the list of user specific file names to be skipped: Files, matching the defined file name, are NOT saved in the WasteBasket.

Command syntax is:

```
ADDFNIGNORE <filename-template> [[,] <LiT>]
```

where

`<filename-template>`	defines the file(s) to be added to the WasteBasket Wildcards are supported.
`<LiT>`	defines the LifeTime of the file. Valid entries are: - IMMEDTATELY = the file is purged right away - DOGTIME = the file gets the default LiT set - INFINITE = the file is never purged - 0 .. 16000 = number of days a file should reside in the WasteBasket. When missing, IMMEDIATELY is assumed.

e.g.

```
ADDFNIGNOTE $*.*.object, immediately
```

24. ALTER

Alters the life time of a file in the WasteBasket.

Command syntax is:

```
DELETE <file-set>|<waste-num> [,] LIFETIME|LIT [nn|DOGTIME|INFINITE]
```

where

`<file-set>`	defines the file(s) to be altered in the WasteBasket Wildcards are supported. e.g.: `ALTER $*.*.ZZSA*`
`<waste-num>`	defines a file to be altered by its WasteBasket number. Wildcards are NOT supported. e.g.: `ALTER 47`
`LIFETIME|LIT`	required key word
`nn`	number of days in the range of 0 .. 16000
`DOGTIME`	Use the value from the WBCONFIG file. This as well is the default
`INFINITE`	The file is allowed to stay forever.

e.g.

```
ALTER secom700.secom, lifetime infinite
ALTER *.*.*,lifetime 2
```

25. BACKUP

Sends a file to the WasteBasket WITHOUT purging it from the system.
This feature is intended to e.g. save a source file before it is changed.
Multiple copies of the same file can be backed up to the WasteBasket without naming problems.

Command syntax is:

```
BACKUP <file-set> [[,]<LiT>]]
```

where

`<file-set>`	defines the GUARDIAN file(s) to be sent to the WasteBasket. Wildcards are supported.
`LiT`	optional; life time of the file in the WasteBasket Valid values are: `nn` number of days in the range of 0 .. 16000 `DOGTIME` Use the value from the WBCONFIG file. `INFINITE` The file is allowed to stay forever. Default is BACKUPLIFETIME defined in WBCONFIG.

e.g.

```
BACKUP secom700.secomsrc, 45
BACKUP *
```

26. CLEANUP

Checks the WasteBasket directory and the WasteBasket files in locations $vol.WASTEGHS for consistency, and deletes invalid entries.

- A WasteBasket file, mentioned in the directory file WBDIR, has to exist in the WasteBasket location $vol.WASTEGHS. In case it does nit, the entry in WBDIR is purged.
- A file in the WasteBasket location $vol.WASTEGHS has to be mentioned in the directory WBDIR. In case it is not, it is purged from the system.

Administration access right is required to execute this command.

Command syntax is:

```
CLEANUP
```

27. CLS

Clear-Screen cleans the 6530 screen by positioning the cursor to the first character in the screen, and erasing the screen buffer.

Command syntax is:

```
CLS
```

Very handy!

28.COMMANDS

Displays all available commands, or detailed help of a defined command.
Identical with the HELP command - implemented to make life easier!

Command syntax is:

```
COMMANDS [<command>]
```

29. CONFIG

Displays the values read from WBCONFIG, all user specific file names and file codes, and run time parameters.

Command syntax is:

```
      CONFIG
```

Example:

```
$GHS1 WASTEB <WB>>> config
Prompt is:                        <WB>>>>
Default life time:                6  days
Default BACKUP life time:         3  days
DogTime:                          12:34
Global     NON waste file names: $DSMSCM.XXXTOKEN.*          immediately
                                  *.OBJECT                    same day
                                  *.XXXXYYYY                  next day
                                  *.ZZBI????                  immediately
                                  *.ZZZZ????                  next day
Global     NON waste file codes: *NONE*
Individual NON waste file names: $*.*.XXXXYYYY               immediately
Individual NON waste file codes: *NONE*
DCOM:                             *NOT* configured
DEALLOCATE:                       Activated
REMOVEACL:                        *NOT* activated
Administrative access rights:     SA.CARL
                                  *
WBLIB File Name:                  $GHS1.LAUNCHER.GHSCLIB
                                  $GHS1.WASTEB.WBLIB800
WBDOG process name:               $WBDOG ($SYSTEM.WASTEGHS.WBDOG, FCode 800)
WBSRV process name:               $WBSRV ($SYSTEM.WASTEGHS.WBSRV, FCode 800)
WBCOM file code:                  100
WasteBasket expires on:           31Dec2100 23:59:59
$GHS1 WASTEB <WB>>>
```

30. DELETE|PURGE

The DELETE|PURGE command deletes a file (or set of files) from the WasteBasket.
Files can be addressed by their regular name, or the WasteBasket number.

Command syntax is:

```
DELETE|PURGE <file-set>|<waste-num> [!]
```

where

`<file-set>`	defines the file(s) to be removed from the WasteBasket Wildcards are supported. e.g.: **`DELETE $*.*.ZZSA*`**
`<waste-num>`	defines a file to be deleted by its WasteBasket number. Wildcards are NOT supported. e.g.: **`DELETE 1A47`**
`[!]`	Directs WBCOM to perform the remove without asking the user for intervention in case file duplicates have to be removed.

e.g.

```
DELETE *
DELETE $*.*.ZZSA* !
DELETE 0A123
```

31. DELFCIGNORE

Deletes a file code from the user specific ignore list.

