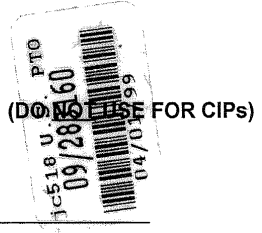


IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
REQUEST FOR FILING  
(RULE 53(b)(1))

For Design or Utility Applications



Rule 53(b)(1) PATENT APPLICATION:

- Continuation )
- Divisional ) application under 37 CFR 1.53(b)(1)

Group Art Unit: 2776

of pending prior application of

Examiner: Homere, J.

Inventor(s): FARBER et al.

Parent Appln. No.:	08	960,079	Atty. Dkt.	PM 252465	
	Series Code ↑	↑ Serial No.		New M#	Client Ref

Parent Filed:	October 24, 1997	(Our Deposit Account No. 03-3975
This Case Filed:	April, 1999	(Our Order No. 7018/252465
Title:	IDENTIFYING DATA IN A DATA PROCESSING SYSTEM	C# / New M#

Date: April 1, 1999

Asst. Commissioner of Patents  
Washington, DC 20231

(Parent Matter No. 243063 )

Sir:

To effect the above-requested filing today:

1. Attached is a copy (which must be filed) of this application, including:

- Abstract
- Specification and claims (94 pages) (must be attached)
- Drawings (must be attached if originally filed): 24 sheet(s)/set:  1 set informal;  Formal of size  A4  11"

1A. Always X one box, only:

- (1)  Signed declaration or oath as originally filed in prior application attached
- (2)  NO declaration or fee is enclosed; therefore, this is a filing under Rule 53(f).

2.  This application is hereby filed by less than all of the inventors named in the prior application. Petition is hereby made requesting deletion as inventor(s) of the following who is/are not inventor(s) of the invention being claimed in this application:

- |          |           |
|----------|-----------|
| 1. _____ | 2. _____  |
| 3. _____ | 4. _____  |
| 5. _____ | 6. _____  |
| 7. _____ | 8. _____  |
| 9. _____ | 10. _____ |

3. The entire disclosure of the prior application is considered as being part of the disclosure of the accompanying application and is hereby incorporated therein by reference thereto.

4.  Priority is claimed under 35 U.S.C. 119/365 based on filing in \_\_\_\_\_ of \_\_\_\_\_ (country)
- |     | <u>Application No.</u> | <u>Filing Date</u> |     | <u>Application No.</u> | <u>Filing Date</u> |
|-----|------------------------|--------------------|-----|------------------------|--------------------|
| (1) | _____                  | _____              | (4) | _____                  | _____              |
| (2) | _____                  | _____              | (5) | _____                  | _____              |
| (3) | _____                  | _____              | (6) | _____                  | _____              |

- a.  \_\_\_\_\_ (No.) Certified copy/copies attached.  
 b.  Certified copy/copies previously filed on \_\_\_\_\_ in \_\_\_\_\_ U.S. Application No. \_\_\_\_\_ / \_\_\_\_\_, filed on \_\_\_\_\_ series code ↑ serial no.  
 c.  Certified copy/copies filed during International stage of PCT/ \_\_\_\_\_ / \_\_\_\_\_

4. (a)  Domestic priority is claimed from PCT/ \_\_\_\_\_ / \_\_\_\_\_, filed \_\_\_\_\_  
 (b)  Benefit is claimed of Provisional Application No. 60/\_\_\_\_\_, filed \_\_\_\_\_

5.  Prior application is assigned to kiNETech, Inc.  
 by assignment recorded June 23, 1995 Reel 7593 Frame 0036  
 (Date)

6.  Attached is the following number of Assignments (including original and all later successive ones by different assignors): 1 and respective new Cover Sheets. (Do **NOT** file old cover sheets.)  
 (Assignments in parent **must be refiled** with new Cover Sheets in this continuing application if you want it/them recorded against the continuing application.)

Please return the recorded Assignment to the undersigned.

7.  The power of attorney in the prior application is to Dale S. Lazar, Reg. No. 28,872  
 \_\_\_\_\_  
 (Name and Reg. No.)  
 whose current address is as in item 8 below.

