File and Volume Management

Bull DPS 7000

File Recovery Facilities User's Guide

Subject:

This manual describes how to protect and recover user files using the techniques of Before and After Journalization, and system utilities in HA and non-HA environments. Additional information included in this manual concerns the TCRF utility, the functions provided for the BLUE and GREEN JAS versions and HA facilities.

Special Instructions:

Revision 04 of this manual is valid for users of GCOS 7-V9 TS9662.
Revision 03 remains valid for GCOS 7-V8 TS8560.
Revision 02 remains valid for GCOS 7-V7 TS7458.
Revision 01 remains valid for GCOS 7-V7 TS7254.
Revision 00 remains valid for GCOS 7-V6.

Software Supported:

GCOS 7-V9 TS9764

Software/Hardware required:

Date:

September 1999

Bull S.A.
CEDOC
Atelier de reprographie
34, rue du Nid de Pie BP 428
49004 ANGERS Cedex 01
FRANCE

Bull HN Information Systems Inc.
Publication Order Entry
FAX: (978) 294-7411
MA30/865A
Technology Park
Billerica, MA 01821
U.S.A.

47 A2 37UF Rev05

47 A2 37UF Rev05

Preface

Scope and Objectives

This manual describes the facilities available for the protection and recovery of user files by *JAS* (Journalization Advanced Service) that includes the Before Journal and the After Journal. Retrieval functions and special utilities associated with these Journals are also described. File recovery is treated for both non-HA (High Availablity) and HA environments.

Intended Readers

The manual is intended primarily for the System and JAS Administrators and Console Operators, and for all users with an interest in file protection.

Structure

Section 1 introduces the terminology and concepts of journalization, and explains how to make use of journal facilities.

Section 2 describes the features of JAS in terms of its catalog and directory and, the Before and After files. It describes how to handle files and the techniques used in their recovery.

Section 3 describes installing and maintaining JAS. It describes in detail the JAS commands for managing and handling the files.

Section 4 describes Private JAS.

Section 5 describes HA (High Availability) in terms of the service it provides to JAS and TDS. It gives a resume of the CMSC (Complex Management Service) commands for managing the service.

Section 6 treats the ROLLFWD utility and its effects on journalization.

Section 7 describes JRU (Journal Recovery Utility) functions and utilities.

Section 8 describes the TCRF (Transactional Context Recovery Facility) utility in the context of a non-HA environment.

Section 9 deals with the DUMPJRNL utility which enables extracting private data from the After Journal to be written into a user journal.

Appendix A gives hints on how to ensure the Stability and Consistency of User files.

Appendix B describes the messages issued by the different File Recovery Facilities (Before Journal and Rollback, After Journal and Rollforward, DUMPJRNL and TCRF) and either displayed on the console, or printed out in the JOR (Job Occurrence Report).

A glossary and an index end the document.

Bibliography

General, Management, Operation

System Overview 47 A2 22UG
IOF Terminal User’s Reference Manual, Part I 47 A2 38UJ
IOF Terminal User’s Reference Manual, Part II 47 A2 39UJ
IOF Terminal User’s Reference Manual, Part III 47 A2 40UJ

Messages and Return Codes Directory 47 A2 10UJ
JCL Reference Manual 47 A2 11UJ
GCOS 7 System Administrator’s Manual 47 A2 54US
GCOS 7-V8 System Installation Configuration and Updating Guide 47 A2 19US
GCOS 7-V9 System Installation Configuration and Updating Guide 47 A2 23US
Console Messages Message Directory 47 A2 61UU

TDS

TDS Concepts 47 A2 26UT
TDS Administrator’s Guide 47 A2 32UT
TDS/COBOL Programmer’s Manual 47 A2 33UT

HA

High Availability Concepts 47 A2 22UT
High Availability Administrator’s Guide 47 A2 23UT

File and Data Management

UFAS-EXTENDED User’s Guide 47 A2 04UF
Data Management Utilities User’s Guide 47 A2 34UF
GAC EXTENDED User’s Guide 47 A2 12UF
Catalog Management User’s Guide 47 A2 35UF

GPL System Primitives 47 A2 34UL
COBOL System Calls 47 A2 04UL

Syntax Notation

The following notation conventions are used in this manual when describing the syntax of commands:

UPPERCASE	The keyword item must be coded exactly as shown.
Lowercase	Indicates a user-supplied parameter value. The symbolic name digitsn is used to represent a string of decimal digits of maximum length n.
[item]	An item within square brackets is optional.
{item 1} {item 2}	A column of items within braces means that one value must be selected if the associated parameter is specified.
{item 3}	The default value (if any) is underlined.
( )	Parentheses must be coded if they enclose more than one item.
. . .	An ellipsis indicates that the preceding item may be repeated one or more times.

Table of Contents

Table of Graphics

1. Introduction

This manual describes how to protect files against total or partial loss by using File Recovery Facilities.

This section briefly explains the journalization techniques and the protection offered to applications.

1.1 Functions Provided by File Recovery Facilities

File Recovery Facilities allow the contents of files to be protected from total or partial loss, caused by:

- incomplete update on program abort or system crash,
- physical destruction of the media containing the file,
- or logical destruction of the file.

File Recovery Facilities makes use of two journals, the Before Journal and the After Journal, and an associated set of functions including the support of HA (High Availability) requirements. Details of HA-specific features are given in *HA Specific Features* (chapter 5).

In addition, File Recovery Facilities offer TDS applications a User Journal service.

1.1.1 Fundamental Concept of File Recovery Unit (FRU)

An FRU (File Recovery Unit) is a portion of a program which takes a file or set of files from one consistent state at the beginning of its execution to another consistent state at the end of its execution. A program consists of one or several FRUs, each FRU being one of the following:

- an entire Batch or IOF step,
- part of a Batch or IOF step between the beginning of the step and the first checkpoint encountered,
- part of a Batch or IOF step between two checkpoints,
- part of a Batch or IOF step between the last checkpoint encountered and the end of the step,
- a TDS commitment unit or an IQS query.

All the data accessed by an FRU in updating mode is locked, in order to prevent another FRU from accessing that data. The lock-unit is the CI (or page).

When an FRU terminates normally, all the modified data is committed. The files are in a new consistent state.

When an FRU terminates abnormally, the files are returned to the state they were in at the start of the FRU, that is, they are returned to their previous consistent state. Updates already made are rolled back.

1.1.2 Recovery from Software Incident Causing Incomplete Update

User files can be protected against software incidents that cause incomplete update by either *Immediate Update* with the *Before Journal* or *Deferred Update* with the *After Journal.*

Files protected by *Immediate Update* with *Before Journalization* and left in an inconsistent state after a program abort or a system crash will be *rolled back* to the last consistent state. See *Before Journalization* (section 1.2.3).

Files protected by *Deferred Update* and *After Journalization* left in an inconsistent state after a program abort or a system crash will be dynamically *rolled forward* to the next consistent state. See *After Journalization* (section 1.2.4).

1.1.3 Recovery from Physical Media Destruction or Logical File Destruction

User files can be protected against hardware incidents and certain software misuses by the *After Journal*.

User files which are destroyed are rolled forward. This is done by running the RESTORE and ROLLFWD utilities.

The *After Journal* also protects the files against faulty recovery. If GCOS 7 components cannot deliver the list of FRUs because the system files containing the state of the FRUs are unavailable, the *After Journal* is able to make its own decisions. See *Adding Available After Journal Files* (section 2.4.3).

1.1.4 Supporting High Availability Requirements

HA is a function based on disk coupled system configurations enabling reduction of the unavailability time of an HA application (typically a TDS application) when the system crashes.

HA is implemented by CMSC (Complex Management Service) which detects this failure from the backup system and provides the means of starting up a backup HA application. On startup of the backup application, journalized files will be immediately recovered. See Section 4.

1.1.5 Offering a User Journal Service

TDS user records can be stored in the After Journal file while the application is running. These records may be subsequently extracted from the After Journal file using the DUMPJRNL utility. See *DUMPJRNL Utility* (chapter 9).

1.2 Journalization Techniques

Journalization ensures user file integrity. *Before* and *After Journals* are tools which enable recovering user files to a consistent and stable state when an incident such as an abort of an application or system crash occurs.

1.2.1 Deferred Update

While an FRU is updating a file, the system keeps an image of each record as it is updated *in a buffer*. The system physically writes the record to the file *after* the Commitment has been taken.

Deferred Updates are available only for files used by TDS.

1.2.2 Immediate Update

When an FRU is updating a file, the system physically writes the record to the file *before* the Commitment is taken.

Immediate Updates are available for files used by Batch, IOF and TDS.

1.2.3 Before Journalization

Before Journalization is to be used with Immediate Updates.

While a program is updating a file, the system keeps a *Before Image* of each record before it is updated, and writes the CI (control interval) or page containing the record to a Before Journal file.

If the updating FRU terminates abnormally thus leaving the update of the file incomplete, the records from the Before Journal are extracted, and used to overwrite the records in the file. The file is thus restored by being *rolled back* to its last consistent state.

1.2.4 After Journalization

After Journalization is to be used with Deferred Updates.

While an FRU is updating a file, the system keeps an *After Image* of each record as it is updated *in a buffer*, and writes each *After Image* to the After Journal file. The system physically writes the record to the file only after the Commitment has been taken.

If a TDS terminates abnormally thus leaving the update of the file incomplete because some commitments are not consolidated, the records from the After Journal are extracted, and used to overwrite the records in the file. The file is thus restored by being *rolled forward* to its next consistent state.

The After Journalization mechanism is also to be used for recovering from physical incidents.

A file can be reconstructed by:

- restoring a version that has previously been saved,
- and overwriting each updated record with its *After Image*.

1.3 Choosing the Level of Protection by Application

1.3.1 TDS

File protection is especially important in a distributed processing environment where the TDS application is accessed by several users.

The type of journalization can be selected at TDS generation time by specifying the *level of protection* in the FILE INTEGRITY parameter of TP7GEN.

TDS verifies that this protection level is compatible with the type of journalization defined in the description of the file in the catalog or in the runtime JCL statement DEFINE.

Table 1-1 . TDS Protection Levels

Level	**Journalization Techniques**
NONE	NONE
MEDIUM	Immediate Updates with Before Journalization only
HIGH	Immediate Updates with Before and After Journalization OR Deferred Updates and After Journalization

NOTE:

Where the *After Journal* is specified with *Deferred Update*, TDS may also require the *Before Journal* as well depending on the runtime needs of the system. See the *TDS Administrator's Guide.*

1.3.2 Batch and IOF

In Batch and IOF mode, the *Before Journal* can be used with checkpoints or commitments in programs which update files.

Journalization techniques used can be:

- NONE.
- *Before Journal* only.
- *After Journal* only.
- both *Before Journal* and *After Journal*.

1.3.3 Combined Applications

A file updated by several Batch, IOF and TDS applications must be handled with the same level of protection by each application. Consistency of the file is otherwise endangered and it cannot be restored by being rolled forward.

NOTE:

The JOURNAL option can be specified both in the catalog and in the JCL statement DEFINE (see *Specifying the Journal Option* in the next chapter). However, to ensure that ALL steps using the file have the same protection level, you are strongly recommended to specify the protection level only in the catalog description of the file, and not elsewhere.

The relation between TDS Protection Levels and Batch and IOF Protection Levels is as follows:

Table 1-2. Relation Between TDS and Batch/IOF Protection Levels

TDS Protection Level	Batch and IOF Protection Level
NONE	NONE
MEDIUM	Before Journal only
HIGH	Before and After Journals

1.4 Journalization Advanced Service (JAS)

JAS provides File Recovery Facilities through the Before and After Journals, and the utilities associated with the After Journal.

On a given site, this service can be implemented by one or several occurrences of JAS, each JAS occurrence being in charge of the integrity of a specific set of files linked to it.

There are two types of JAS:

- the SYS JAS which is the public JAS,
- the BLUE JAS and the GREEN JAS which are the private JAS. They only protect files cataloged in auto-attachable catalog.

If the system JAS feature is only available on site, the journalization may be done only on the Public SYS JAS.

If the Private JAS feature is only available on site, the two types of JAS are provided: the two private JAS enable the System Administrator to divide the load of file journalization among the three JAS.

If the HA feature is available on site (if the site is part of an HA complex), the two types of JAS are provided: the Public SYS JAS is the non-HA type JAS and the two private JAS are the HA-type JAS. The HA-type JAS allows TDS applications which journalize in one of these JAS to switch to a backup system when the active system crashes in an HA complex. Non-HA type JAS does not have this backup facility. See the *High Availability Concepts Manual.*

The Private JAS mean the BLUE JAS and the GREEN JAS in both Private JAS feature and HA feature.

Table 1-3. SYS, BLUE, and GREEN JAS

	SYS JAS	**BLUE JAS**	**GREEN JAS**
no special features	System JAS	Unavailable	Unavailable
Private JAS feature available on site	System JAS	**Private JAS non HA**	**Private JAS non HA**
Site part of an HA complex	System JAS	**Private HA-type JAS**	**Private HA-type JAS**

2. General Description of JAS Features

2.1 JAS General Description

2.1.1 Domain of Application

2.1.1.1 Type of Applications

The type of application may be:

- Batch,
- Interactive (IOF),
- Transactional (TDS).

2.1.1.2 Type of Files and Characteristics

User files should be cataloged in auto-attachable catalogs. This is mandatory for Private JAS (HA or non HA).

After Journalization identifies the user file by its *efn* (external-file-name) and its *catalog*. The file must always be updated through the ***same*** attached catalog to ensure protection against physical destruction. A catalog which is auto-attachable guarantees this security.

NOTE:
If the catalog of a journalized user file has to be changed, a FORGET_USER_FILE command must be issued.

In order to protect a file cataloged in a non-auto-attachable private catalog, the system journalizes, rolls forward and performs a save on such a file ***only if*** the private catalog is cataloged in SITE.CATALOG and is attached without the DEVCLASS and MEDIA attributes. This avoids the risk of confusing two different files with the same name cataloged in two different private catalogs with the same name.

Files cataloged in the SITE.CATALOG are supported. However, cataloging user files in this catalog is not recommended for performance reasons. Files whose names start with "SITE." or "SYS." are not supported. Uncataloged files are supported by SYS JAS but are not protected against duplicate names. The use of uncataloged files is not recommended.

The techniques for file protection are:

- Before Journalization which allows immediate recovery (rollback) with immediate updates,
- After Journalization which allows immediate recovery (dynamic rollforward) with deferred update, and also deferred recovery (static rollforward).

A JAS protects:

- by using After Journalization and Before Journalization:
  - sequential, relative and indexed UFAS user files for UPDATE processing mode,
  - IDS user files for UPDATE processing mode.
- by using Before Journalization only:
  - relative UFAS user files with direct access mode only for APPEND or OUTPUT processing modes,
  - indexed sequential UFAS user files only for APPEND processing mode.

NOTES:

1. The protected file types and processing modes depend on whether the access method is UFAS or IDS/II. Refer to the appropriate documentation.
2. Dynamic rollforward of sequential UFAS user files is not supported and deferred update is not allowed for these files.
3. A private JAS protects access to files of catalogs linked to it even in input mode and even to files other than UFAS and IDS/II. See chapter 4: *Private JAS*.

2.1.1.3 Limits

A JAS can protect user files of up to 200 applications running on a system, of any application type (TDS, Batch or IOF).

The number of user files simultaneously protected per application is limited to 500 opened files per TDS.

JAS can protect up to 3200 user files not simultaneously updated. These protected user files may be cataloged in an unlimited number of auto-attachable catalogs.

The maximum number of simultaneously active commitment units per TDS is limited to 2047.

The maximum number of processes per TDS is limited to 250.

2.1.1.4 Before Journal Characteristics

The Before Journal supports the following GCOS 7 applications, file types and processing modes.

Table 2-1. Before Journal Characteristics

APPLICATIONS	Batch Interactive (IOF) Transactional (TDS)
FILE TYPES	UFAS IDS/II
PROCESSING MODES	UPDATE for all file formats and organizations. OUTPUT for UFAS Relative with direct access mode. APPEND for UFAS Relative with direct access mode, and UFAS Indexed Sequential files.

NOTE:
With UFAS sequential files, the file-label is always journalized when the file is opened. To rollback to the last checkpoint, first close the file before taking a checkpoint, then re-open the file.

2.1.1.5 After Journal Characteristics

The After Journal supports the following GCOS 7 applications, file types and processing modes.

Table 2-2. Before Journal Characteristics

APPLICATIONS	Batch Interactive (IOF) Transactional (TDS)
FILE TYPES Protection Deferred Updates	UFAS Sequential, Relative, Indexed IDS/II UFAS Relative, Indexed IDS/II ***Dynamic rollforward of sequential UFAS files is not supported***
PROCESSING MODES	UPDATE for all UFAS file organizations ***All files opened in OUTPUT mode and Indexed Sequential files opened in APPEND mode are not protected***

Since Dynamic Rollforward of Sequential UFAS user files is not supported, Deferred Update is not allowed.

2.1.2 Specifying the JOURNAL Option

The JOURNAL option can be specified in both the catalog and in the JCL statement DEFINE.

However, to guarantee that ***all*** the steps using the file have the same level of protection, you are strongly recommended to specify the file protection level in the catalog only, and not elsewhere.

The JOURNAL option specified in the JCL statement DEFINE of a step allows this step to define, for its own use, a specific level of protection for a given file. This option must be used with the utmost care, since the level of protection is not retained for other steps using the file.

When the journal options are specified in the catalog, specifying these options in the JCL statement DEFINE allows *extension or restriction of protection* according to the following rules.

- *Extension of Protection:* If access rights are installed and if the user starting the step does not have OWNER or RECOVERY access rights, the options in DEFINE take effect if they offer the same or more protection as that entered in the catalog. For example, a file protected by AFTER in the catalog can be protected by BOTH for the execution of a step, thus allowing the step to be rolled back if it aborts.
- *Restriction of Protection:* If access rights have been installed and if the user starting the step has OWNER or RECOVERY access rights, *or* if access rights have not been installed, then the options of the JCL statement DEFINE override those entered in the catalog. An example is that a file protected by BOTH in the catalog can be accessed without journalization (NO) for the execution of a step (to allow overnight batch followed by save of the files).

These rules can be summarized as follows:

Table 2-2. JOURNAL Options

	No option JCL statement DEFINE	**Option JCL statement DEFINE**
no access rights installed *or* OWNER and RECOVERY rights	options in catalog	options in DEFINE
access rights installed *without* OWNER and RECOVERY rights	options in catalog	options in catalog

Restriction of the protection provided by AFTER journalization leads to unreliability since static rollforward of files can no longer be guaranteed. This restriction must therefore be used with caution - for example, such an access must be followed by a save of the files. For control checking, all access with inhibition of the AFTER option in the catalog is noted in the JAS directory. A warning message appears on rollforward and an exclamation mark (!) precedes the name of the file in the execution report of the MNJAS LIST command. This notification disappears on each save of the file with journalization.

NOTE:

During the inhibition by a JCL statement DEFINE of the options BEFORE and AFTER as specified in the catalog, the step is connected to SYS JAS at initialization. From TS8560, this connection does not prevent the MNJAS TRANSFER function from executing normally.

2.1.2.1 In the Catalog

Journal options may be cataloged for a given file (using the JCL statement CATALOG or CATMODIF or the GCL MDF command):

```
JOURNAL={ NO | BEFORE | AFTER | BOTH }
```

NO:	The file will not be journalized (default option)
BEFORE:	The file will be journalized in the Before Journal only
AFTER:	The file will be journalized in the After Journal only. If the file is used in a TDS step, it will be journalized in both the Before and After Journals.
BOTH:	The file will be journalized simultaneously in the Before Journal and in the After Journal. BOTH is recommended for protection against both physical destruction and software problems.

NOTE:

Specifying the After Journal in the catalog ensures the permanence of file protection against hardware incidents or faulty recovery.

Each time a cataloged file with the JOURNAL option is assigned and opened in the previously listed processing mode the journal(s) specified in the catalog are used.

If the JOURNAL option is not declared in the catalog, it may be specified in the JCL statement DEFINE. However, the file is journalized only during the execution of this step.

2.1.2.2 In JCL Statement DEFINE

A DEFINE statement can be used on each file to be journalized. The syntax is as follows:

```
DEFINE ifn, JOURNAL={ NO | BEFORE | AFTER | BOTH }
```

ifn:	Internal file name of the file to be journalized.
NO:	The file will not be journalized.
BEFORE:	The file will be journalized in the Before Journal only.
AFTER:	The file will be journalized in the After Journal only. If the file is used in a TDS step, it will be journalized in both the Before and After Journals.

BOTH:	The file will be journalized simultaneously in the Before Journal and in the After Journal. BOTH journals is recommended for protection against both physical destruction and software problems.

The system will automatically journalize the file if the corresponding journal exists. If the Journal does not exist, the step will abort.

NOTE:
If the option JOURNAL=AFTER has been specified for a file used by a TDS step, the system will override this option so that it becomes JOURNAL=BOTH. The file will be journalized simultaneously in both Before and After Journals.

If the option JOURNAL=BOTH is specified for a file used by a TDS step, deferred updates are recommended. See the *TDS Administrator's Guide*.

2.1.3 Activating Journalization

2.1.3.1 Activating the Before Journal

To activate the Before Journal for files protected by SYS JAS in a particular step, the JCL of the step must contain:

- either an ASSIGN statement for a file which has been cataloged with the option JOURNAL=BEFORE or BOTH,
- or a DEFINE statement containing the option JOURNAL=BEFORE or BOTH.

If there is no reference in the JCL to at least one file journalizing in the Before Journal, Before Journalization will not be activated for the step. An attempt to dynamically ASSIGN and OPEN a file cataloged with the option JOURNAL=BEFORE or BOTH, will result in an abnormal condition and the code NJAFT will be returned. For an explanation of this return code, see the *Error Messages and Return Codes Directory*.

For files protected by private JAS, the Before Journal can be activated in the same way. However, it can also be dynamically activated by dynamically assigning in the program, *without any reference in the JCL*, a file protected by a private JAS.

During the execution of a Batch or IOF step, journalized files should not be dynamically deassigned because if they were, the Before Journal would be unable to roll them back if the step aborted or the system crashed. The minimum precaution is to take a commit immediately after a dynamic deassign.

In addition, in batch and IOF environments, the ifn (internal file name) remains assigned until the end of the step. The result is that the file may not be assigned in another step. Any further dynamic assign in the same step on this ifn for another file leads to the abnormal code IFNASG being returned and the message DF3 *ifn* ALREADY ASSIGNED being issued.

2.1.3.2 Activating the After Journal

To activate the After Journal for files protected by SYS JAS in a particular step, the JCL of the step must contain:

- either an ASSIGN statement for a file which has been cataloged with the option JOURNAL=AFTER (or BOTH),
- or a DEFINE statement containing the option JOURNAL=AFTER (or BOTH).

If there is no reference in the JCL to any journalized file, the After Journal will not be activated for this step and the step will not be able to journalize. Each time an attempt is made to dynamically assign and open a file cataloged with the option JOURNAL=AFTER or JOURNAL=BOTH, the code NJAFT will be returned and the message JP53 will be sent to the JOR. See *Error Messages and Return Codes Directory*.

For files protected by Private JAS, the After Journal can be activated in the same way. However, it can also be dynamically activated by dynamically assigning in the program, *without any reference in the JCL*, a file protected by a Private JAS.

2.1.4 Concurrent Access

When a file is shared by several concurrent batch steps which are updating it, each step must specify that the file is to be protected by the Before Journal. This is to ensure that the file can be rolled back to the last synchronization point in the case of either an incomplete update or a deadlock.

TDS steps can use Deferred Updates even if another step is updating the same file. If the step is Batch or IOF, it must specify that the file is to be protected by the Before Journal.

A file is protected by Before Journal in the case of concurrent access, if it is cataloged with the journalization options BEFORE or BOTH or assigned with these options in the DEFINE statement.

2.1.5 Programming Considerations

A file may be journalized in the Before Journal only if the reference mode is MOVE. The integrity of the file is not guaranteed in LOCATE mode which only applies to GPL.

If GPL is used in a Batch or IOF environment, a journalized file using LOCATE mode is deassigned when the step aborts or when the system crashes. It will not be rolled back. For these files the message:

```
SYSTEM JL01 ROLLBACK CANNOT BE PERFORMED ERROR CODE=xxxxxxxx
```

will be sent to the JOR.

If a file is defined with SHARE=FREE in WRITE mode for several steps, it is shared without control. If it is journalized by at least one of the steps, its integrity cannot be guaranteed if rollback occurs since there is no synchronization either between the updates and rollback or between different rollbacks.

However, if the file is defined with SHARE=MONITOR, it is shared under the control of GAC (Generalized Access Control). Synchronization is guaranteed between rollbacks and journalization.

The maximum size of a journalized file block is 32 Kbytes whatever the device class of the disk allocated to the Before Journal file.

From TS8560, the GPL H_INVFRU primitive and the COBOL H_JAP_HINVFRU entry point allow a batch/IOF step to invalidate the updates of all journalized files, from the last clean point taken. Before calling this function, you must close all journalized files.

The immediate invalidation of updates concern only the Before journalized files. The function is therefore rejected if at least one file is only journalized in After mode. However the After Journalization is recommended because you can then perform a static rollforward after a rollback failure or disk loss.

If the step aborts or if the system crashes after the primitive call, the usual recovery mechanism functions normally.

See the *GPL System Primitives* and *COBOL System Calls* manuals.

2.2 JAS Files

After Images

After Images are recorded in an unlimited set of sequential files.

All the applications using the same JAS journalize in the same After Journal file and the images are logged sequentially in time.

GAC ensures that two FRUs (File Recovery Unit) cannot update the same record simultaneously. GAC guarantees that the After Images of the same object are always written in the order of the locks applied. The After Images belonging to two FRUs simultaneously executing can be written to the Journal Files in any order. GAC and the After Journal allow the FRU to be serialized.

Before Images

In each system, the three JAS, namely, Public SYS JAS, and Private BLUE JAS and Private GREEN JAS, use the same Before Journal file called SYS.JRNAL. This file is allocated by TAILOR at system installation and is cataloged in SYS.CATALOG.

Before Images are handled with the minimum of user intervention.

Before Images are recorded in the Before Journal as they are created, and are applied in reverse order when the file is rolled back.

To guarantee the integrity of the journalized file, the Before Image of a CI (Control Interval) is recorded before any modifications are made to the CI concerned. Each Before Image is prefixed by a field which contains information needed for the rollback.

Before Images belonging to several different files are stored in the same Before Journal file.

Batch and IOF Applications

Before Images of a Batch or IOF step are logged in a subfile of the SYS.JRNAL.

When there is not enough space in the SYS.JRNAL to log the Before Images of a step, the Before Journal subfile associated with the step can be extended outside the SYS.JRNAL.

From TS8560, if you know that the quantity of Before images is such that it overloads the SYS.JRNAL, specifying a H_BJRNL order in the JCL creates a first extension for the step, without waiting for SYS.JRNAL to overflow, and directs the Before images into this extension instead of into the SYS.JRNAL.

See section *Before Journal Extensions (Batch and IOF)* later in this chapter.

TDS Applications

A subfile of the SYS.JRNAL is dedicated to each TDS but no Before Images are logged in the SYS.JRNAL file. They are logged in a TDS Before Journal File. A TDS Before Journal File is dedicated to each TDS. This file may be located on up to four media to improve performance by distributing the I/O operations.

See section *TDS Before Journal File (TDS)* later in this chapter.

2.2.1 JAS Catalog

When a Private JAS is created, a private catalog dedicated to it is created. This private catalog is used to catalog all the JAS files, namely:

- the JAS directory,
- and the JAS Primary and Secondary Journal Files.

This catalog is auto-attachable.

SYS JAS uses SYS.CATALOG which is permanent on the system.

2.2.2 JAS Directory

The JAS directory is a system file attached to a JAS. There is one JAS directory per JAS. This directory is created and cataloged in the JAS Catalog when the JAS is created by the JAS administrator through the MNJAS CREATE command. The file is extended without operator intervention.

The JAS directory stores the following information:

- the physical characteristics, the identification and sequencing of the After Journal files,
- the identification of user files that have been journalized and information about their last save date,
- the identification of TDS user journals and last date of DUMPJRNL,
- and the identification of media that are to be used for journalization.

The space used in the JAS directory is a function of the number of active journal files and the number of logged aborted FRUs that are stored in each primary journal file. Consequently, the percentage of space used by the JAS directory is continuously increasing. The administrator will need to recover space from time to time.

The ways to recover the JAS directory space are:

- to activate the TRANSFER_PRIMARY function which transfers primary journal files to secondary journal files,
- to activate the RECYCLE_JOURNAL_FILE function which transforms an active journal file to an available journal file,
- to save all the journalized files, which recycles only those active journal files that contain obsolete After Images,
- to extract the user journal of all the TDS applications that have journalized, a procedure which also recycles the active journal files,
- to delete old user files which are no longer being journalized.

Before an overflow occurs, the JP22 message is sent to the console once the threshold of available space on the media (2%) is reached. Until JAS directory space is recovered, all starting steps will receive the JP23 message in their JOR and will be rejected.

An overflow on the JAS directory stops journalization by the JAS and prevents all the journalizing steps from making any update on the user files. Results are then unpredictable. The percentage of used space can be found through the MNJAS (MAINTAIN_JAS) utility.

When the JAS directory space is considered too small to ensure required application processing and file integrity, the JAS administrator can create a larger JAS directory when recreating the JAS. To do this, journalized user files must first be saved and the JAS must be deleted.

NOTE:

Although it is a queued file, the JAS directory is not a library and must not therefore be accessed throug the LIBMAINT utility.

2.2.3 Files Specific to Before Journalization

Before journalization makes use of a set of files: the Before Journal files. These files are of three different types:

- **The SYS.JRNAL file:** Part of this system file is used to keep track of each step (Batch, IOF or TDS) that uses the Before Journal at a given time. The other part of this file is used for the journalization of Batch and IOF steps.
- **Before Journal extensions:** Used by Batch and IOF steps, these files are created when no more space remains available for journalization in the SYS.JRNAL file or on request by specifying a H_BJRNL JCL order.
- **TDS Before Journal files:** TDSs do not journalize in the SYS.JRNAL file. Each TDS owns its own Before Journal files: its primary TDS Before Journal files and its Commit extensions.

NOTES:

1. The Before Journal files are volatile. Only the Before Images taken from the last restart point (the beginning of the current FRU) are kept in these files.
2. The After Journal files containing the After Images of the user files must not be on the same media as their corresponding SYS.JRNAL file, Before Journal extensions and TDS Before Journal files. Different media must be used so that, in the case of media failure preventing immediate recovery using the Before Journal, a deferred recovery (static rollforward) is still possible.

2.2.3.1 The SYS.JRNAL File

This file is used to keep track of each step (Batch, IOF or TDS) that uses the Before Journal at a given time, and for the journalization of Batch and IOF steps.

Each step (Batch, IOF or TDS) using the Before Journal is represented by an entry in the SYS.JRNAL file. The CONFIG parameter BJSIMU defines the maximum number of steps that may use the Before Journal at a given time, and therefore the number of entries in the SYS.JRNAL file.

The SYS.JRNAL file is shared in read and write mode in a multiprogramming environment. A number of steps and TPRs can journalize simultaneously while others roll back their files after an abort.

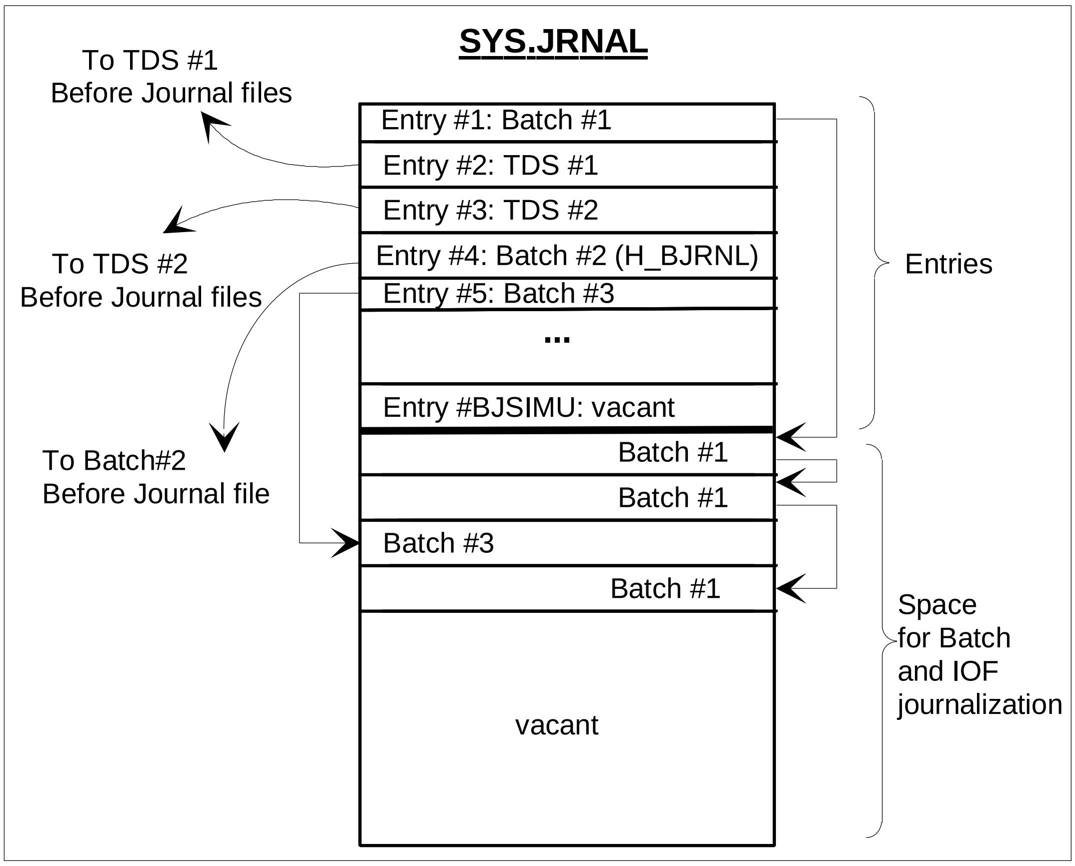


Figure 2-1. The SYS.JRNAL File

2.2.3.2 Before Journal Extensions (Batch and IOF)

Batch and IOF steps journalize primarily in the SYS.JRNAL file. But, although the SYS.JRNAL file is volatile, it may overflow if the cumulative size of journalized records exceeds the size of the file. The Before Journal can be extended without operator intervention.

From TS8560, in order not to overload the SYS.JRNAL with the Before images of a step which creates too many journal records, specifying a H_BJRNL order in the JCL creates a first extension for the step, without waiting for SYS.JRNAL to overflow, and redirects the Before images into this extension instead of into the SYS.JRNAL.

The Before Journal space available to a batch or IOF step may be extended in up to 15 files. These extensions are created with the block size of the SYS.JRNAL file and a size of 3168 file-blocks of 512 bytes or 720 file-blocks of 4 Kbytes. Provided that space is available, they may be expanded on the same media.

The list of media that can be used for creating the Before Journal extensions can be provided:

- either through the JCL statements ASSIGN H_BJRNL and ASSIGN H_BJRNL1, leading to the creation of a first extension as soon as the step is journalized for the first time,
- or by answering the JL02 message.

The ASSIGN H_BJRNL JCL statement has the following format:

```
ASSIGN H_BJRNL    DVC=device-class
                  MEDIA=(media-name[, media-name[, ...[, ...]]])
                  FILESTAT=TEMPRY, NEXT, POOL;
```

The ASSIGN H_BJRNL1 JCL statement has the same format.

Each of these statements may specify a list of up to four media. The specified media must all have the same device class, but this device class need not be the same as the SYS.JRNAL device class.

The media specified in the ASSIGN H_BJRNL1 constitute a spare list which will only be used if one or more of the media provided by the ASSIGN H_BJRNL are not available. A maximum number of four media taken from any of these two lists will therefore be used for the Before Journal extensions.

If ASSIGN H_BJRNL is not specified, or if there is insufficient space on the specified media, the following message will be sent:

```
JL02 ron MORE SPACE NEEDED FOR BEFORE JOURNAL, WHERE ?
```

As long as there is enough space on the media specified in the reply, no further questions appear on the console. A question is sent for each step needing an extension. If too many questions appear, it means the size of the SYS.JRNAL file is not appropriate and should be increased, or that a batch/IOF application has consumed too much Before Journal (no checkpoint taken).

When either the next checkpoint or the end of the step is reached, all the Before Journal extensions belonging to the step are deleted and their corresponding disk space is released. The same applies when rollback occurs.

Before Journal extensions must be avoided as far as possible, because they are time consuming.

NOTE:

I/O requests on the Before Journal are accounted in the JOR for Batch and IOF steps and are printed out in the following form:

```
JOURNAL ON BEFORE: number of i/o requests : nnn
```

where nnn includes the number of I/Os on the SYS.JRNAL file and the number of I/Os on the Before Journal extensions (whose individual ifn's are H_Jxxxx).

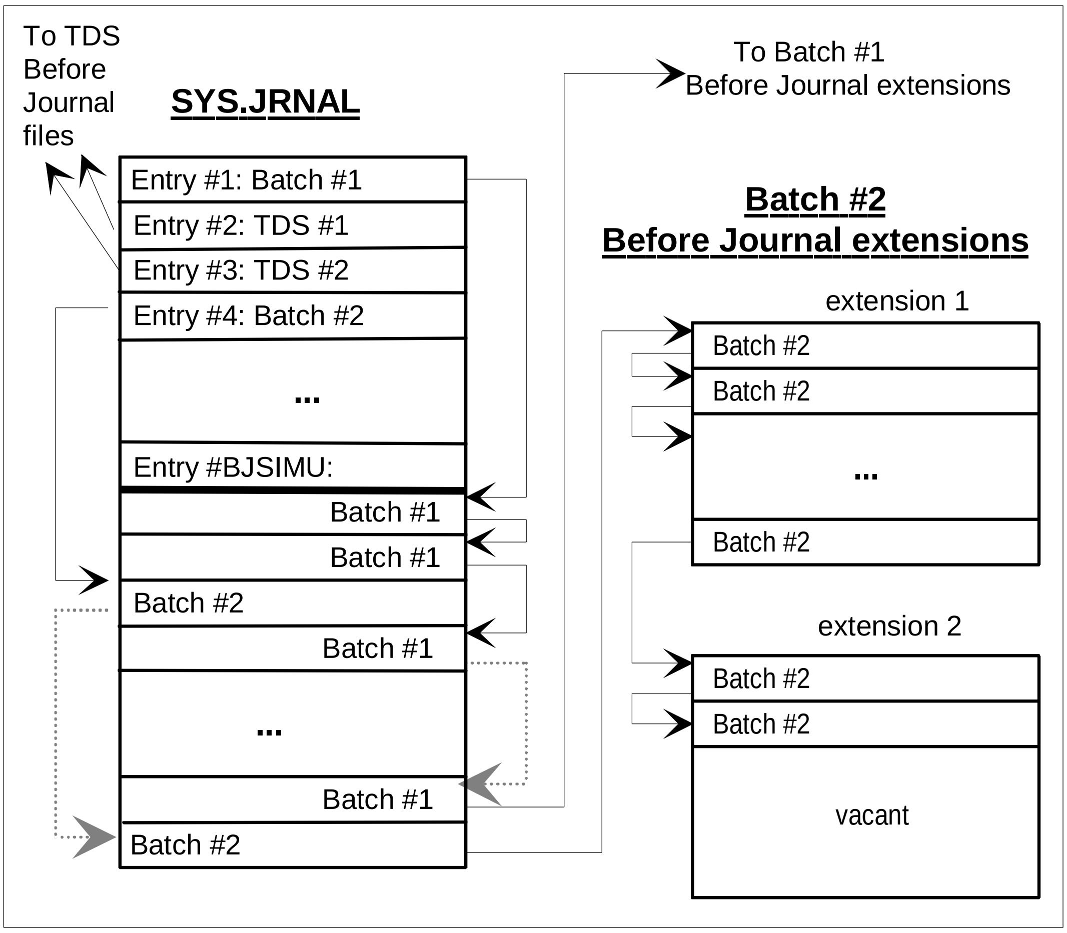


Figure 2-2. Before Journal Extensions

2.2.3.3 TDS Before Journal Files

TDSs do not journalize in the SYS.JRNAL file. Each TDS owns its own Before Journal files.

The TDS Before Journal files are of two types:

- **Primary TDS Before Journal files:** Each primary TDS Before Journal file is divided into two parts. The first part is used to keep track of the commitment units using this TDS Before Journal file at a given time. The other part is used for commitment unit journalization. These files may be allocated on up to four media. The use of more than one media minimizes disk access conflicts between the commitments units, thus enabling better performance.
- **Commit extensions** that are created when an overflow occurs in one of the primary TDS Before Journal files.

The TDS Before Journal files are created with the same block size as the SYS.JRNAL file.

The list of media can be provided:

- either through the JCL statements ASSIGN H_BJRNL and ASSIGN H_BJRNL1,
- or by answering the JL07, JL08 or JL09 messages (see console messages in Appendix B: *Messages from File Recovery Facilities*).

The ASSIGN H_BJRNL JCL statement has the following format:

```
ASSIGN H_BJRNL,  DVC=device-class
                 MEDIA=(media-name[, media-name[, ...[, ...]]])
                 FILESTAT=TEMPRY, NEXT, POOL;
```

The ASSIGN H_BJRNL1 JCL statement has the same format.

Each of these statements may specify a list of up to four media. The specified media must all have the same device class but this device class can be different from the SYS.JRNAL device class.

The media specified in the ASSIGN H_BJRNL1 constitute a spare list which will only be used if one or more of the media provided by the ASSIGN H_BJRNL are not available. A maximum number of four media taken from any of these two lists will therefore be used for the TDS Before Journal files.

The primary TDS Before Journal files are created at the beginning of the TDS session. They are all created with the same size which is computed at the beginning of the TDS session:

- At TDS Cold Restart, this size is directly linked to the maximum number of commitment units that may journalize simultaneously under TDS (the number of terminals the TDS can support simultaneously, as declared in the TDS generation).
- At TDS Warm Restart, this size is automatically increased to take account of Commit extensions that have been created during the previous session.

NOTE:

The maximum number of commitment units that may journalize simultaneously under TDS cannot exceed 2047.

If Deferred Update is used, the size of the primary TDS Before Journal files can be reduced by specifying the clause:

`MAXIMUM BEFORE JOURNAL COMMITMENTS UNITS IS` *`nbbjcu`*

at TP7GEN, where nbbjcu is the maximum number of simultaneous transactions that need the Before Journal (i.e., those transactions that use SUPPRESS DEFERRED UPDATE and CI SPLITTING).

A commitment unit journalizes in only one primary TDS Before Journal file. If the size of the primary TDS Before Journal file for a commitment unit becomes insufficient, a Commit extension is created for this commitment unit on the same media. The size of such an extension increases automatically as needed.

When the commitment unit terminates, the Commit extensions are destroyed and the space is released.

At the end of the TDS session, all the space needed for before journalization is released.

NOTES:

1. The messages JL00, JL06 and JL07 are written in the JOR after the end of a TDS session giving the maximum size for the TDS Before Journal files that has been reached during this session. They also give the number of successful and unsuccessful Commit extensions, and the number of rollbacks that have been performed during the session. If the TDS step was terminated by a system crash, the number of rollbacks does not appear in the JOR because this information is not available.
2. I/O requests in the Before Journal are accounted for in the JOR of the TDS steps and are printed out in the following form:

   ```
   JOURNAL ON BEFORE: number of i/o requests : nnn
   ```

   where nnn includes the number of I/Os on the different TDS Before Journal files (whose individual ifn's are H_Jxxxx) and the few I/Os that are performed on the SYS.JRNAL file.

Figure 2-3 shows the relationship in the case of four primary TDS Before Journal files.

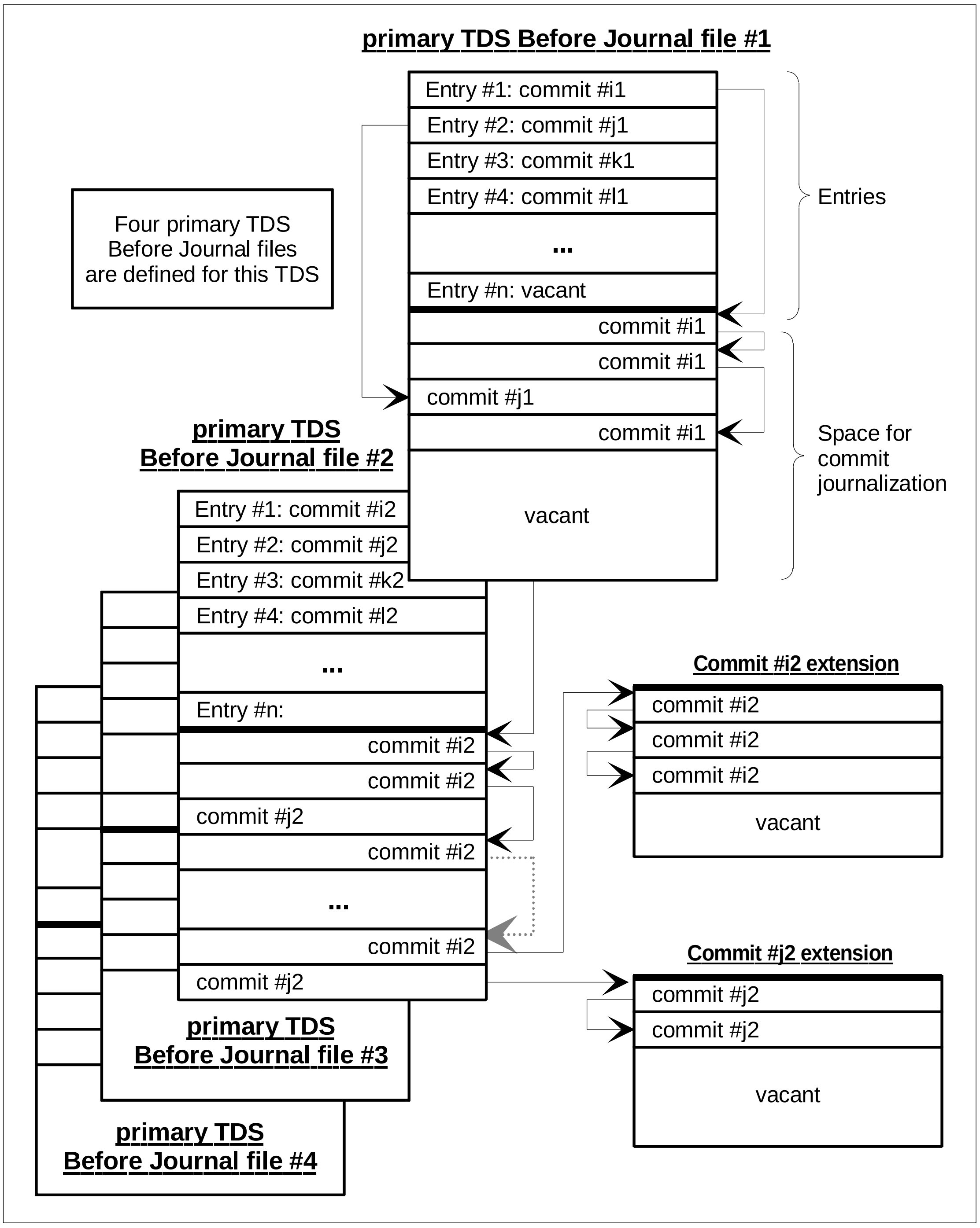


Figure 2-3. The TDS Before Journal files

2.2.3.4 Changing the Size of the Before Journal

If the Before Journal has to be extended too often, the System Administrator can change the size of this file as follows:

1. Wait until all steps cease journalizing, and then deallocate SYS.JRNAL.
2. Preallocate SYS.JRNAL with a new size using the following JCL (in this example, the SYS.JRNAL is located on an FSA volume with a file block size of 512 bytes):

```
$PREALLOC SYS.JRNAL, DVC=devclass, UNIT=BLOCK,
          GLOBAL=(MD=volume_name, SIZE=new_journal_size),
          NONE=(BLKSIZE=512),
          FILESTAT=CAT, EXPDATE=365;
```

3. Shut down the system.
4. Perform a Cold Restart on the system.

The results of the JOR giving the maximum size in file blocks used, enable the System Administrator to gauge the increase for the Before Journal file. If a step is aborted due to a system crash, this information is lost.

Make sure that the new size allocated for the Before Journal is large enough because extending the file is time-consuming. See below *Choosing the Block Size for the Before Journal.*

When the Before Journal block size is 512 bytes, the SYS.JRNAL file size must be chosen in the following range:

- minimum: 132 x BJSIMU file blocks,
- maximum: 660,000 + (66 x BJSIMU) file blocks.

When the Before Journal block size is 4096 bytes, the SYS.JRNAL file size must be chosen in the following range:

- minimum: 30 x BJSIMU file blocks,
- maximum: 150,000 + (15 x BJSIMU) file blocks.

2.2.3.5 Choosing the Block Size for the Before Journal

Disk space consumption for the Before Journal files (the SYS.JRNAL, the Before Journal extensions, and the TDS Before Journal files) can be considerable if the block size for these files does not fit the data block size of the disks on which they are located.

From GCOS 7-V7 TS7458, the System Administrator can choose the block size of the Before Journal files that best suits the disk configuration. The block size is first used for preallocation of the SYS.JRNAL file, then for Before Journal extensions and the TDS Before Journal files.

Two values are possible:

- 512 bytes, which is the correct value for CKD/VBO or FSA disks,
- 4096 bytes, which is the correct value for CKD/FBO disks.

If there is a mixture of CKD/VBO and CKD/FBO disks, use the higher value, 4096 bytes.

If there is mixture of FSA and CKD/FBO disks, use the value for the type of disk that prevails for the Before Journal files (SYS.JRNAL, Before Journal extensions, TDS Before Journal files).

Use the GCOS 7 installation tools (GIUF and Tailor) to define the block size of the Before Journal while dealing with the SYS.JRNAL file. By default, the value is that which is appropriate for the disk supporting the SYS.JRNAL file.

To change the block size of Before Journal files, proceed as follows:

1. Wait until journalization has ceased for all steps.
2. Deallocate SYS.JRNAL.
3. Preallocate SYS.JRNAL with the new block size using the following JCL:

```
$PREALLOC  SYS.JRNAL, DVC=devclass, UNIT=BLOCK,
           GLOBAL=(MD=volume_name, SIZE=journal_size),
           NONE=(BLKSIZE=new_journal_block_size),
           FILESTAT=CAT, EXPDATE=365;
```

4. Shut down the system.
5. Perform a Cold Restart.

NOTES:

1. In an HA environment, the Before Journal block sizes of the two members must be identical. This is only checked by the recovery step; if they are not identical, the takeover fails.
2. If TCRF is to be used, the Before Journal block size of the backup system must be the same as that of the production system.

2.2.4 Files Specific to After Journalization

2.2.4.1 After Journal

Each After Journal file is single-volume. The After Journal records the After images in a set of sequential journal files on disks called *primary Journal files*.

The After images can be transferred into a set of sequential files (on tapes or cartridges) called *secondary Journal files*.

The sequence of primary and secondary journal files created is maintained in the JAS directory.

The After Journal files are cataloged in the JAS catalog with SHARE=ONEWRITE for the primary files.

The name of an After Journal file is an alphanumeric character string and has the following values:

Primary Journal File Name:

Format: `jas_name.JA.Jimedianame`

where:

jas_name	the name of the JAS: SYS, BLUE or GREEN.
i:	the journal sequence number assigned to each primary journal file uniquely identified on a volume; the maximum number is 8, the numbers being chosen by the After Journal.
medianame:	the name of the disk volume supporting the corresponding Primary Journal file.

Secondary Journal File Name:

Format: `jas_name.JA.J1medianame` where:

jas_name	the name of the JAS: SYS, BLUE or GREEN
medianame:	the name of the volume (MT or CT) which supports the corresponding Secondary Journal file. From TS8560, for dual copies, only media whose names end with an odd number (ODD) are accepted. You construct the associated media for the secondary copy by adding 1 to the numeric characters which end the media name of the primary copy. Examples: primary copy=AFT005, secondary copy=AFT006 primary copy=AFT039 secondary copy=AFT040 If the media name of the primary copy ends with the largest possible numeric value given by the number of characters that end the name of the primary copy, the media name of the secondary copy will end with the same number of "0"s. Examples: primary copy=AFTER9 secondary copy=AFTER0 primary copy=AFT999 secondary copy=AFT000

Since disks and tapes do not have the same names, there cannot be any confusion between the primary and the secondary Journal file names since these files cannot be allocated on the same volume.

From GCOS 7 TS7254, secondary file organization has changed. The old and new organizations are not compatible but do not affect user visibility:

- a TRANSFER command after passing to or from the new technical status, causes switching to a new secondary file,
- a static rollforward or a DUMPJRNL will take into consideration the two types of file organizations and a secondary file can be read and processed even if it has been created under a different technical status.

2.2.4.2 Choosing the Size of After Journal Block

The default size of the journal block is 8192 bytes of which 8162 is used for data. The Administrator can change the block size through the CREATE or MODIFY_PARAMETERS commands of the MNJAS utility. All available and active primary files must first be removed. See chapter 4: *Private JAS*.

The administrator should define a block size to obtain a fill rate close to 80% depending on whether steps regularly journalize. It is better to begin with a large block size and reduce it progressively to obtain the optimum rate. The MNJAS LIST command provides statistics on the block fill rate. The journal block size must be larger than 512 bytes.

If the journal files are on VBO disks, the block size must be shorter than the track size. To obtain the most efficient use of track space, a block size less than half the track size should be chosen.

If the journal files are on FBO disks, the maximum block size is 32000 bytes including 30 bytes required by the Access Method.

If the journal files are on CKD/FBO disks, it is recommended to use a block size in multiples of 4Kbytes. In that way, the wasted space is reduced, because the Journal blocks are always mapped onto one or several physical multiples of 4Kbytes. But if the Journal block is not almost 100% full, there may be considerable amounts of wasted space.

2.2.4.3 Adding Available After Journal Files

The first time the System Administrator uses the MNJAS CREATE command for adding available After Journal files. Thereafter, media can be made known to the system through the MNJAS MODIFY_MEDIA command.

For Primary After Journal files, before cataloging and allocating the new files, the PRIMARY_MEDIA option of the CREATE command and the ADDPRIMMD option of the MODIFY_MEDIA command check that no old After Journal files still reside on the disks. If so, the function is rejected.

For Secondary After Journal files, tape or cartridge must be empty:

- either be empty or contain an obsolete non-cataloged file,
- or an old After Journal file, in which case, the system will uncatalog and deallocate the old After Journal file before cataloging and creating a new After Journal file.

Names of media used for secondary After Journal files must always be different from names of media used for primary After Journal files. Otherwise the function is rejected.

Tape or cartridge cannot be used for the After Journal if it contains a still or once cataloged file which is not an After Journal file. The tape or cartridge is refused and the system will try another media.

After Journal files can also be cycled automatically, as a user option, whenever any individual After Journal file becomes obsolete through *save*, DUMPJRNL, FORGET_USER_FILE and FORGET_USER_JOURNAL functions.

2.2.4.4 Obsolete After Images

For each journalized file, the system keeps track of the date corresponding to the last save of the file concerned. The date recorded on completion of the save corresponds to:

- the end of the save for a static save,
- the beginning of the save for a dynamic save.

An After image whose date is earlier than the date recorded for the save is redundant since it corresponds to an update already included in the last save. Such images are referred to as *obsolete* After images.

When a static save is performed (FILSAVE utility), all After images are treated as obsolete.

When a dynamic save is performed (DYNSAVE utility), only those After images whose date is earlier than the beginning of the dynamic save are treated as obsolete. After images whose date is later than the beginning of a dynamic save are not considered to be obsolete **even if they correspond to updates included in the save**.

Additionally, an After image of a file which is the subject of a FORGET_USER_FILE command is also treated as obsolete.

The DUMPJRNL utility records the date of the last extraction of the User Journal performed for the TDS concerned. Data that constitutes the User Journal is treated as obsolete once the extraction has been performed.

Additionally, User Journal data of a TDS which is the subject of a FORGET_USER_JOURNAL command is also treated as obsolete.

2.2.4.5 Recycling Active After Journal Files

After Journal files can be removed from the list of active files either manually or automatically and returned to the list of available After Journal files. Primary After Journal files are not deleted by this function. Secondary After Journal files are uncataloged.

Manual Cycling

After Journal files may be removed from the active list when their images are considered obsolete. This is done by the MNJAS RECYCLE_JOURNAL_FILE command. Before using this command, the System Administrator must ensure that all such journalized files have been saved. If not, essential images for rollforward may be lost.

The *recycle* function removes:

- either one active After Journal file and its previous After Journal files from the list,
- or all the active After Journal files created after a specified date and time,
- or all the active After Journal files containing only obsolete images.

The recycled After Journal files are returned to the list of available After Journal files.

RECYCLE_JOURNAL_FILE can be issued while files are being journalized *with the exception of the current After Journal file.*

Automatic Cycling

AUTOCYCLE=1 specified in the MNJAS CREATE and MODIFY_PARAMETERS commands, automatically removes an After Journal file from the active list once all the After images on that After Journal file are obsolete. The After Journal file is then returned to the list of available After Journal files.

In the case of FILSAVE or FILDUPLI, and DUMPJRNL, this does not apply to the *current* After Journal file. To recycle the current file, the MNJAS RECYCLE_JOURNAL_FILE command must be used instead while no journalization is active.

Automatic cycling occurs on files containing obsolete After Images when one of the following is performed:

- either a user file is saved,
- or the DUMPJRNL utility is run,
- or a FORGET_USER_FILE or FORGET_USER_JOURNAL command is issued.