Command syntax is:

```
DELFCIGNORE <filecode-template> [[,] ALL]
```

where

`<filecode-template>`	is the file code template entry to be deleted from the user specific ignore list. Valid entries are 0 .. 65535. Wildcards are supported, e.g. **`1??`**
`ALL`	Keyword. When present, **`<filecode-template>`** is treated as template, that is: All matching file codes are deleted!

e.g.

```
DELFCIGNOTE 1??,ALL
```

The command:

```
DEFFCIGNORE 1??
```

causes WBCOM to delete the single entry **`1??`**, while the command

```
DEFFCIGNORE 1??,ALL
```

causes WBCOM to delete all entries matching template **`1??`**

32. DELFNIGNORE

Deletes a file name from the user specific ignore list.

Command syntax is:

```
DELFNIGNORE <filename-template> [[,] ALL]
```

where

`<filename-template>`	defines the file name template to deleted from the user specific ignore list. Wildcards are supported. e.g.: **`DELFNIGNORE $*.*.ZZSA*`**
`ALL`	Keyword. When present, **`<filename-template>`** is treated as template, that is: All matching file names are deleted.

e.g.

```
ADDFNIGNOTE $*.*.object
```

The command:

```
DEFFNIGNORE $*.*.o*
```

causes WBCOM to delete the single entry **`$*.*.o*`**, while the command

```
DEFFNIGNORE $GHS*.*.o,ALL
```

causes WBCOM to delete all entries matching template **`$GHS*.*.o`**

33. EXIT

Terminates the interactive WBCOM session.

The Ctrl-Y key combination does the same.

34. FC

The good old “Fix Command” command.
When executed at the WBCOM prompt, the last command is re-displayed for editing.

35. FILEINFO|INFO

Displays detailed information of files in the WasteBasket.
Files can be addressed by their regular name or name template, or the WasteBasket number.

Abbreviated command: FI

Command syntax is:

```
INFO [<file-set>|<waste-num> [,DETAIL][,OWNER [<file-purger>]]
```

where

`<file-set>`	defines the file(s) to be removed from the WasteBasket Wildcards are supported.
`<waste-num>`	defines a file by its WasteBasket number. Wildcards are NOT supported. e.g.: **`DELETE 1R47`**
`DETAIL`	displays detail information of a file in the WasteBasket.
`OWNER <file-purger>`	filters the output by the user, who purged the file(s). A missing **`<file-purge>`** name is filled with the current WBCOM users name.

e.g.

```
INFO
INFO $SYSTEM.SECOM*.*
INFO $*.*.object
```

Example:

```
$GHS1 WASTEB <WB>--> info abcd,detail

$GHS1.WASTEB.WBCOMSRC
    WasteBasket:            $GHS1.WASTEGHS.W00002MV
    Code:                   101
    Type:                   Unstructured
    Owner:                  100,5
    Security:               OO--
    EOF:                    303990
    Creation Time:          15Nov2011 15:54:06
    Last Open:              22Nov2011 15:14:19
    Last Modification:      22Nov2011 15:10:57
    Purge Date:             22Nov2011 15:18:26
    Purger:                 SA.CARL
    Configured life time: 2
    Remaining life time:  Will be purged in 2 days
$GHS1 WASTEB <WB>-->
```

36. FILES

Displays the files of the current WBCOM location, or the files of a given location.

Abbreviated command: F

Command syntax is:

```
       FILES [[<$vol>.]<subvol>]
```

where

<$vol>	defines the volume to be used to display files
<subvol>	defines the subvol to be used to display files

e.g.

```
       FILES
       FILES $SYSTEM.SECOM
```

Example:

```
$GHS1 SECOM700 <WB> >> files

$GHS1.SECOM700

ASD30     ASD30H    ASD30N    BULKSTON  BULKSTOP  G5619646  gard      IMPEX
IMPEXN    MYPERSEC  MYPERSEC  MYPERSEC  MYPERSEC  MYPERSEC  MYPERSEC  MYPERSEC
MYPERSEC  MYPERSEC  SECOMCI   SECOMCIN  SECOMCT   SECOMCTN  XXXXYYYY
$GHS1 SECOM700 <WB> >>
```

- In case the configuration attribute SHOWOWNERONLY is set to ON, only files, purged by the WBCOM user, are displayed.
- Files shown in upper case characters can be managed by the WBCOM user.
- Files shown in lower case characters were NOT purged by the WBCOM user, but someone else. These files can NOT be managed by the WBCOM user.

37. GIVE

Changes the ownership of a file in the WasteBasket.

Command syntax is:

```
        GIVE <file-set>|<waste-num> [,] <owner>
```

where

`<file-set>`	defines the file(s) to be given to a new owner. Wildcards are supported. e.g.: **`GIVE $*.*.ZZSA*,CarlWeber`**
`<waste-num>`	defines a file to be given by its WasteBasket number. Wildcards are NOT supported. e.g.: **`GIVE 4Z47,ghs.christi`**
`<owner>`	Is the new owner of the file in the WasteBasket. This does NOT change the physical owner of the file! <owner> is a name, such as SUPER.SUPER, or CarlWeber, NOT an ID!

e.g.

```
        GIVE *,super.super
```

Changing the waste basket ownership of a file resets the License as well as the PROGID flag!

38.HELP

Displays an overview of all available commands, or command specific help.

Command syntax is:

```
        HELP [<command>]
```

where

`<command>`	is a WBCOM command. When missing, an overview of all available commands is displayed.

e.g.

```
        HELP
        HELP RESTORE
```

Example:

```
$GHS1 WASTEB <WB> >> info *

$GHS1.WASTEB

              Code              EOF Last Modification   Owner  RWEP  WBNum   LiT
OBJECT         100            46160 31Jan2011 10:09:06 100,5   UUOO   5A343   3
PNWASTE        101              838 27Jan2011 17:19:40 100,5   UUOO     1B2   0
WBCOM          100 L         868352 31Jan2011 18:07:32 100,5   OOAO     44C   3
WBLIB          100            68906 27Jan2011 15:19:32 100,5   UUOO     138  Inf
Total EOF: 984.256
$GHS1 WASTEB <WB>>>
```

`Code`
`Last Modification`
`Owner`
`RWEP`
	are file attributes of files in the WasteBasket.