- a.  Recognize as associate attorney Brian Siritzky, Reg. No. 37,497  
 \_\_\_\_\_  
 (Name, Reg. No. and Address)

8. **Address all future communications to Intellectual Property Group of Pillsbury Madison & Sutro LLP, Ninth Floor, East Tower 1100 New York Avenue, N.W., Washington, D.C. 20005-3918**

9.  **Amend the specification** by inserting before the first line the sentence:--This is a  
 continuation  division of Application No. 08/960,079, filed October 24, 1997  
 series code ↑ serial no.  
 which is a continuation of 08/425,160, filed April 11, 1995, now abandoned. --

9. (a)  **Amend the specification** by inserting before the first line: --This application claims the benefit of Provisional Application No. 60/\_\_\_\_\_, filed \_\_\_\_\_ --

10.  It has been recently determined that this new continuing application is entitled to small entity status.  
 Hence:  
 (No.) Verified Statement(s) establishing "small entity" status under Rules 9 & 27 were/are:  
 filed in above prior application (and hence applicable hereto)  
 attached.

11. Petition to extend the life of the above prior application to at least the date hereof  
 (one box)  is being concurrently filed in that prior application (Use Form PAT-111).  
 (must be)  was previously filed in that prior application (Check length of prior extension).  
 (X'd)  is not necessary for copendency (**Double check** before X'ing this box).

- 12.  **INFORMATION DISCLOSURE STATEMENT:** Attached is Form PTO-1449 listing all of the documents cited by Applicant and the PTO in the parent application(s) relied upon under 35 USC 120 and referenced in item 9 above. Per Rule 98(d) copies of those documents are not required now. Please consider those documents and advise that they have been considered in this new application as by returning a copy of the enclosed Form PTO-1449 with the Examiner's initials in the left column per MPEP 609. .
- 13.  Attached is a Rule 103(a) Petition to Suspend Action.
- 14.  **PRELIMINARY AMENDMENT to be entered before fee calculation:** (Do not make amendments here except for correction of improper multiple dependencies or cancellation of whole claims or multiple dependencies for purpose of reducing the filing fee per MPEP §§ 506 and 607; do not cancel all claims).

Please cancel claims 1-45 and 50-53 without prejudice. The remaining claims correspond to non-elected Groups III & IV from the Examiner's Restriction Requirement of June 4, 1996.

**FILING FEE**

THE FOLLOWING FILING FEE IS BASED ON

-->>>>CLAIMS AS FILED AND CHANGED BY PRELIMINARY AMENDMENT IN ITEM 14<<<<<<

**NOTE:** If box 1A2 is X'd, do not pay fees, but leave lines 15-22 and 27-32 blank.

				Large/Small Entity		Fee Code
15. Basic Filing Fee . . . . . Design Application				\$310/\$155		106/26
16. Basic Filing Fee . . . . . Not Design Application				\$760/\$380	+380	101/201
17. Total Effective Claims	5	minus 20 =	0	x \$18/\$9	+0	103/203
18. Independent Claims	3	minus 3 =	0	x \$78/\$39	+0	102/202
19. If <u>any proper</u> multiple dependent claim (ignore improper) is present,				\$260/\$130	+0	104/204
20. <b>Subtotal =</b>				<b>\$380</b>		
21. If "petition" box 13 above is X'd, add petition fee. . . . . \$130					+0	122
21A. If box 6 above is X'd, add Assignment recording fee . . . . . \$ 40					+40	581
22. <b>TOTAL FILING FEE ATTACHED =</b>					<b>\$420</b>	

(carry forward to Item 31)

- 23.  ATTACHED:
- 24.  Preliminary Amendment attached (to be entered after assigning Appln. No.)
- 25.  The following PRELIMINARY AMENDMENT is to be entered after assigning Appln. No.:

26.

**ADDITIONAL FEE CALCULATION FOR  
PRELIMINARY AMENDMENT  
PER BOXES 24/25**

	Claims remaining after amendment	Highest number previously paid for	Present Extra	Additional Fee	
					<u>Large/Small Entity</u> <u>File Code</u>
27.	Total Effective Claims *	minus ** 20	= 0	x \$18/\$9 = \$ 0	(103/203)
28.	Independent Claims *	minus *** 3	= 0	x \$78/\$39 = + 0	(102/202)
29.	If amendment enters proper multiple dependent claim(s) into this application for the <u>first time</u> , add (per application) .....			\$.260/\$130	+ 0 (104/204)
30.				<b>ADDITIONAL FEE</b>	<b>\$ 0</b>
31.				plus FEE from item 22 on page 3	+ 420
32.				<b>TOTAL FEE ATTACHED</b>	<b>\$ 420</b>