When AUTOCYCLE=1 is specified, only the After Images created since the last time *file save* was run are kept. It is always possible to remove After Journal files manually independently of file saves.

2.2.4.6 Removing Available After Journal Files

REMPRIMMD and REMSECMD options of the MNJAS MODIFY_MEDIA command remove After Journal files from the list of available After Journal files as follows:

- files on disks are deleted,
- files on tapes or cartridges are uncataloged.

MODIFY_MEDIA can be run (for a different media than that of the Active Journal file) while files are being journalized.

Any attempt to remove *active* After Journal files is rejected.

2.2.4.7 Switching After Journal Files

When no step is journalizing, the current After Journal file is not assigned. If a new step wants to journalize, the system assigns the current After Journal file unless the MNJAS SWITCH command has been executed. A new After Journal file is used in the following cases:

- when the end of file of the current After Journal file is reached,
- if the ROLLFWD or DUMPJRNL utility needs to access the current active After Journal file; journalization switches to a new After Journal file.

When an After Journal block cannot be written to an After Journal file due to an I/O error, the system will try to write the block to another After Journal file (the next file of the same medium or a new medium). No information is lost and journalization can continue. The system tells the operator that such a switch has occurred through the JP21 console message.

When a COLD or CLEAN restart is performed after a system crash, the system may switch the After Journal file to isolate the part of the After Journal file which contains uncertain information about aborted Batch and IOF steps. A doubt may only arise when the Batch or IOF step was taking a checkpoint or was in the termination phase when the crash occurred.

A similar After Journal file switch occurs when a system error such as the unavailabilty of TDS swap files or an error in the JAS directory, makes it impossible to determine which FRUs have aborted (TDS SWAP files unavailable or error on JAS directory).

The system also switches After Journal files when predating is detected in the journalization dates. See *Difference in Date and Time Between Systems* (section 5.7.2).

The MNJAS LIST command can be used to display the cause of switching, except for ROLLFWD or DUMPJRNL /Journalization conflict.

NOTE:

The list of available After Journal files is not circular. The first available After Journal file encountered at the beginning of the list is always the one eligible for switching.

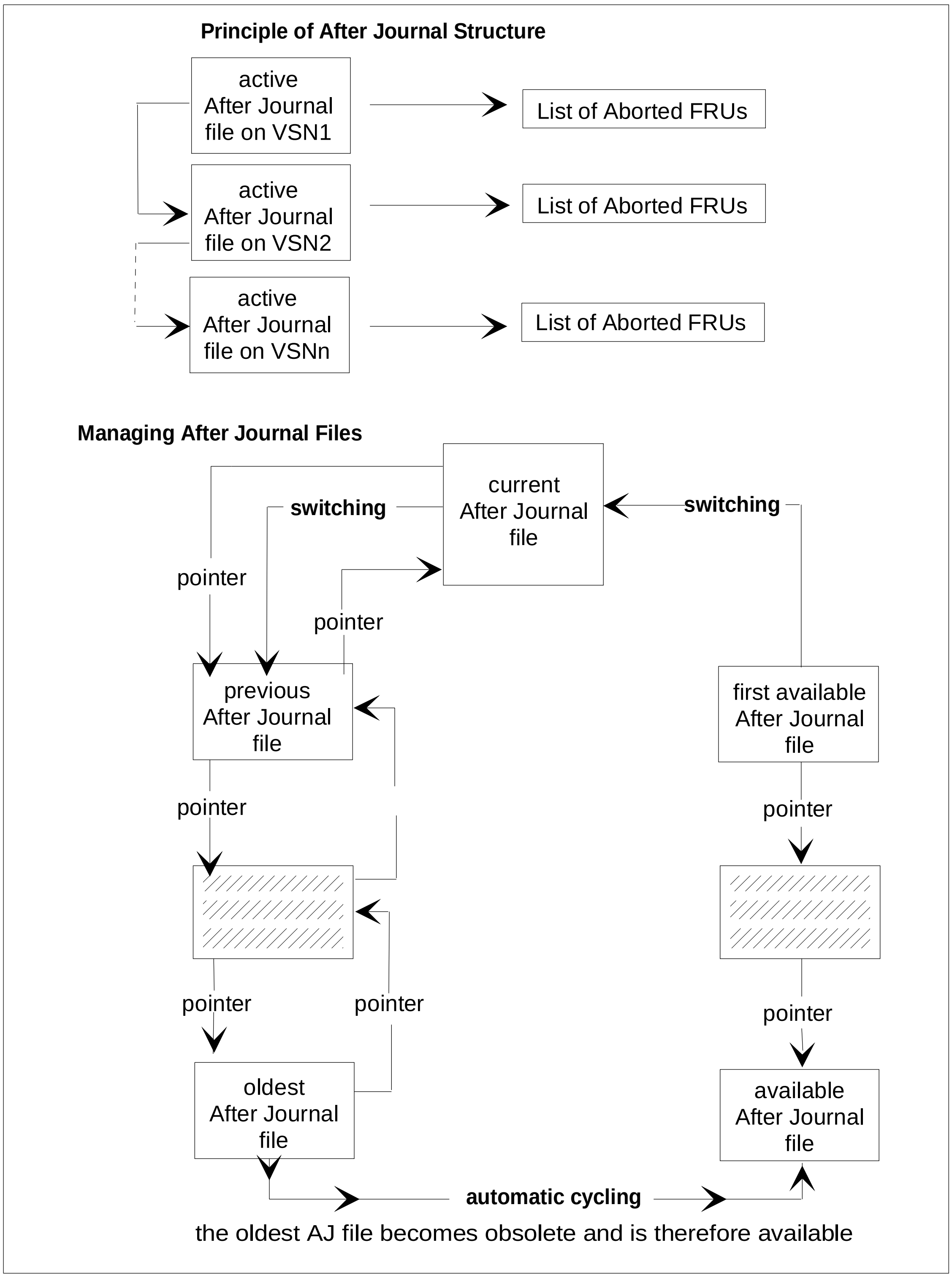

Principle of After Journal Structure
active After Journal file on VSN1
List of Aborted FRUs
active After Journal file on VSN2
List of Aborted FRUs
active After Journal file on VSNn
List of Aborted FRUs
Managing After Journal Files
current After Journal file
switching
switching
pointer
pointer
previous After Journal file
first available After Journal file
pointer
pointer
pointer
pointer
pointer
oldest After Journal file
available After Journal file
automatic cycling
the oldest AJ file becomes obsolete and is therefore available

Figure 2-4. Description of After Journal Files in JAS Directory

2.2.5 JAS Files Integrity

The term *damage*d applied to a file means that the volume containing the file cannot be accessed or that the information in the file is not correct. Errors such as those involving I/O operations and hardware defects affect the integrity of files.

2.2.5.1 Incidents on JAS Catalog

If the JAS catalog is damaged, the JAS directory and journal files can no longer be accessed.

These files that can no longer be accessed must be rebuilt manually as follows:

- issue a BUILD_CATALOG GCL command for a new jasname.CATALOG in the case of BLUE or GREEN JAS,
- issue a BUILD_DIRECTORY GCL command for a new jasname.JA,
- catalog jasname.JADIR with the SHARE=DIR option,
- issue the MNJAS LIST command to list all the active journal files and the list of the available journal files,
- catalog all the active and available primary journal files and all the active secondary journal files in jasname.CATALOG with SHARE=ONEWRITE for the primary After Journal files.

If the JAS catalog (or in the case of SYS JAS, the SYS.JA directory of SYS.CATALOG) is saved regularly, its recovery is simplified since its involves:

- restoring the catalog (or the SYS.JA directory of SYS.CATALOG),
- starting the After Journal,
- printing the After Journal,
- and updating the catalog with the last modifications including the addition of new files.

2.2.5.2 Incidents on JAS Directory

The integrity of the JAS directory is fundamental for ensuring the user file integrity. If the system has mirrored disks, the System Administrator is advised to allocate the JAS directory on them for a better security.

The JAS directory is damaged:

- if it does not contain correct information about aborted FRUs which may occur when:
  - the TDS SWAP or RECOV files are unavailable at recovery time (message JP44 on the console, System error notified in the MNJAS LIST command),
  - the JAS directory was unavailable at recovery time, or an IO error has occurred on it (messages JP43 and JP44 on the console, System error notified in the MNJAS LIST command).
- if the JAS directory is definitively unavailable (messages JP41, JP47, and JP74 on the console, message JP46 in the JOR of journalizing steps).

As far as possible, the System records in the JAS directory that such an error occurred and prevents new steps from journalizing. ***The JAS directory must be recovered***:

- for HA-type JAS, the System automatically recovers the JAS directory when it becomes available,
- for SYS JAS, the System Administrator is responsible for recovering the JAS directory,
- for non-HA Private JAS, the System Administrator can recover the JAS directory or wait for its automatic execution.

When the SYS JAS directory has been unavailable, the system cannot prevent new steps from journalizing. Therefore, the messages JP41, JP43 or JP44 warn the System Administrator that the JAS directory has been unavailable. The administrator will have to recover the JAS directory before:

- any new journalizing step is started,
- and/or before the system is restarted.

The two ways for recovering the JAS directory, which allow rollforwards on user files:

- If the JAS directory is now available, ***run JRU functions***.

  All FRUs considered active before the incident are aborted. Since some commitment units not committed may be considered committed, it is necessary to check the output.

In the case where the JAS directory is damaged, recovery is ensured:

- *by executing the JRU utility for SYS JAS, or non-HA Private JAS*
- *at the beginning of a journalisation session for non-HA Private JAS*
- *for HA-type JAS, each time the JAS is started. Therefore the JAS service should be stopped and started again if necessary, since JRU functions are started automatically in this case.*

User files that are damaged can then be recovered using the Rollforward utility.

- If the JAS directory is still unavailable but a saved copy of it can be restored, the MNJAS REBUILD command can recover the JAS directory from its restored copy (if journalization is active, a copy of the JAS directory can be obtained by using the SHARE = DIR parameter). If the JRU functions have not succeeded in recovering the JAS directory, the MNJAS REBUILD command can recover the JAS directory from its current state. The JAS catalog must have first been checked and updated where necessary to reflect the latest status of the After Journal files. All active and available primary files are to be cataloged, whereas only active secondary files are to be cataloged.

  Only user files which were known by the JAS in the saved Directory can then be restored by ROLLFWD.

If none of these methods allows the JAS directory to be recovered, the System Administrator will have to build a new JAS by using the MNJAS CREATE command. Journalization will then become available again.

2.2.5.3 Incidents on Before Journal Files

Since the Before Journal file is a volatile file that is reused, rollback cannot be delayed. Rollback occurs when the FRU (file recovery unit) terminates or aborts, or at Warm Restart after a system crash.

When an incident occurs on a Before Journal file, user files cannot be rolled back. Then, they can be recovered by running the FILREST utility followed by the ROLLFWD utility if they were journalized with JOURNAL=BOTH which is recommended. See the *Data Management Utilities User's Guide*.

However, if the disk supporting the SYS.JRNAL file or one of the TDS or batch Before Journal files is unavailable for some time, it is better to recover the disk drive and then restart the system. If not, only a static rollforward can reset the status of user files.

In the case where the system has been interrupted and can only be restarted on COLD or CLEAN, it is recommended for SYS JAS, to use the spare system disk (refer to section *Impossible Warm Restart* later in this document) and, for those sites that possess it, to start the TCRF facility to recover the files. After that, the first system can be restarted on COLD or CLEAN without affecting the integrity of the files.

In the case of HA JAS, it is recommended to *force* switching of the JAS on the member which was in the BACKUP state at the time of the system crash. See *System Crash during Recovery at Takeover* (section 5.7.4).

2.2.5.4 Incidents on After Journal Files

If a non-recoverable I/O error occurs while writing a block to the After Journal**,** the current After Journal file is closed and the block is written to another journal file located on another media, if available. Integrity of JAS files and user files is ensured.

If no more file space is available for journalization, steps and TDS transactions will abort. Journal files must be made available again by using the ADDPRIMARY option of the MODIFY_MEDIA command, the RECYCLE_JOURNAL_FILE command or the FILSAVE utility. File integrity is maintained allowing steps and TDS transactions to be repeated.

If an I/O *read* error is detected during the execution of a TRANSFER_PRIMARY command on a Primary Journal file, the transfer stops. User files must be saved to avoid having to roll them forward.

If an I/O *write* error is detected during the execution of a TRANSFER_PRIMARY command on a Secondary Journal file, the file and its dual copy (if any) are closed and the transfer restarts from an internal checkpoint on the *next* available Secondary Journal file and its dual copy (if any). Integrity of JAS files and user files is ensured.

If a system interrupt occurs during the execution of a TRANSFER command, a new TRANSFER command will lead to the switching of secondary After Journal files.

When an I/O error occurs during Rollforward on a Primary After Journal file or on a single copy Secondary After Journal file, user files although stable might not be consistent. Such files can be recovered manually.

If an I/O error is detected during Rollforward on a double copy After Journal file, processing switches without operator intervention to the double copy except for mounting of the media where required.

2.2.5.5 JAS Access Rights

The After Journal files and the After Journal directory are all cataloged files. Access to these files is restricted to the SYSADMIN project.

Unauthorized users submitting utilities which attempt to read or update the After Journal directory and the After Journal files, will be denied access.

After journalization which involves writing to the After Journal files and to the After Journal directory is done by the system. Access rights do not therefore affect After Journalization.

2.3 JAS Abnormal Events

Incidents detected in the Journals are logged in the SYS.ERLOG system file with CODE=JRNAL. The PRLOG utility is executed to print out the contents of the SYS.ERLOG for debugging purpose.

Other JAS events logged in the System TRACE (SYS.SWLOG system file), DOMAIN=JAS are also useful for debugging. They concern system date errors and HA-type JAS service state changes, as well as the After journal part used by dynamic rollforward.

2.3.1 Return Codes JRNAL (Before Journal Incidents)

The following describe abnormal events which may occur in the Before Journal and their consequences for the user.

Return Code JRNALERR

JRNALERR is sent when conflicting information in the tables of the Before Journal has been detected. The cause of this inconsistency is recorded in the SYS.ERLOG file.

Action:	Contact the Service Center.

Return Code CPERR

CPERR means that an I/O operation on the Before Journal has not been performed.

Action:	Contact the Service Center.

Return Code IOFAIL

IOFAIL means an I/O operation on the Before Journal has been incorrectly performed:

- either it has not completed,
- or it has given erroneous results.

Action:	Contact the Service Center.

Return Code NOTALL

NOTALL means that *rollback* on *all* user files has not completed successfully.

Action:	Refer to the JOR or contact the Service Center.

Return Code TPUNKN

TPUNKN means that a transaction tries to journalize a file whose name starts with "SITE." or "SYS.". This is not supported.

Action:	Change the name of the file and catalog it preferably in an autoattachable catalog.

2.3.2 Return Codes JAP (After Journal and JAS Incidents)

An After Journal is considered as being damaged when:

- either its internal tables are in an unstable state,
- or its files including the JAS directory are damaged.

The following describe abnormal events which may occur during the After Journal processing and during the JAS controls and their consequences for the user.

Return Code JRNALERR

JRNALERR means an inconsistency has occurred in the After Journal tables or directory:

- either due to a directly related error,
- or as a consequence of other anomalies.

User files and the After Journal files are not affected.

Action:	Contact the Service Center.

Return Code ABORTPG

ABORTPG with the following message displayed on the console:

```
JP14  jasname: NO MORE AVAILABLE PRIMARY AFTER JOURNAL FILE.
```

means that the system aborts all journalizing steps. Since the After Journal is not affected, file integrity involving the recovery of user files is guaranteed.

Action:	To restart the aborted steps, ensure that new Primary Journal files are available through one of the following commands:

- to move primary Journal files to available secondary Journal files
  `MNJAS jas_name COMMAND = 'TRANSFER ... '`

- to recycle primary journal files
  `MNJAS jas_name COMMAND = 'RECYCLE_JOURNAL_FILE ... '`

- to add a new media list.
  `MNJAS jas_name COMMAND='MODIFY_MEDIA ADDPRIMARY=new_media_list'`

In an HA environment, ABORTPG is also returned to a journalizing step which requires to take a checkpoint or a commit, or to open a file or to terminate whereas the JAS has received the order TERMINATE STRONG.

Return Code SEQERR

SEQERR is sent if a previously detected error has not been processed by the application.

Action:	Search for the original cause of the problem in the SYS.ERLOG file and/or in the JOR of the application. When starting an HA-type TDS in an HA environment, *JAP 16, SEQERR* appearing in the JOR means that the complex is not switchable.
Action:	Use the DJAS command to check if a batch/IOF application uses ***both*** SYS JAS and the HA-type JAS, which prevents the complex from becoming switchable.

Return Code NJAFT

NJAFT is sent when the After Journal files:

- either do not exist,
- or have not been correctly accessed at GCOS 7 restart for the following reasons:
  - either the JAS directory was not created and managed through the MNJAS DELETE and CREATE functions,
  - or the disk supporting the JAS directory could not be accessed at GCOS 7 restart.

Action:	Either use MNJAS to correct the situation or check the volumes supporting the After Journal file and perform another GCOS 7 restart.

JAP3 NJAFT can also be sent when the connection to the JAS SYS has not been done at step initialization. Either the Before Journal or the After Journal has not been defined:

- either by a static ASSIGN of at least one file cataloged with JOURNAL = BEFORE or AFTER or BOTH,
- or by at least one DEFINE specifying JOURNAL = BEFORE or AFTER or BOTH.

This journal definition is mandatory for journalizing files protected JAS SYS.

Action: Modify the JCL.

Return Code ITMNAV

ITMNAV means no After Journal file is available for a step requiring After Journalization.

Action: To add new media, key in:

```
MNJAS COMMAND='MDMD ADDPRIMMD=vsn';
```

Return Code DAMAGED

DAMAGED means that After Journalization is no longer possible due to the occurrence of a system error. See chapter 6: *ROLLFWD Utility*. The return code appears in a message displayed at the console and the JP46 message is printed in the JOR.

Action: Start JRU or rebuild the JAS directory and check the result.

Return Code FUNCNAV

JAP 3, FUNCNAV means the file being opened is cataloged in a private catalog which is:

- either not cataloged in the SITE.CATALOG,
- or attached with the DEVCLASS and MEDIA attributes.

The message JP52 is sent to the JOR. If the file was to be opened for the FILSAVE utility, a save is performed but the warning message JP36 is sent to the JOR and ***no*** date is recorded in the JAS directory.

FUNCNAV is set at file opening if a file protected by a private JAS is not cataloged in an auto-attachable catalog.

Action:	Modify the catalog declaration.

Return Code SHLVVIOL

In an HA environment, SHLVVIOL is set when rules of use of HA JAS are not respected. For Rules for files or steps, see *MANAGING THE PRIVATE JAS* section later in this document. For the LINK command, see section *LINK* later in this document.

In a non-HA environment, SHLVVIOL is set if a TDS tries to open a user file not protected by the same JAS as the system TDS file.

Return Code JAP 5 CONFLICT

This return appears at Process Group initialization with JP46 : JOURNALIZATION CANNOT BE PERFORMED, and means either:

- the After Journal files SYS.JA.*Jimedianame* are cataloged with a sharing mode different from ONEWRITE
- the After Journal files are busy by another job

Action:	In the first case, modify the sharing mode of the After Journal file and restart the jobs
	In the second case, free the After Journal files and restart the jobs

Other Return Codes

Other return codes represent anomalies such as:

- abnormal events in the After Journal,
- I/O errors in the After Journal directory.

For I/O errors in the After Journal directory, perform the following procedure:

- abort the step to which the code returned,
- abort all journalizing steps (the integrity of all the files even those in Deferred Update is guaranteed),
- check the journal directory through the command:
  `MNJAS` *`jasname`* `COMMAND='LIST;'`

- if any journal file is lost, follow the sequence given:
  - enter the commands
    ```
    MNJAS jasname COMMAND='DELETE;'
    MNJAS jasname COMMAND='CREATE;'
    ```
  - run FILSAVE on all journalized files. See *Data Management Utilities User's Guide*.
- if no journal file is lost, restore JAS directory from a save and enter the command:
  ```
            MNJAS jasname COMMAND='REBUILD;'
  ```

System Errors

If the message JP44 appears at the console, an error has occurred in journalization. All journalizing steps must be aborted.

Action:

Since the integrity of user files in Deferred Update is not guaranteed:

- either run JRU to recover the After Journal for SYS JAS and non-HA Private JAS
- or stop, then start the HA-type JAS service so that JRU functions are automatically run
- or start a step to be connected to a non-HA Private JAS.

The MNJAS REBUILD command can also recover the JAS directory.

2.3.3 Application Incidents

Batch/IOF Applications

When a Batch or IOF application aborts, the Journals are notified of the incident. The system tells the Journals which After Images are invalid and which Before Images are to be applied.

TDS Applications

The consequences of a TDS application abort, depend on the nature of the abort:

- for a silent or fatal TDS abort, all the CUs (commitment unit) either terminate normally or abort,
- when the TDS has been killed through the TJ or the TSRV FORCE=1 command, some CUs are committed but have not completed, whereas other CUs abort.

In both cases, immediate recovery is performed as follows:

- in both cases, the aborted CUs are rolled back and the user files are reset to a stable state as far as ROLLBACK is concerned,
- for committed but uncompleted CUs in the second case, dynamic rollforward applies the After Images to consolidate those CUs, and user files are reset to a stable state as far as ROLLFWD is concerned.

2.4 Consequences of System Events on JAS

2.4.1 System Restart

The effects of System Restart on the Journals and the consequences for user files depend on the previous state of the system, namely:

- either System Shutdown when operations are normally terminated,
- or system crash due to fatal conditions which no longer enable the system to function at all.

NOTE:
COLD and CLEAN restarts are the same as far as the Journals are concerned. That is the reason why messages refer only to COLD RESTART in both cases.

After System Shutdown

When the command TERMINATE_SYSTEM is issued, the GCOS session terminates normally, resulting in:

- all TDS sessions completing,
- and all Batch and IOF steps either completing or remaining suspended at their last checkpoint.

Whatever the type of Restart after Shutdown, user file integrity is guaranteed. No recovery need be performed.

WARM RESTART:

is the normal way to restart the system after a System Shutdown. Journal files and user files are stable and consistent. All applications are ready to restart.

COLD or CLEAN RESTART:

the restart of Batch and IOF steps depends on their state at Shutdown:

- *all* steps which completed normally, are ready to restart since their journal files and their user files are stable and consistent,
- steps which were suspended are not restarted even though the journal files and their user files are stable and consistent at the state of their last checkpoint(s).

After System Crash

In the case of a system crash, running steps are suddenly interrupted.

WARM RESTART:

- the system provides the JAS with a list of aborted TDS FRUs, aborted Batch and IOF steps for activating immediate recovery.
- Journal files and user files are reset stable and consistent.

COLD or CLEAN RESTART:

- After Journal files are stable and in a consistent state.
- TDS Before Journal files and Batch/IOF extensions are deleted.
- The SYS.JRNAL file is cleaned.
- User files need deferred recovery.

NOTE:

If the After Journal files are on VBO disks, the current journal file may be left unstable. Answer YES to the question "FILE SALVAGING?".

Recovery depends on the application:

TDS-only Applications

User files opened at the time of system crash are neither stable nor consistent. In the absence of a system context for immediate recovery, only deferred recovery is possible. See *Deferred Recovery* (section 2.6.2).

Batch and IOF applications

In the absence of a system context for the status of the aborted or committed steps, the JAS cannot perform immediate recovery. The After Journal enters its own recovery mode in which case deferred recovery may then be necessary.

2.4.2 Setting the System Date and Time

The terms *time* and *date* separately imply both date **and** time. Both these terms are used according to their context.

Setting the correct system date is important for the After Journal because of the Deferred Recovery mechanism. In Deferred Recovery (File Restore followed by a Static Rollforward), a set of After Images is applied to the restored user file. This set of After Images is defined by two dates: the start date of the Static Rollforward and its end date.

Each After Image is stamped with the system date of its creation.

2.4.2.1 Predating

The Deferred Recovery mechanism relies on the assumption that the After Images are found chronologically in the After Journal files. This assumption is not always accurate since predating may have occurred.

The system administrator can change the GCOS 7 system time at will. He can change it at system restart (ISL option), or with GCOS 7 running (the MODIFY_TIME command). Predating may occur if the system time is set backwards.

CAUTION:
From GCOS 7-V7 TS7458 onwards, system time can be set backwards (using the MODIFY_TIME command) even when journalization is active.

Each predating generates a time overlap. A time overlap stretches between two dates. Inside this time overlap, After Images are not found chronologically in the After Journal files.

If the system time has been set backwards during a system restart, the time overlap generated is delimited by:

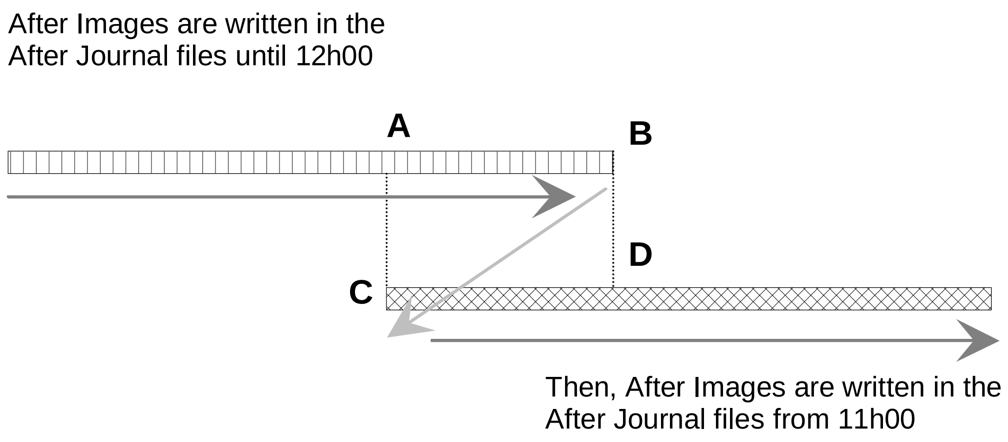
- the system time specified at system restart and
- the last system time known in the previous GCOS 7 session.

If the system time has been set backwards using a MODIFY_TIME command, the time overlap generated is given in the TM18 message that follows the MODIFY_TIME command:

```
TM18 Journal After MAY BE IN PREDATING MODE TILL date and time
```

EXAMPLE:

At 12 noon, while journalization is active, the system date is set backwards to 11h00. This generates a time overlap from 11h00 to 12h00.


After Images are written in the
After Journal files until 12h00
A
B
D
C
Then, After Images are written in the
After Journal files from 11h00

"A" and "C" After Images are stamped 11h00.

"B" and "D" After Images are stamped 12h00.

When predating occurs, After Journal switching is automatic and the predating is indicated by the MNJAS_LIST command as the reason for the switching.

❑

2.4.2.2 Effect of Predating on the Deferred Recovery Mechanism

The only problem that can be generated by predating concerns the Deferred Recovery mechanism. Static Rollforward cannot be safely performed if either the begin date or the end date of this Rollforward is set inside the time overlap.

In the example in the previous paragraph, performing a Static Rollforward with a begin date set to 11h30 would be an ambiguous request, as the starting point of this Rollforward can be taken as "11h30 between A and B" or "11h30 between C and D".

As the system administrator has complete control of the situations under which predating can occur, these constraints are easily dealt with.

Rollforward BEGDATE

If a Static Rollforward has to be undertaken and a time overlap exists:

- specify the BEGDATE parameter for the Rollforward **explicitly**,
- specify a BEGDATE **outside the time overlap**. The BEGDATE must match the date of the save you want to restore. However, if the date of the save you want to restore falls inside the time overlap, specify another BEGDATE just prior to the beginning of the time overlap and make use of the SKIPERR parameter.

In general, do not perform saves (either static or dynamic) inside the time overlap.

Rollforward ENDDATE

If a Static Rollforward has to be undertaken but there is a time overlap, wait for the end of the time overlap before sending the Rollforward command.

2.4.3 Decisions Taken by the JAS

The application is responsible for managing the state of its FRUs and for processing its commitments. The state of the FRU(s) is logged in a special file called the *Commitment Logging file* and is supplied to the JAS for recovery purposes.

The conditions in which the JAS takes its own decisions are when:

- the commitment logging files cannot be read, the reasons being I/O errors, unavailability of TDS SWAP files, After Journal migration, or Cold and Clean System Restart for batch/IOF after a crash during Checkpoint,
- an internal incident occurs affecting the JAS such as the unavailability or inaccessibility of the JAS directory.

When a Cold or Clean Restart follows a crash that occurred during a checkpoint in a journalizing step, the JAS declares all ready-to-commit Batch and IOF FRUs *committed*. Immediate recovery is not possible and user files remain unavailable until deferred recovery is performed. The following message appears at the console for each application concerned:

```
JP45  Jas_name: Xron.ssn[.ckn] COMPLETED WITH REGARD TO JOURNAL
      AFTER COLD OR CLEAN SYSTEM RESTART.
      OPENED USER FILES REMAIN UNSTABLE ;
      RUN FILREST + ROLLFWD FOR EACH FILE.
```

When the JAS directory file is inaccessible at system restart, JRU or the MNJAS REBUILD command can be used to make heuristic decisions on:

- all Batch and IOF steps,
- and TDS commitments.

whose aborted or committed states have yet to be recorded in the JAS directory.

JRU or the MNJAS REBUILD command must also be used when the message JP44 appears at the console. See *Incidents on JAS directory* (section 2.2.5.2).

2.5 Coupled Systems

The two ways of using Coupled Systems where file integrity is concerned, are:

1. Each system is used for the processing of a set of files through some TDS, Batch and IOF applications. Each System uses its SYS JAS to journalize its files. If one of the systems fails, the other system having stopped its own file processing, can be temporarily used as the *backup* of the failed system. This is not true for non-HA Private JAS.
2. Both systems are used for the processing of the same set of files through the same TDS, Batch and IOF applications, insuring the High Availability of the TDS. Both systems use HA-type JAS, BLUE JAS and GREEN JAS, to protect the associated user files. This enables TDS applications running on any of the two systems and journalizing in one of the HA-type JAS to switch onto the *backup* system without interrupting and without imposing any constraints on the TDS, batch and IOF steps that are running on the *backup system.*

Coupled Systems offer a better availability of transactional applications than single systems.

The following describes how to use Coupled Systems *without* HA:

- Since each system has its own SYS JAS, the SYS JAS files of a system (SYS.JRNAL, SYS.JADIR, and the SYS.JA.J-medianame) should ***not*** be on the same media as those used for the SYS JAS files of the other system. The reason is that the names of the files do not identify the system they belong to. Confusion in identifying the file on the appropriate system might provoke the destruction of After Journal files on the coupled system and prevent correct file recovery.
- User files located on shared media should be processed by one system. If they are also to be processed by the coupled system, their recovery requires using both SYS JAS, one on each of the systems, and is the user's responsibility.
- In the same way, if the two systems use the non-HA Private JAS, each system must have its own Private JAS files and care must be taken not to locate these Private JAS files on the same media for both systems.

To use one system as a *backup* when the other system is unavailable, allocate the SYS JAS files as well as the user files on shared media provided that the two systems use different media for SYS JAS files and to proceed as follows:

- stop journalization on the *backup* system,
- run the TCRF functions (See chapter 7: *JRU (Journal Recovery Utility and Functions*)),
- on the *backup* system, restart applications using the After Journal files of the failed system,

- before the failed system recovers, stop all journalization on the backup system,
- finally catalog the SYS JAS files of the *backup* system again since TCRF would have modified the catalog for the SYS JAS files.

In a *coupled* system configuration, although TCRF offers some functions equivalent to HA, it has the following restrictions:

- TCRF is intended to function where one of the systems is transaction-oriented and the other is batch-oriented. *It is not intended to be used as HA where coupled systems are both transaction-oriented. When one system is used as the backup of the other system, the former cannot use its own SYS JAS.*
- The switching to the backup system is manual, whereas in a HA context the takeover is automatically performed on detection of a system crash.
- The delay for TDS applications to be restarted on the backup system is much greater than the delay in case of HA takeover.
- The constraints on the location of SYS JAS files do not exist in an HA context: they can be located on non-shared media. To use TCRF, SYS JAS files must be located on *sharable* media provided that they are not on the same media as the SYS JAS files of the other system.
- In a non-HA *coupled* system environment, having two SYS JAS, one on each system may lead to difficulties for file journalization and file recovery. File integrity is the user's responsibility. However in an HA *coupled* system environment, using the same HA-type JAS for both systems enables journalization and recovery to take place on both systems (depending on the activity of the services) without operator intervention.

The best availability and the best operability are provided by HA. See chapter 4: *Private JAS.*

2.6 User File Recovery

2.6.1 Immediate Recovery

Immediate recovery involves rollback and dynamic rollforward and is necessary when abnormal events occur such as:

- system crash or any abnormal system shutdown,
- abort of Batch and IOF steps,
- and abort of the TDS application or transaction.

2.6.1.1 Rollback

Rollback returns journalized user files to their logical status at the beginning of each aborted FRU. User Files are therefore consistent and in a stable state.

At Restart after system crash or step abort, *rollback* is activated without operator intervention to restore the logical status of journalized files to the beginning of an FRU. Messages are written to the JOR of the step and appear at the submitter terminal. If the submitter is not logged on, messages are sent to the console.

The message RR01 is displayed at the console. See the *Console Messages manual*.

The *rollback* of an aborted CU (commitment unit) also occurs without operator intervention. The number of rollbacks performed during the session is recorded in the JOR of the TDS application.

From Release V3 onwards, *Rollback* is a function performed by GCOS irrespective of the option REPEAT specified in the STEP statement. REPEAT allows the step to be restarted once rollback has been performed.

When the files have been successfully rolled back, the message JL04 is written to the JOR of the step for which *rollback* was performed.

At Warm System Restart, if no step can be rolled back, a status message is sent to the JOR of the system job X0001.

An unsuccessful rollback is recorded in the JOR of the submitter’s step and is logged in the SYS.ERLOG file.

Dynamic Rollback

Dynamic *rollback* is invoked by some IDS/II verbs such as ERASE ALL to restore the database to the status it had when the IDS/II verb was invoked. IDS/II makes the decision to roll back its database independent of checkpoints or commitments. Dynamic Rollback is available in Batch, IOF and TDS environments.

2.6.1.2 Dynamic Rollforward

Dynamic *rollforward* returns journalized user files to their logical status at the termination of each FRU successfully committed. User files are therefore consistent and in a stable state.

Dynamic rollforward is activated:

- either at abnormal termination of a TDS step,
- or on Warm System Restart after a system crash.

Dynamic *rollforward* consists of applying After Images to update user files accessed in Deferred Update mode and modified by a committed FRU. It applies only to TDS applications when they abort.

At the end of a dynamic *rollforward*, a status is sent to the JOR of the step for which the recovery from an abort, was made.

If an incident occurs on a user file to be rolled forward, dynamic *rollforward* continues for the other user files.

A user file not recovered by dynamic *rollforward* is inaccessible to the applications concerned. The only way to recover such a file is to perform deferred recovery.

Messages are written to the JOR and appear at the console. Incidents detected during dynamic rollforward are recorded in the JOR of the submitter’s step and logged in the SYS.ERLOG file.

2.6.2 Deferred Recovery

Deferred recovery consists of two phases:

- File Restore.
- and Rollforward.

Deferred recovery is required:

- either when a hardware failure has occurred on the volume containing the user file,
- or when immediate recovery on a user file cannot be performed or has failed.

2.6.2.1 File Restore

File Restore is performed by the utilities:

- either FILREST (JCL) or RESTORE_FILE (GCL),
- or FILDUPLI (JCL) or COPY_FILE (GCL) to write the user file from its last save.

FILREST has an option (CKJRNL=1) which enables the user to check the consistency of the file with regard to the last save date known to the After Journal. From TS7254, this option runs whether the user file is VBO or FBO.

The following utilities make a file save known to the After Journal:

- either FILSAVE or DYNSAVE (JCL without EXPORT option) or SAVE_FILE (GCL with UPDJRNL=1),
- or FILDUPLI (JCL) or COPY_FILE (GCL) with UPDJRNL=1.

2.6.2.2 EpochBackup 7

When a save is made through EpochBackup 7, the system updates the last save date. Then when the restore is done through EpochBackup 7, the system checks the last save date unless the keyword FORCE_JA is used with the RESTORE_FILE or RESTORE_LIST commands (these commands are available within the EpochBackup 7 command monitor).

2.6.2.3 Static Rollforward

Static *rollforward* sets the user files to their latest consistent status corresponding to the date of the last committed update. This date may be the current date.

The *rollforward* utility (ROLLFWD) identifies the journal files and asks for the media supporting these files to be mounted. *Rollforward* recognizes which After Images belong to aborted FRUs and ignores them. It applies only valid After Images on restored versions of user files. See chapter 5: *HA Specific Features*.

If all file saves have been made known to the After Journal, and if the last save has been used to restore the file, there is no need to specify the beginning date to the ROLLFWD utility.

From TS8560, under certain conditions you can rollforward a file on a save copy which has a different name from the journalized file. Refer to chapter 6: *ROLLFWD Utility*.

3. JAS Installation and Administration

3.1 JAS Installation

The term *JAS* concerns both SYS JAS and Private JAS. The term *Private JAS* covers both BLUE JAS and GREEN JAS. The characteristics of *JAS* are:

- there is one *Before Journal* per system if Before Journalization is required,
- there might be one SYS JAS (hence one SYS.JADIR) per system,
- and potentially one BLUE JAS (hence one BLUE.JADIR) and one GREEN JAS (hence one GREEN.JADIR), shared between two systems in an HA context.

HA is an unbundled optional product which has its own MI (Marketing Identifier). It can be installed only when purchased, to allow generation of the complex, start up the CMSRs (complex management servers) and use of the HA-type JAS.

Private JAS is also an unbundled optional product which allows the use of the two Private JAS in a non-HA-type context.

Installing JAS involves creating the files it needs to operate. The JAFTER Tailor command is used to create the SYS.JADIR and primary journal files for the first time. If SYS.JADIR is to be deleted and created again, the command CREATE of the MNJAS (MAINTAIN_JAS) processor must be used.

A SYS JAS created on or restored from Releases prior to GCOS 7 V7 can be administered with the V7 Release, ***if it is present at system restart*. *However simply preallocating the SYS JAS directory does not enable journalization.***

NOTES:

1. Journal files already created by the old JAGEN utility can be used without running the CREATE command.
2. To execute the CREATE command for journal files already created by the old JAGEN utility, first issue the DELETE command before executing CREATE.
3. Releases prior to V6 cannot administrate a JAS created with the CREATE command.
4. Although the SYS.JADIR file is no longer mandatory, it is recommended to create SYS JAS on each system. It is mandatory if After Journalization is to be used.

3.1.1 Control JAS Files

For a description of these files refer to *JAS Files* (section 2.2).

JAS CATALOG

- SYS JAS uses SYS.CATALOG which is permanent on the system. SYS.CATALOG is created by TAILOR at system installation.
- BLUE.CATALOG and GREEN.CATALOG are private catalogs created when BLUE JAS and GREEN JAS, respectively, are created by the MAINTAIN_JAS processor. Both catalogs:
  - are auto-attachable,
  - in an HA context, must be allocated on *shared disks*.

JAS Directory

- d creates the SYS.JADIR and primary journal files for the first time. If SYS.JADIR is to be deleted and created again, the MNJAS CREATE command must be used.
- BLUE.JADIR and GREEN.JADIR as well as the After Journal files for the BLUE and GREEN JAS are created with the MNJAS CREATE command. In an HA context, they must be allocated on *shared disks*.

3.1.2 Files Specific to Before Journalization

SYS.JRNAL

- In each system, the three JAS use ***one*** Before Journal file called SYS.JRNAL. This file is allocated by the TAILOR JBEFORE command at system installation and is cataloged in the SYS.CATALOG.
- There are two SYS.JRNALs on coupled systems, one per member. However, they *must* be allocated on disks which are:
  - *different* and distinct,
  - and *shared* (for HA) or *sharable* (to run TCRF).
- Default values for the SYS.JRNAL file size and block size are provided by the GCOS 7 installation tools GIUF and TAILOR as shown in the table below:

Table 3-1. SYS.JRNAL Default File Size and Block Size

Disk Type	File Size	Block Size
VBO: MS/D500	21 cylinders * (17136 blocks x 512 bytes)	512 Bytes
MS/B10	28 cylinders ** (16800 blocks x 512 bytes)	512 Bytes
FBO: MS/D500	3720 blocks	4096 Bytes
MS/B10	3720 blocks	4096 Bytes
MS/FSA	16368 blocks	512 Bytes

*	One cylinder of an MS/D500 disk supports: 816 blocks x 512 bytes, or 144 blocks x 4096 bytes
**	One cylinder of an MS/B10 disk supports: 600 blocks x 512 bytes, or 120 blocks x 4096 bytes

These values are tailored to a BJSIMU of 124.

- When the specified Before Journal block size is 512 bytes, (132 x BJSIMU) blocks are needed.

- When the specified Before Journal block size is 4096 bytes, (30 x BJSIMU) blocks are needed.
- The JAS administrator may change the default size of the SYS.JRNAL file, by reallocating the file and then executing a Cold (or Clean) System Restart.

3.1.3 Files Specific to After Journalization

Primary and Secondary After Journal Files

- The *After Journal* files are created through the MNJAS CREATE command. The parameters in CREATE determine the resources of the JAS. These resources can later be modified through the MODIFY_PARAMETERS and MODIFY_MEDIA commands.
- primary and secondary After Journal files are attached to each JAS. These files are created and cataloged in the *jasname*.CATALOG:
  - either when the JAS is created by the JAS administrator,
  - or subsequently through the ADDPRIMMD and/or ADDSECMD options of the MODIFY_MEDIA command.
- The name of the file set is *jasname*.JA.Ji*medianame*, where i=1 through 8, the file set being used to store the After Images.
- primary After Journals are cataloged with SHARE=ONEWRITE.
- BLUE.JA.Ji*medianame* and GREEN.JA.Ji*medianame* must be allocated on *shared disks*.
- The disk device types for After Journal Files are MS/FSA, MS/D500 and MS/B10.
- *primary After Journal Files can only be on disks.*

3.1.4 Specific HA Files

SYS.JASBLUE and SYS.JASGREEN Files

- For BLUE JAS, two SYS.JASBLUE files are created, one on each system (member) of the complex. Similarly, for GREEN JAS, two SYS.JASGREEN files are created.
- The SYS.JASBLUE and the SYS.JASGREEN files created on one system are exclusive to that system. These files must ***not*** be shared between the two system (SHARE=DIR, DUALSHR=NORMAL) and **must** be allocated on different disks. Since these files are system files, their disk format *can* be VBO or FBO.
- These files can be created:
  - either by the TAILOR JAS command where the volume specified can be SYSVOL, RSDVOL (RSDVOL1) or NRDVOL (NRDVOL1)
  - or by the GIUF BUILD_JAS command which allocates the files on the P-set or P2-set.
- The SYS.JASBLUE and the SYS.JASGREEN files contain information about the state of the HA-type JAS on the specific system concerned. They are created by TAILOR at system installation and cataloged in SYS.CATALOG, with the following characteristics:

- *For FBO disks:*
  - Organization:
    ```
    BFAS = LINKQD
    BLKSIZE = 400
    RECFORM = VB
    RECSIZE = 200
    DIRSIZE = 1
    (INCRSIZE = 0, MAXSIZE =0)
    ```
  - Allocation:
    ```
    UNIT = BLOCK
    SIZE = 3
    ```
- *For VBO disks:*
  - Organization:
    ```
    BFAS = LINKQD
    BLKSIZE = 400 RECFORM = VB RECSIZE = 200
    DIRSIZE = 1
    (INCRSIZE = 0, MAXSIZE =0)
    ```
  - Allocation:
    ```
    UNIT = CYL SIZE = 1
    ```

3.2 JAS Maintenance

The JAS is administrated through the MNJAS processor. MNJAS enables creating, modifying and deleting all the JAS files. MNJAS also enables managing the user files protected by the JAS. Some MNJAS commands therefore cannot be used while journalization is active.

MNJAS processor is described in section *Maintaining the JAS* (section 3.2.3) to section *JAS Administrative Commands* (section3.2.5). Before using MNJAS commands it is often useful to get information about the activity of the JAS. The operator command DISPLAY_JAS (DJAS) lists the steps connected to each JAS. See *DISPLAY_JAS Command (DJAS)* (section 3.2.2).

At system restart, if the SYS.JADIR file which exists cannot be accessed, the message JP41 is sent to warn that no journalization can take place during the GCOS session. In this case, none of the MNJAS commands can be executed.

3.2.1 The JAS Administrator

The JAS Administrator is in charge of managing a JAS. When a system is installed with ccess rights, the JAS Administrator is the only one who can read and update any information in a JAS. The JAS administrator must have the SYSADMIN project rights.

Recovering user files requires the access right RECOVERY to these files.

Access to the TDS user journal is also controlled by the DUMPJRNL utility. Refer to *Access Rights* (section 9.2.4).

A JAS can protect user files belonging to several projects cataloged in auto-attachable private catalogs.

From Release V6 onwards, the JAGEN utility has been replaced by the MNJAS utility to modify the options of the After Journals.

If the system has its access rights set through the GIUF function PROTECT_GCOS, the JAS Administrator with SYSADMIN rights is the only user able to update information on the JAS files, and to maintain these files. See the *System Installation, Configuration and Updating Guide*.

3.2.2 DISPLAY_JAS Command (DJAS)

DISPLAY_JAS is a GCOS 7 command available to the operator and is entered at S: level. It can be issued for one specified JAS, or for all the JAS on the system by default.

DISPLAY_JAS displays jobs that are *connected* to the JAS at the moment the command is issued. It gives for the specified JAS and for each connected job:

- the RON (run occurrence number),
- the job name,
- the name of the submitter,
- the job class,
- and the load module in execution.

Using DISPLAY_JAS is particularly recommended:

- before starting a *weak* takeover, to list the TDS, Batch and IOF applications so as to select the jobs to be terminated,
- before using the TRANSFER_PRIMARY command, see *TRANSFER_PRIMARY* (section 3.2.5.17),
- before using an MNJAS command, which requires the JAS to be in the EMPTY state for which the JAS must be terminated through the TSRV command.

Syntax

```
{ DISPLAY_JAS }
{             }
{ DJAS        }

   [          { alljas             }]
   [ JAS_NAME={                    }]
   [          { SYS | BLUE | GREEN }]
```

Parameter Description

JAS_NAME:	name of the JAS for which the list of connected step is required. *Default:* all.

EXAMPLE:

Consider the following configuration:

- TDS1 is an HA-type TDS running on BLUE JAS,
- BAT1 is a Batch application running on SYS JAS,
- and BAT2 is a Batch running on BLUE JAS.

The effect of the DJAS will be:

```
S: DISPLAY_JAS   JASNAME=BLUE;
   time JAS:BLUE   X151.1  TP7JCLAC  OPERATOR P  TDS1
                   X168.2  BAT2      USER2    P  H_BAT2

S: DJAS;
   time JAS:SYS    X167.5  BAT1      USER1    P  H_BAT1
        JAS:BLUE   X151.1  TP7JCLAC  OPERATOR P  TDS1
                   X168.2  BAT2      USER2    P  H_BAT2
        JAS:GREEN  NO JOBS LINKED TO THIS JAS
```

❑

3.2.3 Maintaining the JAS

MNJAS commands are listed with a resume of their functions. These commands allow the JAS Administrator to execute all the ordinary functions required to install and maintain JAS. The user, however, can only issue some of them. Commands are classified according to how they manage user files or JAS services. In each category, they are listed in alphabetical order.

Administration of JAS Resources

CREATE: allocates

- a JAS catalog except for SYS JAS
- a JAS directory cataloged in the JAS catalog
- and journal files to the JAS.

DELETE: deallocates

- Journal files of a JAS
- the JAS directory
- and the JAS catalog except for the SYS JAS.

LIST: gives information on the After Journal. This option is allowed to all users.

MODIFY_MEDIA: adds and removes After Journal Media.

MODIFY_PARAMETERS: modifies After Journal files Parameters.

REBUILD_DIRECTORY: reconstitutes the directory for the JAS concerned.

RECYCLE_JOURNAL_FILE: switches an After Journal file from *active* state to *available*.

TRANSFER_PRIMARY: moves primary After Journal files to secondary after Journal files.

SWITCH: switches primary file of the JAS concerned for the next journalization session.

Administration of User Files

DISPLAY_LINK: display the name of the JAS linked to a private catalog.

FORGET_USER_FILE: cancels a user file.

FORGET_USER_JOURNAL: cancels a TDS user journal.

LINK: links a private catalog to a JAS.

JAS Services

DISPLAY: displays the current default JAS name.

JAS: changes the default JAS name.

QUIT: quits the MNJAS utility.

STATUS: determines how JAS is to proceed when an error is detected.

The user is allowed the following commands:

- LIST with DETAILED=0 option
- DISPLAY
- DISPLAY_LINK
- JAS
- QUIT

NOTES:

1. All the commands described are available both in interactive and batch mode. STATUS has no effect in interactive mode except if COMFILE is used.
2. MNJAS is activated through either the GCL MAINTAIN_JAS command or the JCL JASMAINT command.
3. When a command is executed, a JOR and a Command Report are produced in Sysout or PRTFILE for batch mode processing. In interactive mode, the report is sent:

   - to the console and to the Sysout, if no PRTFILE is used
   - but only to the PRTFILE if it is used.

The following table summarizes the correspondence between MNJAS commands of the V6 Release and the equivalent JAGEN commands used in previous releases.

Table 3-2. MNJAS and JAGEN Commands Correspondence

MAINTAIN_JAS Commands	Old JAGEN Commands
CREATE	GEN1, APPEND1, GEN2, APPEND2
DELETE	CLEAR
FORGET_USER_FILE	DELETE
FORGET_USER_JOURNAL	DELETE
LIST	PRINT
MODIFY_MEDIA	GEN1, APPEND1, APPEND2, REMOVE
MODIFY_PARAMETERS	GEN1,GEN2
QUIT	no corresponding command
RECYCLE_JOURNAL_FILE	RECYCLE
TRANSFER_PRIMARY	TRANSFER

NOTE:

***DISPLAY_LINK**, **LINK**, **DISPLAY**, **JAS**, **REBUILD_DIRECTORY** and **SWITCH** commands do not have JAGEN equivalents.*

EXAMPLES OF USE:

- The first time when creating journal files and giving the characteristics of the After Journal, specify:
```
CREATE SYS jadir_size=...,jadir_volume=...,journal_size=...,
           primary_media=...,primary_dvc=...,
           secondary_media=...,secondary_dvc=...;
```
- To recycle selected After Journal files, specify one of the following:
```
RECYCLE LASTDATE=...
RECYCLE LASTFILE=...
RECYCLE OBSOLETE
```
- To change the blocksize of the journal, specify:
```
TRANSFER or RECYCLE LASTDAY=TODAY;
MODIFY_MEDIA REMPRIMMD='list_of_primary_media';
MODIFY_PARAMETERS BLOCK_SIZE=...;
MODIFY_MEDIA ADDPRIMMD='list_of_primary_media';
```
- To pass from tape to cartridge for secondary files, specify:
```
RECYCLE LASTFILE=...;
MODIFY_MEDIA REMSECMD='list_of_secondary_media';
MODIFY_PARAMETERS SECONDARY_DVC=CT/...;
MODIFY_MEDIA ADDSECMD='list_of_secondary_media';
```

❑

3.2.4 Accessing the MAINTAIN_JAS Utility

The MNJAS command allows the JAS Administrator to execute all functions to maintain a JAS and allows users to execute some functions concerning their files.

JAS administrator must have the security access rights of the project SYSADMIN.

Users without SYSADMIN rights may only execute the DISPLAY, DISPLAY_LINK, JAS, LIST and QUIT commands.

GCL Syntax

```
{ MAINTAIN_JAS }
{              }
{ MNJAS        }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   [{ COMFILE=file78  }]
   [{                 }]
   [{ COMMAND=char255 }]

   [ PRTFILE=file78 ]
```

JCL Syntax

```
{ JASMAINT }
{          }
{ MNJAS    }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   [ { COMFILE=file-78 }]
   [ {                 }]
   [ { COMMAND=char255 }]

   [ PRTFILE=file78 ]
```

Parameter Description

JAS_NAME:	Name of the JAS to be managed. SYS JAS is allocated by default when invoking the MNJAS processor.
COMFILE:	File containing the MNJAS commands. COMFILE is mutually exclusive to COMMAND.

COMMAND:	Character string containing the MNJAS commands. COMMAND is mutually exclusive to COMFILE.
PRTFILE:	File receiving the MNJAS report. When specified, no report appears at the console in interactive mode.

3.2.5 JAS Administrative Commands

Once the MNJAS processor is activated, the prompt M: appears and the MNJAS commands can then be entered.

Commands are listed in alphabetical order.

3.2.5.1 CREATE

CREATE performs the following functions:

- allocates the JAS catalog (for BLUE JAS or GREEN JAS only),
- allocates the JAS directory cataloged in the JAS catalog,
- completes the creation of a JAS by:
  - setting the JAS generation options such as Cycling Mode, primary and secondary *After Journal* Files device class, space for *After Journal* files per media,
  - preallocating and cataloging the primary *After Journal* files,
  - and updating the list of available primary and secondary *After Journal* files by adding the new *After Journal* files in the directory.

For SYS JAS, the first Create function is done by TAILOR during the system configuration.

This command must be used *once and only once* for the complete generation of a JAS (Public SYS JAS, Private BLUE JAS and Private GREEN JAS). If the CREATE command is to be issued on the same JAS, the command DELETE must first be used before CREATE (otherwise the CREATE command will be rejected).

NOTE:
If the DELETE function does not work correctly as may happen, for instance, if the journal is not in a consistent state, the JAS administrator must delete all the JAS files one by one. See *DELETE* (section 3.2.5.2).

Syntax

```
{ CREATE }
{        }
{ CR     }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   [{ JASCAT_VOLUME }                              ]
   [{               }=name6:MS/{ FSA | D500 | B10 }]
   [{ CATV          }                              ]

   [{ JADIR_SIZE }               ]
   [{            }={ 120 | dec4 }]
   [{ DIRSZ      }               ]

   [{ JADIR_VOLUME }                              ]
   [{              }=name6:MS/{ FSA | D500 | B10 }]
   [{ DIRV         }                              ]

   [{ JOURNAL_SIZE }              ]
   [{              }={ 80 | dec4 }]
   [{ JRNLSZ | JSZ }              ]

   [{ BLOCK_SIZE }                ]
   [{            }={ 8192 | dec4 }]
   [{ BLKSZ      }                ]

   [{ PRIMARY_MEDIA }                      ]
   [{               }=( name6 [,name6 ]...)]
   [{ PRIMMD | PMD  }                      ]

   [{ PRIMARY_DVC    }                         ] see Parameter
   [{                }=MS/{ FSA | D500 | B10 }] Description
   [{ PRIMDVC | PDVC }                         ]

   [{ SECONDARY_MEDIA }                      ]
   [{                 }=( name6 [,name6 ]...)]
   [{ SECMEDIA | SMD  }                      ]

   [{ SECONDARY_DVC } { MT/Td/[ Ddddd ]}]
   [{               }={                }]
   [{ SECDVC | SDVC } { CT/LIB | CT/xx }]

   [{ AUTOCYCLE }             ]
   [{           }={ 1 | bool }]
   [{ AUTOCY    }             ]

   [{ SECDCOPY      }             ]
   [{               }={ 0 | bool }]
   [{ SECDCP | SDCP }             ]
```

Parameter	**Description**
JAS_NAME:	name of the JAS to be created. The default is the *jas_name* previously specified when either invoking the MNJAS processor or submitting the MNJAS JAS command.
JASCAT_VOLUME:	volume to be used for allocating the JAS catalog. For BLUE and GREEN JAS, this parameter is mandatory. For SYS JAS, this parameter is ignored.
JADIR_SIZE:	space to be allocated to the JAS directory in units of 100 Kbytes. The default is 120 x 100 Kbytes. The file created has an increment size of 5 x 100 Kbytes.
JADIR_VOLUME:	volume to be used for allocating the JAS directory. The only device type allowed is disk. If no value is provided, *jas_name*.JADIR file will be allocated on a resident disk.
JOURNAL_SIZE:	maximum space to be allocated on each primary media to the primary journal files in units of cylinders. The default is 80 cylinders. The minimum accepted value is 8 cylinders. This size is the global amount of space reserved on each media for the primary After Journal files. The system creates up to 8 primary files and the specified amount of space is divided among them.
BLOCK_SIZE:	maximum size of the records containing the After Images. It must have a value in the range of 512 bytes through 32000 bytes. Refer to *Choosing the Size of After Journal Block* (section 2.2.4.2).
PRIMARY_MEDIA:	the list of primary journal file media. A maximum of 20 media may be provided. ***At least one media must be specified.***
PRIMARY_DVC:	device class for supporting primary journal files. From TS7254 onwards, only disks are supported for all JAS.

SECONDARY_MEDIA:	the list of secondary journal file media with their associated device class. A maximum of 20 media may be provided. If SECDCOPY is used, only the name of the first copy need be specified and the last character of this secondary media name must be odd. The name of the second copy is deduced from the name of the first copy. The ending odd number is replaced by the following even number. The name of a media is a string of 1 to 6 characters.
SECONDARY_DVC:	external form of device class (device type allowed is tape or cartridge or cartridge library) for supporting secondary journal files.
AUTOCYCLE:	defines the cycling option for the Journal file.
Automatic	AUTOCYCLE=1 (default) means that active journal files are set in the list of available journal files if they no longer contain any valid After Images.
Manual	AUTOCYCLE=0 means that the cycling option will be specified in the RECYCLE_JOURNAL_FILE command.
SECDCOPY:	Secondary copy (dual copy) for secondary Journal Files. The names of the copies are deduced from the parameter SECONDARY_MEDIA.

