Software

Subject: This manual explains how the General Access Control (GAC-EXTENDED) system may be used on DPS7000 systems. The purpose of this facility is to make file sharing more efficient by reducing conflicts while maintaining fast access time.

Special instructions: This Revision 2 is valid for all users of GCOS7 Release V6. Revision 1 remains valid for earlier releases.

Software supported: GCOS7 - Release V6

Date: March 1991
Copyright © Bull S.A., 1991

Bull acknowledges the rights of proprietors of trademarks mentioned herein.

Suggestions and criticisms concerning the form, content, and presentation of this manual are invited. A form is provided at the end of this manual for this purpose.

Bull disclaims the implied warranties of merchantability and fitness for a particular purpose and makes no express warranties except as may be stated in its written agreement with and for its customer. In no event is Bull liable to anyone for any indirect, special, or consequential damages.

The information and specifications in this document are subject to change without notice. Consult your Bull Marketing Representative for product or service availability.
Preface

OBJECTIVES

This manual describes how the General Access Control facility (GAC-EXTENDED) may be used on DPS7000 systems. The purpose of this facility is to make file sharing more efficient by reducing conflicts while maintaining fast response times. The general aspects of the GCOS7 file sharing and locking mechanisms are also described.

INTENDED READERSHIP

The manual is intended for all users of GAC-EXTENDED.
ASSOCIATED DOCUMENTS

General, Management, Operation:

System Overview ........................................................................................................... 47 A2 04UG

IOF Terminal User’s Reference Manual
  Part 1, Introduction to IOF ......................................................................................... 47 A2 21UJ
  Part 2, GCL Commands VBO ................................................................................... 47 A2 22UJ
  Part 2, GCL Commands FBO .................................................................................... 47 A2 23UJ
  Part 3, Directives & General Processor Commands ................................................... 47 A2 24UJ
  Part 4, Appendices .................................................................................................... 47 A2 25UJ

Messages and Return Codes Directory ........................................................................ 47 A2 10UJ

JCL Reference Manual ................................................................................................. 47 A2 11UJ

System Installation Configuration and Updating Guide .............................................. 47 A2 17US

IQS V4.0

IQS-V4 Software Release Bulletin ................................................................................ 47 A2 70UR
IQS Advanced User’s Guide .......................................................................................... 47 A2 76UD
IQS Reference Manual, Volume 1 ................................................................................ 47 A2 77UD
IQS Reference Manual, Volume 2 ................................................................................ 47 A2 78UD
IQS/TDS User’s Guide .................................................................................................. 47 A2 81UD
Preface

IQS V4.1

IQS-V4 Advanced User's Guide ................................................................. 47 A2 76UR
IQS-V4 Reference Manual, Volume 1 ...................................................... 47 A2 77UR
IQS-V4 Reference Manual, Volume 2 ...................................................... 47 A2 78UR
IQS-V4/TDS User's Guide ....................................................................... 47 A2 81UR

TDS:

TDS Concepts ........................................................................................... 47 A2 20UT
TDS Administrator's Guide ................................................................. 47 A2 21UT

IDS/II:

IDS/II Reference Manual .......................................................................... 47 A2 11UD

File and Data Management:

UFAS-EXTENDED User's Guide .............................................................. 47 A2 04UF
Data Management Utilities User's Guide .............................................. 47 A2 26UF
Catalog Management User's Guide ...................................................... 47 A2 15UF

GPL:

GPL Reference Manual ........................................................................... 47 A2 35UL
GPL System Primitives Volume 1 .......................................................... 47 A2 38UL
GPL System Primitives Volume 2 .......................................................... 47 A2 39UL
STRUCTURE OF THIS MANUAL

This manual has four sections and three appendices.

Section 1 is a general introduction, which demonstrates the advantages of using GAC-EXTENDED to control concurrent access to resources.

Section 2 provides a full description of the GCOS7 file sharing and locking mechanisms, including detailed definitions of Commitment Units and Control Intervals.

Section 3 contains all the information necessary for users of Batch, IOF (IQS) and TDS to make the best use of GAC-EXTENDED.

Section 4 describes how GAC-EXTENDED handles conflicts and deadlocks. This section may be omitted by most users of GAC-EXTENDED, but it may be interesting reading for those users who would like to know more about how GAC-EXTENDED works internally.

Appendix A contains a list of the return codes associated with GAC-EXTENDED plus a short description of the cause of the error.

Appendix B lists the most common statistical and error messages which GAC-EXTENDED may output in order of message number.

Appendix C describes the GAC statement of the CONFIG utility which specifies certain parameters used by GAC-EXTENDED.

NOTATION CONVENTIONS

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

UPPERCASE The keyword item must be specified exactly as shown. The upper case is merely a convention; in practice you can specify the item in upper or lower case.

lowercase Indicates a user-supplied parameter value.

[item] An item within square brackets is optional.

{item | item | item} Braces indicate of choice of possible values only one of which may be selected. The default value (if any) is underlined.

( ) Parentheses indicate that a single value or a list of values can be specified.

... An ellipsis indicates that the preceding item may be repeated one or more times.
### Table of Contents

1. Introduction to GAC-EXTENDED and its Facilities ........................................ 1-1
   1.1 GENERAL INTRODUCTION ........................................................................ 1-1
   1.2 GAC-EXTENDED AND UFAS-EXTENDED .................................................. 1-2
   1.3 FILE SHARING ............................................................................................ 1-3
      1.3.1 File Sharing Without GAC-EXTENDED ............................................. 1-3
      1.3.2 File Sharing With GAC-EXTENDED .................................................. 1-3
   1.4 COMMITMENT UNITS .................................................................................. 1-5
      1.4.1 Concurrent Access ............................................................................... 1-6
   1.5 MULTIPLE UPDATES .................................................................................. 1-7
   1.6 SUMMARY OF THE ADVANTAGES OF GAC-EXTENDED .......................... 1-8

2. File Sharing Concepts .................................................................................... 2-1
   2.1 TYPES OF FILE SHARING UNDER GCOS7 ........................................ 2-1
      2.1.1 The SHARE Parameter ........................................................................ 2-2
      2.1.2 The ACCESS Parameter ...................................................................... 2-3
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>RULES</td>
<td>2-6</td>
</tr>
<tr>
<td>2.3</td>
<td>FILE SHARING WITHIN TDS AND WITH OTHER JOBS</td>
<td>2-7</td>
</tr>
<tr>
<td>2.3.1</td>
<td>Several Readers</td>
<td>2-7</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Several Readers and One Writer</td>
<td>2-7</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Several Readers and Several Writers</td>
<td>2-9</td>
</tr>
<tr>
<td>2.4</td>
<td>ACCESS TO GAC-EXTENDED</td>
<td>2-11</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Resources Associated With Files</td>
<td>2-11</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Resources Not Associated With Files</td>
<td>2-11</td>
</tr>
<tr>
<td>2.5</td>
<td>COMMITMENT UNITS</td>
<td>2-12</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Introduction</td>
<td>2-12</td>
</tr>
<tr>
<td>2.5.2</td>
<td>The Management of Commitment Units</td>
<td>2-12</td>
</tr>
<tr>
<td>2.5.3</td>
<td>The Commitment of Modified Data</td>
<td>2-14</td>
</tr>
<tr>
<td>3.</td>
<td>The Use of GAC-EXTENDED</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1</td>
<td>INTRODUCTION</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2</td>
<td>NOTES FOR ALL USERS</td>
<td>3-2</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Lock List</td>
<td>3-2</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Use of Journals</td>
<td>3-3</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Deadlock</td>
<td>3-4</td>
</tr>
</tbody>
</table>
3.3  THE USE OF GAC-EXTENDED WITH BATCH AND IOF JOBS ................................. 3-6
  3.3.1  Introduction ........................................................................................................... 3-6
  3.3.2  JCL Required for Batch and GCL Required for IOF ............................................ 3-6
  3.3.2.1 The Recommended Use of READLOCK ............................................................... 3-7
  3.3.3  Processing the Current Record Pointers ............................................................... 3-8
  3.3.3.1 Important Notes ................................................................................................. 3-8
  3.3.4  NUMLOCK Parameter .......................................................................................... 3-9
  3.3.5  Setting Commitments ............................................................................................ 3-9
  3.3.5.1 Points to Remember .......................................................................................... 3-9
  3.3.5.2 Programming Required ..................................................................................... 3-10
  3.3.5.3 Commitment Call Syntax ................................................................................... 3-10
  3.3.6  Checkpoints Versus Commitment Calls ............................................................... 3-13
  3.3.7  Aborts and Deadlock Handling ............................................................................. 3-15
  3.3.8  Use of the Before Journal ...................................................................................... 3-15
  3.3.9  Sequential Processing of UFAS Files .................................................................... 3-16

3.4  THE USE OF GAC-EXTENDED WITH IQS .......................................................... 3-17
  3.4.1  Introduction to IQS .............................................................................................. 3-17
  3.4.2  GCL For IQS ......................................................................................................... 3-17
  3.4.2.1 The Specification of READLOCK ....................................................................... 3-18
  3.4.3  Taking Commitments in IQS ................................................................................ 3-18
  3.4.3.1 Automatic Commitments .................................................................................. 3-18
  3.4.3.2 Manual Commitments ....................................................................................... 3-19
  3.4.4  Processing Current Record Pointers ..................................................................... 3-19
  3.4.5  Lock List ............................................................................................................... 3-19
  3.4.6  Long Wait Notification .......................................................................................... 3-20
  3.4.7  Deadlock Notification in IQS ................................................................................ 3-20

3.5  THE USE OF GAC-EXTENDED WITH TDS .......................................................... 3-21
  3.5.1  Commitment Units ............................................................................................... 3-24
  3.5.2  MAXIMUM NUMBER OF LOCKED PAGES Clause .............................................. 3-24
  3.5.3  Processing Current Record Pointers ..................................................................... 3-25
  3.5.4  Long Wait Notification ........................................................................................ 3-25
## 4. Locking and Conflicts

### 4.1 LOCKING

#### 4.1.1 Introduction

#### 4.1.2 Locking Modes

- **4.1.3 Read Locks**
  - **4.1.3.1 Shared Read**
  - **4.1.3.2 Exclusive Read**
  - **4.1.3.3 Statistical Read**

#### 4.1.4 Write Lock

#### 4.1.5 Example

#### 4.1.6 Summary

### 4.2 CONFLICTS

#### 4.2.1 Locking Mode is Exclusive

#### 4.2.2 Locking Mode is Shared

#### 4.2.3 Locking Mode is Statistical

#### 4.2.4 More Complex Conflict Situations

### 4.3 DEADLOCKS

#### 4.3.1 The General Situation

#### 4.3.2 How GAC-EXTENDED Deals With Deadlocks

#### 4.3.3 Avoidance of Deadlock Using Exclusive Read

### 4.4 SUMMARY OF THE EFFECTS OF SHARING REQUESTS UNDER GAC-EXTENDED
# Appendices

## A. Return Codes

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>A-1</td>
</tr>
<tr>
<td>Format</td>
<td>A-1</td>
</tr>
<tr>
<td>LIST OF RETURN CODES</td>
<td>A-3</td>
</tr>
</tbody>
</table>

## B. Error Message Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Message Index</td>
<td>B-1</td>
</tr>
</tbody>
</table>

## C. The CONFIG Statement: GAC

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The CONFIG Statement: GAC</td>
<td>C-1</td>
</tr>
</tbody>
</table>

## Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>i-1</td>
</tr>
</tbody>
</table>
Illustrations

Figures

1-1 File Sharing With and Without GCAC-EXTENDED ................................................... 1-4
1-2 Concurrent Access Under GCAC-EXTENDED .......................................................... 1-6
1-3 Data Consistency Problems ................................................................................... 1-7
2-1 SHARE = NORMAL ................................................................................................ 2-4
2-2 SHARE = ONEWRITE .......................................................................................... 2-4
2-3 SHARE = MONITOR ............................................................................................ 2-5
2-4 File Sharing Between TDS and Batch or IOF Readers ............................................ 2-7
2-5 File Sharing Between Several Readers and One Writer .......................................... 2-8
2-6 File Sharing Among Several Readers and One TDS Writer ................................... 2-8
2-7 File Sharing Between Several TDS, IOF and Batch Readers and Writers ............... 2-9
2-8 Subdivision into Commitment Units ........................................................................ 2-13
3-1 The Effect of Commitment and Checkpoint Calls With Abort ............................... 3-14
4-1 Conflict When Locking Mode is Exclusive ............................................................. 4-7
4-2 Conflict When Locking Mode is Shared ................................................................. 4-8
4-3 No Conflict - Locking Mode is Statistical ............................................................. 4-8
4-4 How Conflicts are Handled by GCAC-EXTENDED ................................................ 4-9
4-5 Consistency of Data .............................................................................................. 4-10
4-6 How GCAC-EXTENDED Resolves a Deadlock Situation ...................................... 4-12
4-7 Avoiding Deadlock Using Exclusive Read ............................................................ 4-14

Tables

3-1 Before Journal Requirements in Relation to PMD and File Access ......................... 3-4
3-2 Rules of Locking Protocol Compatibility for TDS ................................................... 3-22
3-3 The Restrictions on the Use of GCAC-EXTENDED Imposed by the SHARE and ACCESS Parameters ........................................................... 3-23
4-1 Definition of Read and Write Access .................................................................... 4-6
4-2 Effects of Requests to Share Files Under GCAC-EXTENDED .............................. 4-15
A-1 GCAC-EXTENDED Internal Function Numbers ................................................... A-2
1. Introduction to GAC-EXTENDED and its Facilities

1.1 GENERAL INTRODUCTION

The General Access Control Facility for Large Systems (GAC-EXTENDED) is a resource sharing facility running under GCOS7 Releases starting from V3A7 and V3B7. It allows several users in any of the three processing environments, Batch, IOF (IQS), and TDS, to access the same resources concurrently. The main function of GAC-EXTENDED is to ensure the consistency and integrity of each user's data by resolving any conflicts which may occur, while maintaining a fast response time.