33. \*If the entry in this space is less than the entry in the next space, the "Present Extra" result is "0"

34. \*\*If the "Highest number previously paid for" (see item 17 above) is less than 20, write "20" in this space

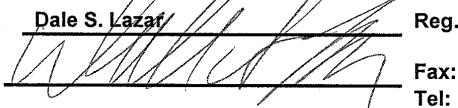
35. If the "Highest number previously paid for" (see item 18 above) is less than 3, write "3" in this space

**CHARGE STATEMENT:** Upon the filing of a Declaration pursuant to Rule 60(b) or 60(d), the Commissioner is hereby authorized to charge any fee specifically authorized hereafter, or any missing or insufficient fee(s) filed, or asserted to be filed, or which should have been filed herewith or concerning any paper filed hereafter, and which may be required under Rules 16-18 (missing or insufficient fee only) now or hereafter relative to this application and the resulting Official document under Rule 20, or credit any overpayment, to our Account/Order Nos. shown in the heading hereof for which purpose a duplicate copy of this sheet is attached.

**This CHARGE STATEMENT does not authorize charge of the issue fee until/unless an issue fee transmittal form is filed.**

**Pillsbury Madison & Sutro LLP  
Intellectual Property Group**

1100 New York Avenue, N.W.  
Ninth Floor, East Tower  
Washington, D.C. 20005-3918  
Tel: (202) 861-3000  
DSL/BS:kim  
Atty./Sec.

By Atty: Dale S. Lazar      Reg. No. 28872  
Sig:       Fax: (202) 822-0944  
Tel: (202) 861-3527

**NOTE No. 1:** File this Request in duplicate with 2 postcard receipts (PAT-103) & attachments  
**NOTE No. 2:** Is extension in parent necessary for copendency? DOUBLE CHECK Item 11 above.

# APPLICATION UNDER UNITED STATES PATENT LAWS

Invention: David A. Farber and Ronald D. Lachman

Inventor(s): IDENTIFYING DATA IN A DATA PROCESSING SYSTEM

Cushman Darby & Cushman, L.L.P.  
1100 New York Avenue, N.W.  
Ninth Floor, East Tower  
Washington, D.C. 20005-3918  
Attorneys  
Telephone: (202) 861-3000

This is a:

- Provisional Application
- Regular Utility Application
- Continuing Application
- PCT National Phase Application
- Design Application
- Reissue Application
- Plant Application

## SPECIFICATION

CDC-100 3/95

INS. C1  
B1  
C4

~~IDENTIFYING DATA IN A DATA PROCESSING SYSTEM~~

BACKGROUND OF THE INVENTION

1. Field of the invention

This invention relates to data processing systems and, more particularly, to data processing systems wherein data items are identified by substantially unique identifiers which depend on all of the data in the data items and only on the data in the data items.

2. Background of the Invention

Data processing (DP) systems, computers, networks of computers, or the like, typically offer users and programs various ways to identify the data in the systems.

Users typically identify data in the data processing system by giving the data some form of name. For example, a typical operating system (OS) on a computer provides a file system in which data items are named by alphanumeric identifiers. Programs typically identify data in the data processing system using a location or address. For example, a program may identify a record in a file or database by using a record number which serves to locate that record.

In all but the most primitive operating systems, users and programs are able to create and use collections of named data items, these collections themselves being named by identifiers. These named collections can then, themselves, be made part of other named collections. For example, an OS may provide mechanisms to group files (data items) into directories (collections). These directories can then, themselves be made part of other directories. A data item may thus be identified relative to these nested directories using a

sequence of names, or a so-called pathname, which defines a path through the directories to a particular data item (file or directory).

5 As another example, a database management system may group data records (data items) into tables and then group these tables into database files (collections). The complete address of any data record can then be specified using the database file name, the table name, and the record number of that data record.

10 Other examples of identifying data items include: identifying files in a network file system, identifying objects in an object-oriented database, identifying images in an image database, and identifying articles in a text database.

15 In general, the terms "data" and "data item" as used herein refer to sequences of bits. Thus a data item may be the contents of a file, a portion of a file, a page in memory, an object in an object-oriented program, a digital message, a digital scanned image, a part of a video or audio signal, or any other entity which can be  
20 represented by a sequence of bits. The term "data processing" herein refers to the processing of data items, and is sometimes dependent on the type of data item being processed. For example, a data processor for a digital image may differ from a data processor for an  
25 audio signal.