Constraints

- HA-type JAS must be in the EMPTY state. See *HA Concepts*.
- If *jas_name* is BLUE or GREEN, JASCAT_VOLUME is mandatory.
- If the SECONDARY_MEDIA parameter is used, the SECONDARY_DVC parameter is mandatory.
- SECONDARY_MEDIA and PRIMARY_MEDIA must be different media.
- JADIR_VOLUME must have a disk device type.
- Mimimum JOURNAL_SIZE is 8 cylinders.
- Minimum BLOCK_SIZE is 512 bytes and its maximum is 32000 bytes.
- For PRIMARY_DVC, the device type must be disk. Cartridges are ***NOT*** supported and from TS7254: tape is no longer allowed.

- If the JAS has already been created, the DELETE command must first be successfully performed on the JAS concerned before the CREATE command can be reissued.
- *On coupled systems, the media for JADIR_VOLUME and PRIMARY_MEDIA must be* ***different*** *for the two SYS JAS.*
- The CREATE command is reserved for the SYSADMIN project.

Messages Issued

```
CR jas_name PERFORMED
CR jas_name FAILED : ERROR DURING INSTALLATION JOB EXECUTION
CR jas_name REJECTED : FUNCTION RESERVED FOR SYSADMIN
```

EXAMPLES:

```
1. CR JAS_NAME=BLUE
   CATV=(disk0:ms/d500),DIRV=disk1:ms/b10
   PMD=(disk2,disk3,disk4),PDVC=ms/d500
   SMD=(tap1,tap2,tap3),SDVC=mt/t9/d6250,AUTOCY=0;
```

- BLUE JAS is created with:
  - BLUE.CATALOG on disk0
  - 120 x 100 Kbytes for BLUE.JADIR on disk1
  - 80 cylinders for the primary journal files on disk2, disk3 and disk4
  - and secondary journal files on tap1, tap2 and tap3
  - the cycling mode of the journals being manual
  - a single copy being used for secondary journal files.

```
2. CR JAS_NAME=GREEN
   CATV=(md1:ms/d500),DIRSZ=10,DIRV=md2:ms/b10
   JOURNAL_SIZE=30
   PMD=(md3,md4),PDVC=ms/d500;
```

- GREEN JAS is created with:
  - GREEN.CATALOG on media md1
  - 10 x 100 Kbytes for the journal directory GREEN.JADIR on media md2
  - 30 cylinders for primary journal files on the disks md3 and md4
  - the cycling mode for the journals being automatic
  - no secondary journal file is defined.

```
3. CR JAS_NAME=GREEN
   CATV=(md0:ms/d500),DIRV=md1:ms/d500
   PMD=(md2,md3,md4),PDVC=(ms/b10)
   SMD=(secA1,secB1),SDVC=(mt/t9/d6250),SDCP=1;
```

- GREEN JAS is created with;
  - GREEN.CATALOG on media md0
  - 120 x 100 Kbytes for the journal directory GREEN.JADIR on media md1
  - 80 cylinders for primary journal files on disks md2, md3 and md4
  - the cycling mode for the journals being automatic
  - and secondary journal files on tapes secA1, secA2, secB1 and secB2; secA2 being the copy of secA1 and secB2 the copy of secB1.

```
4. CR JAS_NAME=SYS
   DIRSZ=10,PMD=md1,PDVC=ms/d500;
```

- SYS JAS is created with:
  - 10 x 100 Kbytes for the journal directory SYS.JADIR allocated on resident disk
  - 80 cylinders for primary journal files on media md1
  - the cycling mode for the journals being automatic
  - and no secondary journal file is defined.

In this example for SYS JAS, no catalog is allocated and the directory is allocated on a resident disk.

❑

3.2.5.2 DELETE

DELETE performs the following functions:

- deletes the primary journal files and clears the list of available journal files by deleting these journal files from the JAS directory, whatever the JAS,
- removes the *jasname*.JA directory from the JAS catalog,
- deletes the JAS directory whatever the JAS,
- for BLUE JAS and GREEN JAS:
  - deallocates the JAS catalog,
  - and removes the *jasname* directory from SITE.CATALOG.

Syntax

```
{ DELETE }
{        }
{ DL     }

   [ JAS_NAME={ SYS | BLUE | GREEN }]
```

Parameter Description

JAS_NAME:	name of the JAS to be deleted. The default is the *jas_name* previously specified when either invoking the MNJAS processor or submitting the MNJAS JAS command.

Constraints

- The DELETE command can be executed only when the *jas_name* JAS is stopped. That is, when the HA-type JAS is in the EMPTY state or when no journalization is running on SYS JAS.
- To execute the command, all primary and secondary journal files must be available. If there are any journal files still active, first issue the `RECYCLE_JOURNAL_FILE LASTDATE=TODAY` command.
- The DELETE command is reserved for the SYSADMIN project.

Messages Issued

```
DL jas_name PERFORMED.
DL jas_name REJECTED  : JOURNALIZATION IS ACTIVE.
DL jas_name FAILED : ERROR DURING INSTALLATION JOB EXECUTION.
DL jas_name REJECTED : FUNCTION RESERVED FOR SYSADMIN
```

EXAMPLES:

1. `DL JAS_NAME=BLUE;`

- The following actions are taken:
  - clears the list of available journal files for BLUE JAS
  - deletes all primary journal files
  - deletes BLUE.JADIR
  - deletes BLUE.CATALOG
  - and deletes the directories BLUE and BLUE.JA.

```
2. DL JAS_NAME=SYS;
```

- The following actions are taken:
  - clears the list of available journal files for SYS
  - deallocates all primary journal files
  - deallocates the SYS.JADIR and uncatalogs it.

NOTE:
If the DELETE command does not function correctly, the administrator must delete all the files and the directory of the JAS one by one, then reissue DELETE, as shown in the examples below:

❑

EXAMPLE: for SYS JAS

```
1. DELETE_FILE SYS.JADIR BYPASS;

2. LSDIR SYS.JA;
```

- lists all the journal files to be deleted, see Paragraph 3.

```
3. DELETE_FILE SYS.JA.J1media-name
              .
              .
              .

4. DELETE_DIR SYS.JA;

5. MAINTAIN_JAS COMMAND='DELETE SYS;';
```

❑

EXAMPLE: for BLUE JAS

```
1. DELETE_FILE BLUE.JADIR BYPASS;

2. LSDIR BLUE.JA;
```

Lists all the journal files to be deleted.

```
3. DELETE_FILE BLUE.JA.J1media-name
              .
              .
              .

4. DELETE_DIR BLUE.JA;

5. DELETE_CATALOG BLUE;

6. DELETE_DIR BLUE;

7. MAINTAIN_JAS COMMAND='DELETE BLUE;';
```

❑

3.2.5.3 DISPLAY

DISPLAY displays the name of the current JAS in the MNJAS processor. See example.

Syntax

```
{ DISPLAY }
{         }
{ D       }
```

Parameter Description

None

Constraints

None

Messages Issued

```
    CURRENT JAS : { SYS | BLUE | GREEN }
```

EXAMPLE:

```
1. MAINTAIN_JAS BLUE ;
   D ;
```

The DISPLAY command displays `CURRENT JAS : BLUE`.

```
2. MAINTAIN_JAS ;
   D ;
```

The DISPLAY command displays `CURRENT JAS : SYS`.

❑

3.2.5.4 DISPLAY_LINK

DISPLAY_LINK displays the name of the JAS linked to the catalog and therefore protecting all the files of the catalog. If the catalog is not auto-attachable, its file can only be protected by SYS JAS.

Syntax

```
{ DISPLAY_LINK }
{              }
{ DPLK | DLK   }

   { CATNAME     }
   {             }=file78
   { CAT         }
```

Parameter Description

CATNAME:	name of the catalog for which the command will display the name of the JAS protecting it.

NOTE:

In an HA environment, when two SITE.CATALOGs are used, you are recommended to execute the command DISPLAY_LINK on both systems for each user file catalog, so as to check that MNJAS LINK has been executed on both systems for each catalog.

If MNJAS LINK has been executed on only one system, the inconsistency will be detected on the other system, and DISPLAY_LINK will issue a warning. If a TDS is started in the ACTIVE state on the system where the MNJAS LINK was last executed, no problem will be detected. But the inconsistency will prevent file opening at TDS restart on the other system on takeover.

Constraints

- The catalog must be created with the option ATTACH=AUTO.
- The argument of CATNAME takes the format *name*.CATALOG.
- No link can be created for SYS.CATALOG and SITE.CATALOG.

Messages Issued

```
catname IS LINKED TO JAS { SYS | BLUE | GREEN }
DPLK REJECTED : CATALOG NOT FOUND
catname HAS INCONSISTENT LINK: RETYPE LINK COMMAND.
DPLK REJECTED: CATALOG NOT AUTO-ATTACHABLE
DPLK REJECTED: COMMAND NOT AVAILABLE FOR A SYSTEM CATALOG
```

EXAMPLE:

```
DLK CATNAME = mycat ;
```

Displays the name of the JAS to which the mycat catalog is linked.

❑

3.2.5.5 FORGET_USER_FILE

FORGET_USER_FILE deletes a user file entry in a JAS directory.

The After Images related to a user file, stored before the execution of the FORGET_USER_FILE command, will no longer be related to this user file. The user file becomes unknown to the After Journal. This function is to be used only for user files which are no longer required by the user.

By default, the command verifies if there are any valid After Images for the user file. The command verifies that the file has not been opened in update mode since the last SAVE done with:

- either the FILSAVE utility without the EXPORT option,
- or the FILDUPLI utility with the UPDJRNL option.

The JAS administrator is advised not to use the FORCE option. If FORCE=1 is specified, no control is done and an updated user file may remain unrecoverable because its After Images would have been invalidated.

NOTE:
This function recycles active journal files if auto cycling mode is applied.

Syntax

```
{ FORGET_USER_FILE }
{                  }
{ FGTUF            }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   {{ FILE | F }={ file78 | * }}
   {{                          }
   {{ OLDGEN={ 0 | bool }      }

   [{ CATNAME }        ]
   [{         }=file78 ]
   [{ CAT     }        ]

   [ FORCE={ 0 | bool } ]
```

Parameter Description

JAS_NAME:	name of the JAS to be processed. The default is the *jasname* specified previously when either invoking the MNJAS processor or submitting the MNJAS JAS command.
FILE:	name of the user file entry which is to be cancelled. If * is specified, all files cataloged in the *catname* catalog will be *forgotten*. CATNAME is mandatory if FILE is specified. *FILE is mutually exclusive with OLDGEN.*
OLDGEN:	deletes all user files which are in the KEPT state. KEPT is a state of a user file entry in relation to the file generation group of the catalog. *OLDGEN is mutually exclusive with FILE.*
CATNAME:	name of the catalog in which the file is or was cataloged. FILE is mandatory if CATNAME is specified.

FORCE:	Action for the validity of a file:
=0	*Default:* Verifies that the last open date in update mode known by the *jas_name* JAS is older than the last known save date for the files specified in the FILE parameter. If not, there are probably valid After Images remaining in the journal files and the file for which these images are intended is not forgotten. ***FORCE=0 is always recommended.***
=1	***No control is done and the files specified in FILE are forgotten. NO recovery is possible if an incident occurs for a forgotten user file.***

Constraints

- The function is available whether journalization is active or not.
- For BLUE JAS and GREEN JAS, the function is available only when the JAS is in the "active" state.
- CATNAME is mandatory if FILE is specified, and vice versa. It must have the format *name*.CATALOG.
- FILE and OLDGEN are mutually exclusive.
- The FORGET_USER_FILE command is reserved for the SYSADMIN project.

Messages Issued

```
FGTUF  jas_name REJECTED : service_name NOT ACTIVATED
FGTUF  jas_name REJECTED : FUNCTION RESERVED FOR SYSADMIN
```

EXAMPLES:

1. `FGTUF  JAS=BLUE FILE=`*`myfile`*` CAT_NAME=`*`myfile_catalog`*`;`

   The entry of the user file *myfile* cataloged in *MYFILE_*CATALOG is deleted in the BLUE.JADIR, only if *myfile* was not opened in update mode since its last save.

2. `FGTUF  OLDGEN;`

   All the files in the KEPT state in the journal directory JADIR of the current JAS are deleted.

3. `FGTUF  JAS=SYS FILE=`*`myfile`*` CAT_NAME=`*`myfile_catalog`*` FORCE;`

   The entry of the user file *myfile* cataloged in *MYFILE_*CATALOG is deleted in the SYS.JADIR and no control is performed.

4. `FGTUF  JAS=GREEN FILE=* CAT_NAME=`*`myfile_catalog`*`;`

   All the entries for files cataloged in *MYFILE*_CATALOG and not opened in update mode since their last save, are deleted in the GREEN.JADIR.

❑

3.2.5.6 FORGET_USER_JOURNAL

When a TDS user journal no longer exists from the TDS user's point of view, the entry in the JAS directory can be deleted by issuing the FORGET_USER_JOURNAL command.

As a consequence of the introduction of the function XDUMPJRNL, for integrity reasons, the system considers that there might be valid data for each user journal after the last extraction. This implies that the FORGET_USER_JOURNAL command of MNJAS does not work without the FORCE parameter. The system administrator is advised to check whether or not the user journal to be "forgotten" has been opened since the last execution of the DUMPJRNL or XDUMPJNRL utility.

NOTE:
This function recycles active journal files, if autocycling mode is applied.

Syntax

```
{ FORGET_USER_JOURNAL }
{                     }
{ FGTUJRNL            }

   [{ JAS_NAME={ * | SYS | BLUE | GREEN }]

   TDS=name4

   [{ FORCE }={ 0 | bool }]
```

Parameter Description	
JAS_NAME:	name of the JAS to be processed.
	The command will only delete the TDS user journal in the JAS specified.
	If * is specified, the command will delete the TDS user journal in ***ALL*** SYS JAS, BLUE JAS and GREEN JAS.
	If no value is provided, the entry of the TDS user journal will be deleted in the directory of whichever JAS is current.
	To find the JAS name in which the data are logged, issue the LIST command to all JAS versions known on the site. Each LIST will display the names of the TDS which log data in their user journals.
TDS:	name of the TDS whose user journal is to be cancelled.
	The data related to a TDS user journal that has been stored before the execution of FORGET_USER_JOURNAL will no longer be associated to this TDS user journal and cannot therefore be extracted by the DUMPJRNL utility.
FORCE:	Action for the validity of a file:
=0	*Default:* Checks that the last open date of the TDS user journal known to *jasname* JAS is later than the date of the last known execution of DUMPJRNL for this TDS. If the last open date is earlier, the user journal is not forgotten since there are probably records still valid for the TDS user journal in the journal files. So for protection, use FORCE=0.
=1	***No control is done and the TDS user journal is forgotten.*** User images can no longer be extracted by DUMPJRNL.

Constraints

- The function is available whether journalization is active or not.
- For BLUE JAS and GREEN JAS, the function is available only when the JAS concerned is in the "active" state. If * is specified, the function is processed for SYS JAS, for BLUE JAS only if it is in the "active" state, and for GREEN JAS only if it is in the "active" state.
- The FORGET_USER_JOURNAL command is reserved for the SYSADMIN project.

Messages Issued

```
FGTUJRNL  jas_name REJECTED : service_name NOT ACTIVATED.
FGTUJRNL  jas_name REJECTED : FUNCTION RESERVED FOR SYSADMIN
```

EXAMPLES:

1. `FGTUJRNL  TDS=mytp;`

   FORGET_USER_JOURNAL finds the *jasname* JAS concerned with the *mytp* user journal and deletes its entry in the *jasname*.JADIR only if the *mytp* user journal was not opened since the last execution of DUMPJRNL utility for *mytp*.

2. `FGTUJRNL  JAS_NAME=GREEN  TDS=mytp;`

   FORGET_USER_JOURNAL looks for the entry of the *mytp* user journal in GREEN.JADIR and deletes it only if the *mytp* user journal was not opened since the last execution of DUMPJRNL utility for *mytp*.

3. `FGTUJRNL  JAS_NAME=*  TDS=mytp;`

   FORGET_USER_JOURNAL looks for the entry of the *mytp* user journal in:

   – SYS.JADIR
   – BLUE.JADIR if BLUE was started then GREEN.JADIR if GREEN was started and deletes it each time it is found if the *mytp* user journal was not opened since the last execution of DUMPJRNL utility for *mytp*.

4. `FGTUJRNL  TDS=mytp  FORCE;`

   FORGET_USER_JOURNAL finds the *jasname* JAS concerned with the *mytp* user journal and deletes its entry in the *jasname*.JADIR. No control is performed.

❑

3.2.5.7 JAS

JAS changes the current JAS in the MNJAS processor.

Syntax

```
JAS

   [ JAS_NAME={ SYS | BLUE | GREEN }]
```

Parameter Description

JAS_NAME:	name of the JAS which becomes the current value. If no name is specified, the current *jasname* is not modified.

Constraints

A current JAS should be defined before entering any other MNJAS command with default value for JAS_NAME.

Messages Issued: none

EXAMPLE:

```
JAS JAS_NAME=BLUE;
```

BLUE JAS now becomes the current JAS.

❑

3.2.5.8 LINK

LINK links the user files of a private catalog to a specific JAS thereby protecting all the user files in the catalog. A private catalog when first created is linked to SYS JAS. LINK is subsequently used to change the protection of user files in a private catalog from one JAS to another.

The JAS administrator must ensure that the files in the private catalog are no longer known to the ***current*** JAS linked to the catalog before issuing the LINK command to link the catalog to another JAS.

EXAMPLE:

The catalog *mycat* is initially linked to SYS JAS and the JAS administrator wants to link it to BLUE JAS.

Solution: The procedure recommended is as follows:

- First issue *LIST SYS DETAILED* to get the list of all user files known to SYS JAS.
- If any files cataloged in *mycat* are listed, issue *FORGET_USER_FILE JAS_NAME=SYS FILE=* CATNAME=mycat;*. See note below.
- Then issue *LINK JAS_NAME=BLUE CATNAME=mycat*;.

NOTES:

1. Files which have their last open date in update mode known by SYS JAS older than their last known save date, will not be deleted by FORGET_USER_FILE. Execute the FILSAVE utility on such files to change their date, before reissuing FORGET_USER_FILE. Issue the LINK command, only when all the files in the catalog are no longer known to SYS JAS.
2. In an HA environment, when two SITE.CATALOGs are used, the LINK command must be executed on both systems for each of the catalogs to ensure their consistency. Otherwise file opening on one of the systems will be prevented with the code JAP 3, SHLVVIOL being returned.

❑

Syntax

```
{ LINK }
{      }
{ LK   }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   { CATNAME }
   {         }=file78
   { CAT     }
```

Parameter Description

JAS_NAME:	name of the JAS to be linked to a private catalog. The default is the *jas_name* specified previously when either invoking the MNJAS processor or submitting the MNJAS JAS command.
CATNAME:	name of a private catalog containing user files to be protected.

Constraints

- The private catalog must be created with the option ATTACH=AUTO.
- The argument of CATNAME takes the format: *name*.CATALOG.
- The LINK command is reserved to SYSADMIN project.
- The SITE.CATALOG and the SYS.CATALOG which is reserved for system files cannot be linked to a JAS. The files cataloged in the SITE.CATALOG can only be protected by SYS JAS.
- For the LINK command to work properly, the catalog to be linked to JAS must not be in use by another step; for example, LINK cannot be used on a TDS catalog if it is in operation.

Messages Issued

```
catname HAS BEEN LINKED TO jasname
LK jas_name REJECTED: CATALOG NOT FOUND
LK jas_name REJECTED: FUNCTION RESERVED FOR SYSADMIN
LK jas_name REJECTED: COMMAND NOT AVAILABLE FOR A SYSTEM CATALOG
```

EXAMPLES:

1. `LK  JAS=BLUE  CAT=mycat;`

   On successful completion of the command, all the files cataloged in *mycat* will be protected by BLUE JAS.

2. `LK  CAT=mycat;`

   On successful completion of the command, all the files cataloged in *mycat* will be protected by the current JAS.

❑

3.2.5.9 LIST

LIST gets information from the JAS directory. The available information is:

- the JAS characteristics,
- the list of available After Journal files (this includes files available for journalization and tapes or cartridges available for the TRANSFER_PRIMARY function),
- the list of active After Journal files (this includes the current After Journal file and the previous After Journal files), and for each file:
  - statistics about the filling of blocks,
  - the reason for switching to the next journal file,
- the list of journalized user files (or files in KEPT state) and, for each file, the last save date,
- the list of the TDS user journals and, for each one, the last date of DUMPJRNL,
- the percentage of currently used space in the JAS directory.

Parameters can be specified with the LIST command to retrieve specific elements of the available information.

The report is sent to the Sysout of the caller and, in interactive mode, to the console. Journalization continues during the printing of the report.

NOTES:

1. A Save date of a user file is its last Save date in the JAS directory only if the file has already been journalized in the JAS. SAVE must have been performed using:

   - either the FILSAVE utility without the EXPORT option

   - or the FILDUPLI utility with the UPDJRNL option.

2. During the execution of the FILREST utility, the request to test the date of the *save* is performed on the information in the JAS directory on the *last save* known to the JAS.

3. A user file for which After journalization has been inhibited, is flagged with an exclamation mark "!" preceding its name.

4. For each journalized user file, a character S or D is placed in front of the date of the last save. This character indicates whether the save was static (S) or dynamic (D).

Syntax

```
{ LIST }
{      }
{ LS   }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   [ { DETAILED }
   [ {          } = { 0 | bool }]
   [ { DTLD     }

   [ AVAILABLE = { 0 | bool } ]

   [ ACTIVE = { 1 | bool } ]

   [ TDS = { 0 | bool } ]

   [ FILES = { filename | prefix* | * } ]
```

Parameter Description

JAS_NAME:	name of the JAS to be listed. The default is the *jasname* specified previously when either invoking the MNJAS processor or submitting the MNJAS JAS command.
DETAILED:	if set to 1, all available information is supplied: – JAS characteristics – list of available After Journal files – list of active After Journal files – list of journalized (or in KEPT state) user files – list of TDS user journals – percentage of space used in the JAS directory. When DETAILED=1, the keywords AVAILABLE, ACTIVE, TDS, and FILES are ignored. The default value is DETAILED=0: detailed information is not supplied.
AVAILABLE:	when set to 1, list of available After Journal files is supplied: – primary After Journal files available for journalization – secondary After Journal files (tapes or cartridges) available for the TRANSFER_PRIMARY function The default value is AVAILABLE=0.

ACTIVE: if set to 1, the list of active After Journal files (the current and previous After Journal files) is supplied. For each file the following details are also given:
- statistics about the filling of blocks
- the reason for switching to the next After Journal file.

ACTIVE=1 is the default value.

TDS: if set to 1, the list of TDS user journals is supplied together with the date of the last DUMPJRNL in each case.

The default is TDS=0.

FILES: the value supplied with FILES specifies a selection in the set of journalized (or KEPT state) user files:
- FILES=’ ’ means no files selected
- FILES=*filename* means the user file filename is selected
- FILES=*prefix** means all user files whose name starts with prefix are selected
- FILES=* means all the files in the set are selected.

The last Save date is given for each user file selected. The default value is FILES=’ ’ (no files selected).

Constraints

- For BLUE JAS and GREEN JAS in HA context, execution of LIST requires that the JAS concerned be started. SYS JAS need not be started up for the command to apply.
- From TS7458, all IOF users have access to all information (no information is reserved for the SYSADMIN project).
- A temporary file is used for the Sysout. When the report is very large and if a problem occurs when it is processed, the message JA13 ERROR DURING PRINT SYSOUT appears at the console. Use a PRTFILE in interactive mode or run the command in batch mode.

Messages Issued

```
LS jas_name REJECTED : service_name NOT ACTIVATED.
```

EXAMPLES:

1. `LS JAS=SYS;` Displays the following information about the SYS JAS directory:
   - the JAS characteristics
   - the list of active After Journal files
   - the percentage of space currently used in the JAS directory.

2. `LS JAS=SYS FILES=CUSTOMER.A*;`
   Displays the information in the previous example plus information about all files beginning with CUSTOMER.A.

3. `LS JAS=GREEN DETAILED=1;`
   Displays all information about the GREEN JAS directory.

4. `LS DTLD;` Displays detailed information about the JAS directory currently invoked by MNJAS processor;

❑

EXAMPLE: of After Journal Description

```
               *****************************************
               *      AFTER JOURNAL DESCRIPTION        *
               *              NAME : SYS               *
               *****************************************
DATED : 09:16:26  DEC 21, 1993
-------
CURRENT CHARACTERISTICS
-----------------------
 DEVICE CLASS       : MS/B10
 MAXIMUM BLOCK SIZE : 8162
 CYCLING MODE       : AUTOMATIC
 ALLOCATION SIZE    : 8 (UNIT=CYLINDER)
TRANSFER CHARACTERISTICS
------------------------
 DEVICE CLASS       : MT/T9
 COPIES             : 1
LIST OF AVAILABLE FILES FOR JOURNALIZATION
------------------------------------------
 ON DISK R1F6B3 :  SYS.JA.J2R1F6B3
                   SYS.JA.J3R1F6B3
                   SYS.JA.J4R1F6B3
                   SYS.JA.J5R1F6B3
                   SYS.JA.J6R1F6B3
                   SYS.JA.J7R1F6B3
                   SYS.JA.J8R1F6B3
LIST OF AVAILABLE MEDIAS FOR TRANSFER
-------------------------------------
 MPHB01
 MPHB02
DESCRIPTION OF ACTIVE JOURNAL FILES
-----------------------------------
CURRENT FILE   :  SYS.JA.J1R1F6B3
------------
 FILE TYPE                 : PRIMARY,RESULTS FROM JOURNALIZATION
 FILE SEQUENCE NUMBER      : 12
 VOLUME SERIAL NUMBER      : R1F6B3
 BEGINNING DATE            : 10:26:18.334  DEC 14, 1993
 LAST WRITING DATE         : 07:57:35.032  DEC 16, 1993
 NUMBER OF SYNCHRONIZATION POINTS : 41
 NUMBER OF FORCED PHYSICAL WRITES : 22
 NUMBER OF BLOCKS                 : 87
 FILLING OF BLOCKS:78% OF BLOCKS ARE FILLED BETWEEN 90 AND 100%
                    4% OF BLOCKS ARE FILLED BETWEEN 40 AND  60%
                    4% OF BLOCKS ARE FILLED BETWEEN 20 AND  40%
                   14% OF BLOCKS ARE FILLED BETWEEN  0 AND  20%
```

❑

NOTE:

NUMBER OF SYNCHRONIZATION POINTS is the number of checkpoints or TDS commits that have been journalized. NUMBER OF FORCED PHYSICAL WRITES is the number of checkpoints or commits that have caused a physical write to the After Journal file. The ratio of the two values indicates the grouping factor of the commits in a physical I/O.

EXAMPLE: List of Journalized Files

```
LIST OF JOURNALIZED FILES
-------------------------
****************************************************************
*       EXTERNAL FILE NAME       *     LAST DATE OF SAVE      *
*         CATALOG NAME           *                            *
****************************************************************
* FIRE.REL01                     *S09:47:46.595  SEP 30, 1993 *
*   FIRE.CATALOG                 *                            *
*                                *                            *
* FIRE.REL02                     *S09:52:57.073  SEP 30, 1993 *
*   FIRE.CATALOG                 *                            *
*                                *                            *
* GIF1.UFASREL                   * XXXXXXXXXXXXXXXXXXXXXXXXXX *
*   GIF1.CATALOG                 *                            *
*                                *                            *
* GIF1.UFASREL1                 * XXXXXXXXXXXXXXXXXXXXXXXXXX *
*                               *                            *
* FIRE.REL03                    * XXXXXXXXXXXXXXXXXXXXXXXXXX *
*   FIRE.CATALOG                *                            *
*                               *                            *
* FIRE.REL06                    * XXXXXXXXXXXXXXXXXXXXXXXXXX *
*   FIRE.CATALOG                *                            *
*                               *                            *
* FIRE.REL05                    * XXXXXXXXXXXXXXXXXXXXXXXXXX *
*   FIRE.CATALOG                *                            *
*                               *                            *
* FIRE.REL04                    * XXXXXXXXXXXXXXXXXXXXXXXXXX *
*   FIRE.CATALOG                *                            *
**************************************************************
LIST OF KEPT JOURNALIZED FILES
------------------------------
 NONE
LIST OF TDS USER JOURNAL FILES
------------------------------
 ************************************************
 *    TDS NAME    *    LAST DATE OF DUMPJRNL    *
 ************************************************
 *                *                             *
 *    TS11        * XXXXXXXXXXXXXXXXXXXXXXXXXXX *
 *                *                             *
 ************************************************
    % SPACE CURRENTLY USED IN DIRECTORY : 0
```

❑

3.2.5.10 MODIFY_MEDIA

MODIFY_MEDIA is reserved for the SYSADMIN project for the following functions:

- to modify the maximum journal size used on each primary journal media during the execution time of the command,
- to add primary journal media,
- to add or to define secondary journal media,
- to remove an available primary journal media,
- and to remove an available secondary journal media.

Syntax

```
{ MODIFY_MEDIA }
{              }
{ MDMD         }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   [{ JOURNAL_SIZE }      ]
   [{              }=dec4 ]
   [{ JRNLSZ | JSZ }      ]

   [{ ADDPRIMMD }                      ]
   [{           }=( name6 [,name6 ]...)]
   [{ ADDPMD    }                      ]

   [{ ADDSECMD }                      ]
   [{          }=( name6 [,name6 ]...)]
   [{ ADDSMD   }                      ]

   [{ REMPRIMMD }                      ]
   [{           }=( name6 [,name6 ]...)]
   [{ REMPMD    }                      ]

   [{ REMSECMD }                        ]
   [{          }=( name6 [,name6 ]...)  ]
   [{ REMSMD   }                        ]
```

Parameter Description

All parameters are optional. If absent, the previous values are retained.

JAS_NAME:	name of the JAS to be modified. The default is the *jasname* specified previously when either invoking the MNJAS processor or submitting the MNJAS JAS command.
JOURNAL_SIZE:	maximum space to be allocated on each primary media to the primary journal files in units of cylinders (dec-4) for this command only. If it is not provided, the journal size specified in the CREATE command or the MODIFY_PARAMETERS command will be used by default. The minimum accepted value is 8 cylinders. This size is the global amount of space reserved on each media for the primary After Journal files. The system creates up to 8 primary files and the specified amount of space is divided among them.

ADDPRIMMD: A maximum of 20 media may be provided for:

- adding (appending) new primary journal media in the *jas_name* JAS directory
- and preallocating the primary journal files on the specified media.

Files are cataloged in the *jasname* JAS catalog. The primary journal files just added become available journal files for the JAS to use depending on its requirements. Since primary journal files are allocated, mounting of media is required.

If more than 20 media are required, the ADDPRIMMD command may be executed as many times as necessary. See Note 1.

ADDSECMD: A maximum of 20 secondary journal file media may be provided to be stored in the *jasname* JAS directory. Since files are not preallocated, no mounting of media is required.

If SECDCOPY is used, only the name of the first copy need be specified and the last character of this secondary media name must be "odd". The name of the second copy is deduced from the name of the first copy. The ending odd number is replaced by the following even number.

If more than 20 media are required, the command may be executed as many times as necessary. However, this command checks that the media added have not been given previously, in which case *it does not add the same media twice but continues with the next media.* See Note 2 hereafter.

REMPRIMMD: list of the media (20 maximum) to be removed from the list of primary journal media. All the primary After Journal files must be available on these media.

REMSECMD: list of the media (20 maximum) to be removed from the list of secondary journal media. If SECDCOPY is used, only the name of the first copy need be specified. The second copy will be automatically removed. The secondary After Journal files must be available on these media.

NOTES:

1. A single disk may support a set of journal files belonging to several JAS. On the other hand, one disk may not be added twice for the same JAS. In this case, an error message is returned.
2. The MODIFY_MEDIA command is available only if the JAS administrator has defined the JAS parameters by performing either the CREATE command or the MODIFY_PARAMETERS command. Otherwise it is not executed and an error message is returned.

Constraints

- For BLUE JAS and GREEN JAS, MODIFY_MEDIA requires that the JAS concerned be started. SYS JAS need not be started up for the command to apply.
- If the JOURNAL_SIZE parameter is used, the ADDPRIMMD parameter is mandatory.
- Minimum JOURNAL_SIZE is 8 cylinders.
- Journal parameters must have been previously defined by either the CREATE or the MODIFY_PARAMETERS command.
- The MODIFY_MEDIA command is reserved to SYSADMIN project.
- Secondary media whose name is given in ADDSMD and primary media must be different media.

Messages Issued

```
MDMD jas_name REJECTED : service_name NOT ACTIVATED.
MDMD jas_name REJECTED : FUNCTION RESERVED FOR SYSADMIN
***** MEDIA ALREADY USED FOR JOURNALIZATION.
      NO FILE APPENDED
```

EXAMPLES:

1. MDMD JAS_NAME=BLUE JOURNAL_SIZE=80
   ADDPMD=(disk5,disk6,disk7);

   Allocates new primary journal files for BLUE JAS on the media: disk5, disk6 and disk7, with a maximum used size per media equal to 80 cylinders.
2. MDMD JAS_NAME=SYS ADDSMD=(t4,t5,t6);
   Adds new secondary journal media t4, t5 and t6 to the list of available secondary journal media of SYS JAS.

3. MDMD JAS_NAME=GREEN REMPMD=(PRIM1,PRIM2);
   Removes primary media PRIM1 and PRIM2 from the list of available primary media for GREEN JAS.

❑

3.2.5.11 MODIFY_PARAMETERS

MODIFY_PARAMETERS performs the following functions:

- modifies the default maximum JOURNAL_SIZE per primary media,
- modifies the default BLOCK_SIZE (maximum size of the records containing the after images),
- modifies the primary journal device type,
- defines or modifies the secondary journal device type,
- modifies the AUTOCYCLE option,
- and modifies the SECDCOPY option.

Syntax

```
{ MODIFY_PARAMETERS }
{                   }
{ MDPAR             }

    [ JAS_NAME={ SYS | BLUE | GREEN } ]

    [{ JOURNAL_SIZE }      ]
    [{              }=dec4 ]
    [{ JRNLSZ | JSZ }      ]

    [{ BLOCK_SIZE }      ]
    [{            }=dec4 ]
    [{ BLKSZ      }      ]

    [{ PRIMARY_DVC    }                        ]
    [{                }=MS/{ FSA | D500 | B10 }] see Parameter
    [{ PRIMDVC | PDVC }                        ] Description

    [{ SECONDARY_DVC } { MT/Td/[ Ddddd ] }]
    [{               }={                 }]
    [{ SECDVC | SDVC } { CT/LIB | CT/xx  }]

    [{ AUTOCYCLE }      ]
    [{           }=bool ]
    [{ AUTOCY    }      ]
    [{ SECDCOPY      }      ]
    [{               }=bool ]
    [{ SECDCP | SDCP }      ]
```

Parameter Description

All parameters are optional. If absent, the previous values are maintained.

JAS_NAME:	name of the JAS to be modified. The default is the *jasname* specified previously when either invoking the MNJAS processor or submitting the MNJAS JAS command.
JOURNAL_SIZE:	new maximum space to be allocated on each primary media to the primary journal files in units of cylinders. The minimum accepted value is 8 cylinder. Used space of already allocated primary journal files is not modified. The global amount of space is divided among up to 8 primary files.
BLOCK_SIZE:	maximum size of the records containing the after images. It must have a value in the range 512 bytes to 32000 bytes. The block size may be changed only if there is no primary active journal file and if the list of available primary journal files is empty.
	An attempt to change it when not allowed causes the function to abort. *The TRANSFER_PRIMARY command (3.2.5.17) must be issued before attempting again. If there is no secondary media the RECYCLE_JOURNAL_FILE command (3.2.5.14) and the MODIFY_MEDIA command specifying REMPRIMMD (3.2.5.10) must be issued.*

PRIMARY_DVC:	device class for supporting primary journal files. The device class may be changed only if there is no primary active journal file and if the list of available primary journal files is empty. From TS7254 onwards, only disk devices are supported for all JAS. *An attempt to change the device class when not allowed causes the function to abort. The TRANSFER_PRIMARY command (3.2.5.17) must be issued before attempting again. If there is no secondary media, the RECYCLE_JOURNAL_FILE command (3.2.5.14) and the MODIFY_MEDIA command specifying REMPRIMMD (3.2.5.10) must be issued.*
SECONDARY_DVC:	declares or modifies the external form of device class for supporting secondary journal files. The device type allowed is tape or cartridge or cartridge library. The device class may be changed only if: – there is no active secondary journal file, – and the list of available secondary journal files is empty. An attempt to change it when not allowed causes the function to abort. *The RECYCLE_JOURNAL_FILE command (3.2.5.14) and the MODIFY_MEDIA command specifying REMSECMD (3.2.5.10) must then be issued before attempting again.* Until the JAS administrator gives this value, the functions MODIFY_MEDIA (ADDSECMD, REMSECMD) and TRANSFER_PRIMARY remain unavailable.
AUTOCYCLE:	modifies the recycling option for the Journal File:
Manual	AUTOCYCLE=0: Active Journal files are set in the list of available journal files only by using the command RECYCLE_JOURNAL_FILE. This mode is useful when a user file must be recovered from several saves and not just from the last one.

Automatic	AUTOCYCLE=1: Active journal files are set in the list of available journal files as soon as they no longer contain any valid After Image. A valid After Image concerns a modification since the last SAVE of a user file. If the previous mode was *MANUAL*, it may be changed to *AUTO* and the active Journal Files are automatically recycled if they contain any valid After Image.
SECDCOPY:	option secondary copy (dual copy) for secondary journal files. This option may be changed only if there is no Active secondary Journal file and if the list of available secondary journal files is empty. *An attempt to change it when not allowed causes the command to abort with an error message. The RECYCLE_JOURNAL_FILE command (3.2.5.14) and the MODIFY_MEDIA command specifying REMSECMD (3.2.5.10) must then be issued before attempting again.*

Constraints

- For BLUE JAS and GREEN JAS, MODIFY_PARAMETERS requires the JAS concerned to be started up. SYS JAS need not be started up for the command to apply.
- Minimum journal_size is 8 cylinders.
- Minimum block_size is 512 bytes and maximum is 32000 bytes.
- Block_size may be changed only if there is no active or available primary journal file.
- For PRIMARY_DVC, the device type must be disk. Cartridges are ***NOT*** supported and from TS7254: tape is no longer allowed.
- primary device type may be changed only if there is no active or available primary journal file.
- secondary device type may be changed only if there is no active or available secondary journal file.
- SECDCOPY option may be changed only if there is no active or available secondary journal file.
- The MODIFY_PARAMETERS command is reserved to SYSADMIN project.

Messages Issued

```
MDPAR jas_name REJECTED : service_name NOT ACTIVATED.
MDPAR jas_name REJECTED : FUNCTION RESERVED FOR SYSADMIN
```

EXAMPLES:

1. `MDPAR  JAS=BLUE JOURNAL_SIZE=80 SDVC=mt/t9/d6250;`

   The next time MODIFY_MEDIA JAS_NAME=BLUE ADDPRIMMD ADDSECMD is performed, the default maximum size allocated to primary journal files will be 80 cylinders for each new media (10 cylinders per Journal File) and the secondary media will have a device type MT/T9/D6250.

2. `MDPAR  JAS=SYS SDVC=mt/t9/d6250;`

   The next time MODIFY_MEDIA JAS_NAME=SYS ADDSECMD is performed, the secondary media will have a device type MT/T9/D6250.

3. `MDPAR  JAS=GREEN AUTOCY=1 SDCP=1;`

   Changes *cycling mode* to AUTOCYCLE and accepts DUAL COPY for GREEN JAS.

❑

3.2.5.12 QUIT

QUIT exits from the MNJAS processor and returns the user to S: level.

Syntax

```
{ QUIT  }
{       }
{ Q | / }
```

Parameter Description

None

Constraints

None

Messages Issued

None

EXAMPLE:

```
QUIT;
```

Leaves the current processor MNJAS and returns to level S:.

❑

3.2.5.13 REBUILD_DIRECTORY

REBUILD_DIRECTORY reconstitutes the JAS directory after damage, from a save of this file and information logged in the After Journal files. The directory so reconstructed allows static rollforward of all the user files known to the JAS at the moment when the directory was saved. The internal references of the user file journalized for the first time after the directory has been saved, cannot be retrieved in this *save*. It also allows resumption of journalization.

To be able to reconstruct the directory under optimum conditions, the system administrator must periodically:

- transfer the primary files to secondary files using the MNJAS TRANSFER command,
- make a save of the JAS directory, if possible just after execution of the TRANSFER function but always immediately after the introduction of a new user file in journalization. The system administrator can save the JAS directory at any time during journalization by using the SHARE=DIR parameter. It is therefore strongly advisable to make a save each time a change occurs in the JAS directory (new user file, or new After Journal file, transfer, recycling, FILSAVE, etc.). If the save is not made, the duplicated copies of secondary files must first be uncataloged when running the REBUILD function.

When a JAS directory is damaged, the system administrator must:

- restore the directory from its most recent copy,
- enter the MNJAS REBUILD_DIRECTORY command (alias RB, REBUILD),
- ensure that the tapes or cartridges supporting the secondary files and necessary for reconstructing the directory are mounted (requests to mount the media are normally sent to the console operator when the files are assigned),
- if the command has been issued in interactive mode which is preferable, the system administrator can at any moment interrupt the execution of the reconstruction function through break. For this, messages are sent indicating the start of each reconstruction phase, the same report being also sent to the Sysout. The phases follow on from one another for as long as no error is encountered. Certain messages only serve as flags for Service Center personnel. Other

messages allow the system administrator to verify that the After Journal files used for reconstruction are the right ones and that they are:

- in the correct active or available status,
- primary or secondary files (or a duplicate copy of a secondary file).

For example, if in the list of After Journal files present in the JAS catalog, a file which should be active is not found, the system administrator can interrupt processing and restart it after taking care to put the JAS catalog in the required state *(after having cataloged the journal file).*

- verify the execution report: the Service Center may be consulted if an error occurs,
- start printing the JAS directory through the MNJAS LIST command if the LIST option has not be specified in the REBUILD_DIRECTORY command. The system administrator can then recycle the *active* journal files which are not necessary for future rollforward.

After the execution of the REBUILD function, rollforward or DUMPJRNL can be run on any file or TDS known before the *save* of the directory was taken. *The date of the start of a rollforward is important:* any *save* of a user file taken ***after*** the *save* of the directory is not recorded in the directory - it is therefore essential to explicitly specify the start date of the rollforward.

Likewise, if RESTORE_FILE is to be performed on a file, it is preferable not to use the CKJRNL option (check last save date in After Journal).
The reconstruction function can also be executed for recovering a consistent state of the directory when the JRU function fails to do so. In this case, reconstruction can start from the directory in its current state. All user files are then recoverable and journalization is once again authorized.

Syntax

```
{ REBUILD_DIRECTORY }
{                   }
{ REBUILD | RB      }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   { LIST={ 1 | bool }}
```

Parameter Description

JAS_NAME:	name of the JAS whose directory is to be reconstituted. The default name is the name of the JAS specified previously when either invoking MNJAS or submitting an MNJAS command.
LIST:	LIST (LIST=1) provides a detailed listing if the REBUILD_DIRECTORY command terminates normally.

Constraints

- HA-type JAS must be in the EMPTY state.
- No After Journalization must be active.

Messages Issued

The execution report is sent to the Sysout and, in interactive mode, to the console. When the process for reconstituting the JAS directory is started, an immediate listing which can be interrupted appears.

1. When no anomalies occur, messages for a successful reconstruction are treated in the paragraph *Normal Operations* below.
2. When an anomaly is detected, or an I/O error occurs, messages for failure to reconstruct the directory are treated in the paragraph *Unsuccessful Execution* below.

EXAMPLES:

```
MNJAS COMMAND='REBUILD JAS_NAME=BLUE';
```
Reconstitutes for BLUE JAS, its JAS directory from a copy and from the After Journal files and prints the reconstituted directory.

```
MNJAS COMMAND='REBUILD';
```
Reconstitutes the directory for the JAS previously specified.

```
MNJAS COMMAND='REBUILD LIST=0'
```
Reconstitutes the directory for the JAS previously specified, without listing the rebuilt JAS directory.

❑

Normal Operations

After having restored the directory from its most recent copy and having verified that the catalog is valid, the system administrator submits in IOF or batch:

```
   MNJAS COMMAND='REBUILD [JAS_NAME=jasname] [LIST=bool]
```

An immediate account is given of the operations in reconstructing the JAS directory.

Messages that appear during normal processing are:

```
REBUILD OF DIRECTORY FOR JAS jasname
```

Check that the JAS is the correct one.

```
TABLE CREATION
VALIDATION PHASE
RETRIEVE MEDIA IN RESTORED DIRECTORY
RETRIEVE AFTER JOURNAL FILES FROM JAS CATALOG:
   list of efn's
```

Check that the list of cataloged files is correct.

```
   efn2 CONSIDERED DUAL FOR: efn1
```

Check that the duplicate of a secondary file is correct.

```
SORT AFTER JOURNAL FILES ON DATES
SCAN SECONDARY AFTER JOURNAL FILE: efn
```

Stop processing if the secondary file (and those preceding) are not needed for rollforward of files. Remove the secondary file from the catalog and restart the reconstruction function.

```
ERASE OLD CHAIN OF AFTER JOURNAL FILES
UPDATE ;MEDIA
   efn CONSIDERED AS AVAILABLE AFTER JOURNAL FILE
```

Check that the file is available.

```
REINIT ;JOURNAL
   efn CONSIDERED AS ACTIVE PRIMARY AFTER JOURNAL FILE
```

Check that the file is primary and active.

```
   efn CONSIDERED AS ACTIVE SECONDARY AFTER JOURNAL FILE
```

Check that the file is secondary and active.

```
SCAN SECONDARY AFTER JOURNAL FILE: efn
```

The secondary file can be reread provided that the file has not been closed at transfer (as in the case of crash or IO error).

```
SCAN PRIMARY AFTER JOURNAL FILE: efn
   READ
   CREATE AFTER JOURNAL FILE SUBFILE FOR: efn
UPDATE ;JOURNAL
INVALID ABORTED ACTIONS
UPDATE ;FILE
RELEASE RESOURCES
```

The function terminates with the following message:

```
   REBUILDING OF JAS DIRECTORY SUCCESSFULLY COMPLETED
```

Where applicable, the LIST function (LIST option in the REBUILD_DIRECTORY command) is executed.

Unsuccessful Execution

The execution report of an unsuccessful reconstruction function enables pinpointing of the operation at which MNJAS encountered an error and the search for a solution with the help of the Service Center. Each reconstruction phase starts with the transmission of the corresponding message. The system administrator can interrupt processing in interactive mode if the behavior of the REBUILD function is not what is required.

An error message is sent in the form of:

- either a return code which is meant for the Service Center (error information is written to SYS.ERLOG),
- or the following explicit messages which allow the system administrator to intervene.

```
REBUILD OF DIRECTORY FOR JAS jasname
   ERROR: return-code
```

Meaning:	The JAS table has not been created at a previous system restart (JP41 message has been sent). No action with the After Journal is possible.
Action:	Shutdown the system, restart it, then issue the command.

```
TABLE CREATION
   ERROR: return-code
```

Meaning:	The load on the system prevents the allocation of the necessary resources.
Action:	Get the Service Center to analyze the return-code. Terminate non-essential jobs to improve the situation.

```
VALIDATION PHASE
   ONLY JOURNALIZATION ON DISK IS SUPPORTED
   ERROR: return-code
```

Meaning:	Reconstruction can only be done from a *save* of the directory where journalization was made on a disk.
Action:	As in *Meaning*.

```
RETRIEVE MEDIA IN RESTORED DIRECTORY
   ERROR: return-code
```

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

RETRIEVE AFTER JOURNAL FILES FROM JAS CATALOG:

- `INVALID AFTER JOURNAL FILE NAME:` *`efn`*

Meaning:	The *efn* is not an After Journal file.
Action:	Decatalog the file and check the JAS catalog, then restart reconstructing the directory.

- `ERROR: JAGEN DVCERR`

Meaning:	The device type of the file does not conform to that of the primary files as declared in the restored directory and therefore reconstruction cannot proceed.
Action:	As in *Meaning*.

- `ERROR:` *`return-code`*

Meaning:	Problem in accessing the JAS catalog.
Action:	Check the JAS catalog, then restart reconstructing the directory.

- `ERROR IN ACCESSING AFTER JOURNAL FILE`

Meaning:	Problem in accessing file information.
Action:	Check with the Service Center.

SORT AFTER JOURNAL FILES ON DATES

- `ERROR FOR ALLOCATING DEVICE:`

Meaning:	A device for reading secondary files cannot be allocated.
Action:	Free a device for allocation to the file, then restart reconstructing the directory.

- `READ ERROR ON AFTER JOURNAL FILE:` *`efn`*

Meaning:	The first block of the file cannot be read.
Action:	If the file is not the right After Journal file, decatalog it, then restart reconstructing the directory.

- `INVALID BLOCK SIZE FOR PRIMARY AFTER JOURNAL FILE`

Meaning:	Reconstruction cannot proceed since the block size of the primary file does not correspond to that recorded in the restored directory.
Action:	Check with the Service Center.

```
SCAN SECONDARY AFTER JOURNAL FILE: efn
   ERROR: return-code
```

Meaning:	If the primary copy of the secondary file is lost, recreate a new primary copy from the duplicate of the secondary file. Only the primary copy is read for reconstruction.
Action:	As in *Meaning*.

```
ERASE OLD CHAIN OF AFTER JOURNAL FILES
   ERROR: return-code
```

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

```
UPDATE ;MEDIA
   ERROR: return-code
```

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

```
REINIT ;JOURNAL
   ERROR: return-code
```

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

SCAN PRIMARY AFTER JOURNAL FILE: *efn*
ERROR: *return-code*

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

CREATE AFTER JOURNAL FILE SUBFILE FOR: *efn*
ERROR: *return-code*

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

READ

- PREDATING DETECTED ON THIS FILE

Meaning:	Predating has been detected during processing and therefore reconstruction cannot proceed.
Action:	Call the Service Center.

- ASSIGN ERROR ON AFTER JOURNAL FILE: *efn*

Meaning:	The *efn* cannot be assigned.
Action:	Either correct this problem or reconstruct the directory without this file.

- OPEN ERROR ON AFTER JOURNAL FILE: *efn*

Meaning:	The *efn* cannot be opened.
Action:	Either correct this problem or reconstruct the directory without this file.

UPDATE ;JOURNAL
ERROR: *return-code*

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

INVALID ABORTED ACTIONS
ERROR: *return-code*

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

UPDATE ;FILE
ERROR: *return-code*

Meaning:	Problem in accessing the restored directory.
Action:	Get the Service Center to analyze the return-code.

RELEASE RESOURCES
ERROR: *return-code*

Meaning:	Problem in deallocating the necessary resources for reconstructing the directory.
Action:	Quit MNJAS immediately. No consequence to the REBUILD function.

The following general messages can appear at different levels of the reconstruction function:

```
ACCESS ERROR IN ;JOURNAL SUBFILE
ACCESS ERROR IN ;FILE SUBFILE
ACCESS ERROR IN ;MEDIA SUBFILE
ACCESS ERROR IN ;ACTIVE SUBFILE
```

Meaning:	Error in accessing the restored directory.
Action:	Restore and reconstruct the directory again on another disk.

```
ERROR IN TABLE: FAJOT
ERROR IN TABLE: RAJOT
```

Meaning:	Error in the reconstruction tables.
Action:	Consult the Service Center.

3.2.5.14 RECYCLE_JOURNAL_FILE

RECYCLE_JOURNAL_FILE transforms an Active After Journal file into an Available After Journal file.

This command is generally used when the *cycling mode* is *MANUAL*. In *MANUAL cycling mode*, this command enables Journal Files to be reused when they do no longer contain any useful information.

RECYCLE_JOURNAL_FILE is also useful when the *cycling mode* is *AUTO*. Journal Files except the current one, are automatically recycled when all the After Images of these files are obsolete. An After Image is obsolete as soon as one save of the journalized user file to which the image belongs, has been made.

The command is therefore useful in *AUTO cycling mode* when automatic recycling is prevented by some obsolete files no longer being saved. It is also useful for recycling the current After Journal file which is not recycled through a save file.

When a Journal File is recycled from Active to Available, all the previous Journal Files, primary as well as secondary, are also recycled.

The three modes of recycling are:

- **LASTFILE -** *Indicates the last Journal File and all previous ones are to be recycled*:

  The command LIST gives the names of the Active Journal files with the dates of the first and last After Images which are recorded.

- **LASTDATE -** *Indicates the last Journal Date for recycle:*

  RECYCLE_JOURNAL_FILE recycles all the journal files which only contain After Images older than the date given. (The date and time can be specified in milliseconds.)

- **OBSOLETE -** *Journal files whose After Images are obsolete, are recycled.*

NOTE:

Warning messages are written in the report in the case of recycling of a Journal File that contains no obsolete After Images.

The system administrator is recommended to ensure that all the user files have been ***saved*** before issuing this command with the options LASTFILE or LASTDATE.

Syntax

```
{ RECYCLE_JOURNAL_FILE | RCYJRNLF }
{                                 }
{ RECYCLE | RCY                   }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   {{ LASTFILE }                                }
   {{          }=file78                         }
   {{ LSTF     }                                }
   {                                            }
   {          { TODAY                          }}
   { LASTDATE={ -1M                            }}
   {          { -1Y                            }}
   {          { [yy]yy.mm.dd/hh[.mm[.ss[.msmsms]]]}}
   {                                            }
   { OBSOLETE=bool                              }
```

Parameter Description

JAS_NAME:	name of the JAS to be processed. The default is the jasname specified previously when either invoking the MNJAS processor or submitting the MNJAS JAS command.
LASTFILE:	name of the last Journal File to be recycled. The previous Journal File will also be recycled.
LASTDATE:	TODAY: recycles all the active journal files. -1M recycles only the journal files that contain After Images older than TODAY (current date and time) minus one month. -1Y recycles only the journal files that contain After Images older than TODAY (current date and time) minus one year. [yy]yy:mm:dd/hh[.mm[.ss[.msmsms]]]: recycles only the journal files that contain After Images older than the LASTDATE specified.
OBSOLETE:	recycles the files containing obsolete After Images.

Constraints

- For BLUE JAS and GREEN JAS, RECYCLE_JOURNAL_FILE requires that the JAS concerned has been started up. SYS JAS need not be started up for the command to apply.
- One and only one of the three parameters LASTFILE, LASTDATE, and OBSOLETE must be defined.
- It is allowed to recycle the Current Journal File only when no step is journalizing in the JAS, and when no ROLLFWD/DUMPJRNL utility is running using this JAS.
- This command is not available while the TRANSFER_PRIMARY command is running.
- The RECYCLE_JOURNAL_FILE command is reserved to SYSADMIN project.

Messages Issued

```
RCY jas_name REJECTED : service_name NOT ACTIVATED.
RCY jas_name REJECTED : FUNCTION RESERVED FOR SYSADMIN
```

EXAMPLES:

1. `RCY  JAS=BLUE  LSTF=BLUE.JA.J4-vsn;`

   Recycles for BLUE JAS, the Journal file BLUE.JA.J4-vsn and all ones older.

2. `RCY  JAS=SYS  LASTDATE=1996.03.01/06.15;`

   Recycles for SYS JAS, only the Journal Files which contain After Images older than March 1, 1996, 6.15am.

3. `RCY  OBSOLETE;`

   Recycles for the current JAS, all the Journal Files which only contain obsolete After Images.

❑

3.2.5.15 STATUS

STATUS determines how the MNJAS processor is to proceed for subsequent commands where an error occurs in batch mode or in interactive mode if a COMFILE is used.

Syntax

```
{ STATUS }
   { ONLY | RESET | EVEN | IGNORE }
```

Parameter Description

ONLY:	***default:*** executes the next command only if **no** error is detected.
RESET:	resets the error status of the last command to zero; the job is *not* aborted.
EVEN:	executes the next command even if an error has been detected on the last one, without changing the status.
IGNORE:	the status will no longer be tested in subsequent commands and the job will continue irrespective of the errors detected.

Constraints

STATUS is effective only in batch and is not visible to the IOF user, except if COMFILE is used.

Messages Issued: None

EXAMPLES:

1. `RECYCLE LASTFILE=SYS.JA.J... SYS.JA.J...` is busy for a rollforward
   ```
   STATUS RESET
   LIST
   TRANSFER
   ```
   The abnormal status sent by RECYCLE is reset, and the LIST and TRANSFER commands are executed.

2. `RECYCLE LASTFILE=SYS.JA.J... SYS.JA.J...` is busy for a rollforward
   ```
   STATUS EVEN
   LIST
   TRANSFER
   ```
   The abnormal status sent by RECYCLE is *kept*. The LIST command is executed and the step is aborted after its execution. The TRANSFER command is therefore not executed.

❑

3.2.5.16 SWITCH

SWITCH allows the current primary file to be switched for a subsequent journalization session.

Syntax

```
{ SWITCH_PRIMARY }
{ SW | SWITCH    }

   [ JAS_NAME={ SYS | BLUE | GREEN }]
```

Parameter Description

JAS_NAME:	the JAS for which the current primary file is to be switched. The default is the JAS specified when either invoking JAS or submitting an MNJAS command.

Constraints

- HA-type JAS must be ACTIVE.
- Journalization must not be active.