`WBNum`	is the WasteBasekt internal file number; can be used to address a file.
`LiT`	Displays the remaining life time of the file in the WasteBasket - Inf = Infinite = file lives forever and has to be purged manually - a numeric value defines the remaining life time in days. - A zero (0) means, that the file is purged with the next WBDOC run.

39. LISTOPENS

Lists those files from the WasteBasket, which were open while being purged by the REALPURGE command with the OPEN option.

Command syntax is:

```
LISTOPENS
```

Example:

```
$GHS1 WASTEB <WB>>> list
    File in data base               Original file name
$SYSTEM.WASTEGHS.W000026I   ->  $SYSTEM.SYS00.ZZBI0000
$SYSTEM.WASTEGHS.W000026J   ->  $SYSTEM.SYS00.ZZBI0001
$GHS1 WASTEB <WB>>>
```

40. MAKEFAST

This command is used to manually direct WBCOM to:

- Optionally purge the existing $SYSTEM.WASTEGHS.WBFASTG file.
- Optionally create a new $SYSTEM.WASTEGHS.WBFASTG file.
- Read and process the $SYSTEM.WASTEGHS.WBCONFIG file.
- Write the WBCONFIG attributes into the WBFAST file.

Command syntax is:

```
MAKEFAST [!]
```

where

`! (exclamation mark)`	directs WBCOM to purge the existing WBFAST file, and to create a new one.

Example:

```
$GHS1 WASTEB <WB--> makefast !
*** WBFAST purged
*** WBFAST created
*** WBFAST current
$GHS1 WASTEB <WB-->
```

A newly created MAKEFAST files is read by all WasteBasket processes:

- $WBDOG
- $WBSRV
- WBLIB

No re-start is required.

41. REALDELETE|REALPURGE

The REALDELETE|REALPURGE command deletes a file (or set of files) from the file system WITHOUT saving them to the WasteBasket.

The OPEN feature even allows purging open files!

Command syntax is:

```
REALDELETE|REALPURGE <file-set> [,CLP] [,OPEN] [!]
```

where

`<file-set>`	defines the file(s) to be removed from the system. Wildcards are supported. e.g.: `DELETE $*.*.ZZSA*`
`CLP`	When present causes WBCOM to set the Clear-On-Purge attribute on the file(s) to be purged.
`OPEN`	When present directs WBCOM to even purge an OPEN file.
`[!]`	Directs WBCOM to perform the remove without asking the user for intervention in case of the presence of a filename-template.

e.g.

```
REALPURGE *
REALPURGE $*.*.OBJECT,CLP !
```

Restriction: Only NOT exclusively opened files can be purged by the command with the OPEN attribute. e.g. an EDIT type file, being opened for read/write access (this is the normal case), can NOT be purged.
Open files, purged by this command, can NOT be restored with the RESTORE command!

42. REMOVEACL

Removes the SAFEGFUARD ACL from a set of files in the WasteBasket.

Command syntax is:

```
  REMOVEACL <file-set>|<waste-num>
```

where

`<file-set>`	is the name of a file from which the ACL has to be removed. Wildcards are supported.
`<waste-num>`	defines a file by its WasteBasket number. Wildcards are NOT supported. e.g.: **`4Z47`**

In case a file in the WasteBasket is addressed by its internal number, it always has to start with a numeric character. This may be a leading zero, e.g.: **`0AWER3Z`**

43. RENAME

Renames the original name of a file in the WasteBasket.

Command syntax is:

```
  RENAME <file-set1> [,] <file-set2>
```

where

`<file-set1>`	is a file-set to be renamed.
`<file-set2>`	is the name of the target file-set.

Legal file sets are:

- **`$A.B.C -> $D.E.F`**
- **`$A.B.C -> $D.E.*`**
- **`$A.B.* -> $D.E.*`**

44.RESTORE

Restores a file or set of files from the WasteBasket.
Files can be addressed by their regular name or name template, or the WasteBasket number.

Command syntax is:

```
        RESTORE <file-set>|<waste-num>[,VOL <location>][,PURGE][,KEEPWASTE][,NEWEST]
```

where

`<file-set>`	defines the file(s) to be removed from the WasteBasket Wildcards are supported. e.g.: `DELETE $*.*.ZZSA*`
`<waste-num>`	defines a file to be deleted by its WasteBasket number. Wildcards are NOT supported. e.g.: `DELETE 47`
`VOL <location>`	When present defines the location, to which the file has to be restored. This can be any local $VOL.SUBVOL.
`PURGE`	When present causes WBCOM to purge a possibly existing target file.
`KEEPWASTE`	Causes WasteBasket to keep a copy of the restored file in the WasteBasket.
`NEWEST`	Directs WBCOM to restore the most recent version of a file from the WasteBasket.

e.g.

```
        RESTORE MYINFO
        RESTORE ZZSA*
        RESTORE $GHS1.SECOM700.*,NEWEST
```

Example:

```
$GHS1 WASTEB <WB> >> restore wbcom,vol $ghs2.test,purge
$GHS1.WASTEGHS.A0001200 restored to \GINKGO.$GHS2.TEST.WBCOM
1  files restored
$GHS1 WASTEB <WB> >>
```

45. STATISTICS

Displays statistics from the WasteBasket system.

Command syntax is:

```
      STATISTICS [<file-purger>]
```

where

`<file-purger>`	optionally defines the file purger[3], for which statistical information has to be returned. Wildcards are NOT supported.