In all of the prior data processing systems the names or identifiers provided to identify data items (the data items being files, directories, records in the  
30 database, objects in object-oriented programming, locations in memory or on a physical device, or the like) are always defined relative to a specific context. For instance, the file identified by a particular file name can only be determined when the directory containing the  
35 file (the context) is known. The file identified by a pathname can be determined only when the file system (context) is known. Similarly, the addresses in a

process address space, the keys in a database table, or domain names on a global computer network such as the Internet are meaningful only because they are specified relative to a context.

5           In prior art systems for identifying data items there is no direct relationship between the data names and the data item. The same data name in two different contexts may refer to different data items, and two different data names in the same context may refer to the same data item.

10           In addition, because there is no correlation between a data name and the data it refers to, there is no *a priori* way to confirm that a given data item is in fact the one named by a data name. For instance, in a DP system, if one processor requests that another processor deliver a data item with a given data name, the requesting processor cannot, in general, verify that the data delivered is the correct data (given only the name). Therefore it may require further processing, typically on the part of the requestor, to verify that the data item it has obtained is, in fact, the item it requested.

15           A common operation in a DP system is adding a new data item to the system. When a new data item is added to the system, a name can be assigned to it only by updating the context in which names are defined. Thus such systems require a centralized mechanism for the management of names. Such a mechanism is required even in a multi-processing system when data items are created and identified at separate processors in distinct locations, and in which there is no other need for communication when data items are added.

20           In many data processing systems or environments, data items are transferred between different locations in the system. These locations may be processors in the data processing system, storage devices, memory, or the like. For example, one processor may obtain a data item from another processor or from an

φ



external storage device, such as a floppy disk, and may incorporate that data item into its system (using the name provided with that data item).

5           However, when a processor (or some location) obtains a data item from another location in the DP system, it is possible that this obtained data item is already present in the system (either at the location of the processor or at some other location accessible by the processor) and therefore a duplicate of the data item is  
10       created. This situation is common in a network data processing environment where proprietary software products are installed from floppy disks onto several processors sharing a common file server. In these  
15       systems, it is often the case that the same product will be installed on several systems, so that several copies of each file will reside on the common file server.

          In some data processing systems in which several processors are connected in a network, one system is designated as a cache server to maintain master copies  
20       of data items, and other systems are designated as cache clients to copy local copies of the master data items into a local cache on an as-needed basis. Before using a cached item, a cache client must either reload the cached item, be informed of changes to the cached item, or  
25       confirm that the master item corresponding to the cached item has not changed. In other words, a cache client must synchronize its data items with those on the cache server. This synchronization may involve reloading data  
30       items onto the cache client. The need to keep the cache synchronized or reload it adds significant overhead to existing caching mechanisms.

          In view of the above and other problems with prior art systems, it is therefore desirable to have a  
35       mechanism which allows each processor in a multiprocessor system to determine a common and substantially unique identifier for a data item, using only the data in the data item and not relying on any sort of context.



operation of at least some or all of the following features:

5 the system stores at most one copy of any data item at a given location, even when multiple data names in the system refer to the same contents;

the system avoids copying data from source to destination locations when the destination locations already have the data;

10 the system provides transparent access to any data item by reference only to its identity and independent of its present location, whether it be local, remote, or offline;

15 the system caches data items from a server, so that only the most recently accessed data items need be retained;

when the system is being used to cache data items, problems of maintaining cache consistency are avoided;

20 the system maintains a desired level of redundancy of data items in a network of servers, to protect against failure by ensuring that multiple copies of the data items are present at different locations in the system;

25 the system automatically archives data items as they are created or modified;

the system provides the size, age, and location of groups of data items in order to decide whether they can be safely removed from a local file system;

30 the system can efficiently record and preserve any collection of data items;

the system can efficiently make a copy of any collection of data items, to support a version control mechanism for groups of the data items;

35 the system can publish data items, allowing other, possibly anonymous, systems in a network to gain access to the data items and to rely on the availability of the data items;





09223150-040190  
00000-091660

5 multiprocessor data processing system 100, some or all of  
the processors 102 may be disconnected from the network  
of processors for periods of time. Such disconnection  
may be part of the normal operation of the system 100 or  
it may be because a particular processor 102 is in need  
of repair.