NOTE:
A journalization session starts with the connection to a JAS for the first step journalized. It ends with the disconnection from the JAS for the last step journalizing in it.

Messages Issued

```
jasname: a switching of the primary After Journal file will
occur at the beginning of the next journalization session.

command rejected: journalization is active.
```

The reason for switching primary files is given when the command LIST is issued:

```
SWITCHING DUE TO REQUEST
```

3.2.5.17 TRANSFER_PRIMARY

TRANSFER_PRIMARY enables the JAS Administrator to transfer primary journal files to secondary journal files on tapes or cartridges as follows:

- primary journal files are compacted by omitting all the After Images which belong to aborted FRUs (commitments and/or steps) thereby speeding up the search of those valid Images to be applied in the ROLLFWD/DUMPJRNL utilities.
- The space used in the JAS directory to store the list of aborted File Recovery Units is released.

- Active Journal Files completely transferred are returned to the list of available journal files for journalization.
- The command may execute even while steps are still using the JAS.

By issuing TRANSFER_PRIMARY periodically with a frequency depending on the space available for journalization and the amount of journalized data, the JAS Administrator may:

- reduce the disk space dedicated to the Journal
- produce journal files on removeable media (tapes or cartridges).

NOTE:
GCL command syntax incorporating TRANSFER_PRIMARY is:

```
EJR PROC=MAINTAIN_JAS
ENQUEUE=enqueue_time
VALUES=(COMMAND='TRANSFER;')
```

Syntax

```
{ TRANSFER         }
{ TRANSFER_PRIMARY }
{ TFRPRIM | TFRP   }

   [ JAS_NAME={ SYS | BLUE | GREEN }]

   [{ NEXT_MEDIA }              ]
   [{            }={ 0 | bool }]
   [{ NXMD       }              ]

   [ END={ LEAVE | UNLOAD }]

   [ FORCE_ABORT={ 0 | bool }]

   [ WAIT_DEVICE={ 0 | bool }]
```

Parameter Description

JAS_NAME:	name of the JAS for which TRANSFER will be executed. The default is the jasname specified previously when either invoking the MNJAS processor or submitting MNJAS JAS command.
NEXT_MEDIA:	forces switching to a new secondary Journal file.

END:	determines how last tape or cartridge is to be handled at the end of the transfer:
=LEAVE	***default:*** media remains loaded
=UNLOAD	media is unloaded.
FORCE_ABORT:	determines if the step is to abort:
=0	***default***: no abort
=1	if the STATUS given by the TRANSFER function is severity 2 (*status* >= 1000), it is *forced* to 3 and the step aborts.

STATUS Values with Conditions are as Follows:

STATUS=0	complete transfer
STATUS=1	no transfer since there is no primary active file
STATUS=100	no transfer since another transfer is in progress
STATUS=200	partial transfer since rollforward or DUMPJRNL is in progress
STATUS=210	no transfer since rollforward or DUMPJRNL is in progress
STATUS=500	partial transfer since at least one primary active file is blocked by the journalization
STATUS=1200	*default:* partial transfer since primary file is BUSY
STATUS=1300	*default:* partial transfer since no more secondary files are available
STATUS=1400	*default:* partial transfer since a secondary file is BUSY
STATUS=2000	*default:* no transfer due to lack of resources
STATUS=2200	*default:* no transfer since at least one primary file is BUSY
STATUS=2300	*default:* no transfer since no more secondary files are available
STATUS=2400	*default:* no transfer since secondary file is BUSY

STATUS=2500	*default:* no transfer since at least one primary active file is blocked by the journalization
STATUS=11200	if FORCE_ABORT: partial transfer since a primary file is BUSY
STATUS=11300	if FORCE_ABORT: partial transfer since no more secondary files are available
STATUS=11400	if FORCE_ABORT: partial transfer since a secondary file is BUSY
STATUS=12000	if FORCE_ABORT: no transfer due to lack of resources
STATUS=12200	if FORCE_ABORT: no transfer since at least one primary file is BUSY
STATUS=12300	if FORCE_ABORT: no transfer since no more secondary files are available
STATUS=12400	if FORCE_ABORT: no transfer since a secondary file is BUSY
STATUS=12500	if FORCE_ABORT: no transfer since at least one primary active file is blocked by the journalization
STATUS=>20000	other anomaly
WAIT_DEVICE:	is a new parameter added as of TS8560. This parameter determines whether or not the step waits for the freeing of the devices necessary for transfer when there is an insufficent number.
=0	***default***: no wait. 2000 STATUS (or 12000 in the event of FORCE_ABORT) is mounted.
=1	Wait mode. The submitter of the command is notified of this wait (via a JP93 message). The operator receives a request to free one or two devices depending on whether or not there is a dual copy, and whether one or two are absent (JP94 message).

Constraints

None

Messages Issued

Normal

```
                    ********************************
                    *                              *
                    *    TRANSFER PRIMARY REPORT   *
                    *           NAME : SYS         *
                    *                              *
                    ********************************
PHASE : 1
-------
  PRIMARY SENDER : SYS.JA.J1R1F6B3
  SECONDARY RECEIVER : SYS.JA.J1MPHB01
  STATUS : THE PRIMARY SENDER FILE IS COMPLETELY TRANSFERRED

PHASE : 2
-------
  STATUS : NO ACTIVE FILE CANDIDATE FOR TRANSFER
```

Abnormal

```
TFRP jasname REJECTED : service_name NOT ACTIVATED
TFRP jasname REJECTED : CONCURRENT RECYCLE_JOURNAL_FILE FUNCTION
TFRP jasname REJECTED : TRANSFER_PRIMARY FUNCTION ALREADY RUNNING
TFRP jasname REJECTED : FUNCTION RESERVED FOR SYSADMIN
TFRP jasname : NO PRIMARY FILE CANDIDATE FOR TRANSFER
```

EXAMPLES:

1. `TFRP  JAS=GREEN;`

   Transfers primary Journal Files to secondary Journal Files for GREEN JAS.

2. `TFRP  NEXT_MEDIA;`

   Executes TRANSFER_PRIMARY for the current JAS and forces switching to the next secondary media.

❑

Using the Transfer Function

Description of the Function

The prerequisite for the TRANSFER_PRIMARY command is the successful execution of either the CREATE SECONDARY_MEDIA or MODIFY_MEDIA ADDSECMD command.

The features of the function provided by the command are:

- Only one TRANSFER_PRIMARY can execute at a given time on a JAS. Any attempt to run another TRANSFER_PRIMARY before the completion of the first will be rejected.
- The function automatically determines the PRIMARY active journal files to transfer. A PRIMARY Active Journal file can only be transferred when all its blocks do not contain images of a running FRU. Although the command DJAS displays the list of steps connected to this JAS, the JAS Administrator can determine which are journalizing and preventing the *transfer* function.
- The PRIMARY Journal file on which TRANSFER_PRIMARY is running, is the *SENDER* file. The SECONDARY Journal file receiving the After Images is the *RECEIVER* file.
- A tape or a cartridge, used as the *RECEIVER* of the TRANSFER_PRIMARY function, must either not contain a file at all or contain a file which can be:
  - either an obsolete non-cataloged file,
  - or an old Journal file.
- TRANSFER_PRIMARY keeps exclusive use of one device for the RECEIVER throughout execution (a tape or a cartridge device). If the SECDCOPY option is used, TRANSFER_PRIMARY keeps two devices for its exclusive use. If at TRANSFER_PRIMARY start, no device of the right device type is available, or only one device is available where secondary copy is used,
  - the command is immediately ended with STATUS 2000 (or 12000 in the event of FORCE_ABORT),
  - if the parameter WAIT_DEVICE is specified, the step is enqueued. The submitter of the command is notified of this wait (via a JP93 message). The operator receives a request to free one or two devices depending on whether or not there is a dual copy, and whether one or two are absent (JP94 message). The step is only released when the necessary resources are freed or if a job CANCEL is provoked, followed by a break when the command is executed in IOF.

- Switching to a new *RECEIVER* file occurs:
  - when the current *RECEIVER* file is full or an incident occurs on the current *RECEIVER* file,
  - on user request (NEXT_MEDIA option),
  - or after a system crash during a transfer.
- Once a PRIMARY Active Journal file has been transferred, it is:
  - removed from the chain of active journal files,
  - and returned to the list of available journal files for journalization.
- While running, each phase of the function is described in the utility report, namely: *SENDER* file, *RECEIVER* file, status of the phase and abnormal events. Abnormal events are recorded in the SYS.ERLOG file for debugging purposes.

Use of SECDCOPY Option

If the JAS administrator has chosen the SECDCOPY option when submitting the CREATE or MODIFY_PARAMETER command, two secondary journal files are simultaneously updated. The command requires that both media are available.

The convention for naming the copies is as follows:

- The name of the first copy is a string of up to 6 characters, the last character being odd. This name is provided when submitting either CREATE SECONDARY_MEDIA or MODIFY_MEDIA ADDSECMD.
- The name of the second copy is deduced by the utility whereby the ending odd number is replaced by the even number following the odd number of the first copy. The JAS administrator must execute VOLPREP utility for the two media.

The two *RECEIVER* files are always identical. *RECEIVER* files are switched when:

- either one of the two current *RECEIVER* files is full,
- or an incident occurs on one of the two current *RECEIVER* files.

Normal End of Transfer

The TRANSFER_PRIMARY function stops when one of the following occurs:

- no further PRIMARY Active Journal file is to undergo TRANSFER_PRIMARY,
- no further SECONDARY *RECEIVER* file is available,
- the *SENDER* file is busy and cannot be handled by TRANSFER_PRIMARY.

The result of a successful execution of the TRANSFER_PRIMARY function is the following:

- all PRIMARY active journal files, including the Current, are transferred. All active journal files have SECONDARY type,
- the chain of active journal files is mixed whereby SECONDARY files follow PRIMARY files. No PRIMARY Active Journal file is partially transferred.

NOTES:

1. The transfer of a PRIMARY Active Journal file may be supported by several SECONDARY active journal files.
2. A SECONDARY Active Journal file may support the transfer of several PRIMARY active journal files depending on the sizes of the *SENDER* files and the *RECEIVER* tapes or cartridges.
3. ROLLFORWARD and DUMPJRNL functions are designed to read the chain of active journal files, whatever their types (primary or secondary).

Abnormal End of Transfer

If the TRANSFER_PRIMARY function aborts, or the system crashes while the function is running, the state of the JAS may be stable and consistent, or inconsistent.

This depends on whether the function was in the process of changing information in the JAS directory when the abort occurred. That is, a new Active SECONDARY Journal file was being inserted, or an old Active PRIMARY Journal file was being deleted, in the chain of active journal files.

If an old Active PRIMARY Journal file was being deleted, the next time an After Journal function is activated, a system function will automatically reset the After Journal to a stable state. The After Journal is activated, for example, on journalization or on submission to either the MNJAS processor or the ROLLFWD utility.

If the system function cannot do this due to an unrecoverable error such as an I/O error on the JAS directory, the state of the After Journal may be either unstable but consistent, or inconsistent.

If the After Journal is unstable but consistent, the following functions remain available:

- journalization,
- static and dynamic ROLLFWD, DUMPJRNL, MNJAS, MODIFY_MEDIA and DELETE functions,
- and RECYCLE_JOURNAL_FILE function for an Active PRIMARY file.

The other functions, particularly the TRANSFER_PRIMARY function, are not available. User files continue to be protected. The After Journal will be in a stable state again when an Active PRIMARY Journal file can be recycled (RECYCLE_JOURNAL_FILE) after the journalized user files have been saved.

If the After Journal is left in an inconsistent state, try rebuilding the JAS directory from a copy of itself.

If the After Journal is still in an inconsistent state, the protection of journalized user files is not ensured. Journalized user files must be saved and a RECYCLE must be performed.

The progress of the TRANSFER_PRIMARY function is noted in the JAS directory. In the case of abnormal termination, TRANSFER_PRIMARY is restarted by the JAS administrator. The function will resume from the last logged entry in the JAS directory.

Abnormal events are recorded in the SYS.ERLOG system file for debugging purposes. The utility PRLOG prints out the contents of this file.

Simultaneities

The TRANSFER_PRIMARY command may run concurrently with :

- ***all*** MNJAS commands except CREATE, DELETE, RECYCLE_JOURNAL_FILE and TRANSFER_PRIMARY,
- Journalization,
- Dynamic Rollforward, ROLLFWD and DUMPJRNL utilities.

File Assignments

If one of the *RECEIVER* files is busy, TRANSFER_PRIMARY processing is automatically enqueued and restarted when the file becomes available again.

However if the *SENDER* file is busy, the *RECEIVER* file is deassigned. TRANSFER_PRIMARY processing is set waiting for five minutes and the following message keeps reappearing:

```
JP92 jasname : ron COMMAND TRANSFER WAITS FOR efn JOURNAL FILE
                   [ USED BY ROLLFWD/DUMPJRNL ]
```

When TRANSFER_PRIMARY processing restarts, JP92 no longer appears and an assignment for the *RECEIVER* file is reissued. If TRANSFER_PRIMARY processing encounters a busy *SENDER* more than ten times consecutive, it terminates and must be restarted later by the JAS Administrator.

Changes in Technical Status

From TS7254, the format of secondary files has changed. The new format improves the performance of the TRANSFER function and rollforward.

GCOS 7 supports and processes these changes as follows:

- A secondary file created in a previous technical status can be read by rollforward. Switching to a new secondary file is automatically done when transferring under the new technical status.
- When the system is reset to the previous technical status, a secondary file created with the new format can be read by rollforward. Switching to a new secondary file is automatically done when transferring under the previous technical status.

4. Private JAS

4.1 Overview

4.1.1 Goal of the Product

Private JAS is a new feature, available from TS7560/TS8560 onwards, that enables the System Administrator to divide the load of file journalization among three JAS: the system JAS (SYS JAS), and two Private JAS - BLUE JAS and GREEN JAS.

This 3-way partition has advantages in two areas, performance and application security.

Performance

- These two Private JAS handle their own files (BLUE.JADIR, GREEN.JADIR), which allows the JAS SYS to offload files
- Acceleration of static Rollforwards because each JAS manages a minimum of files; and speeding up of TRANSFER and DUMPJRNL.

Application Security

The possibility to confine any particular application to a Private JAS (development applications, for example) without affecting other applications running under JAS.

4.1.2 Main Features

The way to ensure the Journal protection of a file by a private JAS is to **LINK** its private catalog to this JAS. The "linking" of a private catalog to a JAS is performed through the MAINTAIN_JAS option LINK command. In return, each file of a catalog linked to a JAS, if this file is journalized **in After mode** will be Journal-protected by this private JAS. If a file is only journalized in Before mode, it is not considered as protected by a **specific** JAS.

A step opening a file protected by a Private JAS will be connected to this Private JAS. The connection of steps to JAS must obey certain rules:

- A batch or IOF application can be connected to one, two or three of the three JAS
- A TDS will be connected to only one JAS, from start to finish. This implies that all files journalized in **After** mode and accessed by a TDS must be protected by the same JAS. This also implies that a file only journalized in **Before** mode can be accessed by a TDS whatever the JAS linked to its catalog

The installation and administration of the Private JAS will be by means of standard Journal utilities: MNJAS commands, ROLLFWD and DUMPJRNL utilities.

Three constraints must be noted:

- With "HA": HA and Private JAS features cannot coexist.
- With TCRF: the TCRF utility does not allow the recovery of Private JAS.
- With RDDF7: the use of RDDF7 is not extended to files protected by Private JAS.

4.1.3 Co-existence of Three JAS on One Site

Three JAS can be found on one site: the JAS SYS and the two Private JAS, BLUE and GREEN. **The functions of JAS SYS remain unchanged**, it is not affected by the BLUE and GREEN JAS.

The Private JAS BLUE and GREEN are independent both of each other and of JAS SYS. The sole link is that the SYS JAS Directory (SYS.JASDIR) and a primary JAS SYS file, being indispensable in the process of connecting steps to JAS, must be available on site when utilizing Private JAS.

4.2 File Protection/Step Connection

4.2.1 Rules for File Protection

The way to ensure the Journal protection of a file by a Private JAS is to link its private catalog to this JAS. In return, each file of a catalog linked to a JAS, if this file is journalized in After mode will be Journal-protected by this private JAS. If a file is only journalized in Before mode, it is not considered as protected by a specific JAS. (For journalised file types and the respective processing modes, see *Before Journal Characteristics* (section 2.1.1.4) and *After Journal Characteristics* (section 2.1.1.5.)

All files to be protected by a Private JAS must be allocated on FBO disks.

To LINK a private catalog to a Private JAS, the catalog must be auto-attachable. Files that are not catalogued, or are catalogued in a not auto-attachable will remain protected by the JAS SYS (if protected).

A private JAS may protect, of course, the files of several auto-attachable catalogs.

The "linking" of a private catalog to a Private JAS is performed through the **MAINTAIN_JAS** option LINK command, which is a Sysadmin command

To delete the LINK between a catalog and a Private JAS, link the catalog to JAS SYS.

4.2.2 Rules for Step Connection

These rules concern connnection between applications and JAS. Connection consists of having the step known by the JAS.

The operator command **DISPLAY_JAS** gives the list of steps connected to a JAS. Different rules govern connections for the Batch and IOF steps and for the TDS steps, as described below.

4.2.2.1 Batch and IOF

A Batch or IOF application can be connected to **one, two or three** of the three JAS.

Connection at PGI (Process Group Initialization)

If there are journalization options either implicit (ASSIGN of catalogued files with journalization options in the catalog) or explicit (DEFINE JOURNAL=...), the step is connected to JAS SYS in Before and/or After, according to the demands.

Otherwise, no other connection is realized.

Connection during the opening of a file protected by a JAS

If the file must be protected by the JAS SYS, then:

- if the step was connected by PGI to JAS SYS: it stays connected to JAS SYS
- if the step was not connected by PGI to JAS SYS: the file opening aborts.

If the file must be protected by a Private JAS, then the step is connected to the Private JAS (if it isn’t already). If the step is already connected to another JAS, this other connection continues.

Disconnection

This takes place at the PGT (Process Group Termination).

4.2.2.2 TDS

A TDS will be connected to **only one** JAS from start to finish. In spite of that, a TDS may access a file journalized in Before mode whatever the JAS linked to the catalog of this file.

<u>Connection at PGI (Process Group Initialization)</u>

If there are journalization options either implicit (ASSIGN of catalogued files with journalization options in the catalog) or explicit (DEFINE JOURNAL=...), the step is connected to JAS SYS in Before and/or After, according to the demands.

Otherwise, no other connection is realized.

<u>Connection during initialization of the TDS session</u>

If the TDS systems files are not catalogued in a catalog linked to a Private JAS, then:

- if the step was connected by PGI to JAS SYS: it stays connected to JAS SYS
- if the step was not connected by PGI to JAS SYS: initialization of theTDS session aborts.

If the TDS systems files are catalogued in a catalog linked to a Private JAS, then the step is disconnected from the JAS SYS (if it was). This is the sole case of a disconnection to a JAS at a moment other than PGT.

<u>Disconnection</u>

This takes place at the PGT (Process Group Termination).

4.2.2.3 Specific Rules

Two rules apply equally to connections for Batch and IOF steps and to TDS.

Prevention of a file journalization (DEFINE JOURNAL=NO) does not prevent connection to its JAS. The step is always connected to this JAS according to the journalization options implicit in the file (journalization options presented in the catalog). A file whose After journalization was inhibited by a DEFINE will be flagged - by the MNJAS command LIST - with an exclamation mark "!" preceding its name.

A file dynamically assigned can be journalized in a private JAS, even if no DEFINE or ASSIGN order was present in the JCL.This is true when the file is journalized in After or/and Before mode, assuming that its catalog is linked to a private JAS. This is not the case for JAS SYS.

4.2.3 Incidents When Connecting to a JAS

4.2.3.1 Batch and IOF

Failure when opening a file

During a file opening, if this file has to be protected by the JAS SYS, and if the step has not been connected at the PGI to the JAS SYS, then the opening aborts with JAP 3, NJAFT and the message:

```
JP53 OPEN REJECTED FOR external_file_name USER FILE
CONNECTION TO SYS JAS NOT DONE AT STEP INITIATION
```

FUNCNAV at a file opening

During the opening of a file catalogued in a not-autoattachable catalog, but which was relinked to a Private JAS when it was autoattached, then the opening is rejected with the JAP 3 FUNCNAV and the message:

```
JP56 OPEN REJECTED FOR <...efn...> USER FILE BECAUSE ITS CATALOG
FILE IS LINKED TO A PRIVATE JAS AND HAS NOT AUTO_ATTACH OPTION.
```

4.2.3.2 TDS

Abort initialization of TDS session

During initialisation of the TDS session, if the TDS systems files are not catalogued in a catalog linked to a Private JAS, and if the step was not connected at PGI to JAS SYS, then initialization of the session aborts with the return code: JAP 16 SEQERR.

Failure when opening a file

All file opening following initialization of the TDS session, with journalization in After mode, submits to the rule that a TDS can be connected to only one JAS. The return code JAP 3 SHLVVIOL is returned and the message JP51 is sent:

```
JP51 OPEN REJECTED FOR <...efn...> USER FILE PROTECTED BY JAS
TDS IS NOT CONNECTED TO THIS JAS
```

4.3 Managing the Private JAS

4.3.1 Installation

Before using a Private JAS, the System Administrator must:

- Create the Private JAS that (s)he counts on using.
- Create the links between catalogs containing the files to be protected and the JAS.

4.3.1.1 Preparing a Site and Applications

Before creating the Private JAS several points must be carefully considered. The repartition of applications on the JAS must be defined. In particular you must take account of the fact that TDS can only use a single JAS and that consequently, all journalized files accessed by the TDS should be in the catalogs linked to this JAS. This partitioning could then lead to the creation of new catalogs under which you can bring together the files that are to be protected by the same JAS.

Each JAS must be defined. In particular you must calculate the size of the journal files. Do not neglect performance: if possible install the journal files of different JAS on different media (and also on different media from the protected file media). Describe the procedures for managing the JAS: saving, and how to restart in case of an accident.

4.3.1.2 Creation of Private JAS

The MNJAS processor uses the CREATE command to make all journal, catalog and JAS directory files, and to initiate the parameters for the After journalization.

Private JAS are not created by IUF tools. Contrary to JAS BLUE and GREEN created in an "HA" environment, there is no creation of SYS.JASBLUE and SYS.JASGREEN.

Once the MNJAS CREATE command is given, the Private JAS becomes operational immediately (no need to reboot).

4.3.1.3 Creation of Links Between Catalogs and Private JAS

This operation is made by the command LINK. The command DISPLAY_LINK displays these links.

LINK and DISPLAY_LINK are MAINTAIN_JAS domain commands (described in Chapter 3). LINK connects the user files of a private catalog to a specific JAS. DISPLAY_LINK displays the name of the JAS linked to a catalog. This operation can be carried out even if the Private JAS has not been created.

NOTE:
If this operation results in the journal protection for certain files being transferred from one JAS to another, then prior to this it is best to save the files and issue the FORGET_USER_FILE (MNJAS) for all these files.

4.3.2 Administration of Private JAS

4.3.2.1 General Considerations About Journalization

Before journal

This is controlled by SYS.JRNAL, which is a file shared by the different JAS.

After journal

This runs in private journal files inside Private JAS. These files are BLUE.JA.J*imedianame* for the JAS.BLUE, and GREEN.JA.J*imedianame* for the JAS.GREEN in the manner of JAS.SYS files. Associated JAS Directories are called BLUE.JADIR and GREEN.JADIR. All these files are catalogued in the respective BLUE.CATALOG and GREEN.CATALOG.

Recovery

Recovery mechanisms for Private JAS are identical to those for the JAS system. Recovery after a system crash is effected in Rerun on a Restart Warm.

Repeat after checkpoint

Contrary to HA-type JAS, this function is operational with the Private JAS in the case of a system crash and on ABORT of an application.

4.3.2.2 Using MNJAS Commands

The MNJAS processor contains commands that allow you to change the journalization parameters, to list active or available journal files, and to list user files recognised by the After journal or the TDS "User Journals".

All MNJAS commands have a keyword JAS_NAME that allows you to indicate the JAS that you want to process. If you do not give this keyword then the respective JAS will be that which was previously specified when either invoking the MNJAS processor or submitting the MNJAS JAS command.

In other respects, the command DISPLAY_JAS, which is an operator command, allows listing of the steps connected to each JAS.

4.3.2.3 ROLLFWD and DUMPJRNL

For these utilities, the name of the JAS to be processed does not have to be specified. The ROLLFWD and DUMPJRNL utilities access JAS linked to the catalog of the files to be rolled forward or the TDS from which the user journal is to be extracted.

Such a journal step utility only works on a single JAS. The rollforward will not be carried out for a file protected by a different JAS from that protecting the FIRST FILE specified.

4.3.2.4 Saving and Restoring Protected Files

The saving and restoration of files access dynamically the system files of the corresponding JAS.

4.3.2.5 JRU

For Private JAS, the journal recovery utility (JRU) is dynamically activated at the start of journalization, in case it should be needed. It can also be launched in static mode, as happens for the JAS SYS, by the command:

```
EJR JRU LIB=SYS.HSLLIB VL=(JAS_NAME);
```

4.4 Restrictions and Special Constraints

4.4.1 Coupled Systems

In a coupled system environment, the System Administrator must take care that for each Private JAS, that files protected by this Private JAS as well as the system files of this Private JAS (JAS catalog, JAS Directory, and the JAS After journal files) are accessed through only one system. They must never be shared by two systems. They can however be built in two versions, one for each system, on different media.

4.4.2 HA

No coexistence is possible between "HA" and Private JAS.

If a site possesses the MI "HA", it is that feature which is implemented on site.

4.4.3 TCRF

The TCRF utility does not allow the recovery of Private JAS.

The use of TCRF on a system different from the interrupted system can always be made on the JAS SYS, as long as the Private JAS are not created on the backup system. If the private JAS are active on the backup system, rollback of Private JAS-protected files is equally effected during a TCRF passage. But private JAS-protected files remain inaccessible in so far as rollforward was not effected for these files. And this rollforward can only run at the reboot of a hung system.

4.4.4 RDDF7

Use of the Remote Duplicate Database Facility, or Disaster Backup, is limited to files protected by JAS SYS.

A file protected by a private JAS which has been declared - wrongly - protected by an RDDF7 occurrence cannot be followed on a distant site because the After images corresponding to this file will not be transmitted to RDDF7 occurrences of the site where they are being worked on.

5. HA Specific Features

5.1 Description of High Availability (HA)

The information is described in this Section as follows:

- a summary of the general concepts of HA which are treated in more detail in the *High Availability Concepts Manual*,
- with a precise description of how to use HA-type JAS.

For the definition of terms used to describe the HA environment, see the Glossary.

HA (High Availability) is a function implemented for disk coupled systems. HA reduces the time that an HA-type TDS is unavailable when a system crash occurs. The purpose of HA is to reduce the unavailability time of a TDS supporting 300 sessions and 50 files, to less than 4 minutes.

CMSC (Complex Management Service) is the administration function which manages HA services such as activating, terminating and taking over TDS and JAS.

This function is distributed on the two *member systems* through CMSRs (Complex Management ServeRs). The two CMSRs communicate together. They exchange commands and information about the state of the servers and the member of which they are in charge.

Without operator intervention, CMSC provides the means for:

- detecting a system crash
- reestablishing the service by starting up a backup HA application

5.2 HA Services

HA Services are declared at the generation of a complex (CRCXGEN) and are provided for by two Servers:

- the first server being ACTIVE to execute the effective functions of the service,
- the second server being BACKUP and not participating in the effective service functions.

Only CMSC can *start and restart* HA Services.

The two types of services in a complex are **Main Services** and **Used Services**. HA-type TDS is a *Main Service*. HA-type JAS (either BLUE JAS or GREEN JAS) is a *Used Service* which is at the disposal of HA-type TDS. One HA-type TDS can only use one HA-type JAS.

HA-type JAS ensures the integrity and availability of user files to maintain the High Availability of TDS. See *MANAGING THE PRIVATE JAS* (section 4.3).

BATCH and IOF applications can use HA-type JAS. However, since they are not in themselves HA services, they need their submitter to restart them on **takeover**.

Takeover consists of migrating the activities of services from one member to another member of a complex without disruption, thereby providing service continuity.

HA-type TDS and the HA-type JAS it uses, are switched from BACKUP to ACTIVE on the other member without operator intervention on takeover. The sequence of events at the restart of a TDS after takeover following a system crash, is similar to a TDS warm restart, namely:

- reconnection of the sessions,
- immediate file recovery (rollback and dynamic rollforward) of the journalized files,
- and restart of the transactions from the last commitment when the recovery phase terminates.

Takeover is **global** since all the running HA Applications are switched simultaneously over to the other member.

Member failures which lead to a system crash of the member causes ***automatic takeover*** without operator intervention.

Manual takeover is at the decision of the operator who can issue the TAKEOVER command specifying one of the following options:

WEAK:	activates a weak shutdown of the HA services, thus allowing them to complete normally. Active servers are switched to Backup while Backup HA servers are set Active. Neither File Recovery nor the Resynchronization process of Mirrored Disks is needed. *There is no guarantee when the services will be available*. The operator may have to intervene to speed up the process by terminating a few applications using HA services.
STRONG:	leads to the immediate system crash of the co-member.
FORCE:	FORCE is to be used when the co-member is no longer operational but nevertheless a system crash cannot be performed on the co-member, for example, System Check.

5.2.1 Main Service TDS

Takeover is possible if the active server and the backup server are both operational.

The possibility of takeover determines the type of TDS declared, namely:

- *Not Watched by CMSC*

  The TDS functions as in V3 or V5 and is neither monitored nor looked after by CMSC. No takeover is therefore possible.

- *Watched by CMSC, non-HA type*

  The TDS is declared WATCHED BY CMSC at TP7GEN but it has not been declared on both members of the complex during complex generation (CRCXGEN). Although this type of TDS is monitored by CMSC, no takeover is possible.

- *Watched by CMSC, HA type*

  The TDS is declared WATCHED BY CMSC at TP7GEN and it has been declared on both members during CRCXGEN. This type of TDS is monitored and looked after by CMSC and the takeover of this service is always available *if the active server and the backup server are both operational*.

5.2.2 Used Service JAS

The type of JAS which can be found on a system in an HA environment are:

- ***non-HA JAS***

  SYS JAS cannot be used for takeover. It is not declared at CRCXGEN (complex generation) for generating the complex. Since SYS JAS is local and specific to a system, there is only one SYS JAS per system and therefore two SYS JAS in a complex.

- ***HA-type JAS***

  BLUE JAS and **GREEN JAS** can be used by HA applications for takeover. Both JAS are declared for the complex at CRCXGEN. Since both HA-type JAS can be declared per complex, one HA type JAS can be in active mode on a member and the other HA type JAS can be in active mode on the other member.

  The two HA-type JAS can also be in active mode on the same member, for example, after takeover, regardless of whether the two HA-type JAS were ***initially*** active on the same member.

  An HA-type JAS can also be used by non-HA TDS, declared WATCHED BY CMSC. However in this case, no takeover can take place.

  An HA-type JAS becomes switchable when an HA-type TDS is started in a complex. Consequently, if BLUE JAS and GREEN JAS are used simultaneously on a complex, they are either both switchable or both unswitchable.

5.3 Using HA-Type JAS

HA-type JAS ensures two functions with regards to the HA process:

- ***User File Integrity***

  with journalization techniques, the HA-type JAS guarantees the integrity of **journalized** files in the case of TDS Abort, System Restart and Takeover, thus maintaining continued TDS availability.

- ***User File Availability***

  by preventing user files except libraries from being accessed on the *backup member*, while the HA-type TDS is currently accessing them on the *active member*. This mechanism guarantees HA applications access to these files in the case of takeover. **Protected** files are not necessarily journalized but are useful for starting the HA-type TDS or restarting it on the backup system. The availability of libraries is managed only through the dual system sharing rules.

5.3.1 Rules for File Protection

To protect a file by an HA-type JAS, link its private catalog to this JAS. All the files of this catalog except libraries are then protected by the same HA-type JAS.

The set of rules regarding user files when using HA-type JAS are as follows:

- All these files must be allocated on FBO disks.
- **All these files and their catalogs must be allocated on shared disks,** so as to be accessible when a takeover takes place.
- The user files protected by an HA-type JAS must be cataloged in auto-attachable catalogs linked to this JAS. Linking is performed by the MAINTAIN_JAS option LINK SYSADMIN command.
- Although an auto-attachable catalog cannot be linked to more than one JAS, an HA-type JAS may protect the files of several auto-attachable catalogs.
- An HA-type JAS controls ***all*** the files of the catalogs linked to it *whatever their journal options and the processing mode in which they are accessed.*
- *A file* ***protected*** *by an HA-type JAS cannot be accessed by an application on a system if this JAS is not in the* ***ACTIVE*** *(SWITCHABLE or UNSWITCHABLE) state on this system. See Special Condition for Unprotected Read (section 5.3.4) for Exceptions.*

To facilitate installing TDS HA, libraries are no longer protected by HA-type JAS even if they are cataloged in a catalog linked to this HA-type JAS. Only the usual sharing rules between coupled systems apply and can sometimes prevent opening a library if it is used on the other member. For example, invalidating the DEBUG trace if the library *tdsname*.DEBUG cannot be opened at the startup of TDS.

5.3.2 Rules for Step Connection

These rules concern *connections* between applications and JAS.

The *connection* of a step to a JAS ensures that the files protected by this JAS will be effectively protected when accessed by this step. That is, their integrity and their availability are guaranteed. Connection consists in having the step known by the JAS.

The operator command DISPLAY_JAS, gives the list of steps connected to a JAS. It may be used as an aid in observing the following rules.

Batch or IOF

A Batch or IOF Application can be *connected* to one or two JAS. A Batch or IOF step which has journalized files assigned in its JCL, is *connected* to SYS JAS at step initiation.

The Batch or IOF step will be *connected* to an HA-type JAS when it opens a file protected by an HA-type JAS. If this HA-type JAS is SWITCHABLE, the step will be *disconnected* from SYS JAS in order to be *connected* to the HA-type JAS. If this HA-type JAS is UNSWITCHABLE, the step will be *connected* to both SYS JAS and the HA-type JAS.

The Batch or IOF application will be *disconnected* from an HA-type JAS, when it no longer accesses files protected by this HA-type JAS if these files are not journalized. The application will be *disconnected* from an HA-type JAS *only at the end of the step if any of these files protected by this HA-type JAS are journalized.*

No Batch or IOF step can be connected to two HA-type JAS at the same time.

NOTE:

The MNJAS (MAINTAIN_JAS) processor, as a step, is processed in a specific way. The commands which require the JAS activity start the connection of the MNJAS step to the current JAS during the execution of the command. Between two commands, MNJAS is not connected to any JAS. See *JAS Installation and Administration* (Chapter 3).

TDS

A TDS will be *connected* to ***one and only one JAS*** from start to termination. Before the TDS initialization session, however, TDS is considered a Batch application by the JAS. At TDS step initiation, the TDS will therefore be *connected* to SYS JAS. If no other step was previously *connected* to SYS JAS, the following message corresponding to the activation of SYS JAS, will appear at the console:

```
SYS:  AFTER JOURNAL IS ON MEDIUM medianame
      CURRENT AFTER JOURNAL FILE IS SYS.JA.Jimedianame
```

followed by the message corresponding to the activation of the HA-type JAS such as BLUE JAS, if no other step was previously *connected* to the BLUE JAS:

```
BLUE: AFTER JOURNAL IS ON MEDIUM medianame
      CURRENT AFTER JOURNAL FILE IS BLUE.JA.Jimedianame
```

which is issued when the step known as a TDS is connected to this JAS.

Journalization will effectively take place on BLUE JAS. The connection to SYS JAS has been suppressed, which can be verified by using the DJAS command.

NOTE:

If TDS Backup JCL contains DEFINE statements with journal options, the Backup TDS will be connected to the JAS SYS -as if it wanted to journalize-. It is not recommended to include DEFINE statements in the JCL of the Backup TDS. Such statements if inserted, may impede some After journalization functions from executing correctly on the SYS JAS of the member where the TDS is in the Backup state.

5.3.2.1 Rules for TDS Applications

The rules governing TDS applications are:

- All the files accessed by an HA-type TDS must be protected by a *single* JAS.
- A TDS *Not Watched by CMSC* may only use files protected by the non-HA SYS JAS.
- A non-HA TDS *Watched by CMSC* can use the non-HA SYS JAS or an HA-type JAS (BLUE JAS or GREEN JAS). BLUE JAS and GREEN JAS, in this case, must be *UNSWITCHABLE.*
- An HA-type TDS may only use BLUE JAS and GREEN JAS, both of which therefore being *SWITCHABLE.*

- Within the same complex, the following types of TDS can run simultaneously:
  - several non-HA TDS using non-HA SYS JAS,
  - with several HA-type TDS using HA-type BLUE JAS and/or GREEN JAS, both of which being SWITCHABLE.

  However, a non-HA TDS using an HA-type JAS (BLUE JAS or GREEN JAS) cannot run simultaneously with an HA-type TDS using an HA-type JAS (BLUE JAS and/or GREEN JAS).

EXAMPLE 1:

If an HA-type TDS using BLUE JAS is executing, the resulting effects are as follows:

- the BLUE JAS is in a SWITCHABLE state,
- a non-HA TDS *Watched by CMSC* designed to use BLUE JAS or GREEN JAS cannot be started,
- the HA-type TDS cannot open a file protected by GREEN JAS or SYS JAS.

EXAMPLES 2:

If a non-HA TDS *Watched by CMSC* using BLUE JAS is executing, the BLUE JAS is in an UNSWITCHABLE state and an HA-type TDS designed to use BLUE JAS or GREEN JAS cannot be started.

❑

5.3.2.2 Rules for Batch and IOF Applications

The rules governing Batch and IOF applications are:

- A Batch or IOF application may use files protected by one non-HA SYS JAS and/or one HA-type JAS (BLUE JAS or GREEN JAS) in an UNSWITCHABLE state.
- A Batch or IOF application may use files protected by one HA-type JAS in a SWITCHABLE state, on condition that ***all*** protected files used by this application are protected by this JAS.

5.3.3 Transgression of Rules

The consequences of violating any of the rules on files or applications, are as follows:

- when the user file is opened:
  - either `JAP 3, SHLVVIOL` or `JAP 3, FUNCNAV` is returned to the application and the explicit message JP51, JP52 or JP53 is written to the JOR of the application,
  - or `DFASG 39, OVRVIOL` if the file is not on an FBO disk.
- when a TDS declared WATCHED BY CMSC is started up, CMSR sends an explicit message.

When user files are not allocated on shared disks, no error message is sent and the files can be opened, but **they will not be available in case of takeover**, which is incompatible with the High Availability of the HA-type TDS.

Considering the rules described above and since an HA-type JAS controls all the files of the catalogs linked to it (except libraries), the System Administrator is advised to devote one or several specific catalogs to each HA application.

By doing so, files (except libraries) which are not journalized but which are cataloged in a catalog linked to the HA-type JAS, are protected, thereby ensuring their availability in case of takeover.

5.3.4 Special Condition for Unprotected Read

An additional rule enables bypassing the rules described above to allow reading files without the control of JAS, where the application needs to access such data:

- *Applications using an **HA-type JAS** will be allowed to **read** a file protected by another JAS (HA-type or a non-HA) if the following parameters for the file are set accordingly:*

  SHARE to **DIR**, **ONEWRITE**, **MONITOR** or **FREE**

  and

  DUALSHR to **ONEWRITE** or **FREE**.

- Similarly, non-HA applications using non-HA SYS JAS will be allowed to ***read*** such a file protected by an HA-type JAS.

JAS no longer has any control over such accesses when these conditions are met. No connection is established between the step which opens a file in such conditions and the JAS *provided that the step does not write in other protected files*. This step will not appear in the list of steps connected to the JAS displayed by command DISPLAY_JAS.

A file protected by an HA-type JAS can therefore be accessed by an application in read mode even if this JAS is in the BACKUP or EMPTY state on the member where the application is running.

Since unprotected reads are not controlled by the JAS, there is no guarantee on Takeover that these reads can still be performed.

The best way to access a protected file is to do so on the member where the associated JAS is in the ACTIVE state.

5.3.5 Repeat After Checkpoint

When a Batch or IOF application using an HA-type JAS, takes checkpoints, in case of ABORT of the application, or in case of CRASH, the files it updates will be recovered up to where the last checkpoint was taken.

In the case of ABORT, the application will be restarted from the last checkpoint. The REPEAT option must be specified for the application to be able to take checkpoints.

In the case of CRASH:

- either the JAS is taken over, and there is no way to restart such a step on another system: the application cannot be restarted without intervention.
- or the JAS will be reset to ACTIVE state on the same member, but later than system restart: therefore, at system restart, if repeated, the step will receive the return code `JAP 2, ABORTPG`. The REPEAT option is inhibited with HA after system crash. The application cannot be restarted without intervention. The application must therefore be designed to take into account a restart procedure.

5.4 Preparing the HA Environment

5.4.1 Creating the JAS

Before an HA complex becomes operational, the JAS must be installed. Installing JAS involves creating the files it needs to operate. The JAS is installed and maintained through the MNJAS processor.

All the MNJAS commands are common to all types of JAS, but most of them have an additionary constraint when used with HA-type JAS: they can only be executed when the HA-type JAS is in ACTIVE state, namely:

- FORGET_USER_FILE
- FORGET_USER_JOURNAL
- LIST
- MODIFY_MEDIA
- MODIFY_PARAMETERS
- RECYCLE_JOURNAL_FILE
- and TRANSFER_PRIMARY.

For a description of the MNJAS processor, and the syntax and use of MNJAS commands, see *JAS Installation and Administration* (Chapter 3).

5.4.2 Preparing the HA Applications

File Protection

Private catalogs used by HA applications must be linked to an HA-type JAS by the MNJAS LINK command issued by SYSADMIN. If two SITE.CATALOGs are used on the complex, the LINK command must be issued to each system (member) of the complex. In this way user files and TDS files (in case of TDS applications) will be protected by the JAS to which their catalog has been linked.

File Accessibility

Files accessed by HA applications must be accessible at any time. ***To ensure that they will be accessible in case of takeover, they must be allocated on shared disks***.

TDS Before Journal files used by an HA TDS must reside on shared disks. Any Before Journal extensions for Batch and IOF jobs, which use BLUE JAS or GREEN JAS, must also reside on shared disks.

It is recommended not to place on shared disks, files processed by non-HA TDS and therefore protected by SYS JAS so that immediate recovery of these files is not impeded by the switching of HA TDS.

JAS Activity

During the preparation of HA applications, if files protected by an HA-type JAS are to be accessed by utilities such as:

- ROLLFWD, FILREST, FILSAVE or DUMPJRNL,
- and MNLIB, MNF or MDF, or FILPRINT.

the JAS concerned must be in the ACTIVE state, see START_SERVICE command.

Most MNJAS commands on an HA-type JAS can only be executed when the JAS is in an ACTIVE state. See *JAS Installation and Administration* (Chapter 3).

For ensuring High Availability and the best file integrity at Takeover, the exclusive access on catalogs linked to an HA-type JAS and all accesses on journalized files should be performed on the member where the JAS is in the *ACTIVE* state.

5.5 Preparing and Monitoring JAS Services

5.5.1 Declaring Services

For detailed information on the syntax of JAS services, see the *High Availability Administrator's Guide.*

The HA-type TDS must be declared WATCHED BY CMSC at TP7GEN.

HA-type JAS and TDS must be described once through the CRCXGEN Utility.

The directives of the COMPLEX description language used to declare these services:

COMPLEX_SERVICE_TYPE (CXSRVTYP):

Syntax

```
CXSRVTYP  TYPE=JAS ;
```

To declare that HA-type JAS are to be used on the complex.

COMPLEX_SERVICE (CXSRV)

Syntax

```
CXSRV  NAME=BLUE  TYPE=JAS ;
```

and/or

```
CXSRV  NAME=GREEN  TYPE=JAS;
```

To define that BLUE JAS and/or GREEN JAS are to be used on the complex:

- an HA-type TDS named TPHA, that uses BLUE JAS must be declared as follows:

  ```
  CXSRV NAME=TPHA TYPE=TDS USED_SERVICE=(BLUE);
  ```

- a non-HA TDS named HNTD but in *current mode* (WATCHED BY CMSC) that uses GREEN JAS must be declared as follows:

  ```
  CXSRV NAME=HNTD TYPE=TDS USED_SERVICE=(GREEN);
  ```

- a non-HA TDS named NTNH but in *current mode* (WATCHED BY CMSC) that uses SYS JAS must be declared as follows:

  ```
  CXSRV  NAME=NTNH  TYPE=TDS ;
  ```

NOTE:
USED_SERVICE does not apply to SYS JAS.

COMPLEX_SERVICE_MAP (CXSRVMAP)

Syntax

```
COMPLEX_SERVICE_MAP  COMPLEX_SERVICE=BLUE
                     TYPE=JAS
                     MEMBER=(SIT1,SIT2);
```

This must be issued to define that the HA-type BLUE JAS used on the SIT1 and SIT2 members of the complex. Members are listed in the order of the one that is ACTIVE (SIT1) and the one which is its BACKUP (SIT2).

Unlike TDS Services that can be declared on one member only, ***BLUE JAS and GREEN JAS must be declared on the two members.***

5.5.2 Monitoring HA-Type JAS

An HA-type JAS, being a used service of an HA-type TDS, is started automatically when its main service TDS is started.

However, a JAS can be started up independently of an HA-type TDS. If operations involving files protected by this JAS are to be performed, the JAS must be in an ACTIVE state.

Examples of operations involving files protected by a JAS are:

- TDS generation (TP7GEN) in the case where the TDS catalog has already been linked to the HA-type JAS,
- work on the TDS libraries,
- processing batch applications,
- performing some MNJAS functions or utilities such as:
  - ROLLFWD, FILSAVE and FILREST,
  - MNLIB, MNF and MDF.

The commands for monitoring an HA-type JAS appear below.

START_SERVICE (SSRV)

The SSRV command enables the JAS to be started on the complex. As a prerequisite, the Complex and the Member must have been previously started through the SCMSR (START_CMSR) and SMB (START_MEMBER) commands, respectively.

When a JAS is started through the SSRV command, its state will change from EMPTY to ACTIVE on a member and from EMPTY to BACKUP on the other. If the complex is SWITCHABLE, ie there is already at least one HA-type TDS running on it, the ACTIVE server will be in a SWITCHABLE state. If the complex is not SWITCHABLE, in the case where there is no HA-type TDS running on it, the ACTIVE server will be in an UNSWITCHABLE state.

If the JAS has been declared in USED_SERVICE parameter of the COMPLEX_SERVICE directive of a given TDS, it will be automatically started up when TDS is started up and will be ACTIVE on the member where TDS is ACTIVE.

If the state of the JAS when started up was ACTIVE UNSWITCHABLE, starting up an HA-type TDS using the JAS will cause the state of the JAS to change to ACTIVE SWITCHABLE.

On termination of a TDS, if there is no other HA-type TDS running in the complex, the state of the JAS will return to ACTIVE UNSWITCHABLE.

NOTE:

If the After Journal is corrupted for example, where the list of aborted commitment units cannot be obtained for a TDS formally aborted or cannot be recorded in the JAS Directory, or where the JAS directory update phase has not yet occurred, START_SERVICE of this JAS will start JRU (Journal Recovery Utility) for this JAS. In this case, an H_RECOV step, performing JRU is executed before the JAS is set ACTIVE.

If after JRU is executed the JAS still cannot be set *ACTIVE*, the JAS must be deleted and created again.

EXAMPLE:

```
S: START_SERVICE BLUE;
   CM06 SERVICE BLUE STARTED ON SIT1
   CM06 SERVICE BLUE STARTED ON SIT2
```

❑

NOTE:

*Parameter OPTION is **ignored** by the SSRV command if the service is JAS.*

TERMINATE_SERVICE (TSRV)

The TSRV command terminates the JAS on the complex and sets the state of the JAS to EMPTY on the two members.

If TSRV is issued to a JAS currently used by a *WATCHED BY CMSC* TDS (HA-type or non-HA), CMSR will reject the command:

EXAMPLE:

```
S: TERMINATE_SERVICE BLUE;
   CM19  TSRV BLUE REJECTED : BLUE STILL USED
```

TSRV issued to a JAS not currently used by a *WATCHED BY CMSC* TDS (HA-type or non-HA), will result in a *Weak Termination.*

If FORCE (FORCE=1) is specified, a *Strong Termination* of the JAS will be performed.

The use of the option FORCE is very dangerous since any requested journalization action following the command will provoke the abort of the step which requested it such as checkpoint, commit and open_file.

❑

NOTE:
Parameter OPTION is ***ignored*** *by the TSRV command if the service is JAS.*

Weak Termination of an HA-type JAS

- A *Weak Termination* of an HA-type JAS consists of terminating all activity on this JAS. From the ACTIVE state, it can set the state of the JAS to EMPTY or BACKUP depending on the origin of the order sent by the CMSC to the JAS:
  - a TAKEOVER_MEMBER command specifying OPTION=WEAK sets the JAS to BACKUP,
  - a TSRV jas-name command issued by the operator sets the JAS to EMPTY.

- The HA-type JAS will take the following measures:
  - If a job is already *connected* to this JAS, the executing step will terminate normally.
  - Any new connection attempt by a step will be refused:

    for a TDS the initialization session will fail,

    for a Batch or IOF application, opening a file protected by this JAS will be rejected.
  - The message JP79 will appear repeatedly at the console.

    To terminate the JAS, the operator should display the list of steps connected to this JAS through the DISPLAY_JAS command in order to *disconnect* each step from the JAS.

    If the operator does not intervene, the JAS will terminate when all *connected* steps terminate.

Strong Termination of an HA-type JAS

- A *Strong Termination* of an HA-type JAS can be caused by:
  - a TSRV jas_name OPTION=FORCE command issued by the operator,
  - a TSRV OPTION=FORCE order issued by the CMSC as a result of a TCMSR OPTION=FORCE command entered by the operator.
- The HA-type JAS will take the following measures:
  - If a job is already *connected* to the JAS, the executing step will continue normally but any control request such as checkpoint, commit, open_file and step termination subsequent to the command will cause the requesting step to abort.
  - Any new connection attempt by a step will be refused:

    - for a TDS the initialization session will fail,
    - for a Batch or IOF application, opening a file protected by this JAS will be rejected.
  - The message JP79 will appear repeatedly at the console. To terminate the JAS, the operator should display the list of steps connected to this JAS through the DISPLAY_JAS command in order to *disconnect* each step from the JAS. See *Display Information about JAS* (section 5.5.3).
  - The JP78 message will be sent to the JORs of all the steps *connected* to this JAS and will be displayed at IOF user terminals connected to this JAS.

TAKEOVER_MEMBER (TKMB)

OPTION=STRONG or *OPTION=FORCE*

When Takeover occurs either when a TAKEOVER_MEMBER command (STRONG or FORCE) is issued or following a System Crash, SWITCHABLE HA-type JAS in an ACTIVE state on the *failed* member are taken over by the other member *still alive*. These JAS change in state from BACKUP on the *still alive* member to SWITCHABLE ACTIVE. HA applications depending on these JAS can therefore restart with the minimum of delay.

During switching, all the files protected by these JAS undergo *immediate recovery*.

CMSC directs every switchable HA-type JAS in the BACKUP state on the *still alive* member to switch to the ACTIVE state. For each JAS concerned, a job called the ***Recovery Step*** (H_RECOV) is started to perform *immediate recovery* on the files protected by this JAS.

If the HA environment has been correctly configured with all the files necessary for takeover being allocated on shared disks, and if these disks are accessible at the time of takeover, immediate recovery requires no operator intervention. After recovery, files are once again consistent and stable, and the resources for journalization on the member where takeover took place become available. The JAS is now ready to revert to ACTIVE on this member. HA applications using this JAS can be started on this member.

OPTION=WEAK

This option activates a *WEAK Termination* of the HA-type JAS that were SWITCHABLE and in an ACTIVE state on the co-member. The *formerly* ACTIVE JAS are set to BACKUP and the *formerly* BACKUP JAS are set to ACTIVE. There is no need for file recovery since the JAS terminate only when there is no longer any activity on them.

As there is no guarantee of the unavailability time, the operator may have to intervene to speed up the process by terminating a few applications using HA-type JAS. The DISPLAY_JAS command shows which applications use the JAS.

5.5.3 Display Information about JAS

The information concerning the JAS may be of different types:

- It may be dynamic information about the JAS service such as the state of the two servers, which is given by the CMSC management operator command DSRV.
- It may be information about the protection of the user files by the JAS, which is given by the MNJAS DPLK command and is available to the user.
- It may be dynamic information about the steps which use the JAS, which is given by the operator command DISPLAY_JAS.

DISPLAY_SERVICE (DSRV)

See *High Availability Administrator's Guide*.

The DSRV (CMSC) command displays information about the JAS on the complex, namely:

- the status (EMPTY, ACTIVE, BACKUP, SWITCHABLE or UNSWITCHABLE) of the JAS on each member,
- and the list of TDS that use this JAS as declared at CRCXGEN (complex generation) time.

EXAMPLE:

Consider a configuration of the following complex:

- the members are SIT1 and SIT2,
- BLUE JAS is SWITCHABLE in ACTIVE state on SIT1 and in BACKUP state on SIT2,
- TDS1 and TDS2 are HA-type TDS using this JAS.

The information supplied by the DSRV command on such a complex is:

```
S: DISPLAY_SERVICE BLUE;

   <time> <CM> SERVICE BLUE:

   TYPE  SIT1:REQ_STATE         EFF_STATE
   JAS        ACTIV_SWITCHABLE  ACTIV_SWITCHABLE

         SIT2:REQ_STATE         EFF_STATE
              BACKUP            BACKUP

   MAIN_SERVICE: TDS1, TDS2
   NO USED SERVICES
```

NOTE:
The CMSC commands, DISPLAY_MEMBER (DMB) and the DISPLAY_COMPLEX, display information about the members and the complex and the state of their respective services.

❑

MNJAS Option DISPLAY_LINK (DPLK)

See *DISPLAY_LINK* (section 3.2.5.4).

The MNJAS DPLK command displays the link between a private catalog and a JAS. If no MNJAS LINK command has been issued for a catalog, the catalog is implicitly linked to SYS JAS.

EXAMPLE:

TDS1.CATALOG is the catalog of an HA-type TDS which has been linked to the BLUE JAS.

The effect of the DPLK command will be:

```
S: MNJAS;

M: DISPLAY_LINK  TDS1.CATALOG;

   TDS1.CATALOG
   IS LINKED TO JAS BLUE
```

When 2 SITE.CATALOGs are used, it is recommended to execute the DPLK command on both systems for each user file catalog to ascertain that MNJAS LINK has been executed on ***both*** *systems for each catalog.*

Where MNJAS LINK has only been executed on one system, the system will detect the anomaly and DPLK command will display the following message:

```
TDS1.CATALOG
HAS INCONSISTENT LINK: RETYPE LINK COMMAND.
```

If a TDS is started in ACTIVE state on the system where the MNJAS LINK command was last executed, no anomaly will be signalled. *However inconsistency in the link will prevent files being opened at TDS restart on the* ***other*** *system if takeover occurs. File opening will be refused with the code* `JAP 3, SHLVVIOL` *being returned.*

❑

DISPLAY_JAS (DJAS)

- See *DISPLAY_LINK* (section 3.2.2).
- The DISPLAY_JAS command displays Jobs that are *connected* to the JAS at the moment the command is issued. It gives for the specified JAS and for each *connected* Job:
  - the RON (run occurrence number),
  - the Job name,
  - the name of the submitter,
  - the Job Class,
  - and the Load Module in execution.
- In particular, the DISPLAY_JAS command can be used before starting a *weak* Takeover to list the TDS, Batch and IOF applications for selecting those Jobs to be stopped.

EXAMPLE:

- Consider the following configuration:
  - TDS1 is an HA-type TDS running on BLUE JAS
  - BAT1 is a Batch application running on SYS JAS
  - and BAT2 is a Batch running on BLUE JAS
- The effect of the DISPLAY_JAS will be:

```
S: DISPLAY_JAS   JASNAME=BLUE;

   time JAS:BLUE   X151.1  TP7JCLAC  OPERATOR P  TDS1
                   X168.2  BAT2      USER2    P  H_BAT2

S: DJAS;

   time JAS:SYS    X167.5  BAT1      USER1    P  H_BAT1
        JAS:BLUE   X151.1  TP7JCLAC  OPERATOR P  TDS1
                   X168.2  BAT2      USER2    P  H_BAT2
        JAS:GREEN  NO JOBS LINKED TO THIS JAS
```

❑

5.6 HA-Type JAS and Recovery

After system crash, rollback and rollforward are performed for file recovery:

- during *Takeover* when the HA-type JAS that were SWITCHABLE, are ordered to switch from the BACKUP state to ACTIVE on the coupled system,

  or

- at Restart of the *failed* member when the recoveries of HA-type JAS that were UNSWITCHABLE and SYS JAS are performed at the same time.

HA-type JAS on a complex are **both** either SWITCHABLE or UNSWITCHABLE. When a crash occurs,

- either *Takeover* occurs to ensure the recoveries of the HA-type JAS; recovery for SYS JAS will occur on the restart of the *failed* system (HA-type JAS are SWITCHABLE),
- or no *Takeover* occurs, in which case, the recoveries for both HA-type JAS and SYS JAS will take place on the restart of the *failed* system (HA-type JAS are UNSWITCHABLE).