GAC-EXTENDED is easy to use from any processing environment with a minimum of programming effort. No restrictions are imposed on concurrent access from different processing environments, including access by users under different TDS jobs.

GAC-EXTENDED may be used to access UFAS files (all organizations), and IDS/II areas. Access control for IDS/II areas may be specified using IDS/II DML statements; refer to the IDS/II Reference Manual. Note however that GAC-EXTENDED may not be used to regulate the sharing of BFAS files.

GAC-EXTENDED may be accessed from COBOL, FORTRAN, and GPL programs.

Note that GAC-EXTENDED cannot control a file in both parts of a coupled system concurrently.
1.2 GAC-EXTENDED AND UFAS-EXTENDED

GAC-EXTENDED was created to take advantage of the possibilities offered by the UFAS-EXTENDED access method. UFAS-EXTENDED is described in the manual *UFAS-EXTENDED User's Guide*. Refer to this manual for all problems of migration and co-existence with UFAS-EXTENDED files.

For large transactional applications, the UFAS-EXTENDED/GAC-EXTENDED access methods and the CI Manager provide configuration limits with the support of:

- a large number of files,
- a large number of buffers,
- a large sized buffer pool.

These important functional enhancements have been complemented by a significant improvement in the overall reliability of the UFAS-EXTENDED and GAC-EXTENDED products, by way of improved error recovery mechanisms and the dynamic reinitialization of damaged control structures wherever possible.

Moreover, the format of UFAS files has been modified in the direction of FBO (Fixed Block Organization). The previous format is still fully supported for reasons of compatibility.

The GAC-EXTENDED product for V6 has been enhanced in several ways compared with the original GAC-EXTENDED product introduced with V3A7 and V3B7.

- The specification of NUMLOCK for Batch and IOF, and MAXIMUM NUMBER OF LOCKED PAGES for TDS, (which allowed the user to determine the maximum number of locks to be effected for a Commitment Unit), is still accepted for compatibility reasons, but is no longer taken into account. This treatment has been replaced by an alternative regulation mechanism handled internally by GAC.

- In TDS mode, all locks are now released after a long duration wait (LONGWAIT), and the aborted CU is restarted in DIE_WAIT mode, as is the case after a deadlock (see Subsection 4.3).

- The internal interfaces between GAC and other system components (for example, TDS, UFAS, IDS/II) have been simplified to increase overall system efficiency.

- The internal locking mechanisms used by GAC to control accesses to the GAC tables have been modified to permit a greater number of simultaneous accesses.
1.3 FILE SHARING

1.3.1 File Sharing Without GAC-EXTENDED

When GAC-EXTENDED is not used for file sharing, the unit of access is the whole file. You are therefore granted or denied access to the whole file. This means that if a user requests exclusive access to a single record in a file, once this access is granted, other users may be prevented from accessing any record in the file.

1.3.2 File Sharing With GAC-EXTENDED

Although file-level sharing may be quite acceptable on small applications and on many systems of limited size, it may become an excessive constraint on large and busy applications, particularly for files that need to be shared between Batch, IOF (especially IQS), and TDS jobs. If this is the case, it is probably worth the extra cost in terms of to use GAC-EXTENDED to regulate file sharing.

When file sharing is controlled by GAC-EXTENDED, several users in the different processing environments may read and write to the same file concurrently. Sharing is controlled within the same unit of file transfer, known as a Control Interval (CI). In other words, access conflicts may arise under GAC-EXTENDED only where several users wish to access the same CI concurrently.

Figure 1-1 compares in terms of waiting time the file sharing situation with and without GAC-EXTENDED.
CLASSIC FILE SHARING SITUATION (WITHOUT GAC-EXTENDED)

USER 1 ← process file → USER 1
USER 2 ← wait → USER 2
USER 3 ← wait → USER 3

FILE SHARING CONTROLLED BY GAC-EXTENDED, WITH CONTROL INTERVALS IN COMMON

USER 1 ← process file → USER 1
USER 2 ← process file → wait → continue → USER 2
USER 3 ← process file → USER 3

FILE SHARING CONTROLLED BY GAC-EXTENDED, NO COMMON CONTROL INTERVALS

USER 1 ← process file → USER 1
USER 2 ← process file → USER 2
USER 3 ← process file → USER 3

Figure 1-1. File Sharing With and Without GCAC-EXTENDED
1.4 COMMITMENT UNITS

GAC-EXTENDED ensures consistency of data by means of the Commitment Unit concept. A Commitment Unit is a set of related accesses made to the same file or to different files by a given user. While the updates are being performed, the data in the files in question is considered to be inconsistent, and cannot be accessed by any other user. When all the updates have been carried out satisfactorily, a Commitment is taken; this signals the end of the current Commitment Unit, and the data in the files involved may now be accessed by other users.

If you are working in Commitment UnitsBatch processing environment, you have full control over the specification of Commitment Units.

If you are working under Commitment UnitsIOF, you will not normally be concerned about Commitments; Commitment UnitsIQS is a special case of IOF processing. For IQS, the default mode is for Commitments to be taken automatically, at the end of each request, although you can choose to take Commitments manually.

For Commitment UnitsTDS, the scope of a Commitment Unit may be determined automatically by TDS or you may specify it yourself.
1.4.1 Concurrent Access

Figure 1-2 shows that GAC-EXTENDED allows concurrent read and write access to a file by different types of user and limits concurrent access to a Control Interval within a file to one writer or several readers. Thus, USER A and USER B cannot both write to the Control Interval CI1 at the same time.

Figure 1-2. Concurrent Access Under GAC-EXTENDED

W : WRITE
R : read
1.5 MULTIPLE UPDATES

This subsection deals with the problem of multiple updates, which occurs when several concurrent users wish to update the same logical record and GAC-EXTENDED is not in use. The problem is this:

- Each step in turn would access the record, read it into the appropriate buffer, and modify the contents in its memory area.
- As each modified copy is produced, the last copy written would overwrite previous ones.

We can illustrate this situation by means of an example. Suppose we have two users, each of whom reads a stock count and updates it: assume that the initial stock count is 10.

```
Initial stock = 10 items

PROG 1                               PROG 2
1  READ STOCKFIL                    2  READ STOCKFIL
   STOCK = STOCK - 2                   STOCK = STOCK - 1
3  WRITE STOCK                      4  WRITE STOCK

sequence:           final value
1,2,3,4     gives   STOCK = 9
1,2,4,3     gives   STOCK = 8
1,3,2,4     gives   STOCK = 7
2,4,1,3     gives   STOCK = 7
```

*Figure 1-3. Data Consistency Problems*

It is obvious that although, without GAC-EXTENDED, any of the four sequences is possible, only the last two sequences give the correct result.

This demonstrates why access security and data security must be guaranteed for the duration of an updating sequence; any other user must be prevented from accessing a logical record while it is being updated, and all specified operations on a logical record must be able to be done irrespective of the current activity of other users.
1.6 SUMMARY OF THE ADVANTAGES OF GAC-EXTENDED

GAC-EXTENDED offers the following advantages:

- Both file and user resources can be shared using GAC-EXTENDED.

- Conflict situations are dealt with independently of the processing environments of the users involved, because "waits" are imposed.

- GAC-EXTENDED restricts its control to the Commitment Unit, potentially the smallest logical unit of processing within a program, rather than to an entire step or transaction. This can significantly increase the speed at which files are processed. After a hardware or a software failure, processing is resumed from the beginning of the Commitment Unit where the failure occurred.

- Deadlock situations are dealt with by aborting and restarting Commitment Units automatically.

- There is a scheduling mechanism for Commitment Units to ensure satisfactory regulation of the GAC internal tables.

- There is explicit control of the processing of the current record pointers at commitment time.

- The product is very reliable due to the error recovery mechanisms which have been incorporated. There is dynamic reinitialization of damaged control structures when necessary. Thus the possibility of system crashes and aborts is significantly decreased.

- The use which GAC-EXTENDED makes of certain internal mechanisms has been improved for performance reasons.
2. File Sharing Concepts

2.1 TYPES OF FILE SHARING UNDER GCOS7

With GCOS7, several types of disk file sharing (sometimes called "sharing modes") are available. The type of sharing that is specified for a particular file affects the degree of control imposed by the system over access to that file.
2.1.1 The SHARE Parameter

The SHARE parameter is used to define the type of sharing required for a file. It may be specified:

- either in the JCL ASSIGN statement,
- or as one of the GCL file assignment parameters for non-cataloged files, or in the catalog for cataloged files.

See Subsection 2.2.

It may take the following values:

NORMAL which allows the file to be assigned to several readers OR one writer. This is the default value.

ONEWRITE allows the file to be assigned to several readers AND one writer. With this mode of sharing, the readers perform what is known as a statistical read, which means reading when they are not completely certain that the data in the file is logically consistent. However, the physical consistency of the file itself is not in doubt, and the writer may be certain that his access is valid.

MONITOR allows the file to be assigned concurrently to several readers and several writers with file access being dynamically controlled by GAC-EXTENDED. This is the most sophisticated sharing mode as it allows each user of the file to regard himself as the sole user of the file, and also ensures the consistency and integrity of data.

FREE allows the file to be assigned concurrently to several readers and several writers, but no controls are imposed. This is the most dangerous sharing mode as the consistency and integrity of data are not guaranteed. You have to establish controls yourself to avoid conflicts and ensure that the data is always valid. This mode of sharing is not allowed for TDS files, or for UFAS files.

DIR is the sharing mode which is valid only for libraries. A library sub-file (member) may be accessed by several readers or one writer.

UNSPEC is specified only in the catalog. It means that a value for the SHARE parameter is to be given when the file is assigned, as for a non-cataloged file.
2.1.2 The ACCESS Parameter

The ACCESS parameter defines the way in which the file may be processed, and may also be used to modify the sharing mode as defined by the SHARE parameter. It may be specified:

- in the JCL ASSIGN statement,
- as one of the GCL File Assignment parameters.

See Subsection 2.2.

The ACCESS parameter may take the following values:

- **WRITE** allows the processing modes INPUT, OUTPUT, APPEND, and INPUT-OUTPUT, which are specified when the file is opened. This is the default value.
- **READ** allows INPUT only.
- **SPWRITE** allows the same processing modes as WRITE, except that the current step has exclusive access to the file, regardless of the value of SHARE. Even when SHARE = MONITOR, the writing is done without the control of GAC-EXTENDED (except for TDS files).
- **SPREAD** allows, like READ, the INPUT processing mode only, but in addition the current step has exclusive access to the file, regardless of the value of SHARE. Even when SHARE = MONITOR, the reading is done without the control of GAC-EXTENDED. (This also applies to TDS files.)
- **RECOVERY** is used for obtaining exclusive access to a file for the purpose of file recovery.
- **ALLREAD** allows a number of steps to read the file concurrently regardless of the value of SHARE. The SHARE parameter need not be identical for all the steps. No step may write to the file while the other steps are reading it. ALLREAD is compatible with itself and with READ. This value may be used to avoid using GAC-EXTENDED where several steps require concurrent read of a file cataloged with SHARE = MONITOR, as the value of the SHARE parameter is ignored.

The most important combinations of the sharing and access modes for files are shown in the diagrams which follow.
Figure 2-1. SHARE = NORMAL

Figure 2-2. SHARE = ONEWRITE

Note that with SHARE = ONEWRITE, only one writer is allow
Note that with SHARE = ONEWRITE, only one writer is allowed.