Example:

```
$GHS1 WASTEB <WB> >> stat
  Vol       Files    UsedBytes    AllocBytes    %Free   %WB
$GHS1        702      96116374     140019712     18     0.19
$DSMSCM      179      59646418      63938560     51     0.08
$GHS2          8         53904        327680     71     0.00
$SYSTEM       32      14853744      15466496     14     0.02
--------------------------------------------
             921     170670440     219752448


Summary for user SA.CARL:
  Vol       Files    UsedBytes    AllocBytes
$GHS1        649      93871076     135432192
$DSMSCM      179      59646418      63938560
--------------------------------------------
             828     153517494     199370752

WBSRV CPU busy:                 00:00'00,069.654
Number of STATISTIC calls:      2
          SNUM generations:     98
          WD writes:            98
          PURGE access checks:  97
Next WasteBasket file number:       3PX
$GHS1 WASTEB <WB> >>
```

[3] A “File Purger” is the user, who purged a file. He automatically becomes the owner of the file in the WasteBasket.

46.SUBVOLS

Displays known subvols from the WasteBasket. It works exactly like the FUP SUBVOLS command.

Command syntax is:

```
      SUBVOLS [[$vol.]subvol]
```

Example:

```
$GHS1 SECOM700 <WB>>> subvols
$GHS1
    CARL      SECOM700  TOKEN     WASTEB
$GHS1 SECOM700 <WB>>>
```

47. USERS

Lists users, having files in the WasteBasket, along with the number of files and the user specific space.

Command syntax is:

```
USERS [<user>]
```

where

`<user>`	defines the user(s) to be displayed. Wildcards are supported.

Example:

```
$GHS1 SECOM700 8>   wbcom users
            UsersName                Files     DiskSpace
CW.SUPER                                 8        53.904
SA.CARL                                828   153.517.494
SUPER.SUPER                             85    17.099.042
$GHS1 SECOM700 9>
```

48. VOLS

Displays all available volumes, and optional volume details.

Command syntax is:

```
VOLS [<$vol>][,DETAIL]
```

where

<$vol>	defines a volume template of the volumes to be displayed.
DETAIL	optional keyword; when present displays volume data, such as %free.

e.g.

```
VOLS $GHS*,DETAIL
```

Example:

```
$GHS1 WASTEB 76> wbcom vols *,detail
Volume      Total       Free    %Free   Count        Biggest
$DSMSCM     73406     37963.61    51      497       22265.86
$GHS1       73406     14337.25    19     4185        4006.72
$GHS2       73406     52284.63    71      226       44688.65
$SYSTEM     73406     10516.03    14      984        4596.72
$GHS1 WASTEB 77>
```

49. VOLUME

Changes the WBCOM default location.

Abbreviated command: V

Command syntax is:

```
        VOLUME [<$vol>.<subvol>|<$vol>|<subvol>]
```

where

<$vol>	defines the volume to be used in the default location.
<subvol>	defines the subvol to be used in the default location

e.g.

```
        VOLUME
        VOLUME $GHS1
        VOLUME SECOM700
        VOLUME $GHS2.CHRISTI
```

50. WHO

Displays the current WBCOM users session attributes.

Command syntax is:

```
WHO
```

Example:

```
$GHS1 WASTEB <WB> >> who
WasteBasket DB version: 0
You are user:           SA.CARL
Current location is:    $GHS1.WASTEB
$GHS1 WASTEB <WB> >>
```

51. WBCONFIG

The WasteBasket behavior can be configured to some extent.
The attributes are stored in an EDIT type file named$ $SYSTEM.WASTEGHS.WBCONFIG.
Use EDIT/TEDIT to change it according to your needs.

These attributes can be configured:

ADMIN	Allows the definition of administrators. The user, who purged a file, becomes the owner of the file. A file owner has administrative access rights on his files. In addition, up to 10 administrators can be defined as owners of purged files. An entry consists of the - key word ADMIN, followed by - the administrators name, or name template, followed by - the name of the "file purger", or a name template. All names support wildcards. **`e.g.   ADMIN  GHS.CARL  GHS.*`** This entry allowed GHS.CARL to access all files, owned by member of the GHS-group. SUPER.SUPER always is an administrator. **Default is: No administrators are defined.**
BACKUPLIFETIME	Defines the default life time of a file that was sent to the WasteBasket by the BACKUP command. Valid values are: 0 .. 16000. **Default is 2 days.**
DCOM[4]	Configures WBDOC to start DCOM when cleaning the WasteBasket. Valid entries are ON|TRUE or OFF|FALSE. **Default is: OFF|FALSE.**
DEALLOCATE	When set to ON|TRUE causes WBLIB to perform a de-allocate of empty allocated extents when mowing a file to the waste basket! Valid entries are ON|TRUE or OFF|FALSE. **Default is: OFF|FALSE.**
DEFAULTLIFETIME	Defines the default life time of files in the WasteBasket. Valid values are: 0 .. 16000. **Default is: 6 days.**
DOGTIME	Defines the hour and minute of a day, when $WBDOG should perform the clean-up. Time is in 24 hour notation. 00:00 midnight 12:00 noon 14:40 2:40 PM 07:00 7:00 AM **Default DOGTIME is: 5 minutes past Midnight (00:05)**

[4] Tandem highly recommends being VERY carefully with this feature!

`NONWASTEFILE`	File names, matching the template, are NOT saved to the WasteBasket. These entries may have a life time defined: - A NOT defined life time causes WasteBasket to purge the file right away. - A life time of 0 means, that the file is purged by WBDOG with the next run. - Any other number in the range of 1 .. 16,000 defines the days, the file in question should persist. Entries are NOT case sensitive. Wildcards are supported. Up to 45 templates can be configured. **Default is: NO file names are configured.**
`NONWASTEFILECODE`	Files with file codes, matching the defined one(s), are NOT saved to the WasteBasket. These entries may have a life time defined. Wildcards are NOT supported. Up to 45 file codes can be configured. **Default is: NO file codes are configured.**
`PROMPT`	Defines the WBCOM prompt. Up to 8 bytes are allowed. The final prompt is composed of: - the current users location - the defined prompt e.g.: `$GHS1 WASTEB <WB-->` **Default of defined prompt is: "<WB-->"**
`REMOVEACL`	Because WasteBasket does not perform a real purge an a file, but a rename, an optionally existing SAFEGUARD ACL stays intact. WasteBasket can be directed to purge the ACL. Valid entries are: ON|TRUE and OFF|FALSE. **Default is: OFF|FALSE**
`SHOWOWNERONLY`	Directs WBCOM to display all files, or only those, purged by the WBCOM user. Valid entries are ON|TRUE or OFF|FALSE. **Default is: ON|TRUE.**
`WBDOGNAME`	Defines the name of the WBDOG process. This name has to be identical with the one used in the STARTDOG TACL Macro. **Default is: $WBDOG**