Recovery during *Takeover* is mutually exclusive to recovery at Restart of the *failed* system. A lock mechanism (HA-LOCK) prevents both recoveries executing simultaneously. If *Takeover* occurs, the Restart of the *failed* member will be held until the process of *Takeover* has completed. The process of *Takeover* involves recovering user files protected by the HA-type JAS. When the *failed* member is then restarted, user files protected by SYS JAS will be recovered.

NOTE:
Some JRNAL errors logged in the system error logging file during this restart should be considered as normal.

During *Takeover*, CMSC switches the SWITCHABLE HA-type JAS from *BACKUP* to *ACTIVE* state thereby starting the **Recovery Step** (H_RECOV) for each JAS. Two H_RECOV steps may therefore run simultaneously, one recovering the user files protected by JAS BLUE and the other recovering the user files protected by JAS GREEN.

Every journalized file is recovered. CMSC restarts TDSs declared *WATCHED BY CMSC*. However, Batch and IOF applications are not restarted.

If a Batch or IOF application was updating files protected by an HA-type JAS, these files will be recovered up to the beginning of an FRU where the last checkpoint was taken. The application cannot be restarted without intervention since the REPEAT option is inhibited with HA after system crash. If the step is repeated, the code `JAP 2, ABORTPG` will be returned. The application must therefore be designed to take into account a restart procedure.

The termination of the **Recovery Step** is either SUCCESSFUL or FAILED:

- SUCCESSFUL termination results in the Journal resources being in a state to enable journalization on the former BACKUP system. Files are once again in a consistent state and the resources needed for journalization on this member are available.

  CMSC then switches the state of the JAS concerned to ACTIVE and journalization can resume on this member.

- FAILED termination is due to some Journal resources mandatory for journalization on the former BACKUP system not being accessible or operational.

  The state of the JAS will not switch to ACTIVE and the JAS will continue to remain in the BACKUP state.

If the **Recovery Step** cannot even be started because the JAS Directory is inaccessible on the BACKUP system, the state of the JAS will be set to EMPTY. See *JAS Directory not Accessible* (section 5.9.1).

Although recovery is automatic, the operator may be asked to intervene in the case where some files cannot be accessed. If however, the operator is not present on the site, the recovery step will take the decision itself on whether to stop or to proceed with as much recovery as possible and allow journalization to start on this member.

5.7 Incidents During Takeover

The Recovery Step in charge of immediate recovery on a system coupled to one that has *crashed*, has to take into account circumstances that can affect Takeover, namely:

- non-accessibility of a file,
- detection of date inconsistency between the two systems,
- a TDS not being able to supply the list of aborted commitments to the Recovery Step,
- the After Journal being corrupted,
- and a further *crash* occurring during the execution of the recovery step.

5.7.1 File Inaccessibility

This generally happens if the coupled system has not been configured with the compulsory requirement for HA in that:

- ***either the file has not been allocated on a sharable disk,***
- ***or the disk is not effectively shared when the takeover takes place.***

A file can also be inaccessible if it is currently being accessed by another step. In this case, the step is arbitrarily *killed* by the Recovery step. If the step was repeatable, it is not restarted.

Files concerned with the processes of recovery and of future journalization on the system previously BACKUP, are considered either essential or non-essential. Non-essential files, however, can be essential to a specific HA application such as a user file.

5.7.1.1 Essential Files

Files that are essential to the processes of recovery and/or journalization are:

- the JAS directory, BLUE.JADIR and/or GREEN.JADIR,
- *After Journal* files, BLUE.JA.JiVolname and/or GREEN.JA.JiVolname,
- and the SYS.JRNAL of the failed system.

The operator is first informed that the file is inaccessible. An *Intervention Request* appears in message RY01 followed optionally by message RY02 for the operator to intervene to make the file available:

- if the operator makes the file accessible within the permitted delay (about 2 minutes), through a RDV command for example, recovery proceeds,
- if the operator does not make this resource available to the system within the permitted delay, recovery does not take place and the recovery is declared FAILED; the JAS will not be able to switch to ACTIVE on this member at the end of the recovery step and will remain in the BACKUP state.

NOTE:
In this case, there is no *Decision Request*, see the following Paragraph.

5.7.1.2 Non-Essential Files

Files that are non-essential to the processes of recovery and/or journalization are:

- *Before Journal* extents and the TDS Before Journal,
- and user files.

If a non-essential file is missing, journalization on this system can still take place but not on the file concerned:

- either because the user file is inaccessible,
- or because the file cannot be recovered since the *Before Journal* cannot be accessed; for example, if the TDS Before Journal is missing, TDS system files cannot be set back to a stable state, and the TDS cannot be restarted.

A message first appears at the console to inform the operator that the file is inaccessible.

An *Intervention Request* is then issued to the operator to make the file available:

- if the operator can make it available, the recovery proceeds further.
- if the operator cannot make the file available, a *Decision Request* appears in message RY11 for the operator to decide if recovery is to proceed:
  - in the absence of an answer within the permitted delay, recovery will proceed,
  - if YES, recovery will proceed as far as possible,
  - if NO, recovery is not attempted and is declared FAILED; the JAS will not switch to ACTIVE on this member.

- if the operator does not answer within the permitted delay, the access to the file will be reattempted another time in case another job was accessing the file exclusively and has now released it. If the file still cannot be accessed, the Decision Request message will appear. Since the operator considered absent will no longer be consulted, the repeated attempt to access a file can only occur once during the recovery step. The sequence of messages printed in the output of the recovery step shows that the access has been attempted twice and that the recovery has been performed on this file if the second access was successful.

Recovery is declared SUCCESSFUL if no *Recovery Failed* is encountered. CMSC is informed that the HA-type JAS can be switched on this member and restart its activity. On Takeover, the TDS resumes.

However, some files may be in an unstable state if problems occur. See Note in *Decisions Affecting File Recovery* (section 5.7.1.3).

5.7.1.3 Decisions Affecting File Recovery

Errors detected by the recovery step will

- either cause the step to be interrupted, in which case recovery can still take place:
  - either on the *failed* system when it restarts,
  - or on the *backup* system when another takeover is attempted.
- or allow the step to continue depending on the severity of the error and on the decision taken by the operator.

The conditions for proceeding with the recovery are the following:

- if the recovery step is interrupted and therefore Takeover has failed, the *failed* system can be restarted as a preferable solution; recovery takes place at system restart if all the resources are accessible from this member,
- if the *failed* system cannot be restarted, the resources which were not accessible at takeover time must be made accessible whatever the cost and the takeover should be once again attempted with the command `TAKEOVER_MEMBER FORCE`; this command starts another recovery step,
- if the resources essential to Recovery and Journalization are still unavailable, the recovery step will continue to fail,
- if the resources are still unavailable but non-essential to Recovery and Journalization, Takeover can still take place if the operator answers YES to message RY11 or does not answer at all; however, some files may be unstable and no further immediate recovery will be possible for such files.

NOTE:

If a user file is not accessible to the recovery step, and if the operator chooses to continue, only this file will be left in an inconsistent and unstable state.

If a TDS Before Journal or a Before Journal Extension is not accessible to the recovery step, and if the operator chooses to continue, all the journalized files, formerly accessed by the journalizing step will be left in an inconsistent and unstable state. If among them is a TDS system file such as TDS.CTLM, TDS will not be able to start on the Backup system.

Recovering unstable files involves restoring the files from a previous save, then using the ROLLFWD utility on these files.

Recovery can also take place without operator intervention. If the operator does not answer either **once** to an *Intervention Request* or **once** to a *Decision Request*:

- timeout occurs after about 2 minutes,
- the operator is considered absent from the site,
- and no further requests are made until the end of the recovery.

If a *Decision Request* is not answered by the operator, the system will decide to continue the recovery, because the inaccessible file is not essential to the process of recovery and journalization.

The outcome of an HA-type JAS recovery is independent of the outcome of another HA-type JAS recovery.

EXAMPLES:

```
RECOVERY SUCCESSFUL can be displayed for BLUE JAS
```

whereas

```
RECOVERY FAILED can appear for the GREEN JAS
```

Only BLUE JAS will be able to switch to ACTIVE state.

❑

5.7.2 Difference in Date and Time Between Systems

Both Members of the Complex must have the same system time. It is important to offer application users continuity of system time since, in the event of a system crash, HA reduces non-availability of TDS applications running under GCOS 7 to a few minutes. Moreover, if a system time difference exists between the two members, **predating** on the HA-type JAS involved may occur at takeover time.

This is the third case that can lead to a predating situation (predating can also occur after a system restart with date change or through the MODIFY_TIME command). See "*Setting the System Date and Time*" in Chapter 2.

A comparison of the results of the DISPLAY_TIME command performed on the two members of the complex gives a good indication of the time overlap that might be generated.

The same constraints on time overlap apply with this kind of predating:

- Rollforward BEGDATE

  If a Static Rollforward has to be undertaken and a time overlap exists:

  - specify the BEGDATE parameter for the Rollforward **explicitly**,
  - specify a BEGDATE **outside the time overlap**. The BEGDATE must match the date of the save you want to restore. However, if the date of the save you want to restore falls inside the time overlap, specify another BEGDATE just prior to the beginning of the time overlap and make use of the SKIPERR parameter.

  In general, do not perform saves (either static or dynamic) inside the time overlap.

- Rollforward ENDDATE

  If a Static Rollforward has to be undertaken but there is a time overlap, wait for the end of the time overlap before sending the Rollforward command.

At initial synchronization of the CMSRs (CMSC servers), the system time difference between the two Members of the Complex is checked. If this difference is greater than one minute, the following message is issued:

```
CM37 WARNING : THE TIME IS NOT THE SAME ON THE TWO MEMBERS,
               PREDATING MAY OCCUR AT TAKEOVER TIME.
```

This message points out that the internal clocks of the two Members are shifting in a very divergent way. Contact the Service Center to analyze these shifts.

No particular action is required immediately on the complex, but pay specific attention to predating in the case of a Takeover.

Another alternative is to resynchronize the two system times using the MODIFY_TIME command. If this will cause a predating situation, check first that the constraints on predating can be easily satisfied. Remember that when you use the MODIFY_TIME command, the time deviation (TIMEDEV) of the site is also modified. Do not forget, therefore, to check the TIMEDEV parameter when you subsequently restart GCOS 7.

5.7.3 Journal Recovery on Takeover

5.7.3.1 TDS Aborted Commitment Units

The Recovery Step needs to know which CUs (commitment unit) are aborted and which units have been committed for the necessary rollback and rollforward operations. If a TDS cannot access its RECOV file or its SWAP files, it is unable to supply the list of its aborted commitment units.

When this happens, the Recovery Step will itself supply a substitute list of aborted CUs. This list is obtained by reading the After Images in the Primary After Journal files. The list represents the *worst* case in that some CUs ready to be committed may be considered committed. The files accessed by these CUs may be set to their next consistent state instead of being rolled back to their previous consistent state. The information that the recovery step has taken its own decisions and the After images which have been invalidated, are printed in the output of the recovery step. Check the result.

If the Recovery Step was not able to supply this list, the After Journal would be corrupted. This is why this specific function is considered as a Journal Recovery function, and called a JRU function.

NOTES:

This specific function of the Recovery Step is very rarely used. The Administrator, however, should be aware that when this function is used, dynamic rollforward although declared *successful* will not be performed for this TDS. This is the reason why all the images that might have been applied are printed in the Output of the Recovery Step.

This specific function of the Recovery Step can only be effective if journalization on the files is AFTER or BOTH. If the files are only protected by the *Before Journal*:

- no commit will be considered aborted,

- and no rollback will be performed.

In an HA environment, it is therefore highly recommended to select HIGH protection at TDS generation.

5.7.3.2 Corrupted After Journal

The After Journal may be corrupted when a TDS running on the *active* system aborts before the system crashes and:

- either the TDS is unable at that moment to provide the list of aborted commitment units for the recovery,
- or the list cannot be stored in the JAS Directory.

Since the Recovery Step can only deal with applications which were running at the time when the system crashed, it does not perform any recovery for that TDS. The *After Journal* is corrupted because it is not able to know which After Images registered by that TDS are valid.

The sequence of recovery is as follows:

- two Steps of the H_RECOV Load Module are executed sequentially without operator intervention for this JAS (see Note):
  - the first RECOV for immediate recovery,
  - the second JRU (Journal Recovery Utility) for *After Journal* recovery,
- the JAS is then able to switch to the ACTIVE state.

NOTE:

Since Takeover may occur for both BLUE JAS and GREEN JAS, up to four H_RECOV Steps may execute.

5.7.4 System Crash During Recovery at Takeover

Let A be the system which has crashed first and B the system on which the recovery takes place.

What happens if B crashes during the takeover recovery?

Recovery works on the principle that it can always resume from wherever it stops on whichever member. File integrity is guaranteed.

The HA-LOCK mechanism is designed to privilege restarting System B before System A.

When System A restarts, the console dialog is as follows:

- first the message:

```
CM93 WARNING HALOCK RETAINED BY CO-MEMBER.
MEMBER member-name IS WAITING FOR IT TO BE RELEASED
```

- then the following question:

```
CM94 WARNING HALOCK STILL RETAINED BY CO-MEMBER.
MEMBER member-name IS WAITING FOR IT TO BE RELEASED.
IF CO-MEMBER NOT OPERATIONAL, REPLY <FORCE> TO THIS MESSAGE.
```

Recommended Procedure

Restarting System B should be privileged, therefore this question should not be answered.

When System B restarts, the recovery which was being performed by the recovery step before the crash is not resumed on System B. CMSR on System B will start the services previously *active* on System A and set them to the *backup* state.

Once these services are in *backup* on System B, HA-LOCK on System B will be released and will enable System A restart to proceed. Immediate recovery of the user files is then performed on System A, and the services are started and set to *ACTIVE* on System A.

***Alternative Procedure*:**

If restart of System B cannot be performed.

If System B cannot reach the point where HA-LOCK is released (it systematically crashes when it restarts for example), the HA site administrator must:

- stop activity on System B completely, wait on ISL options,
- answer FORCE to the question CM71 asked on System A.

  As a consequence of FORCE, HA-LOCK allows the restart of System A. User files will be recovered. CMSR on System A will start the services in the *BACKUP* state.
- enter the command TAKEOVER_MEMBER OPTION=FORCE on System A.

  This command will not cause the startup of a recovery step for the JAS *active* on System A before the first crash. Recovery was already performed at System A restart. However, the services are switched to the *ACTIVE* state.

  If there was one JAS *active* on System B before the first crash, TAKEOVER_MEMBER OPTION=FORCE will trigger the startup of a recovery step for that JAS and will switch that JAS to the *ACTIVE* state.

  When System B restarts, CMSR will start the services in the *BACKUP* state.

In both procedures, recovery takes place on System A.

Performing COLD or CLEAN restart on System A precludes any recovery of user files. To prevent the restart of system A until the user files are completely recovered, if COLD or CLEAN restart is absolutely necessary on system A, the HA site administrator should force recovery on System B:

- stop activity on System A completely, wait on ISL options,
- let system B restart: the services will be started and set to *BACKUP*,
- execute the command TAKEOVER_MEMBER OPTION=FORCE on System B to initiate the startup of the recovery step and to start the services in the *ACTIVE* state.

  When System A restarts, CMSR on System A will start the services in the *BACKUP* state.

5.8 Output of the Recovery Step

All the messages which report the way the recovery was performed are sent to the SYSOUT of the recovery step. They include:

- the list of files to be recovered,
- for each step, the messages which report the result of the rollback and the result of the rollforward for each file,
- and all the consulting messages and error messages sent either to the console and/or to the JOR of RECOV,

which enable establishing the whole sequence of processing by the recovery step.

EXAMPLE:

When Takeover occurred, two HA-type TDSs, TDS2 Ron X305 and TDS5 Ron X306, were:

- running on BLUE JAS,
- and sharing the same files.

Recovery was performed by Job X268 named RECOV on site BP59:

The SYSOUT of RECOV X268:2 appears overleaf.

```
*******************************************************************
*AUTOMATIC RESTORE FILE INTEGRITY FOR FILES PROTECTED BY JAS BLUE*
**** GCOS7
****                        R E C O V
****                     VERSION: V610   DATED: JAN 31,1991  ****
*******************************************************************
*******************************************************************


-------------------------------------------------------------------
USER FILES TO BE POTENTIALLY RECOVERED
-------------------------------------------------------------------
FILE NAME                                    PMD     JOURNAL
-------------------------------------------------------------------
TDS2.CTLM                                    UP      BEFORE
GIFB.UFASREL                                 UP      BOTH
GIFB.UFASREL1                                UP      BOTH
GIFB.UFASREL3                                UP      BOTH
GIFB.UFASISEQ                                UP      BOTH
GIFB.UFASISEQ3                               UP      BOTH
GIFB.UFASSEQ                                 UP      BOTH
TDS5.CTLM                                    UP      BEFORE
GIFB.UFASREL                                 UP      BOTH
GIFB.UFASREL1                                UP      BOTH
GIFB.UFASREL3                                  UP      BOTH
GIFB.UFASISEQ                                  UP      BOTH
GIFB.UFASISEQ3                                 UP      BOTH
GIFB.UFASSEQ                                   UP      BOTH
------------------------------------------------------------------
------------------------------------------------------------------
FILE RECOVERY REPORT
------------------------------------------------------------------
------------------------------------------------------------------
X305 . 1      TDS TDS2                      JOURNAL OPTION: BOTH
------------------------------------------------------------------
*** FILE: TDS2.CTLM
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL1
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL3
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASISEQ
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASISEQ3
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASSEQ
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 10.
*** FILE: GIFB.UFASREL1
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASREL3
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASISEQ
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASISEQ3
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
------------------------------------------------------------------
X306 . 1      TDS TDS5                    JOURNAL OPTION: BOTH
------------------------------------------------------------------
*** FILE: TDS5.CTLM
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL1
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL3
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASISEQ
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 23.
*** FILE: GIFB.UFASISEQ3
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASSEQ
```

```
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASREL1
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASREL3
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASISEQ
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASISEQ3
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
-------------------------------------------------------------------
RY50    BLUE: RECOVERY SUCCESSFUL.
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
10:51:59  BP59:X268 OPERATOR RECOV  INSTALL  OPERATOR  DEC 03,1991
```

❑

EXAMPLE OF RECOV USING A JRU FUNCTION:

When Takeover occurred:

- an HA-type TDS, TDS3 Ron X51, was running on the BLUE JAS,
- recovery has been performed by the Job X102 named RECOV.

The Sysout of RECOV X102:2 appears as follows:

```
******************************************************************
AUTOMATIC RESTORE FILE INTEGRITY FOR FILES PROTECTED BY JAS BLUE
*** GCOS7                                                     ****
****                        R E C O V                         ****
****                    VERSION: V610   DATED: JAN 31,1991    ****
******************************************************************
******************************************************************

------------------------------------------------------------------
USER FILES TO BE POTENTIALLY RECOVERED
------------------------------------------------------------------
FILE NAME                                      PMD     JOURNAL
------------------------------------------------------------------
TDS3.CTLM                                      UP      BEFORE
GIFB.UFASREL                                   UP      BOTH
GIFB.UFASREL1                                  UP      BOTH
GIFB.UFASREL3                                  UP      BOTH
GIFB.UFASISEQ                                  UP      BOTH
GIFB.UFASISEQ3                                 UP      BOTH
GIFB.UFASSEQ                                   UP      BOTH
------------------------------------------------------------------

-------------------------------------------------------------------
FILE RECOVERY REPORT
```

```
------------------------------------------------------------------
******************************************************************
RECOV PROVIDES THE LIST OF ABORTED COMMITMENTS UNITS FOR TDS: TDS3
******************************************************************
******************************************************************
*                                                                *
*                Commitment unit declared aborted                *
*      List of file images belonging to this commitment unit     *
*                                                                *
******************************************************************
EFN: GIFB.UFASREL
Operation type (Update)
adresse
    0 0000 00000000
Record image
    0 0000 F0F0F0F0 ... 4040F1F5 0000000001UFAS-RELAT91 11 25  15
   32 0020 40F3F540 ... 40404040  35 41 70       01
   64 0040 40404040 ... 40404040
   96 0060 40404040
Operation type (Update)
adresse
    0 0000 00000001
Record image
    0 0000 F0F0F0F0 ... 4040F1F5 0000000002UFAS-RELAT91 11 25  15
   32 0020 40F3F540 ... 40404040  35 41 79       02
   64 0040 40404040 ... 40404040
   96 0060 40404040
******************************************************************
*                                                                *
*                Commitment unit declared aborted                *
*      List of file images belonging to this commitment unit     *
*                                                                *
******************************************************************
EFN: GIFB.UFASISEQ
Operation type (Delete)
Record key
    0 0000 F0F0F0F0 ... F0F10000 0011                  0000000001
Operation type (PUT/WRITE)
Record key
    0 0000 F0F0F0F0 ... F0F10000 0012                  0000000001
Record image
    0 0000 F0F0F0F0 ... 4040F1F5 0000000001UFAS-INDEX91 11 25  15
   32 0020 40F4F140 ... 40404040    41 39 53     01
   64 0040 40404040 ... 40404040
    ======
 1952 07A0 40404040 ... 40404040
 1984 07C0 40404040 ... 40404040
Operation type (Delete)
Record key
    0 0000 F0F0F0F0 ... F0F20000 0013                  0000000002
Operation type (PUT/WRITE)
```

```
Record key
    0 0000 F0F0F0F0 ... F0F20000 0014                  0000000002
Record image
    0 0000 F0F0F0F0 ... 4040F1F5  0000000002UFAS-INDEX91 11 25 15
   32 0020 40F4F140 ... 40404040    41 40 63     02
   64 0040 40404040 ... 40404040
    ======
 1952 07A0 40404040 ... 40404040
 1984 07C0 40404040 ... 40404040
*****************************************************************
THE IMAGES EDITED HERE ABOVE ARE CONSIDERED AS INVALID BY RECOV.
*****************************************************************
-----------------------------------------------------------------
X51  . 1    TDS TDS3                        JOURNAL OPTION: BOTH
-----------------------------------------------------------------
*** FILE: TDS3.CTLM
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL1
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL3
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASISEQ
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 11.
*** FILE: GIFB.UFASISEQ3
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASSEQ
ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS 0.
*** FILE: GIFB.UFASREL
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASREL1
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASREL3
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASISEQ
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
*** FILE: GIFB.UFASISEQ3
DYNAMIC ROLLFORWARD SUCCESSFUL. NBR OF ROLLED FORWARD RECORDS: 0.
-----------------------------------------------------------------
RY50    BLUE: RECOVERY SUCCESSFUL.
```
❑

5.9 Abnormal Events in an HA Environment

This paragraph deals with serious failures which may happen in a HA environment which may be related to the takeover operation. Let A be the failed system and B the backup system.

The failures can be classified according to their cause:

- the Directory of the JAS is not accessible, either on member B at takeover time or on member A at system restart time. See *JAS Directory not Accessible* (section 5.9.1).
- the state file of the JAS is not accessible either on member B at takeover time or on member A at system restart time, or when the JAS service is started. See *JAS State File not Accessible* (section 5.9.2).
- a TDS is not able to supply the list of aborted commitments. See *List of Aborted TDS CU not Supplied* (section 5.9.3).

Some of these failures can be combinations of causes.

5.9.1 JAS Directory not Accessible

5.9.1.1 JAS Directory not Accessible at Takeover

Recovery cannot take place at takeover. The recovery step is not even started. Message JP74 is sent. The corresponding JAS service is set to EMPTY state on system B.

Since no recovery is possible on this system, the HA-LOCK held by B will be quickly released, and recovery will take place on the other system A at restart, provided that the JAS Directory is available from that system. The corresponding JAS service is set to BACKUP state on system A.

The System Administrator should:

- stop this service through the use TERMINATE_SERVICE command (TSRV jasname),
- make the JAS Directory available from system B,
- and perform a START_SERVER command (SSRV jasname) to set the service ACTIVE on system A and BACKUP on system B as it was before the first system failure.

5.9.1.2 JAS Directory not Accessible at System Restart

This will be detected at the first activation of the JAS which succeeds in accessing the JAS Directory.

Two cases must be considered:

- either the journalization had taken place on the same member and:
  - **there was no HA-type TDS** using the JAS at the time of crash, therefore there was no takeover. User file recovery could not be done, in which case, JP43 appears at the console. The activation of the JAS starts an H_RECOV step which performs the JRU functions. User files remain unstable, but they can be recovered with FILREST and ROLLFWD utilities,
  - **there was an HA-type TDS** using the JAS at the time of crash. If the JAS Directory was not available for the recovery at takeover time, as described in the preceding paragraph, immediate recovery needed after the first system failure still could not be performed,

    The activation of the JAS causes message JP75 to appear at the console, declaring that recovery is needed. The System Administrator can:
  - provoke the crash of system A and execute command TKMB FORCE so as to start the immediate recovery on system B, as explained in documentation on message JP75. See *Messages from File Recovery Facilities* (appendix B)),
  - execute the START_SERVICE command for the JAS, which will start a H_RECOV step which performs the JRU functions,
- or the journalization had taken place on the other member. In this case the activation must be performed through a TKMB FORCE command on this system, after making sure that the other system is not operational, so as to activate the immediate recovery with an H_RECOV step.

If the JAS Directory remains inaccessible, no START_SERVICE command can function. The service remains in the EMPTY state. The JAS Directory should be rebuilt by using the MNJAS REBUILD command. See *Incidents on JAS directory* (section 2.2.5.2) and *REBUILD_DIRECTORY* (section 3.2.5.13).

5.9.2 JAS State File not Accessible

5.9.2.1 JAS State File not Accessible at Takeover

An inaccessible JAS state file does not prevent recovery from taking place at takeover as long as the JAS Directory can be accessed. After the recovery step is completed, message JP73 (02) is sent. The corresponding JAS service is set to the EMPTY state on system B.

When system A restarts, the JAS service is set to the BACKUP state.

The System Administrator:

- should stop this service through the TERMINATE_SERVICE command (TSRV jasname),
- make the JAS State file available from system B,
- and perform a START_SERVER command (SSRV jasname) to set the service ACTIVE on system A and BACKUP on system B as it was before the first system failure.

5.9.2.2 JAS State File not Accessible at System Restart

User files recovery necessary at system restart will not be performed. This situation is therefore the same as when the JAS directory is not accessible at system restart for which the JP42 message is sent. See *JAS Directory not Accessible at System Restart* (section 5.9.1.2).

5.9.2.3 JAS State File not Accessible at JAS Startup

Message JP73 (01) is sent. The corresponding JAS service remains in the EMPTY state.

5.9.3 List of Aborted TDS CU not Supplied

At System Restart

The consequences of this incident and the way the JAS takes its own decisions about the state of the commitments are described in *Incidents on JAS directory* (section 2.2.5.2) and *Decisions taken by the JAS* (section 2.4.3).

At Takeover Time

The Recovery Step itself supplies a list of aborted Commitments Units, taking its own decisions about the state of the commitments. See *TDS Aborted Commitment Units* (section 5.7.3.1).

NOTE:

In the event of simultaneous failure of both units (power failure, for example), it is recommended to restart the backup unit first, then execute the TAKEOVER FORCE command. Once this command has finished, the primitively active unit can be restarted.

6. ROLLFWD Utility

The deferred recovery operation consists of two phases, see *Deferred Recovery* (section 2.6.2):

- restoring user files from saved files through the FILREST utility (RESTORE_FILE GCL command), see *Data Management Utilities User's Guide (FBO and VBO),*
- and bringing them to an advanced state with the ROLLFWD utility, which is the subject of the Section.

From TS8560, a static rollforward is available on a copy of the file with a different name. This new "Rollforward on Copy" functionality is described in *ROLLFORWARD ON COPY* (section 6.8).

From TS9662, the JCL statement enables you to run a static rollforward with up to 99 files at a time.

6.1 User Interface

The rollforward utility brings restored user file(s) to a given point of restart by using After Images.

GCL and JCL syntax are almost identical. The differences are:

- STEPOPT and SIZEOPT which are JCL parameters,
- BEGDATE and ENDDATE where, in GCL, the year can be input using two or four digits. In JCL, only two digits are used.
- From TS9662, the JCL statement enables you to run a static rollforward with up to 99 files at a time, while the GCL command has a limit of 25 files at a time.

GCL Syntax

```
ROLLFWD  { outfile1 | OUTFILES=(( outfile1 )[...( outfile25 )])}
         [ BEGDATE=[yy]yy.mm.dd/hh[.mm[.ss[.msmsms]]]]
         [ ENDDATE=[yy]yy.mm.dd/hh[.mm[.ss[.msmsms]]]]
         [ DUMP={ NO | DATA }]
         [ SKIPERR={ 5 | dec5 }];
```

JCL Syntax

```
ROLLFWD  { outfile1 | OUTFILES=(( outfile1 )[...( outfile99 )])}
         [ BEGDATE=yy.mm.dd/hh[.mm[.ss[.msmsms]]]]
         [ ENDDATE=yy.mm.dd/hh[.mm[.ss[.msmsms]]]]
         [ DUMP={ NO | DATA }]
         [ SKIPERR={ 5 | dec5 }];
```

Parameter Description

outfile_list: Designates the name of the user file, or the list of user files, to be rolled forward.

Where several outfiles are specified, the parameter OUTFILES must be specified, each outfile being enclosed within brackets.

BEGDATE: Specifies the start date for the ROLLFORWARD. If not present, ROLLFORWARD takes as BEGDATE the last date on which the outfiles were saved. This date is recorded in the JAS directory when the FILSAVE or DYNSAVE utility (SAVE_FILE GCL command) is run. When the last save date for an outfile is not known, the date of the first update is taken as the start date.

This parameter should not be specified in most conditions. Use it only when ROLLFORWARD has to be executed from a save that is not the last save known to the system or to encompass a predating overlap.

[YY]YY.MM.DD/HH[.MM[.SS[.mSmSmS]]]: Year, Month, Day, Hours, Minutes, Seconds, Milliseconds are 2, or 3, or 4 digit numbers separated by a "." or a "/". "Year" can be expressed as two or four digits in the GCL syntax and as two digits only in the JCL syntax.

ENDDATE:	Specifies the date of the outfile status. If not present, ROLLFORWARD applies the after images up to the last image known at the time the utility is activated.
	[YY]YY.MM.DD/HH[.MM[.SS[.mSmSmS]]]: Year, Month, Day, Hours, Minutes, Seconds, Milliseconds are 2, or 3, or 4 digit numbers separated by a "." or a "/". "Year" can be expressed as two or four digits in the GCL syntax and as two digits only in the JCL syntax.
DUMP:	NO (default).
SIZEOPT:	Specifies size options for ROLLFWD equivalent to those in the JCL SIZE statement.
	Default values are SIZE=352 Kbytes and POOLSIZE=256 Kbytes. These values may be modified subject to the following minimums:

```
SIZE >= 120 + (6* (maxCI-4)) in Kbytes
POOLSIZE >= 60 + (6* (maxCI-4)) in Kbytes
```

	where maxCI is the maximum size of the CI in Kbytes.
STEPOPT:	Specifies step options controlling the execution of ROLLFWD. All options except REPEAT are acceptable.
SKIPERR:	Specifies the maximum number of errors beyond which the application of images is interrupted. Default is 5.

SKIPERR and Dynamic Save

When a dynamic save is performed (DYNSAVE utility), only those After Images whose dates are **earlier** than the beginning of the dynamic save are considered to be obsolete. After Images whose dates are **later** than the beginning of the dynamic save are not considered to be obsolete *even if they correspond to updates included in the save.*

Therefore, if the last save was dynamic and the BEGDATE parameter is not specified, the DUPKEY and RECNFD return codes are not counted as errors since they are quite normal in the Dynamic Save/Rollforward process.

If the last save was dynamic **but BEGDATE was specified**, the return codes DUPKEY and RECNFD are still counted as errors.

Constraints

The system determines which JAS is (are) associated to the user file(s) given as a parameter:

- If several files are to be rolled forward -an outfile list is given as parameter- they must be protected by the same JAS. If they are protected by different JAS, ROLLFWD only processes the files protected by the first JAS it encountered and sends warning messages for all the other files.
- If the JAS is an HA-type JAS, ROLLFWD can only be run if the JAS is in ACTIVE state.
- The number of files to be rolled forward at the same time is limited to 25 files. From TS9662, this limit is 99 files if you use the JCL statement.

NOTE:

MNJAS DISPLAY_LINK command enables to check which JAS protects the catalogs of user files.

6.2 Restoring Files Before Running ROLLFWD

Files must first be restored before the ROLLFWD utility is executed.

6.2.1 Date of the Last Save

The system keeps track of the last save date for each journalized file. This date can play an important part in the Save/Restore/Rollforward process. It is used as follows:

- when a Restore/Rollforward is performed, a check can be made so that the restored file corresponds to the last save of the file,
- when a Restore/Rollforward is performed, the BEGDATE parameter of the Rollforward can be ignored,
- After Journal files can be recycled automatically (Automatic Cycling).

Systematic recording of the last save date is highly recommended so that these possibilities are enabled, in particular for the administrator to be able to ignore the BEGDATE.

6.2.2 Save

Some system utilities available for backing up files record the date of last save, others do not. Table 6-1 shows which commands modify the save date and the conditions that apply:

Table 6-1. Utilities that Modify the Last Save Date

GCL/ JCL	Name of Utility	Backup on tape or disk	Conditions	Modify last save date

GCL	SAVE_FILE	Tape	UPDJRNL=1 (default)
			UPDJRNL=0
				Yes
				No

JCL	FILSAVE	Tape	EXPORT not specified EXPORT specified	Yes No
JCL	DYNSAVE	Tape	EXPORT not specified EXPORT specified	Yes No

GCL	COPY_FILE	Disk	UPDJRNL=1
			UPDJRNL=0 (default)
				Yes
				No

JCL	FILDUPLI	Disk	UPDJRNL specified UPDJRNL not specified	Yes No

NOTES:

1. The system updates the date of last save when the save is made through EpochBackup 7.
2. The GCL volume level utility SAVE_DISK and the JCL utilities VOLSAVE and VOLDUPLI do not update the date of last save.

6.2.3 Restore

When a Restore or Rollforward is performed, a check can be made to ensure that the restored file corresponds to the last saved version of the file. Table 6-2 shows which commands check the last save date and which do not.

Table 6-2. Utilities that Check the Last Save Date

GCL/ JCL	Name of Utility	Conditions	Check last save date

GCL	RESTORE_FILE	CKJRNL=1 (default)
		CKJRNL=0
			Yes
			No

JCL	FILREST	NCKDATE not specified NCKDATE specified	Yes No

NOTES:

1. When the restore is done through EpochBackup 7, the system checks the last save date except when the keyword FORCE_JA is used in the RESTORE_FILE or RESTORE_LIST command (these commands are available within the EpochBackup 7 command monitor).
2. ROLLFWD can only operate on a complete restored version of a multivolume file resulting from a consistent set of extents.

6.3 How ROLLFWD Works

ROLLFWD rolls forward up to 25 user files protected by the After Journal of a JAS. From TS9662, this limit is 99 files if you use the JCL statement.

ROLLFWD applies a set of After Images to the restored versions of the files. This set of After Images is delimited by two dates, the start and end dates of the roll forward.

The start date of the roll forward is either:

- the start date specified by the user (BEGDATE),
- or, if no date is specified, (***recommended***):

  - the date recorded by the dynamic save,

  - or the date of the first modification after a static save.

The end date of the roll forward is either:

- the end date specified by the user (ENDDATE),
- or, if no date is specified, (***recommended***):

  - the current date (execution time of the ROLLFWD utility).

ROLLFWD retrieves the identifiers of the journal files it requires and asks, where needed, for the media containing them to be mounted,

ROLLFWD ignores After Images belonging to incomplete updates (incomplete updates are those that were either rolled back using the Before Journal or not written to the user files using Deferred Updates applicable to TDS). This ensures that, when ROLLFWD is completed, the files are in a consistent state.

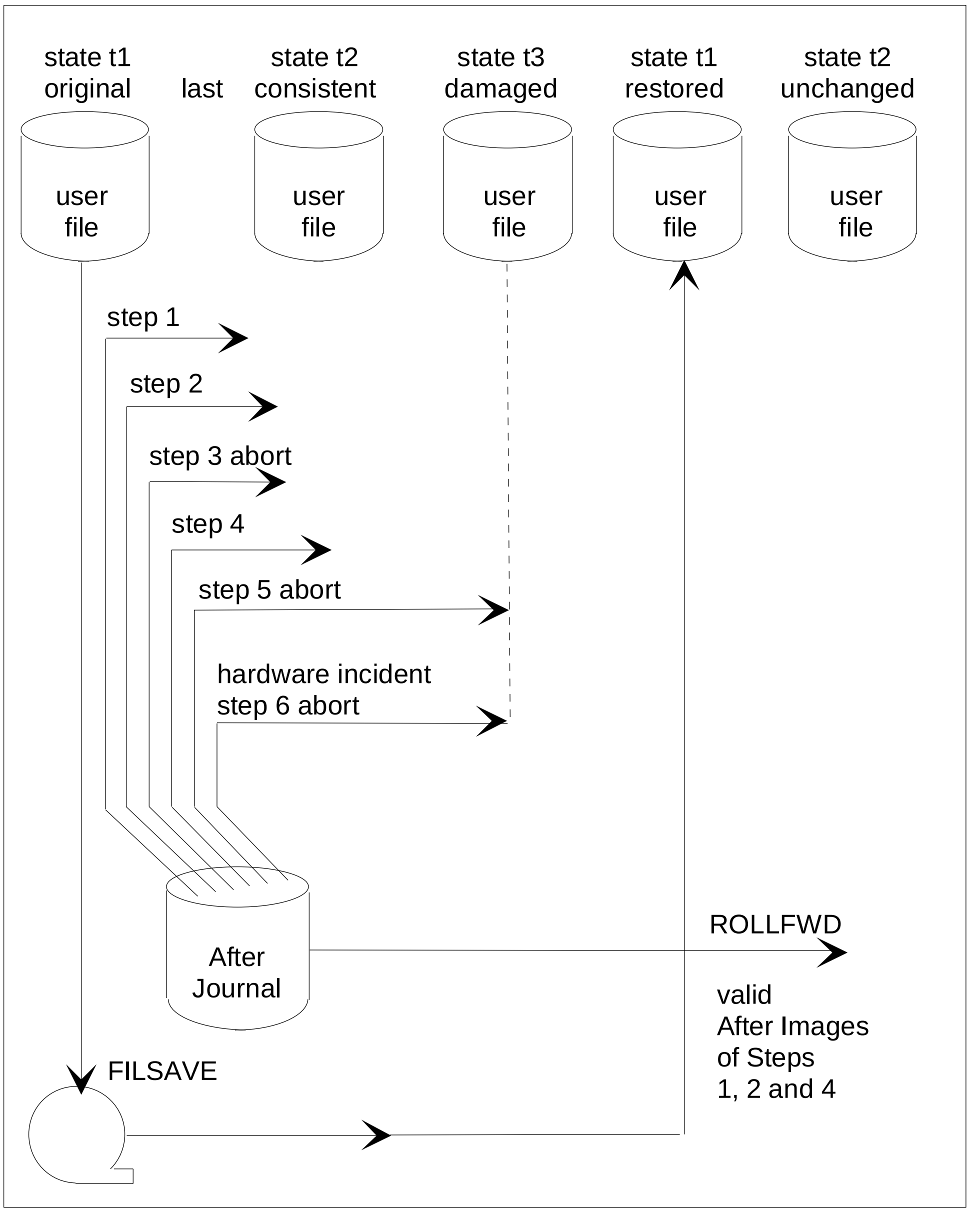


Figure 6-1. Operation of the ROLLFWD Utility

6.4 Resource Management

6.4.1 Configuration Requirements

Files opened in UPDATE processing mode by ROLLFWD, are assigned with the option MOUNT=ALL so that they will *all* be online while the utility is running.

A ROLLFWD step requires either a tape handler or a disk drive to read the journal files depending on the media of the After Journal files.

6.4.2 File Assignments

Since ROLLFWD must be the only step working on files to be rolled forward, it assigns them with the option ACCESS=RECOVERY to ensure exclusive access.

A *file cataloged with access rights* can be submitted to ROLLFWD only by a user belonging to a project defined with RECOVERY.

6.4.3 Simultaneity of Actions Performed on a JAS

Files can be rolled forward while After Journalization takes place on other files.

If the current journal file needed by ROLLFWD is being used, the file is closed so that it can be accessed by ROLLFWD. A new file belonging to the list of files available for the After Journal, is opened for step(s) which continue to journalize.

If a journalizing step begins executing while ROLLFWD is running, the After Journal switches to the next file in the list only if the current journal file is needed by ROLLFWD.

ROLLFWD steps will be queued while waiting to access the journal files they need.

The utilities ROLLFWD and DUMPJRNL can run simultaneously on journal files and like all jobs they will be queued while awaiting access.

Files can be rolled forward while the MNJAS TRANSFER_PRIMARY command is executing.

6.5 Results of ROLLFWD

6.5.1 Results of ROLLFWD for the Output Files

Results of ROLLFWD on each file and the subsequent action to be taken, are as follows:

- Recovery is successful: the file is set to the last consistent state corresponding to the time specified in ENDDATE by ROLLFWD. Normal operations can proceed.
- Recovery is unsuccessful due to abnormal event(s) occurring ***before*** ROLLFWD opened the file: since the file is still in its restored state, resubmit ROLLFWD on the file.
- Recovery is unsuccessful due to abnormal event(s) occurring ***after*** ROLLFWD opened the file: subsequent action depends on the file access method:
  - for UFAS, restore the file first, then resubmit ROLLFWD
  - for IDS/II, resubmit ROLLFWD without restoring the area.

The following message appears in the SYSOUT when ROLLFWD opens the file:

```
**************************************************************
    OPEN IS REQUIRED FOR
    EFN = external_outfile_name CAT = catalog_name
**************************************************************
```

For each file concerned, whatever the outcome of ROLLFWD:

- a summary report is written to the JOR,
- and, a detailed report is written to the SYSOUT.

Where recovery is unsuccessful, a report of abnormal event(s):

- is written to the JOR and the SYSOUT,
- and is recorded in the SYS.ERLOG file.

PRLOG utility is run to print out the contents of SYS.ERLOG for debugging purposes.

6.5.2 Results of ROLLFWD for the Step

If the recovery is successful for *all* the files:

- the step is declared completed,
- and *rollforward* is declared successful.

If the recovery is successful for *some* files and not for others:

- the step is declared completed with severity level 2,
- and *rollforward* is declared partially successful.

If the recovery is *unsuccessful* for all the files:

- the step is declared aborted with severity level 3,
- and *rollforward* is declared unsuccessful.

6.6 Error Conditions

6.6.1 Errors in Recovery

An error in recovery occurs where an After Image cannot be applied to the record affected.

Recovery on a file proceeds for up to five consecutive errors after which the file is abandoned. The number of consecutive errors, can be changed by using the SKIPERR option of ROLLFWD. Recovery continues on other files.

If the last save was dynamic and the BEGDATE parameter was not specified, the return codes DUPKEY and RECNFD are not considered to be errors since they are perfectly normal return codes in the Dynamic Save / Rollforward process.

6.6.2 Read Error on JAS Directory

If a read error occurs on the JAS directory during rollforward, the JAS directory can be restored from a *save* and rebuilt through the MNJAS REBUILD command. User files can then be rolled forward.

6.6.3 Read Error on Primary After Journal File

If a read error occurs on a primary journal file, rollforward of user files cannot complete. However, rollforward can be performed on these files by specifying the beginning and end dates to skip the images in the journal file that cannot be read.

6.6.4 Read Error on Secondary After Journal File

If a READ error occurs on a secondary journal file defined with *secondary dual copy:*

- ROLLFWD switches from the bad copy to the other without operator intervention if the duplicate copy is mounted,
- and continues to roll forward the user files.

NOTE:
In the case of a second I/O error on the secondary dual copy or when only a single copy is used, ROLLFWD aborts.

6.7 Abnormal End of ROLLFWD

The final result depends on the conditions under which ROLLFWD once submitted, terminates abnormally such as:

- ROLLFWD aborting on a fatal severity,
- a system crash occurring while ROLLFWD is running,
- or anomalies preventing ROLLFWD from continuing.

6.7.1 Available Final Result

If the final result is available, each file processed by ROLLFWD is listed with:

- a summary report written to the JOR,
- and, a detailed report written to the SYSOUT.

A report of abnormal event(s) causing the abnormal termination of ROLLFWD:

- is written to the JOR and the SYSOUT,
- and is recorded in the SYS.ERLOG file.

PRLOG utility is run to print out the contents of SYS.ERLOG for debugging purposes.

6.7.2 Unavailable Final Result

If the final result is unavailable, subsequent action depends on the file access method:

- for UFAS, restore the file first, then resubmit ROLLFWD with the same BEGDATE as before,
- for IDS/II, *restoring the area is not necessary*, instead resubmit ROLLFWD specifying as BEGDATE:
  - either the last date written periodically by ROLLFWD to the SYSOUT when the areas were recovered successfully,
  - or the same BEGDATE as before, if no date is written in the SYSOUT.

The following message appears in the SYSOUT when ROLLFWD is restarted *after abnormal termination*:

```
*************************************************************
    EFN = external_outfile_name CAT = catalog_name
    WARNING: THE INPUT VERSION IS UNSTABLE AT OPEN TIME
*************************************************************
```

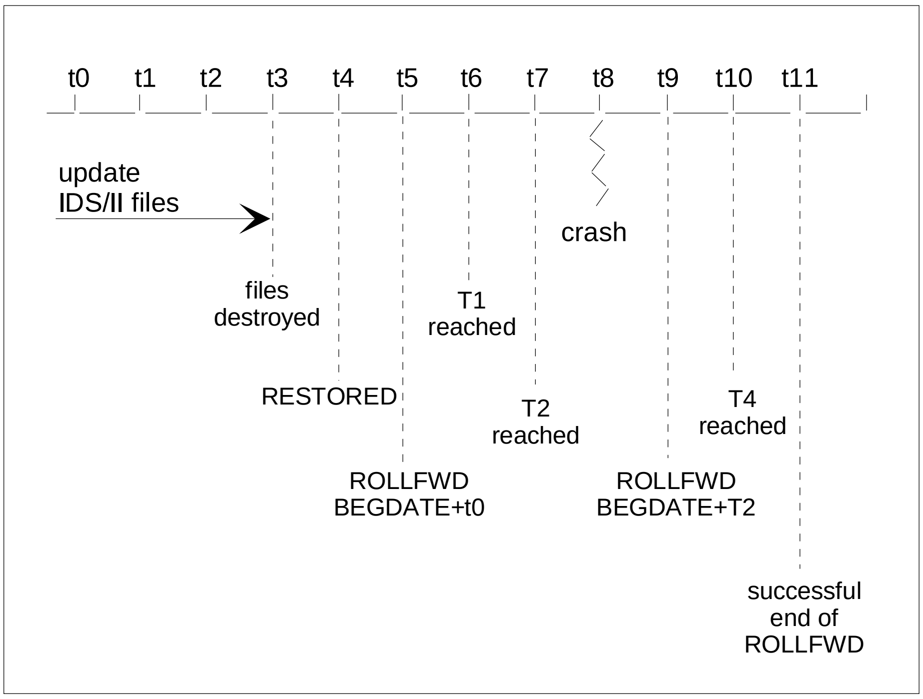


Figure 6-2. The Operation of ROLLFWD on IDS/II Files

6.8 Rollforward on Copy

It is possible to perform the rollforward of a file from a copy restored using a name different to that of the journalized file. As a result, the journalized file can continue to be journalized while the rollforward on the copy file allows file restoration to progress to a clean point.

The person using this function is responsible for supplying a correctly restored copy of a journalized file.

Function Description

This command carries out the application of journalized AFTER images by file modification: not to the file itself, but to a copy of this file. This command can be used to simultaneously process up to 12 files.

Modification of the journalized file may be in progress at the same time. In this case, it is up to the user to provide an end of application date for images. This date must correspond with a clean point on the file. The rollforward does not carry out any clean point checks.

This command only processes journalized files that are present in an auto-attachable catalog, or in SITE.CATALOG.

Specific messages are output concerning rollforward on copies.

Some checks are made on the number of copy files to be rolled forward, and on the files actually provided:

- If the number of files provided is 0 (zero), or more than 12 (twelve) the job aborts with the message "UNDEFINED LABEL".
- If the number of files provided as the first parameter is less than the number of files copied, the job is executed, but provokes the message "INCOHERENCE BETWEEN NUMBER OF FILES AND SUPPLIED FILES". The rollforward is only executed for the files correctly provided.
- If the number of files provided as the first parameter is more than the number of files copied, the job is executed, but provokes the message "INCOHERENCE BETWEEN NUMBER OF FILES AND SUPPLIED FILES". The rollforward is only executed for the files whose name has been correctly provided.

The user must give different file names for the copy files to rollforward and the journalized files. No checks are carried out.

If a journalized file is mentioned several times for different copy files to rollforward, the images relating to this journalized file are only applied to the first of these copy files to rollforward.

Dates must be input with the year given using 2 (two) digits (not four).

All journalized files must be protected by the same JAS.

Activating the Function

In V8560, The function can be called by executing the job RFW_ONCOPY from SYS.HSLLIB by EJR, RUN or INVOKE.

EXAMPLE:

```
RUN RFW_ONCOPY, SYS.HSLLIB
VL=(number of copy files to rollforward,
   copy file to rollforward, corresponding journalized file,
   ... , copy file to rollforward, corresponding journalized file,
   [BEGDATE = yy.mm.dd/hh[.mm[.ss[.msmsms]]]],
   ENDDATE = yy.mm.dd/hh[.mm[.ss[.msmsms]]],
   [DUMP = {NO|DATA}]
   [SKIPERR= {5|dec5}]
   [SIZE = 352|dec 5]
   [POOLSIZE = 256|dec 5]
   [NBBUF = 160|dec 5]
```
❑

Parameter Description

number of copy files to rollforward	This number must be a decimal between 1 and 12. It indicates the real number of files to rollforward. The number of files given in VALUES must therefore be twice as much.

The following two parameters make up a pair, repeated up to 12 (twelve) times. At least one pair must be present.

copy file to rollforward	The images that correspond to the file whose ID follows in the VALUES list are applied to this file.
corresponding journalized file	A journalized file whose images are applied to the copy file with the preceding ID in the VALUES list. This file is always put in an auto-attachable catalog or in SITE.CATALOG.

The position of the following parameters is not important.

BEGDATE	Rollforward start date. It is recommended not to provide this value if the save dates are stored in the JAS Directory. Warning messages may be sent when this date is provided, to specify that the date does not correspond with the schedules stored by the Journal in its Directory. **Important:** The year must be entered using 2 (two) digits, and not 4 (four).
ENDDATE	**Mandatory** rollforward end date. It is strongly recommended to provide a date which corresponds to a clean point on the file. **Important:** The year must be entered using 2 (two) digits, and not 4 (four).
SKIPERR	Decimal number giving the maximum authorized number of consecutive errors without interrupting rollforward.
DUMP	Used to perform a dump if the step aborts.
SIZE, POOLSIZE, NBBUF	Correspond with indications given in a JCL SIZE order, to change the size of control memory or the number of buffers to use.

This new function works in the same way as the utility ROLLFWD (with the same messages and incidents), except that the end date is mandatory. In this rollforward's sysout, a note is made that the rollforward is performed on a copy file, whose name appears:

EXTERNAL FILE NAME:	outfile name
LIKE:	journalized file name
CATALOG:	catalog name of the journalized file

In the Step's JOR, we see the following:

RFW121 Nth File:	EFN= name of file rolled forward
LIKE:	EFN= name of journalized file ROLLFORWARD IS SUCCESSFUL

❑

7. JRU (Journal Recovery Utility and Functions)

7.1 Introduction

The function of JRU (Journal Recovery Utility) is to restore the After Journal to an approximative logical state when it has been corrupted because:

- either a TDS was unable to supply the list of aborted commitment units,
- or the JAS directory was not accessible at system restart after a crash or at takeover.

If the After Journal is corrupted, journalization is prohibited and the message JP44 *(02)* appears at the console:

```
JP44 jasname: AFTER JOURNAL MAY BE LEFT IN AN INCONSISTENT STATE
              DUE TO A SYSTEM ERROR  RC = return code
```

For SYS JAS and non-HA Private JAS, the message JP44 *(01)* appears at the console:

```
JP44 jasname: AFTER JOURNAL MAY BE LEFT IN AN INCONSISTENT STATE
              DUE TO A SYSTEM ERROR  RC = return code
              RUN THE SYSTEM AFTER JOURNAL RECOVERY UTILITY (JRU)
```

All journalizing steps abort with message JP44 appearing both at the console and in their JORs.

After Journalization can only resume after JRU is executed. See *Incidents on JAS directory* (section 2.2.5.2).

*For **SYS JAS***, the operator must activate JRU.

For private ***non-HA-type JAS***, JRU is activated at the start of the journalization session, but the operator can activate JRU to avoid aborting TDS or batch steps.

For ***HA-type JAS***, JRU is activated if necessary when the JAS itself is started.

Before the After Journal is stabilized, the utilities ROLLFWD and DUMPJRNL cannot be executed.

The JRU functionality is implemented within the load module H_RECOV. It consists of two steps, the first being activated only for SYS JAS. See *How ROLLFWD Works* (section 6.3).

After JRU has been executed:

- run RESTORE followed by ROLLFORWARD for each file which is unstable and check its contents since images which should have been applied may have been invalidated (check against listed invalidated modifications),
- save all user files using the FILSAVE utility,
- and recycle the After Journal files concerned to start journalization with a consistent After Journal.

7.2 Activating JRU

JRU must be executed when the After Journal is corrupted:

*For **SYS JAS***, JRU may only be submitted under the project SYSADMIN by the JCL command:

```
EJR JRU SYS.HSLLIB;
```

For private non-HA-type JAS, JRU is a service job which:

- may be submitted under the project SYSADMIN by the JCL command:

  ```
            EJR JRU SYS.HSLLIB VL=jasname;
  ```

- is automatically submitted at the start of a journalization session, if necessary.

*For **HA-type JAS***, JRU is a service job which is activated automatically when the JAS receives the order to switch to the *ACTIVE* state:

- either on submission of the SSRV command which switches the state of the JAS from *EMPTY* to *ACTIVE*, except in the case where user files must first be recovered before the JAS can be started (for which message JP75 appears on the console),
- or when the state of the JAS switches from *BACKUP* to *ACTIVE* in the case of takeover when JRU is activated after the RECOV step has completed.

The operator can therefore activate JRU by terminating (TSRV *jasname*) and restarting the JAS (SSRV *jasname*).

Necessary Resources

- *jasname*.JADIR, the JAS directory,
- the primary After Journal files whose contents have yet to be transferred. The volume(s) containing these files must be available since JRU may access the same file several times during its execution.

JRU Restriction

The JAS directory and After Journal files must not be used simultaneously by other jobs.

7.3 How JRU Functions

JRU is composed of two modules:

- the first module determines which After Journal files on VBO disks are to be salvaged and does so. The result of the first module is equivalent to the results of the first H_RECOV step and the FILCHECK step. See *First step of the H_RECOV load module for JRU* (section 7.3.1),
- the second module corresponds to the second H_RECOV step and performs recovery on the After Journal files. See *Second step of the H_RECOV load module for JRU* (section 7.3.3).

Only the second module of JRU is executed for Private JAS since all files are FBO.

7.3.1 First Step of the H_RECOV Load Module for JRU

The first JRU step determines which After Journal files referenced in the JAS directory are unstable and are located on VBO disks. JRU compiles a list of all such files.

Since private After Journal files are never located on VBO disks, this first step is never executed for Private JAS.

7.3.2 Correcting Errors by FILCHECK

The FILCHECK utility only applies for SYS JAS.

JRU activates the FILCHECK utility on the files previously found to be unstable and located on VBO disks. When FILCHECK discovers inconsistencies in a file, it outputs a detailed report listing the errors detected and corrected, in the file and in its label status. If FILCHECK is unable to correct an error detected, it suggests actions which the user should take.

For further information about FILCHECK, refer to the *Data Management Utilities User's Guide*.

7.3.3 Second Step of the H_RECOV Load Module for JRU

For Private JAS, only this step is executed.

7.3.3.1 Logical Salvaging on After Journal

JRU performs logical salvaging on the current After Journal file and informs the user of the action it has taken.

7.3.3.2 Aborted Batch and IOF Steps

JRU compiles a list of the Batch and IOF steps still known as active steps in the JAS directory, which must be now considered aborted since they have not terminated. Each step is listed with its ron (run occurrence number), its step number and its checkpoint number.

7.3.3.3 Aborted TDS Commitment Units

JRU considers aborted those TDS commitment units:

- which have generated After Images to update user files,
- and which appear in the system After Journal as not having committed.

JRU writes these commitment units to the JAS directory.

NOTE:
A commitment unit which has generated only user journal records, and which does not appear as having committed in the system After Journal, is not considered aborted by JRU.

JRU may not declare aborted certain commitment units which have not committed from the point of view of TDS. Such commitments have passed through the TDS logical commitment stage. The number of commitments in this case is extremely limited, and can never exceed the number of TDS simultaneities.

To facilitate detecting these commitments so that the user can do whatever is necessary, JRU outputs for each commitment declared aborted, the user file images and user journal records belonging to that commitment. If a TDS journal is used, these images are preceded by:

- a page identifying the TDS see *TDS Identification* (section 7.4.4),
- and a heading for each commitment, see *Aborted Commitment Units* (section 7.4.5).

The user journal records are output in the order of that they are journalized. The user identifier, and the date and time of journalization is given for each record. The data is taken from the user journal header as defined in the *TDS Administrator's Guide*. The outputs are shown in hexadecimal on the left and in alphanumeric format on the right.

