BS2000/OSD-BC V8.0

Files and Volumes Larger than 32 GB

Edition December 2009

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:
manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2000

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Contents

1 Introduction

BS2000/OSD supports files and volumes with a capacity of up to 4 terabytes. Files and volumes which exceed 32 GB are known as “large files” and “large volumes”, and together they are referred to as “large objects”.

On S, SX and SQ servers, large volumes can be configured on the external disk subsystems (Symmetrix, FibreCAT CX) which are connected via FibreChannel.

Large files and large volumes are only supported in special pubsets, which must be assigned attributes for using these large objects. These pubsets are also called “large pubsets” and cannot be imported into OSD-BC versions earlier than V5.0.

In certain data structures, including the catalog entry, the 3-byte fields that have been used for page addressing must be converted to 4-byte fields.
Converting 3-byte fields to 4-byte fields affects all components, products and applications that work directly or indirectly with these fields. The TPR interfaces have been modified accordingly, but not all of the TU user interfaces can compatibly support these large files.

1.1 Target groups

This manual is intended for programmers and systems support staff. It aims to facilitate migration to large files and large volumes.

In order to work with this manual, you need very good knowledge of DMS and the SDF command language (for systems support) or the Assembler programming language (for programmers).

1.2 Structure of the manual

The chapter “Large objects in BS2000/OSD” describes the fundamental theoretical aspects associated with the introduction of large objects (large volumes and large files). The changes that affect pubsets, volumes and files in BS2000/OSD are outlined and general restrictions are mentioned. There is a classification for user programs which describes how the individual programs handle large files.

The chapter “Systems support” outlines the role of systems support in introducing and maintaining large objects.
Emphasis is placed on the additional tasks in the field of “Pubset Management”. But there are also various aspects of “Catalog and File Management” and “System Initialization and Parameter Service” that need to be addressed. One section outlines the changes in the command interfaces which result from the introduction of large objects.
In addition, adjustments to individual utilities that are important in the context of systems support and that have been affected by the extensions for large objects will be mentioned. The end of the chapter provides a summary of potential sources of errors and conflicts.

The chapter “Users and programmers” contains detailed descriptions of the extensions made to non-privileged command interfaces and Assembler macro interfaces to accomodate large files. In addition, there are notes on RFA and high-level programming languages that are affected by large objects. The end of the chapter provides a summary of potential sources of errors and conflicts.

The chapter “Notes on migration” contains notes on the introduction of large files. Necessary technical extensions are described in the previous chapters. Possible steps for migration and the considerations that should be included in planning will be outlined here.

Readme file

Information on any functional changes to the current product version and additions to this manual can be found in the product-specific Readme file.

Readme file online

Readme files are available to you online under the various products and in addition to the product manuals at *http://manuals.ts.fujitsu.com*.

Readme file under BS2000/OSD

On your BS2000 system you will find Readme files for the installed products under the file name:

`SYSRME.<product>.<version>.D`
(e.g. `SYSRME.BS2CP.170.D` for BS2000/OSD-BC).

Please refer to the appropriate system administrator for the user ID under which the Readme file can be found. You can also obtain the complete path name of the Readme file using the following command:

```
/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>,LOGICAL-ID=SYSRME.D
```

You can view the Readme file with the `/SHOW-FILE` command or by opening it in an editor or print it on a standard printer using the following command (e.g. for BS2000/OSD-BC):

```
/PRINT-DOCUMENT FROM-FILE=SYSRME.BS2CP.170.D,LINE-SPACING=*BY-EBCDIC-CONTROL
```

Additional product information

Current information, version and hardware dependencies and instructions for installing and using a product version are contained in the associated Release Notice.
These Release Notices are available at *http://manuals.ts.fujitsu.com*.

1.3 Notational conventions

References to other publications in the text are given in the form of abbreviated titles with a reference number in square brackets. The full title of each publication is given in the "Related publications" section as of page 101.

Commands mentioned in this manual are described in the "Commands" manual [10]. The macros mentioned are described in the following manuals: "Executive Macros" [11] and "DMS Macros" [3]. The metasyntax of the commands or macros can be found in the corresponding manuals.

⚠ This symbol indicates warnings that make the reader aware of possible data loss or serious conflicts.

2 Large objects in BS2000/OSD

In the following descriptions, large volumes and large files will be referred to as “large objects”.

The continual growth of disk storage capacity and of data that needs to be available online has been taken into account in OSD-BC V5.0 in the enlargement of previously available disk and file sizes of around 32 GB.
As of OSD-BC V5.0:

– the maximum capacity of a volume is approx. 4 TB (2 147 483 647 PAM pages)

– the maximum file size is approx. 4 TB (2 147 483 647 PAM pages)

This means that within the operating system, 4-byte block numbers and 4-byte counters must be used systematically for file sizes and disk sizes.

Block numbers and block counters become visible at different interfaces to the user in BS2000/OSD. Although 4-byte fields have been used exclusively in all new versions of these interfaces, some old interface versions may still use 3-byte fields. This may lead to compatibility problems for files ≥ 32 GB (and in some rare cases also for volumes ≥ 32 GB).

The compatibility aspect has been taken into account in that new pubset types have been created as special containers for large objects. This has allowed the previous world of “small objects” to be separated off and facilitates the successive introduction of large volumes and large files.

Different conditions apply, however, when introducing large volumes and large files.

– The introduction of large volumes is designed to allow simple, low-cost expansion of capacity in the Data Center. This process is not visible to users or programs.

– Large files are necessary in order to facilitate further growth of applications, where the size of individual (and generally relatively few) files will exceed the former limit of 32 GB in the near future.
In this context, it will be necessary in some cases to modify the programs affected, that, for instance, use old versions of particular interfaces or implicitly assume a maximum file size of 32 GB, in order to guarantee data integrity when the 32-GB limit is exceeded. Programs that are not designed to work with large files are not affected.

2.1 Pubsets

Standard pubsets cannot contain any large objects. Only LARGE_OBJECTS pubsets are able to accommodate large objects. Two types of these pubsets are available:

– LARGE_OBJECTS pubsets without large files:

  This pubset type allows large volumes, but limits the file size allowed to 32 GB. This means that the pubset may contain volumes ≥ 32 GB, but necessarily does not currently contain such volumes.

  The introduction of large volumes can only affect the PHP of the extent lists, a value which is of no interest to the application program and cannot under any circumstances be directly altered by it. Thus, this pubset type allows large volumes to be integrated into existing application programs with (almost) no problems. From the point of view of the user, these pubsets behave (almost) as conventional pubsets.

– LARGE_OBJECTS pubsets with large files:

  This pubset type allows large volumes and large files, but they do not necessarily exist.

  It allows large files to be separated from programs that use interfaces that are incompatible and supports the step-by-step introduction of large volumes and - at a second step - of large files.

Both new pubset types are only recognized as of OSD-BC V5.0 and cannot be imported into older versions. This information is stored using the appropriate version indicator in the Standard Volume Labels (SVLs) of the PUBRES or VOLRES of the Control Volume Set. In addition, the version field is assigned a value in the base record of the PUBRES. The value shows that the volume cannot be supported in OSD-BC < V5.0. This ensures that the pubset cannot be imported into older versions.

Large objects are supported by both SF and SM pubsets. To achieve this, two additional attributes exist:

LARGE_OBJECTS	Attribute for supporting volumes ≥ 32 GB
LARGE_FILES_ALLOWED	Attribute controls whether large files (files ≥ 32 GB) are also supported.

i In the case of SM pubsets, the attributes LARGE_OBJECTS and LARGE_FILES_ALLOWED are also properties of the pubset, rather than of (individual) volume sets, since volume sets would otherwise be visible on different interfaces to the user and could cause incompatibilities.

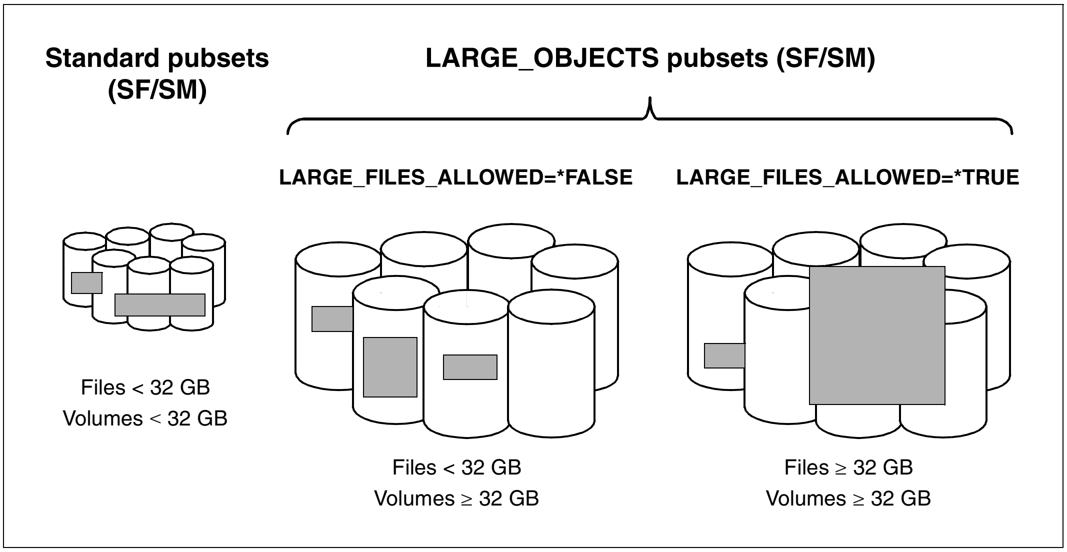


Figure 1: Attributes of LARGE_OBJECTS pubsets

Defining LARGE_OBJECTS pubsets

The properties LARGE_OBJECTS and LARGE_FILES_ALLOWED are static and can be specified when configuring the pubset with SIR (see page 25).

Using the SET-PUBSET-ATTRIBUTES command, existing standard pubsets can be upgraded to LARGE_OBJECTS pubsets with and without support for large files (see page 38). The following upgrades are possible in this context:

$$\begin{tikzcd}
\text{Standard pubset} \arrow[d] \arrow[dd, bend right] \\
\text{LARGE\_OBJECTS pubset without large files} \arrow[d] \\
\text{LARGE\_OBJECTS pubset with large files}
\end{tikzcd}$$

When the pubset is imported, the attributes are added to the MRSCAT.

Warning!

Conversion back to a standard pubset is not possible.

The existence of large files can in certain cases affect the executability of applications (see chapter “Users and programmers” as of page 47).

Restrictions for LARGE_OBJECTS pubsets

– LARGE_OBJECTS pubsets cannot be imported into OSD-BC < V5.0.

– Shared pubset clusters with OSD-BC < V5.0 are not possible.

– LARGE_OBJECTS pubsets without large files are valid as home pubsets, but those with large files are not.

i Another catalog format - EXTRA LARGE - was introduced in BS2000/OSD-BC V6.0B. LARGE_OBJECTS pubsets for large volumes or large files can be used irrespective of pubsets with the catalog format EXTRA LARGE. In other words a pubset with an EXTRA LARGE catalog by default does not allow any large volumes and files. For details, see section “Catalog formats” on page 20.

Summary of pubset management interfaces affected by 32 GB objects

Interface	**Change**
Privileged commands	
SET-PUBSET-ATTRIBUTES	Specification whether existing pubsets are to be upgraded to LARGE_OBJECTS pubsets with or without large files.
SET-PUBSET-ATTRIBUTES	Two S variables display the attributes LARGE-VOLUMES and LARGE-FILES: var(*LIST).LARGE-VOL and var(*LIST).LARGE-FILE
SHOW-MASTER-CATALOG-ENTRY	Output of LARGE_OBJECTS properties for LARGE_OBJECTS pubsets
Macros	
STAMCE	Output of MRSCAT entries related to LARGE_OBJECTS
Privileged utilities	
SIR	Installing and extending pubsets, initializing volumes in respect of LARGE_OBJECTS

Table 1: Summary of pubset management interfaces affected by 32-GB objects

2.2 Volumes

Like pubsets, large volumes have their own attribute: LARGE_VOLUME.

The LARGE_VOLUME attribute characterizes a logical volume and indicates that the volume has a gross capacity ≥ 32 GB.
The LARGE_VOLUME attribute is stored in the base record of the SVL when the volume is initialized with VOLIN. The version field is simultaneously set to a value that corresponds to OSD-BC V5.0, so that it cannot be processed in older versions.

Just as the LARGE_OBJECTS characteristic is for pubsets, the LARGE_VOLUME characteristic is permanent and is stored in the SVL.

Access to large volumes is prevented in OSD-BC < V5.0 by the version indicator in the base record of the SVL; utilization is not possible under any circumstances.

Restrictions for large volumes

– BS2000/OSD
  Access to large volumes only possible as of OSD-BC V5.0.

– Large volumes can only be added to LARGE_OBJECTS pubsets.

– VOLIN
  VOLIN supports a maximum capacity of 2 TB for any formatting.

– Large volumes are not allowed as private disks. Any attempt to initialize a large volume as a private disk will be rejected with the following error message:

```
NVL0146   DISK CAPACITY GREATER EQUAL 32GB IS NOT PERMITTED FOR PRIVATE DISKS
```

Volume management interface affected by 32-GB objects

Interface	**Change**
Privileged utilities	
SIR	Installing and extending pubsets, initializing volumes in respect of LARGE_OBJECTS

Table 2: Summary of volume management interfaces affected by 32-GB objects

High-performance support of large volumes

If the load on the volume increases in accordance with the larger amount of data when large volumes are used, provisions should be made to ensure high-performance support:

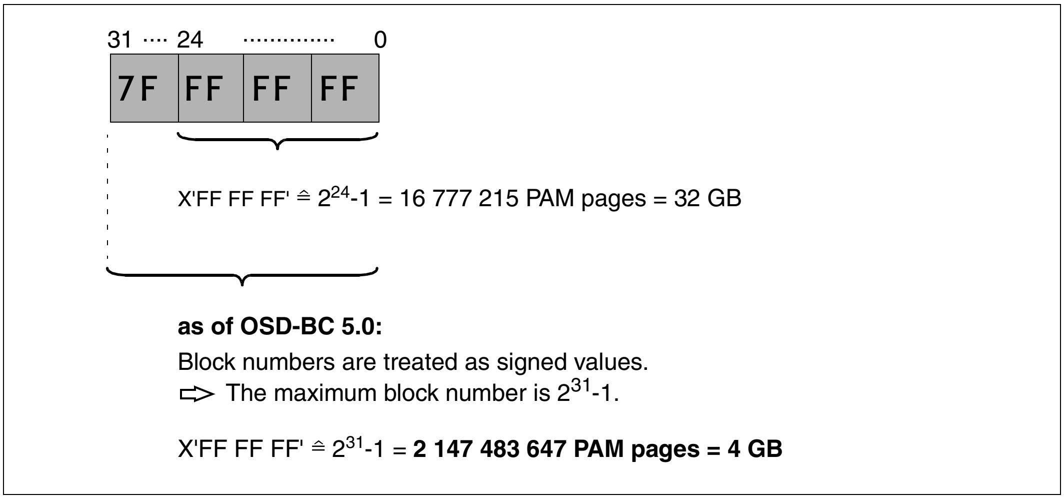
– External disk storage subsystems which employ the RAID hard disk system can be employed to increase performance (and to enhance data security). When large volumes are used, RAID 5, RAID 6 or preferably RAID 1/0 is recommended. In this case the data of a logical volume is distributed over multiple physical disks (striping).

– In addition, use of the BS2000/OSD function PAV (Parallel Access Volume) or preferably dynamic PAV is recommended on S servers in order to parallelize the disk inputs/outputs. On SX and SQ servers the corresponding function using RSC for disk I/Os is available by default in OSD/XC V2.0 and higher.

Combining both measures results in considerable enhancements in TP and batch operation (in the case of multitask batch), in terms of both the I/O times and the throughput. Also refer to the “Performance Handbook” [12].

2.3 Files

The maximum file size is approx. 4 TB (2 147 483 647 PAM pages). This means that within the operating system, 4-byte block numbers and 4-byte counters must be used systematically for file sizes and disk sizes (see figure 2). (The term “block” in this context is synonymous with the terms “PAM page” and “half page”.)