Figure 2-3. SHARE = MONITOR
2.2 RULES

1. Sharing is possible only for permanent disk files, not temporary ones.

2. The default value for the SHARE parameter of a non-cataloged file is NORMAL.

3. For a cataloged file, the SHARE option should be specified in the catalog, with NORMAL as the default value. If a cataloged file has an associated assign parameter which specifies a different value for SHARE from that given in the catalog, then the catalog value will override the value supplied at assign time, and the step will be given exclusive access to the file (i.e. ACCESS = READ becomes ACCESS = SPREAD, and ACCESS = WRITE becomes ACCESS = SPWRITE). However, you are advised not to use this feature to avoid sharing a file with other steps; ACCESS = SPREAD or ACCESS = SPWRITE should be used explicitly where appropriate.

4. For several steps to share a file, they must all have the same value for the SHARE parameter (specified explicitly or not). If the file is cataloged this is generally the case, as the SHARE parameter is set up in the catalog. If the file is not cataloged, the first step to be activated which has a different value for SHARE must wait for access until the file is free again. The messages:

   DF01 Xaaa FILE yyyyy USED BY: Xbbb
   RS05 Xaaa.c WAITS FOR FILE yyyyy ON zzzz

   are output.

   Exceptions to this rule take place if:

   ACCESS = READ and SHARE = NORMAL or SHARE = ONEWRITE (for a non-cataloged file).

   ACCESS = ALLREAD, irrespective of the value of SHARE.

   Note also that although several steps may use the same value for the SHARE parameter of a file, a step having ACCESS = SPWRITE or SPREAD will have exclusive access to the file.

5. It is not possible for a step to share a UFAS file opened in OUTPUT mode with another step, because OUTPUT mode indicates creation. The second step which tries to access the file is aborted with the return code:

   DUFAS 20, SHLVVIOL

6. A file preallocated with the JCL statement ALLOCATE or the GCL File Allocation parameters ALC(i) will always be processed in exclusive mode (ACCESS = SPREAD, or implicitly, ACCESS = SPWRITE), regardless of the value for the ACCESS parameter when the file is assigned.

7. Access rights on a file may not be exceeded during execution or the action will be aborted. This means, for example, that no writing may be attempted if ACCESS = READ has been declared.
2.3 FILE SHARING WITHIN TDS AND WITH OTHER JOBS

2.3.1 Several Readers

The sharing mode shown in Figure 2-4 does not use GAC-EXTENDED to control the sharing of files between different jobs. The exclusive read lock, which is the default for TDS transactions, is processed as shared read, and a lock is not imposed on the Control Interval when a record is read.

2.3.2 Several Readers and One Writer

Figure 2-5 shows the case where several TDS readers are sharing a file with several Batch or IOF readers, and one Batch or IOF writer, while Figure 2-6 shows several TDS readers sharing with several Batch or IOF readers and one TDS writer.
In both cases the TDS readers must use the statistical read protocol as specified by the clause in the TDS generation program:

SUPPRESS CONCURRENT ACCESS CONTROL FOR file-name
File Sharing Concepts

The TDS writer may use the following read protocols:

**EXCLUSIVE** the default protocol for TDS transactions. A Control Interval is locked in EXCLUSIVE mode when a record within that Control Interval is being accessed.

**SHARED** specified by the clause file-name in the TDS generation program. This allows several users to read a Control Interval concurrently, but prevents another user from writing to a record in the Control Interval or from reading the Control Interval in exclusive mode at the same time. The Control Interval is locked in SHARED mode.

**STATISTICAL** specified by the clause SUPPRESS CONCURRENT ACCESS CONTROL for transactions which only read a file. Possible inconsistencies may arise if the file is updated concurrently by another user.

For the sharing situation shown in Figure 2-6, GAC-EXTENDED applies only to TDS users. For TDS readers it is restricted to the statistical read visibility. For the situation shown in Figure 2-7, all the GAC-EXTENDED facilities are available to all users.

### 2.3.3 Several Readers and Several Writers

![Diagram showing file sharing between TDS, IOF, and Batch readers and writers.](image)

Figure 2-7. File Sharing Between Several TDS, IOF and Batch Readers and Writers
Figure 2-7 shows a typical multiple sharing situation, where several TDS readers and writers are sharing with several Batch and IQS readers and writers. In this case, any read protocol may be used in TDS, Batch and IQS. The specification of the read protocol for TDS transactions is shown above. For Batch users, READLOCK is specified in the JCL statement DEFINE. For IOF (IQS) users, it is specified in the DEF(i) parameter group of the GCL command EXEC_PG. For Batch and IOF, shared read is the default protocol (READLOCK = NORMAL), while exclusive read and statistical read are specified for a particular file by READLOCK = EXCL and READLOCK = STAT respectively.

READLOCK = NORMAL and READLOCK = EXCL ensure full consistency of data. Writes are always exclusive, i.e. a Control Interval is locked in exclusive mode whenever a record is written to it. Integrity of data is guaranteed and all the GAC-EXTENDED facilities are available to TDS, Batch and IOF users.

READLOCK = EXCL is also recommended in the case where there is one writer and several readers as shown in Figures 2-5 and 2-6, but when the readers require the file to be logically consistent.
2.4 ACCESS TO GAC-EXTENDED

2.4.1 Resources Associated With Files

GAC-EXTENDED is activated for all UFAS files and IDS/II areas assigned with SHARE = MONITOR and for TDS-controlled files assigned with SHARE = NORMAL or SHARE = ONEWRITE.

2.4.2 Resources Not Associated With Files

Is activated at each request to lock resources which are not associated with files: data areas, TDS CONTROLLED COMMON-STORAGE etc. It is not necessary to specify anywhere that GAC-EXTENDED is to be used to control the sharing of such resources. However, these locks are reserved for processors such as TDS, and their scope is restricted to the interior of a Commitment Unit. Refer to the TDS Administrator's Guide.
2.5 COMMITMENT UNITS

2.5.1 Introduction

This subsection deals in more detail with the concepts of Commitment Units which were introduced in earlier sections of this manual. The information contained here is mainly background information, and many users of GAC-EXTENDED will not need to refer to it, but there are circumstances in which you may find it useful to have a more detailed knowledge of how GAC-EXTENDED controls file sharing.

When you assign a file with SHARE = MONITOR, you signal your intention to co-operate with other users by sharing information in the file. GAC-EXTENDED helps do this by resolving local conflicts which may occur when accessing records or pages, but you must provide some information about what you intend to do, and also respect certain rules in order to allow each user to benefit from the co-operation. You specify what you intend to do via a locking protocol and split the processing of your application into Commitment Units, thus allowing the whole system to run correctly.

2.5.2 The Management of Commitment Units

A program may be seen as a sequence of actions which consult or modify files. The program sees a file in a certain state before processing begins, and as a result of its processing, the file is in a different state when processing is complete. During program execution, the data in the file is inconsistent: some modifications may be partially complete, and some logical relations between record contents may be temporarily meaningless. But at the beginning and the end of the program, and possibly at some specific intermediate points, the file is consistent relative to the actions performed.

A is a sequence of actions which allows one file or several files to pass from an initial consistent state to a final consistent state, during which sequence the files must be considered as logically inconsistent, relative to the current processing.

Sharing control is maintained by GAC-EXTENDED over the affected Control Intervals from the beginning of a given Commitment Unit until its conclusion.

A Commitment Unit that performs actions on a file must see a consistent file or part of a file, and until the commitment is taken, none of the actions performed must be visible to other users.

If a Commitment Unit cannot terminate normally (by a commitment being taken), other users sharing the file must see the file in its initial state, as if the failed Commitment Unit had never been executed.

During the execution of a Commitment Unit, certain Control Intervals of a file may be locked, which causes programs which need to access some of those CIs to be halted until the CIs they require are freed at the end of the CU which locked them. To obtain the best performance possible, a commitment should be taken as soon as possible to free the resource for other users.
When a commitment is taken, the actions performed during the Commitment Unit are written to the files concerned, and all the locks are released. The files are then in a consistent state relative to the Commitment Unit. In other words, before the commitment is taken, all the actions performed during the CU are visible only to one user, and may be undone, (using the Before Journal), but after the commitment has been taken, the result of the actions performed is visible to all the users in the system, and the actions cannot be undone.

The definition of commitments may be explicit (Batch, TDS, IOF (IQS)), or implicit (TDS, IOF (IQS)).

A commitment consequently determines the point, when GAC-EXTENDED is in operation, at which a step may be restarted after a software or a hardware failure. This means that a Batch step must be "repeatable"; i.e., REPEAT must be present in the STEP statement associated with the step.

The specification of commitments (and therefore of Commitment Units) by the user or by the system is explained in Section 3 for each processing environment. See Figure 2-8 for an outline on how a program might be split up into Commitment Units.

(open file)
CU1   read CI a
     write CI b
     -----------------------------COMMITMENT
     read CI c
     read CI d
     read CI e
CU2
     write CI d
     write CI c
     -----------------------------COMMITMENT
     read CI a
CU3   write CI f
     write CI g
     (close file)
     -----------------------------COMMITMENT

Figure 2-8. Subdivision into Commitment Units

Note: Figure 2-8 is a simple case where a single file is accessed. In practice, several files may be accessed and modified from the same Commitment Unit.
2.5.3 **The Commitment of Modified Data**

Before discussing the controls imposed by GAC-EXTENDED on read and write access to Control Intervals in order to ensure consistency of data, we are going to consider in simplified terms what happens during a typical updating sequence.

1. **read x:** the Control Interval containing the variable x is transferred from disk to the user's buffer.

   ..............................

   ..............................

2. **x = x + 1:** the variable x is modified in the user's work area in memory.

   ..............................

   ..............................

3. **write x:** the new value of x is transferred from the memory work area to the user's buffer.

   ..............................

   ..............................

   ..............................

   ..............................

   ..............................

4. **read y:** (where y is not in the same Control Interval as x). The Control Interval containing y is transferred to the user's buffer.

In the above example, the value of variable x is modified; however, since the new value of x is not stored in the file until the user's buffer is required again, x would be considered to be an unstable data item between steps 1 and 4 above. It would be sensible to enclose stages 1 to 3 in the same Commitment Unit and take a commitment soon after stage 3.

The choice of when and how often to take a commitment is an important one. In Batch mode, a commitment involves a checkpoint which is expensive in terms of the number of I/O operations on backing store. This has to be balanced against the time-delay consequences of large Commitment Units. The ways in which delays can occur are discussed in Section 4.
3. The Use of GAC-EXTENDED

3.1 INTRODUCTION

This section describes how to use GAC-EXTENDED in the different processing environments: and.

For the Batch environment, GAC-EXTENDED is implemented via the JCL statements and DEFINE associated with the files to be shared.

For IOF, GAC-EXTENDED is implemented via the File Assignment and File Define parameters of the GCL statements associated with the files to be shared.

For TDS, the parameters relating to the use of GAC-EXTENDED are defined at TDS generation.

The rest of this section is intended to provide all the information necessary to use GAC-EXTENDED to obtain the best and most rapid results when sharing files.

You should have some knowledge of the catalog (refer to the Catalog Management User's Guide), and also of the and After Journals (refer to the File Recovery Facilities User's Guide).
3.2 NOTES FOR ALL USERS

3.2.1 Lock List

GAC-EXTENDED maintains a fixed size lock list that contains, for all files controlled by GAC-EXTENDED, details of all the Control Intervals currently locked by Commitment Units, together with the type of locking. The size of the lock list is determined at system configuration time via the LOCKSIZE parameter of the CONFIG utility. (See Appendix C.) The lock list has a maximum capacity of 19,900 entries.

The handling of the lock list is done automatically by GAC-EXTENDED. The NUMLOCK parameter for Batch, and the MAXIMUM NUMBER OF LOCKED PAGES clause for TDS, may be retained for compatibility reasons, but will in fact be ignored. Each entry in the lock list contains the Control Interval number, the type of locking applied, and the identification of the user that created the entry. When a commitment is taken, all the entries in the lock list associated with the user in question are freed.

Under normal operation, there is a 1:1 correspondence between the number of lock list entries allocated and the number of resources locked.

Each user may dynamically request as many resources as required. GAC-EXTENDED attempts to regulate the flow of incoming Commitment Units with a view to avoiding lock list overflow problems. The execution of a Commitment Unit is authorized only if GAC-EXTENDED estimates, as a function of the number of active Commitment Units and of the number of resources already allocated, that its running will not fill the lock list beyond a tolerated threshold. If an attempt is made to exceed this threshold, the Commitment Unit in question will be made to wait until GAC-EXTENDED considers that the likelihood of a lock list overflow is very small.

GAC-EXTENDED can only estimate the resulting occupation of the lock list, since all lock requests are entirely dynamic. Thus, despite the precautions taken by GAC-EXTENDED, it is still possible for an overflow to occur. If this happens, the Commitment Unit which provoked the overflow is aborted (with the return code TABOV). In addition, if any concurrent Commitment Unit has consumed a large number of resources (more than 20% of the lock list capacity), then this Commitment Unit is also aborted. The consequent release of resources should help to alleviate the situation. In addition, GAC-EXTENDED restarts "big consumers" in a special degraded mode, where there is a 1:n correspondence between the number of lock list entries allocated and the number of resources locked.
It should be noted that other Commitment Units may be aborted while waiting for the release of resources during this period.