`WBLIBFILE`	Defines up to three file name of the WBLIB file.
	e.g.: `WBLIBFILE $SYSTEM.WASTEGHS.WBLIB`
	`WBLIBFILE $GHS1.LAUNCHER.GHSCLIB`
	`WBLIBFILE $SYSTEM.WASTEGHS.WBLIB800`
	Default is $SYSTEM.WASTEGHS.WBLIB

`WBSRVNAME`	Defines the name of the WBSRV process. This name has to be identical with the one used in the STARTSRV TACL Macro. **Default is: $WBSRV**

When the changes are made, run the WBCOM program: It processes the changes and makes them available to WBLIB, and the $WBDOG and $WBSRV server programs in the WBFAST file.

Program re-starts are NOT necessary.

When WBCOM is executed, the changes are taken into account right away by:

- $WBDOG with the next run
- $WBSRV with the next message
- WBLIB with the next PURGE operation

The active settings can be displayed with the **`WBCOM CONFIG`** command:

```
$GHS1 SECOM700 <WB>--> config
Prompt is:                        <WB>-->
Default life time:                6  days
Default BACKUP life time:         3  days
DogTime:                          12:34
Global     NON waste file names:  $DSMSCM.XXXTOKEN.*          immediately
                                  *.OBJECT                    same day
                                  *.XXXXYYYY                  next day
                                  *.ZZBI????                  immediately
                                  *.ZZZZ????                  next day
Global     NON waste file codes:  *NONE*
Individual NON waste file names:  $*.*.XXXXYYYY               immediately
Individual NON waste file codes:  *NONE*
DCOM:                             *NOT* configured
DEALLOCATE:                       Activated
REMOVEACL:                        *NOT* activated
Administrative access rights:     SA.CARL
                                  *
WBLIB File Name:                  $GHS1.LAUNCHER.GHSCLIB
                                  $GHS1.WASTEB.WBLIB800
WBDOG process name:               $WBDOG ($SYSTEM.WASTEGHS.WBDOG, FCode 800)
WBSRV process name:               $WBSRV ($SYSTEM.WASTEGHS.WBSRV, FCode 800)
WBCOM file code:                  100
WasteBasket expires on:           31Dec2100 23:59:59
$GHS1 WASTEB <WB>-->
```

52. WBDOG

The WasteBasket product is not a surrogate for BACKUP and RESTORE, but a tool to make life easier on the Tandem platform for system managers, developers, operators, and users with system access.
Purging a file no longer results in a real loss of it, but a purged file can be restored from the WasteBasket environment.

The number of purged files and their required disk space may be quite large. To prevent the WasteBasket from filling up disk volume it does make sense to automatically purge files from the Wastebasket, when they reached a given age, e.g. 6 days.

This is done by the WBDOG process.
It should be named $WBDOG and controlled by $ZZKRN as a persistent process.

The related TACL Macro files to manage the $WBDOG process:

- INFODOG
- STARTDOG
- STOPDOG

are part of the delivery.

A SUPER-Group user is required to successfully execute these files.

When WBCOM is started it checks, if $WBDOG is already running. In case it is NOT, a warning is displayed.

Besides cleaning-up the WasteBasket, WBDOG can start DCOM to compress a volume. This is controlled by the DCOM entry in WBCONFIG.

Start WBDOG

1. Logon to SUPER.xxx[5].

2. Execute the STARTDOG TACL Macro:
   ```
   [run] STARTDOG
   ```

3. Check if the start was successful by executing the TACL Macro INFODOG:

17. Logoff from SUPER.xxx.

Once WBDOG is successfully attached to $ZZKRN, no further actions are needed.

Stop WBDOG

1. Logon to SUPER.xxx.

2. Execute the STOPDOG TACL Macro:
   ```
   [run] STOPDOG
   ```

3. Check if the stop was successful by executing the TACL Macro INFODOG:

18. Logoff from SUPER.xxx.

Info WBDOG

1. Execute the INFODOG TACL Macro:
   ```
   [run] INFODOG
   ```

Besides using WBDOG as the automatic WasteBasket watch dog, it can be started interactively to perform the clean-up task right away. In this case DCOM is NOT started.

[5] SECOM users can create a command that performs the ID switch without the need to know the password of the SUPER-Group user

53. WBSRV

The WasteBasket WBLIB Library uses some system features making it necessary to license it. To avoid licensing WBLIB, these functions are transferred into a $ZZKEN controlled process named $WBSRV.

The $WBSRV process handles these tasks. It:

1. Creates a monotone numbers to be used to name the files in the WasteBasket system.
2. Updates the WBDIT directory file.
3. Checks the purge access right of a user on a file.
4. Collects statistical information.

The WBSRV process should be named $WBSRV and executed as a $ZZKRN controlled persistent process.

The related TACL Macro files to manage the $WBSRV process:

- INFOSRV
- STARTSRV
- STOPSRV

are part of the delivery.

A SUPER-Group user is required to successfully execute these files.

Start WBSRV

4. Logon to SUPER.xxx[6].

5. Execute the STARTSRV TACL Macro:
```
[run] STARTSRV
```

6. Check if the start was successful by executing the TACL Macro INFOSRV:

19. Logoff from SUPER.xxx.

Once WBSRV is successfully attached to $ZZKRN, no further actions are needed.