X'FF FF FF' ≙ 2^{24}-1 = 16 777 215 PAM pages = 32 GB

as of OSD-BC 5.0:
Block numbers are treated as signed values.
⇨ The maximum block number is 2^{31}-1.

X'FF FF FF' ≙ 2^{31}-1 = **2 147 483 647 PAM pages = 4 GB**

Figure 2: 3-byte and 4-byte fields and the resulting file and data volume sizes

Extending the catalog entry

The introduction of 4-byte fields for the following data stored in the catalog entry was a key aspect in the lifting of the 32-GB limit for volume and data sizes.

– FILE-SIZE, the storage space allocated for the file

– HIGHEST-USED-PAGE, the amount of storage that currently contains data

– LHP (logical halfpage number) and PHP (physical halfpage number) of the individual extents in the extent list, that allocate “physical” halfpages of volumes to the logical halfpages.

3-byte and 4-byte fields

Block numbers and block counters become visible at different interfaces to the user in BS2000/OSD. Although 4-byte fields have been used exclusively in all new versions of these interfaces, some old interface versions may still use 3-byte fields. If files ≥ 32 GB exist, you may experience compatibility problems, in a few cases also when “only” volumes ≥ 32 GB exist.

Additional format for the extent list

The introduction of 4-byte LHPs and 4-byte PHPs meant that an additional format had to be introduced for the extent list.

The correlation between the maximum size of volumes and files and the field width of LHP and PHP is depicted in figure 3:

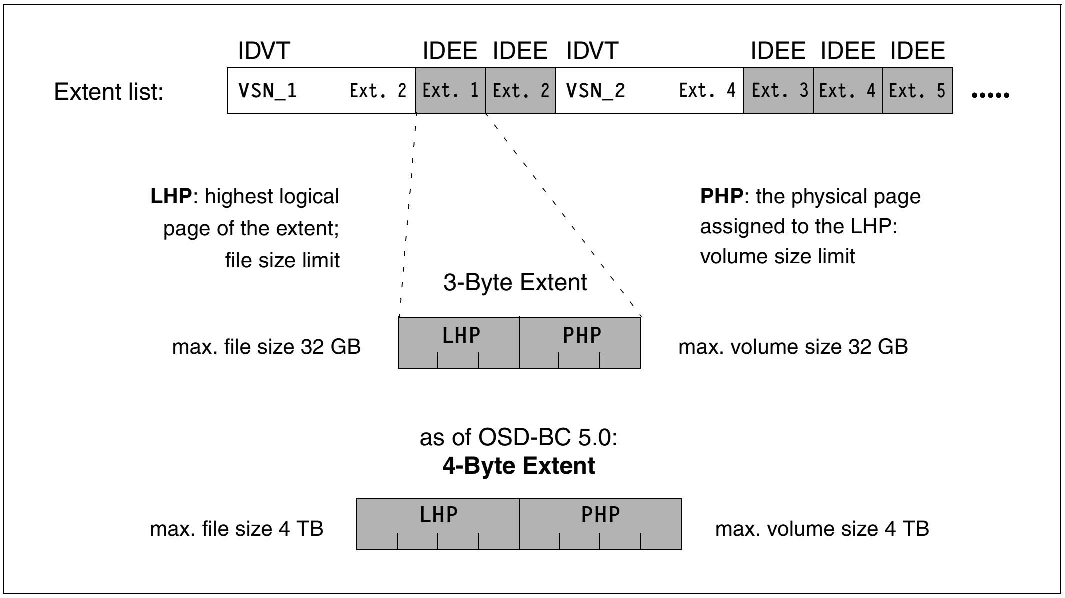


Figure 3: Correlation between file and volume sizes and the field width of LHP and PHP

This format cannot be interpreted in OSD-BC < V5.0. In order to achieve as much downward compatibility as possible, both formats of the extent list are supported as of OSD-BC V5.0.

– In general, the “old” format with 3-byte block numbers will be used.

– Only in the case of large files or of files that are wholly or partly located on large volumes, will the new format with 4-byte block numbers be used.

Extent lists therefore contain either extents with 3-byte block numbers or extents with 4-byte block numbers.

Further steps are taken to ensure that files with 4-byte extent lists cannot be accessed in OSD-BC < V5.0. This was achieved through the introduction in OSD-BC V5.0 of special pubset types (see page 12) to contain large volumes and large files, which cannot be imported into previous versions.

Restrictions for large files

- No large files can exist on the home pubset.
- The dump file $TSOS.SLEDFILE (SLED file) cannot be ≥ 32 GB.
- The paging file cannot be ≥ 32 GB.
- A SYSEAM file cannot be ≥ 32 GB.
- SIR does not support large files when reading from ARCHIVE tapes.
- Files where BLKCTRL=PAMKEY are not supported, since the logical page number is stored as a 3-byte field in the system section of the PAMKEY.

Summary of file management interfaces affected by 32 GB objects

Interface	**Change**
Privileged commands	
SET-PUBSET-ATTRIBUTES	Specification whether existing pubsets are to be upgraded to LARGE_OBJECTS pubsets with or without large files.
SET-PUBSET-ATTRIBUTES	Two additional S variables display the attributes LARGE-VOLUMES and LARGE-FILES: var(*LIST).LARGE-VOL and var(*LIST).LARGE-FILE
SHOW-MASTER-CATALOG-ENTRY	Output of LARGE_OBJECTS properties for LARGE_OBJECTS pubsets
Non-privileged commands	
ADD-FILE-LINK	Additional operand for large files
SHOW-FILE-LINK	Output of the “File can become large” attribute
SHOW-FILE-ATTRIBUTES	The length of different output fields has been extended
Macros	
FCB	Additional operand for large files
FILE	Additional operand for large files
FSTAT	Effort involved in check and conversion if VERSION=0/1
OPEN	Address problems of semantics
RDTFT	Output of the “File can become large” attribute
DIV	Additional operand for large files; extended range of values for BLOCK and SPAN
FPAMACC	Extended range of values for BLOCK
FPAMSRV	Additional operand for large files
STAMCE	Output of MRSCAT entries related to LARGE_OBJECTS

Table 3: Summary of file management interfaces affected by 32 GB objects

2.4 Catalog formats

Each pubset has a catalog in which the allocations of the files and the job variables in catalog entries are stored (TSOSCAT file catalog).

– On a standard pubset the catalog comprises a maximum of 8,192 4K blocks (NORMAL catalog format). This corresponds to approx. 60,000 - 80,000 files / JVs.

– On a LARGE_OBJECTS pubset the catalog comprises a maximum of 16,184 4K blocks (LARGE catalog format). This corresponds to approx. 120,000 - 160,000 files / JVs.

If this space has been exhausted, no new IDs can be configured in the pubset. It may also not be possible to create new files or enlarge files for existing users.

In BS2000/OSD-BC V6.0B the size of the catalog has been increased by 98% (15,808 additional catalog blocks for volume sets and SF pubsets). This extended catalog has the EXTRA LARGE catalog format. In order to be able to utilize the extended catalog and the larger data configuration which it permits, the catalog can optionally be converted to EXTRA LARGE format when the pubset is imported or when configuration takes place with SIR. The pubset can then no longer be imported into OSD-BC < V6.0B. Conversion back to the previous catalog format is not possible.

i LARGE-OBJECTS pubsets for large volumes or large files can be used independently of pubsets with the catalog format EXTRA LARGE. In other words a pubset with an EXTRA LARGE catalog by default also permits no large volumes and files. The pubset attributes for large objects must be assigned explicitly.

On an SM pubset with the EXTRA LARGE catalog format, the 3 special catalogs for the catalog entries for private disk/tape files of the job variables and of the migrated/unallocated files of the SM pubset can be extended by dynamic addition of further special catalogs. Up to 100 subcatalogs are possible in each case.

Details of the structure of the file catalog are provided in the “Introductory Guide to Systems Support” [5].

Automatic extension of the catalog

The CMS recognizes when a catalog is 90% full and automatically extends it provided this is possible without changing the catalog format.

For special catalogs of the EXTRA LARGE type, a new subcatalog is then created if none of the existing subcatalogs can be extended. Systems support can add new subcatalogs manually using the ADD-CATALOG-FILE command.

Outputting the current catalog situation

The SHOW-PUBSET-CATALOG-ALLOCATION command displays the catalog type (NORMAL, LARGE, EXTRA LARGE), the occupancy rate per catalog (utilization/file size) and the extendability per catalog. This output can be specified either for a single pubset or for all pubsets imported locally on a server and in master mode.

2.5 User programs for configurations with large files

As has already been mentioned, it cannot be assumed that all programs have been prepared for accessing large objects, i.e. that they can address 4-byte block numbers and block counters, although it must be borne in mind that the interfaces available to user applications only relate to accessing and processing files and their metadata. The following description is therefore confined to large files. Large disks will not be handled here, since there are no user interfaces available, as has been mentioned. Program behavior can thus be classified as follows:

Class A:	A program is able to process large files without restrictions. This behavior is defined as capable of LARGE_FILES (LARGE_FILES-capable).
Class B:	A program has not been explicitly prepared for processing large files and/or their metadata. It is, however, able to reject corresponding access that it regards as illegal, or alternatively, it does not access files or their metadata. This behavior is defined as compatible with LARGE_FILES (LARGE_FILES-compatible).
Class C:	A program has not been prepared for processing large files and cannot perform a defined rejection of corresponding access. This behavior is defined as incompatible with LARGE_FILES (LARGE_FILES-incompatible).

The corresponding classification for products of the software configuration of BS2000/OSD-BC V8.0 can be found from page 81.

For configurations that contain large files, programs that are compatible with or capable of LARGE_FILES must be presupposed. LARGE_FILES compatibility is to be regarded as the norm. Growth over 32 GB will initially be limited to a relatively small number of files. Only programs that access these must be capable of LARGE_FILES.

3 Systems support

This chapter outlines the role of systems support in introducing and maintaining large objects.

Emphasis is placed on the additional tasks in the field of “Pubset Management”.
But there are also various aspects of “Catalog and File Management” and “System Initialization and Parameter Service” that need to be addressed.

One section outlines the changes in the commands interfaces which result from the introduction of large objects.

In addition, adjustments to individual utilities that are important in the context of systems support and that have been affected by the extensions for large objects will be mentioned.

The end of the chapter provides a summary of potential sources of errors and conflicts.

3.1 Pubset Management

Systems support staff are responsible for the installation, reconfiguration and maintenance of pubsets, including LARGE_OBJECTS pubsets. Tools in the SIR utility (see page 25f) and the pubset management commands (see page 38ff) are available for this purpose.

A pubset that can accept large objects is known as a LARGE_OBJECTS pubset. Both SF and SM pubsets can possess the LARGE_OBJECTS property. In the case of SM pubsets, LARGE_OBJECTS is a property of the entire pubset, rather than of one or more volume sets.

The maximum size of a LARGE_OBJECTS pubset is 2 147 483 647 PAM pages or 4 TB. This limit is a result of the former field width for counters in the user catalog and for internally maintained counters for disk storage management.

Warning!

The existence of large files can in certain cases affect the executability of applications (see the section “Executability of utilities in environments with large objects”, on page 42, and the table “Executability of programs in environments with large files” in the appendix, on page 82).

Before a pubset is assigned attributes that allow it to support large files, it should be ensured that the software is configured accordingly. Appropriate measures should therefore be taken in advance to clarify such issues and make any necessary adjustments.

3.1.1 Installing and extending LARGE_OBJECTS pubsets with SIR

Volumes and files ≥ 32 GB can be allowed for SF and SM pubsets. These pubsets are then referred to as LARGE_OBJECTS pubsets.

Indicating whether large objects are permitted in a pubset

In order to allow large volumes and/or files, the SIR statement DECLARE-PUBSET provides two suboperands for creating and extending pubsets.

```
//DECLARE-PUBSET PUBSET-TYPE=...(...,ACTION=...(...,
                 LARGE-DISKS-ALLOWED=*NO/*YES(LARGE-FILES-ALLOWED=*NO/*YES)))
```

The LARGE-DISKS-ALLOWED=*NO/*YES operand determines whether or not volumes with a capacity ≥ 32 GB are allowed in a volume set of the pubset. The property “large volumes” does not depend on whether large files should be allowed or not.

The LARGE-FILES-ALLOWED=*NO/*YES operand determines whether or not files in the pubset may be ≥ 32 GB in size. Large files may only be stored in pubsets that support large volumes (LARGE-DISKS-ALLOWED=*YES).

If no explicit permission for large volumes and large files is entered in this operand, SIR creates a pubset that does not allow any large volumes/files (*NO is the default value), and thus generates a pubset format that is also compatible with earlier versions.

Warning!

Permission for large volumes and files cannot be removed from a pubset.
A pubset that permits large files and volumes is not compatible with OSD-BC versions earlier than V5.0.

SIR does not support large files when reading from ARCHIVE tapes.
Any large file on the input volume is ignored, the error message `SIR0728` is output, and the next file is processed.

```
SIR0728   COPYING OF FILE '(&00)' ABNORMALLY TERMINATED. FILE SIZE
          EXCEEDS THE LIMIT OF 32GB
```

Changes to the LARGE_OBJECTS attributes only take effect the next time the pubset is initialized (using the IMPORT-PUBSET command).

Removing inconsistencies

If a large volume is specified when a pubset is created (ACTION=*INSTALL), but LARGE-DISKS-ALLOWED=*YES was not specified, the volume is rejected with the message `SIR0308`.

```
SIR0308   DISK FOR VSN '(&00)' EXCEEDS CAPACITY OF 32 GB; DISK REJECTED
```

If this large volume is to be included in this pubset, the LARGE-DISKS-ALLOWED=*YES operand must be specified afterwards.

If a large volume is specified when a pubset is extended (ACTION=*EXTEND), but LARGE-DISKS-ALLOWED=*YES was not specified, the volume is rejected with the message `SIR0308` (see above).
If this large volume is to be included in this pubset, the pubset property must be changed with the SET-PUBSET-ATTRIBUTES ...,LARGE-VOLUMES=*ALLOWED command before SIR can add the volume.

3.1.2 Upgrading and extending existing pubsets

Indicating whether large objects are permitted in a pubset

Existing pubsets that have not been imported can be upgraded to large pubsets with and without large files using the SET-PUBSET-ATTRIBUTES command (see page 38).

Changes to the LARGE_OBJECTS attributes only take effect the next time the pubset is initialized (using the IMPORT-PUBSET command).

Adding large volumes to a pubset

Individual volumes can be added to an SF pubset or to a volume set in an SM pubset using the MODIFY-PUBSET-PROCESSING command; new volume sets can be added to an SM pubset in the same way.

If the volumes to be added are large volumes or if the (empty) volume sets that are to be added contain large volumes, the pubset being extended must be a LARGE_OBJECTS pubset. If this is not the case, the attempt to extend the pubset will be rejected with message `DMS1383` (with insert `'06'`):

```
DMS1383   VOLUME INCONSISTENT. ERROR TYPE '(&00)'. COMMAND REJECTED

          '06': A volume is larger than 32 GB, but the pubset doesn't have
                the attribute for large volumes.
```

The addition of a large volume in the context of IMPORT processing will only be successful if the pubset to be extended is a LARGE_OBJECTS pubset. If this is not the case, the attempt to extend the pubset will be aborted with message `DMS037E` (with insert `&00='04'`):

```
DMS037E   FORMAT OF VOLUME '(&02)' INCONSISTENT WITH PUBSET/VOLUMESET
          '(&00)'. IMPORT PUBSET TASK ABORTED WITH ERRORCODE '(&01)'

          '04': a volume is larger than 32 GB, but the pubset doesn't have
                the attribute for large volumes.
```

3.1.3 Importing LARGE_OBJECTS pubsets

LARGE_OBJECTS pubsets can only be imported into OSD-BC versions as of V5.0 (IMPORT-PUBSET command). It is not possible to import them into older versions, since utilization of this volume is made impossible by the version information in the standard volume label (SVL) of the PUBRES (for SF pubsets) or the VOLRES of the control volume set (for SM pubsets).