If there are no "big consumers" at the time of a lock list overflow, GAC-EXTENDED makes successive reductions of 10% to the number of Commitment Units permitted to be simultaneously active, until the overflow situation is rectified. In this way, GAC-EXTENDED continues to function without having to impose constraints on the number of resources allocated to a given Commitment Unit.

3.2.2 Use of Journals

This subsection explains the relevance to GAC-EXTENDED of the concept of journalization. More details on how and After Journals are used may be found in the File Recovery Facilities User's Guide.

If GAC-EXTENDED is to be used to control the sharing of a file (that is, if SHARE = MONITOR, ACCESS = WRITE, and the file is opened with the processing mode UPDATE), Before journalization must be specified for the file. In the TDS environment only, the Deferred Update mechanism and the After Journal can be used instead of the Before Journal for files controlled by GAC-EXTENDED.

For the Batch and IOF environments, JOURNAL = BEFORE must be specified either in the catalog entry for the file or when the file is defined (in the JCL statement $DEFINE or in the GCL parameter group DEF(i)). For TDS the use of the Before Journal is specified in TDSGEN, see subsection 3.4.

When the processing mode for a file is INPUT, it is useless to request Before journalization. The requirements for journalization in relation to the value of the ACCESS parameter and the processing mode are shown in Table 3.1.

The Before Journal works in the following way: if Before journalization is specified for a file, each time a record is modified in a Control Interval of that file, a copy of the unmodified Control Interval is stored in the Before Journal. The modifications are also registered in the buffers. (Physical file updates are made according to the demands on the buffers.) Using the Before Journal entries, these physical modifications can be undone if an error occurs during the execution of a Commitment Unit. The CU can be restarted with the file in its original unmodified state (rolled back). The Commitment Unit concept in conjunction with the Before Journal guarantees the integrity of data. For Batch, IOF, and IQS files, Before journalization is required if locks may be requested for the file (this depends on the SHARE and ACCESS information, as described in Section 2); otherwise the step will abort with the return code LOKVIOL when an attempt is made to open the file.

The Deferred Update mechanism works in the following way: each time a record is modified, it is written after modification to an After Journal buffer. The modified records are written from the buffer to the file itself only when a commitment is taken. If an incident happens at this time, the After Journal is used to restore the file to a stable state.
### Table 3-1. Before Journal Requirements in Relation to PMD and File Access

<table>
<thead>
<tr>
<th>File Access</th>
<th>Batch or IOF</th>
<th>TDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS OPEN</td>
<td>Deferred Update + After Journal</td>
<td>No Deferred Update</td>
</tr>
<tr>
<td>SPWRITE IN NO</td>
<td>meaningless</td>
<td>NO</td>
</tr>
<tr>
<td>UP NO**</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>SPREAD IN NO</td>
<td>meaningless</td>
<td>NO</td>
</tr>
<tr>
<td>ALLREAD IN NO</td>
<td>meaningless</td>
<td>NO</td>
</tr>
<tr>
<td>WRITE* IN NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>UP YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>READ IN NO</td>
<td>meaningless</td>
<td>NO</td>
</tr>
</tbody>
</table>

YES Before Journal is required for GAC-EXTENDED.

NO Before Journal is not required for GAC-EXTENDED.

** Before Journal is not required for GAC-EXTENDED, which is not activated. However the Before Journal must be assigned to enable the file to be rolled back after an abort before restarting from a previous checkpoint.

* With GAC-EXTENDED, several TDS steps may share the same file in deferred write mode which makes this mode possible with ACCESS=WRITE.

### 3.2.3 Deadlock

Deadlock occurs when the following three conditions are all met:

- A user X is waiting for an object which is currently being accessed in exclusive mode by another user Y.
- The user Y requires an object which is currently being accessed in exclusive mode by user X.
- Each user requires access to the object being accessed by the other before releasing his own object.
**Examples:**

1. With One File

   You have already updated CI A and you require CI B. Another user has already updated CI B and requires CI A.

2. With Two Files

   You have already updated the CI1 of file A, and you now need to access the CI2 of file B. Another user has already updated the CI2 of file B and now requires access to the CI1 of file A.

   When deadlock occurs, one or more of the conflicting Commitment Units is aborted. The files involved are rolled back and the Commitment Units are restarted automatically when the situation has been resolved.
3.3 THE USE OF GAC-EXTENDED WITH BATCH AND IOF JOBS

3.3.1 Introduction

In the following paragraphs, Batch jobs include jobs in a stored job stream activated by the operator, and jobs initiated from Remote Batch or Interactive terminals. IOF jobs are job steps issued under IOF.

This subsection explains how to program a Batch or an IOF job to get the best results from GAC-EXTENDED. Since waiting time during file access is generally not crucial for Batch or IOF jobs, we are mainly concerned with situations where Batch and IOF programs are accessing files which are being used concurrently by TDS and IQS users; the aim is to prevent any significant increase in the TDS or IQS terminal response time; caused by interference from Batch jobs.

3.3.2 JCL Required for Batch and GCL Required for IOF

To use GAC-EXTENDED where the files to be shared are contained in a Batch or an IOF job, the JCL statement (or the File Assignment parameters of the EXEC_PGGCL procedure EXEC_PG) associated with each non-cataloged file must contain the following:

\[
\text{SHARE} = \text{MONITOR}, \quad \text{ACCESS} = \{\text{WRITE}, \text{READ}\}
\]

The JCL statement DEFINE, or the file definition parameters of EXEC_PG, associated with each file must have the format:

\[
\text{JOURNAL} = \{\text{BEFORE}, \text{BOTH}\} \quad \text{[READLOCK} = \{\text{NORMAL, EXCL, STAT}\}]
\]

Note that = SPREAD, SPWRITE or ALLREAD may also be specified, but GAC-EXTENDED will not be fully effective in this case (no locks will be applied to the file by the step). The same applies for files defined with READLOCK = STAT (in this case, there is a further optimization in that locks will not even be requested).
Three Operation modes of operation are to be distinguished:

- **non-GAC mode** SHARE=MONITOR has not been specified. GAC-EXTENDED is not activated for the file.

- **ineffective GAC mode** SHARE=MONITOR has been specified. GAC-EXTENDED is activated for the file, but locks are not applied.

- **fully effective GAC mode** SHARE=MONITOR has been specified. GAC-EXTENDED is activated for the file and locks are applied at each file access.

For a cataloged file the SHARE value should be stored in the catalog and the JOURNAL value may be stored in the catalog. If this is the case, they need not be specified in the JCL. READLOCK can be specified only in the DEFINE statement.

If JOURNAL = BEFORE is not specified for the file, the step will abort with the return code LOKVIOL when attempting to open it in INPUT-OUTPUT mode, if fully effective GAC mode has been specified.

The concept of journalization as it applies to the sharing of Batch files with GAC-EXTENDED is dealt with in the appropriate subsections of this section.

The use of READLOCK is described below.

The $STEP statement for the step enclosure from which the files are accessed must contain the REPEAT parameter. If this is not specified, the step will abort with the return code OPTERR on attempting to open the first file controlled by GAC-EXTENDED.

### 3.3.2.1 The Recommended Use of READLOCK

- **READLOCK = NORMAL**, which is the default value, means that any number of users may read a file concurrently, but only one user can write to the file.

- **READLOCK = STAT** allows any user to read a file, even if another user is updating it. (GAC-EXTENDED is completely bypassed.)

- **READLOCK = EXCL** means that one user can read a file or one user can write to it.
READLOCK = NORMAL is recommended for most situations. However, the value STAT may be specified for READLOCK in the following cases:

1. If you are only reading a file and are not concerned about the possibility of reading unstable data which has been modified but not committed.

2. If the inter-relationship between the records to be read need not be preserved.

The use of the exclusive read lock (READLOCK = EXCL) should be restricted to:

- Situations where deadlocks might occur frequently if a shared lock is used. See "LOCKING" in Section 4 of this manual.

- Certain situations where several related records from perhaps more than one file are to be processed, to ensure that no other user can read from any of the records until the result has been stored.

### 3.3.3 Processing the Current Record Pointers

The CURRENCY option can be specified as a parameter at commitment time. This option has the values L(OSE), (discard pointers), or K(EEP) (retain pointers). IT APPLIES TO ALL FILES OPEN WHEN THE COMMITMENT IS TAKEN. If no option is specified, the current record pointers are lost or retained according to the LOCKMARK option.

If LOCKMARK is specified for a given file, the current record pointers will be retained at the end of the Commitment Unit. If it is not specified, the current record pointers will be lost by default.

The option overrides the LOCKMARK option in all cases.

When GAC-EXTENDED is not effective in Batch or IOF, for example if SHARE = MONITOR and ACCESS = SPWRITE have been specified (see Table 3.3), and a commitment is processed as a checkpoint, then the current record pointers are retained.

### 3.3.3.1 Important Notes

Note that requesting the conservation of the current record pointers between two Commitment Units does not necessarily mean that the current record pointers are guaranteed since, between the two Commitment Units, the record associated with the current record pointers may have been modified or deleted after the lock was released. The option should therefore be used with care. This topic is explained further in the UFAS-EXTENDED User's Guide.

Note also that the KEEP option has not been implemented for IDS/I1 areas.
3.3.4 NUMLOCK Parameter

As explained earlier in this section, the NUMLOCK parameter previously used in a Batch environment is no longer taken into account.

3.3.5 Setting Commitments

3.3.5.1 Points to Remember

With a Batch step, you have full control over where to set , and the decision on how to divide up such a step into Commitment Units is a very important one. Three criteria should be borne in mind:

1. The program should be split into Commitment Units such that all read and update operations necessary to leave one file or a set of related files in a logically consistent state are performed in the same unit. The beginning of a is the point from which any restarts caused by deadlocks or system failures are made.

2. The longer the Commitment Unit, the longer the wait imposed on other users (TDS, IQS) when a conflict occurs.

3. Since a commitment involves a checkpoint, large overheads are incurred if commitments are taken too frequently.

To avoid excessive overheads in Batch programs using GAC-EXTENDED, it is recommended that the number of be restricted as far as possible. There should be not more than one call per 50 to 100 input/output operations. Moreover, if the risk of conflicts with TDS users is low, it is possible to have one call every 1000 input/output operations. Note however that if READLOCK = STAT is used for all the files of a Batch step, there is no need to worry about setting Commitment Units since other users cannot be locked out.

While a step is under the control of GAC-EXTENDED, you should note the following general recommendations on its use:

- If the program issues only reads, you may specify READLOCK = STAT if no special consistency is needed. System performance will be improved, and there will be no need to take commitments. If you specify READLOCK = NORMAL, the files will be protected against concurrent writers, but you must ensure that your Commitment Units are of a convenient length.

- If the program issues updates, the lack of division into Commitment Units may become unacceptable because too many locks may be held over too long a period. In this case, it may be better to assign the file with SHARE = NORMAL, instead of SHARE = MONITOR.
3.3.5.2 Programming Required

The first Commitment Unit starts with the first access of GAC-EXTENDED by the step (generally the first OPEN for a file controlled by GAC-EXTENDED). The first commitment call ends the first Commitment Unit and starts the second. Subsequent commitment calls divide the program into more Commitment Units. The last Commitment Unit ends at the first commitment call following the closing of the last file controlled by GAC-EXTENDED. If no commitment call is given after this file is closed, the end of the step closes the last Commitment Unit. If no commitment calls are specified, the entire step is treated as a Commitment Unit.

If an abort occurs before the first commitment call following an OPEN for a file controlled by GAC-EXTENDED, the restart takes place either from the most recent checkpoint or from the start of the step. (See 3.3.6 "Checkpoints Versus Commitment Calls.

3.3.5.3 Commitment Call Syntax

The syntax of the depends on the programming language (COBOL 74 and COBOL 85, FORTRAN 77 and GPL), but the parameters of the call are essentially the same. Examples of commitment calls in these languages are given below, together with the appropriate parameter declarations, followed by a description of the parameters.

**COBOL 74 and COBOL 85**

The following declarations are required:

```
77     o-mode     COMP-2.
77     o-ckinf    PIC X (32).
77     i-numlock  COMP-1 VALUE -1.
77     i-nochkpt   PIC X VALUE SPACE.
77     i-currency  PIC X.
```

The call has the following format:

```
CALL "H_GAC_UCOMIT" USING o-mode, o-ckinf, i-numlock, i-nochkpt, i-currency.
```
The Use of GAC-EXTENDED

FORTRAN 77

The following declarations are required:

INTEGER mode
CHARACTER ckinf
INTEGER numlock
CHARACTER nochkpt
CHARACTER currency
DATA numlock /-1/, nochkpt/' '/,

The call has the following format:

CALL H_GAC_UCOMIT (mode, ckinf, numlock, nochkpt, currency)

Note: (COBOL and FORTRAN). If five parameters are passed, then currency may be set explicitly to "L" or "K". If any other value is specified, or if only four parameters are passed, the LOCKMARK option is taken into account.