Stop WBSRV

4. Logon to SUPER.xxx.

5. Execute the STOPSRV TACL Macro:
```
[run] STOPSRV
```

6. Check if the stop was successful by executing the TACL Macro INFOSRV:

20. Logoff from SUPER.xxx.

Info WBSRV

2. Execute the INFOSRV TACL Macro:
```
[run] INFOSRV
```

When $WBSRV is NOT running, the WasteBasket system does NOT catch and store files being purged, but a PURGE works as it would do without having the WasteBasket installed.

[6] SECOM users can create a command that performs the ID switch without the need to know the password of the SUPER-Group user

54. Q & A

Q: Which file codes are supported by WBLIB?

A: WBLIB comes in these file codes:
100 = executable on K-, S-, Itanium and Blade systems.
800 = executable on Itanium and Blade systems.
The file code of the program, that gets WBLIB attached, defines the file code of WBLIB to be used.
File code 700 is not supported, because the pTAL compiler does not support the procedure call: File_SetLastError_

Q: Which file attributes does WasteBasket restore along with the original file?

A: ALL file attributes, including the AUDIT, CLP, PROGID, LICENSE and TRUST, are restored for the file. This as well implies all file related time attributes.
In case a file is NOT restored to its original place, or in case the KEEPWASTE attribute is given when restoring a file, the files creation time stamp changes.

Q: What happens to SAFEGUARD ACLs?

A: Because WasteBasket does not really purge a file but renames it to a new location, the ACL moves along with the rename. WasteBasket can be directed to delete the ACL of purged files by activating the attribute REMOVEACL in WBCONFIG.

Q: Who has access to files in the WasteBasket?

A: The user, who purged a file, is the owner of the file, NOT the original file owner!
e.g. when Alias user CarlWeber successfully purged a file belonging to 13,4, then CarlWeber is allowed to manage the file in the WasteBasket, not the GUARDIAN user with the ID 13,4 or related Alias users!

Q: Does have SUPER.SUPER all rights in WasteBasket?

A: Yes - it does!

Q: Can files be restored to a remote system?

A: No. Restoring a file to a remote system is NOT supported.

Q: Some files, such as ZZBI-files, do not show up in the WasteBasket. What's going wrong here?

A: Check the configuration file WBCONFIG: Possibly these files are excluded from being saved by the WasteBasket system.

Q: Is it possible that the WasteBasket causes an error: Disk is full (43)?

A: No - because a PURGE results in a RENAME, which does NOT consume additional disk space.
BUT: Because it is a rename, the disk space of the file is still allocated!
A file create may end up in error 43!

Q: What happens with bad files, e.g. such returning error 59?

A: A file, that cannot be renamed, but returns an error > 9 to the File_Rename_ operation, is NOT saved to the WasteBasket, but permanently purged from the file system.

Q: What happens to a file when the user has no purge access rights?

A: The user gets an error 48 (security violation) and the file stays in place.
This is the normal behavior of GUARDIAN.

Q: The STAT command shows some WasteBasket occupied space for a volume, while there are no files.

A: When the SHOWOWNERONLY parameter is activated, only files, owned by the WBCOM user, are displayed. In case there are files in the WasteBasket NOT belonging to the current user, these files are NOT displayed, but they are taken into account by the statistic.

Q: Which time stamps are saved?

A: The wastebasket saves these two time stamps:
- Last Open
- Last Modification
In addition, the creation time stamp is unchanged, when the file is restored to the volume it was purged from.
In case it is restored to a different volume, or in case the KEEP attribute is given, the creation timestamp is changed to the RESTORE time.

Q: Is it possible to purge an OPEN file?

A: The PURGE operation does NOT support a purge of an open file, nor does a WBLIB enhanced program.
But the WBCOM utility allows the purge of an open file with the REALPURGE command and the OPEN attribute.

Q: How many files can WasteBasket manage?

A: The number of manageable file is: 21.767.823.360. This is hopefully sufficient for even huge systems with some hundred volumes.
The internal and external representation of the files in the WasteBasket system is based on numbers computed to the base of 36 in the range of: 0000000 to 9ZZZZZZ.

Q: What happens when the $WBSRV server is not accessible?

A: In this case WBLIB does NOT save a purged file, but performs an ordinary purge.