10 Within a data processing system 100, the data  
may be organized to form a hierarchy of data storage  
elements, wherein lower level data storage elements are  
combined to form higher level elements. This hierarchy  
can consist of, for example, processors, file systems,  
regions, directories, data files, segments, and the like.  
For example, with reference to FIGURE 2, the data items  
on a particular processor 102 may be organized or  
15 structured as a file system 116 which comprises regions  
117, each of which comprises directories 118, each of  
which can contain other directories 118 or files 120.  
Each file 120 being made up of one or more data segments  
122.

20 In a typical data processing system, some or  
all of these elements can be named by users given certain  
implementation specific naming conventions, the name (or  
pathname) of an element being relative to a context. In  
the context of a data processing system 100, a pathname  
25 is fully specified by a processor name, a filesystem  
name, a sequence of zero or more directory names  
identifying nested directories, and a final file name.  
(Usually the lowest level elements, in this case segments  
122, cannot be named by users.)

30 In other words, a file system 116 is a  
collection of directories 118. A directory 118 is a  
collection of named files 120 -- both data files 120 and  
other directory files 118. A file 120 is a named data  
item which is either a data file (which may be simple or  
35 compound) or a directory file 118. A simple file 120  
consists of a single data segment 122. A compound file  
120 consists of a sequence of data segments 122. A data

segment 122 is a fixed sequence of bytes. An important property of any data segment is its size, the number of bytes in the sequence.

5 A single processor 102 may access one or more file systems 116, and a single storage device 104 may contain one or more file systems 116, or portions of a file system 116. For instance, a file system 116 may span several storage devices 104.

10 In order to implement controls in a file system, file system 116 may be divided into distinct regions, where each region is a unit of management and control. A region consists of a given directory 118 and is identified by the pathname (user defined) of the directory.

15 In the following, the term "location", with respect to a data processing system 100, refers to any of a particular processor 102 in the system, a memory of a particular processor, a storage device, a removable storage medium (such as a floppy disk or compact disk),  
20 or any other physical location in the system. The term "local" with respect to a particular processor 102 refers to the memory and storage devices of that particular processor.

25 In the following, the terms "True Name", "data identity" and "data identifier" refer to the substantially unique data identifier for a particular data item. The term "True File" refers to the actual file, segment, or data item identified by a True Name.

30 A file system for a data processing system 100 is now described which is intended to work with an existing operating system by augmenting some of the operating system's file management system codes. The embodiment provided relies on the standard file management primitives for actually storing to and  
35 retrieving data items from disk, but uses the mechanisms of the present invention to reference and access those data items.





6. Copy File or Directory;
7. Move File or Directory;
8. Get File Status; and
9. Get Files in Directory.

5 Remote mechanisms are used by the operating system in responding to requests from other processors. These mechanisms enable the capabilities of the present invention in a peer-to-peer network mode of operation. The following remote mechanisms are described:

- 10 1. Locate True File;
2. Reserve True File;
3. Request True File;
4. Retire True File;
5. Cancel Reservation;
- 15 6. Acquire True File;
7. Lock Cache;
8. Update Cache; and
9. Check Expiration Date.

20 Background mechanisms are intended to run occasionally and at a low priority. These provide automated management capabilities with respect to the present invention. The following background mechanisms are described:

- 25 1. Mirror True File;
2. Groom Region;
3. Check for Expired Links; and
4. Verify Region; and
5. Groom Source List.

30 Extended mechanisms run within application programs over the operating system. These mechanisms provide solutions to specific problems and applications. The following extended mechanisms are described:

- 35 1. Inventory Existing Directory;
2. Inventory Removable, Read-only Files;
3. Synchronize directories;
4. Publish Region;
5. Retire Directory;

6. Realize Directory at location;
7. Verify True File;
8. Track for accounting purposes; and
9. Track for licensing purposes.

5           The file system herein described maintains  
sufficient information to provide a variety of mechanisms  
not ordinarily offered by an operating system, some of  
which are listed and described here. Various processing  
performed by this embodiment of the present invention  
10 will now be described in greater detail.

In some embodiments, some files 120 in a data  
processing system 100 do not have True Names because they  
have been recently received or created or modified, and  
thus their True Names have not yet been computed. A file  
15 that does not yet have a True Name is called a scratch  
file. The process of assigning a True Name to a file is  
referred to as assimilation, and is described later.  
Note that a scratch file may have a user provided name.