GPL

The following declarations are required:

DCL mode FIXED BIN (31);
CL ckinf CHAR (32);
DCL numlock FIXED BIN (15) INIT (-1);
DCL nochkpt CHAR (1) INIT (" ");
DCL currency CHAR (4);

The call has the following format:

$H_COMMIT MODE = mode CKINF = ckinf
[,NUMLOCK = numlock]
[,NOCHKPT = nochkpt]
[,CURRENCY = currency];

If CURRENCY is not specified, LOCKMARK setting applies. If CURRENCY is specified, only "KEEP", "LOSE" and the null string " " are accepted. The null string causes the LOCKMARK setting to apply. For an explanation of the notation used for the parameters in the above call, refer to the GPL Reference Manual.
Description of Parameters

MODE
A numeric variable which specifies whether the current execution mode is normal or restart after an incident. If the execution mode is normal, MODE is set to zero. If the execution mode is restart, MODE is set to the value of the step completion code at the time of the incident.

CKINF
This parameter is a 32-character output string. Depending on certain conditions, some of the characters in the string may be set to 1. Using [1] to indicate the leftmost character in the string, and [32] the rightmost character, then:

[6] = 1: DEBUG has been specified in the $STEP statement. Checkpoints are not taken.

[14] = 1: The checkpoint has failed. See the Job Occurrence Report.

[16] = 1: A major error has occurred. See the Job Occurrence Report.

[32] = 1: The next checkpoint will not be taken.

NUMLOCK
This parameter is no longer taken into account. It is retained only for compatibility with previous releases.

NOCHKPT
This parameter can be either blank or non-blank. If GAC-EXTENDED is fully effective for the step (there is at least one file open for which locks may be applied), it is ignored. It is meaningful only if GAC-EXTENDED is NOT fully effective for the step. In this case, when NOCHKPT is blank, commitment calls are processed as checkpoints. A commitment call for which NOCHKPT has any other value does not result in a checkpoint. This feature allows programs to be run without alteration whether the files they use are controlled by GAC-EXTENDED or not.

CURRENCY
This parameter specifies whether the current record pointers are lost or retained after a commitment has been taken. For COBOL 74, COBOL 85, and FORTRAN 77, CURRENCY is either L (lose pointers) or K (keep pointers). For GPL, CURRENCY is either LOSE (lose pointers) or KEEP (keep pointers). If CURRENCY is not specified, the default applies. See above for an explanation of the default values.
3.3.6 Checkpoints Versus Commitment Calls

When GAC-EXTENDED is fully effective for the step (locks may be applied to at least one open file), checkpoints are meaningless, and checkpoint calls are therefore ignored. Commitment calls however, are fully recognized only if GAC-EXTENDED is fully effective for the step. If GAC-EXTENDED is not fully effective for the step, commitment calls are processed as checkpoints unless NOCHKPT is not blank, when they are ignored. This situation may be summed up as follows (see Figure 3-1):

- When GAC-EXTENDED is not fully effective for the step, checkpoint calls are recognized and commitment calls are processed as checkpoints (unless NOCHKPT is not blank).

- As soon as GAC-EXTENDED becomes fully effective for the step (when a file is opened for which locks may be applied), checkpoint calls are ignored and commitment calls are fully recognized.

- After the last Commitment Unit, i.e. after the commitment call following the close of the last file for which locks may be applied using GAC-EXTENDED, checkpoint calls are recognized once again and commitment calls are processed as checkpoints (unless NOCHKPT is not blank).

Note that in Figure 3-1 the value * is given as an example for NOCHKPT not blank.
Figure 3-1. The Effect of Commitment and Checkpoint Calls With Abort
3.3.7 Aborts and Deadlock Handling

A deadlock may occur if two users are each waiting for a file which is being accessed by the other. One of the conflicting Commitment Units is aborted.

If a deadlock or a step abort occurs in a step for which GAC-EXTENDED is fully effective, the restart is made from the most recent commitment call. If there is no previous call, the restart is performed from the most recent checkpoint or from the beginning of the step.

If the abort occurs before any file has been opened with SHARE = MONITOR, the restart is made from the most recent checkpoint or from the beginning of the step if there has been no previous checkpoint.

If the abort occurs after the last Commitment Unit, the restart will be performed from the most recent checkpoint. If there is no checkpoint call after the last Commitment Unit, the restart is performed from the last commitment call.

If an abort occurs in a step which contains commitment calls but for which GAC-EXTENDED is not fully effective, the restart is performed from the most recent checkpoint call (which may be a commitment call if NOCHKPT = " "). If there is no previous call, the restart is performed from the beginning of the step.

Further information about how GAC-EXTENDED handles conflicts can be found in Section 4 of this manual.

3.3.8 Use of the Before Journal

When using the , bear in mind the following points:

1. The job step in question must be declared "repeatable" (via the REPEAT parameter) so that GAC-EXTENDED can restart the CU with the aid of the Before Journal. If REPEAT is not specified, the step will abort with the return code OPTERR when an attempt is made to open the first file for which locks may be applied by GAC-EXTENDED.

2. Because of the presence of the Before Journal, the action taken by GAC-EXTENDED after a deadlock is automatic. However, if a program error causes the step to abort, the decision whether or not to repeat the CU is made by the system operator. If a CU in a Batch step is aborted and restarted, you are notified by a message in the JOR.

3. For cataloged files the value of the JOURNAL parameter may be stored in the catalog, and that value will apply by default. If access rights are used, only the owner of the file can change the value stored in the catalog.
3.3.9 Sequential Processing of UFAS Files

When a file is being processed sequentially under GAC-EXTENDED and a commitment is taken, it is normally impossible to continue in sequence because the current record pointer is no longer valid. The following solutions are available to avoid this situation.

1. Specify the KEEP value for the CURRENCY option, or LOCKMARK at JCL level (see the paragraph of this Section entitled “Processing Current Record Pointers”); this ensures that the current record pointers are not destroyed when a commitment is taken.

2. If the file is only being read, specify the value STAT for the READLOCK parameter. In this case no explicit commitment calls will be required in the program (but logical consistency of data is not guaranteed).

3. In COBOL, use the START verb after a commitment to continue the sequential access.

Note that once a commitment has been taken all locks on the Control Intervals used are released. Therefore care must be taken when using since the current record pointer may have been changed by a concurrent Commitment Unit before the same user again makes use of the current record pointer in the following Commitment Unit. For example, before one user accesses and locks the CI following the one which was addressed by the current record pointer just before the commitment, any other user is able to access that CI. If another user modifies a record in that CI (i.e., locks the CI in exclusive mode) and you later attempt to access it, the step may be aborted with an abnormal return code. From a COBOL program it is possible to anticipate this situation, and avoid the abort by intercepting the return code. See the COBOL User’s Guide for the detection of return codes; this is in the section “USE AFTER ERROR”.

3.4 THE USE OF GAC-EXTENDED WITH IQS

3.4.1 Introduction to IQS

IQS is the main application of GAC-EXTENDED in the IOF environment. There are certain differences in the way in which GAC-EXTENDED is used under IQS compared to the way it is used under standard IOF. These differences are explained here. The main difference is that you can choose to restart a query in the case of deadlock or long wait.

IQS may also be run in the Batch and TDS environments. In Batch mode, IQS queries run in exactly the same way as any Batch program (see the previous section).

When IQS is running under TDS, GAC-EXTENDED is active for all TDS-controlled files used in an IQS/TDS session which have been assigned with SHARE = NORMAL or SHARE = ONEWRITE. Queries are restarted automatically after deadlock and long wait.

For further information about running IQS under TDS, refer to the IQS/TDS User’s Guide.

3.4.2 GCL For IQS

To use GAC-EXTENDED where the files to be shared are IQS files, do the following:

Assign each file to be controlled by GAC-EXTENDED in the GLC command EXEC_PG as follows:

```plaintext
ASGi . . . . SHARE = MONITOR, ACCESS = (WRITE|READ)
```

If ACCESS = READ, the file definition parameters may be omitted.

If ACCESS = WRITE, the following File Define Parameters should be added to EXEC_PG for each file to be updated:

```plaintext
DEFi . . . . JOURNAL = {BEFORE|BOTH}
                   [,READLOCK = {NORMAL|EXCL|STAT} ] ;
```

Note that = SPREAD, SPWRITE or ALLREAD may also be specified, but GAC-EXTENDED will not be fully effective in this case (no locks will be applied to the file by the step). The same is true for files defined with READLOCK = STAT (in this case, there is a further optimization in that locks will not even be requested).

If the file in question is a cataloged file, SHARE = MONITOR will be included in the catalog entry, and JOURNAL = BEFORE may be included in the catalog entry.
3.4.2.1 The Specification of READLOCK

READLOCK = NORMAL, which is the default value, means that any number of users may read a file concurrently, but writing to a file is done in exclusive mode. This should be the most commonly used value for READLOCK.

READLOCK = STAT allows any user to read a file, even if another user is updating it. This value may be specified if you are only reading a file and are not concerned about the possibility of reading data which has been modified but not committed by another user.

READLOCK = EXCL, which means that both read and write operations are performed in exclusive mode, should only be used to avoid deadlocks (see "LOCKING", in Section 4), or when updating several related records from more than one file, to ensure that no other user can read the data until all the records have been updated.

3.4.3 Taking Commitments in IQS

Can be taken in one of two ways: automatic or manual.

3.4.3.1 Automatic Commitments

By default, commitments are taken automatically. That is, the AUTOCOMMIT ON command has been set by the system. In automatic mode, a commitment is taken:

- when an OPEN command is issued,
- when a CLOSE command is issued,
- when a TERM command is issued,
- when a / request is used to exit from a REVIEW UPDATE session or from a CHANGE command,
- at the end of the execution of a query.

Of course, a commitment is also taken when a COMMIT or ROLLBACK statement is executed within a query.
3.4.3.2 Manual Commitments

If manual control of commitments is required, the AUTOCOMMIT OFF command must be set. In manual mode, a commitment is still taken automatically in certain cases:

- when an OPEN command is issued,
- when a CLOSE command is issued,
- when a TERM command is issued.

In addition, a COMMIT or ROLLBACK command can be issued where required. The equivalent Query statements COMMIT and ROLLBACK can also be issued. (ROLLBACK rolls back the file to its state at the time of the previous commitment.)

You can switch between automatic and manual commitment modes by issuing further AUTOCOMMIT ON and AUTOCOMMIT OFF commands.

Note that a cannot be restarted automatically after a functional abort of the current Commitment Unit. Instead, the query is re-executed from the beginning after the files have been rolled back.

3.4.4 Processing Current Record Pointers

In IQS, locks are always destroyed automatically after a commitment has been taken. The current record pointers are conserved if the LOCKMARK option was specified in the file definition parameters; otherwise they are lost (default option).

3.4.5 Lock List

If a lock request causes the lock list to overflow, the Commitment Unit is aborted with the return code XAC 7 TABOV. The IQS user is notified by the following IQS message:

```
OVERFLOW OF GAC SPACE. THE CURRENT PROCEDURE IS CANCELLED.
DO YOU WANT TO RESTART IT (YES OR NO) ?
```

If you specify restart, the Commitment Unit is restarted automatically when the overflow situation has been resolved.

Note that from time to time, for each UFAS file within a query, IQS automatically releases from the lock list those entries which relate to records which have not been modified, and which are no longer current records.

Return codes are explained in Appendix A.
3.4.6 **Long Wait Notification**

Long wait notification is a facility provided by GAC-EXTENDED for users of IQS running under the IOF environment, since response times are particularly crucial for these users.

When one user wishes to access a Control Interval which is currently being accessed by another user, if the CI is not free after a certain time, the following message is sent to the user who is waiting:

```
V: EXEC
   LONGWAIT. YOU CAN WAIT OR CANCEL THE QUERY
   DO YOU WANT TO WAIT (YES OR NO) ?
```

If you reply YES, the query will eventually continue automatically when the CI is free. If you reply NO, the query will be aborted.

Note that if you reply YES, you are forced to wait until the CI is free. You cannot interrupt the waiting period by pressing BREAK.

The maximum acceptable waiting time is declared when the GCOS7 system is configured using the CONFIG parameter, LONGWAIT. (See Appendix C.)