The user file images are output following the user journal records, also in the same order that these are journalized. At each change of file, the file identifier is repeated. The structure of the lines output is shown in Paragraph 6.4.5 with the following explanation:

- type of operation such as PUT/WRITE, UPDATE and DELETE,
- the record address *not applicable to UFAS relative files*,
- the image of the updated record in hexadecimal format and in alphanumeric format.

7.3.4 Final Report

JRU prints out the result of its actions for each TDS at the end of the report. If JRU has terminated correctly, the user must reset the user files updated by TDS to their last logical state by:

- running RESTORE followed by ROLLFORWARD for the unstable files,
- then checking their contents manually according to the listed invalidated modifications.

When this is done,

- save all user files using the FILSAVE utility,
- and recycle the After Journal files concerned to start journalization with a consistent After Journal.

7.3.5 Non-Recoverable Errors

If JRU detects a non-recoverable error, the JAS directory remains in an inconsistent state. Journalization however, will be allowed. HA-type JAS will switch to ACTIVE state. A warning message is sent to the JOR and in the Sysout of JRU:

```
RY350  WARNING THE JOURNALIZATION IS AUTHORISED INSTEAD UNSTABLE.
```

Message RY350 always means that future rollforwards are not guaranteed. The operator should always check in the SYSOUT of the JRU that recovery has been successful for whatever JAS JRU was activated, and is advised to:

- ensure that no journalization takes place on this JAS since the journal files are not in a consistent state,
- run FILREST followed by ROLLFWD for each file which is unstable and check its contents manually since images which should have been applied may have been invalidated,
- save all user files through the FILSAVE utility,
- and RECYCLE the After Journal files concerned to start journalization with a consistent After Journal.

The MNJAS REBUILD command can be used to reconstruct the Directory if the JRU function fails to do so. The REBUILD function operates on a *save* of the JAS directory or on the file itself.

7.4 Output from JRU

In the course of its operation, JRU produces several types of report. Examples of each type of report output by JRU are treated.

7.4.1 List of After Journal Files Processed by FILCHECK

```
**************************************************************
*             CHECK AND RESTORE LABELS FOR                   *
*             SYSTEM AFTER JOURNAL FILES                     *
*                EXECUTE FILCHECK                            *
**************************************************************

REPORTING TIME: 10:53:14 MAR 28, 1990

**************************************************************
*                                                            *
*                FILCHECK LAUNCHED FOR FILES                 *
*                                                            *
**************************************************************

SYS.JA.J1.FIRE
```

7.4.2 Logical Salvaging on After Journal

```
**************************************************************
*                LOGICAL SALVAGING OF                        *
*             SYSTEM AFTER JOURNAL FILES                     *
**************************************************************

REPORTING TIME: 10:53:34 MAR 28, 1986

**************************************************************
*             LIST OF CHECKED AFTER JOURNAL                  *
*                       FILES                                *
**************************************************************

jasname.JA.J1FIRE
RY227 - LAST ADDRESS HAS BEEN UPDATED
RY245 - CURRENT LAST ADDRESS IN DIRECTORY HAS BEEN UPDATED
```

7.4.3 Aborted Steps

```
**************************************************************
*               LIST OF BATCH/IOF STEPS                      *
*                  DECLARED ABORTED                          *
**************************************************************

           REPORTING TIME: 10:53:48 MAR 28, 1986
RY255 - IDENTIFICATION STEP - RON: X1316 STEPN = 1 CKPT = 1
```

7.4.4 TDS Identification

```
**************************************************************
*                                                            *
*                TDS IDENTIFICATION: ACT                     *
*                                                            *
**************************************************************
```

7.4.5 Aborted Commitment Units

```
J-DATE: 28/03/86   J-USID: BIB017   J-TXMU: 139   J-TPMU: 1

J-TIME: 10:28:18   J-TYPE: 01

J-HEADER

    0 0000  F0F10000 00880001 C2C9C2D6 F1F74040 C3023F3A BEC1C3E3
            40F2F8F0 F3F8F600 01 @ @@IB017 C@@@@ACT 280386

   32 0020  52C160E6 09C94040 40004B                     @A-WRI @

DATA

    0 0000  C160E6D9 C940F0F1 F2F74040 40404040 40404040 40404040
            40404040 40404040 A-WRI 0127

   32 0020  40404040 40404040 40404040 40404040 40404040 40404040
            40404040 40404040

   64 0040  40404040 40404040

J-TIME: 10:28:18   J-TYPE: 09

J-HEADER

    0 0000  F0F90000 00880001 C2C9C2D6 F1F74040 C3023F45 10C1C3E3
            40F2F8F0 F3F8F600 09 @ @@IB017 C@@@@ACT 280386

   32 0020  020012                                         @ @

EFW: FICHADK

OPERATION TYPE (PUT/WRITE)

RECORD KEY

    0 0000  F0F1F2F7 40F04000 00000000 00000000 00000000 00000000
            00000000 00000000 0127.0

   32 0020  00000000 00000000 00000000 00000000 00000000 00000000
            00000000 00000000

   64 0040  00000000 00000000 00000000 00000000
```

7.4.6 Final Report

```
***************************************************************
*  THE DIRECTORY jasname.JADIR IS RECOVERY FOR THE TDS: ACT  *
*       TO RECOVER THE STATE OF YOUR FILES YOU MUST DO       *
*        - RESTORE FILES THAT ARE UPDATED BY THE TDS         *
*        - EXECUTE ROLLFORWARD TO REACH THE LAST STATE       *
***************************************************************
```

❑

8. Transactional Context Recovery Facility (TCRF)

8.1 Introduction to TCRF

TCRF (Transactional Context Recovery Facility) operates in a non-HA coupled system environment in order to enable TDS applications running on one of the two systems to be restarted on the backup system if the first system has failed. It has three main objectives:

- To provide information about the failed system which includes:
  - a list of all TDS and batch/IOF files currently in use,
  - a list of all the steps being executed which access these files.
- To ensure the recovery of the files as far as possible.
- To enable restarting the TDS applications and reconnecting the users. Journalization will be then done in the SYS.JRNAL file of the backup system and in the After Journal files of the failed system (including the SYS.JADIR file).

TCRF can be used solely to perform the recovery of the files journalized in a system that has failed. This is particularly useful when the failed system has to undergo a COLD or CLEAN restart which prevents immediate recovery (rollback and dynamic rollforward) from being performed.

TCRF can be run in the following system configurations:

- a *single* DPS 7000 system on which two separate sets of system volumes are available, recovery occurring between one set of system volumes and the other. This use of TCRF enables recovering the user files which a mandatory COLD or CLEAN restart would have left unstable,
- a *coupled* DPS 7000 system, recovery occurring on the available backup system if the other system fails.

TCRF has to be activated by the SYSADMIN project.

TCRF is not supported if the backup system being in private mode (supporting non-HA Private JAS), the private JAS are created.

8.1.1 General Description

The general principle of TCRF consists of:

- performing the recovery of the files using the system files and SYS JAS files (including the SYS.JRNAL) of the failed system,
- enabling the execution of the TDS applications with journalization in the SYS JAS of the failed system (excluding SYS.JRNAL); the SYS.CATALOG of the backup system is modified to ensure that.

Returning to the failed system once recovered requires the following precautionary measures:

- on the backup system, terminate all steps journalizing in the SYS JAS of the failed system,
- modify the SYS.CATALOG of the backup system so that it will no longer catalog the SYS JAS files of the failed system,
- then restart the failed system.

8.1.2 Recovering Journalized Files

Integrity is ensured for all files protected by SYS JAS, in the following way:

- The JAS Directory SYS.JADIR is updated with aborted FRUs. This sets the After Journal to a stable state, and allows files protected by the After Journal to be statically or dynamically rolled forward.
- Files protected by the Before Journal are rolled back to their last stable state.
- Files protected by the After Journal and accessed in deferred update mode are rolled forward.

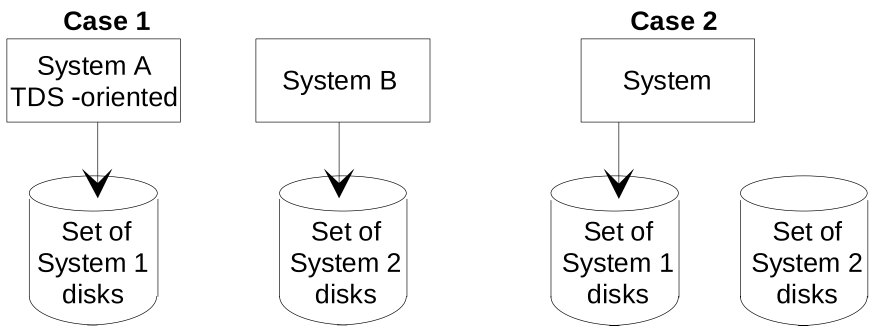
8.2 TCRF Components

The TCRF job is made up of the following components:

- A sequence of JCL called TCRFAB, which is stored in the system source language library SYS.HSLLIB.
- Two utilities, H_DUALAB and H_DUAL1AB, stored in the system load module library, SYS.HLMLIB, which control the file recovery operations.
- A set of general purpose utilities.

The following diagram illustrates how TCRF works.

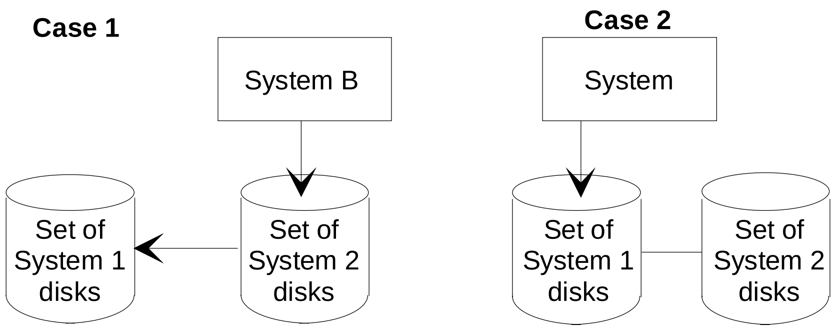
System(s) in Normal operation



Case 1: System A and B are coupled. System A is transaction-oriented, while System B cannot support an application journalizing in the After Journal at the time of TCRF. Each system functions with its own set of system disks.

Case 2: A monosystem runs TDS and Batch applications with a set of System 1 disks while another set of System 2 disks is present but not operational.

System(s) when Failure Occurs



Case 1: System A crashes. If System B runs jobs journalizing in the After Journal, these must be terminated before TCRF is activated. TCRF ensures recovery of the journalized files of all steps executing on System A before the crash. TDS sessions can then be restarted manually on System B.

Case 2: The monosystem crashes. It is reinitialized from the set of System 2 disks. TCRF ensures the recovery of the journalized files of all the steps before the crash. TDS sessions can then be manually restarted.

Figure 8-1. Purpose of TCRF

8.3 Preparations for Running TCRF

8.3.1 Introduction

The basic requirement of TCRF is that processing on the backup system is not interrupted. Procedures must be established for taking over the resources on the backup system which are required to run the TDS applications. Resources include memory, disk volumes and terminals.

Restarting TDS applications requires knowing which peripherals to transfer, and which jobs are likely to be in the HOLD state when the backup system is overloaded. Installations must be able to switch disk volumes between the two systems. Telecommunications lines must be able to switch over to the other system.

All system and user files on the main system must be available to the backup system. This is not a problem for user files when private catalogs or the SITE.CATALOG are shared, but precautions should be taken in the case of system files.

If the main system uses an After Journal, and if any TDS applications which use the After Journal are to be transferred to the backup system, the backup system must not have an After Journal which is active at the time of transfer.

8.3.2 Hardware Considerations

The volumes of the failed system, which hold the system files, the TDS files and the user files, must be accessible to the backup system while TCRF is running. The drives on which these volumes are mounted must either be declared shared at system initialization between the two systems or manually switchable. In the latter case, the operator must issue the command: `RDV MSnn [SHARED]`.

Since, after a system failure, the controller supporting the system volume is always in the HOLD state, the operator must reload the failed system to release the controller before running TCRF. If this is not possible, the operator must use the command RDV FORCE for every inaccessible volume.

8.3.3 Software Considerations

8.3.3.1 System Files on the Failed System

The following system files of the failed system must be accessible from the backup system while TCRF is being run:

SYS.SYSTEM	Must be cataloged in the SYS.CATALOG.
SYS.PVMF(i)	This set of files must be present in all cases and cataloged in the SYS.CATALOG.
SYS.CATALOG	Must be present.
SYS.JRNAL	Must be present and cataloged in the SYS.CATALOG.
SYS.KNODET	Must be present and cataloged in the SYS.CATALOG.
SYS.JADIR	Must be cataloged in the SYS.CATALOG.
SYS.JA.Ji	After Journal files may be used by the static and dynamic rollforward, or as current files for journalization when applications restart. They must be cataloged in the SYS.CATALOG.

8.3.3.2 Resident Files on the Failed System

The files used by TDS, including SWAP and DEBUG files, should not be declared resident at TP7PREP. It is good practice to catalog them. If they are declared resident, however, they must reside on shared volumes; otherwise they will not be accessible from the backup system when the TDS restarts.

8.3.3.3 System Files on the Backup System

The following system files of the backup system must be present while TCRF is being run:

SYS.SYSTEM	In all cases.
SYS.CATALOG	Must be present.
SYS.JRNAL SYS.JADIR SYS.JA.Ji	} Must not be located on the same media as the corresponding files of of the failed system.

8.3.3.4 SYS.JRNAL Block Size

From GCOS 7-V7 TS7458, the system administrator can choose the block size of the Before Journal files so that it is best adapted to the disk configuration. As a result, when TCRF is to be used, you must make sure that the Before Journal block size on the backup system is the same as that on the failed system.

8.3.3.5 TDS Cataloged Files

The three main TDS files, *tds-name*.CTLN, *tds-name*.CTLM, and *tds-name*.RECOV, **must** be accessible to the backup system before TCRF can execute. These files must either be cataloged in a private automatically attachable catalog *which is recommended*, or be declared as resident files *which is not advised*.

In the following cases, these files are already accessible to the backup system:

- The files are cataloged files, and there is one SITE.CATALOG file shared between the two coupled systems.
- The files are uncataloged resident files, on a shared volume.

In all other cases, these files must be explicitly made available. This is done under the SYSADMIN project as follows:

For cataloged files when there are separate SITE.CATALOG files, the catalog containing the entries for these files must be attached to the SITE.CATALOG of the backup system. This can be done in one of two ways:

- Either execute the following sequence of JCL (or corresponding GCL commands) on the backup system:

```
CATALOG X, TYPE = DIR;

CATALOG X.CATALOG, TYPE = FILE, SHARE = FREE;

CATMODIF X.CATALOG, DVC = dvc, MD = media;

CATMODIF X.CATALOG, AUTOATT;
```

- where X.CATALOG is the name of the private catalog, and *dvc* and *media* are the device class and media name respectively of the disk volume which holds this catalog. This is the easiest solution.
- Or run TP7PREP on the backup system, using the same parameters as when it was run on the failed system, with the keyword DEAL=N.

For uncataloged resident files which are not held on a shared volume, the backup system must be interrupted and restarted, with the volume declared as resident.

NOTE:
If each system has its own SITE.CATALOG, the lists of catalog objects such as user-names, projects, billings, passwords and access rights must be consistent for both systems.

8.3.3.6 System Date

The term *date* means date and time.

The date of the failed system and the backup system must be consistent. When the After Journal is transferred, the date of the backup system must be *later* than that of the failed system. Otherwise, there is a risk of predating. Refer to "*Setting the System Date and Time*" in Chapter 2.

8.3.3.7 Transaction-Oriented Systems

If either of the two options TDS or ALL is specified in the call to TCRF, there must not be any TDS active on the backup system with the same name as a TDS on the failed system.

8.3.3.8 Communications Considerations

The NETGENs (Network Generation) of the two systems and the SYSGENs (System Generation) of their respective frontend processors, must be compatible if both systems are to be mutual backups for each other in resuming communications with all correspondents, namely:

- the remote systems declared by one system must also be declared with identical values by the coupled system,
- all terminals declared for the frontend processor of one system must also be declared with identical values for the frontend processor of the coupled system.

The frontend processor can be either the Datanet, the CNP7 or mainway. Refer to *Networks: Overview and Generation Manual*.

8.3.3.9 Buffer Space and Memory Allocation

The backup system need not have the same amount of memory as the active system. It is possible to alter the JCL of the system running the TDS to modify the amount of memory required by TDS through the SIZE parameter.

NOTE:

It is normal to restart only some TDS for performance reasons. For example, a user with four TDS may restart only two of them, the other two waiting until the failed system has been repaired.

8.4 Running TCRF

This paragraph describes how TCRF is activated, the input parameters and the output report.

Refer to *How to Use TCRF* (section 8.5) and *How to Return to the Original System* (section 8.6) ***before*** running TCRF to be able to take the necessary precautions and decisions concerning the restart of journalization on the Backup system and the return to the repaired system. If the step-by-step sequence indicated in these paragraphs is not observed, the integrity of user files and Journal files may not be assured.

8.4.1 Activation

The JCL required to run TCRF is called TCRFAB and is located in SYS.HSLLIB.

TCRF is normally run from a terminal using the GCL command ENTER_JOB_REQUEST, but it can also be run in batch, as a job, using the JCL statement RUN.

TCRF can only be run by a registered user of the SYSADMIN project.

TCRF is interactive in dialog and response from the submitter. The submitter of TCRF must therefore remain logged until TCRF completes.

8.4.2 User Input to TCRF

TCRF begins by asking three questions, in the following order:

```
INTERRUPTED SYS.CATALOG DISK DEVICE CLASS?
ANSWER, MS/D500, MS/B10...
```

Give the device class of the disk containing the System Catalog of the failed system. This device class may be MS/B10, MS/D500 or MS/FSA according to the site configuration.

```
INTERRUPTED SYS.CATALOG VOLUME SERIAL NUMBER?
```

Give the name of the disk on which the System Catalog of the failed system is located.

```
REQUESTED BACK UP?
TYPE ONE OF THE THREE VALUES: INFO, TDS, ALL.
ANSWERS = ('INFO', 'TDS', 'ALL');
```

Key in the apppropriate answer for how TDS is to restart. The options are explained below in *TDS - ALL* (section 8.4.3.2).

NOTE:

The option **TDS** originally restricted recovery to TDS. It now has the same functionalities as the option **ALL** so that it will always guarantee the integrity of files even when they are shared between Batch and TDS.

8.4.3 Backup Options

8.4.3.1 INFO

The INFO option provides the following:

- lists the activity of the system when it failed, namely:
  - the user files that were being used,
  - and the steps that were executing and accessing those files.
- unlocks the files on the shared volumes that were locked by the failed system to enable the TDS(s) to access these files from the backup system,
- and salvages monovolume VBO BFAS sequential files by the FILCHECK utility.

List of Files That Were Being Used

```
INTERRUPTED SITE SYS.CATALOG DISK: EXPV3S .USER FILES

[FILE NAME          * FILEST * JRNL  * ACCESS  * DVC * V.S.N.]

[FILE.CATALOG       * CAT    *       *         * MS  * PCD550]

[ACT.CTLN           * CAT    *       * WRITE   * MS  * PCD550]

[TSU-SCHEMA-LIB     * UNCAT  *       *         * MS  * PCD550]

[TCRF.CATALOG       * CAT    *       * WRITE   * MS  * PCD550]

[TSUAREA1           * UNCAT  *       *         * MS  * PCD550]

[FICHADK            * UNCAT  *       * WRITE   * MS  * PCD550]

[ACT.SWAP           * CAT    *       * WRITE   * MS  * PCD550]

[ACT.RECOV          * CAT    *       * WRITE   $ MS  * PCD550]

[;2000111..SYSJRNL  * TEMP   *       *         * MS  * PCD550]

[ACT.CTLM           * CAT    * BEFORE* WRITE   * MS  * PCD550]
```

NOTES:

1. FILEST (file status) can be TEMP, UNCAT or CAT.
2. JRNL is the option in the catalog for file with BOTH, BEFORE or AFTER.
3. ACCESS is WRITE for file opened in APPEND, UPDATE or OUTPUT. No processing mode means the file was either being read or skipped.

List of Steps That Were Currently Executing

```
STATE OF CURRENT STEPS ON INTERRUPTED SITE.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

JOB IOF       USER=SYSADMIN  PROJECT=SYSADMIN  BILLING=SYSADMIN

        ->STEP H_LIBMAINT                   WAS IN EXECUTION

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

JOB ACTAFTA   USER=SYSADMIN  PROJECT=SYSADMIN  BILLING=SYSADMIN

        ->STEP ACT                          WAS IN EXECUTION

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

JOB ACTASTOR  USER=OPERATOR  PROJECT=SYSADMIN  BILLING=INSTALL

        ->STEP BIBI                         WAS IN EXECUTION
```

8.4.3.2 TDS - ALL

This option recreates the environment for the warm restart of the TDS(s).

Depending on the level of file integrity required by the active TDS, files protected by the Before Journal are rolled back. Files protected by the After Journal and accessed in Deferred Update mode are rolled forward.

A File Recovery Report is printed showing how each file has been rolled forward or rolled back.

When all the files have been updated, for each TDS application, the following message appears on the console from which TCRF was activated:

```
BU37 tds-name TDS SUBSYSTEM IS READY FOR REACTIVATION
              (AFTER JOB COMPLETION)
```

The MNJAS command LIST is then executed. Examine the output of this command if the following message appears at the submitter's terminal:

```
   BACK UP WARNING: ANOMALIES HAVE BEEN DETECTED IN SYS.JADIR
                    OF FAILED SITE.
                    FUTURE ROLLFORWARD NOT GUARANTEED.
                    PLEASE CHECK REPORT OF STEP H_JAGEN.
```

If the recovery has succeeded,

- the failed system SYS JAS directory references replace those of the backup system in the SYS.CATALOG of the backup system,
- and the After Journal files of the failed system SYS JAS are cataloged in the SYS.CATALOG of the backup system.

This enables journalization to continue on the same After Journal files, even though on another system.

The system administrator must therefore save the references of the backup system to the SYS JAS files before executing TCRF, so as to be able to restore them to the initial state if returning to the original system.

After TCRF has completed, the TDS identified in the console messages can be restarted, provided that the corresponding sharable modules have been loaded on the backup system.

The restart of TDS is quicker if the following modules *on both systems* are available:

- its sharable modules are loaded,
- its load modules are pre-initialized.

File Recovery Report

```
---------------------------------------------------------------
.STEP MTI                  GLOBAL JOURNAL OPTION: BOTH
_______________________________________________________________

                       .TYPE : T.D.S. SUBSYSTEM

 .FILE : DIRECT2
  .MEDIA : K014
   .ROLLBACK PERFORMED. NUMBER OF ROLLED BACK BLOCKS:        0

 .FILE : SEQ2
  .MEDIA : K014
   .ROLLBACK PERFORMED. NUMBER OF ROLLED BACK BLOCKS:        0

 .FILE : ISEQ2
  .MEDIA : K014
   .ROLLBACK PERFORMED. NUMBER OF ROLLED BACK BLOCKS:      110

 .FILE : MLDSISEQ
  .MEDIA : K014
   .ROLLBACK PERFORMED. NUMBER OF ROLLED BACK BLOCKS:        0

 .FILE : DIRECT1
  .MEDIA : K014
   .ROLLBACK PERFORMED. NUMBER OF ROLLED BACK BLOCKS:        0

 .FILE : SEQ1
  .MEDIA : K014
   .ROLLBACK PERFORMED. NUMBER OF ROLLED BACK BLOCKS:        0

 .FILE : ISEQ1
  .MEDIA : K014
   .ROLLBACK PERFORMED. NUMBER OF ROLLED BACK BLOCKS:        0

 .FILE : DIRECT2
  .DYNAMIC ROLLFORWARD SUCCESSFUL. ROLLED FORWARD RECORDS=  38

 BU37 MTI TDS SUBSYSTEM IS READY FOR REACTIVATION
```

8.5 How to Use TCRF

8.5.1 Releasing the Resources of the Failed System

The INFO option prints information on the volumes of the failed system containing the following system files, in which case these volumes must be mounted and online:

- SYS.CATALOG.
- SYS.SYSTEM.
- SYS.KNODET.
- SYS.PVMF(i).

If ALL or TDS options are specified, the following resources must also be made available to the *backup* system:

- resources concerning the TDS, namely:
  - all files used by the TDS,
  - all load modules of the TDS,
  - and the sharable module library which contains the load modules
- the Before and After Journals,
- and all terminals.

If some of these files are on shared volumes, the drives must be released, as follows:

- reload the failed system,
- for each statically shared drive in the HOLD state, enter the command,
- `MDHW IN`
- on the backup system to allow this system to take exclusive control of the drive.

NOTE:

In some extremely rare cases, some drives are not released when the failed system is reloaded. The disks must be set to the OFF-LINE state to remove any protection set on them by the failed system. ***However, this should be done only if it is absolutely necessary as some disks take a long time to restart.***

8.5.2 General Operations

Perform the following after the drives have been released from the failed system:

- Switch over any directly connected terminals to the backup system.
- Load FEPS if it has not already been loaded.
- Ensure that all files accessed by the TDS and cataloged in private catalogs are accessible by the backup system.
- Run TCRF, and if necessary, hold those jobs that do not have the required resources.
- Load the TDS sharable modules into backing store, if not already been loaded.
- Restart the TDS.

8.5.3 JAS Directory Updating Operations

TCRF is responsible for updating the SYS JAS Directory with the FRUs aborted on the failed system. In order to do so, TCRF needs access to some files:

- CTLN, RECOV and SWAP of all the TDS that were running on the failed system. If one of them is not accessible, the messages BU11, BU12, BU13 and BU14 will appear to the console:
  - if the operator keys in *STOP*, TCRF terminates and the recovery will be performed when the failed system is restarted,
  - if the operator keys in *CONTINUE*, TCRF proceeds on the other TDS, but no dynamic rollforward will be executed, and the user files will remain unstable.
- the SYS JAS Directory of the failed system SYS.JADIR. If it is not accessible, the messages BU15 and BU16 will appear to the console:
  - if the operator keys in *STOP*, TCRF terminates and the recovery will be performed when the failed system is restarted,
  - if the operator keys in *CONTINUE*, TCRF goes on, but no dynamic rollforward will be executed, and the user files will remain unstable.

In the case where the access to one of these files has not been possible and where the operator has keyed in CONTINUE after the execution of TCRF, the system administrator must:

- execute JRU for SYS JAS in order to be able to restart TDS or issue the MNJAS REBUILD command,
- and perform static recovery of user files which remain unstable, with the ROLLFWD utility.

8.5.4 Rollback Operations

If the backup system does not have enough disk drives to load the files for rollback operations from the failed system, the following messages will appear at the console for each file that cannot be rolled back:

```
BU01  tcrf-ron ROLLBACK NOT POSSIBLE FOR THE UPDATES
      MADE BY JOB user-ron tds-name
BU02  ON FILE:
      EFN =efn
      MEDIA = media-name
      RC = return-code
```

followed by:

```
BU03  DO YOU WANT TO HOLD JOBS
      TO PERFORM A TOTAL ROLLBACK FOR THIS STEP?
      ANSWER YES OR NO.
      IF YES, ANSWER WHEN MEDIA MOUNTED.
```

Total Rollback

If total rollback is required:

- mount the required media using the `MDHW` command,
- key in YES.

If rollback is successful, the following message appears:

```
BU37  tds-name TDS SUBSYSTEM IS READY FOR REACTIVATION
```

A warm restart for the application can be performed.

Partial Rollback

If the required media cannot be mounted (NO has been the answer to BU03), the following message appears for allowing partial rollback for each step:

```
BU04  ron DO YOU WANT
      TO ONLY ROLLBACK THE OTHER FILES?

      ANSWER YES OR NO.

      IF YES, NON ROLLED BACK FILES INTEGRITY

      IS NOT GUARANTEED.
```

- YES performs partial rollback and unused Before Images are deleted. The following message then appears:

  ```
  BU44  tds-name TDS SUBSYSTEM PARTIALLY READY.
                 REACTIVATION QUESTIONABLE.
  ```

- The TDS can now be started on the backup system.
- NO results in none of the files being rolled back. The contents of the Before Journal are saved, and the files can be recovered when the failed system is restarted.

8.5.5 Dynamic Rollforward Operations

TCRF is responsible for rolling the user files forward. In order to do so, TCRF needs access to some files:

- the After Journal files active when the failure occurred in the other system. If one of these files is not accessible, the messages BU17, BU18 and BU19 will appear to the console. The action taken by TCRF depends on the operator response:
  - If *YES* after having freed the resource, TCRF resumes its processing,
  - If *NO*, TCRF terminates and the recovery will be performed when the failed system restarts.
- the user files accessed by the TDS applications in Deferred Update mode.
- If a drive supporting one of these files is not accessible, the messages BU20, BU21, BU22 and BU23 will appear to the console. The action taken by TCRF depends on the operator response:
  - If *YES* after having released the resource, TCRF resumes its processing,
  - If *NO*, TCRF stops the dynamic rollforward for this TDS and proceeds on the other TDS. No dynamic rollforward will be performed when the failed system restarts, but the After Journal can be used for deferred recovery.
- If a file cannot be rolled forward, the messages BU24, BU25 and BU26 will appear to the console. The action taken by TCRF depends on the operator response:
  - If *RETRY*, TCRF terminates, and the recovery will be performed when the failed system restarts,
  - If *UNSTABLE*, TCRF performs the dynamic rollforward for all the other files.
  - A deferred recovery will be necessary to recover the user file which remains unstable.

8.6 How to Return to the Original System

When the TDS applications have been transported from the failed system to the backup system, restarting the failed system is easy if the session on the backup system terminates normally. Apply the ***normal procedure***.

It is therefore strongly recommended to return to the original system, once it has been repaired, after the session on the backup system has terminated normally. If a system failure occurs on the backup system, the backup system should be restarted before returning to the original system. If it is not possible, apply the ***emergency procedure***.

To operate the procedures correctly, it is necessary to understand the following:

- the SYS.CATALOG used on the backup member was the *backup* system SYS.CATALOG,
- the SYS JAS Directory used on the backup member was the *failed* system SYS JAS Directory,
- the SYS JAS After Journal files used on the backup member were the *failed* system SYS JAS After Journal files. Some new secondary After Journal files may have been created if MNJAS TRANSFER_PRIMARY has been used on backup system,
- the SYS.JRNAL of the *failed* system has been purged of all that concerned the steps which were rolled back by TCRF,
- the TDS Before Journal files of the TDS which were transported on the *backup* system may have been located in a non-shared media of the *backup* system.

8.6.1 Normal Procedure

The following is to be done if the failed system does not restart before the backup session has normally terminated (all TDSs must have terminated).

1. Before the failed system is restarted, update its SYS.CATALOG with the new secondary Journal files that may have been created. This can be done on the backup system:

```
ATT CAT1=(SYS.CATALOG MD=failed_system_media
                      DVC=failed_system_dvc
    CAT2=(SYS.CATALOG);
CATMOVE FROM=SYS.JA INFILE=(SYS.CATALOG CAT=2
    OUTFILE=(SYS.CATALOG CAT=1) REPLACE;
```

2. Update the SYS.CATALOG of the backup system with the references of its own SYS JAS directory of the backup system.
3. Perform a restart of the failed system: a warm restart is recommended if there are batch jobs to be repeated. The date and time of the repaired system must be identical to those of the backup system.
4. Do not repeat the steps which were rolled back by TCRF on the backup system. In any case, a warm System Restart tries to perform rollback of all steps that were journalizing in the Before Journal at the time of failure. Some errors logged in SYS.ERLOG must therefore be considered as normal events.
5. A cold start of the TDS is preferable, because the TDS Before Journal file may have been located on a non-sharable media.

If the failed system can restart before the backup session has terminated normally, use an alternate SYS.JADIR on each system. See *Emergency Procedure* (section 8.6.2).

8.6.2 Emergency Procedure

If a failure which occurs on the backup system during the backup session cannot be repaired, do not restart the first failed system if its SYS.CATALOG references its own SYS JAS Directory.

If the first failed system is repaired and then restarted, the list of steps it restarts is different from that known in the SYS.JADIR and based on the backup session. No recovery can therefore be executed, and the After Journal becomes inconsistent and the user files are damaged.

To avoid this, ensure that a SYS JAS Directory is not referenced by the SYS.CATALOG of the two systems, at the same time.

The following procedure is recommended:

- when the coupled environment is installed, create an alternate SYS JAS Directory on both systems (***this alternate SYS.JADIR file, not created by MNJAS CREATE, must never be used for journalization***), using the following command :

```
MDF SYS.JADIR CLEARMD;
```

For a VBO site:

```
PREALLOC
   SYS.JADIR, FILESTAT = CAT, DVC = xxx,
   GLOBAL = (MD = volname1, SIZE = 1),
   BFAS = (LINKQD =  (BLKSIZE = 4070, RECSIZE = 200,
                      RECFORM = VB, DIRSIZE = 1)),
   EXPDATE = 365;
```

For an FBO site:

```
PREALLOC
   SYS.JADIR, UNIT = BLOCK, DVC = MS/FSA,
   GLOBAL = (MD = volname1, SIZE = 120),
   LINKQD = (TYPE = NONE, BLKSIZE = 4070, RECSIZE = 200,
             MAXSIZE = 680, DIRSIZE = 5,
             RECFORM = VB, LTRKSIZE = 1),
```

For both types of site:

```
MDF SYS.JADIR DEVCLASS=devclass2
              MD=volname2, IMPORT;
```

- * volname2 contains the SYS.JADIR created through the MNJAS CREATE utility.
- Figure 8-2 shows alternating files.

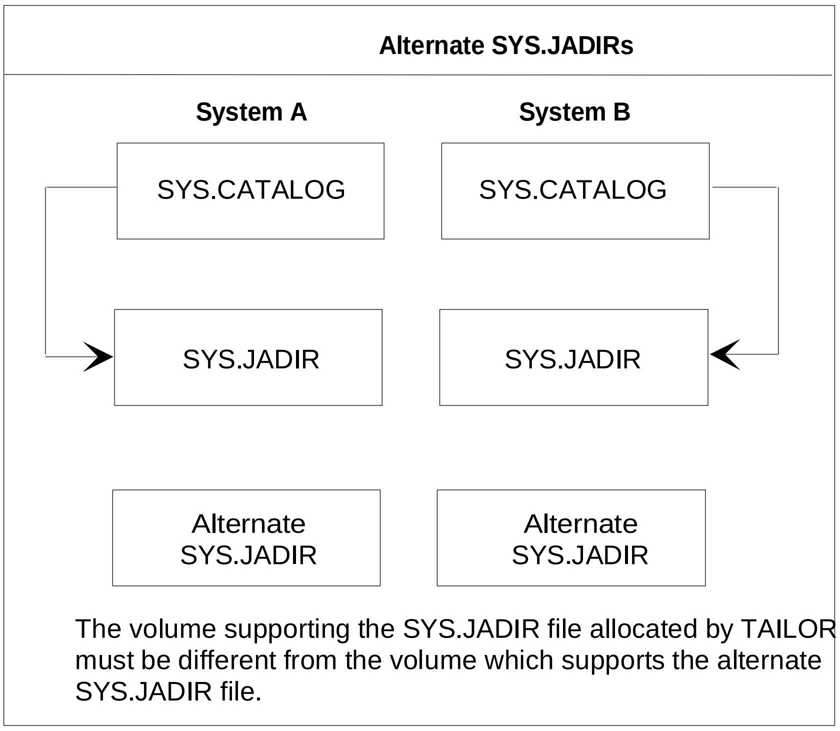

Alternate SYS.JADIRs
System A
System B
SYS.CATALOG
SYS.CATALOG
SYS.JADIR
SYS.JADIR
Alternate SYS.JADIR
Alternate SYS.JADIR
The volume supporting the SYS.JADIR file allocated by TAILOR must be different from the volume which supports the alternate SYS.JADIR file.

Figure 8-2. Alternate SYS.JADIRs

- After TCRF has been activated, run the following JCL from the backup system to catalog the alternate SYS.JADIR of the failed system in its SYS.CATALOG:

```
ATTACH SYS.CATALOG, DVC = xxx, MD = failed-system-media
CATMODIF SYS.JADIR, CAT = 1, DVC = xxx,
     MD = failed-system-alternate-sys.jadir-media
```

- Figure 8-3 shows the access between directory and catalog. System A is the failed system, and System B the backup system.

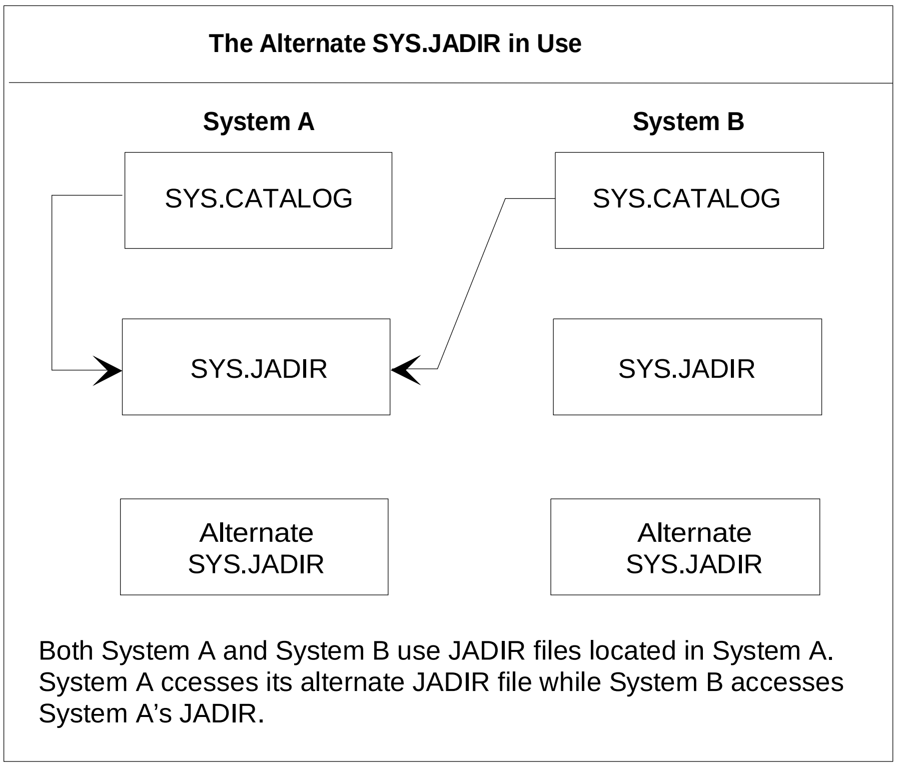

The Alternate SYS.JADIR in Use
System A
System B
SYS.CATALOG
SYS.CATALOG
SYS.JADIR
SYS.JADIR
Alternate SYS.JADIR
Alternate SYS.JADIR
Both System A and System B use JADIR files located in System A.
System A ccesses its alternate JADIR file while System B accesses System A's JADIR.

Figure 8-3. Use of Alternate SYS.JADIR

- If the application on the backup system has terminated normally, run the following JCL on the backup system:

```
ATTACH SYS.CATALOG, DVC = xxx, MD = failed-system-media
CATMODIF SYS.JADIR, CAT = 1, DVC = xxx,
          MD = failed-system-true-sys.jadir-media
```

- The original SYS.JADIR will be accessed and the alternate SYS.JADIRs will once again be backups as shown in Figure 8-2.
- Follow the Normal Procedure described in Paragraph 8.6.1.

- If the session crashes on the backup system, perform ISL and system restart on the repaired system; now the SYS JAS Directory referenced by the failed System Catalog is the alternate one.
- TCRF can then be run on the repaired System A in order to back up System B. TCRF once executed results in the following situation:

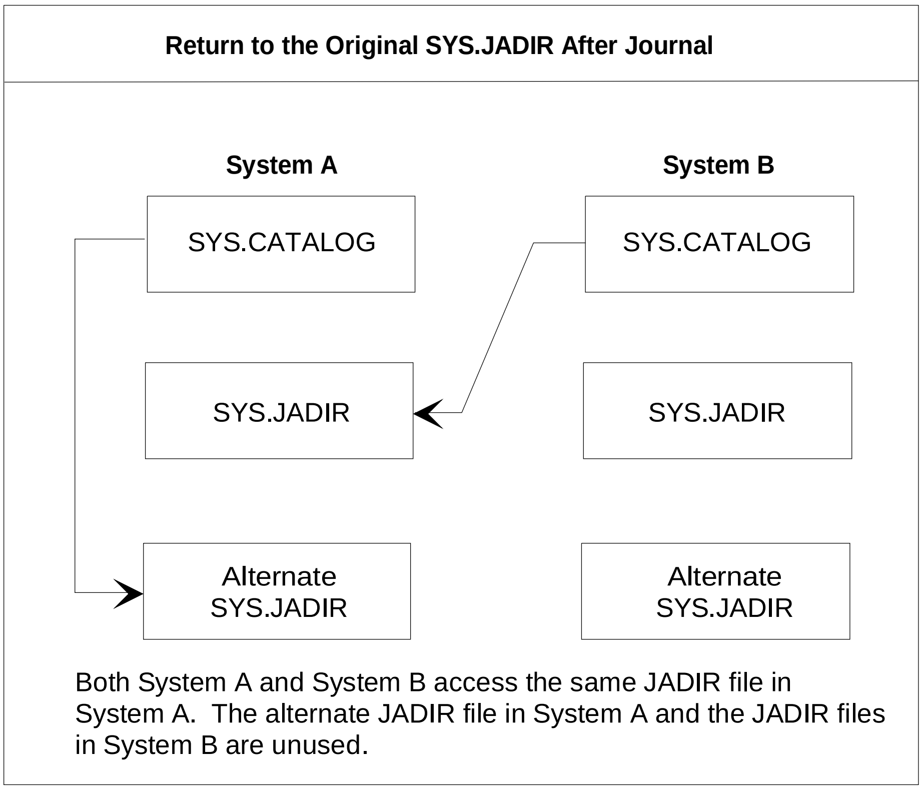

Return to the Original SYS.JADIR After Journal
System A
System B
SYS.CATALOG
SYS.CATALOG
SYS.JADIR
SYS.JADIR
Alternate SYS.JADIR
Alternate SYS.JADIR
Both System A and System B access the same JADIR file in System A. The alternate JADIR file in System A and the JADIR files in System B are unused.

Figure 8-4. Return to Original SYS.JADIR After Failure

- If new files have been added to the After Journal, the System Catalog of the repaired System A is updated. A System Restart of the repaired system A must be performed before restarting TDS to allow for journalization. All references on the backup System B to the After Journal must be removed, as in the preceding case (8.6.1).

NOTE:

Each time an alternate SYS.JADIR is referenced at System Restart, After Journalization is not possible. JAP,5 NJAFT and JP46 messages are sent to the JOR.

❑

9. DUMPJRNL Utility

While TDS is running, TDS users are able to journalize their own private data in the After Journal files of the JAS associated to the TDS.

The DUMPJRNL utility provides functions for:

- extracting this data *called the User Journal* from the After Journal,
- and writing it in APPEND mode to a sequential output file.

A new function, XDUMPJRNL, permits the extraction of the "user journal" of up to 25 TDSs at one time, with or without supplying the "start of extraction" date. This function is described below in the section "The XDUMPJRNL Command".

NOTE:

As a consequence of the introduction of this new function, for integrity reasons, DUMPJRNL and XDUMPJRNL both consider that there might be valid data for each user journal after the last extraction. This implies that the FORGET_USER_JOURNAL command of MNJAS does not work without the FORCE parameter.

9.1 User Interface

GCL and JCL syntax are almost identical. The differences are:

- STEPOPT and SIZEOPT which are JCL parameters,
- BEGDATE and ENDDATE where the year can only be expressed as two digits in the JCL syntax.

GCL Syntax

```
DUMPJRNL   TDS=name4

   OUTFILE=( sequential-output-file )

   [ OUTDEF=( define_parameters )]

   BEGDATE=[yy]yy.mm.dd/hh[.mm[.ss[.msmsms]]]

   [ ENDDATE=[yy]yy.mm.dd/hh[.mm[.ss[.msmsms]]]]

   [ DUMP={ NO | DATA }]
```

Parameter Description

TDS:	Specifies the name of the TDS for which the data records are to be extracted:
OUTFILE:	Mandatory identifier for the output file to which data extracted from the After Journal is appended. **NOTE:** The record size of the outfile must be: – *variable*, a fixed record length may lead to meaningless padding of the appended records – and *at least as large as the largest record appended*; an insufficient record length leads to a DUMPJRNL abort with LNERR (LENGTH ERRONEOUS) return code.
BEGDATE:	Mandatory starting date for DUMPJRNL of the format: [YY]YY.MM.DD/HH[.MM[.SS[.mSmSmS]]]: Year, Month, Day, Hours, Minutes, Seconds, Milliseconds are 2, or 3, or 4 digit numbers separated by a "." or a "/". "Year" can be expressed as two or four digits in the GCL syntax and as two digits only in the JCL syntax.

ENDDATE:	Optional end date for DUMPJRNL of the same format as BEGDATE. If omitted, DUMPJRNL extracts data records up to the last date known at the time the utility was activated.
DUMP:	NO (default)
STEPOPT:	Allows specifying options to control the execution of DUMPJRNL; all standard values except REPEAT are valid.
SIZEOPT:	Allows specifying size parameters as defined for the SIZE statement. See the *JCL Reference Manual*.

Constraints:

The system determines which JAS is associated to the TDS given as a parameter. If the JAS is an HA-type JAS, DUMPJRNL can only be run if the JAS is in ACTIVE state.

NOTE:

MNJAS DISPLAY_LINK command enables to check which JAS protects the TDS catalog (tdsname.CATALOG).

9.2 Resource Management

9.2.1 Configuration Requirements

The output file is a sequential file opened in APPEND mode by DUMPJRNL and assigned with the MOUNT option specified by the user. The default option is MOUNT=ALL.

Depending on the media containing the journal files, a DUMPJRNL step will require either a magnetic tape handler or a disk drive to read the journal files.

9.2.2 File Assignments

The output file is assigned by DUMPJRNL with ACCESS=WRITE.

The journal files needed by a DUMPJRNL step are assigned one-by-one in INPUT mode. DUMPJRNL also needs temporary files for journalizing data and later extraction.

9.2.3 Simultaneity of Actions Performed on a JAS

DUMPJRNL can run independently of:

- After Journalization and
- the MNJAS TRANSFER_PRIMARY command.

If the current journal file required by DUMPJRNL is being used by at least one active journalizing step, the file is closed. The After Journal resumes on a new file from the list of journal files.

Similarly, if a step activates journalization while DUMPJRNL executes, the After Journal switches to the next file in the list only if the current journal file is needed by DUMPJRNL.

ROLLFWD and DUMPJRNL utilities can run at the same time as DUMPJRNL steps, all being normally scheduled and enqueued while awaiting access to journal files required.

9.2.4 Access Rights

If the three following conditions are true:

- the access rights are installed on the site,
- the TDS is cataloged,
- the administrator has created the *tdsname*.DUMPJRNL_CRTL directory in the TDS catalog,

then the DUMPJRNL utility verifies that the project of the user has at least the READ access right on the *tdsname*.DUMPJRNL_CRTL object.

If the user does not have READ access rights, the following error message is displayed in the JOR:

```
RFW119 "ABNORMAL PROCESSING RC = ..."
```

If one of these three conditions is not true, then every user can access information in the User Journal through the DUMPJRNL utility.

9.3 Result of DUMPJRNL

On successful execution of DUMPJRNL, the user data extracted from the After Journal is available in the output file.

If abnormal events occurring **before** the output file was opened cause DUMPJRNL to abort, resubmit DUMPJRNL without restoring the initial state of the outfile.

If abnormal events occurring **after** the output file was opened cause DUMPJRNL to abort, restore the outfile to its initial state before resubmitting DUMPJRNL.

The output file is opened by DUMPJRNL only if the following message is present in SYSOUT:

```
**************************************************************
    OPEN IS REQUIRED FOR THE OUTFILE
**************************************************************
```

A detailed report of the utility is written in the SYSOUT; a report of abnormal events is written in the JOR and SYSOUT.

Abnormal events are also recorded in the SYS.ERLOG file. The PRLOG utility prints out the contents of SYS.ERLOG for debugging purposes.

9.4 DUMPJRNL and Automatic Journal File Cycling

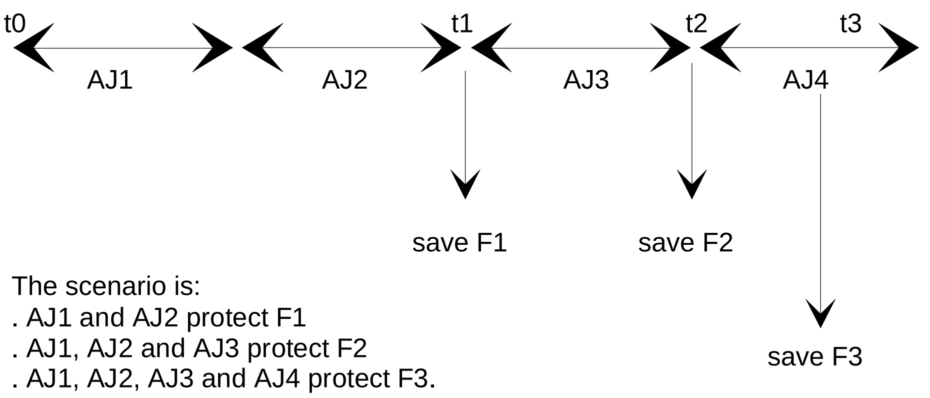
The After Journal of a JAS reuses obsolete journal files. A journal file becomes obsolete when all the user files it protects through journalization have been saved and when all the output file records have been extracted once.

Executing DUMPJRNL may therefore start recycling on all the After Journal files of the JAS, except the active one.

IMPORTANT:
If DUMPJRNL executes successfully, the TDS user data once extracted being considered obsolete by the After Journal, the media is made eligible for reuse. There is no guarantee that the same TDS user records can be extracted a second time.

F1, F2 and F3 are user files protected by the After Journal.

AJ1, AJ2, AJ3 and AJ4 are After Journal files protecting the user files.



The scenario is:
. AJ1 and AJ2 protect F1
. AJ1, AJ2 and AJ3 protect F2
. AJ1, AJ2, AJ3 and AJ4 protect F3.

At T1, even if F1 has been saved, AJ1 and AJ2 cannot be reused since they still protect F2 and F3. Similarly at T2, even if F2 has been saved, AJ1 and AJ2 still continue to protect F3. At T3, only when F3 has been saved, AJ1 and AJ2 can be reused.

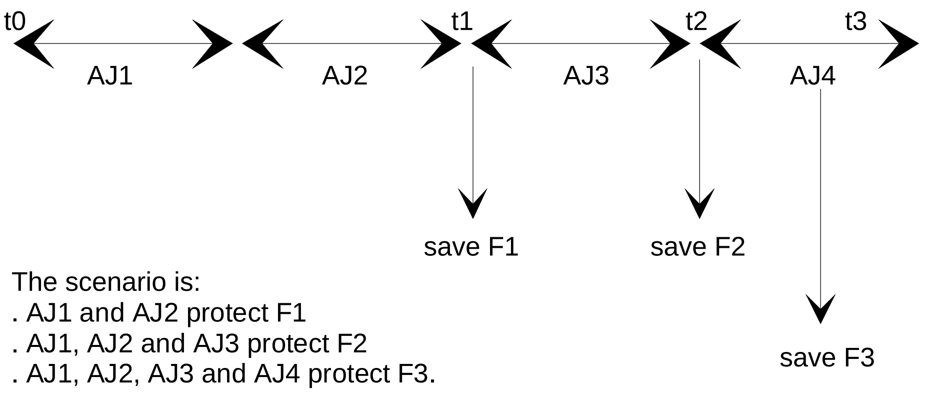
Figure 9-1. SAVE and Cycling

F1, F2 and F3 are user files protected by the After Journal.

AJ1, AJ2, AJ3 and AJ4 are After Journal files protecting the user files.

TDS TEST journalizes its own data:

```
DUMPJRNL  TDS-TEST
OUTFILE=outfile
BEGDATE=t0
ENDDATE=t3;
```



The scenario is:
. AJ1 and AJ2 protect F1
. AJ1, AJ2 and AJ3 protect F2
. AJ1, AJ2, AJ3 and AJ4 protect F3.

Data belonging to TDS TEST is collected from t0 to t3. When the user files F1, F2 and F3 have all been saved, AJ1 and AJ2 can then be reused because TDS TEST data has by then been extracted.

Figure 9-2. DUMPJRNL and Cycling

9.5 Error Conditions

9.5.1 Error in Appending a User Record

If DUMPJRNL cannot append a user record in the output file, it stops extracting the data from the After Journal and aborts. On unsuccessful execution, journal files are no longer recycled.

9.5.2 Read Error on JAS Directory

If a read error occurs on the JAS directory during the execution of DUMPJRNL, the JAS directory can be rebuilt by using the MNJAS REBUILD command.

The DUMPJRNL utility can then be run.

9.5.3 Read Error on Primary Journal Files

If a read error occurs on a Primary journal file, DUMPJRNL cannot complete.

However, the DUMPJRNL utility can be executed by specifying the beginning and end dates to skip the images of the journal file that cannot be read.

9.5.4 Read Error on Secondary Journal File

If a READ error occurs on a secondary journal file for which the secondary dual copy option was specified, DUMPJRNL:

- switches without operator intervention from the bad copy to the other provided that the duplicate copy can be mounted,
- and continues to extract user records.

9.6 Abnormal End of DUMPJRNL

The final result depends on whether DUMPJRNL aborts or a system crash occurs.

9.6.1 Final Result Available in SYS.OUT

A detailed report of DUMPJRNL is written to the SYS.OUT file. A report of abnormal events is written to both the JOR and the SYS.OUT file.

9.6.2 Unavailable Final Result

If the final result of DUMPJRNL is unavailable:

- restore the outfile to its initial state,
- and resubmit DUMPJRNL.

9.7 The XDUMPJRNL Command

The new XDUMPJRNL command makes it possible to extract the "user journal" for 1 to 25 TDSs at the same time. The user must have access rights to the directory TDSname.DUMPJRNL_CTRL for each of the TDSs.

Function Description

The user can either supply a start date (that is, a date which will apply to all the TDSs), or not supply a start date. The latter is recommended.

Extraction begins either at the start date entered in the command (that is, the same start date is applied to all the TDSs), or at the end of the extraction date of the previous extraction command (that is, an individual start date is "calculated" for each TDS).

Extraction ends either at the date given in the command or at the date of the start of execution of the XDUMPJRNL command. Note that the date of the end of extraction for the DUMPJRNL command, if not supplied in the command, is the date of the start of the command, which may be different from the date of the start of extraction, if the job containing the DUMPJRNL command is suspended before its execution.

The output files, which must exist prior to the command execution, may be treated with the processing parameters supplied in the OUTDEF parameter. These parameters apply for the processing of all the output files related to the various TDSs.

The results of the XDUMPJRNL job may be:

- valid for all the TDSs; the step terminates normally,
- valid for certain TDSs and unsatisfactory for others; the step terminates with a severity 2 code; a message is recorded in the JOR and the DUMPJRNL is partially successful (the output files contain information on some TDSs and not on others),
- invalid for all the TDSs; the step (and the job) are declared aborted with severity 3. Messages in the JOR or in the SYSOUT (depending upon the origin of the fault) indicate the reason(s).

Activating the Function

The function can be called by starting the XDUMPJRNL job from SYS.HSLLIB using EJR, RUN, or INVOKE.

EXAMPLE:

```
RUN XDUMPJRNL.SYS.HSLLIB VL= (number of TDSs,
TDS 1 name, TDS 1 output file .....,
TDS n name, TDS n output file,
[BEGDATE = yy.mm.dd/hh[mm[.ss[.msmsms]]]],
[ENDDATE = yy.mm.dd/hh[mm[.ss[.msmsms]]]],
[DUMP= {NO|DATA}]
[OUTDEF= 'DEFINE parameters common to all the output files'])
```

❑

Parameter Description

The following parameters are positional:

NUMBER OF TDSs	This is a decimal number from 1 to 25. It indicates the number of TDSs for which the "user journal" is to be extracted.

The following two parameters constitute a pair of parameters. They are repeatable (up to 25 pairs). There must be at least one pair of these parameters:

TDS NAME	Name of a TDS for which the "user journal" is to be extracted.
TDS OUTPUT FILE	This is the sequential file that is to receive the "user journal" images of the TDS whose name is given in the previous parameter.

The following parameters are not positional:

BEGDATE	This is the start date for extraction of "user journal" images of the various TDSs. If you want to start the extraction at the end date of the previous extraction for this TDS (or for the first time, at the date of the first use of the "user journal" of this TDS), **you are recommended not to supply BEGDATE**. Warning messages may be presented to indicate, when this date is supplied, that it does not correspond to the ranges recorded by the Journal in its Directory. Note that the year must be entered in 2-character form.

ENDDATE	This date corresponds to the end of the extraction of the "user journal" images. If it is not supplied, it is set at the date of the start of execution of the command. Note that the year must be entered in 2-character form.
DUMP	This indicator allows a dump to be taken if the step aborts.
OUTDEF	This string between quotes (') describes the parameters with which the output file will be handled; e.g., NBBUF. If there are several TDSs, these parameters apply to all the output files.