Some of the processing performed by the present  
20 invention can take place in a background mode or on a  
delayed or as-needed basis. This background processing  
is used to determine information that is not immediately  
required by the system or which may never be required.  
As an example, in some cases a scratch file is being  
25 changed at a rate greater than the rate at which it is  
useful to determine its True Name. In these cases,  
determining the True Name of the file can be postponed or  
performed in the background.

#### Data Structures

30           The following data structures, stored in memory  
110 of one of more processors 102 are used to implement  
the mechanisms described herein. The data structures can  
be local to each processor 102 of the system 100, or they  
can reside on only some of the processors 102.

GOOG-1016-Page 19 of 126

5 The data structures described are assumed to  
reside on individual peer processors 102 in the data  
processing system 100. However, they can also be shared  
by placing them on a remote, shared file server (for  
instance, in a local area network of machines). In order  
to accommodate sharing data structures, it is necessary  
that the processors accessing the shared database use the  
appropriate locking techniques to ensure that changes to  
the shared database do not interfere with one another but  
10 are appropriately serialized. These locking techniques  
are well understood by ordinarily skilled programmers of  
distributed applications.

It is sometimes desirable to allow some regions  
to be local to a particular processor 102 and other  
15 regions to be shared among processors 102. (Recall that  
a region is a unit of file system management and control  
consisting of a given directory identified by the  
pathname of the directory.) In the case of local and  
shared regions, there would be both local and shared  
20 versions of each data structure. Simple changes to the  
processes described below must be made to ensure that  
appropriate data structures are selected for a given  
operation.

The local directory extensions (LDE) table 124  
25 is a data structure which provides information about  
files 120 and directories 118 in the data processing  
system 100. The local directory extensions table 124 is  
indexed by a pathname or contextual name (that is, a user  
provided name) of a file and includes the True Name for  
30 most files. The information in local directory extension  
table 124 is in addition to that provided by the native  
file system of the operating system.

The True File registry (TFR) 126 is a data  
store for listing actual data items which have True  
35 Names, both files 120 and segments 122. When such data  
items occur in the True File registry 126 they are known  
as True Files. True Files are identified in True File

15

607070 0972220

registry 126 by their True Names or identities. The table True File registry 126 also stores location, dependency, and migration information about True Files.

5 The region table (RT) 128 defines areas in the network storage which are to be managed separately. Region table 128 defines the rules for access to and migration of files 120 among various regions with the local file system 116 and remote peer file systems.

10 The source table (ST) 130 is a list of the sources of True Files other than the current True File registry 126. The source table 130 includes removable volumes and remote processors.

15 The audit file (AF) 132 is a list of records indicating changes to be made in local or remote files, these changes to be processed in background.

20 The accounting log (AL) 134 is a log of file transactions used to create accounting information in a manner which preserves the identity of files being tracked independent of their name or location.

The license table (LT) 136 is a table identifying files, which may only be used by licensed users, in a manner independent of their name or location, and the users licensed to use them.

Detailed Descriptions of the Data Structures

25 The following table summarizes the fields of an local directory extensions table entry, as illustrated by record 138 in FIGURE 3.

Field	Description
Region ID	identifies the region in which this file is contained.
Pathname	the user provided name or contextual name of the file or directory, relative to the region in which it occurs.
True Name	the computed True Name or identity of the file or directory. This True Name is not always up to date, and it is set to a special value when a file is modified and is later recomputed in the background.

Take

30

16

001010 0512260

5  
  
  
10

Field	Description
Type	indicates whether the file is a data file or a directory.
Scratch File ID	the physical location of the file in the file system, when no True Name has been calculated for the file. As noted above, such a file is called a scratch file.
Time of last access	the last access time to this file. If this file is a directory, this is the last access time to any file in the directory.
Time of last modification	the time of last change of this file. If this file is a directory, this is the last modification time of any file in the directory.
Safe flag	indicates that this file (and, if this file is a directory, all of its subordinate files) have been backed up on some other system, and it is therefore safe to remove them.
Lock flag	indicates whether a file is locked, that is, it is being modified by the local processor or a remote processor. Only one processor may modify a file at a time.
Size	the full size of this directory (including all subordinate files), if all files in it were fully expanded and duplicated. For a file that is not a directory this is the size of the actual True File.
Owner	the identity of the user who owns this file, for accounting and license tracking purposes.

Each record of the True File registry 126 has the fields shown in the True File registry record 140 in FIGURE 4. The True File registry 126 consists of the database described in the table below as well as the actual True Files identified by the True File IDs below.