3.4.7 **Deadlock Notification in IQS**

If an IQS Commitment Unit has to be aborted for , the system informs you of this by the message below, and also asks if you want to restart the query:

```
V: EXEC
   <user input>
   DEADLOCK. THE CURRENT QUERY IS CANCELLED.
   DO YOU WANT TO RESTART IT (YES OR NO) ?
```

If you do not want to restart the query and therefore reply NO, the system returns to command level and prompts you with V: as usual.

If you indicate that the is to be rerun by typing YES, the system waits until the deadlock is resolved and restarts the query.

When working with IQS commands via menus, you have no choice when a deadlock occurs. The current query is cancelled, the files are rolled back, and a message is issued:

"DEADLOCK. THE CURRENT QUERY IS CANCELLED".

Then the previous screen is repeated to ask you for a new IQS command (or the same one if required).

Further information about how GAC-EXTENDED resolves deadlocks can be found in Section 4.
3.5 THE USE OF GAC-EXTENDED WITH TDS

In this subsection, two types of file sharing are considered; sharing between TDS and another TDS or Batch/IQS, and sharing within TDS itself.

The user-defined files are declared as "controlled" or "non-controlled" by TDS in TDSGEN.

Files which are designated as non-controlled in the TDS environment may not be shared using GAC-EXTENDED.

Files which are designated as controlled by TDS are accessed through GAC-EXTENDED. They are read by default in exclusive mode. However, another read mode may be specified in the MESSAGE statement of the Transaction Section, as shown below.

The clause of the MESSAGE statement:

SUPPRESS CONCURRENT ACCESS CONTROL FOR filename-1 [,filename-2]...

allows the TPRs of the transaction to read the specified TDS-controlled files or IDS/II areas without using GAC-EXTENDED to apply CI locks. The files will be read in statistical read mode, which means that logical inconsistencies may occur if one TPR tries to read a record while a TPR of another transaction is modifying it.

The clause:

SHARED READ FOR filename-3 [,filename-4]...

specifies that the TDS-controlled files specified are to be read under GAC-EXTENDED in "shared read" mode rather than in "exclusive read" mode. This means that any number of readers using this mode may access a CI concurrently. Writing is always exclusive.

Note that the same file name may not appear in both the SUPPRESS CONCURRENT ACCESS CONTROL list and the SHARED READ list in the same MESSAGE statement.

If at least one of the files in a transaction is accessed through GAC-EXTENDED, the Before Journal is allocated by default for the transaction, provided that the Before Journal was previously specified in the JCL for the TDS job. The Before Journal may be suppressed for a transaction by including in the MESSAGE statement the clause:

SUPPRESS BEFORE JOURNAL

The SUPPRESS BEFORE JOURNAL clause is ignored when a transaction is restarted by TDS after an abort due to lack of buffer space.

The rules of locking protocol compatibility are shown in Table 3.2. For more information, see the TDS Administrator's Guide.
### Table 3-2. Rules of Locking Protocol Compatibility for TDS

**GAC-EXTENDED FULLY EFFECTIVE**

- **step start**
  - checkpoint call
  - ABORT
  - OPEN : GAC becomes fully effective
  - ABORT
  - commitment call NOCHKPT = *
  - ABORT
  - checkpoint call
  - ABORT
  - standard commitment call NOCHKPT blank
  - ABORT
  - close last GAC file
  - commitment call NOCHKPT = *
  - ABORT
  - checkpoint call
  - ABORT

**GAC-EXTENDED NOT FULLY EFFECTIVE**

- **step start**
  - checkpoint call
  - ABORT
  - commitment call NOCHKPT = *
  - ABORT
  - standard commitment call NOCHKPT blank
  - ABORT
  - commitment call NOCHKPT = *
  - ABORT
  - checkpoint call
  - ABORT
The Use of GAC-EXTENDED

Table 3-3. The Restrictions on the Use of GAC-EXTENDED Imposed by the SHARE and ACCESS Parameters

<table>
<thead>
<tr>
<th>ASSIGNMENT OF FILES CONTROLLED BY TDS</th>
<th>PERMITTED READ PROTOCOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACCESS</td>
</tr>
<tr>
<td>SHARE</td>
<td></td>
</tr>
<tr>
<td>MONITOR</td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>YES</td>
</tr>
<tr>
<td>WRITE</td>
<td>YES</td>
</tr>
<tr>
<td>NORMAL</td>
<td></td>
</tr>
<tr>
<td>READ (processed like SHARED READ mode)</td>
<td>YES</td>
</tr>
<tr>
<td>WRITE</td>
<td>YES</td>
</tr>
<tr>
<td>ONEWRITE</td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>NO</td>
</tr>
<tr>
<td>WRITE</td>
<td>YES</td>
</tr>
<tr>
<td>NORMAL/ONEWRITE/MONITOR</td>
<td></td>
</tr>
<tr>
<td>SPREAD (processed like SHARED READ mode)</td>
<td>YES</td>
</tr>
<tr>
<td>NORMAL/ONEWRITE/MONITOR</td>
<td></td>
</tr>
<tr>
<td>SPWRITE</td>
<td>YES</td>
</tr>
</tbody>
</table>
3.5.1 Commitment Units

Within the TDS environment, commitments may be taken implicitly or explicitly.

1. If the IMPLICIT COMMITMENT option is included in the MESSAGE statement at TDS generation time, a commitment is taken automatically:
   - at the end of the transaction
   - at each conversation (SEND EGI) and/or each time a WAIT-TIME is specified.

   If you do not want a commitment to be taken at any of these points, specify:

   CALL NOCMIT.

2. If the IMPLICIT COMMITMENT option is not included in the MESSAGE statement, commitments are taken whenever you request them, that is at the end of each TPR in which:

   CALL DFCMIT.

   is specified.

In all cases when commitments are taken, they are taken at the end of the TPR concerned. A commitment is also taken at the end of a transaction.

To maintain an acceptable level of system performance, a conversation must always be a point of commitment.

If a deadlock occurs, the current Commitment Unit is restarted from the beginning.

3.5.2 MAXIMUM NUMBER OF LOCKED PAGES Clause

As discussed earlier in this Section, the MAXIMUM NUMBER OF LOCKED PAGES clause previously used in a TDS environment is no longer taken into account.
3.5.3 Processing Current Record Pointers

The TDS user may choose to retain the current record pointer for a file accessed in a TPR after a commitment has been taken. This is done within the TPR using the PROCEDURE-DIVISION statement:

```
CALL "KEEP-CURRENCIES" USING ifn, status.
```

where ifn is the name of the file whose current record pointers are to be kept and status is the character in which the status of the call is returned.

The current record pointers are not invalidated at the start of the next Commitment Unit. (This does not apply for IDS/II area files.)

Inconsistencies may arise due to modifications by other users. Refer to subparagraph 3.4.3.1.

3.5.4 Long Wait Notification

Normally, the use of GAC within the TDS environment ensures that conflicts and deadlock situations are handled automatically.

The LONGWAIT parameter of the GAC statement in the CONFIG utility specifies the maximum length of time that a user must wait for a CI. The default value of LONGWAIT is one minute. (See Appendix C.)

If the conflict is not resolved by the end of the longwait time, the CU is aborted and the TDS user is informed of this at the terminal via the TDS service message:

```
WAIT FOR RESOURCE ALLOCATION
```

This message is received only if IMMEDIATE DELIVERY is specified in the service message statement of the TDS generation program, otherwise it is suppressed.

At the end of the longwait time, the CU is aborted and is restarted automatically when the conflict is resolved. Between the abort and restart of the CU concerned, other CUs may execute, thus increasing overall efficiency.

If you try to access a Control Interval that is locked by an "unmapped" transaction (see the TDS Concepts Manual), you will be informed immediately. If the transaction that has locked the Control Interval is "mapped", wait until the end of that transaction's current TPR; at that point, if a commitment is taken, the Control Interval becomes available. If there is no commitment, but unmapping takes place, the long wait notification applies. If there is no unmapping, wait until one of the above situations occurs (at the end of a subsequent TPR).
4. Locking and Conflicts

4.1 LOCKING

4.1.1 Introduction

Locking is used to define the degree of sharing which is permitted for a given Control Interval by a given Commitment Unit; that is, whether the Control Interval may be shared with other Commitment Units (and if so, whether write or only read access is allowed), or whether the Control Interval is exclusive to the current Commitment Unit.

Locks are applied by GAC-EXTENDED for a particular user when a record is accessed (read or written) and are removed at the end of the user's current CU.
4.1.2 Locking Modes

There are two true locking modes:

- **SHARED** mode, which allows several readers to access the same resource.
- **EXCLUSIVE** mode, which allows only one user to access a resource.

These are the only locking modes available for resources which are not associated with files. For file resources, there is an additional refinement.

GAC-EXTENDED automatically locks Control Intervals to be updated in EXCLUSIVE mode, and lets you define the lock level for read only mode (SHARED or EXCLUSIVE).

If the STATISTICAL READ locking protocol is used, any read access is accepted, but any attempt to write is rejected. This mode is useful for Commitment Units which only require to read data, without being concerned as to its logical consistency. The use of this locking mode avoids the overheads associated with the use of GAC-EXTENDED, since no locks are held (or indeed requested).

All locks for reading and updating a file are normally held until the commitment is taken. However, in IQS, the access method may request that read locks be released before the commitment is taken.
4.1.3 Read Locks

4.1.3.1 Shared Read

Use of this lock demonstrates willingness to share read access to Control Intervals with other users.

This protocol is recommended for most sharing situations since any number of readers may access the same Control Interval concurrently (whereas they will all have to wait if the Control Interval they are requesting is exclusive to another user).

4.1.3.2 Exclusive Read

Using this protocol:

- access to a Control Interval is not allowed as long as the CI is being accessed by any other user in SHARED or EXCLUSIVE mode,
- once access has been granted, no other user may access the Control Interval in SHARED or EXCLUSIVE mode until the end of the current Commitment Unit.

This protocol should be used with discretion since it forces other users to wait until one user has finished processing, that is, until the end of that user's current Commitment Unit.

However, it should be chosen whenever a high rate of writes and rewrites is expected, as it will help to avoid deadlocks in this situation.

A specific example of the use of the EXCLUSIVE READ protocol is given below, under "Avoidance of Deadlock".

4.1.3.3 Statistical Read

This allows a Control Interval to be read, but not written to, at any time regardless of whether there are any readers or a writer currently accessing the CI in question. Any attempt by a reader in STATISTICAL mode to modify such a CI causes the calling program to be aborted with the return code:

DUFAS 32, ARVIOL

This protocol is useful when statistical information is needed quickly (about, for example, the number of occurrences of a given item or items in the file). In such a situation, it is not really important if another user is updating an occurrence of the same item, since the user reading in STATISTICAL mode is concerned with data items collectively, rather than with the accuracy of a particular item.
Obviously, the logical consistency of statistical accesses cannot be guaranteed, since no locks are set to prevent another user from updating a particular record.

Statistical read should not be confused with the "dirty" read which may happen between coupled systems, because, with statistical read, the current record pointer is protected for the duration of the Commitment Unit against file reorganizations which may be happening concurrently on UFAS files.

Only when statistical read operations are performed in an IDS area may anomalies be encountered in the physical consistency of the pointers. In this case the Commitment Unit is aborted with the return code:

DIACS xx, DATAERR

If the Commitment Unit is aborted in the TDS environment, it is automatically reactivated in exclusive mode (so that there is no risk of the pointers being simultaneously updated). This only happens once however, to avoid causing the data in the file to become permanently inconsistent.

If the Commitment Unit is aborted in the Batch environment, dynamic reactivation does not take place.

Note that if a user specifies READLOCK = STAT for a GAC-EXTENDED file, no entry is made in the lock list when the file is read.

4.1.4 Write Lock

One user is not allowed to write to a record while another user has the Control Interval locked, i.e. until the end of the current Commitment Unit where write access or exclusive or shared read access to that Control Interval has been specified. Once a user has access to the Control Interval, no other user will be able to access it (apart from a statistical reader), until the locks are released (i.e., until the end of the current Commitment Unit).

If a Control Interval already has a shared lock, any attempt to write to the record will automatically upgrade the lock on that Control Interval to exclusive; this will cause the lock requestor to wait until any shared locks applied by other users have been released. This procedure is GAC-EXTENDED's way of ensuring data consistency and security.
4.1.5 Example

Consider a section of a GPL program updating a shared file 2 from an input file 1.

The GCL contains the following statements:

ASG1 FILE1................. SHARE = NORMAL;
ASG2 FILE2................. SHARE = MONITOR;
DEF2 FILE2................. JOURNAL = BEFORE;

The program will contain the following code:

Step start

$H_OPEN 1, PMD = INPUT;       First commitment
$H_OPEN 2, PMD = UPDATE       Records kept in EXCLUSIVE
$H_GET 1;
$H_PUT 2;
..                      The current pointer on 2 is lost
..                      since no LOCKMARK option is specified.
$H_COMMIT
$H_GET 1;                     Last commitment
$H_PUT 2;
$H_CLOSE 1;
$H_CLOSE 2;

Step end
4.1.6 Summary

The table below gives a summary of how the different types of read and write access are defined for a file.