Messages

If there is an inconsistency in the parameters entered, the following message is sent to SYSOUT:

```
INCONSISTENCY:  INCOHERENCE BETWEEN NUMBER OF TDS AND LIST OF
TDS AND OUTPUT FILES
```

If the extraction is successful for the "user journals" of all the TDSs specified in the list, the following message is sent to the JOR:

```
RFW125: DUMPJRNL IS SUCCESSFUL
```

If the "user journal" extraction is not successful for at least one of the TDSs specified in the list, the following message is sent to the JOR:

```
RFW126: DUMPJRNL IS PARTIALLY SUCCESSFUL; REFER TO DETAILED
SYSOUT REPORT
```

If the "user journal" extraction is not successful for any of the TDSs specified in the list, the following message is sent to the JOR:

```
RFW127: DUMPJRNL IS UNSUCCESSFUL
```

If a TDS is not protected by the same JAS as the first TDS specified in the list, the following message is sent to the JOR:

```
TDS NOT PROTECTED BY THE SAME JAS AS THE FIRST TDS
```

The following messages concern the dates:

```
WARNING: BEGDATE IS ANTERIOR TO END DATE OF LAST DUMPJRNL

WARNING: BEGDATE AND ENDDATE ARE ANTERIOR TO END DATE OF LAST
       DUMPJRNL

WARNING: BEGDATE IS POSTERIOR TO END DATE OF LAST DUMPJRNL

WARNING: BEGDATE IS POSTERIOR TO DATE OF FIRST USER JOURNAL
RECORD
```

A. Insuring File Stability and Consistency

A.1 Introduction

Improved file integrity detects and reports certain system problems and file handling errors which leave files in an inconsistent state through

- the return code FLNAV which appears in the error message,
- and STATUS=UNSTABLE (ROLLFW/ROLLBACK) in LIST_FILE ALL.

FLNAV is useful in reporting an incident that has already occurred. The purpose of this Appendix is to explain how to prevent files from being inconsistent and consequently to prevent the user application from aborting.

A.2 Avoiding Incidents

The measures to prevent the loss of data involve using catalogs and the journals.

A.2.1 File Protection Level

All user files must be cataloged with the option JOURNAL=BOTH to allow for maximum protection and recovery.

Specifying After Journalization ensures that user files can be recovered if there is a failure on the disk containing them. It also means that the files can still be recovered even if Immediate Recovery, Rollback and Rollforward cannot be executed correctly. It enables TDS moreover to use Deferred Updates which ensures better performance for data access.

Defining the journalization level for a file in its catalog entry, ensures that the file will always have the same level of protection, no matter what step uses it.

Defining the journalization level in the JCL DEFINE statement or the GCL DEFI parameter group for each step using the file, cannot ensure consistency of the file. A step defined with JOURNAL=NONE executing concurrently with a step defined with JOURNAL=BOTH on the same file leaves the file with no consistent protection. If the ROLLFWD utility is used on such a file, the resulting file will be very different from what it was before the incident occurred. It will probably be totally inconsistent; the user will not be notified.

If a file is not systematically journalized as AFTER, it will be necessary to save this file after the processing where After journalization was inhibited. In this way, file recovery by rollforward is guaranteed.

A.2.2 Before Journal

Ensure that the Before Journal is of the correct size, as follows:

- Calculate the ***maximum*** number of steps that will use the Before Journal simultaneously and update the CONFIG parameter BJSIMU accordingly.
- Allocate a SYS.JRNAL file that is sufficiently large.

At the start of a TDS session, the Before Journal computes the size of the extent that it will need, from the statistics on the size it occupied in previous TDS sessions. If using Deferred Updates, specify in TP7GEN the maximum number of commitments which simultaneously use the Before Journal in order to reduce the space taken up by the TDS Before Journal file. See *TDS Before Journal Files* (section 2.2.3.3).

If a batch has such an amount of journalization that it may fill the SYS.JRNAL and force other steps to create extensions with operator questions, it is necessary to put an ASSIGN H_BJRNL clause in the JCL of this batch. This clause will direct the journal records for this step straight to its own Before Journal extension on the specified medium. See *Before Journal Extension (Batch and IOF)* (section 2.2.3.2).

The Before Journal must be systematically extended each time it becomes full. All applications must be left to terminate normally. *Do not use TERMINATE_JOB.* When the applications terminate:

- increase the size of the SYS.JRNAL file,
- stop the system,
- and restart it with Cold or Clean Restart.

A.2.3 After Journal

Primary After Journal files must be on disk. Use the MNJAS TRANSFER_PRIMARY command to transfer these files asynchronously to tapes or cartridges in order to use the minimum disk space for journalization.

Ensure that the JAS directory is of the correct size. When the JAS directory is full, system processing is unpredictable since all the files could be declared FLNAV with no recovery possible. Do not allow the JAS directory to exceed 80% occupancy. Check its fill rate regularly using the MNJAS LIST command. See *JAS directory* (section 2.2.2).

The number of After Journal files and their sizes must be defined correctly. The following warning message requires action to be taken:

```
JP13   jasname: AFTER JOURNAL IS ON MEDIUM cur_volume_name.
       CURRENT AFTER JOURNAL FILE IS efn
       WARNING : LAST MEDIUM
```

Enter the MNJAS commands MODIFY_PARAMETERS and MODIFY_MEDIA to allocate more After Journal files and more space for each one.

Issue the TRANSFER_PRIMARY command daily:

- to maintain the maximum amount of free space available on the disks,
- and to ensure that obsolete information contained in the JAS directory is deleted; where possible, specify the dual copy option for Secondary Journal files.

Make a save of the JAS Directory daily, immediately after execution of the TRANSFER_PRIMARY function. The system administrator can save the JAS directory at time during journalization by using the SHARE=DIR parameter. It is strongly advisable to make a save each time a change occurs in the JAS directory (new user file, new After Journal file, transfer, recycling, FILSAVE, etc.).

A.2.4 System Disk

In a non-HA environment, the installation must have at least one spare system disk. If there is no more space on the system disk, a copy of the disk can be saved on to tape. In some cases, the tape copy can provide the only way to restart the system without losing files.

A.3 What to do if an Incident Occurs in a Non-HA Environment

A.3.1 Step Abort

On step abort, the system recovers the files without decisions from the operator. However, operator intervention is required:

- in making available the After Journal disk or tape, and setting the drive or handler READY,
- and in keying in responses to control the functioning of TDS and the state of its journals during the recovery on abort phase.

The operator must ***never*** respond with a CANCEL_REQUEST.

A.3.2 System Crash

Cold or Clean Restart is only for startup after system shutdown.

System Crash must ***always*** be followed by a Warm Restart so that the system can recover files automatically.

As in the case of step abort, operator intervention may be required.

A.3.3 Impossible Warm Restart

If Warm Restart cannot be performed, use the spare System Disk.

Run TCRF (Transactional Context Restart Facility) on the backup system specifying the ALL option. See *Transactional Context Recovery Facility (TCRF)* (chapter 8).

TCRF will recover the files to a stable state, and will create the environment necessary to restart the TDSs.

If TCRF is unsuccessful, the files which are flagged unavailable with return code FLNAV, must be recovered through the utilities FILREST (JCL) or RESTORE_FILE (GCL) followed by ROLLFWD.

When TCRF has been executed successfully, the backup system becomes the current system.

In non-HA environments, if the two private JAS's are created on the backup system, TCRF cannot run, and produces the message BU93. To get around this:

- Save all journalized files on the BLUE and GREEN JAS's.
- Save JAS system files: JAS catalog, JAS Directory and JAS Journal files.
- Execute the MNJAS DELETE command for each of the private JAS's.
- Run TCRF to recover all journalized files on the JAS SYS and, if necessary, continue TDS processing
- At the end of these processes, execute the CREATE command for both private JAS's, then restore all system files for each JAS.

A.3.4 Cold or Clean Restart After System Crash (To Be Avoided)

Cold or Clean Restart after a system crash has serious consequences:

- Journals cannot be recovered by the system:
  - the Before Journal is completely lost,
  - the After Journal is in an inconsistent and incomplete state.
- and all the files open at the time of the crash are unstable.

To recover the files:

- first restore them from a previous save,
- then run ROLLFWD.

Updates to files by Batch and IOF steps should be checked if message JP45 appears.

A.3.5 Loss of a TDS SWAP File and/or RECOV File

When a TDS SWAP or a TDS RECOV file is lost, the warning message JP44 appears at the console for operator action as follows:

- execute JRU (Journal Recovery Utility) or issue the MNJAS REBUILD command, to rebuild a stable SYS.JADIR file, see Note,
- then, for each file used by the TDS, run FILREST (JCL) or RESTORE_FILE (GCL) followed by ROLLFWD. ROLLFWD is described in *ROLLFWD Utility* (chater 6).

NOTE:
For more information on JRU, see *JRU (Journal Recovery Utility and Functions)* (chapter 7).

A.3.6 Loss of Before Journal

Files cannot be rolled back if the SYS.JRNAL file or any of its TDS extensions are lost. FILREST (RESTORE_FILE) followed by ROLLFWD, must be run for all the files protected by the Before Journal and left unstable.

If the Before Journal is lost while the system is running, allow all the applications to terminate normally so that the files remain in a stable state. *Do not use TERMINATE_JOB*. A file will become unstable only if it requires rollback which is then unsuccessful. When the applications terminate:

- allocate a new SYS.JRNAL file,
- stop the system,
- then restart it with Cold or Clean Restart.

A.3.7 Loss of After Journal

If the After Journal is lost while the system is running, all currently executing jobs must be terminated as soon as possible:

- for batch and IOF applications, issue TERMINATE_JOB,
- for TDS, issue the Administrator command TERMINATE_TDS.

The necessity of terminating TDS is even greater where the TDS uses Deferred Updates on its files. After all the steps have terminated, the files should be stable. The After Journal can then be rebuilt.

During recovery, incidents and the corrective actions to be taken are as follows:

- If the JAS Directory was inaccessible at system restart (JP41 appears at the console), a consistent state may be recovered by using the methods described in *Incidents on JAS directory* (section 2.2.5.2). If the JAS directory is lost, the file and its entry in the JAS catalog can be restored from its last save and then rebuilt through the MNJAS REBUILD command. Unstable user files can be recovered through the FILREST and ROLLFWD utilities.
- If one of the After Journal files is lost, unstable user files can be recovered to the state they were in, depending on their last save date:
  - either before the first image of the lost After Journal file,
  - or after the last image of the lost After Journal file.

EXAMPLE:

After Journalization has occurred from Monday to Friday and Wednesday's After Journal file has been lost. The user files saved on Monday and Tuesday can be restored to the state they were in on Tuesday evening. The user files saved on Thursday or later can be restored to their current state.

When user files have been recovered as best as possible, save all the files using FILSAVE (SAVE_FILE).

❑

A.3.8 Loss of User File

Restore the file from a previous save using FILREST (RESTORE_FILE) followed by ROLLFWD to return the file to a stable state.

A.4 Recovering an Unavailable (FLNAV) File

A.4.1 Introduction

When an attempt is made to open an unavailable file, the code FLNAV is returned. The file is unavailable because it is potentially inconsistent. Several methods are available to restore such files to a stable state.

A.4.2 Restore and Static Rollforward

To recover a file to a stable state

- restore it from a previous save,
- and perform a static rollforward.

The file must have been previously saved through FILSAVE or FILDUPLI (SAVE_FILE or COPY_FILE); all the applications which use the file since the last save must be specified with JOURNAL=AFTER or JOURNAL=BOTH to invoke continuous after journalization.

Then save the file again.

A.4.3 LOAD_FILE or CREATE Utility

The GCL LOAD_FILE command or the JCL CREATE utility, can be used to rebuild a stable output UFAS file from an unstable input file *only if restore and static rollforward are impossible.*

The file is stable only as far as UFAS file organization is concerned. The records in the output file are correctly linked for UFAS to access without problems.

However, the logical consistency of the file is not guaranteed. For example, if a TDS application has updated the file twice, one set of the updates may have been written to the file but not the other. It is up to the Administrator of the file to check its logical coherence. Updates being processed at the time of failure can be validated by the Administrator, if they are known.

After using LOAD_FILE or CREATE, save the file with FILSAVE (SAVE_FILE). The internal organization of the resulting file is different from that of the previously saved file since UFAS recovers the physical space occupied by CIs (control interval) previously deleted.

A.4.4 DBUTILITY (IDS/II)

The DBUTILITY utility can be used to force the stability of an IDS/II area file through the *Patch Label Reset Transient State* command *only if restore and static rollforward are impossible.*

Once stability has been forced, IDS/II no longer guarantees the consistency of the file:

- either from the point of view of the internal organization since some links may be incorrect,
- or from the point of view of the logical consistency of records.

Although the *Patch Label Reset Transient State* command does not guarantee the consistency of area organization, DBUTILITY allows checking consistency by accessing and modifying records directly. It may be useful to rebuild part of an area manually. See the *IDS/II Administrator's Manual*.

A.4.5 Label Patching

Label patching will more than likely damage the file and should be restricted only to those files which have been opened but which have been left unused.

Although files can once again be accessed, their internal organization and logical consistency is not guaranteed.

Patching labels should not be used if Restore and Static Rollforward are possible.

A.5 Debugging Information

A STAR concerning the stability and consistency of a file should include all the following items:

- the JOR of the executing step,
- the log and trace reports taken from SYS.SWLOG,
- the PRLOG printout,
- the output from the MNJAS LIST,
- the result of FILLIST (LIST_FILE) for the file concerned,
- the dump of the JAS directory file,
- copy of the console listing,
- sysout of the MNJAS step run by the System Administrator.

❑

B. Messages from File Recovery Facilities

B.1 Before Journal and Rollback Messages

Before Journal and Rollback Messages are listed in the order of messages that appear at the console for the MAIN operator, followed by the messages printed in the JOR. The console messages also appear in the Console Message Directory for the installation. The JOR messages also appear in the *Messages and Return Codes Message Directory*.

B.1.1 JL Console Messages

```
JL00   *ron ROLLBACK NOT SUCCESSFUL
```

JL00 is sent to the submitter of the journalizing step.

```
JL01   *ron BEFORE JOURNAL FULL. CONTINUE?
```

JL01 is sent to the submitter of the journalizing step. Once logged off, the default answer is NO.

```
JL02   *ron MORE SPACE NEEDED FOR BEFORE JOURNAL. WHERE?
```

```
JL03   *ron EXTENSION ON volume-name NOT POSSIBLE
            [RC = return-code]
```

JL03 is sent to the main operator.

```
JL05   *ron BEFORE JOURNAL I/O ERROR. CONTINUE?
```

```
JL06   *ron SYSTEM WITHOUT BEFORE JOURNAL.CONTINUE?
```

JL05 and JL06 are sent to the submitter of the journalizing step. Once logged off, the default answer is NO.

```
JL07   *ron SPACE NEEDED FOR BEFORE JOURNAL FOR TDS tds-name.
            WHERE?
```

JL07 is sent to the submitter of the TDS step. Once logged off, the message is rerouted to the MAIN operator.

```
JL08   *ron SPACE ALLOCATION ON volume-name NOT POSSIBLE.
            [RC = return-code]
```

JL08 accompanies message JL07.

```
JL09   *ron BEFORE JOURNAL FOR TDS tds-name ON volume-list?
```

JL09 is sent to the *submitter* of the TDS step. Once logged off, the default answer is YES.

```
JL10   *ron INVALID REPLY: CONTINUE
            (BEFORE JOURNAL MANDATORY: STEP UNDER GAC CONTROL)

JL11   *ron INVALID REPLY: previous-reply
            (POSSIBLE ANSWERS: possible-answers)

JL12   *ron INVALID REPLY: previous-reply (INVALID DEVCLASS)

JL13   *ron INVALID REPLY: previous-reply (INVALID VOLNAME)
```

JL10, JL11, JL12 and JL13 are returned to the user keying in an invalid reply to the previous message. The invalid reply appears in the message. Possible answers in message JL11 depend upon the previous message and context:

If the previous message expected the answer:

- YES or NO, the answer to JL11 must also be YES or NO
- NO, CONTINUE or volume-name, the answer to JL11 must also be NO, CONTINUE or volume-name
- NO or volume-name [device-class], the answer to JL11 must also be NO or volume-name [device-class].

```
JL14   SYSTEM WITHOUT BEFORE JOURNAL.
       SIZE TOO SMALL. SHOULD BE AT LEAST EQUAL TO n BLOCKS
```

JL14 is sent to the MAIN operator before SYSTEM READY if the Before Journal is too small for the value declared for BJSIMU at CONFIG. The size of the SYS.JRNAL file must be increased.

```
JL15   SYSTEM WITHOUT BEFORE JOURNAL.
       IF ABNORMAL RUN PRLOG: [RC = return code]
```

JL15 is sent to the MAIN operator before SYSTEM READY. If the problem cannot be identified, run the PRLOG utility.

```
JL16   *ron INVALID REPLY:
       previous reply (INVALID VOLUME ORGANIZATION)
```

JL16 is returned to the user keying in an invalid reply to the previous message. The invalid reply appears in the message.

```
JL17   SYSTEM WITHOUT BEFORE JOURNAL
```

JL17 is sent to the MAIN operator before SYSTEM READY if the block size of the Before Journal is incorrect (diffferent from 512 or 4096). Correct the SYS.JRNAL bock size.

B.1.2 JL JOR Messages

B.1.2.1 Normal Termination

```
JL00   BEFORE JOURNAL MAXSIZE = xx FILE BLOCKS
```

xx indicates the maximum number of file blocks used to journalize between two restart points, namely:

- checkpoints,
- beginning of step and end of step in a Batch or IOF environment,
- beginning of step and end of step in a TDS environment.

This statistic allows adjusting the size of the Before Journal file:

- in a batch or IOF environment, to minimize the number of extensions,
- in a TDS environment, where attempts to extend the Before Journal on the media specified by TDS have been unsuccessful due to lack of space. See JL06.

```
JL06   BEFORE JOURNAL EXTENSIONS SUCCESSFUL=xx, UNSUCCESSFUL=yy
```

JL06 is sent only at the end of a TDS session to indicate the number of attempts to extend the Before Journal during the TDS session; 0 values indicate ***no*** attempts, hence optimum performance.

```
JL07   ROLLBACK: xx COMMITMENT UNIT(S), yy DYNAMIC
```

JL07 is sent only at the end of a TDS session. It indicates how many commitment units were rolled back, and how many dynamic rollbacks (for IDS files) were performed during the session.

```
JL08   SYSTEM WITHOUT BEFORE JOURNAL: STEP LAUNCHED BY OPERATOR
```

JL08 appears if the operator has answered YES to JL06. The step has been executed without before journalization.

JL09 appears if:
- an I/O error has occurred on the Before Journal,
- ***and*** if the operator has answered YES to the console message JL05.

File integrity is not assured since the files are unprotected until the next checkpoint has been taken. Once a checkpoint has been taken, journalization restarts and file protection resumes.

```
JL10   BEFORE JOURNAL FULL STEP CONTINUED WITHOUT BEFORE JOURNAL
```

JL10 indicates that:
- the step has reached the 15th extension of the Before Journal,
- and the operator has answered YES to the console message JL01.

File integrity is not assured since files are not protected until the next checkpoint is taken. When a checkpoint is taken:
- the space occupied by the Before Journal is released,
- journalization restarts,
- and the file protection resumes.

```
JL11   BEFORE JOURNAL CANCELLED BY { OPERATOR | USER }
       STEP LAUNCHED WITHOUT BEFORE JOURNAL
```

JL11 appears if CONTINUE has been issued for the console message JL02. File integrity is not assured and files are no longer protected.

B.1.2.2 Step Abort

```
JL00   BEFORE JOURNAL MAXSIZE = xx FILE BLOCKS
```

```
JL06   BEFORE JOURNAL EXTENSIONS SUCCESSFUL=xx, UNSUCCESSFUL=yy
```

```
JL07   ROLLBACK: xx COMMITMENT UNIT(S), yy DYNAMIC
```

See *Normal Termination* (section B.1.2.1) for explanations.

```
JL08   SYSTEM WITHOUT BEFORE JOURNAL. STEP ABORTED BY OPERATOR
```

The operator has answered NO to console message JL06. The step has not been executed because journalization is mandatory and no Before Journal is available.

```
JL09   I/O ERROR ON JOURNAL FILE.
       { BEFORE JOURNAL CANCELLED BY OPERATOR                 }
       { BEFORE JOURNAL CANCELLED BY USER                     }
       { FILES ARE IN SHARE=MONITOR                           }
       { STEP ABORTED BY SYSTEM: BEFORE JOURNAL IS MANDATORY }
```

An I/O error has occurred on the journal and the operator has answered NO to console message JL05. No further updates are made to the journalized files. The files can be rolled back.

```
JL10   BEFORE JOURNAL FULL.
       { BEFORE JOURNAL CANCELLED BY OPERATOR                }
       { BEFORE JOURNAL CANCELLED BY USER                    }
       { FILES ARE IN SHARE=MONITOR                          }
       { STEP ABORTED BY SYSTEM: BEFORE JOURNAL IS MANDATORY }
```

The step has reached the 15th extension of the Before Journal and the operator has had to answer NO to the console message JL01. The step has aborted and the files are able to be rolled back.

To solve this problem, either add checkpoints or commitments within the step, to decrease the size of the data to be rolled back in case of error, or provide sufficient space on the media for journalization to take place.

```
JL11   BEFORE JOURNAL CANCELLED BY { OPERATOR | USER }
```

The operator or the submitter has answered NO to console message JL01. The step has aborted and no further updates are made to the journalized files. The files are able to be rolled back.

B.1.2.3 Rollback Messages

```
SYSTEM   JL01: ROLLBACK CANNOT BE PERFORMED ERROR CODE = xx
```

An error has occurred due to a system malfunction. The error code indicates the type of error. This message is not followed by any other rollback messages. Console message JL00 is also sent, and the error is logged in the SYS.ERLOG file. All the files are in an unstable state.

```
JL02   ROLLBACK FAILURE ERROR CODE = xx
```

An error has occurred due to a malfunction in the rollback mechanism. Depending on the type of error, the rollback may have been stopped, or it may have been continued as far as possible. Console message JL00 is sent. All the files are in an unstable state.

```
JL03   efn: ROLLBACK NOT PERFORMED:
            { CANNOT READ FICB                        }
            { FILE HAS BEEN DEASSIGNED BEFORE ROLLBACK  }
            { I/O ERROR CODE = xx                     }
            { [NON ROLLED BACK FILE BLOCKS = yy]      }
            { ROLLBACK CANCELLED BY OPERATOR          }
            { UNABLE TO ASSIGN THIS FILE              }
            { UNABLE TO CLOSE THIS FILE               }
            { UNABLE TO FIND THIS FILE IN SHARING TABLE }
            { UNABLE TO OPEN THIS FILE                }
```

No rollback at all has been performed for this file. If this message is not preceded by message JL02, the number of blocks that have not been rolled back is given. JL00 appears and the error is logged in the SYS.ERLOG file. The file is in an unstable state and must be checked. If this message is associated with the JNBUNKN return code, the file is consistent but its state may be unstable.

```
JL04   efn: ROLLBACK SUCCESSFUL. ROLLED BACK FILE BLOCKS = xx
```

The file has been rolled back successfully, and the number of blocks rolled back is indicated.

```
JL05   efn: ROLLBACK NOT SUCCESSFUL ERROR CODE = xx
       ROLLED BACK FILE BLOCKS=yy [NON ROLLED BACK FILE BLOCKS=zz]
```

A partial rollback has been performed for this file. The error type is indicated by the error code. The number of blocks rolled back is given. If this message is not preceded by message JL02, the number of blocks that have not been rolled back is also given. The console message JL00 appears. The file is in an unstable state and must be checked.

```
JL06   efn: FILE LEFT UNSTABLE. UNABLE TO CLOSE THIS FILE
```

The file has been rolled back completely, but the labels have not been updated. The console message JL00 appears and the error is logged in the SYS.ERLOG file. The file is not accessible but it is in a consistent state.

```
JL20   ROLLBACK INITIATED BY USER
```

Meaning:

JL 20 is printed in the JOR of a batch/IOF step which has performed an immediate call to the primitive H_INVFRU allowing invalidation of the user-file update made by the program. This message precedes the messages sent by the rollback for the user files.

Action: None

B.1.2.4 System Crash

Messages In the JOR of X0001

Only rollback messages may appear in the JOR of X0001.

```
SYSTEM   JL01: ROLLBACK CANNOT BE PERFORMED ERROR CODE = xx
```

A system malfunction has occurred, and no step can be rolled back. The error is logged in the SYS.ERLOG file.

Messages in the User JOR

All messages concerning journalization are lost after a system crash.

```
SYSTEM   JL01: ROLLBACK CANNOT BE PERFORMED. ERROR CODE = xx

JL02   ROLLBACK FAILURE ERROR CODE = xx

JL03   efn: ROLLBACK NOT PERFORMED:
            { CANNOT READ FICB                            }
            { FILE HAS BEEN DEASSIGNED BEFORE ROLLBACK    }
            { I/O ERROR CODE = xx                         }
            { [NON ROLLED BACK FILE BLOCKS = yy]          }
            { ROLLBACK CANCELLED BY OPERATOR              }
            { UNABLE TO ASSIGN THIS FILE                  }
            { UNABLE TO CLOSE THIS FILE                   }
            { UNABLE TO FIND THIS FILE IN SHARING TABLE   }
            { UNABLE TO OPEN THIS FILE                    }

JL04   efn: ROLLBACK SUCCESSFUL. ROLLED BACK FILE BLOCKS = xx

JL05   efn: ROLLBACK NOT SUCCESSFUL ERROR CODE = xx
            ROLLED BACK FILE BLOCKS = yy
            [NON ROLLED BACK FILE BLOCKS = zz]

JL06   efn: FILE LEFT UNSTABLE. UNABLE TO CLOSE THIS FILE.
```

See *Rollback Messages* (section B.1.2.3) for explanations.

B.2 After Journal and ROLLFWD Messages

B.2.1 JP Messages

```
JP11   jas_name: AFTER JOURNAL IS ON MEDIUM cur_volume_name.
       CURRENT AFTER JOURNAL FILE IS efn.
```

Meaning:

JP11 appears at the console for information to notify the operator that the After Journal is being written to efn on cur_volume_name.

The message appears when:
- the After Journal File specified by efn is opened
- ***and*** there is still another available After Journal file on cur_volume_name.

Action: None

```
JP12   jas_name: AFTER JOURNAL IS ON MEDIUM cur_volume_name.
       CURRENT AFTER JOURNAL FILE IS efn.
       NEXT MEDIUM WILL BE next_volume_name.
```

Meaning:

JP12 appears at the console to notify the operator that the cur_volume_name used by the After Journal is almost full. The next_volume_name is given so that the operator can make the media available for journalization to continue on when cur_volume_name is full.

If next_volume_name is not available, the system will request the operator to make it available.

The message appears when:
- the After Journal File specified by efn is opened
- there is no available After Journal file on cur_volume_name
- the system has determined next_volume_name.

Action:

Make next_volume_name available.
No action is necessary if it is already available.

```
JP13   jas_name: AFTER JOURNAL IS ON MEDIUM cur_volume_name.
       CURRENT AFTER JOURNAL FILE IS efn.
       WARNING : LAST MEDIUM.
```

Meaning:

JP13 appears on the console to notify the operator that the current medium used by the After Journal is almost full and that there is no other medium available for future journalization. If no new After Journal file is made available:

- journalization will be interrupted when the current After Journal file has been filled,
- and all the journalizing steps at that time will be aborted.

JP13 is sent when the following conditions are combined:
- the After Journal file specified by *efn* is open,
- there is no available After Journal file on the medium specified by *cur_volume_name*,
- and there is no available medium to support an After Journal file.

The message is repeated until:
- either new After Journal files are made available,
- or journalization has ceased.

Action:

Make new After Journal files available as soon as possible.

Depending on the operational environment, the possible solutions are the following MNJAS commands:

```
1. MNJAS jasname COMMAND = 'TRANSFER' ;
```

The effect of this command is:
- to move the Primary After Journal files to available Secondary After Journal files,
- to make available the Primary After Journal files once transferred.

```
2. MNJAS jasname COMMAND = 'RECYCLE_JOURNAL_FILE parameter' ;
```

The effect of this command is to recycle the Primary After Journal files.

```
3. MNJAS jasname
   COMMAND = 'MODIFY_MEDIA ADDPRIMARY = new_media_list';
```

The effect of this command is to add new Primary After Journal media.

JP14 `jas_name: NO MORE AVAILABLE PRIMARY AFTER JOURNAL FILE.`

Meaning:

JP14 appears on the console to notify the operator that there is no other primary After Journal file available for after journalization. After journalization will be interrupted until new After Journal Files become available. All journalizing steps will abort as soon as they need the After Journal.

The message is repeated until:
- either new After Journal files are made available,
- or journalization has ceased.

Action:

Make new After Journal files available as soon as possible.

Depending on the operational environment, the possible solutions are the following MNJAS commands:

```
1. MNJAS jasname COMMAND = 'TRANSFER' ;
```

The effect of this command is:
- to move the Primary After Journal files to available Secondary After Journal files,
- to make available the Primary After Journal files once transferred.

```
2. MNJAS jasname COMMAND = 'RECYCLE_JOURNAL_FILE parameter' ;
```

The effect of this command is to recycle the Primary After Journal files.

```
3. MNJAS jasname
   COMMAND = 'MODIFY_MEDIA ADDPRIMARY = new_media_list';
```

The effect of this command is to add new Primary After Journal files.

Once After Journal files have been made available, the steps may be repeated by answering YES to the question: `ABORTED job, REPEAT ?.`

JP21 `jas_name: UNABLE TO WRITE ON AFTER JOURNAL FILE efn.`

Meaning:

JP21 appears on the console to warn the operator that an error occurred when writing on the *efn* of the primary After Journal file:
- either an I/O error has occurred while writing to the After Journal file identified by *efn*,
- or if CANCEL_REQUEST for the medium switching to standby.

The current block is written to a subsequent Primary After Journal file to ensure the integrity of journalized files.

Action: None

JP22 `jas_name: THE jas_name.JADIR DIRECTORY FILE IS ALMOST FULL.`
`ANY NEW JOURNALIZING ACTION WILL ABORT.`
`DELETE IT AND CREATE IT LARGER.`

Meaning:

The After Journal has detected that its directory is almost full and may no longer ensure file integrity. See Note.

No new FRU can start, but all those currently executing will terminate normally. Any attempt to open a user file will be rejected.

Action:

First try to purge the directory file of obsolete information through the MNJAS commands TRANSFER and FORGET_USER_FILE.

If still not successful, stop the steps using the After Journal and recreate a larger JAS Directory:

- enter the command `MNJAS jas_name COMMAND=DELETE` to delete the After Journal,
- then recreate it with a larger JADIR_SIZE, using MNJAS command CREATE.

Journalization can then resume.

The utilities ROLLFWD and DUMPJRNL cannot execute on After Images logged before this operation. So before deleting the JAS, run FILSAVE on each protected active user file and run DUMPJRNL for each active TDS.

NOTE:
If the *jasname*.JADIR file is expandable, an extent is automatically created and no JP22 message appears.

JP23 `jasname: THE jasname.JADIR DIRECTORY FILE IS ALMOST FULL.`
`STEP REJECTED.`

Meaning:

JP23 appears in the JOR of the starting step. The jasname JAS has detected that its jasname.JADIR file is almost full and may no longer ensure file integrity. No journalizing step can start before a new jasname.JADIR is created.

Action: None

JP36
```
jasname: efn
THE DATE OF THIS SAVE HAS NOT BEEN RECORDED IN THE JAS
FOR SECURITY AND PROTECTION PURPOSE.
THE PRIVATE CATALOG OF THIS FILE MUST BE ATTACHED
WITHOUT DEVCLASS, MEDIA SPECIFICATION.
```

Meaning:

JP36 is printed in the JOR of the FILSAVE utility. The file being saved is cataloged in a private catalog. Another file which has an identical name in a catalog of the same name, has already been journalized.

Since the rules for journalization and FILSAVE are not applicable, the system cannot decide if the file being saved is one already known to the jasname JAS. The save date is not recorded in the jasname After Journal, resulting in the following:

- the oldest save date known by the jasname After Journal is therefore not updated,
- and consequently, no automatic cycling is performed for the journal files.

Action**:** None

JP37
```
jasname: efn
SAVE DATE RECORDED IN THE JAS IS date
```

Meaning:

JP37 is printed in the JOR of the FILSAVE utility, except if SILENT=1 the user asks if there should not be a message. The specified file journalized in AFTER mode, has been successfully saved. The date given to the nearest millisecond has the format: `HH.MM.SS.mSmSmS Month Day Year`.

This date may be used as input to the ROLLFWD utility. If option AUTOCYCLE is not used, the last save date will be taken into account if no date is given in input.

If the AUTOCYCLE option is used in the parameters defining the jasname JAS, journal files are automatically recycled at the end of the save when the saved file was the oldest known by the JAS.

Action: None

JP38
```
jasname: efn
ERROR IN SYSTEM DATE. SAVE DATE CANNOT BE RECORDED.
```

Meaning:

JP38 is printed in the JOR of the FILSAVE utility. The date on which the file was saved is earlier than the last jasname After Journal date. The file save is aborted.

Action: None

JP41
```
JOURNALIZATION IS IMPOSSIBLE DURING THIS GCOS SESSION.
(01) SYSTEM UNABLE TO ACCESS THE SYS.JADIR FILE; RUN PRLOG.
     [return code]
(02) ERROR IN JOURNAL TABLE CONSTRUCTION; RUN PRLOG.
     [return code]
```

Meaning:

JP41 is sent during system restart if an incident makes further journalization unavailable.

The explanation of the error condition given in the message text is:

(01)	the SYS.JADIR file cannot be accessed, see Note
(02)	the resources necessary for journalization are not available.

Any attempt to journalize in SYS JAS will be rejected:
- when a journalizing step is initialized,
- at the start of a TDS,
- or when an attempt is made to open a user file.

Action:

Run the PRLOG utility to print out the SYS.ERLOG file. Ensure that all resources as indicated in the message are available. Shut down the system and perform a COLD restart.

NOTE:
If SYS.JADIR does not exist or is not cataloged, JP41 does not appear.

JP42
```
jasname: JOURNALIZATION IS IMPOSSIBLE
DURING THIS GCOS SESSION.
ERROR WHEN ACCESSING THE efn JAS FILE ;
RUN PRLOG. [return code]
```

Meaning:

JP42 appears on the console and is sent at system restart if an incident makes:
- either further journalization on a specific JAS unavailable,
- or the JAS state file specified by efn (SYS.JASjasname) inaccessible.

If there was journalization on this jasname JAS when the system was previously interrupted by a failure, user file recovery although necessary is not performed.

Action:

If jasname JAS was SWITCHABLE at the previous system interruption, the jasname JAS must be started on the other member of the complex in order to enable the user files recovery. If JAS startup is attempted on this member, message JP75 will be sent.

If the JAS was not SWITCHABLE at the previous system interruption, starting up the jasname JAS will trigger the startup of the job JRU (Load Module H_RECOV) which will reset the logical state of the jasname JAS. Dynamic recovery of user files, however, will not be possible. For each file concerned, run the utilities FILREST then ROLLFWD.

Before starting up the jasname JAS:

- run PRLOG utility since all journalization incidents are logged in the SYS.ERLOG file,
- and make the JAS state file accessible.

NOTE:
If the jas_state_file does not exist or is not cataloged, JP42 does not appear.

JP43
```
jasname: RECOVERY IS IMPOSSIBLE.
SYSTEM UNABLE TO ACCESS THE jasname.JADIR FILE ; RUN PRLOG.
[return code]
```

Meaning:

JP43 is sent at system restart if the jasname JAS cannot recover user files because it cannot access the jasname.JADIR directory.

User files recovery is not performed. Any attempt to open them will result in the abnormal return code FLNAV.

In the case of an HA-type JAS which was SWITCHABLE at the previous system interruption, starting up jasname JAS on this member will cause message JP75 to be sent.

In the case of an HA-type JAS which was UNSWITCHABLE at previous system interruption, starting up jasname JAS on this member, will start up the job JRU (Load Module H_RECOV) which will reset the logical state of the jasname JAS. Dynamic recovery of user files, however, will not be possible.

Action:

Before starting the jasname JAS:

- run the PRLOG utility because all journalization incidents are logged in the SYS.ERLOG file,
- make the jasname.JADIR file accessible,
- and in the case of SYS JAS, run JRU to reset the logical state of SYS JAS,
- if JRU fails, issue the MNJAS REBUILD command.

If user files still cannot be recovered, run the utilities FILREST followed by ROLLFWD on each file affected.

```
JP44   jasname: AFTER JOURNAL MAY BE LEFT IN AN INCONSISTENT STATE
       DUE TO A SYSTEM ERROR.
       (01) [return code]
            RUN THE AFTER JOURNAL RECOVERY UTILITY (JRU).
       (02) [return code]
```

Meaning:

JP44 is sent when the jasname JAS has been corrupted because a TDS application has been unable to provide the list of aborted commitments.

(01)	For SYS JAS or non-HA Private JAS, JRU automatically starts at the beginning of a journalization session.
(02)	for an HA-type JAS, JRU automatically starts when *jasname* JAS is restarted.

Future rollforwards are not guaranteed and any attempt to journalize in the Before or After Journal will be rejected with the abnormal return code DAMAGED:

- when a journalizing step is initialized,
- at TDS startup,
- or when an attempt is made to open a user file.

Action:

(01)	run JRU to reset the logical state of the JAS
(02)	terminate the JAS (TSRV jasname), then restart the JAS (SSRV jasname); restarting the JAS will start up JRU (load module H_RECOV) which resets the logical state of the JAS.

After JRU has executed, run the utilities FILREST then ROLLFWD on all the user files accessed by the TDS.

If JRU fails, issue the MNJAS REBUILD command.

```
JP45    jasname: Xron.ssn[.ckn] COMPLETED WITH REGARD TO JOURNAL
        AFTER COLD OR CLEAN SYSTEM RESTART.
        OPENED USER FILES REMAIN UNSTABLE ;
        RUN FILREST + ROLLFWD FOR EACH FILE.
```

Meaning:

JP45 is sent at Cold or Clean System Restart for each step which was:

- either terminating,
- or taking a checkpoint.

When the system crashed, the checkpoint_number appears if a checkpoint has been taken.

User files opened at the time of crash have not been rolled back and remain unstable.

No message is sent for each step running at the time of crash, but not in termination or checkpoint phase. Such steps are considered aborted and the opened user files remain unstable.

Action:

To recover each file affected, run the utilities FILREST followed by ROLLFWD.

```
JP46    jasname1: JOURNALIZATION CANNOT BE PERFORMED.
        (01) jasname2  JAS IS NOT AVAILABLE ON THIS SYSTEM.
        (02) jasname2.JADIR FILE IS NOT ALLOCATED.
        (03) THE SYSTEM AFTER JOURNAL IS DAMAGED.
             PLEASE, CONTACT THE SYSTEM ADMINISTRATOR.
        (04) gr4.
```

Meaning:

JP46 is printed in the JOR of the executing step.

(01)	at the start of a step requesting jasname2 JAS if neither jasname2 JAS or SYS JAS is available. JP41 or JP42 precedes this message at system restart. The step aborts.
(02)	at the start of a step requesting jasname2 JAS if jasname2.JADIR has not been allocated or cataloged in jasname2.CATALOG. The step aborts.
(03)	applicable to SYS JAS. JP44 (01) precedes this message.
(04)	system error.

Action:

(01)	Create the JAS requested.
(02)	Allocate jasname2.JADIR and catalog it in jasname2.CATALOG.
(03)	Run JRU.
(04)	Run the PRLOG utility to print the ERLOG file; if the message persists call the Service Center.

JP47 `jasname1: JOURNALIZATION CANNOT BE PERFORMED.`
`(01)``THE AFTER JOURNAL IS DAMAGED. THE STEP JRU HAS BEEN LAUNCHED. CHECK IT IS SUCCESSFUL.`
`(02)` `IMPOSSIBLE TO ACCESS jasname.JADIR DIRECTORY FILE.`

Meaning:

JP47 appears when a step begins a journalization session on a non-private JAS:

(01)	when the After Journal is damaged
(02)	when it is impossible to access the JAS directory.

Action:

(01)	The step JRU has been launched. But a TDS session or the file opening for a batch/IOF is rejected with JAP 5 DAMAGED. Try again once JRU is completed. If JRU is not successful, launch the REBUILD command of MNJAS to recover the JAS integrity.
(02)	Make the JAS directory accessible and repeat the action.

```
JP51   OPEN REJECTED FOR efn USER FILE PROTECTED BY jasname1 JAS
       (01) STEP ALREADY CONNECTED TO SWITCHABLE jasname2 JAS.
       (02) jasname1 JAS IS NOT STARTED.
       (03) STEP ALREADY CONNECTED TO HA jasname2 JAS.
       (04) STEP IS WORKING ON SYS JAS.
       (05) TDS IS NOT CONNECTED TO THIS JAS.
```

Meaning:

JP51 is printed in the JOR if an attempt is made to open a user file efn when one of the rules for using the JAS is not respected:

(01)	The step is connected to a switchable JAS. It is only allowed to open files protected by this JAS. The step may only successfully open the efn protected by the switchable jasname1 JAS when jasname2 JAS becomes unswitchable.
(02)	jasname1 JAS must be started before the step can open the user file.
(03)	The step is connected to BLUE JAS and tries to use a file protected by GREEN JAS or vice versa. The user file must be protected by the ***same*** JAS as that to which the step is connected.
(04)	The step uses files protected by SYS JAS and now attempts to open a file protected by switchable jasname1 JAS. The step may only successfully open the efn protected by the jasname1 JAS, when this JAS becomes unswitchable.
(05)	The TDS tries to open a user file protected by jasname1 JAS whereas it was declared at CRCXGEN, as protected by another JAS.

Action: Ensure the above conditions are satisfied.

JP52 `OPEN REJECTED FOR efn USER FILE BECAUSE ITS CATALOG FILE`
(01) `IS NOT CATALOGUED IN SITE.CATALOG`
(02) `IS LINKED TO A HA JAS AND HAS NOT AUTO-ATTACH OPTION.`

Meaning:

JP52 is printed in the JOR of the step when an attempt is made to open a user file efn whose catalog checks are not satisfied.

The conditions for which the catalog is not valid are:

(01)	the catalog concerned is not cataloged in the SITE.CATALOG.
(02)	the catalog of the user file is linked to an HA-type JAS and is not auto-attachable.

Action:

(01)	Ensure that the SITE.CATALOG contains the catalog.
(02)	Ensure that this catalog is auto-attachable before retrying opening the file.

JP53 `OPEN REJECTED FOR efn USER FILE.`
(01) `CONNECTION TO SYS JAS NOT DONE AT STEP INITIATION.`
(02) `SYS JAS NOT AVAILABLE ON THIS SITE.`

Meaning:

JP53 is printed in the JOR of the step when an attempt is made to open the user file efn and the After Journal is unavailable for the step.

SYS.JAS is unavailable for the following reasons:

(01)	SYS JAS was not activated when the step was initialized through a JCL command referencing a journal option.
(02)	SYS JAS has not been created.

If *(02)* appears accompanied by the message JP41 at the console, an incident has occurred at system restart.

Action:

(01):
If at least one file needed by the step is protected by the AFTER Journal, it must:
- either be statically assigned and cataloged with JOURNAL=AFTER or BOTH,
- or include the JCL command DEFINE with JOURNAL=AFTER or BOTH.

If at least one file needed by the step, is protected by the BEFORE Journal, it must:
- either be statically assigned and cataloged with JOURNAL=BEFORE or BOTH,
- or include the JCL command DEFINE with JOURNAL=BEFORE or BOTH.

(02): Create SYS JAS.
If JP41 was printed at system restart, make the SYS JAS Directory available and activate JRU.

```
JP56   OPEN REJECTED FOR efn USER FILE BECAUSE ITS CATALOG FILE IS
       LINKED TO A PRIVATE JAS AND HAS NOT AUTO-ATTACH OPTION
```

Meaning:

JP56 is printed in the JOR of the step when an attempt is made to open a user file efn whose catalog is linked to a non-HA Private JAS, and is not auto-attachable.

Action:

Ensure that this catalog is auto-attachable before trying to reopen the file.

```
JP57   OPEN REJECTED FOR efn USER FILE PROTECTED BY jasname JAS.
       ONLY SYS JAS IS SUPPORTED ON THIS SYSTEM.
```

Meaning:

JP57 is printed in the JOR if an attempt is made to open a user file whose catalog is linked to a non-HA Private JAS when the marketing identifiers "3JAS" or "HA" have not been installed.

Action:

Link the catalog of the user file to SYS JAS through the LINK command of MNJAS, or install a marketing identifier that allows the use of a Private JAS.

```
JP61   jasname: DYNAMIC ROLLFORWARD NOT SUCCESSFUL FOR ron
```

Meaning:

This message is sent when a TDS is rolled forward dynamically:

- either at TDS abort,
- or at system restart.

See JP63, JP64, JP65 or JP66. Some user files using Deferred Updates, may remain unstable.

Action:

Execute the utilities FILREST followed by ROLLFWD for each file concerned.

```
JP62   jasname: efn USER FILE
       DYNAMIC ROLLFORWARD SUCCESSFUL, ROLLED FORWARD RECORDS = n
```

Meaning:

JP62 is for information only and is printed in the JOR of the TDS concerned for each file on which dynamic rollforward was successfully performed.

Action: None

```
JP63   jasname: DYNAMIC ROLLFORWARD CANNOT BE PERFORMED.
                [return code]
```

Meaning:

JP63 is printed in the JOR of the TDS concerned as a warning that all the resources such as segments, files and memory, needed for dynamic rollforward, cannot be created or allocated. A return code may accompany this message. User files may remain unstable.

Action:

Execute the utilities FILREST followed by ROLLFWD for each file concerned.

```
JP64   jasname: DYNAMIC ROLLFORWARD FAILURE,
                READ ERROR ON JOURNAL FILES.
```

Meaning:

JP64 is printed in the JOR of each TDS that is dynamically rolled forward. The dynamic rollforward mechanism cannot read the After Journal files. The system does not know which user file or which TDS is involved. User files may remain unstable.

Action:

Execute the utilities FILREST followed by ROLLFWD for each file concerned.

JP65
```
jasname: efn USER FILE
DYNAMIC ROLLFORWARD NOT PERFORMED.
(01) UNABLE TO CREATE A FD FOR THIS FILE : [return code]
(02) UNABLE TO ASSIGN THIS FILE : [return code]
(03) UNABLE TO OPEN THIS FILE : [return code]
(04) DYNAMIC ROLLFORWARD CANCELLED BY OPERATOR.
```

Meaning:

JP65 is printed in the JOR of the TDS concerned if dynamic rollforward cannot be performed for the following reasons:

(01)	the system cannot create a file descriptor,
(02)	the system cannot assign the user file,
(03)	the system cannot open the user file,
(04)	the operator has cancelled a request for mounting the disk volume containing the file.

Action:

Execute the utilities FILREST followed by ROLLFWD for each file concerned.

JP66
```
jasname: efn USER FILE
DYNAMIC ROLLFORWARD NOT SUCCESSFUL, ROLLED FORWARD
RECORDS=n
(01) NUMBER OF ERRORS = m
(02) UNABLE TO CLOSE THIS FILE. return-code
```

Meaning:

JP66 is printed in the JOR of the TDS concerned indicating the following error conditions:

(01)	An error has occurred while applying After Images to the file efn.
(02)	The After Images have been successfully applied to the file efn but it cannot be closed.

Action:

Execute the utilities FILREST followed by ROLLFWD for each file concerned.

```
JP71   jasname: UNABLE TO START. SYS.JRNAL FILE IS UNAVAILABLE
                OR TOO SHORT.
```

Meaning:

JP71 appears at the console when CMSC attempts to start an HA-type JAS on a member and Before Journalization is not operational on the member because SYS.JRNAL:

- either is not accessible at System Restart,
- or is too small.

Action:

Create SYS.JRNAL if it does not exist. If it does exist, allocate a larger extent for the file. In both cases, perform a Cold or Clean System Restart.

```
JP72   jasname: UNABLE TO START. CONCURRENT DELETE OR CREATE
                COMMAND IS RUNNING.
```

Meaning:

JP72 appears at the console when CMSC attempts to start an HA-type JAS on a member while a concurrent MNJAS is executing on the member.

Action:

Wait for the MNJAS step to terminate before proceeding.

```
JP73   jasname: UNABLE TO ACCESS efn FILE.
              (01) JAS IS NOT STARTED.
              (02) JAS IS BEING STOPPED.
```

Meaning:

JP73 appears on the console to indicate that the status file SYS.JASjasname specified by efn is not accessible on the member when CMSC attempts:

(01)	to start an HA-type JAS,
(02)	to read or modify the state of an HA-type JAS.

Action:

Ensure that SYS.JASjasname is available for access.

```
JP74   jasname: UNABLE TO ACCESS jasname.JADIR DIRECTORY FILE.
                JAS IS BEING STOPPED.
```

Meaning:

JP74 appears at the console when a CMSC command has been issued to set an HA-type JAS in the ACTIVE state on a member but jasname.JADIR cannot be accessed on the member.

Action:

Ensure that jasname.JADIR is available for access.

```
JP75   jasname: RECOVERY IS NEEDED. REFER TO HA DOCUMENTATION
```

Meaning:

JP75 appears at the console when a CMSC command has been issued to set an HA-type JAS in the ACTIVE state on the member *A* and the HA-type JAS has discovered that Recovery which should have been performed was not. Recovery was prevented probably because jasname.JADIR could not be accessed.

The jasname JAS was previously in the state:

- either in EMPTY state, after System Restart of member *A*,
- or in BACKUP state after a TAKEOVER_MEMBER with STRONG or FORCE option on member *A*.

Action:

In the following procedure, member A is the system for which the message JP75 has appeared (member *B* is the other system):

- first prevent automatic restart of member *A*,
- issue the appropriate command according to the following conditions:
  - if another SWITCHABLE HA-type JAS is ACTIVE on member *A*, provoke its crash through the command TKMB STRONG,
  - if not, provoke its crash through the SPOS command SR.
- wait for CMSC to resynchronize on member *B*:
  - if there was an ACTIVE SWITCHABLE HA-type JAS on member *A*, a Recovery Step is started for that JAS,
  - if jasname JAS (the JAS to be started up on member *A*) was in the BACKUP state on member *B*, a Recovery Step is activated for this JAS.
- issue the command DSRV jasname to determine the status of jasname JAS:
  - if it is ACTIVE and SWITCHABLE, the corrective action is complete,
  - if it is EMPTY, it must be set to BACKUP through the command SSRV jasname MEMBER=(B,A) RESYNC,
  - when it is BACKUP, enter the command TKMB FORCE on member *B* to activate a Recovery Step for the JAS concerned.

- when the step terminates, the user files are recovered in a stable state; restart member *A*.

If Recovery cannot take place on member *B*, issue the command SSRV jasname on member *A* to activate JRU for the HA-type JAS concerned. *Although this allows reconstituting a consistent journal, user files are left in an unstable state.*

```
JP76   jasname: see applicable text following
       JOURNALIZATION REMAINS UNAVAILABLE.

       (01) RECOVERY OF USER FILES NOT COMPLETED.
       (02) RECOVERY OF JOURNAL FILES NOT COMPLETED.
```

Meaning:

JP76 appears at the console when an attempt has been made to Switch or Activate an HA-type JAS:

(01)	an attempt has been made to Switch an HA-type JAS but the Recovery Step has failed to complete successfully. Takeover is impossible but user file recovery can still be performed at restart of the other system.
(02)	The startup of JRU has failed when the CMSC command was issued to set the state of an HA-type JAS to ACTIVE. Activating jasname JAS is therefore impossible.

Action:

Run the utility PRLOG to print out the contents of the SYS.ERLOG file to analyze and correct the problem and try to reactivate the JAS by issueing the command SSRV jasname.

```
JP77   jasname: BUSY BY BATCH STEPS USING SYS JAS OR NON HA
                TDS STEPS UNABLE TO BECOME SWITCHABLE;
                STOP THESE STEPS.
```

Meaning:

JP77 appears at the console when a CMSC command has been issued to set the state of an HA-type JAS to ACTIVE-SWITCHABLE whereas this JAS is already protecting a non-HA TDS or a Batch Step using SYS JAS.

Action:

Terminate the non-HA TDS or wait for the Batch Step to terminate.

```
JP78   jasname: STOP STRONG REQUESTED FOR THIS JAS.
                ABORT OF THE STEP IS REQUESTED.
```

Meaning:

JP78 is printed in the JOR(s) of all the steps using the JAS. Either a CMSC command TSRV STRONG has been issued to the JAS or timeout has occurred on the JAS. All the steps using this JAS receive the return code ABORTPG when journalization is attempted on this JAS. JAS will effectively terminate when all the steps have aborted.

If the step is connected to the JAS without journalizing, it only needs to be disconnected from the JAS, which will be obtained when it has deassigned all the files protected by this JAS.

Action:

Terminate or disconnect the step.

```
JP79  jasname:STOP REQUESTED FOR THIS JAS.
              THERE ARE STILL STEPS CONNECTED TO IT.
              USE DJAS COMMAND TO IDENTIFY STEPS TO BE TERMINATED.
```

Meaning:

JP79 appears at the console when a CMSC TERMINATE command with FORCE or WEAK has been issued to the JAS. If there are Batch or IOF steps still connected to this JAS, the message appears repeatedly at the console until all the steps terminate.

If a *strong termination* is required, the message JP78 is sent in addition, to the JOR of each step concerned. JAS is effectively terminated when all the steps abort or are disconnected from the JAS.

A step which has journalized will only be disconnected when it terminates. However, a step which accesses protected files but not journalized, will be disconnected from the JAS when it no longer accesses these protected files. Changing a library under MNLIB for example, will disconnect the step from the JAS which has protected the assigned library if the new library is no longer protected by this JAS.

Action:

Issue the MNJAS DISPLAY_JAS command to list the steps connected to this JAS. Terminate the steps.

```
JP80   JAS: jasname
      (01): NO JOBS LINKED TO THIS JAS
      (02): ron.ssn job_name user_name job_class lm_name
```

Meaning:

JP80 appears on the console for each JAS when the command DJAS is issued.

(01)	self-explanatory.
(02)	occurs as many times as there are jobs connected (or linked) to the JAS.

Action: None

```
JP91   jasname: STEP NOT REPEATED BECAUSE IT IS WORKING
                WITH A HA JAS
```

Meaning:

JP91 is printed at system restart in the JOR of each job working with an HA JAS if the step is repeatable.

At GCOS READY, CMSC restarts the HA-type JAS either on the same member or on the other member of the complex. TDSs *WATCHED BY CMSC* will start without operator intervention on the appropriate member. Batch and IOF steps, and TDSs not *watched* by CMSC must be restarted by their respective users once the HA-type JAS resumes.

Action: None

```
JP92   jasname: ron COMMAND TRANSFER WAITS FOR efn JOURNAL FILE.
                (01) no text
                (02) USED BY ROLLFWD OR DUMPJRNL UTILITY.
```

Meaning:

JP92 appears at the console when the sender file concerned by the command TRANSFER is busy:

(01)	the reason is not known,
(02)	either the ROLLFWD or the DUMPJRNL utility is running.

The MNJAS step is set to an inactive state for five minutes before reusing the sender file. If the file continues to be busy for more than ten consecutive times, TRANSFER stops.

Action:

(01)	find out why the file is busy and make it available.
(02)	none.

```
JP93   jasname: COMMAND TRANSFER WAITS FOR n DEVICE(S) devclass
```

Meaning:

This message is sent to the administrator's JOR and console if he/she is connected. The MNJAS step is suspended until the necessary resources have been freed. If the command was sent in IOF, a break will stop the step wait and interrupt the command. The JAS for which the TRANSFER command came is specified. If the transfer is done on dual copy of the secondaries, "n" will have a value of "1" or "2" depending on the number of devices unavailable. If not, "n" will equal 1. The devclass of the device to be freed is given.

Action:

Ensure that an operator releases the necessary resources.

```
(01)JP94 jasname: ron COMMAND TRANSFER WAITS FOR 1 DEVICE devclass
          PLEASE, FREE THE NECESSARY RESOURCE

(02)JP94 jasname: ron COMMAND TRANSFER NEEDS 2 DEVICES devclass
          AND WAITS FOR THEM
          PLEASE, FREE THE NECESSARY RESOURCES

(03)JP94 jasname: ron COMMAND TRANSFER NEEDS 2 DEVICES devclass
          AND WAITS FOR ONE OF THEM
          PLEASE, FREE THE NECESSARY RESOURCE
```

Meaning:

This message is sent to the operator's console. There are three possible messages. The message depends on the number of devices necessary and unavailable: single copy, dual copy with one device unavailable, or dual copy with two devices unavailable.

1. Single copy scenario
2. Dual copy with one device unavailable scenario
3. Dual copy with two devices unavailable scenario

The MNJAS step waits for the necessary resources to be freed. The JAS for which the TRANSFER command came is specified. The devclass of the device to be freed is given, along with the ron of the MNJAS step.

Action:

The operator must release one or two devices to allow secondary file transfer. If this is not possible, and if the command is executed in batch mode, the operator can kill the enqueued job. If the command is executed in IOF, The system administrator can perform a break.

This message is repeated. It is stopped as soon as the required resources are allocated, or in the event of an Administrator IOF break.

B.2.2 RF Messages

```
RF11   jasname: ron WARNING
       IF NO CONCURRENT TRANSFER COMMAND IS SUBMITTED
       see applicable message option following
       volume1 volume2 volume3 volume4 volume5
       volume6 volume7 volume8 volume9 volume10

       (01) ROLLFWD WILL USE THE FOLLOWING MEDIA
       (02) DUMPJRNL WILL USE THE FOLLOWING MEDIA
```

Meaning:

RF11 appears at the console after the utility has identified the list of media supporting the After Journal files it has to read:

(01)	applies to the ROLLFWD utility,
(02)	applies to the DUMPJRNL utility.

This message is sent for every 10 media.

`IF NO CONCURRENT TRANSFER COMMAND IS SUBMITTED` appears as a *warning* that the specified media may change if a TRANSFER occurs on the After Journal files before being read by the ROLLFWD or DUMPJRNL utility.