When a LARGE_OBJECTS pubset is imported, the attributes LARGE_OBJECTS and LARGE_FILES_ALLOWED (see page 12) are transferred from the DMS record of the SVL into the dynamic section of the MRSCAT. The layout of the MRSCAT has been extended to include these two attributes (among other things).

MRSCAT fields that were previously empty were used for this extension, which means that the length of the MRSCAT and its offsets have not changed. Thus, the layout of the MRSCAT is downward-compatible on both source and object level.

3.1.4 Generating SM pubsets with SMPGEN

The SMPGEN utility allows an SM pubset to be generated from several SF pubsets. If the SF pubsets include LARGE_OBJECTS pubsets, the resulting SM pubset will also be a LARGE_OBJECTS pubset.

If at least one of the SF pubsets that are being combined has the property LARGE_FILES_ALLOWED, then a LARGE_OBJECTS pubset with the property LARGE_FILES_ALLOWED is created.

3.1.5 Recovering SM pubsets with large objects

If an SM pubset is recovered that has a corrupt control volume set, the information determining whether the pubset is a LARGE_OBJECTS pubset must also be recovered.

The LARGE_OBJECTS attributes, therefore, are not only stored in the SVL of the VOLRES of the control volume set, but also in the SVLs of the other volume sets in the SM pubset.

OSD-BC V5.0 and higher offers the “Exclusive import of a pubset with pubset conversion” function for recovery purposes.
This uses a special IMCAT to create an SF pubset from a volume set in order to enable access to data stored on a volume set, regardless of the validity of the control volume set. When it is converted, the SF pubset inherits the LARGE_OBJECTS properties of the SM pubset.
When the SF pubsets taken from the volume sets are combined to form an SM pubset using SMPGEN, the LARGE_OBJECTS properties are transferred, as described in the section “Generating SM pubsets with SMPGEN” on page 27.

3.1.6 LARGE_OBJECTS pubsets in clusters

LARGE_OBJECTS can also be operated as shared pubsets, but all systems involved in sharing must be running a BS2000 version as of OSD-BC V5.0.

Remote import of LARGE_OBJECTS pubsets is possible on OSD-BC < V5.0. The functionality of the LCS cluster (see “HIPLEX MSCF” manual [7]) is then available for the pubset, with exception of remote access to the file catalog. Remote access to the file catalog is rejected with the error message `DMS0501`.

```
DMS0501   REQUESTED CATALOG NOT AVAILABLE OR WILDCARDS IN USER IDENTIFICATION
```

3.1.7 Large files in the POSIX file system

Previously, only files smaller than 2 gigabytes were supported in POSIX. This was because the data within a file was addressed with a variable of the data type integer (signed). This could only address a maximum of 2^{31}–1 bytes, i.e. 2 gigabytes.

This limit caused problems with different applications, for instance, with print files with memory-intensive graphics. Consequently, more users called for the maximum file size to be increased. This was also supported by the standardization authorities and led to a new standard being defined for large files.

Standard for large files

The central point of this standard is that a “long long” variable is used to address data within a file. This data type consists of an integer pair, which enables an address to be 2^{63}–1 bytes long, including the sign.

This new class of files should of course be as compatible as possible with the existing ones, so that existing programs can also work with large files without any great difficulty. That is to say, it should be possible to process the large files where possible with the same interfaces as previous files, at least in terms of syntax and semantics.

3.1.7.1 Large POSIX file systems

The maximum size of a POSIX file is also limited by the size of the file system in which it is located, i.e by the size of the container file in BS2000.

Since internal addressing now uses a long long-type variable, a much greater range can be addressed. Consequently, container files and thus POSIX file systems as well can be larger than 2 gigabytes.

The following limit values exist for BS2000 files and POSIX file systems:

Size of a BS2000 file	**Size of the POSIX file system**
max. 4096 gigabytes (= 4 terabytes)	max. 1024 gigabytes (= 1 terabyte)

The difference between the maximum size of BS2000 files and the maximum container size is determined by the nature of the implementation in POSIX.

The size of a container file and thus of a POSIX file system is defined when it is created with the administration tool POSINST (see the “POSIX Basics” manual [13]).

3.1.7.2 Large POSIX files

Large POSIX files are files of a POSIX file system which can be larger than 2 gigabytes. Large POSIX files can only be created in POSIX file systems which are based on a large container and can thus exceed the limit value of 2 gigabytes, see the previous section.

The maximum size of a POSIX file is limited by the size of the container file which contains it. You can also specify a maximum file size in POSIX, which applies to all files of the POSIX file system (command *ulimit* or parameter FILESIZE in the POSIX information file).

Program interfaces for large POSIX files

To work with POSIX files, there are a number of C-library functions, such as *open()*, *close()*, which are made available by CRTE. A subset of these functions is available in 64-bit form, so that they can process large POSIX files. These functions have the same name, with the additional suffix “64”, e.g. *open64()*. Some data structures and data types were also converted to 64-bit form.

For further information on the program interface, see the “POSIX Basics” manual [13].

Commands for large POSIX files

Most file processing commands of the POSIX shell can recognize and sometimes also process large POSIX files. They fall into two categories:

large file aware This command can process large POSIX files correctly. Some of the commands in this category can only process large files up to a certain file size, for instance *cpio* up to a maximum of 8 GB.

large file safe This command recognizes large POSIX files but rejects processing them, e.g. with an appropriate message.

For further information on the command interface, see the “POSIX Basics” manual [13].

3.2 Catalog and file management

Since OSD-BC V5.0 a version of the catalog entry has been available to support large objects. In some cases, the extensions have been introduced in a source-incompatible form in order to ensure that components produced on the basis of OSD-BC V5.0 and higher take into account the changes for “large objects”. Since the extension of the catalog entry and the corresponding access function for programming system exits may be of interest, a short summary of the changes is given below.

The catalog management caller interface has the new version number 5. The caller does not need to make any changes to the call, but must compile from scratch, if they want to process large objects.

3.2.1 Changes to the catalog entry (CE)

– The additional version identifier (X'FF03') has been introduced.

– An indicator (bit) has been introduced for identifying catalog entries with large objects (large extent list). The indicator identifies whether the format used is 3-byte or 4-byte.

– The description of the extent list for large objects uses 4-byte format; an extent list that does not belong to large objects remains in 3-byte format, under another name.
An additional substructure exists for the extent list (DCAEE4 or DCAEE4I), where the fields for the LHP and the PHP are each 4 bytes in length. The old substructure for the old extent list form (IDEE or DCMIDEET) with 3-byte fields has been retained.

– An additional 4-byte field has been introduced for the file size in the main part of the catalog entry, although the old 3-byte field has been retained (under another name) in the old part.

– The maximum size for extent lists has been extended to 2496 bytes.
For large files, the maximum size for extent lists has been extended to 2496 bytes.
A large file, like a small file, can thus contain a maximum of 310 extents.

– An additional 4-byte field has been introduced for the last halfpage pointer in the main part of the catalog entry, although the old 3-byte field has been retained (under another name) in the old part.

3.2.2 Assigning a 4-byte extent list to a file

There are various versions of the structure of the catalog entry within the LARGE_OBJECTS pubsets and in particular the extent list (3-byte and 4-byte variants). The user has no direct influence over which variant is assigned to a file. This happens implicitly via DMS, where a file is assigned a 4-byte extent list if:

– the size of the file exceeds the 32-GB limit, or

– the file is assigned an extent that is located on a volume with capacity ≥ 32 GB by the allocator.

4-byte extent lists are not converted back, even when the file becomes smaller than 32 GB or no longer has an extent on a large volume.

3.2.3 Creating SYSEAM files

The size of SYSEAM files must always be less than 32 GB.

The size of SYSEAM files is determined by specifications in the MRSCAT entry.
It is defined with the ADD-MASTER-CATALOG-ENTRY EAM=*PAR(...) command when the MRSCAT entry is created, and can be changed for an existing entry with the MODIFY-MASTER-CATALOG-ENTRY EAM=*PAR(...) command.

If no EAM specifications exist in an MRSCAT entry, the corresponding system parameters apply. Thus the maximum size specification possible (64512 units) guarantees that a SYSEAM file always remains smaller than 32 GB.
Furthermore, if a primary or secondary allocation is necessary, the appropriate parameterization within EAM ensures that a SYSEAM file cannot become larger than 32 GB.

The only way that a SYSEAM file can exceed the 32-GB limit is if it is explicitly created by systems support using FILE or CREATE-FILE and the corresponding file size.

The first EAM access to this file will however be rejected, and the message `DMS0854` will be output to the console.

```
DMS0854   FILE SIZE OF SYSEAM FILE (&00) MAY NOT EXCEED 32 GIGABYTES
```

The action associated with this message is to delete the corresponding SYSEAM file and to have systems support create the file again with a file size that is not critical.

3.2.4 Creating catalogs in the EXTRA LARGE format

Catalogs in the EXTRA LARGE format can manage a larger number of files and JVs than catalogs in the NORMAL format (in the case of a standard pubset) or LARGE format (in the case of a LARGE_OBJECTS pubset). The EXTRA LARGE catalog format is therefore particularly suitable for LARGE_OBJECTS pubsets in which the number of files and JVs can increase steeply. It can, however, also be used for standard pubsets. See also section “Catalog formats” on page 20.

The EXTRA LARGE catalog format is incompatible with OSD-BC < V6.0B. Catalogs are consequently not automatically created in this format, but only if it is explicitly requested when the pubset is configured or imported. Conversion back to the NORMAL or LARGE format is not possible.

Catalog format when configuring the pubset with SIR

The SIR statement DECLARE-PUBSET for configuring and extending pubsets provides the following operand to determine the catalog format:

```
//DECLARE-PUBSET PUBSET-TYPE=...(...,ACTION=...(...,
                 TSOSCAT-TYPE=*NORMAL/*EXTRA-LARGE)))
```

The TSOSCAT-TYPE=*NORMAL/*EXTRA-LARGE operand determines the catalog format of the pubset:

– The default is *NORMAL. This results in a standard pubset being assigned the NORMAL catalog format and a LARGE_OBJECTS pubset being assigned the LARGE catalog format. However, a LARGE_OBJECTS pubset is configured only if the LARGE-DISKS-ALLOWED=*YES operand is also specified explicitly (see section “Installing and extending LARGE_OBJECTS pubsets with SIR” on page 25).

– *EXTRA-LARGE assigns the pubset (both a standard and a LARGE_OBJECTS pubset) the EXTRA-LARGE catalog format.

The catalog format for a volume set can be determined using the same operand in the SIR statement BEGIN-VOLUME-SET-DECLARATION.

Catalog format when generating an SM pubset with SMPGEN

The SMPGEN utility routine enables an SM pubset to be generated from multiple SF pubsets. If at least one of the SF pubsets being combined has the EXTRA-LARGE catalog format, all the catalogs of the resulting SM pubset are assigned the EXTRA-LARGE catalog format. The catalogs are converted to the EXTRA LARGE format the first time the SM pubset is imported (irrespective of the EXTRA-LARGE-CAT-CONV operand in the IMPORT-PUBSET command).

Importing with conversion of the catalog format

When the pubset is activated, conversion to the EXTRA LARGE catalog format can be requested explicitly in the IMPORT-PUBSET command.
The current catalog format can be queried using the SHOW-PUBSET-CATALOG-ALLOCATION command.

3.3 System initialization and parameter service

3.3.1 Restrictions for home pubsets

LARGE_OBJECTS pubsets are valid as home pubsets, as long as they do not contain large files. Thus, the specification LARGE_FILES_ALLOWED=*YES is not permitted for any pubsets which are IPL-capable.
Startup of a home pubset with this pubset attribute will be aborted.

3.3.2 System parameter FST32GB

(Customer) applications that do not access the critical data fields FILE-SIZE and HIGHEST-USED-PAGE could also be affected by the changes introduced for the support of large files. The system parameter FST32GB is offered to aid migration.

FST32GB only affects the following FSTAT interfaces:

– Version=0 (corresponds to Version=710) when a fully-qualified filename is specified (although not when a file generation group is specified with GEN=YES)

– Version=1 (corresponds to Version=800), where the FNAM operand was not specified

For more information regarding FSTAT, see page 57ff.

FST32GB = 0 (default setting)

If at least one of the files in the set selected by FSTAT is ≥ 32 GB, the FSTAT call is rejected with the return code `X'00000576'`.

FST32GB = 1 (ignore overflow)

If at least one of the files in the set selected by FSTAT is ≥ 32 GB, an overflow of the 3-byte fields of the PHP in the extent list is always tolerated and no error message is issued. In the event of an overflow, the value X'FFFFFF' is assigned to the data fields that cannot be displayed.

Note

The system parameter FST32GB is not evaluated if the indicator LARGE_PUBSET_ACCESS=YES is set in the FSTAT program interface (see page 60).

3.4 Extended commands

This section outlines the changes in the command interfaces which result from the introduction of large objects. The changes can refer to the extensions to the syntax or to layout changes which occur when information is output.

Summary

Command/operand	**Change**
Privileged commands	
SET-PUBSET-ATTRIBUTES LARGE-VOLUMES= *UNCHANGED/*ALLOWED(LARGE-FILES= *UNCHANGED/*ALLOWED)	Additional operand; determines whether existing pubsets are to be upgraded to LARGE_OBJECTS pubsets with or without large files.
SET-PUBSET-ATTRIBUTES	Two additional S variables show the contents of the attributes LARGE-VOLUMES and LARGE-FILES: var(*LIST).LARGE-VOL and var(*LIST).LARGE-FILE
MODIFY-USER-PUBSET-ATTRIBUTES TOTAL-SPACE=*UNLIMITED	OSD-BC V6.0 and higher: Additional operand value which permits the user’s storage space quota for permanent storage space, for temporary storage space or for work files to exceed 4 TB (see PERM-, TEMP- or WORK-SPACE-LIMITS=*PARAMETERS(...))
IMPORT-PUBSET EXTRA-LARGE-CAT-CONV= *NO/*YES	OSD-BC V6.0 and higher: Additional operand which defines whether the catalog is to be converted to EXTRA LARGE format
SHOW-PUBSET-CATALOG-ALLOCATION	OSD-BC V6.0 and higher: Additional command which displays information about the catalog format, allocation and extendability of the catalogs
Non-privileged commands	
ADD-FILE-LINK EXCEED-32GB=*BY-PROGRAM/ *FORBIDDEN/*ALLOWED	Additional operand; determines whether the file size may exceed 32 GB.
SHOW-FILE-LINK	Additional S variable var(*LIST).EXC-32GB; shows the contents of the EXCEED-32GB attribute

Table 4: Commands (part 1 of 2)

Command/operand	**Change**
SHOW-FILE-ATTRIBUTES	Output fields for the number of PAM pages extended to 10 digits OSD-BC V6.0 and higher: If the total number of reserved PAM pages exceeds 32 GB, the information is displayed in units of one thousand PAM pages. Two additional S variables contain the number in units of one thousand PAM pages if the original S variable has reached the 2147483647 PAM pages: var(*LIST).MIGRATE-S1-RESERVED-T and var(*LIST).PUBSET-RESERVED-T
SHOW-JV-ATTRIBUTES	JV V14.0C and higher: Output field for the number of JVs extended to 6 digits
SHOW-MASTER-CATALOG-ENTRY	The LARGE_OBJECTS properties are output for LARGE_OBJECTS pubsets

Table 4: Commands (part 2 of 2)

3.4.1 SET-PUBSET-ATTRIBUTES / SHOW-PUBSET-ATTRIBUTES

Indicating whether large objects are permitted in a pubset

Existing pubsets can be upgraded to large pubsets with the SET-PUBSET-ATTRIBUTES command. The change takes effect the next time the pubset is initialized.

The following two operands determine whether large objects are permissible:

```
/SET-PUBSET-ATTRIBUTES ...,LARGE-VOLUMES=*UNCHANGED/*ALLOWED(
                           LARGE-FILES=*UNCHANGED/*ALLOWED)
```

The LARGE-VOLUMES operand determines whether the pubset may contain large volumes. The default setting *UNCHANGED specifies that the previous setting is retained. If *ALLOWED is specified, the pubset may contain large volumes. This setting cannot be undone and also means that the pubset cannot be imported into versions earlier than OSD-BC V5.0.