55. WBCONFIG configuration file

The WBCONFIG file resides in location $SYSTEM.WASTEGHS.
It can be easily changed with EDIT/TEDIT.
The file below is an example!

```
!
!                    WBCONFIG - WasteBasket Configuration
!                    ====================================
!                                Version 200
!                                -----------
!                                 08Nov2011
!
!
!
! WasteBasket configuration file.
! This file has to be named: $SYSTEM.WASTEGHS.WBCONFIG.
! The location and name are hard coded!
! WBCONFIG is executed by WBCOM, and translated into $SYSTEM.WASTEGHS.WBFAST.
! WBCOM, WBDOG and WBLIB take care of this file, and re-load it automatically
! when a change is discovered.
!
! Creation:     08Nov2011, CW
! Change:
!
! Change:
!
! -----------------------------------------------------------------------------

! ADMIN
! -----
! Up to 10 administrators along with the file purger names can be defined.
! Both entries have to be names, or name templates. IDs are NOT supported.
! SUPER.SUPER is always an administrator.
! Typical entries may be:
!
!    SUPER.*        GHS.*
!    CarlWeber      SA.*
!    GHS.MANAGER    GHS.*
!
ADMIN sa.carl *
! -----------------------------------------------------------------------------

!
! BACKUPLIFETIME
! --------------
! Defines the default life time of a file that was sent to the WasteBasket
! by the BACKUP command.
! Default is 2 days.
!
BACKUPLIFETIME   2
! -----------------------------------------------------------------------------

!
! DCOM
! ----
! WBDOC can be directed to start DCOM after cleaning up the
! WasteBasket. DCOM is started with a priority of 10.
! Valid entries are: ON/TRUE and OFF/FALSE
! Default is: OFF/FALSE
!
DCOM  OFF
! -----------------------------------------------------------------------------
!
```

```
! DEALLOCATE
! ----------
! WBLIB can be directed to de-allocate all unused extents of a file
! before it is moved to the WasteBasket. This saves disk space.
! Valid entries are: ON/TRUE and OFF/FALSE.
! Default is: FALSE
!
DEALLOCATE  OFF
! ---------------------------------------------------------------------------

!
! DEFAULTLIFETIME
! ---------------
! Defines the default life time of files in the WasteBasket used by
! WBDOG. The default time is 6 days.
!
DEFAULTLIFETIME 6
! ---------------------------------------------------------------------------

!
! DOGTIME
! -------
! Defines the time when WBDOG should perform the cleanup.
! Time is in 24 hour notation.
! 00:00 = midnight
! 12:00 = noon
! 14:40 = 2:40 PM
!  7:00 = 7:00 AM
!
DOGTIME   00:05
! ---------------------------------------------------------------------------

!
! NONWASTEFILE
! ------------
! May be some files should not be caught by the WasteBasket, but
! purged right away, or other files should be caught, by get a
! specific LifeTime (LiT).
! This is supported here:
! - Up to 45 templates can be configured.
! - file names allow wildcards; system name is NOT required
! - LifeTime (LiT) ma have these entries:
!   empty - the file is NOT caught by the WasteBasket,
!           but purged right away.
!   0     - file is deleted from the WasteBasket with the
!           next WBDEG run.
!  >1     - file lives at least n more days.
!
!                  FileNameTemp    LiT
!----------------------------   -----
NONWASTEFILE       *.object
NONWASTEFILE       *.zzbi????       0
NONWASTEFILE       *.zzzz????       0
! ---------------------------------------------------------------------------
```

```
!
! NONWASTEFILECODE
! ----------------
!
! May be files with a specific filecode should not be caught
! by the WasteBasket, purged right away.
! Up to 45 file codes can be configured.
!
!                  FileNum     LiT
!-------------------------  -----
! NONWASTEFILECODE   0          0
! NONWASTEFILECODE   7172       0
! NONWASTEFILECODE   18248      0
! -----------------------------------------------------------------------------

!
! PROMPT
! ------
! User defined prompt; up to 8 bytes
! The displayed prompt is build of:
! 1. The users current location in format: $VOL SUBVOL
! 2. The defined prompt.
!    The prompt is surrounded by blanks.
!e.g. A defined promps of: "<WB>>>"
!     is displayed as: "$VOL SUBVOL <WB>>> "
!
PROMPT  <WB-->
! -----------------------------------------------------------------------------

!
! REMOVEACL
! ---------
! Because WasteBasket does not perform a real purge an a file, but a rename,
! an optionally existig SAFEGUARD ACL stays intact.
! WasteBasket can be directed to purge the ACL.
! Valid entries are: ON|TRUE and OFF|FALSE.
! Default is:  OFF
!
REMOVEACL  OFF
! -----------------------------------------------------------------------------

!
! SHOWOWNERONLY
! -------------
! WBCOM displays all files in upper case characters when the WBCOM user is
!  - SUPER.SUPER, or an ALias with the ID of 255,255
!  - is the user who purged the file.
! All other files are displyed in lower case characters.
!
! SHOWOWNERONLY causes WBCOM to suppres all lower case file names.
! Valid entries are: ON/TRUE and OFF/FALSE.
!
! Default is: ON/TRUE
!
SHOWOWNERONLY ON
! -----------------------------------------------------------------------------
```

```
!
! WBDOGNAME
-----------
! Defines the name of the $WBDOG server.
! Default is:  $WBDOG
!
WBDOGNAME  $WBDOG
! ----------------------------------------------------------------------------

!
! WBLIBFILE
! ---------
! Defines up to three file name of the WBLIB file.
! Default is $SYSTEM.WASTEGHS.WBLIB
!
WBLIBFILE  $SYSTEM.WASTEGHS.WBLIB
WBLIBFILE  $SYSTEM.WASTEGHS.WBLIB100
WBLIBFILE  $SYSTEM.WASTEGHS.WBLIB800
! ----------------------------------------------------------------------------

!
! WBSRVNAME
-----------
! Defines the name of the $WBSRV server.
! Default is:  $WBSRV
!
WBSRVNAME  $WBSRV
! ----------------------------------------------------------------------------

! GreenHouse Software & Consulting, 08Nov2011 --------------------------------
```

56. Adding TACLW as service to TELNET

To add a new service to the TELNET services, SCF has to be used:

```
$GHS1 LOGIN 90> scf
SCF - T9082G02 - (30JUN97) (14MAY97) - 05/26/99 16:38:59 System \BEECH
Copyright Tandem Computers Incorporated 1986 - 1996
(Invoking \BEECH.$GHS1.SECOM.SCFCSTM)
 \BEECH $GHS1.LOGIN 1->
```

Attach SCF to the TELSERV process

It may be necessary to enhance all TELSERV instances.

```
 \BEECH $GHS1.LOGIN 1->assume process $ztn01
 \BEECH $GHS1.LOGIN PROCESS $ZTN01 2->
```

Display the actually defined services

```
\BEECH $GHS1.LOGIN PROCESS $ZTN01 2->info service *

TELSERV Info SERVICE

Name        *Type         *Subtype  *Access *Display *Program
TACLH       CONVERSATION  DYNAMIC   ALL     ON       $SYSTEM.SYSTEM.TACLH
TACL        CONVERSATION  DYNAMIC   ALL     ON       $SYSTEM.SYSTEM.TACL
ZVTL        VTL           STATIC    N/A     OFF      N/A
ZTELNET     CONVERSATION  DYNAMIC   N/A     OFF      N/A
ZBLOCK      BLOCK         STATIC    N/A     OFF      N/A
ZCONV       CONVERSATION  STATIC    N/A     OFF      N/A
ZPRINT      PRINT         STATIC    N/A     OFF      N/A
ZSPI        SPI           STATIC    N/A     OFF      N/A
 \BEECH $GHS1.LOGIN PROCESS $ZTN01 3->
```