Action:

Make the volume(s) containing the files available. If the media required are not online, the utility stops and a MOUNT message appears on the console.

RF12
```
jasname: ron WARNING
(01) NEXT MEDIA USED BY ROLLFWD WILL BE volume_name
(02) IF NO CONCURRENT TRANSFER COMMAND IS SUBMITTED
     NEXT MEDIA USED BY ROLLFWD WILL BE volume_name
(03) NEXT MEDIA USED BY DUMPJRNL WILL BE volume_name
(04) IF NO CONCURRENT TRANSFER COMMAND IS SUBMITTED
     NEXT MEDIA USED BY DUMPJRNL WILL BE volume_name
```

Meaning:

RF12 appears at the console when the next medium is allocated to the utility:

(01-02): applies to the ROLLFWD utility
(03-04): applies to the DUMPJRNL utility.

`IF NO CONCURRENT TRANSFER COMMAND IS SUBMITTED` appears as a *warning* that the specified media may change if a TRANSFER occurs on the After Journal files before being read by the ROLLFWD or DUMPJRNL utility:

(01+03): the next medium cannot change because it is supporting an After Journal file which has already been released on TRANSFER.

Action: None

RF21
```
jasname: ron WARNING
RECOVERY FOR tds_name TDS WILL USE THE FOLLOWING MEDIA
volume1 volume2 volume3 volume4 volume5
volume6 volume7 volume8 volume9 volume10
```

Meaning:

RF21 appears at the console when a TDS is dynamically rolled forward after the list of media containing the After Journal files it has to read, has been determined.

This message is sent for every 10 media.

If dynamic rollforward is activated for several TDSs at System Restart, tds_name is ALL.

Action:

Make the volume(s) containing the files available. If the media required are not online, dynamic rollforward stops and a MOUNT message appears on the console.

RF22
```
jasname: ron WARNING
NEXT MEDIA USED BY RECOVERY FOR tds_name TDS
WILL BE volume_name
```

Meaning:

RF22 appears at the console for information only when the next medium for the After Journal is allocated for the dynamic rollforward of a TDS (tdsname) or several TDSs (ALL).

Action: Make sure the media indicated is available.

RF31
```
jasname: ron
(01) ROLLFWD WAITS FOR FILE efn USED BY TRANSFER COMMAND
(02) DUMPJRNL WAITS FOR FILE efn USED BY TRANSFER COMMAND
```

Meaning:

RF31 appears at the console when the utility is started while a TRANSFER command is executing on the current After Journal file:

(01)	applies to the ROLLFWD utility,
(02)	applies to the DUMPJRNL utility.

ROLLFWD or DUMPJRNL is set to the inactive state and will retry after five minutes. If the file is still not available after ten attempts, the utility aborts. The files to be rolled forward or the OUTFILE to be updated by DUMPJRNL, do not need to be restored.

Action: None

B.3 Messages Issued by the Recovery Step

B.3.1 Operator Answers

When a file cannot be accessed, a message describing the problem is sent to the console, the JOR and the SYSOUT of the recovery step. Then message RY01 asks if the operator wants to intervene. The operator should answer YES unless unable to remain on the site.

When message RY02 appears, the operator:

- checks if the site configuration is correct as follows:
  - the file which the recovery step cannot access is on a sharable disk,
  - the disk is effectively shared; if the disk is only premounted, the recovery can be performed but there will be problems when journalization on the failed system resumes.
- then answers:
  - either YES if the file can be made accessible,
  - or NO if the file cannot be made accessible because it is not sharable for example.

The errors detected by the recovery step will:

- either interrupt the step,
- or allow the step to continue depending on:
  - the severity of the error,
  - and the potential decision of the operator.

If the operator does not reply to an ***Intervention Request*** RY01 within 2 minutes, no further intervention requests will be made until the end of the recovery. The operator is considered absent from the site. Recovery runs in ***automatic mode***.

As long as the execution of the recovery step is not in automatic mode it is said to be in ***intervention mode***.

If the operator does not answer to a ***Decision Request*** RY11 within 2 minutes, no further decision requests will be made until the end of the recovery. Recovery runs in ***automatic decision mode***.

Under ***meaning*** for each RY message, the phrase *appears on the console* applies when recovery runs in ***intervention mode***.

B.3.2 RY Messages

RY01
```
jasname:IF YOU WANT TO CHECK THAT THE SITE CONFIGURATION
IS CORRECT, TYPE YES
IF YOU DO NOT WANT TO BE CONSULTED FROM NOW ON, TYPE NO
CHECK?
```

Meaning:

RY01 appears at the Console and is printed in the SYSOUT of the Recovery Step. An error has been detected when accessing a file. This message follows a message indicating which resource is inaccessible.

Action:

If YES is keyed in, the step runs in *intervention* mode and the message RY02 appears. YES requires checking access to drivers and ensuring that the disks are sharable.

The step runs in *automatic decision* mode if:

- either no reply is given within 2 minutes, timeout occurs for which TO (TimeOut) is displayed in the SYSOUT of the Recovery Step,
- or NO is keyed in.

RY02
```
jasname:PLEASE CHECK ALL THE NECESSARY CONDITIONS
        TO RECOV STEP EXECUTION ARE MET.
        WHEN READY, TYPE YES
        IF THE CONDITIONS CANNOT BE MET, TYPE NO
        READY?
```

Meaning:

RY02 appears at the Console and is printed in the SYSOUT of the Recovery Step. This message appears if YES is keyed in to RY01.

Action:

Check that all sharable disks are accessible by the BACKUP member.

If not, issue the following commands:

- `RDV MSnn SHARED` to establish the access path to the disk drive,
- `MDV MEDIA_NAME CLEAR` to remove any lock on the files,
- `[RSCMIR]` option, where applicable, to resynchronize the volume and its mirrored counterpart.

If the conditions cannot be met despite intervention, key in NO. The recovery step will decide according to the severity of the error whether to stop or to further request the operator to continue or to stop.

If the requested conditions are met, key in YES. The recovery step reattempts accessing the file.

If no reply is given within 2 minutes, TO (TimeOut) appears in the SYSOUT of the recovery step and the recovery step continues in *automatic* mode.

```
RY03   jasname:THE OPERATOR DOES NOT WANT TO BE CONSULTED ANYMORE
               OR IS NOT PRESENT ON SITE.
               RECOV STEP SWITCHES TO AUTOMATIC MODE.
```

Meaning:

RY03 appears at the Console and is printed in the SYSOUT of the Recovery Step. This message appears when the step runs in *automatic* mode if in response to RY01:

- either NO is keyed in,
- or the default timeout TO occurs.

The step continues or terminates according to the importance of the missing file.

Action: None

```
RY10   jasname:A RESOURCE NECESSARY TO RESTART ON BACK-UP MEMBER
               IS MISSING. IN AUTOMATIC MODE, RECOV STEP HAS
               DECIDED TO STOP THE RECOVERY PROCESS.
```

Meaning:

RY10 appears at the Console and is printed in the SYSOUT of the Recovery Step. This message appears when the step runs in *automatic* mode if in response to RY01:

- either NO is keyed in,
- or the default timeout TO occurs.

No further recovery takes place since the missing file is essential.

Action: None

```
RY11   jasname:IF YOU WANT TO RUN THE RECOVERY AS FAR AS POSSIBLE,
               TYPE YES.
               IF NOT, TYPE NO. CONTINUE?
```

Meaning:

RY11 appears at the Console and is printed in the SYSOUT of the Recovery Step. This message appears:

- when the step runs in *intervention* mode after YES is keyed in to RY01 and RY02,
- ***and*** when the file required by the step continues to be inaccessible.

Action:

If YES is keyed in, the recovery step continues with the resulting consequences depending on what the inaccessible file is:

- if it is a user file, this file will be unstable and no further immediate recovery will take place for this file,
- if it is a system file such as the TDS before journal, all the files accessed by this TDS may remain unstable, and the TDS itself may not be able to restart.

If NO is keyed in, the recovery step will be immediately interrupted.

In the absence of a reply, the default timeout TO occurs after 2 minutes and the recovery step continues.

```
RY12   jasname:AFTER INTERVENTION OF OPERATOR,
               ACCESS TO INACCESSIBLE RESOURCE IS TRIED AGAIN.
```

Meaning:

RY12 appears at the Console and is printed in the SYSOUT of the Recovery Step. RY12 is for information only and appears if YES is keyed in to RY02.

Action: None

```
RY15   jasname:A RESOURCE NÉCESSARY TO RESTART ON BACK-UP MEMBER
               IS MISSING.
               RECOV STEP IS STOPPED ON DECISION OF OPERATOR.
```

Meaning:

RY15 appears at the Console and is printed in the SYSOUT of the Recovery Step. RY15 is for information only and appears when NO is issued in response to RY11.

Action: See *File Inaccessibility* (section 5.7.1).

```
RY16   jasname:A RESOURCE NÉCESSARY TO RESTART ON BACK-UP MEMBER
               IS MISSING. RECOV STEP IS BEING STOPPED.
```

Meaning:

RY16 appears at the Console and is printed in the SYSOUT of the Recovery Step. RY16 is for information only. An essential resource is not available and the operator **cannot** make it available by ensuring that all sharable disks are accessible by the BACKUP member requested in RY02.

Action: See *File Inaccessibility* (section 5.7.1).

```
RY17   jasname:THE OPERATOR HAS DECIDED TO RUN THE RECOVERY
               AS FAR AS POSSIBLE.
```

Meaning:

RY17 appears at the Console and is printed in the SYSOUT of the Recovery Step. RY17 is for information only and appears if in response to RY11:

- either YES is keyed in,
- or the default timeout TO occurs.

Action: See *File Inaccessibility* (section 5.7.1).

```
RY18   jasname:IN AUTOMATIC MODE, RECOV STEP HAS DECIDED TO RUN
               THE RECOVERY AS FAR AS POSSIBLE.
```

Meaning:

RY18 appears at the Console and is printed in the SYSOUT of the Recovery Step. This message follows on from RY03 and appears when the step runs in *automatic* mode if in response to RY01:

- either NO is keyed in,
- or the default timeout TO occurs.

The missing resource is not essential to recovery and the step continues.

Action: See *File Inaccessibility* (section 5.7.1).

```
RY19   jasname:IN AUTOMATIC DECISION MODE, RECOV STEP HAS DECIDED
               GO ON RUNNING the recovery.
```

Meaning:

RY19 appears at the Console and is printed in the SYSOUT of the Recovery Step. RY19 is the result of the absence of a reply to RY11 where the resource required is not essential for recovery. The step continues without considering this resource.

Action: See *File Inaccessibility* (section 5.7.1).

```
RY20   jasname:JAS DIRECTORY ACCESS PROBLEM.
```

Meaning:

RY20 appears at the Console and is printed in the JOR and SYSOUT of the Recovery Step. The file jasname.JADIR cannot be accessed by the recovery step. RY01 will then appear requesting operator intervention.

Action: See *File Inaccessibility* (section 5.7.1).

RY21 `jasname:JAS DIRECTORY SUBFILE ACCESS PROBLEM.`

Meaning:

RY21 appears at the Console and is printed in the JOR and SYSOUT of the Recovery Step. A *critical* error has occurred when accessing a subfile of jasname.JADIR. No further operator intervention is possible and the Recovery Step terminates.

Action: See *File Inaccessibility* (section 5.7.1).

RY22 `jasname:SEGMENT CREATION IS IMPOSSIBLE.`

Meaning:

RY22 is printed in the SYSOUT of the recovery step. The segment necessary for recovery cannot be reserved. No further operator intervention is possible and the Recovery Step terminates.

Action: See *HA-type JAS and Recovery* (section 5.6).

RY23 `jasname: IMPOSSIBLE TO ACCESS SYS.JRNAL FILE.`

Meaning:

RY23 appears at the Console and is printed in the JOR and SYSOUT of the Recovery Step. SYS.JRNAL cannot be accessed by the step. RY01 will then appear requesting operator intervention.

Action: See *File Inaccessibility* (section 5.7.1).

RY24 `jasname: FD CREATION IS IMPOSSIBLE FOR SYS.JRNAL FILE.`

Meaning:

RY24 is printed in the SYSOUT of the Recovery Step. The system structures necessary for recovery cannot be created. No further operator intervention is possible and the Recovery Step terminates.

Action: See *HA-type JAS and Recovery* (section 5.6).

RY25 `jasname: IMPOSSIBLE TO FIND SYS.JRNAL FEATURES.`

Meaning:

RY25 is printed in the SYSOUT of the recovery step. The physical attributes of SYS.JRNAL cannot be retrieved. No further operator intervention is possible and the Recovery Step terminates.

Action: See *HA-type JAS and Recovery* (section 5.6).

RY26 `jasname: WRONG SYS.JRNAL BLOCK SIZE.`

Meaning:

RY26 is printed in the SYSOUT of the recovery step. The block size of SYS.JRNAL is incorrect. No further operator intervention is possible and the Recovery Step terminates.

Action: See *HA-type JAS and Recovery* (section 5.6).

RY27
```
jasname: IMPOSSIBLE TO CREATE BEFORE JOURNAL
         CONTROL STRUCTURES.
```

Meaning:

RY27 is printed in the SYSOUT of the recovery step. The control structures necessary for rolling back the files cannot be created. No further operator intervention is possible and the Recovery Step terminates.

Action: See *HA-type JAS and Recovery* (section 5.6).

RY28 `TDS tds_name user_ron CANNOT BE RECOVERED.`

Meaning:

RY28 is printed in the SYSOUT of the recovery step. Rollback and/or rollforward cannot be executed for this TDS since it cannot supply the list of CUs (commitment unit) aborted or committed:

- either on recovery before system crash,
- or during the Recovery Step.

Since the files accessed by this TDS remain unstable, this TDS cannot be restarted.

However, recovery continues for other TDSs.

Action: See *Journal Recovery on Takeover* (section 5.7.3).

RY29 `BATCH user_ron CANNOT BE RECOVERED.`

Meaning:

RY29 is printed in the SYSOUT of the recovery step. Recovery cannot be performed on this batch or IOF job.

Action: None

```
RY31   ROLLBACK SUCCESSFUL. NUMBER OF ROLLED BACK BLOCKS number
RY32   ROLLBACK NOT SUCCESSFUL.
       NUMBER OF ROLLED BACK BLOCKS number
```

Meaning:

RY31 and RY32 are printed in the SYSOUT of the recovery step. At the end of the rollback of each file accessed by the step before system crash, either RY31 or RY32 appears. Both messages follow a message naming the file and indicate the result of the rollback on the file.

Action:

Run the ROLLFWD utility on files which have remained unstable.

```
RY33   INCIDENT ON USER FILE:
       (01) ERROR DURING OPEN.
       (02) WRITE ERROR.
       (03) READ ERROR ON JOURNAL.
       (04) CHANEL PROGRAM WRITE ERROR.
       (05) CANCEL REQUEST DURING OPEN.
       (06) I/O ERROR ON FICB.
       (07) FD CREATION IMPOSSIBLE.
       (08) ERROR DURING CLOSE.
       (09) SHARED FILES TABLE ACCESS ERROR.
       (10) A HEURISTIC DECISION HAS BEEN TAKEN
            ON STATE OF COMMIT ACCESSING THIS FILE: EFN=filename
```

Meaning:

RY33 is printed in the SYSOUT of the recovery step. An incident has occurred during the rollback of a step. RY33 appears after a message identifying the user file.

The state of the user file is as follows:
(01)-(09): unstable,
(10): stable but potentially inconsistent since rollback may have been incorrect.

Action: None

RY34 `ROLLBACK PROCESSING STOPPED.`

```
(01) BUFFER ACQUISITION PROBLEM.
(02) SEMAPHORE ACQUISITION PROBLEM.
(03) JOURNAL LOGICAL TRACK ADDRESS ERROR.
(04) CHANNEL PROGRAM BUILD ERROR.
(05) BAD DIALOG WITH TDS.
(06) JCT CONSTRUCTION ERROR.
(07) READ ERROR ON JOURNAL.
(08) EXTENT OPEN DIFFICULTY.
(09) ABNORMAL USER FILE ASSIGN.
(10) ABNORMAL GLOBAL OPEN
(11) NO EOF ON BEFORE JOURNAL FILE.
(12) FD CREATION IMPOSSIBLE.
(13) THE USER FILE CANNOT BE READ IN SYS.JRNAL.
(14) REPORT SEGMENT ACCESS ERROR.
(15) SHARED FILES SEGMENT ACCESS ERROR.
```

Meaning:

RY34 is printed in the SYSOUT of the Recovery Step. An error at the initialization phase of rollback has been detected causing the immediate termination of the Recovery Step. The message text is self-explanatory for all cases.

Action: None

RY35
```
ROLLBACK IS IMPOSSIBLE FOR UPDATES MADE BY JOB user_ron
[tds_name] ON FILE: EFN= file_name RC= return-code
```

Meaning:

RY35 is printed in the SYSOUT of the Recovery Step. An incident has occurred when the file named in EFN was being accessed. RY01 will then appear requesting operator intervention and RY35 will appear again if the file remains inaccessible.

Action:

Run the utilities FILREST then ROLLFWD to reconstruct the file.

RY36
```
ROLLBACK IS IMPOSSIBLE FOR UPDATES MADE BY JOB user_ron
[tds_name]: BEFORE JOURNAL FILE NOT ACCESSIBLE.
RC = return-code
```

Meaning:

RY36 is printed in the SYSOUT of the recovery step. An incident has occurred while a Before Journal file of this TDS was being accessed. RY01 will then appear requesting operator intervention and RY36 will appear again if the file remains inaccessible.

Action:

Run the utilities FILREST then ROLLFWD to reconstruct the file.

```
RY37   RECOVERY IS IMPOSSIBLE FOR UPDATES MADE BY JOB user_ron
       [tds_name] ON FILE: EFN= file_name RC= return-code
```

Meaning:

RY37 is printed in the SYSOUT of the recovery step. Since the file (file_name) cannot be accessed, it cannot be recovered for JOB user_ron using Rollback and/or Rollforward. RY01 will then appear requesting operator intervention and RY37 will appear again if the file remains inaccessible.

Action:

Run the utilities FILREST then ROLLFWD to reconstruct the file.

```
RY40   ROLLFORWARD IS IMPOSSIBLE FOR UPDATES MADE BY JOB
       user_ron [tds_name] ON FILE: EFN= file_name RC= return-code
```

Meaning:

RY40 is printed in the SYSOUT of the Recovery Step. An incident has occurred when the file named in EFN was being accessed. RY01 will then appear requesting operator intervention and RY40 will appear again if the file remains inaccessible.

Action:

Run the utilities FILREST then ROLLFWD to reconstruct the file.

```
RY41  (01) ROLLFORWARD IS IMPOSSIBLE: AFTER JOURNAL FILE file_name
           NOT ACCESSIBLE. RC= return-code
      (02) AFTER JOURNAL FILE file_name NOT ACCESSIBLE.
           RC= return-code
```

Meaning:

RY41 is printed in the SYSOUT of the recovery step. An incident has occurred when the file named in EFN was being accessed. RY01 will then appear requesting operator intervention and RY41 will appear again if the file remains inaccessible.

The occurrence of the incident is indicated as follows:

(01)	during file recovery,
(02)	during the recovery of the After Journal by JRU.

Action:

Run the utilities FILREST then ROLLFWD to reconstruct the file.

```
RY43   ROLLFORWARD NOT PERFORMED FOR FILE file_name
       (01) UNABLE TO CREATE A FD FOR THIS FILE.
       (02) UNABLE TO ASSIGN THIS FILE, RC = return-code
       (03) UNABLE TO OPEN THIS FILE, RC = return-code
       (04) DYNAMIC ROLLFORWARD CANCELLED BY OPERATOR.
```

Meaning:

RY43 is printed in the SYSOUT of the recovery step.

Dynamic rollforward cannot be performed because:

(01)	the system cannot create a file descriptor
(02)	the user file cannot be assigned
(03)	the user file cannot be opened
(04)	intervention to make the file accessible has failed.

Action:

Run the utilities FILREST then ROLLFWD to reconstruct the file.

```
RY44 *** FILE: file_name
         DYNAMIC ROLLFORWARD NOT SUCCESSFUL
         NUMBER OF ROLLED FORWARD RECORDS: records
         (1) NUMBER OF ERRORS: errors
         (2) UNABLE TO CLOSE THIS FILE, RC = return-code
```

Meaning:

RY44 is printed in the SYSOUT of the recovery step.

(01)	an error has occurred while applying After Image(s)
(02)	the file on which After Image(s) have been successfully applied, cannot be closed.

The file remains unstable in both cases.

Action:

Run the utilities FILREST then ROLLFWD to reconstruct the file.

```
RY45 *** FILE: file_name
         DYNAMIC ROLLFORWARD SUCESSFUL. NUMBER OF ROLLED
         FORWARD RECORDS: number
```

Meaning:

RY45 is printed in the SYSOUT of the recovery step. This message is sent for each file for which the dynamic rollforward was successful, each time a step is processed by the recovery step.

Action: None

```
RY46   DYNAMIC ROLLFORWARD NOT SUCESSFUL FOR THIS STEP
```

Meaning:

RY46 is printed in the SYSOUT of the Recovery Step. This message is preceded by a message identifying the step and is sent when a TDS is rolled forward dynamically.

Further explanation of the error is given in the same SYSOUT by the messages RY43, RY47, RY48, or RY44, where applicable.

Some user files specified with the Deferred Update may remain unstable.

Action:

Run the utilities FILREST then ROLLFWD to reconstruct the file.

```
RY47   DYNAMIC ROLLFORWARD CANNOT BE PERFORMED FOR THIS STEP.
```

Meaning:

RY47 is printed in the SYSOUT of the Recovery Step. All the resources concerning segment(s), file(s) and internal structures needed to perform a dynamic rollforward cannot be created.

Action: None

```
RY48   DYNAMIC ROLLFORWARD FAILURE, READ ERROR ON JOURNAL FILES.
```

Meaning:

RY48 is printed in the SYSOUT of the recovery step. Dynamic rollforward cannot be performed since the After Journal files cannot be read.

Action: None

Meaning:

RY50 appears in the JOR and the SYSOUT of the Recovery Step. The step has successfully terminated allowing switching of jasname JAS.

Action: See *HA-type JAS and Recovery* (section 5.6).

RY51 `jasname:RECOVERY FAILED.`

Meaning:

RY51 appears in the JOR and the SYSOUT of the Recovery Step. An error preventing recovery has occurred with a result that jasname JAS cannot be switched.

Action: See *HA-type JAS and Recovery* (section 5.6).

RY52 `jasname:MESSAGE UNKNOWN. RC= return-code`

Meaning:

RY52 appears in the JOR and the SYSOUT of the Recovery Step. An internal error has occurred.

Action: Call the Service Center.

RY61
```
IMPOSSIBLE TO RUN RECOVERY:
(1) INCORRECT SYSTEM STRUCTURES.
(2) INCORRECT COMMAND FORMAT.
(3) UNKNOWN COMMAND.
(4) UNKNOWN JAS NAME.
(5) INCORRECT RECOVERY OPTION.
(6) SYS JAS IS JOURNALIZING OR DOES NOT EXIST.
```

Meaning:

RY61 is printed in the JOR of the Recovery Step. An error during the phase analyzing the call options to the load module H_RECOV, has been detected before invoking:

- either the Recovery Step itself for file recovery,
- or the JRU for recovering the journal(s).

The message text is self-explanatory.

Action: Call the Service Center.

B.4 JRU Messages

The messages emitted by the JRU (Journal Recovery Utility) are identified from RY201 through RY350. They are self-explanatory. See *JRU (Journal Recovery Utility and Functions)* (chapter 7).

Some functions of JRU being called by the recovery step, these messages may appear in the sysout of the recovery step as well.

B.5 TCRF Messages

B.5.1 TCRF

TCRF is executed under the SYSADMIN project. Messages from TCRF appear at the submitter console.

For the duration of the TCRF job, the submitter is advised to remain at the console to reply to questions which require intervention.

Some of the messages are printed in the SYSOUT of the TCRF job on completion.

B.5.2 BU Messages

BU01
```
tcrf-ron ROLLBACK NOT POSSIBLE FOR THE UPDATES MADE BY
          JOB user-ron tdsnm
```

Meaning:

There has been an incident during the assignment or opening of the Before Journal and its extensions or user files.

BU01 is followed by message BU02 and BU03 for user files; otherwise, it is followed by message BU05 or BU07, depending whether the incident takes place during file assignment (BU05) or file opening (BU07).

Action: None

BU02
```
ON FILE:
EFN = efn
MEDIA = volnm-lst
RC = cccccccc --> siu, return code
```

Meaning:

BU02 follows message BU01 and indicates an incident during assignment or opening of a file on which Before Images must be applied. This message is always followed by another message requesting an operator decision.

Action: None

```
BU03  DO YOU WANT TO HOLD JOBS TO PERFORM A TOTAL ROLLBACK
      FOR THIS STEP ?
      ANSWER YES OR NO.
      IF YES, ANSWER WHEN MEDIA MOUNTED.
```

Meaning:

BU03 is preceded by message BU01 and BU02 with the return code BUSY. All disk drives have been allocated to other executing jobs and it is impossible to access the volumes containing the files to be rolled back. The volume containing the file to which an assign is issued, is probably HELD.

Action:

- Either release the required resources and enter YES.
- Or if the resources cannot be relased, enter NO.

If NO is entered, the operator must decide whether to leave all the files for the step concerned in an unstable state or to do a partial rollback.

```
BU04 ron DO YOU WANT TO ONLY ROLLBACK THE OTHER FILES?
         ANSWER YES OR NO.
         IF YES, NON ROLLED BACK FILES INTEGRITY IS NOT GUARANTEED
```

Meaning:

BU04 follows answer NO to BU03.

YES performs partial rollback.
NO results in none of the files being rolled back.

Action:

In the case of YES, run the ROLLFWD utility for the file which has remained unstable.

```
BU05  ron JOURNAL FILE NOT ASSIGNED.
          RC = cccccccc --> siu, return code
```

Meaning:

BU05 is preceded by message BU01.

If the return code is BUSY, message BU08 and BU06 will follow in that order indicating that the disk drive is unavailable (BU08) and requesting the releasing of the volume containing the Before Journal (BU06).

If the return code is not BUSY, no rollback is done if the step concerned is transactional. For a batch step, message BU04 is displayed requesting an operator decision.

Action: See *action* in BU04, BU06 and BU08.

```
BU06  ron MOUNT MEDIA volnm.
          WHEN READY, PRESS EXECUTE.
           OTHERWISE TYPE CR AND PRESS EXECUTE.
```

Meaning:

BU06 is preceded by message BU05.

Action:

When the volume containing the Before Journal is made accessible, the number of the message must be entered and the EXECUTE key pressed.

If the CANCEL_REQUEST command is sent and the executing step is transactional, the step is abandoned; if the executing step is batch, message BU04 is issued.

```
BU07  ron JOURNAL FILE NOT OPENED.
          RC = cccccccc --> siu, return code
```

Meaning:

An incident occurred when the Before Journal and/or its extensions were opened.

Action:

If the executing step is transactional, rollback is not executed. If the executing step is batch, message BU04 is displayed.

```
BU08  ron NO DRIVE AVAILABLE
```

Meaning:

All drives that can be used for this media are not accessible.

Action: See message BU06.

```
BU11  tcrf-ron IMPOSSIBLE TO ACCESS THE TDS SUBSYSTEM FILES
               FOR JOB tds-ron, TDS = tdsnm:
               RC = cccccccc --> siu, return code
```

```
BU12  DO YOU WANT TO STOP THE BACK UP PROCEDURE OR
      CONTINUE WITH NO DYNAMIC ROLLFORWARD PERFORMED
```

```
BU13  AND NO GUARANTEE FOR A STATIC RECOVERY OF THE
      USER FILES JOURNALIZED BY THIS TDS?
```

```
BU14  tcrf-ron ANSWER STOP OR CONTINUE.
```

Meaning:

An incident occurred on the specified TDS while accessing the TDS subsystem files.

Action:

- Either stop the back-up procedure which can be run later,
- Or continue with no dynamic rollforward performed and no guarantee for a static recovery of the user files journalized by this TDS.

```
BU15  trcf-ron IMPOSSIBLE TO ACCESS THE SYS.JADIR OF THE
               FAILED SYSTEM:
               RC = cccccccc -->siu, return code
```

```
BU16  trcf-ron DO YOU WANT TO STOP THE BACK UP PROCEDURE
               OR CONTINUE WITHOUT THE AFTER JOURNAL?
               ANSWER STOP OR CONTINUE.
```

Meaning:

There has been an incident during the access of the After Journal Directory of the failed site.

The line, RC = cccccccc --> siu, return code, provides coded information on error conditions and the system integration unit from which the error originated. Return codes are described in the *Error Messages and Return Codes Manual*.

Action:

- Either STOP the back-up procedure which can be run later,
- Or CONTINUE with the After Journal. In this case, the user files are no longer protected, and may be left in an unstable state.

```
BU17  tcrf-ron DYNAMIC ROLLFORWARD NOT POSSIBLE FOR THE
               UPDATES MADE BY TDS tdsnm DUE TO A LACK OF
               RESOURCE:
```

```
BU18  tcrf-ron AFTER JOURNAL efn BUSY                       (1)
               AFTER JOURNAL MEDIA volnm BUSY               (2)
               NO DRIVE AVAILABLE FOR AFTER JOURNAL MEDIA   (3)
```

```
BU19  tcrf-ron DO YOU WANT TO FREE THE RESOURCE TO PERFORM
               THE DYNAMIC ROLLFORWARD? ANSWER YES (WHEN
               THE RESOURCE HAS BEEN FREED) OR NO
```

Meaning:

A lack of resources means that the dynamic rollforward for the update made by the specified TDS is not possible. The lack of resources may be due to (1), (2) or (3) below:

(1)	an After Journal file is allocated to a job,
(2)	a media of the After Journal is not accessible,
(3)	no drive is available for an After Journal media.

Action:

- Either free the requested resource and continue the back-up procedure ; this is done on YES.
- Or continue without the After Journal on NO leaving the user files unstable; Perform RESTORE and STATIC ROLLFORWARD to recover them.

```
BU20  tcrf-ron DYNAMIC ROLLFORWARD NOT POSSIBLE FOR THE
               UPDATES MADE BY tdsnm:
               RC = cccccccc --> siu, return code

          BU16  trcf-ron DO YOU WANT TO STOP THE BACK UP PROCEDURE
                         OR CONTINUE WITHOUT THE AFTER JOURNAL?
                         ANSWER STOP OR CONTINUE.
```

Meaning:

While the dynamic rollforward of the specified TDS was being performed, an incident occurred during the access of the After Journal files. For an explanation of RC = cccccccc --> siu, return code, see message BU16.

Action:

- Either STOP the back-up procedure which can be run later.
- Or CONTINUE without the After Journal. In this case, the user files may be left in an unstable state. Perform RESTORE and STATIC ROLLFORWARD to recover them.

```
BU21  tcrf-ron DYNAMIC ROLLFORWARD NOT POSSIBLE FOR THE
               UPDATES MADE BY THE JOB tds-ron, TDS tdsnm, ON
               FILE efn
```

```
BU22  tcrf-ron MEDIA vol-lst
               LACK OF RESOURCE: FILE BUSY                  (1)
               LACK OF RESOURCE: NO DRIVE AVAILABLE         (2)
               RC = cccccccc --> siu, return code
```

```
BU23  tcrf-ron DO YOU WANT TO FREE THE RESOURCE TO PERFORM
               THE DYNAMIC ROLLFORWARD FOR THIS TDS?
               ANSWER YES (WHEN THE RESOURCE HAS BEEN FREED)
               OR NO
```

Meaning:

A lack of resources for a user file means the dynamic rollforward for the updates made by a TDS cannot be performed. The lack of resources may be due to (1) or (2) below:

(1)	the user file is busy,
(2)	the user file media is not accessible.

Action:

- Either free the requested resource and continue the back-up procedure; this is done on YES.
- Or continue the back-up procedure on NO leaving the user file unstable. Perform RESTORE and STATIC ROLLFORWARD on the file.

```
BU24  tcrf-ron DYNAMIC ROLLFORWARD NOT POSSIBLE FOR THE
               UPDATES MADE BY JOB tds-ron, TDS tdsnm,
```

```
BU25  tcrf-ron ON FILE efn
               RC = cccccccc --> siu, return code
```

```
BU26  tcrf-ron DO YOU WANT TO TRY THE DYNAMIC ROLLFORWARD
               OF THIS FILE LATER OR LEAVE THIS FILE IN
               UNSTABLE STATE? ANSWER RETRY OR UNSTABLE
```

Meaning:

BU24, 25, and 26 are issued together when a dynamic rollforward was being performed for the updates made by a specified TDS on a specified user file and an incident occurred.

Action:

- Either stop the back-up procedure with RETRY and try dynamic rollforward later.
- Or continue the back-up with UNSTABLE leaving the user file unstable. Perform RESTORE and STATIC ROLLFORWARD for recovery.

BU27 `FILES ROLLBACK PROCESSING STOPPED FOR STEP: lm-nm`

Meaning:

This message indicates an irregularity in the Before Journal; it is followed by a further explanatory message.

Action: None

```
BU28  hh.mm .BUFFER ACQUISITION PROBLEM                       (01)
      hh.mm .SEMAPHORE ACQUISITION PROBLEM                    (02)
      hh.mm .READ FICB PROBLEM                                (03)
      hh.mm .JOURNAL TRACK ADDRESS ERROR                      (04)
      hh.mm .CHANNEL PROGRAM BUILD ERROR                      (05)
      hh.mm .BAD DIALOG WITH TDS                              (06)
      hh.mm .JCT CONSTRUCTION ERROR                           (07)
      hh.mm .READ ERROR ON JOURNAL                            (08)
      hh.mm .EXTENT OPEN DIFFICULTY                           (09)
      hh.mm .ABNORMAL USER FILE ASSIGN                        (10)
      hh.mm .BY REQUEST BY OPERATOR                           (11)
      hh.mm .ABNORMAL GLOBAL OPEN                             (12)
      hh.mm .NO EOF ON THE BEFORE JOURNAL FILE                (13)
      hh.mm .FD CREATION IMPOSSIBLE USER FILE IDENTIFICATION (14)
      hh.mm .SYS.JRNAL CANNOT BE READ                         (15)
      hh.mm .REPORT SEGMENT ERROR                             (16)
      hh.mm .FILE SHARED SEGMENT ERROR                        (17)
      hh.mm .SYS.JRNAL FILE NOT READABLE                      (18)
```

Meaning:

BU28 systematically follows BU27 to indicate the reason why the rollback processing was interrupted. The information given by this message is software support oriented. The rollback was stopped for the step named in BU27, but the rollback will be executed for all the other steps. Messages are self-explanatory.

Action: None

* **BU29** `USER FILES DESCRIPTIONS NOT ACCESSIBLE`

Meaning:

TCRF has failed in its attempt to read the SYS.KNODET of the failed site. This message is preceded by message BU45 or BU50.

Action:

No possible action on the back-up system. Before restarting on the failed system, system file SYS.KNODET must be rebuilt.

```
* BU30  PUSH INIT BUTTON OF INTERRUPTED SYSTEM OTHERWISE,
        FREE SEIZED SHARED DEVICES SETTING THEM POWER OFF
```

Meaning:

This is a reminder to the operator to release disk drives which have been left in the locked state by the failed system.

Action:

The operator must reload the failed system in order to remove locks set by the controllers. If the system cannot be reloaded (and only in this case), set the shared devices in POWER OFF state in order to free them.

```
BU31  ACTIVE STEPS DESCRIPTIONS NOT ACCESSIBLE
```

Meaning:

BU31 indicates that it is impossible to access the descriptions of the active steps on the SYS.PVMF volume of the failed system. The TCRF job stops. This message is preceded by message BU45.

Action: Restart the failed system.

```
BU36  JOURNALIZATION PROCEDURES NOT USED BY ACTIVE STEPS
```

Meaning:

None of the steps that were active on the failed system has requested journalization of its files.

Action: None

```
BU37  tdsnm TDS SUBSYSTEM IS READY FOR REACTIVATION
            (AFTER JOB COMPLETION)
```

Meaning:

BU37 indicates that application tdsnm can be restarted.

Action:

The operator must wait for the completion of the TCRF job before restarting the TDS tdsnm. The SMs (sharable module) must have been loaded in the BKST of the *backup* system.

```
BU38  AFTER JOURNAL PROCESSING ABNORMALLY TERMINATED
      FILES RECOVERY NOT GUARANTEED FOR STEP lm-nm
```

Meaning:

BU38 indicates that an error occurred during the processing of After Journal. If the step concerns files created under TDS, message BU44 will be sent to the operator.

Action:

After the TCRF job has completed, the user may check the following to decide if step lm-nm can be restarted:

- TCRF report (including report of the JAGEN step),
- the PRLOG,
- the SYS.JA and SYS.JADIR of the failed system.

```
BU39  LACK OF PERIPHERAL RESOURCES;
      FILES ROLLBACKING NOT PERFORMED AT ALL
```

Meaning:

As some media was not available, the rollback could not be performed on all the files. When consulted, the operator chose not to perform it at all. He gets confirmation of his decision with this message.

Action: Restart the failed system.

```
BU40  tdsnm TDS SUBSYSTEM IS NOT READY FOR RERUN
```

Meaning:

Dialog between the operator and the Before Journal reveal that it is impossible to obtain the devices required to support the files used by application tdsnm. The specified application cannot be restarted.

Action: Restart the failed system.

```
BU41  FILES STATUS DO NOT IMPLY ROLLBACK PROCESSING
```

Meaning:

No record was modified by the FRU during which the crash happened. There is no Before Image to apply.

Action: None

```
BU42  ROLLBACKING IMPOSSIBLE, JOB ABORTED
```

Meaning:

The reading and/or updating of the Before Journal of the failed system is incomplete. The TCRF job aborted. This message is preceded by message BU45.

Action: Restart the failed system.

```
BU43  FILES ROLLBACKING PARTIALLY PROCESSED:
      LACK OF PERIPHERAL RESOURCES
```

Meaning:

As some media was not available, the rollback could not be performed on all the files. When consulted, the operator chose to perform it partially, that is to say on the available files only. He gets confirmation of his decision with this message.

Action: Run the ROLLFWD utility on all the files which have remained unstable.

```
BU44  tdsname TDS SUBSYSTEM PARTIALLY READY
              REACTIVATION QUESTIONABLE
```

Meaning:

BU44 follows answer YES to BU04.

Only partial rollback has been performed. TDS files may not be stable.

Action:

Recover the TDS file and run the ROLLFWD utility on all the user files which have remained unstable.

```
BU45  BACK UP PROCESSING ERROR:
      RC = cccccccc ---> siu, return code
```

Meaning:

A fatal incident has occurred during the processing of TCRF; the TCRF job is aborted. This message is always followed by another message indicating the type of error.

RC = cccccccc --> siu, return code provides coded information on error conditions and the system integration unit from which the error originated. Return codes are described in the *Error Messages and Return Codes Manual*.

Action:

Operator action depends on the message which follows message BU45.

```
* BU46  SYS.CATALOG NOT OPERATIONAL ON BACK-UP SYSTEM,
        UPDATING OF AFTER JOURNAL CANNOT BE PERFORMED
```

Meaning:

SYS.CATALOG is not present on the back-up site; consequently, it is not possible to copy the information relating to the After Journal of the failed site onto the back-up system. This message is followed by message BU48. The TCRF job is aborted.

Action:

After the SYS.CATALOG has been created on the back-up system, a Cold or Clean Restart must be performed to make SYS.CATALOG known to GCOS; then the TCRF job can be run again.

```
BU47  AFTER JOURNAL IS PRESENTLY ACTIVE
```

Meaning:

BU47 indicates that the back-up system has an After Journal which is utilized during the TCRF job run. It is followed by a display of message BU48.

Action:

The operator can either decide to wait for the completion of all jobs journalizing on the After Journal of the back-up system or cancel them, at which point the TCRF job can be run again.

```
BU48  THEREFORE INTEGRITY OF FILES IS NOT GUARANTEED; JOB ABORTED
```

Meaning:

BU48 indicates that the basic requirements for the processing of TCRF are not satisfied; consequently, the job will abort. It is always preceded by message BU46 or BU47.

Action: See message BU46 or BU47.

```
BU49  CONTROLS ARE SUCCESSFUL.
      SYS.CATALOG FOUND ON: volnm
      BACK-UP AFTER JOURNAL NOT PRESENTLY IN USE
```

Meaning:

The basic requirements for the processing of TCRF are satisfied.

Action: None

```
BU50  SYSTEM I/O ERROR WHILE ACCESSING KNODET FILE
      CHHR = disk address
```

Meaning:

An incident has occurred while TCRF was retrieving the user’s file descriptions from the system disk of the failed system. The TCRF job will abort.

Action: See message BU29.

```
BU51  BACK UP REQUESTED OPTION IS: option
```

Meaning:

BU51 is systematically issued when TCRF is run to indicate and confirm the options chosen by the operator: option=SYSTEM DISK, (YES, NO), RECOVERY (INFO, TDS, ALL).

SYSTEM DISK and RECOVERY are parameters given by the operator. YES or NO are set by the JCL sequence, YES corresponding to the TDS and ALL options, and NO to the INFO option.

Action: None

```
BU52  hh.mm .ROLLBACK PERFORMED. NUMBER OF ROLLED BACK
             BLOCKS : number                                (01)
      hh.mm .ROLLBACK NOT PERFORMED. NUMBER OF ROLLED BACK
             BLOCKS : number                                (02)
```

Meaning:

BU52 is systematically issued in the File Recovery Report for each file, preceded by the file name and its media device and class, in order to indicate that:

(01)	The rollback has been performed for this file.
(-2)	The rollback has not been performed for this file.

For *(02)* BU52 is followed by BU53.

Action: None

```
BU53  hh.mm ***** INCIDENT : ERROR DURING OPEN              (01)
      hh.mm ***** INCIDENT : WRITE ERROR ON USER FILE       (02)
      hh.mm ***** INCIDENT : READ ERROR ON JOURNAL          (03)
      hh.mm ***** INCIDENT : CHANNEL PROGRAM WRITE ERROR    (04)
      hh.mm ***** INCIDENT : CANCEL REQUEST DURING OPEN     (05)
      hh.mm ***** INCIDENT : FILE NOT ASSIGNED              (06)
      hh.mm ***** INCIDENT : I/O ERROR ON FICB              (07)
      hh.mm ***** INCIDENT : FD CREATION IMPOSSIBLE         (08)
      hh.mm ***** INCIDENT : ERROR DURING CLOSE             (09)
```

Meaning:

In the case the rollback was not performed for a file, BU53 follows BU52 and gives the cause of the failure. The information given by this message is software support oriented. Messages are self-explanatory.

Action: None

BU54 `HA FUNCTIONS HAVE BEEN PURCHASED,`
`TCRF AND HA FUNCTIONS ARE UNCOMPATIBLE.`

Meaning:

The takeover of HA applications on the coupled system is automatic and does not need TCRF being executed.

Action: Check that the TDSs are declared as *WATCHED BY CMSC* HA-type TDS.

BU80 `ABNORMAL RETURN CODE FROM PRIMITIVE: primnm gr4`

Meaning:

The call to the service *primnm* was not performed normally: the code *gr4* is returned by the service. None of the codes returned in the case of this message provoke the abort of the application. Some of them come with the *warning* indication.

Action: None

BU81 `THE RUNNING SYSTEM SITE NUMBER (=cpu_sysb) DOES NOT`
`MATCH THE RUNNING SYS.SYSTEM SITE NUMBER (=cpu_nat)`

Meaning:

The site number does not match the sys.system number on the Backup machine. This is a warning message, the execution of TCRF on this machine goes on.

Action: None

BU82 `YOU ARE NOT ALLOWED TO RUN TCRF.`

Meaning:

TCRF has not been bought for the machine on which it is started. The right to execute it is denied. The TCRF job is terminated. The message is preceded by the indication FATAL.

Action: None

BU83 `UNALLOWED CHARACTER STRING IN THE FOURTH FIELD OF THE`
`OPTION STRING.`

Meaning:

The parameters passed in the option string, by means of the JCL sequence, are not correct. The job TCRF is terminated. The message is preceded by the indication FATAL.

Action: The operator may start TCRF again with correct parameters.

BU84 `ABNORMAL RETURN CODE FROM PRIMITIVE: primnm gr4`
`EFN=SYS.SYSTEM, IFN=ifn, MEDIA=volnm`

Meaning:

A primitive which opens or accesses the SYS.SYSTEM file could not be executed normally, during the initial checks. BU84 gives the name of the primitive and the abnormal return code and indicates the internal name of the SYS.SYSTEM file and the media on which it lies. The execution of TCRF goes on.

Action: None

BU85 `"SYSD" ENTRY DOES NOT EXIST IN: EFN=SYS.SYSTEM,`
`IFN=ifn, MEDIA=volnm`

Meaning:

One of the sys.sytem entries necessary for initial checks has not been found. The message contains the internal and external file name of the sys.system and the media on which it lies. The execution of TCRF goes on. This message is software support oriented.

Action: None

BU86 `SYSD PARTITION NOT FOUND. PRIMITIVE: primnm`
`EFN=SYS.SYSTEM, IFN=ifn, MEDIA=volnm`

Meaning:

The primitive which opens the SYSD partition of the sys.system could not be executed normally. The name of the primitive is given as well as the internal and external file name of the sys.system and the media on which it lies. The execution of TCRF goes on. This message is software support oriented.

Action: None

BU88 `THE MAIN SITE NUMBER ASSOCIATED WITH THE "TCRF" MI`
`(=cpu_main) DOES NOT MATCH THE FOREIGN SYS.SYSTEM SITE`
`NUMBER (=cpu_for).`

Meaning:

The CPU number of the backup system does not match the CPU number associated with the TCRF application the customer bought. This message is a warning. The execution of TCRF goes on.

Action: None

BU89 `ABNORMAL RETURN CODE FROM PRIMITIVE: primnm gr4`
`EFN=SYS.SYSTEM, IFN=ifn`

Meaning:

A primitive which works with SYS.SYSTEM could not be executed normally. The message gives the name and return code of this primitive, and the external and internal file names of SYS.SYSTEM.

Action: None

BU93 `IMPOSSIBLE TO RUN TCRF ON A SYSTEM WITH PRIVATE JAS`

Meaning:

There is an attempt to run TCRF while the backup system is installed with a Private JAS.

Action:

After having saved user files protected by the private JAS, and system After Journal files, delete the two private JAS before repeating TCRF. Create the two private JAS again after TCRF and restore all system After Journal files.

❑

Glossary

A

Active State

The state applicable to:
- either the server which implements the *effective* function of a service,
- or the member which supports the ACTIVE server of a service, in which case the member becomes ACTIVE for this service.

Active Journal Files

After journal files in which After Images are logged during the session to rollforward user files. They include Primary as well as Secondary Journal files.

After Image

An *after image* is a copy of a record of a file. It contains the data in its state after it was updated. An after image is date and time stamped (with the date and time of update). After images are used in *rollforward*. Rollforward assumes that the date and time stamps are correct. In predating, this may not be the case.

Available Journal Files

After journal files which are eligible for journalization or for transfer but which are not currently used during the session.

B

Backup State
The state applicable to:
- either the server which implements the *rescue* function of a service,
- or the member which supports the BACKUP server of a service, in which case the member becomes BACKUP for this service.

Before Image
A *before image* is a copy of a C.I. of a file. It contains the data in its state before it is updated. Before images are stored sequentially in journal file(s). Since the sequential order also represents a chronological order, before images do not need a date and time stamp. Before images are used in *rollback*.

C

Commitment Unit (CU)
A unit of processing where the decision is taken to consolidate modifications and apply them to records in protected files. In the case of rollforward, the action of commitment means that Before Images are released and After Images are validated.

Complex
A set of two DPS 7000s supporting a single CMSC. Each of the DPS 7000s in the complex is known as a *member*. Each member has access to a set of *shared disks*. A complex cannot have more than two members.

CMSC (Complex Management Service)
A group of administrative functions which operate and supervise a complex, and monitors its functions.

CMSR (Complex Management Server)
A server which implements *CMSC* on a *member* of a *complex*. A complex has a single CMSC which is supported by two CMSRs.

CRCXGEN (CReate CompleX GENeration)
A processor to which directives are submitted for describing and generating the environment of a Complex.

Current Journal File
After Journal file currently being used for journalization.

D

Deferred Recovery
An operation which brings protected user files back to a consistent state, after a physical damage or an unsuccessful immediate recovery. It consists of restoring the user files from their saved versions, and then, of running the ROLLFWD utility on these restored files.

Deferred Update
A mode of processing user files which consists of ***not*** applying updates as long as the TDS commitment unit is not committed.

DUMPJRNL (Dump Journal)
A function which allows extracting TDS user journal data from the After Journal files in sequential files.

Dynamic Rollforward
An immediate Recovery operation which consists of applying the *after images* not yet applied, on files accessed in *deferred update* mode, for all the TDS commitments units which had committed when the incident occurred.

E

Empty State
Empty is the state of a server whose service has not been started.

F

FRU (File Recovery Unit)
An FRU is a portion of a program which takes a file or set of files from one consistent state at the beginning of its execution to another consistent state at the end of its execution. A program consists of one or several FRUs, each FRU being one of the following:
- an entire batch or IOF step,
- part of a batch or IOF step between the beginning of the step and the first checkpoint encountered,
- part of a batch or IOF step between two checkpoints,
- part of a batch or IOF steps between the last checkpoint encountered and the end of the step,
- a TDS commitment unit or IQS query.

H

HA (High Availability)
A function based on disk coupled systems reducing the unavailability time of an HA application (typically a TDS application) running on one of the systems when this system crashes.
The other system provides the means for an automatic detection of this failure, and for an automatic startup of a backup HA application. During the startup of the backup application, an immediate file recovery will occur for the journalized files.

HA-LOCK
HA-LOCK is used by *HA* to prevent two *members* from attempting a file recovery simultaneously. A member doing a *takeover* requests the HA-LOCK. If the lock is granted, it then proceeds with file recovery. It is possible that the failed member restarts and attempts another file recovery before the first file recovery has finished. HA-LOCK prevents the second file recovery from starting before the first one has finished.

I

Immediate Recovery
An operation which brings protected user files to a consistent state immediately after an incident occurred. This operation is started either at step abort, or at warm restart after a crash, or on takeover. There are two types of immediate recovery:
- Rollback,
- and dynamic rollforward.

J

JAS (Journalization Advanced Service)
A set of facilities and functions protecting and recovering user files. There are two types of JAS:

Public JAS	alias *SYS JAS*, being local and specific to a system. In an HA environment, it is a non-HA JAS and there is only one SYS JAS per system. Since *SYS JAS* cannot be used for takeover, it is not declared at CRCXGEN.
Private JAS	alias *BLUE JAS* and *GREEN JAS*. It can be used either in Private JAS mode or in HA mode by HA applications for takeover. Then both JAS are declared for the complex at CRCXGEN. Since both HA-type JAS can be declared per complex, one HA type JAS can be in active mode on a member and the other HA-type can be in active mode on the other member.

JAS Catalog

A catalog dedicated to each JAS, the name of the catalog being jasname.CATALOG. This catalog is used to catalog the following files:
- the JAS Directory,
- and the Primary and Secondary After Journal Files.

SYS JAS uses SYS.CATALOG which is permanent on the system. SYS.CATALOG is created by TAILOR at system installation.
BLUE.CATALOG and GREEN.CATALOG are private catalogs created when BLUE JAS and GREEN JAS, respectively, are created by the MAINTAIN_JAS processor. Both catalogs:
- are auto-attachable,
- and must be allocated on *Shared disks*.

JAS Directory

A directory attached to each JAS created and cataloged in the jasname.CATALOG when the JAS is created by the JAS administrator. The name of the directory is jasname.JADIR. This directory is used to store such information as:
- the identification of the steps using the JAS whether with *Before Journal* or *After Journal* or both,
- the identification of user files that are journalized,
- the physical characteristics, the identification and sequencing of *After Journal* files,
- and the identification of media that are to be used for After journalization.

BLUE.JADIR and GREEN.JADIR must be allocated on *Shared disks*.

JRU (Journal Recovery Utility)

A function which allows restoring the After Journal to an approximative logical state when it has been corrupted because:
- either a TDS was unable to supply the list of aborted commitment units,
- or the JAS directory was not accessible at system restart after a crash or at *takeover*.

M

Main Service

A service providing the immediate and essential data processing functions for the end-user such as TDS. A service used by the *Main Service* for functional support is called the *Used Service*. Starting the *Main Service* automatically starts up its associated *Used Service(s)*.

Member

A DPS 7000 host system running GCOS 7 and belonging to a *complex*. Each member has its own CMSR running on it. A complex has two members.

MNJAS (MAINTAIN_JAS)
A processor which allows administrating the resources, user files and functions of the JAS (Journalization Advanced Service). JAS Resources comprise its catalog, directory and journal files.

N

Not Watched TDS
Not WATCHED TDS is a TDS which runs without the knowledge of the CMSC. No takeover is therefore possible. Such a TDS:
- cannot be started or restarted through HA operator commands,
- is generated either before Release V6 or in V6 but not specified with *WATCHED BY CMSC* at TDS generation,
- and has its files protected only by SYS JAS.

P

Primary Journal File
An After Journal file in which After Images are actively logged during the session. See Secondary Journal file.

R

Recovery Step
A step which performs the *immediate recovery* of a file during *takeover* for one HA-type JAS that was switchable and was ordered to switch from *backup* to *active* state.

Recycling, also **Cycling**
An operation which sets the status of the obsolete active journal files to available.

Rollback
An operation which involves cancelling all modifications applied to protected files by FRUs which have aborted. This is done by applying *Before Images* which correspond to the aborted FRU to the protected files.

Rollforward
An operation which involves applying After Images logged in active journal files to a user file to return it to a consistent state. See *Dynamic Rollforward* and *Static Rollforward*.

S

Secondary Journal File
An After Journal file transferred from a Primary Journal file, through the MNJAS TRANSFER_PRIMARY command. See Primary Journal file.

Server
The result of projecting a *service* on a *member*. A server is a DPS 7000 job supporting a service.

Service
The implementation of a function on a *complex*. From an HA viewpoint, a service can be either a *main service* or a *used service*.

Static Rollforward
An operation which involves applying *After Images* corresponding to committed FRUs to restored user files.

Switchable State
One of the alternative states of an *active service*, the other being unswitchable.
The switchable/unswitchable attribute is propagated from the service to the *member* and to the *complex*. So at any one time, the complex is switchable or unswitchable. If the complex is unswitchable, a *takeover* is not possible.
For a *main service* to be switchable, all its *used services* must also be switchable. For example, if a TDS application (main service) uses a JAS (used service), then for the TDS application to be switchable, the JAS must also be switchable.
An HA-type JAS is switchable if a current mode HA-type TDS is running on the complex (which uses the HA-type JAS). Otherwise it is unswitchable.

T

Takeover
An operation which involves switching a *member* from the *backup* state to the *active* state.
Takeover is initiated:
- either when the active member crashes,
- or through a CMSC TAKEOVER_MEMBER command.

TCRF (Transactional Context Recovery Facility)
A utility which allows a TDS to be restarted on a coupled system (or an alternate system) without disruption to data integrity when the system on which TDS was running, is no longer available.

U

Used Service
A service intended to provide functions needed by *main services*. A used service cannot use a main service and it cannot use another used service. A typical used service is a ***JAS*** supporting a TDS application which is the main service.

User Journal
TDS user private data which is stored in the After Journal File and which can be retrieved with the DUMPJRNL utility.

W

Watched TDS
A collective name for *WATCHED HA TDS* and *WATCHED non-HA TDS*.
WATCHED HA TDS:
A WATCHED HA TDS runs with the knowledge of *HA*. Such a TDS is:
- monitored by the *CMSC* (*takeover* is possible under the appropriate conditions),
  started and stopped through HA operator commands,
- and restarted automatically by the CMSC on the *member* doing the *takeover*.

If an installation has HA, such TDSs get the full benefit of HA. In principle, all existing TDSs should be converted to current mode HA.
A current mode HA TDS is a TDS which has been:
- specified as *WATCHED BY CMSC* at TDS generation,
- and declared on <u>both</u> members of the complex.

WATCHED non-HA TDS:
A WATCHED non-HA TDS runs with the knowledge of *HA*. Such a TDS is:
- monitored by the *CMSC* but no *takeover* is possible,
- started and stopped through HA operator commands,
- and, after a crash, restarted automatically by the CMSC when the failed *member* is restarted.

A *WATCHED non-HA TDS* is a TDS which has been:
- specified as *WATCHED BY CMSC* at TDS generation,
- but declared on <u>only one</u> member of the complex.

Index

Vos remarques sur ce document / Technical publications remarks form

Titre / Title :	**File Recovery Facilities User's Guide**

N° Référence / Reference No. :	**47 A2 37UF Rev05**	**Date / Dated :**	**September 1999**

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront attentivement examinées. Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.
Your comments will be promptly investigated by qualified personnel and action will be taken as required. If you require a written reply, furnish your complete mailing address below.

NOM / NAME : ________________________________ DATE : ____________________

SOCIETE / COMPANY : ________________________________

ADRESSE / ADDRESS : ________________________________

____

Remettez cet imprimé à un responsable Bull S.A. ou envoyez-le directement à :
Please give this technical publications remarks form to your Bull S.A. representative or mail to:

Bull S.A.
CEDOC
Atelier de reprographie
34, rue du Nid de Pie BP 428
49004 ANGERS Cedex 01
FRANCE

Bull HN Information Systems Inc.
Publication Order Entry
FAX: (978) 294-7411
MA30/865A
Technology Park
Billerica, MA 01821
U.S.A.