The LARGE-FILES operand determines whether large files may also be created on these large volumes.
The default setting *UNCHANGED specifies the previous setting is retained.
If *ALLOWED is specified, large files may be stored. This setting cannot be undone.

Changes to the LARGE_OBJECTS attributes only take effect the next time the pubset is initialized (using the IMPORT-PUBSET command).

Outputting pubset attributes

Pubset attributes are output with the SHOW-PUBSET-ATTRIBUTES command.

Example of the output of /SHOW-PUBSET-ATTRIBUTES for the WORK pubset

```
/show-pubset-attributes work
%
%==============================================================================
% PVSID   SYSID   SHARABILITY   CURRENT   DESIGNATED   BACKUP      ALTERNATE
%                               MASTER      MASTER     MASTER      BACKUP
%------------------------------------------------------------------------------
% WORK     250       *YES       *NONE       *NONE      *NONE         *NONE
%==============================================================================
%                               LARGE                  LARGE
%                               VOLUMES                FILES
%------------------------------------------------------------------------------
%                               *ALLOWED               *ALLOWED
%==============================================================================
```

In the S variable output of the SHOW-PUBSET-ATTRIBUTES command, these specifica-tions are shown in the following two S variables:

– `var(*LIST).LARGE-VOL`
– `var(*LIST).LARGE-FILE`

Contents of both S variables: *NOT-ALLOW or *ALLOW.

Example of the output to the TESTOUT S variable

```
/decl-var testout(type=*struct),mult-elem=*list
/exec-cmd cmd=(show-pub-attr work),text-output=*no,struct-output=testout
/sh-var testout
TESTOUT(*LIST).PUBSET = WORK
TESTOUT(*LIST).PUBSET-TYPE = *STANDARD
TESTOUT(*LIST).CONTROL-VOLSET =
TESTOUT(*LIST).SYS-ID = 250
TESTOUT(*LIST).SHARE = *YES
TESTOUT(*LIST).LARGE-VOL = *ALLOW
TESTOUT(*LIST).LARGE-FILE = *ALLOW
TESTOUT(*LIST).CURR-MASTER = *NONE
TESTOUT(*LIST).DESIGNATED-MASTER = *NONE
TESTOUT(*LIST).BACKUP-MASTER = *NONE
TESTOUT(*LIST).ALT-BACKUP = *NONE
```

3.4.2 MODIFY-USER-PUBSET-ATTRIBUTES

As the maximum size of a volume set is 4 TB, the maximum total size of an SM pubset can be 255 * 4 TB, in other words a little over 1,000 TB. In OSD-BC V6.0B and higher a user can occupy more than 4 TB of storage space on an SM pubset. The MODIFY-USER-PUBSET-ATTRIBUTES command permits storage space quotas > 4 TB:

```
/MODIFY-USER-PUBSET-ATTRIBUTES ...,
             PERM-SPACE-LIMIT=*PAR(TOTAL-SPACE=*UNLIMITED,...),
             TEMP-SPACE-LIMIT=*PAR(TOTAL-SPACE=*UNLIMITED,...),
             WORK-SPACE-LIMIT=*PAR(TOTAL-SPACE=*UNLIMITED,...),
```

Specifying TOTAL-SPACE=*UNLIMITED determines that the specified storage space quota may exceed the limit of 4 TB. This operand can be specified separately for the permanent storage space (PERM-SPACE-LIMIT operand), for the temporary storage space (TEMP-SPACE-LIMIT operand) or for the storage space for work files (WORK-SPACE-LIMIT operand).

Information on allocated storage space quotas is displayed by the SHOW-USER-ATTRIBUTES command with INFORMATION=*PUBSET-ATTRIBUTES or *PUBSET-SUMMARY:

– The output fields PERM-SPACE-LIMIT, TEMP-SPACE-LIMIT and WORK-SPACE-LIMIT contain *UNLIMITED if the relevant quota may exceed the limit of 4 TB.

– The S variables var(*LIST).PERM-TSL, var(*LIST).TEMP-TSL and var(*LIST).WORK-TSL show this specification with the *UNLIM value.

3.4.3 IMPORT-PUBSET / SHOW-PUBSET-CATALOG-ALLOCATION

Conversion when a pubset is imported

When a pubset is imported in OSD-BC ≥ V6.0B, the catalog can be converted to EXTRA LARGE format.

```
/IMPORT-PUBSET PUBSET=...,EXTRA-LARGE-CAT-CONV=*NO/*YES,...
```

The EXTRA-LARGE-CAT-CONV operand specifies whether conversion to EXTRA LARGE format is to take place when a pubset is imported:

- The default is *NO, i.e. the existing catalog format is retained.
- The *YES option requests conversion. After it has been converted, the pubset cannot be imported to OSD-BC < V6.0B again.

Displaying the current catalog format

The SHOW-PUBSET-CATALOG-ALLOCATION command provides information on the catalog format of a pubset, its catalogs, how full these catalogs are and whether the catalogs can be extended.

```
/SHOW-PUBSET-CATALOG-ALLOCATION PUBSET=*ALL/<catid>
```

*ALL causes the information for all pubsets which the home computer has imported exclusively or as the master to be displayed. If a catalog ID is specified, only the information for the pubset specified is requested.

The information on the catalog format is displayed in the output field `CATALOG-FORMAT` or in the S variable `var(*list).FORMAT`.

3.4.4 Non-privileged commands

The changes to the following non-privileged commands are described in Chapter 4 “Users and programmers” as of page 47.

ADD-FILE-LINK	Description on page 49
SHOW-FILE-ATTRIBUTES	Description on page 50
SHOW-FILE-LINK	Description on page 49
SHOW-MASTER-CATALOG-ENTRY	Description on page 49

3.5 Executability of utilities in environments with large objects

This section will refer to a few utilities that are important in the area of systems support. Once again, only the modifications for large files and volumes will be described.

A full description can be found in the following manuals: “HSMS” [8], “FDDRL” [6], “SPACEOPT” [14], “IMON” [9] and - for all the other utilities - the “Utilities” manual [2].

With regard to SIR, see page 25 and with regard to SMPGEN, see page 27.

3.5.1 HSMS / ARCHIVE

In OSD-BC V5.0 and higher the catalog extension for large objects requires the use of the product versions HSMS V6.0 or higher and ARCHIVE V6.0 or higher. Catalog entries for small files are converted into the format of the previous version, so that downward-compatibility is achieved to OSD-BC V3.0.

Compatibility

A set of backup data that consists exclusively of small files can be used in older BS2000/OSD versions without difficulty.

When using RESTORE with a backup set that also contains large files, the large files can only be restored in OSD-BC V5.0 or higher and then only on LARGE_OBJECTS pubsets with LARGE-FILES-ALLOWED.
In all other cases - i.e. when attempting to restore to pubsets with other properties and/or older OSD-BC versions - HSMS and ARCHIVE skip all large files and output an ARC0061 message for each file.

```
ARC0061   FILE GREATER THAN 32GB. FILE NOT PROCESSED
```

Older product versions of HSMS (V4.0 or higher) and ARCHIVE (V4.3 or higher) recognize large files and reject them in a defined manner.

The file entries (F records) in the ARCHIVE directory have been extended to allow upward compatibility and provide support for large files.

SAVEFILES on the S1 level

S1 SAVEFILES are possible on large disks and/or as large files for migration. The selection of a configuration of this type cannot be recommended without reservation, since an import into older BS2000/OSD versions is not possible.

Incremental saves and partial saves

When performing incremental saves and partial saves, the situation can occur that older file versions in the backup set are small files, whereas newer versions are large files. If a backup set of this kind is restored in an environment where large files are not permitted, large files are rejected by ARCHIVE with the message `ARC0061` (see page 42).

RESTORE with K → NK conversion

When files with BLKCTRL=PAMKEY are restored to an environment with NK disks, they are automatically converted to a suitable NK format (DATA or NO) by the PAMINT subsystem. In the context of this conversion, the size of the converted file can exceed 32 GB. PAMINT supports large files. Thus a conversion can also be performed in this case, provided that a LARGE_OBJECTS pubset with LARGE-FILES-ALLOWED is specified.

3.5.2 FDDRL

FDDRL V14.0 or higher allows large volumes to be saved. However, large volumes can only be processed under OSD-BC ≥ V5.0.

3.5.3 VOLIN

VOLIN V14.0 or higher is capable of initializing disks with a net capacity ≥ 32 GB. When a large disk is initialized, the version field in the base record of the SVL is set to a value that corresponds to OSD-BC V5.0. This prevents the utilization of large disks in older versions.

It should be noted that large volumes can only be components of a LARGE_OBJECTS pubset.
Large volumes are not permitted as private disks. Any attempt to initialize a large volume as a private disk will be rejected with the message `NVL0146` :

```
NVL0146   DISK CAPACITY GREATER EQUAL 32GB IS NOT PERMITTED FOR PRIVATE DISKS
```

3.5.4 PVSREN

It is possible to rename LARGE_OBJECTS pubsets using PVSREN V1.4 or higher.

3.5.5 SPACEOPT

SPACEOPT V2.0 or higher supports the reorganization of large volumes and large files.

Large objects are not visible at the SPACEOPT command interface. The number of digits in some fields when outputting to SYSOUT and SYSLIST has been extended in order to take account of the new maximum values for block numbers and file and volume sizes.

3.5.6 DPAGE

Large objects, i.e. both large files and large volumes, can be processed using DPAGE V14.0 or higher. Block numbers up to 2 147 483 687 can be entered, and the layout of the output to SYSOUT has been extended accordingly.

3.5.7 SPCCNTRL

SPCCNTRL V14.0 or higher can display information about large files and volumes. The corresponding output fields (output to SYSOUT) have been extended.

Exception

If, in the DISPLAY CATALOG,ENTRY statement, ENTRY refers to a file ≥ 32 GB, and if the user is not the system administrator (TSOS), the information cannot be returned. The request is rejected with the message `SPC0030`:

```
SPC0030   ERROR EXECUTING (&00)-MACRO, RETURN CODE = (&01)
```

3.5.8 IMON

The installation monitor IMON uses the SIR COPY function to read in files of the delivery package (in ARCHIVE import format, but without the ARCHIVE utility that must be purchased separately). Large files cannot be read in in this way.

Initial installation with FIRST

The boot tape, released for initial installation of OSD-BC on /390 systems and for disasters following complete disk failure, creates a BS2000 starter system only on disks that are smaller than 32 GB.
(This can, of course, be extended to larger disks at a later stage.)

3.6 Potential sources of errors and conflicts

3.6.1 Restrictions for LARGE_OBJECTS pubsets

LARGE_OBJECTS pubsets can only be imported into OSD-BC versions as of V5.0 (IMPORT-PUBSET command). It is not possible to import into older versions, since utilization of this volume is made impossible by the version information in the standard volume label (SVL) of the PUBRES or the VOLRES of the control volume set.

Shared pubsets

LARGE_OBJECTS pubsets can, of course, also be operated as shared pubsets.
The extended catalog structure means that shared pubset clusters are not possible with OSD-BC versions earlier than V5.0.

Home pubsets

LARGE_OBJECTS pubsets without large files are valid as home pubsets.
Thus, the specification LARGE_FILES_ALLOWED=*YES is not permitted for any pubsets which are IPL-capable.
Startup of a home pubset with this pubset attribute will be aborted.

3.6.2 Restrictions for system files

SYSEAM files

The size of SYSEAM files must always be less than 32 GB.

The only way that a SYSEAM file can exceed the 32-GB limit is if it is explicitly created by systems support using the FILE macro or the CREATE-FILE command and the corresponding file size.
The first EAM access to this file is, however, rejected. The file must be deleted by systems support and newly created with the permitted size (see also page 32).

Dump and paging files

The dump file $TSOS.SLEDFILE (SLED file) cannot be ≥ 32 GB.

The paging file cannot be ≥ 32 GB.

3.6.3 Restrictions for large volumes

Access to large volumes is only possible as of OSD-BC V5.0. Large volumes can only be added to LARGE_OBJECTS pubsets.

Large volumes are not allowed as private disks. Any attempt to initialize a large volume as a private disk will be rejected with the following error message:

```
NVL0146   DISK CAPACITY GREATER EQUAL 32GB IS NOT PERMITTED FOR PRIVATE DISKS
```

4 Users and programmers

This chapter contains detailed descriptions of the extensions made to non-privileged command interfaces and Assembler macro interfaces in relation to large files.

This is followed by notes on RFA and high-level programming languages that are affected by large objects.

The end of the chapter summarizes potential sources of errors and conflicts.

4.1 Extended user commands

This section outlines the changes in the non-privileged command interfaces which result from the introduction of large objects. The changes can refer to extensions to the syntax or to changes to the layout of output information.

Summary

Command/operand	**Content of the change**
ADD-FILE-LINK EXCEED-32GB=*BY-PROGRAM/ *FORBIDDEN/*ALLOWED	This operand determines whether the file size may exceed 32 GB
SHOW-FILE-LINK	The S variable var(*LIST).EXC-32GB; displays the contents of the EXCEED-32GB attribute
SHOW-FILE-ATTRIBUTES	Output fields for number of PAM pages extended to 10 digits OSD-BC V6.0 and higher: If the total number of reserved PAM pages exceeds 32 GB, the information is displayed in units of one thousand PAM pages. Two additional S variables contain the number in units of one thousand PAM pages if the original S variable has reached the 2147483647 PAM pages: var(*LIST).MIGRATE-S1-RESERVED-T and var(*LIST).PUBSET-RESERVED-T
SHOW-JV-ATTRIBUTES	JV V14.0C and higher: Output field for the number of JVs extended to 6 digits
SHOW-MASTER-CATALOG-ENTRY	The two S variables: var(*LIST).LARGE-VOL and var(*LIST).LARGE-FILE display the contents of the attributes LARGE-VOLUMES and LARGE-FILES

Table 5: User commands

4.1.1 SHOW-MASTER-CATALOG-ENTRY

The SHOW-MASTER-CATALOG-ENTRY command provides information on the contents of MRSCAT entries.

Outputting pubset attributes

If the pubset is a LARGE_OBJECTS pubset, these properties will be output by the SHOW-MASTER-CATALOG-ENTRY command.

Example of the output of /SHOW-MASTER-CATALOG-ENTRY for the WORK pubset

```
/show-master-catalog-entry work
%PUBSET WORK: SINGLE-FEATURE, LOCAL-IMPORTED, K-FORMAT, LARGE_OBJECTS,
              SHARED, MASTER-HOST=OWN-HOST
```

4.1.2 ADD-FILE-LINK / SHOW-FILE-LINK

Indicating whether large files are permitted

Access to large files can be controlled with the ADD-FILE-LINK command and the operand EXCEED-32GB, without intervention in the program.

```
/ADD-FILE-LINK SUPPORT=*DISK(EXCEED-32GB=*BY-PROGRAM/*FORBIDDEN/*ALLOWED,...)
```

The EXCEED-32GB operand defines whether the processing of large files is allowed (*ALLOWED) or not (*FORBIDDEN).
The specifications in the TU FCB are used for the default setting *BY-PROGRAM.
This operand is entered in the TFT (Task File Table) and is evaluated when the file is opened.

Outputting file attributes

In the S variable output of the SHOW-FILE-LINK command, the validity of large files as specified in the TFT is output in the S variable `var(*LIST).EXC-32GB` (declared using the ADD-FILE-LINK command or the FILE macro).

Contents of the S variables: ” (corresponds to BY-PROGRAM) or *FORBID or *ALLOW.

4.1.3 SHOW-FILE-ATTRIBUTES

Outputting file size

The output fields of the SHOW-FILE-ATTRIBUTES command that relate to the output of the number of PAM pages have been adapted to allow for possible file sizes ≥ 32 GB. Thus the number of output digits has been extended to 10, although leading zeros are not displayed.