Table 4-1. Definition of Read and Write Access

<table>
<thead>
<tr>
<th>ACCESS</th>
<th>MODE</th>
<th>TDS</th>
<th>BATCH / IQS</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>SHARED</td>
<td>generation (MESSAGE): &quot;SHARED READ FOR</td>
<td>READLOCK = NORMAL (default value)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>file-1 [,file-2] ...&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXCLUSIVE</td>
<td>No declaration</td>
<td>READLOCK = EXCL</td>
</tr>
<tr>
<td></td>
<td>STATISTICAL</td>
<td>Generation (MESSAGE): &quot;SUPPRESS CONCURRENT</td>
<td>READLOCK = STAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACCESS CONTROL FOR file-1 [,file-2] ...&quot;</td>
<td></td>
</tr>
<tr>
<td>WRITE</td>
<td>SHARED</td>
<td>Becomes EXCLUSIVE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EXCLUSIVE</td>
<td>Implicit for write access</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STATISTICAL</td>
<td>Forbidden (return code ARVIOL)</td>
<td></td>
</tr>
</tbody>
</table>
4.2 CONFLICTS

Access to a is granted or denied according to both the type of access requested and whether the Control Interval is currently being accessed in exclusive or shared mode.

The term 'conflict' is used to describe any incompatibility between the current locking situation for a Control Interval and the locking required by the user wishing to access that Control Interval. If a conflict occurs, the requesting user is forced to wait until the Control Interval is available, that is, until the lock currently imposed on the Control Interval has been removed at the end of the concurrent Commitment Unit.

4.2.1 Locking Mode is Exclusive

Any Commitment Unit that tries to access a Control Interval locked in exclusive mode by another Commitment Unit is forced to wait until the Control Interval is released.

![Figure 4-1. Conflict When Locking Mode is Exclusive](image-url)
4.2.2 Locking Mode is Shared

Any Commitment Unit that tries to access in exclusive mode a Control Interval locked in shared mode is forced to wait until the Control Interval is released.

![Figure 4-2. Conflict When Locking Mode is Shared](image)

4.2.3 Locking Mode is Statistical

Any Commitment Unit that tries to read a Control Interval in statistical mode will be granted access even if that CI is locked in exclusive mode by another Commitment Unit.

![Figure 4-3. No Conflict - Locking Mode is Statistical](image)
4.2.4 More Complex Conflict Situations

Consider the file sharing situation in Figure 4-4, where several users want to read a Control Interval and one user is trying to update the same Control Interval, to see how GAC-EXTENDED resolves conflicts.

![Diagram showing the interaction between users and control intervals]

Figure 4-4. How Conflicts are Handled by GAC-EXTENDED

Note that the interaction between the above users could be more complex if, for example, each user read and/or updated several Control Intervals in common, in several files, during concurrent Commitment Units.

Another situation is shown in Figure 4-5. USER 1 updates two related records, a and b, in different Control Intervals. USER 2 reads the records a and b. GAC-EXTENDED is used to avoid the situation where USER 2 could read one record before modification and one record after modification.
In Figure 4-5, USER 2 will be forced to wait as soon as he attempts to read a since USER 1 has already locked a in exclusive mode (write a). Only at the end of CU1, that is, after both a and b have been modified and committed, can USER 2 read a and b.

You should bear in mind the following points:

- If USER 2 had read a before USER 1 had attempted to write to a, then USER 1 would have had to wait until the end of CU2.

- If you test for yourself on the basis of read and write locks, you will see that the consistency of the relationship between a and b would still be preserved even if, for example, USER 2 tries to read a and b in reverse order. Note that some access conflicts might provoke a deadlock situation, see below.

- If USER 2 attempts to read with READLOCK = STAT, it is not guaranteed that the values of a and b will be consistent, i.e. USER 2 might receive a modified value for a and an unmodified value for b.
4.3 DEADLOCKS

4.3.1 The General Situation

Whenever a user has to wait for a locked resource, GAC-EXTENDED handles the waiting and rescheduling automatically.

Each time a conflict occurs, a deadlock detection or prevention mechanism is activated.

If the requesting Commitment Unit has not already been aborted due to deadlock, a deadlock detection mechanism is activated. In this case, if a deadlock is detected, the requesting Commitment Unit is aborted (and its resources released), in order to resolve the deadlock situation.

If the Commitment Unit has already been aborted due to deadlock, a deadlock prevention mechanism is activated, based on the DIE-WAIT algorithm. This mechanism works as follows:

If the Commitment Unit TR, requests the resource X, which is currently being accessed by the Commitment Units CU1, CU2, ... ,CUn (shared lock), and this request induces a conflict, then:

1. All Commitment Units holding the resource which were scheduled more recently than (i.e., they are younger than) TR are aborted.

2. If there are Commitment Units scheduled earlier than (i.e. older than) TR, still holding the resource, then TR has to wait until these Commitment Units have finished.

The aborted Commitment Units release their resources and are restarted from the beginning once the contending Commitment Units are completed. That is, a serialization takes place.

IQS users may also request notification in the event of a deadlock. They may thus themselves determine the appropriate action. In TDS mode, it is TDS itself which requests notification after a deadlock, and regains control.

It is more usual however, for deadlocks to occur between just two users attempting to access a file in exclusive mode. This is the classic deadlock situation, as shown below.
4.3.2 How GAC-EXTENDED Deals With Deadlocks

Consider a situation where deadlock can occur over a single Control Interval, using the following model:

<table>
<thead>
<tr>
<th>CU1</th>
<th>CU2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1:</td>
<td>R2:</td>
</tr>
<tr>
<td>read x</td>
<td>read x</td>
</tr>
<tr>
<td>x = x-1</td>
<td>x = x-2</td>
</tr>
<tr>
<td>W1:</td>
<td>W2:</td>
</tr>
<tr>
<td>write x</td>
<td>write x</td>
</tr>
</tbody>
</table>

Let us assume also that the declared locking mode is SHARED in both cases. If the order of processing happens to be R1, W1, R2, W2, or R2, W2, R1, W1, there is no problem. However, GAC-EXTENDED must deal with other processing combinations (e.g., R1, R2, W2, W1). Figure 4-6 shows how GAC-EXTENDED deals with such a situation.

![Figure 4-6. How GAC-EXTENDED Resolves a Deadlock Situation](image-url)
Locking and Conflicts

GAC-EXTENDED detects the deadlock situation, where each user is waiting on the other to access record x. The first time a deadlock is detected, the Commitment Unit in which deadlock is detected (CU1 in this case), is aborted and restarted. If subsequent conflicts occur involving CU1, the younger of the two Commitment Units is aborted, thus preventing a new deadlock situation. This procedure converts a potential processing order that could produce inconsistencies (R1, R2, W2, W1), into the satisfactory order R2, W2, R1, W1.

4.3.3 Avoidance of Deadlock Using Exclusive Read

The above type of situation can be avoided if one or both users specify exclusive read (READLOCK = EXCL). Now consider the previous example, where CU2 has READLOCK = EXCL.
Figure 4-7 shows that the use of an exclusive read lock can avoid a deadlock situation. However, exclusive read locks should be reserved for where the above situation is likely to occur often, to avoid a degradation of performance as a result of the higher frequency of abort/restart operations.
## 4.4 SUMMARY OF THE EFFECTS OF SHARING REQUESTS UNDER GAC-EXTENDED

**Table 4-2. Effects of Requests to Share Files Under GAC-EXTENDED**

<table>
<thead>
<tr>
<th>REQUEST CURRENT SITUATION</th>
<th>SHARED READ</th>
<th>EXCLUSIVE READ</th>
<th>WRITE (EXCLUSIVE)</th>
<th>STATIC READ</th>
</tr>
</thead>
<tbody>
<tr>
<td>CI not locked</td>
<td>access granted</td>
<td>access granted</td>
<td>access granted</td>
<td>access granted</td>
</tr>
<tr>
<td>CI locked by shared readers only</td>
<td>access granted</td>
<td>WAIT</td>
<td>WAIT</td>
<td>access granted</td>
</tr>
<tr>
<td>CI locked by exclusive reader</td>
<td>WAIT</td>
<td>WAIT</td>
<td>WAIT</td>
<td>access granted</td>
</tr>
<tr>
<td>CI locked by writer</td>
<td>WAIT</td>
<td>WAIT</td>
<td>WAIT</td>
<td>access granted</td>
</tr>
<tr>
<td>CU waiting for requesting CU to release CI</td>
<td>access granted</td>
<td>DEADLOCK: requesting (or youngest) CU aborted and restarted</td>
<td>DEADLOCK: requesting (or youngest) CU aborted and restarted</td>
<td>access granted</td>
</tr>
</tbody>
</table>

The number of and situations occurring in a step is reported in the JOR as follows:

GAC01.IFN = ifn  MAXLOCKCOUNT = n1
GAC01.IFN = ifn  CONFLICT CNT = n2  DEADLOCK CNT = n3
A. Return Codes

A.1 INTRODUCTION

This Appendix contains a list of the return codes associated with GAC-EXTENDED, together with a short explanation of what caused the error.

Some return codes, however, are not included here. This is because they refer to errors in the internal operation of GAC-EXTENDED. In such cases you can do nothing to correct the error, and should contact the Service Center.

A.1.1 Format

Each return code has the format:

XAC n mnemonic

where:

n is a number which indicates which of the internal functions of GAC-EXTENDED has generated the return code. A list of these functions and their corresponding numbers is given in Table A.1

mnemonic is one of a set of Bull standard mnemonics which indicate briefly the reason why the return code was output. General information about the interpretation of a mnemonic may be found in the manual "Messages and Return Codes Directory".

The return codes are listed in numerical order of function number and alphabetical order of mnemonic. The most important return codes are marked with an asterisk, *. 
### Table A-1. GAC-EXTENDED Internal Function Numbers

<table>
<thead>
<tr>
<th>Number (n)</th>
<th>Internal Function of GAC-EXTENDED</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TDS clean point and restart: buffer management interface: provision of GAC-EXTENDED statistics.</td>
</tr>
<tr>
<td>2</td>
<td>IQS primitives.</td>
</tr>
<tr>
<td>3</td>
<td>Freeing of locked resources.</td>
</tr>
<tr>
<td>4</td>
<td>Management of GAC-EXTENDED’s internal tables.</td>
</tr>
<tr>
<td>5</td>
<td>Management of Commitment Units; management of the CURRENCY option; initialization of GAC-EXTENDED sequence control blocks.</td>
</tr>
<tr>
<td>6</td>
<td>Taking commitments (Batch and IQS).</td>
</tr>
<tr>
<td>7</td>
<td>Locking of resources.</td>
</tr>
<tr>
<td>8</td>
<td>Unlocking resources locked for reading, except those corresponding to current record pointers (IQS).</td>
</tr>
<tr>
<td>9</td>
<td>Initialization of GAC-EXTENDED tables at system initialization.</td>
</tr>
<tr>
<td>10</td>
<td>Open, close and deassign.</td>
</tr>
<tr>
<td>11</td>
<td>GAC-EXTENDED-TDS interface at the start of a TDS session.</td>
</tr>
<tr>
<td>14</td>
<td>Step supervision.</td>
</tr>
<tr>
<td>15</td>
<td>Asynchronous call mechanism.</td>
</tr>
<tr>
<td>17</td>
<td>Conflict processing.</td>
</tr>
<tr>
<td>18</td>
<td>Notification processing.</td>
</tr>
<tr>
<td>19</td>
<td>Dynamic space allocation at step level.</td>
</tr>
</tbody>
</table>
A.2 LIST OF RETURN CODES

* XAC 1 DEADLOCK

A TDS Commitment Unit has been aborted due to a deadlock. The Commitment Unit is automatically restarted after the conflicting Commitment Unit has completed. The transaction is first rolled back to that point.

XAC 1 NOTOPEN

The file you are working on has been closed.

XAC 1 SHLVVIOL

Sharing rules violation. Restart has been requested for a file which is not controlled by GAC-EXTENDED.

XAC 1 TABOV

External request to abort the Commitment Unit due to a lock list overflow. The Commitment Unit is automatically restarted on resolution of the problem.

XAC 2 NOTOPEN

The file you are working on has been closed.

* XAC 2 NOWAIT

The file specified is currently being accessed by another program, so you cannot access it.

* XAC 2 SHLVVIOL

GAC-EXTENDED has been called for a file which is not controlled by GAC-EXTENDED.

* XAC 2 WRONGORG

The file specified is not a UFAS file.

XAC 4 DAMAGED

An error has occurred in GAC-EXTENDED's internal tables. GAC-EXTENDED will reinitialize and restart automatically.