Add TACLW as a new service

```
\BEECH $GHS1.LOGIN PROCESS $ZTN01 3-> ADD SERVICE TACLW &
,TYPE CONVERSATION &
,SUBTYPE DYNAMIC &
,ACCESS ALL &
,DISPLAY ON &
,PROGRAM $ghs1.system.taclw
\BEECH $GHS1.LOGIN PROCESS $ZTN01 4->
```

Check the ADD

```
 \BEECH $GHS1.LOGIN PROCESS $ZTN01 4->info service *

TELSERV Info SERVICE

Name         *Type          *Subtype   *Access  *Display  *Program
TACL         CONVERSATION   DYNAMIC    ALL      ON        $SYSTEM.SYSTEM.TACL
TACLW        CONVERSATION   DYNAMIC    ALL      ON        $SYSTEM.SYSTEM.TACLW
ZVTL         VTL            STATIC     N/A      OFF       N/A
ZTELNET      CONVERSATION   DYNAMIC    N/A      OFF       N/A
ZBLOCK       BLOCK          STATIC     N/A      OFF       N/A
ZCONV        CONVERSATION   STATIC     N/A      OFF       N/A
ZPRINT       PRINT          STATIC     N/A      OFF       N/A
ZSPI         SPI            STATIC     N/A      OFF       N/A
 \BEECH $GHS1.LOGIN PROCESS $ZTN01 5->
```

Exit SCF

```
 \BEECH $GHS1.LOGIN PROCESS $ZTN01 5-> exit
$GHS1 LOGIN 91>
```

57. STARTDOG TACL Macro

Make sure the

- red marked entries point to location $SYSTEM.WASTEGHS.
- green marked name is identical with the WBDOGNAME entry in WEBCONFIG.

```
?TACL Macro
==
== TACL Macro the install WBDOG as persistent processes.
== Adjust the program names and locations to your environment.
== 04Nov2011, CW
==
#FRAME
#PUSH #INLINEPROCESS

inlprefix -
SCF/inline/

- allow all errors
- assume process $zzkrn

[#IF [#processexists $WBDOG] |then|
- abort  #wbdog
delay 2 seconds
- delete #wbdog
]

- add #wbdog,autorestart 10 &
, highpin on
, priority 100 &
, program    $system.wasteghs.wbdog &
, defaultvol $system.wasteghs &
, name $wbdog &
, startmode application &
, userid 255,255 &
, cpu first &
, startupmsg "6"

- start  #wbdog

- exit

#POP #INLINEPROCESS
#UNFRAME
```

58. STOPDOG TACL Macro

Make sure the

- green marked name is identical with the WBDOGNAME entry in WEBCONFIG

```
?TACL Macro
==
== TACL Macro the de-install WBDOG as persistent processes.
== 10Jan2011, CW
==
#FRAME

[#IF [#processexists $wbdog] |then|

#PUSH #INLINEPROCESS

inlprefix -
SCF/inline/

- allow all errors
- assume process $zzkrn
- abort  #wbdog
- delete #wbdog

- exit

#POP #INLINEPROCESS
]

#UNFRAME
```

59. INFODOG TACL Macro

Make sure the

- green marked name is identical with the WBDOGNAME entry in WEBCONFIG

```
?TACL Macro
==
== TACL Macro the list WBDOG.
== 10Jan2011, CW
==
#FRAME
#PUSH #INLINEPROCESS

inlprefix -
SCF/inline/

- allow all errors
- assume process $zzkrn
- status #wbdog

- exit

#POP #INLINEPROCESS
#UNFRAME
```

60. STARTSRV TACL Macro

Make sure the

- read marked entries point to location $SYSTEM.WASTEGHS.
- green marked name is identical with the WBSRVNAME entry in WEBCONFIG

```
?TACL Macro
==
== TACL Macro to install WBSRV as persistent processes.
== Adjust the program names and locations to your environment.
== 14Oct2011, CW
==
#FRAME
#PUSH #INLINEPROCESS

inlprefix -
SCF/inline/

- allow all errors
- assume process $zzkrn

[#IF [#processexists $wbsrv] |then|
- abort  #wbsrv
delay 2 seconds
- delete #wbsrv
]

- add #wbsrv,autorestart 1 &
, highpin on &
, priority 100 &
, program    $system.wasteghs.wbsrv &
, defaultvol $system.wasteghs &
, name $wbsrv &
, startmode application &
, userid 255,255 &
, cpu first &
, saveabend on

- start  #wbsrv
- exit

#POP #INLINEPROCESS
#UNFRAME
```

61. STOPSRV TACL Macro

Make sure the

- green marked name is identical with the WBSRVNAME entry in WEBCONFIG

```
?TACL Macro
==
== TACL Macro the de-install WBSRV as persistent processes.
== 14Oct2011, CW
==
#FRAME

[#IF [#processexists $wbsrv] |then|

#PUSH #INLINEPROCESS

inlprefix -
SCF/inline/

- allow  all errors
- assume process $zzkrn
- abort  #wbsrv
- delete #wbsrv
- exit

#POP #INLINEPROCESS
]

#UNFRAME
```

62. INFOSRV TACL Macro

Make sure the

- green marked name is identical with the WBSRVNAME entry in WEBCONFIG

```
?TACL Macro
==
== TACL Macro the list WBSRV.
== 14Oct2011, CW
==
#FRAME
#PUSH #INLINEPROCESS

inlprefix -
SCF/inline/

- allow  all errors
- assume process $zzkrn
- info   #wbsrv,detail
- exit

#POP #INLINEPROCESS
#UNFRAME
```

63.Index