The extension concerns the following output fields:

– File size in the header (number of PAM pages reserved for the file)
– HIGH-US-PA for the file properties (number of PAM pages occupied by the file)

The following output fields in the totals lines have also been extended to 10 digits:

– FRE (the total PAM pages that have been reserved for but are not yet occupied by the displayed output set)
– REL (the total PAM pages that can be released for the displayed output set)
– RES (the total PAM pages that are reserved for the displayed output set)

The following applies for OSD-BC V6.0B and higher:

– Output field RES in the totals line:
  When there are more than 2147483647 reserved PAM pages, the display uses units of one thousand PAM pages (8 digits and the right-justified identifier “T”).

– S variables `var(*LIST).MIGRATE-S1-RESERVED` and `var(*LIST).PUBSET-RESERVED`:
  When the value of 2147483647 PAM pages is reached, the corresponding S variable `var(*LIST).MIGRATE-S1-RESERVED-T` or `var(*LIST).PUBSET-RESERVED-T` is assigned the current value in units of one thousand PAM pages.

The modification have been implemented for all output destinations (OUTPUT=*SYSOUT/*SYSLST/*PRINTER/<filename>).

4.1.4 SHOW-JV-ATTRIBUTES

When the SHOW-JV-ATTRIBUTES command is issued, the total number of JV entries output is displayed in the SUM output field in the totals line.

The EXTRA LARGE catalog format which is supported in OSD-BC V6.0B and higher also permits more JV entries. In JV V14.0C and higher the sum of the JVs displayed is consequently shown with 6 digits.

4.2 Extended Assembler macros

This section outlines the changes in the Assembler macro interfaces which result from the introduction of large objects. The changes can refer to the extensions to the syntax or to layout changes which occur when information is output.

The macros mentioned are described in the following manuals: “Executive Macros” [11] (STAMCE macro) or “DMS Macros” [3] (all other macros)

Summary of Assembler macros

Macro/operand or RC	Content of the change	Page
Executive macros		
STAMCE SELECT= LARGE_OBJECTS/ LARGE_FILES_ALLOWED	Extension of the display to include the LARGE_OBJECTS and LARGE_FILES_ALLOWED attributes	53
DMS macros		
FSTAT VERSION=0/1 X'0576'	If FSTAT is called with these versions, you must check whether it is necessary to convert to VERSION=2/3. Extended return code for large files	57
OPEN X'0D9D' X'0D00'	Be aware of problems regarding semantics Additional return code Additional return code	61
FCB LARGE_FILE= *FORBIDDEN/*ALLOWED	Additional operand; only for disk files: LARGE_FILE specifies whether the file may become a “large” file (i.e. whether its size may increase beyond 32 GB during processing).	63
FILE EXC32GB= *FORBIDDEN/*ALLOWED	Additional operand; only for disk files; determines whether or not the file size may exceed 32 GB during processing	64
RDTFT	Additional information about the file property EXC32GB in the output area	65

Table 6: Assembler macros (part 1 of 2)

Macro/operand or RC	Content of the change	Page
DMS macros for special access methods		
DIV		66
LARGE_FILE= *FORBIDDEN/*ALLOWED	Additional operand for FCT=*OPEN: LARGE_FILE specifies whether the file to be opened may become a “large” file that can exceed 32 GB in size.	
OFFSET=number	Changed operand for FCT=*MAP/*SAVE/*RESET: Specifies the first block of the file region to be mapped in virtual address space. The value for OFFSET is limited by the maximum size of a file, thus; – 8388606 when LARGE_FILE= *FORBIDDEN or. – 1073741823 when LARGE_FILE=*ALLOWED.	
SPAN=number	Changed operand for FCT=*SAVE/*RESET: Specifies the length of the file region in 4-KB blocks. The value for SPAN is limited by the maximum size of a file, thus; – 8388607 when LARGE_FILE=*FORBIDDEN or – 1073741824 when LARGE_FILE=*ALLOWED.	
X'000C' X'0030'	Additional return code INVALID_LARGE_FILE Additional return code LARGE_FILE_NOT_SPECIFIED	
FPAMSRV	Address problems of semantics	70
LARGE_FILE= *FORBIDDEN/*ALLOWED	Additional operand for FCT=*OPEN: LARGE_FILE specifies whether the file to be opened may become a “large” file that can exceed 32 GB in size.	
X'0015'	Additional return code INVALID_LARGE_FILE	
FPAMACC		72
BLOCK=number	Changed operand Specifies the direct decimal numeric value for the number of the first logical block to be transferred. The value for BLOCK is limited by the maximum size of a file, thus; – 8388606 when LARGE_FILE=*FORBIDDEN or – 1073741823 when LARGE_FILE=*ALLOWED. (specified in the FPAMSRV macro)	
X'0145'	Additional return code LARGE_FILE_NOT_SPECIFIED	

Table 6: Assembler macros (part 2 of 2)

4.2.1 STAMCE

Outputting pubset attributes

The STAMCE macro provides information about the contents of the MRSCAT entries and also displays the LARGE_OBJECTS attributes.

Extract from the STAMCE DSECT

```
              STAMCE MF=D,PREFIX=D,XPAND=MCE,VERSION=5
:
:
:
2 *
2 DMCFDATT   DS    X        attribute
2 DMCFDLOB   EQU   X'40'    set: large_objects
2 *                              files/volumes with more than 32 GB
2 DMCFDLFA   EQU   X'20'    set: large_files_allowed
2 DMCFDRAI   EQU   X'10'    set: pubset with RAID volumes
2 DMCFDGSV   EQU   X'08'    set: gs volumes
2 DMCFDDRV   EQU   X'02'    set: high availability by DRV
2 DMCFDKEY   EQU   X'01'    set: key pubset
2 *
:
```

Example

The output of attributes addressed with DMCFDATT is laid out in such a way that in the output columns, **YES** stands for “allowed/existing” and **NO** for “not allowed/non existent”. The LARGE_OBJECTS attributes are output in the LOB and LFA columns.

```
STAMCE   START
         PRINT  NOGEN
         BALR   10,0
         USING  *,10
         USING  DSTAM3,6
         USING  DSTAM4,7
         STAMCE MF=E,PARAM=STAM3,VERSION=5
         LR     7,1
         L      6,DMCEAREA
         WROUT  HEADER,WROUTERR
SHOW     MVC    ATTRIB,=C'NO  NO  NO  NO  NO  NO  '
         MVC    CATID,DMCFSCTD
         MVC    PROCESS,DMCFSBCA
TLOB     TM     DMCFDATT,DMCFDLOB
         BZ     TLFA
         MVC    LOB,YES
TLFA     TM     DMCFDATT,DMCFDLFA
         BZ     TRAI
         MVC    LFA,YES
TRAI     TM     DMCFDATT,DMCFDRAI
         BZ     TGSV
         MVC    RAI,YES
TGSV     TM     DMCFDATT,DMCFDGSV
         BZ     TDRV
         MVC    GSV,YES
TDRV     TM     DMCFDATT,DMCFDDRV
         BZ     TKEY
         MVC    DRV,YES
TKEY     TM     DMCFDATT,DMCFDKEY
         BZ     TEND
         MVC    KEY,YES
TEND     NOP    TEND
         CLI    PROCESS,X'00'
         BNE    WROUT2
         MVC    PROCESS,=CL8' '
WROUT2   WROUT  OUTREC,WROUTERR
         LA     6,DMCF#(6)
         CLI    DMCFSCTD,' '
         BNE    SHOW
WROUTERR TERM
```

```
*** DEFINITIONS
STAM3    STAMCE CATID=' ',MF=L,VERSION=5
HEADER   DC     Y(HEADERE-HEADER)
         DC     CL2' '
         DC     CL1' '
         DC     C'CATID  PROCESSOR LOB LFA RAI GSV DRV KEY '
HEADERE  EQU    *
OUTREC   DC     Y(OUTRECE-OUTREC)
         DC     CL2' '
         DC     CL1' '
CATID    DS     CL4
         DC     CL3' '
PROCESS  DS     CL8
         DC     CL2' '
ATTRIB   DS     0CL24
LOB      DS     CL4
LFA      DS     CL4
RAI      DS     CL4
GSV      DS     CL4
DRV      DS     CL4
KEY      DS     CL4
OUTRECE  EQU    *
YES      DC     C'YES’
         LTORG
         DS     0F
OUT      DS     XL4096
DSTAM3   STAMCE MF=D,PREFIX=D,XPAND=MCE,VERSION=5
DSTAM4   STAMCE MF=D,PREFIX=D,XPAND=PL,VERSION=5
         END
```

Print-edited output

```
CATID  PROCESSOR LOB LFA RAI GSV DRV KEY
AARZ             NO  NO  NO  NO  NO  NO
BAU3   KAROBUBE  NO  NO  NO  NO  NO  NO
BLAU             NO  NO  NO  NO  NO  NO
BXT2             YES NO  NO  NO  NO  NO
 :
 :
WORK             YES YES NO  NO  NO  YES
2ATS   PIKASS    NO  NO  YES NO  NO  YES
2CVC   KAROBUBE  NO  NO  NO  NO  NO  NO
4ARB   D015ZE08  NO  NO  NO  NO  NO  YES
 :
 :
```

4.2.2 FSTAT

If the FSTAT macro accesses pubsets with large volumes that do not, however, allow large files, interface behavior is unchanged. Access of this kind is always performed without problems.
Problems can occur if pubsets that also allow large files are accessed.

The FSTAT macro offers the following interface variants:

(a)	Version=0	(corresponds to Version 710, default)	
(b)	Version=1	(corresponds to Version 800)	as of BS2000 version V8.0
(c)	Version=2		as of OSD-BC version V1.0
(d)	Version=3		as of OSD-BC version V3.0

Interface variants that do not need to be converted

If the following variants of FSTAT calls are used exclusively, no incompatibilities occur:

```
FSTAT ...,VERSION=2/3
```

The variants (c) and (d) return the relevant data (extent lists and data fields for File-Size and Last-Page-Pointer) as 4-byte fields. These interfaces do not need to be converted, and they are not affected by the comments below.
The semantics problem discussed in section “OPEN” on page 62, however, must be considered. Each user of this interface must check whether this problem applies to their implementation.

The following variants are not affected either:

– `FSTAT ...,VERSION=0,<partially-qualified filename>`
  `FSTAT ...,VERSION=0,<file generation group with GEN=YES>`

– `FSTAT ...,VERSION=1,FNAM`

Interface variants that need to be checked/converted

In the following section, the remaining variants of (a) and (b) will be examined more closely:

```
FSTAT ...,VERSION=0/1,SHORT/LONG
```

These variants return the catalog information in BS2000 V10.0 format.
The extent lists and data fields for File-Size and Last-Page-Pointer are output with only 3 bytes. For reasons of compatibility, it is not possible to change the layout of these interfaces.

Calls that refer to large objects cause an overflow in the 3-byte fields.

Two different cases must be distinguished. These depend on the contents of the result list from the FSTAT call:

a) No file ≥ 32 GB exists in the result list (set of selected files).

In this instance, FSTAT tolerates the overflow of the 3-byte data field of the PHP in the extent list. The value X'FFFFFF' is assigned to the PHPs that cannot be displayed. It is assumed that the PHPs are never or very rarely evaluated at the interfaces. If this is occasionally not the case, the interface must be converted to version 2 or 3.

This achieves the following:

– The introduction of large volumes can be carried out in a compatible manner, user programs do not need to be changed.

– FSTAT calls with fully-qualified pathnames can be supported compatibly (except for the PHP overflow in the extent list).

▶ No conversion is necessary.

b) At least one file ≥ 32 GB exists in the result list (FSTAT is called with a partially-qualified filename or with wildcards).

Calls of this kind are rejected with the following return code:

X'cc'	X'bb'	X'aaaa'	Explanation
X'00'	X'01'	X'0576'	The selection contains large files.

The following types of FSTAT interfaces are affected:

Type		Comment
I	`FSTAT ...`	VERSION=0 is set as default
II	`FSTAT ...,VERSION=0`	
III	`FSTAT ...,VERSION=1,SHORT`	
IV	`FSTAT ...,VERSION=1,LONG`	
V	`FSTAT ...,VERSION=1`	The SHORT operand is set as default

▶ These calls must be converted to VERSION=2/3.

Summary of FSTAT calls

```
FSTAT <fully-qualified pathname>,VERSION=0/1
```

1. Access only to small files (less than 32 GB):
   No action necessary [1)]
2. Large files are also accessed (≥ 32 GB):
   If the call is made with one of the types `I` to `V`, conversion to VERSION=2/3 is necessary, with the exception of types `I` and `II` where the file generation group and GEN=YES are specified.

```
FSTAT <partially-qualified pathname or pathname with wildcards>,VERSION=0/1
```

1. The result list contains only small files (less than 32 GB):
   No action necessary [1)]

   Warning!

   You must ensure that the requirements are always fulfilled.
   This is probably only possible for pathnames that contain wildcards in the catalog ID and otherwise contain fully-qualified components.
2. The result list can also contain large files (≥ 32 GB):
   If the call is made with one of the types `III` to `V`, conversion to VERSION=2/3 is necessary.

Conclusion

When using the FSTAT interface with VERSION < 2 and FORM=LONG or FORM=SHORT, it is necessary to convert to the most recent interface version under the following circumstances:

a) Large files should be accessible with the FSTAT call in the affected program.

b) The pathname in the FSTAT call is partially-qualified or contains wildcards and it cannot be assumed that the result list contains no large files. Calls with wildcards in the catalog ID are particularly critical here.

It may be necessary to convert the data structures in the program as well as the interface.

[1] “No action necessary” applies here if the result list contains only small files and the overflow of the 3-byte data field of the PHP in the extent list does not present a problem. If this is not the case, conversion to VERSION=2/3 is necessary.

Control using the system parameter FST32GB

FST32GB only affects the following FSTAT interfaces:

– Version=0 (corresponds to Version=710) when a fully-qualified filename is specified (although not when a file generation group is specified with GEN=YES)

– Version=1 (corresponds to Version=800), where the FNAM operand was not specified

Whether the existence of a file ≥ 32 GB in the list of selected files leads to the FSTAT call being rejected with the `X'00000576'` return code (FST32GB=0, default setting) or whether an overflow of the 3-byte fields is always tolerated (FST32GB=1) is set globally on the system by systems support staff. In the latter case, the value X'FFFFFF' is assigned to the data fields that cannot be displayed.

Note

The system parameter FST32GB is not evaluated if the FSTAT indicator (see below) is set.

Control using the FSTAT indicator

Behavior equivalent to FST32GB=1, i.e. ignoring the overflow, can be activated for specific interfaces for FSTAT types `I` to `V` using an indicator.

Note

The FSTAT indicator has a higher priority than the system parameter FST32GB. If the FSTAT indicator is set, FST32GB is not evaluated.
The indicator must be set directly in the parameter list; no support is provided in the FSTAT macro.

Description of the bits in the corresponding DSECT:

```
         FSTAT MF=D,PARMOD=31,VERSION=710
IDBFLAG2 DS    X                               FLAGS 2
IDBLOPYE EQU   X'04'                           2-2 S LARGE PUBSET ACCESS=YES


         FSTAT MF=D,PARMOD=31,VERSION=800
IFLAG0   DC    B'10001100'
ILOPY    EQU   X'04'                           2-2 S LARGE PUBSET ACCESS=YES
```

4.2.3 OPEN

The OPEN macro and the access methods are only affected by the introduction of large files, not by the introduction of large volumes.

The OPEN interface checks whether files are permitted to extend beyond 32 GB and whether the creation of files ≥ 32 GB and access to such files are permitted.

There are two aspects associated with this:

a) Rejection of access to or the creation of large files for access methods that do not allow processing of large files.

b) Indication that a program can create or open files ≥ 32 GB.

Incompatible interface variants

Interfaces where 3-byte block numbers are used are never able to work with files ≥ 32 GB. This affects the following cases:

– All files in key format (BLKCTRL=PAMKEY):
  The logical block numbers in the PAMKEY are only 3 bytes wide.

– 24-bit interface of UPAM
  The field for logical block numbers in the UPAM parameter lists and in the TU FCB is only 3 bytes wide.

– 24-bit interface of SAM
  The logical block numbers are affected as part of the retrieval address.