XAC 5 IFNERR

An invalid file has been specified.
GAC-EXTENDED - User’s Guide

XAC 5 LOKVIOL

- Statistical read has been requested for a TDS file, but the transaction has not been defined with SUPPRESS CONCURRENT ACCESS CONTROL for this file. The transaction is aborted.

- A TDS file has been assigned with SHARE = ONEWRITE, but PROCESSING-MODE INPUT was specified at generation time. The transaction is aborted.

XAC 5 NOTOPEN

The file you are working on has been closed.

* XAC 5 SHLVVIOL

Sharing rules violation. The file specified is not controlled by GAC-EXTENDED.

* XAC 5 SYSOV

No more lock identifiers may be scheduled.

XAC 6 ARGERR

A parameter is missing from H_GAC_UCOMIT (COBOL programs only).

* XAC 6 CONFLICT

TDS or an interactive IQS job has tried to call the H_COMMIT primitive, which is restricted to a purely Batch environment.

XAC 7 ARVIOL

- Updating a file not allowed when READLOCK = STAT.

- Impossible to update a file with neither Before Journal nor Deferred Write protection.

XAC 7 BUSY

Resource requested is already locked (TEST mode).

XAC 7 NOTOPEN

The file you are working on has been closed.

* XAC 7 TABOV

The lock list has overflowed. The Commitment Unit is aborted and restarted automatically when the overflow situation has been resolved. Any "big consumers" are aborted and restarted in a special mode to avoid allocating a large number of lock list entries to the Commitment Unit.

XAC 8 IFNERR

An invalid file has been specified.
Return Codes

* XAC 8 WRONGORG

The specified file is not of UFAS or IDS/II organization.

XAC 9 DAMAGED

There is an error in the GAC-EXTENDED code. Automatic reinitialization does not take place.

* XAC 10 FUNCNAV

GAC-EXTENDED is not available on the system (Batch users only).

* XAC 10 LOKVIOL

- A Batch file cannot be opened for update as the Before Journal is missing.
- An attempt has been made to open for update a TDS file specified with neither the Before Journal nor Deferred Update.
- An attempt has been made to deassign a file, but some of the file resources are still locked.

XAC 10 OPTERR

For Batch steps, the REPEAT option is missing or GAC has been specified in a multiprocessing Batch environment.

* XAC 10 SHLVVIOL

A write-control error has occurred due to a conflict in the values of the SHARE and ACCESS parameters.

* XAC 10 WRONGORG

The specified file is neither a UFAS file nor an IDS/II file.

XAC 19 ENTRYOV

Insufficient TYPE2 space available for GAC internal structures.
B. Error Message Index

This Appendix lists the most common statistical and error messages associated with GAC-EXTENDED, in order of message number. These messages are explained fully in the manual "Error Messages and Return Codes Message Directory".

Statistical Message Descriptions

GAC01. IFN = ifn MAXLOCKCOUNT = n1
GAC01. IFN = ifn CONFLICT CNT = n2 DEADLOCK CNT = n3

Meaning: Information message, indicating for the specified file:
- the maximum number of locks taken on the file by a Commitment Unit (n1),
- the number of access conflicts detected while accessing a CI of the file (n2),
- the number of deadlock situations detected while accessing a CI of the file (n3).

Result: Too high a value for these counters may indicate an inadequate organization of the data base. The message is printed each time a monitored file is deassigned.

GAC03 IFN = ifn SHARE = MONITOR ONLY WITH UFAS DISK FILE.

Meaning: Assigning the value MONITOR to the SHARE parameter of a file indicates that the sharing of the file is to be controlled by GAC-EXTENDED. GAC-EXTENDED may only be specified for UFAS and IDS/II files.

Result: The operation you requested is not performed.

Action: Check the file organization.
GAC-EXTENDED User's Guide

GAC04 IFN = ifn REPEAT OPTION MISSING
Meaning: For Batch steps to be able to lock files using GAC-EXTENDED, the REPEAT option must be specified. This has not been done.
Result: GAC-EXTENDED will not restart after a deadlock.
Action: Specify the REPEAT option.

GAC05 IFN = ifn MONITORED FILE IN TDS MUST BE CONTROLLED
Meaning: A file which is non-TDS-controlled has been assigned with SHARE = MONITOR.
Result: The file will not be opened.
Action: Either alter the value of SHARE or make the file a controlled file.

GAC06 BEFORE JOURNAL MISSING
Meaning: When GAC-EXTENDED is used to control file sharing, the Before Journal must be specified for each of the files to be shared.
Result: A file for which Before journalization has not been specified cannot be shared using GAC-EXTENDED.
Action: Specify that the file is to be shared without GAC-EXTENDED, or specify Before journalization for the file.
C. The CONFIG Statement: GAC

If the GAC-EXTENDED software product has been ordered and validated, the utility contains a statement which specifies certain parameters used by GAC-EXTENDED.

This statement has the format:

```
GAC [ LONGWAIT = { 60|nnn } ] [,LOCKSIZE = {100|nnn} ]
[ ,NBLOCKID = {340|nnnn} ]
```

The meaning of each parameter is as follows:

**LONGWAIT**
This parameter specifies the maximum number of seconds a user in IQS or TDS must wait to access a file before he is informed of the fact and given the option of aborting the offending Commitment Unit. LONGWAIT is specified as a decimal value in the range 1 to 300 inclusive. The default value is 60.

**LOCKSIZE**
This parameter specifies in Kbytes the maximum size of the lock list which contains details of the Control Intervals locked by all the system users at any one time. LOCKSIZE is specified as a 3-digit decimal value in the range 40 to 320 inclusive. The default value is 100. The maximum capacity of the lock list is 19,900 entries.

**NBLOCKID**
This parameter specifies the maximum number of lock-ids, (lock-owner-identifiers), which can be active simultaneously for GAC-EXTENDED files. NBLOCKID is specified as a 4-digit decimal value between 20 and 5460 inclusive. The default value is 340.

It is probable that when your GCOS7 operating system is configured for the first time, the GAC statement will be included in CONFIG with no parameters specified, and therefore the defaults will apply. You should check the JOR and the error reports the first few times that the system is run, to ensure that the parameter values have been correctly specified. For example, if the lock list overflows it may be necessary to increase LOCKSIZE, etc.

If this is the case, CONFIG may be run again with new values specified for the GAC parameters.
## Index

### A
- abort: 3-10, 3-15
- ACCESS Parameter: 3-23
- ACCESS parameter: 2-3, 3-4, 3-6, 3-17
- advantages of GAC-EXTENDED: 1-8
- After Journal: 3-1, 3-3
- ASSIGN (JCL statement): 3-1, 3-6
- AUTOCOMMIT OFF: 3-18, 3-19
- AUTOCOMMIT ON: 3-18, 3-19

### B
- Batch environment: 3-1
- Before Journal: 3-1, 3-3, 3-21
- usage considerations: 3-15
- big consumers: 3-2

### C
- catalog: 3-1
- checkpoints: 3-13
- CKINF parameter: 3-12
- COMMIT: 3-18, 3-19
- Commitment: 1-5
- commitment: 2-13, 2-14
- commitment call parameters: 3-10, 3-10
- Commitment calls: 3-13
- commitment calls restrictions: 3-9
- Commitment Unit considerations management: 1-5, 3-10
- Commitment Units: 2-12
- Batch processing: 1-5
- IOF: 1-5
- IQS: 1-5
- TDS: 1-5
- Commitments IQS: 3-18

### D
- data security: 1-7
- Deadlock: 3-15, 3-24, 4-11, 4-12
- deadlock avoidance: 4-13
- IQS: 3-20
- JOR report: 4-15
- prevention mechanism: 4-11
- Deferred Update: 3-3
- DEFINE statement: 3-7
- degraded mode: 3-2
- DIE-WAIT algorithm: 4-11

### E
- error messages: B-1
- error recovery: 1-8, 2-9
- EXCLUSIVE mode: 3-6
- EXEC_PG
  - GCL procedure: 3-6
<table>
<thead>
<tr>
<th>Section</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td></td>
</tr>
<tr>
<td>FBO (Fixed Block Organization)</td>
<td>1-2, 2-7</td>
</tr>
<tr>
<td>FILE SHARING</td>
<td>1-3, 2-1, 2-6</td>
</tr>
<tr>
<td>file sharing</td>
<td></td>
</tr>
<tr>
<td>by step</td>
<td>2-6</td>
</tr>
<tr>
<td>FILE SHARING rules</td>
<td>2-6</td>
</tr>
<tr>
<td>Modified Data</td>
<td></td>
</tr>
<tr>
<td>multiple sharing</td>
<td>2-10</td>
</tr>
<tr>
<td>multiple updates</td>
<td>1-7</td>
</tr>
<tr>
<td>G, H</td>
<td></td>
</tr>
<tr>
<td>GAC-EXTENDED</td>
<td>2-11</td>
</tr>
<tr>
<td>general recommendations</td>
<td>3-9</td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>IMPLICIT COMMITMENT</td>
<td>3-24</td>
</tr>
<tr>
<td>IOF (IOS) environment</td>
<td>3-1</td>
</tr>
<tr>
<td>IQS</td>
<td>3-17</td>
</tr>
<tr>
<td>J, K</td>
<td></td>
</tr>
<tr>
<td>KEEP considerations</td>
<td>3-16</td>
</tr>
<tr>
<td>KEEP CURRENCY</td>
<td>3-8</td>
</tr>
<tr>
<td>L</td>
<td></td>
</tr>
<tr>
<td>lock list</td>
<td>3-2, 3-19</td>
</tr>
<tr>
<td>lock list overflow</td>
<td>3-2</td>
</tr>
<tr>
<td>locking modes</td>
<td>4-2</td>
</tr>
<tr>
<td>Locking Protocol Compatibility</td>
<td>3-22</td>
</tr>
<tr>
<td>LOCKMARK</td>
<td>3-8, 3-19</td>
</tr>
<tr>
<td>Locks</td>
<td>4-1</td>
</tr>
<tr>
<td>locks</td>
<td>3-7, 3-13, 3-17</td>
</tr>
<tr>
<td>LOKVIOL</td>
<td>3-7</td>
</tr>
<tr>
<td>Long Wait Notification</td>
<td>3-25</td>
</tr>
<tr>
<td>Long wait notification</td>
<td>3-20</td>
</tr>
<tr>
<td>M</td>
<td></td>
</tr>
<tr>
<td>MAXIMUM NUMBER OF LOCKED PAGES</td>
<td>1-2</td>
</tr>
<tr>
<td>MAXIMUM NUMBER OF LOCKED PAGES clause</td>
<td>3-2</td>
</tr>
<tr>
<td>MESSAGE statement</td>
<td>3-21</td>
</tr>
<tr>
<td>MODE parameter</td>
<td>3-12</td>
</tr>
<tr>
<td>NOCHKPT parameter</td>
<td>3-12</td>
</tr>
<tr>
<td>NUMLOCK</td>
<td>1-2, 3-2</td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Operation modes</td>
<td>3-7</td>
</tr>
<tr>
<td>overheads</td>
<td>1-3, 3-9</td>
</tr>
<tr>
<td>O</td>
<td></td>
</tr>
<tr>
<td>P, Q</td>
<td></td>
</tr>
<tr>
<td>query restart</td>
<td>3-19, 3-20</td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Read Locks</td>
<td>3-7, 3-18</td>
</tr>
<tr>
<td>READLOCK</td>
<td>3-6</td>
</tr>
<tr>
<td>response time restart</td>
<td>3-10, 3-15</td>
</tr>
<tr>
<td>Restrictions</td>
<td>3-23</td>
</tr>
<tr>
<td>ROLLBACK</td>
<td>3-18, 3-19</td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>scheduling mechanism</td>
<td>1-8</td>
</tr>
<tr>
<td>SHARE parameter</td>
<td>2-2, 3-23</td>
</tr>
<tr>
<td>SHARE value</td>
<td>3-7</td>
</tr>
<tr>
<td>SHARED READ</td>
<td>3-21</td>
</tr>
<tr>
<td>SHARED READ FOR clause</td>
<td>2-1</td>
</tr>
<tr>
<td>sharing modes</td>
<td>2-4</td>
</tr>
<tr>
<td>STEP statement</td>
<td>2-19</td>
</tr>
<tr>
<td>steps</td>
<td>2-6</td>
</tr>
<tr>
<td>SUPPRESS BEFORE JOURNAL</td>
<td>3-21</td>
</tr>
<tr>
<td>SUPPRESS CONCURRENT ACCESS CONTROL</td>
<td>2-9, 3-21</td>
</tr>
</tbody>
</table>
Index

T

TDS environment 3-1 3-21
U

UFAS file
sequential processing 3-16
UFAS-EXTENDED access method 1-2
updating sequence 2-14

V, W, X, Y, Z

Write Lock 4-4