– 24-bit interface of ISAM

In all the cases described above, the following applies:

– Access to files ≥ 32 GB is rejected with the return code `X'00000D9D'` or `X'00000D00'`, depending on the size of the storage space allocated to the file (FILE_SIZE).

– Exceeding a file size of 32 GB as a result of secondary allocation is prohibited (LARGE_FILE).

Semantic incompatibilities

It is possible that applications use interfaces that employ 4-byte fields for the data fields described above, but that these 4-byte fields are implicitly or explicitly subject to the former limitation to values less than X'00FFFFFF'. Detailed information relating to this can be found in the Appendix, in the section “Semantic Incompatibilities” on page 96.

The following return codes display information about execution of the macro with regard to large files:

X'cc'	**X'bb'**	**X'aaaa'**	**Explanation**
X'00'	X'00'	X'0D9D'	Error when opening a disk file
X'00'	X'00'	X'0D00'	System error when opening a file

4.2.4 FCB

The FCB macro defines the file control block, which is the central source of information for the BTAM, ISAM, SAM and UPAM access methods.

Indicating whether large files are permitted

At program level, the LARGE_FILE operand controls whether large files are permitted.

Operation	Operands
FCB	: : [,LARGE_FILE= $\left\{\begin{array}{l}\underline{\text{*FORBIDDEN}}\\ \text{*ALLOWED}\end{array}\right\}$] :

LARGE_FILE
Only for disk files (ISAM, SAM and UPAM access methods):
The LARGE-FILE operand defines whether or not the logical file size may exceed 32 GB.

= *FORBIDDEN
Default setting: The logical file size may not exceed 32 GB.

= *ALLOWED
Only for files with BLKCTRL ≠ PAMKEY:
The logical file size may exceed 32 GB.

4.2.5 FILE

Allocation behavior

Support for files ≥ 32 GB is only possible for non-PAMKEY files on pubsets where LARGE_OBJECTS=TRUE and LARGE_FILES_ALLOWED=TRUE.

Thus, PAMKEY files on these pubsets may not exceed previous allocation limits for reasons of compatibility. The PAMKEY property of a file is not binding at the point at which FILE is executed; it only becomes definitive when the file is accessed using OPEN.

The behavior for files on all other pubsets (LARGE_OBJECTS=FALSE or TRUE, LARGE_FILES_ALLOWED=FALSE) does not change, i.e. the FILE interface rejects an allocation that exceeds the 32 GB limit.

Indicating whether large files are permitted

At program level, the EXC32GB operand controls whether large files are permitted. The operand does not affect the allocation behavior of the FILE interface.

Operation	Operands

`FILE`	`:`
	`:`
	`,EXC32GB=FORBIDDEN/ALLOWED`
	`:`

EXC32GB
Only for disk files; only for non-PAMKEY files:
The EXC32GB operand determines whether or not the file size may exceed 32 GB when the file is processed. The operand is entered in the TFT (Task File Table) and is only evaluated when the file is opened using OPEN.
During processing, EXC32GB does not affect the space assignment of the FILE call.

= FORBIDDEN
The file size may not exceed 32 GB.

= ALLOWED
The file size may exceed 32 GB.

4.2.6 RDTFT

Outputting file attributes

The RDTFT macro allows information to be output from the TFT.

`RDTFT ...,LINK=...,VERSION=2/3`

Using the VERSION=2/3 and PARMOD=31 operands, an output list is created that also outputs information about the file property EXC32GB (see FILE macro, page 64).

Extract from the RDTFT DSECT

```
               RDTFT MF=D,PLIST=OUTPUT,VERSION=3
   :
   :
   :
   1 ***************** FCB AND DEVICE TYPE INFORMATION
   1 IDRFINFO DS    0HL2
   :
   :
   1 IDRIND15 DS    C                  INDICATOR 15
   1 IDRDESCS EQU   X'80'                7-7 S DESTOC SPECIFIED
   1 IDRDESCY EQU   X'40'                6-6 S DESTOC = YES
   1 IDRCMSGS EQU   X'20'                5-5 S CLOSMSG SPECIFIED
   1 IDRCMSGY EQU   X'10'                4-4 S CLOSMSG = YES
   1 IDRTAPWS EQU   X'08'                3-3 S TAPEWR  SPECIFIED
   1 IDRTAPWY EQU   X'04'                2-2 S TAPEWR  = DEVICE-BUFFER
   1 IDRX32GS EQU   X'02'                1-1 S EXC32GB SPECIFIED
   1 IDRX32GA EQU   X'01'                0-0 S EXC32GB = ALLOWED
   :
```

4.2.7 DIV

The DIV access method (like FASTPAM) uses its own parameter list, rather than the TU FCB for passing parameters.
This DIV(I) parameter list has been extended to include the LARGE_FILE operand for the “Open files” function. The default value *FORBIDDEN prevents uncontrolled access to large files. This function is represented in the parameter list in a bit field that was not used previously and is preset with B'0' (≙ *FORBIDDEN).
The presetting with the default value *FORBIDDEN is therefore guaranteed for programs that were compiled in versions earlier than OSD-BC V5.0.

Indicating whether large files are permitted

The DIV macro with the OPEN function indicates whether large files are permitted. At program level, the LARGE_FILE operand controls whether large files are permitted.

The operand is entered in the TFT (Task File Table) and is evaluated when the file is opened using OPEN.

Operation	Operands
DIV	[,FCT=$\left\{\begin{array}{l}\text{*OPEN}\\ \text{adr}\\ \text{(r)}\end{array}\right\}$] : : [,LARGE_FILE=$\left\{\begin{array}{l}\underline{\text{*FORBIDDEN}}\\ \text{*ALLOWED}\\ \text{adr}\\ \text{(r)}\end{array}\right\}$] :

LARGE_FILE
The LARGE_FILE operand determines whether or not the file size may exceed 32 GB during processing. The operand is entered in the TFT (Task File Table) and is only evaluated when the file is opened using OPEN.

With the form MF=L, only direct specification is permitted.

= *FORBIDDEN
Default setting: The file size may not exceed 32 GB.

= *ALLOWED
The file size may exceed 32 GB.

= adr / (r)
The address of a field that is 1 byte in length and contains the value for LARGE_FILE or the register that contains the value.

Specifying the first block of the data area and the file length

The first block of the file area to be mapped to the virtual address space is specified using the OFFSET operand and the MAP, SAVE, or RESET function. This value is limited depending on the maximum size of a file.

The length of the file area is specified in 4-KB blocks using the SPAN operand and the SAVE or RESET function. This value is limited depending on the maximum size of a file.

Operation	Operands
DIV	[,FCT= $\left\{\begin{matrix}\text{*MAP/*SAVE/*RESET}\\ \text{adr}\\ \text{(r)}\end{matrix}\right\}$]
	:
	:
	[,OFFSET= $\left\{\begin{matrix}\text{number}\\ \text{adr}\\ \text{*equ}\\ \text{(r)}\end{matrix}\right\}$][,SPAN= $\left\{\begin{matrix}\text{number}\\ \text{adr}\\ \text{*equ}\\ \text{(r)}\end{matrix}\right\}$]
	:

OFFSET
Together with SPAN, OFFSET defines the file region for which the window is created. The OFFSET operand can be specified with the MAP, SAVE or RESET functions.

Default setting: OFFSET = 0

With the form MF=L, only direct specification is permitted.

= number
Specifies the first block of the file region to be mapped in virtual address space. The value for OFFSET is limited to the maximum size of a file in 4 KB pages minus 1:

0 ≤ number ≤ 8388606 for LARGE_FILE=*FORBIDDEN

0 ≤ number ≤ 1073741823 for LARGE_FILE=*ALLOWED

= adr
Symbolic address of a 4-byte field containing the numeric value (binary) of the specification for the first block of the file region to be mapped in virtual address space.

= *equ
Equate representing the numeric value of the specification for the first block of the file region to be mapped in virtual address space.
The "*" character must precede the name of the equate.

= (r)
Register containing the numeric value of the specification for the first block of the file region to be mapped in virtual address space.

SPAN
Together with OFFSET, SPAN defines the data area which is referred to in SAVE.
The SPAN operand can be specified with the SAVE or RESET functions.

Default setting: SPAN = 0

With the form MF=L, only direct specification is permitted.

= number
Specifies the length of the file region in 4-KB blocks. The value for SPAN is limited by the maximum size of a file in 4 KB pages minus 1:

0 ≤ number ≤ 8388607 for LARGE_FILE=*FORBIDDEN

0 ≤ number ≤ 1073741824 for LARGE_FILE=*ALLOWED

= adr
Symbolic address of a 4-byte field which specifies the length of the file region in 4-KB blocks (binary).

= *equ
Equate which specifies the length of the file area in 4-KB blocks (binary).
The “*” character must precede the name of the equate.

= (r)
Register containing the length of the file region in 4-KB blocks (binary).

Additional return codes display information about the execution of the macro with regard to large files:

X'cc'	X'bb'	X'aaaa'	Explanation
X'00'	X'01'	X'000C'	The value for LARGE_FILE (OPEN) is neither *ALLOWED nor *FORBIDDEN.
X'00'	X'40'	X'0030'	When a file was accessed in the mode SHARUPD=YES, it was determined that the file size exceeded 32 GB; exceeding 32 GB was not permitted for this file in OPEN.

4.2.8 FPAMSRV

The FASTPAM access method (like DIV) uses its own parameter list, rather than the TU FCB for passing parameters.
This FPAMSRV(I) parameter list has been extended to include the LARGE_FILE=*FORBIDDEN/*ALLOWED operands for the “Open files” function.

The operand controls whether the processing of large files is permitted.
The default value *FORBIDDEN prevents uncontrolled access to large files. This function is represented in the parameter list in a bit field that was not used previously and is preset with B'0' (≙ *FORBIDDEN).
The presetting with the default value *FORBIDDEN is therefore guaranteed for programs that were compiled in versions earlier than OSD-BC V5.0.

Indicating whether large files are permitted

The FPAMSRV macro, with the OPEN function and the LARGE-FILE operand, indicates whether large files are permitted.

Operation	Operands
FPAMSRV	$\texttt{[,FCT=}\left\{\begin{array}{l}\texttt{*OPEN}\\ \texttt{adr}\\ \texttt{(r)}\end{array}\right\}\texttt{]}$: : $\texttt{[,LARGE\_FILE=}\left\{\begin{array}{l}\underline{\texttt{*FORBIDDEN}}\\ \texttt{*ALLOWED}\\ \texttt{adr}\\ \texttt{(r)}\end{array}\right\}\texttt{]}$:

Table 7:

LARGE_FILE
specifies whether the file to be opened may become a “large” file that can exceed 32 GB in size.

Default setting: LARGE_FILE = *FORBIDDEN

With the form MF=L, only direct specification is permitted.

= *FORBIDDEN
Default setting: The file may not become a “large file”.

= *ALLOWED
The file may become a “large file”.

= adr / (r)
The address of a field that is 1 byte in length and contains the value for LARGE_FILE or the register that contains the value.

An additional return code displays information about the execution of the macro with regard to large files:

X'cc'	**X'bb'**	**X'aaaa'**	**Explanation**
X'00'	X'01'	X'0015'	The function was not performed. Invalid specification for LARGE_FILE

4.2.9 FPAMACC

The direct decimal numeric value for the number of the first logical block to be transferred can be specified in the BLOCK operand of the FPAMACC macro.
This value is limited depending on the maximum size of a file (specified in the FPAMSRV macro).

Operation	Operands
FPAMACC	: : [,BLOCK=$\left\{\begin{array}{l}\texttt{number}\\\texttt{adr}\\\texttt{(r)}\end{array}\right\}$] :

BLOCK
Specifies the number of the first logical FASTPAM block within the file to be transferred. The block size is determined with the BLKSIZE operand in the the OPEN function of the FPAMSRV macro. Only integer values are permitted.

With the form MF=L, only direct specification is permitted.

= number
Direct entry of a decimal numeric value for the number of the first logical block to be transferred. The value is limited to the maximum size of a file in 4 KB pages minus 1:

1 ≤ number ≤ 8388606 for LARGE_FILE=*FORBIDDEN (see FPAMSRV macro)

1 ≤ number ≤ 1073741823 for LARGE_FILE=*ALLOWED (see FPAMSRV macro)

= adr / (r)
Symbolic address of a 4-byte field containing the numeric value (binary) or the register which contains this address.

An additional return code displays information about the execution of the macro with regard to large files:

X'cc'	X'bb'	X'aaaa'	Explanation
X'00'	X'40'	X'0145'	When a file was accessed in the mode SHARUPD=YES, it was determined that the file size exceeded 32 GB; exceeding 32 GB was not permitted for this file in OPEN.

4.2.10 Special issues when SHARUPD=YES

When files are accessed in the mode SHARUPD=YES, it is possible that a file with a file size of less than 32 GB could become ≥ 32 GB during processing.

There are two different cases here:

- Callers that are prepared for this situation:
  - by specifying LARGE_FILE=*ALLOWED in the FCB macro
  - by specifying EXCEED-32GB=*ALLOWED in the ADD-FILE-LINK command
- Unprepared callers:
  - by specifying LARGE_FILE=*FORBIDDEN in the FCB macro
  - by specifying EXCEED-32GB=*FORBIDDEN in the ADD-FILE-LINK command

UPAM, FASTPAM and DIV access methods

When a file is accessed with the UPAM, FASTPAM and DIV access methods, the size of the file is checked after each allocator call.
If this check reveals a file size ≥ 32 GB and the LARGE_FILE=*FORBIDDEN attribute or the EXCEED-32GB=*FORBIDDEN attribute is set in the corresponding TFT, processing is aborted.

- UPAM then issues the following return code

`X'000009AD'`	FILE SIZE GREATER 32 GIGABYTES IS NOT ALLOWED

  or the corresponding DMS message

  ```
  DMS09AD       FILE SIZE GREATER 32 GIGABYTES IS NOT ALLOWED
  ```

- In this instance, FASTPAM issues the following return code in its own parameter list FPAMACC(I)

`X'00400145'`	`LARGE_FILE_NOT_SPECIFIED` When a file was accessed in the mode SHARUPD=YES, it was determined that the file size exceeded 32 GB; exceeding 32 GB was not permitted for this file in OPEN.

- In this instance, DIV issues the following return code in its own parameter list DIV(I)

`X'00400030'`	`LARGE_FILE_NOT_SPECIFIED` When a file was accessed in the mode SHARUPD=YES, it was determined that the file size exceeded 32 GB; exceeding 32 GB was not permitted for this file in OPEN.

NK-ISAM access method

Under the NK-ISAM access method, the size of the NK-ISAM file is checked on the basis of the extent list located in the file table entry at every SVC entry point. If this check reveals a file size ≥ 32 GB and if the caller set the LARGE_FILE=*FORBIDDEN attribute in the FCB or the EXCEED-32GB=*FORBIDDEN attribute in the TFT, processing is aborted.

– NK-ISAM then issues the following return code

  `X'00000A23'` FILE SIZE GREATER 32 GIGABYTES IS NOT ALLOWED

  or the corresponding DMS message

```
DMS0A23       FILE SIZE GREATER 32 GIGABYTES IS NOT ALLOWED
```

The **K-ISAM** access method is not affected by this problem.

4.3 RFA

As a communication mechanism, RFA itself is not affected by the introduction of large objects. There are, however, several RFA-capable functions that have the potential to access large files. If RFA connections involve versions that are not equal, access to large objects from versions earlier than OSD-BC V5.0 will be suppressed. These access attempts will be rejected by appropriate version queries in the partner task or in the individual functions.

4.4 Notes regarding programs created in high-level programming languages

In the context of high-level programming languages / programming systems, there are two different aspects of file processing:

– File processing in the framework of program creation using compilers

– File processing when running objects created under a compiler/programming system

File processing by compilers

Support for source programs ≥ 32 GB is neither necessary nor desirable - it is unlikely that input objects of this size will be manageable. The same applies to output objects created by compilers such as object modules, listings and debugging output.
As long as the corresponding objects are managed in libraries, the size of the library, but not the size of the library elements, may exceed 32 GB (see the corresponding notes in the table as of page 89).

File processing by runtime systems

Please refer to the product classification in the Appendix as of page 89 for details of the compatibility of individual runtime systems with large files (corresponding to product class B). In addition, support of large files is offered for selected programming systems.

COBOL

In COBOL programs that were compiled with COBOL85 or COBOL2000 and that work with a COBOL runtime system from CRTE up to V2.3B, any attempt to work with large files will be rejected with the file status '9x'.
Avoiding this limitation by explicitly specifying the
ADD-FILE-LINK ...,EXCEED-32GB=*ALLOWED command is not permitted and will cause undefined behavior, in particular the omission of necessary checks.

For programs that were compiled with COBOL2000 V1.1A or higher and employ the COBOL runtime system from CRTE V2.3C or higher, it is possible to access files ≥ 32 GB via the SEQUENTIAL, LINE SEQUENTIAL and INDEXED file organization methods in COBOL.
The COBOL file organization method RELATIVE is only supported with large files if mapped to BS2000-ISAM.

Similarly, the runtime system from CRTE V2.3C or higher is required when using the COBOL option ENABLE-UFS-ACCESS=YES. There are no restrictions to files smaller than 32 GB caused by the COBOL product for BS2000 files and POSIX files using the SEQUENTIAL and RELATIVE organization methods. Files using the INDEXED organization method are still restricted in terms of size in the POSIX file system.

C

The I/O interfaces in C applications, supported up to CRTE V2.3B inclusive, are not suitable for processing large files. As of CRTE V2.3C, the full functionality of the open64 interface family is available for correct processing of BS2000 files ≥ 32 GB. Please refer to the "C Library functions for POSIX applications" manual [1] for a description of these interfaces.

C++

The IOSTREAM interface supplied with CRTE does not support large files. Using IOSTREAM to open large files will fail. By default, no further information other than the return code zero will be issued (in particular, no error message).
The "errno" variable returns the value 3 (`DMS ERROR`).

If C++ programs need interfaces to large files, they must work with the C API (open64) (see "C" section, above).

Java

Java supports large files up to JENV V1.4.

5 Notes on migration

This chapter contains notes on the introduction of large files. Necessary technical extensions are described in the previous chapters. Possible steps for migration and the considerations that should be included in planning will be outlined here.

5.1 Preliminary considerations

Due to the continual growth of the volume of data stored, the size of individual files can reach around 32 GB, and in the long term can even exceed this boundary. In order to be able to make modifications when and where appropriate, these developments must be carefully monitored and extrapolated.
As of OSD-BC V5.0, the option exists for introducing an environment in which files ≥ 32 GB can be processed.

If large files are to be introduced, it is necessary to clarify the following points in advance:

– Which files will or should exceed 32 GB?

– Which programs/applications should run in an “environment with large files”?

– Which programs/applications operate on these files?

Programs/applications that are identified in this way must be checked to ensure that they will be executable in the future, i.e. that they will be able to process large files. Modifications may be necessary. The following sections will examine this.

5.2 Environment

An appropriate environment must be made available by systems support in order to support large files. It is advisable initially to install this environment for test purposes only, on a test application or a guest system under VM2000.

System requirements

As described in Chapter 2 “Large objects in BS2000/OSD”, large files can only be created in special pubsets, known as LARGE_OBJECTS pubsets, with the LARGE_FILES_ALLOWED attribute.
At least one of these pubsets must therefore be made available. This can be achieved by generation with SIR or by subsequent assignment of attributes using the SET-PUBSET-ATTRIBUTES command. The latter approach has the advantage that existing files can be included and updated.

Software configuration

A further requirement is that the appropriate software configuration for OSD-BC V5.0 is made available. There is a summary of the appropriate product versions in the Appendix, in the “Executability of programs in environments with large files” section, as of page 82.

5.3 Program Conversion

5.3.1 Assembler programs

Assembler programs are directly based on the TU interfaces of BS2000/OSD.

1. Modifications may be necessary for programs that should be executable for large files in the future (LARGE-FILES-compatible programs)
2. In general, modification measures will be necessary for programs that should process large files.

In the first case, the way in which the FSTAT program interface is used, and whether it is used, must be checked. As described in detail in Chapter 4, in the “FSTAT” section as of page 57, interface variants earlier than VERSION=2 only support 3-byte block numbers. When large files are accessed, these interfaces issue the return code x'0576'.
This behavior is only undesirable and incorrect if the return code appears “randomly”, because the result list of the FSTAT call also contains large files, although these do not need to be processed. This can occur for FSTAT calls with partially-qualified filenames or filenames with wildcards. Calls with wildcards in the catalog ID are particularly critical. In this instance, the call is not restricted to the file catalog of a pubset, but among other things includes several or all of the pubsets imported in the system.
When critical FSTAT calls are converted to an interface variant equal to or greater than VERSION=2, it is not only modification of access to output area fields that is necessary. Further use of the counters or block numbers taken from here must also be checked and adapted where appropriate.

In the second case, it is necessary to perform modifications in the context of file processing in addition to the modification of FSTAT calls that may be required. For this purpose, the indication LARGE_FILES=*ALLOWED for OPEN in the FCB (or in the parameter list for FASTPAM and DIV) must then be set. This determines that large files can be processed. A check and possible modification of the program logic may also be necessary in order to ensure that large files can be processed without errors. If you are working with block-oriented access methods (UPAM, FASTPAM) or block numbers (retrieval address in SAM), it is also necessary to ensure that the block numbers can universally be handled as 4-byte fields.
In addition to this direct dependence, there may also be an implicit assumption within the program logic that a block number or file cannot be larger than 2^{24}-1. No exhaustive set of rules can be given for identifying “semantic incompatibilities” of this kind. A list of examples in the form of a checklist is given in the Appendix (“Semantic Incompatibilities” section, page 96).

For test purposes, or if no modifications are necessary, it is possible to set whether large files are permitted via the ADD-FILE-LINK command interface (see page 49).

5.3.2 Programs in high-level programming languages

COBOL

New compilation with COBOL2000 V1.1A or higher and the runtime system CRTE V2.3 or higher is necessary for processing large files.
For further information, see the “COBOL” section on page 75.

C, C++

As of CRTE V2.3C, a new interface family (open64) is supported, which permits large files to be processed. This means that the appropriate interfaces must be included in C/C++ programs and a the programs must be produced again with the runtime system CRTE V2.3 or higher in order to support large files.

For further information, see the “C” section on page 76.

6 Appendix

The appendix contains the following tables and overviews:

– Executability of programs in environments with large and small files, organized alphabetically into two tables:
  – Components of BS2000/OSD-BC (as of page 83)
  – Unbundled products (as of page 89)
– Semantic incompatibilities (page 96)
– Messages referring to large objects (page 97)
  New or changed messages for processing large volumes and files, organized according to message codes

6.1 Executability of programs in environments with large files

In the following tables, the software configuration of OSD-BC V8.0 is organized alphabetically, according to:

– Components of BS2000/OSD-BC
– Unbundled products

and the classification with reference to processing large files. The products are organized alphabetically within the individual tables.

Legend:

Product	Product for OSD-BC V8.0
Version	(Earliest) product version released for OSD-BC V8.0

Classification and observations

Classification A:
The program is able to process large files without restrictions.
This behavior is defined as capable of LARGE_FILES.

Classification B:
Although the program is not prepared for large files and/or their metadata, it is able to reject corresponding accesses that must be regarded as illegal, or, alternatively, it does not access programs and their metadata.
This behavior is defined as compatible with LARGE_FILES.

Classification C:
The program has not been prepared for processing large files and cannot perform a defined rejection of corresponding access.
This behavior is defined as incompatible with LARGE_FILES.

6.1.1 BS2000/OSD-BC

Product	Version	Classification and comments
ACS	V17.0A	Classification B Alias catalog ≥ 32 GB is not supported
ADAM	V17.0A	Classification B
AIDSYS	V17.0A	Classification B
AIDSYSA	V17.0A	Classification B
ANITA	V17.0A	Classification B
APACHE	V02.2	
– APACHE	V02.2	Classification B; runs under POSIX
– PERL	V05.8A	Classification B; runs under POSIX
– TOMCAT	V05.5A	Classification B; runs under POSIX
ASE	V01.0A	Classification B
ASSEMBH-GEN	V01.2C	Classification B
ASTI	V02.0A	Classification B
BINDER	V02.5A	Classification B
BLSSEC	V17.0A	Classification B
BLSSERV	V02.7A	Classification B
BS2CP	V17.0A	Classification B
BS2000-EXEC	V17.0A	Classification B
BS2XC-EXEC	V04.0A	Classification B
BUILDER	V01.0A	Classification B
C-TPR-LZS	V02.5A	Classification A
CALENDAR	V17.0A	Classification B
CALENDAR-TU	V17.0A	Classification B
CAPRI	V02.0A	Classification B
CCOPY	V07.0A	Classification A
CONSTERM	V06.0A	Classification B
COSMOS-BC	V17.0A	Classification B Subsystem declarations only
CPR	V17.0A	Classification A

Table 8: BS2000/OSD-BC (part 1 of 6)

Product	Version	Classification and comments
CRTE-BAS	V01.8A	
– CRTE-BASYS	V01.8A	Classification A
– CRTE-MSG	V01.8A	Classification A
– POSIX-HEADER	V01.8A	Classification A
DAMP	V04.6A	Classification B
DCADITO	V17.0A	Classification A
DIV	V17.0A	Classification A
DIVTRAC	V17.0A	Classification B
DLMUSER	V17.0A	Classification B
DPAGE	V17.0A	Classification A
DSSM	V04.3A	
– DSSM	V04.3A	Classification B
– ROSI	V17.0A	Classification B
– SSCM	V02.3B	Classification B
DWS	V11.0A	Classification B
ELFE	V17.0A	Classification B
ELSA	V01.7A	Classification B
FASTPAM	V17.0A	Classification A
FIRST	V17.0A	Classification B Large volumes and distribution files are not supported on /390 initial installations
FITC	V07.0A	Classification B
GCF	V01.7A	Classification B
GET-TIME	V17.0A	Classification B
GSMAN	V17.0A	Classification B
GSVOL	V01.3A	Classification B
HELGA	V17.0A	Classification B
IDIAS	V17.0A	Classification B
IMON	V03.1A	
– IMON-BAS	V03.1A	Classification B
– IMON-GPN	V03.1A	Classification B
– IMON-SIC	V03.1A	Classification B
INIT	V17.0A	Classification B

Table 8: BS2000/OSD-BC (part 2 of 6)

Product	**Version**	**Classification and comments**
IOCFCOPY	V17.0A	Classification B
IOGEN	V17.0A	Classification A
IORM	V08.0A	Classification A
IOTRACE	V17.0A	Classification B
JENV	V05.1B	Classification A
JITSYS	V05.0A	Classification B
JMP	V02.0B	Classification B
JMU	V14.0B	Classification B
JOBSCHED	V17.0A	Classification B
JPPOPT	V02.6A	Classification B
KDCMON	V17.0A	Classification B
LLMAM	V03.4A	Classification B
LMSCONV	V03.4A	Classification A
LNM	V17.0A	Classification B
MIP	V17.0A	Classification B
MSCFANC	V17.0A	Classification B
MSGMAKER	V01.2A	Classification B
NDMDAMP	V16.0A	Classification B
NKISAM	V17.0A	Classification A
NKISTRAC	V17.0A	Classification B
NKS	V17.0A	Classification B Executable in configurations with large files, but no utilization of large files
NKV	V17.0A	Classification B Executable in configurations with large files, but no utilization of large files
NLMSERVE	V17.0A	Classification B
PAMCONV	V12.1A	Classification C Workaround possibilities with PERCON
PAMINT	V08.0A	Classification A
PASSWORD	V17.0A	Classification B

Table 8: BS2000/OSD-BC (part 3 of 6)

Product	Version	Classification and comments
PLAM	V03.5B	
– PLAM	V03.5B	Classification A
– PMLOG	V03.5B	Classification A
– PMSYS170	V03.5B	Classification A
POSIX-BC	V08.0A	
– POSIX-BC	V08.0A	Classification A
– POSIX-ADDON-LIB	V02.1A	Classification A
– POSIX-NSL	V07.0A	Classification A
– POSIX-SH	V07.0A	Classification A
– POSIX-SOCKETS	V07.0A	Classification A
– POSPRRTS	V01.4A	Classification A
PRSC	V01.0A	Classification B
PTHREADS	V01.1A	Classification A
PVSREN	V04.0A	Classification A
RESLOG	V01.5A	Classification B
RMS	V07.1G	Classification B
SANCHECK	V02.0A	Classification A
SCDM	V08.0A	Classification B
SDF	V04.7A	
– SDF	V04.7A	Classification B
– DISPLAY	V01.1A	Classification B
– FHS-TPR	V08.3A	Classification B
– SDF-CONV	V03.0B	Classification B
– SDF-I	V04.1B	Classification B
– SDF-P-BASYS	V02.5A	Classification B
– SDF-P-BIF	V01.1A	Classification B
– SDF-PAR	V01.1A	Classification B
– SDF-SFC	V03.1A	Classification B
– SDF-SRV	V03.1A	Classification B
– SDF-U	V04.1F	Classification B
– VAS	V02.4A	Classification B
SHOW-FILE	V17.0A	Classification A

Table 8: BS2000/OSD-BC (part 4 of 6)

Product	Version	Classification and comments
SIR	V17.0A	Classification A Only small distribution files when performing a tape copy of SOLIS distribution tapes.
SMI	V01.0A	Classification B
SMPGEN	V17.0A	
– SMPGEN-S	V17.0A	Classification A
– SMPGEN-U	V17.0A	Classification A
SPACEPRO	V01.0A	Classification A
SPCCNTRL	V17.0A	Classification A
SPOOL	V04.9A	
– PRMMAN	V01.4A	Classification B
– PRMPRES	V01.2A	Classification B
– SNRTP	V02.0A	Classification B
– SPCONV	V01.2A	Classification B
– SPOOL	V04.9A	Classification B
– SPOOLSYS	V02.3A	Classification B
– SPSERVE	V02.9B	Classification B
– SPSRVMAN	V02.4A	Classification B
– BS2ZIP	V01.2B	Classification A
SRPMNUC	V17.0A	Classification B
STARTUP	V17.0A	
– IPL	V17.0A	Classification A
– SLED	V17.0A	Classification B
– STRT	V17.0A	Classification A
– BOOT	V17.0A	Classification B
STATUS	V15.2A	Classification B
SYSFILE	V17.0A	Classification B
TANGRAM	V01.5A	
– TANGRAM	V01.5A	Classification B Parameter file is always small
– TANGBAS	V01.5A	Classification B
TGEN	V17.0A	Classification B Large volumes and distribution files are not supported on /390 initial installations

Table 8: BS2000/OSD-BC (part 5 of 6)

Product	Version	Classification and comments
TPCOMP2	V17.0A	Classification B
TPRLAM	V17.0A	Classification A
TSOSLNK	V21.0E	Classification B
TULAM	V17.0A	Classification B
UTM-SM2	V17.0A	Classification B
VAS-TU	V02.1A	Classification B
VOLIN	V17.0A	Classification A
WARTOPT	V17.0A	Classification B
WEBTRANS-OSD	V07.1A	Classification B

Table 8: BS2000/OSD-BC (part 6 of 6)

6.1.2 Unbundled products

Product	Version	Classification and comments
AID	V03.4A	
– AID	V03.4A	Classification B
– LLMAID	V01.1A	Classification B
ARCHIVE	V09.0A	Classification A
ASSEMBH	V01.2D	Classification B Library members less than 32 GB from libraries ≥ 32 GB are supported
AVAS	V08.0A	Classification B
AVAS-SV-BS2	V08.0A	Classification B
COBOL2000	V01.4B	Classification B Library members less than 32 GB from libraries ≥ 32 GB are supported
COBOL85	V02.3A	Classification B No libraries ≥ 32 GB
COLUMBUS85	V01.0C	Classification B
COSMOS	V17.0A	
– COSMOS	V17.0A	Classification B
– CAP	V17.0A	Classification B
– COSMIX	V17.0A	Classification B
– COSMOS-TOOLS	V17.0A	Classification B
– NEGET	V17.0A	Classification B SM2 files can only be accessed via NEGET
CPP	V03.2B	Classification B
CPP-RS	V03.2B	Classification B
CRTE	V02.8A	Classification A
DAB	V09.2A	Classification A
DPRINT	V01.2A	
– DPRINTCL	V01.2A	Classification B
– DPRINTCM	V01.2A	Classification B
– DPRINTSV	V01.2A	Classification B
DRIVE	V03.1A	Classification B

Table 9: Unbundled products (part 1 of 7)

Product	Version	Classification and comments
DRIVE-COMP	V03.1A	Classification B
DRVWIN	V02.1A	Classification B
DRVWIN-C	V02.1A	Classification B
DRVWIN-C-LCS	V02.1A	Classification B
DRV	V03.2A	Classification A
EDT	V17.0A	Classification A in unicode mode Classification B in compatibility mode (files < 26 GB)
ESQL-COBOL	V03.0B	Classification B Library members less than 32 GB from libraries ≥ 32 GB are supported
FDDRL	V17.0A	Classification A
FDDRL-OS	V17.0A	Classification B
FHS	V08.3A	Classification B
FMS	V02.4B	Classification B
FOR1	V02.2C	Classification B Library members less than 32 GB from libraries ≥ 32 GB are supported
HIPLEX-AF	V03.3A	
– HIPLEX-AF	V03.3A	Classification B
– PROP-XT	V01.3A	Classification B
HIPLEX-MSCF	V06.0A	
– MSCF	V17.0A	Classification B
– NSM	V17.0A	Classification B
– XCS-TIME	V17.0A	Classification B
HSMS	V09.0A	
– HSMS	V09.0A	Classification A
– HSMS-API	V09.0A	Classification B
HSMS-CL	V09.0A	Does not run in BS2000
HSMS-SV	V09.0A	Classification B
IFG	V08.3A	Classification B

Table 9: Unbundled products (part 2 of 7)

Product	**Version**	**Classification and comments**
INETSERV	V03.3A	
– MAIL	V03.2A	Classification B
– TCP-IP-AP	V05.1A	Classification B
– TCP-IP-SV	V03.1A	Classification B; läuft unter POSIX
JV	V15.0A	Classification B
LEASY	V06.2A	Classification B
LMS	V03.4A	Classification A
MAREN	V12.0A	Classification A
NFS	V03.0A	Classification A
OMNIS	V08.4A	Classification B Executable in configurations with large files, but no utilization of large files
OMNIS-MENU	V03.4A	Classification B Executable in configurations with large files, but no utilization of large files
OMNIS-PROP	V03.2A	Classification B
ONETSERV	V03.3A	
– BCAM	V20.0A	Classification B
– IPSEC	V01.3A	Classification B
– CMX-BS2000	V01.4A	Classification B
– DCM-DIAG	V01.1A	Classification B
– LWRESD	V01.1A	Classification B
– PLUS	V09.1B	Classification B
– VTSUTRAC	V13.3A	Classification B
– XHCS-SYS	V02.1A	Classification B
– VTSU-B	V13.3A	Classification B
– BCAM-DIAG	V01.0A	Classification B
– PRNGD	V01.1A	Classification B
– SOCKETS-BS2000	V02.4A	Classification B
– DCAM	V13.3A	Classification B
– BCAM-GEN	V01.1A	Classification B

Table 9: Unbundled products (part 3 of 7)

Product	Version	Classification and comments
OPENCRYPT-SERV	V01.3A	
– CRYPT	V01.3A	Classification B
– OCRYSERV	V01.3A	Classification B
OPENFT	V10.0B	Classification A
OPENFT-AC	V10.0B	Classification A
OPENFT-CR	V10.0B	Classification A
OPENFT-FTAM	V10.0B	Classification A
OPENFT-FTP	V10.0B	Classification A
OPENSM2	V08.0A	
– SM2	V17.0A	Classification B
– SM2-TOOLS	V08.0A	Classification B
OPENUTM	V05.3A	
– UTM	V05.3A	Classification B
OPENUTM-CLIENT	V05.3A	
– UTM-CLIENT	V05.3A	Classification B Unbundled due to the C runtime system
OPENUTM-CRYPT	V05.3A	
– UTM-CRYPT	V05.3A	Classification B
OPENUTM-D	V05.3A	
– UTM-D	V05.3A	Classification B
Oracle Database	10gR2	Classification A
OSS	V04.1C	Classification B
PASCAL-XT	V02.2B	Classification B DMS error code
PASCAL-XT-LZS	V02.2B	Classification B DMS error code
PCS	V02.9A	
– PCS	V02.9A	Classification B Parameter file is always small
– PCSDEFINE	V02.9A	Classification B
PERCON	V02.9A	Classification A

Table 9: Unbundled products (part 4 of 7)

Product	Version	Classification and comments
PLI1	V04.2A	Classification B Library members less than 32 GB from libraries ≥ 32 GB are supported; files ≥ 32 GB are not supported
PROP-TPM	V03.0A	Classification B
PROP-XT	V01.3A	Classification B
RAV-BAS	V05.1A	Classification B
RAV-BS2	V05.1A	Classification B
RAV-FT	V05.1A	Classification B
RAV-SX-A	V05.1A	Classification B
RAV-UTM	V05.1A	Classification B
RFA	V17.0A	Classification A
ROBAR-CL	V06.0B	Classification B
ROBAR-SV	V06.0B	Classification B does not run in BS2000
RPG3	V04.0B	Classification B
RPG3-LZS	V04.0B	Classification B Library members less than 32 GB from libraries ≥ 32 GB are supported
RPG3-XT	V04.0B	Classification B Library members less than 32 GB from libraries ≥ 32 GB are supported
RSO	V03.6A	
– RSO	V03.6A	Classification B
– RSOSERVE	V03.6A	Classification B
SBA-BS2	V06.2A	Classification B
SCA	V17.0A	Classification A
SCCA-BS2	V02.0A	
– SCCA-BS2	V02.0A	Classification A
– SYMAPI	V06.6A	Classification A

Table 9: Unbundled products (part 5 of 7)

Product	Version	Classification and comments
SDF-A	V04.1F	
– SDF-A	V04.1F	Classification B
– SDF-SIM	V04.6A	Classification B
SDF-P	V02.5A	Classification B
SECOS	V05.2A	
– SECOS-KRB	V05.2A	Classification B
– GUARDCOO	V05.2A	Classification B
– GUARDDEF	V05.2A	Classification B
– GUARDS	V05.2A	Classification B
– GUARDS-SAVE	V05.2A	Classification B
– SATCP	V05.2A	Classification B
– SATUT	V05.2A	Classification B
– SRPMOPT	V05.2A	Classification B no DMS operations in subsystem
SESAM/SQL	V05.0A	
SESAM/SQL-DCN	V05.0A	Classification A
SESAM/SQL-EE	V05.0A	Classification A
SESAM/SQL-LK	V05.0A	Classification A
SESAM/SQL-SE	V05.0A	Classification A
SESAM/SQL	V06.0A	Classification A
SESAM/SQL-DCN	V06.0A	Classification A
SESAM/SQL-EE	V06.0A	Classification A
SESAM/SQL-LK	V06.0A	Classification A
SESAM/SQL-SE	V06.0A	Classification A
SESDBA	V06.0A	Classification A
SHC-OSD	V07.0A	
– SHC-OSD	V07.0A	Classification A
– SYMAPI	V06.6A	Classification A
– STORMAN	V02.0A	Classification A
SM2-PA	V02.0A	Classification B
SORT	V07.9B	Classification A
SPACEOPT	V05.0A	Classification A
SSA-OUTM-BS2	V05.0B	Classification B

Table 9: Unbundled products (part 6 of 7)

Product	Version	Classification and comments
SSA-SM2-BS2	V05.0A	Classification B
SSC-BS2	V06.0A	Classification B
TASKDATE	V17.0A	Classification B
TIAM	V13.2A	Classification B
TOM-DOC	V03.2A	Classification B Library members less than 32 GB from libraries ≥ 32 GB are supported
TOM-GEN	V02.1B	Classification B
TOM-REF	V03.0B	Classification B
TOM-TI	V03.0A	Classification B
TOMDOORS-M	V05.0D	Classification B
UDS-D	V02.4A	Classification A
UDS-D	V02.5A	Classification A
UDS-IQS – IQS – UDS-IQS	V04.0A V04.0A V04.0A	 Classification A Classification A
UDS/SQL	V02.4	Classification A
UDS/SQL	V02.5A	Classification A
VM2000 – VM2000-HPV – VM2000-MON – VM2000-UTIL	V09.0A V09.0A V09.0A V09.0A	 Classification B Classification B Classification B
VTSU-X.29	V01.5A	Classification B
WEBTRANS-UTM	V07.1A	Classification B

Table 9: Unbundled products (part 7 of 7)

6.2 Semantic Incompatibilities

In Chapter 5, it was mentioned that for the modifications in Assembler code for files ≥ 32 GB in addition to the conversion to 4-byte block numbers and counters, it is necessary to check whether the program logic implicitly assumes that files may not be larger than 32 GB.

The following lists a number of examples of potential problems.

- The highest 3-byte block number X'FFFFFF' has a special meaning.
- “Block numbers” greater than X'00FFFFFF' represent objects other than blocks.
- For calculations with block numbers or counters greater than X'00FFFFFF', overflow can occur.
- The number of digits in input or output fields is not sufficient for displaying such large block numbers or counters.
- When converting hexadecimal numbers to decimal numbers, the field length is too small for the decimal number.
- It is assumed that data structures whose size depends on a file size always find space in virtual memory. This assumption can apply to files less than 32 GB, but not if this size is exceeded.

6.3 Messages referring to large objects

The following displays the messages that are mentioned in this manual with reference to large objects. They are sorted alphabetically according to message code.

```
ARC0061   FILE GREATER THAN 32GB. FILE NOT PROCESSED

          ? Files with a file size greater than 32GB can not be processed
            with this ARCHIVE Version or in this environment

          ! Processing of this file is possible in an environment with
            following features:
            - >= BS2000 OSD V5.0A
            - >= ARCHIVE V6.0A
            - In case of a restoration the destination pubset must allow
              files larger than 32GB


DMS037E   FORMAT OF VOLUME '(&02)' INCONSISTENT WITH PUBSET/VOLUMESET
          '(&00)'. IMPORT PUBSET TASK ABORTED WITH ERRORCODE '(&01)'

          ? (&00): id of SF pubset or volumeset
            (&01): format error
             '01': PAM key volumes and NON-PAM key volumes exists. SF pubsets
                   or volumesets may only consist of volumes with the same
                   PAM key type.
             '02': volumes with different minimal I/O-transfer units exist.
                   SF pubsets or volumesets may only consist of volumes with
                   the same minimal I/O unit.
             '03': volumes with different minimal allocation units exist.
                   SF pubsets or volumesets may only consists of volumes with
                   the same minimal allocation unit.
             '04': a volume is larger than 32 GB, but the pubset doesn't have
                   the attribute for large volumes.
            (&02): vsn of current volume

          ! Check volume identity, maybe initialize some volumes and try
            import again.
```

```
DMS0501   REQUESTED CATALOG NOT AVAILABLE OR WILDCARDS IN USER IDENTIFICATION

          ? Possible reasons:
            Case 1: The class 2 option BMTNUM=0 is set for private disks.
            Case 2: When accessing files with the aid of a MSCF connection,
                    wildcards were used in the user identification.
            Case 3: The required catalog is not imported.
            Case 4: The identification of a volume set from a pubset was
                    specified in the pathname.
            Case 5: The required volume set is not available.

          ! Case 1: Request the system administrator to make the catalog
                    available. Reset the class 2 option BMTNUM to a value
                    other than zero.
            Case 2: Repeat the call without wildcards.
            Case 3: Request the system administrator to import the required
                    catalog.
            Case 4: Specify the identification of the relevant pubset.
            Case 5: Ask the system administrator to make the required volume
                    set available.


DMS0854   FILE SIZE OF SYSEAM FILE (&00) MAY NOT EXCEED 32 GIGABYTES

          ? SYSEAM file (&00) has been created with unallowed file size
            higher than 32 gigabytes (by CREATE-FILE).

          ! Delete SYSEAM-file (&00) and create it again with allowed file
            size below 32 gigabytes.


DMS09AD   FILE SIZE GREATER 32 GIGABYTES IS NOT ALLOWED

          ? In FCB or ADD-FILE-LINK LARGE_FILE=*FORBIDDEN has been set, but
            concerned file exceeeds 32 gigabytes.

          ! Correct program. If the error persists, contact system
            administrator.


DMS0A23   FILE SIZE GREATER 32 GIGABYTES IS NOT ALLOWED

          ? In FCB or ADD-FILE-LINK LARGE_FILE=*FORBIDDEN has been set, but
            concerned file exceeeds 32 gigabytes.

          ! Correct program. If the error persists, contact system
            administrator.
```

```
DMS1383   VOLUME INCONSISTENT. ERROR TYPE '(&00)'. COMMAND REJECTED

          ? (&00): error type
             '01': The raid mode of the volume does not match the
                   pubset/volumeset.
             '02': The GS mode of the volume does not match the
                   pubset/volumeset.
             '03': Returncode "bad volume" from the allocator.
             '04': The dual recording mode of the volume does not match the
                   pubset/volumeset.
             '05': The timestamp of the volume reserved by the slave does not
                   match the master.
             '06': A volume is larger than 32 GB, but the pubset doesn't have
                   the attribute for large volumes.

          ! Dependent on cause.


NVL0146   DISK CAPACITY GREATER EQUAL 32GB IS NOT PERMITTED FOR PRIVATE DISKS

          ? no further information

          ! none


SIR0308   DISK FOR VSN '(&00)' EXCEEDS CAPACITY OF 32 GB; DISK REJECTED

          ? In a SIR run with ACTION=*INSTALL is the operand LARGE-DISKS-
            ALLOWED=*NO specified in the //DECLARE-PUBSET assignment or for
            ACTION=*EXTEND the coresponding pubset attribute is set.

          ! - for ACTION=*INSTALL you have to set the operand LARGE-DISKS-
              ALLOWED to *YES
            - for ACTION=*EXTEND you have to set the coresponding attribute
              of the pubset using the command /SET-PUBSET-ATTRIBUTES ...,
              LARGE-VOLUMES=*ALLOWED before you restart the SIR run.


SIR0728   COPYING OF FILE '(&00)' ABNORMALLY TERMINATED. FILE SIZE EXCEEDS
          THE LIMIT OF 32GB

          ? no further information

          ! none


SPC0030   ERROR EXECUTING (&00)-MACRO, RETURN CODE = (&01)

          ? no further information

          ! none
```

Related publications

The manuals are available as online manuals, see *http://manuals.ts.fujitsu.com*, or in printed form which must be paid and ordered separately at *http://manualshop.ts.fujitsu.com*.

[1] **C Library Functions** (BS2000/OSD)
for POSIX Applications
Reference Manual

[2] **BS2000/OSD-BC**
Utility Routines
User Guide

[3] **BS2000/OSD-BC**
DMS Macros
User Guide

[4] **BS2000/OSD-BC**
Introductory Guide to DMS
User Guide

[5] **BS2000/OSD-BC**
Introductory Guide to Systems Support
User Guide

[6] **FDDRL** (BS2000/OSD)
User Guide

[7] **HIPLEX MSCF** (BS2000/OSD)
BS2000 Processor Networks
User Guide

[8] **HSMS** (BS2000/OSD)
Hierarchical Storage Management System
Volume 1 and 2
User Guide

[9] **IMON** (BS2000/OSD)
Installation Monitor
User Guide

[10] **BS2000/OSD-BC**
Commands
User Guide

[11] **BS2000/OSD-BC**
Executive Macros
User Guide

[12] **Performance Handbook**
User Guide

[13] **POSIX** (BS2000/OSD-BC)
Basics for Users and System Administrators
User Guide

[14] **SPACEOPT** (BS2000/OSD)
Disk Optimization and Reorganization
User Guide

[15] **BS2000/OSD-BC**
System Installation
User Guide

Index