20  
TOMC

Field	Description
True Name	computed True Name or identity of the file.

17

00000000000000000000

	Field	Description
	Compressed File ID	compressed version of the True File may be stored instead of, or in addition to, an uncompressed version. This field provides the identity of the actual representation of the compressed version of the file.
	Grooming delete count	tentative count of how many references have been selected for deletion during a grooming operation.
5	Time of last access	most recent date and time the content of this file was accessed.
	Expiration	date and time after which this file may be deleted by this server.
	Dependent processors	processor IDs of other processors which contain references to this True File.
10	Source IDs	source ID(s) of zero or more sources from which this file or data item may be retrieved.
	True File ID	identity or disk location of the actual physical representation of the file or file segment. It is sufficient to use a filename in the registration directory of the underlying operating system. The True File ID is absent if the actual file is not currently present at the current location.
	Use count	number of other records on this processor which identify this True File.

15 A region table 128, specified by a directory pathname, records storage policies which allow files in the file system to be stored, accessed and migrated in different ways. Storage policies are programmed in a configurable way using a set of rules described below.

20 Each region table record 142 of region table 128 includes the fields described in the following table (with reference to FIGURE 5):

18

T 0190

5

667440-0972260

Field	Description
Region ID	internally used identifier for this region.
Region file system	file system on the local processor of which this region is a part.
Region pathname	a pathname relative to the region file system which defines the location of this region. The region consists of all files and directories subordinate to this pathname, except those in a region subordinate to this region.
Mirror processor(s)	zero or more identifiers of processors which are to keep mirror or archival copies of all files in the current region. Multiple mirror processors can be defined to form a mirror group.
Mirror duplication count	number of copies of each file in this region that should be retained in a mirror group.
Region status	specifies whether this region is local to a single processor 102, shared by several processors 102 (if, for instance, it resides on a shared file server), or managed by a remote processor.
Policy	the migration policy to apply to this region. A single region might participate in several policies. The policies are as follows (parameters in brackets are specified as part of the policy): region is a cached version from [processor ID]; region is a member of a mirror set defined by [processor ID]. region is to be archived on [processor ID]. region is to be backed up locally, by placing new copies in [region ID]. region is read only and may not be changed. region is published and expires on [date]. Files in this region should be compressed.

10

A source table 130 identifies a source location for True Files. The source table 130 is also used to

19

identify client processors making reservations on the current processor. Each source record 144 of the source table 130 includes the fields summarized in the following table, with reference to FIGURE 6:

T0200

097E8260

5	<b>Field</b>	<b>Description</b>
	source ID	internal identifier used to identify a particular source.
	source type	type of source location: Removable Storage Volume Local Region Cache Server Mirror Group Server Cooperative Server Publishing Server Client
10	source rights	includes information about the rights of this processor, such as whether it can ask the local processor to store data items for it.
	source availability	measurement of the bandwidth, cost, and reliability of the connection to this source of True Files. The availability is used to select from among several possible sources.
15	source location	information on how the local processor is to access the source. This may be, for example, the name of a removable storage volume, or the processor ID and region path of a region on a remote processor.

The audit file 132 is a table of events ordered by timestamp, each record 146 in audit file 132 including the fields summarized in the following table (with reference to FIGURE 7):

T0201

20	<b>Field</b>	<b>Description</b>
	Original Name	path of the file in question.
	Operation	whether the file was created, read, written, copied or deleted.
	Type	specifies whether the source is a file or a directory.

20



Field	Description
Processor ID	ID of the remote processor generating this event (if not local).
Timestamp	time and date file was closed (required only for accessed/modified files).
Pathname	Name of the file (required only for rename).
True Name	computed True Name of the file. This is used by remote systems to mirror changes to the directory and is filled in during background processing.

5                    Each record 148 of the accounting log 134 records an event which may later be used to provide information for billing mechanisms. Each accounting log entry record 148 includes at least the information summarized in the following table, with reference to

10                    FIGURE 8:

Field	Description
date of entry	date and time of this log entry.
type of entry	Entry types include create file, delete file, and transmit file.
True Name	True Name of data item in question.
owner	identity of the user responsible for this action.

15                    Each record 150 of the license table 136 records a relationship between a licensable data item and the user licensed to have access to it. Each license table record 150 includes the information summarized in the following table, with reference to FIGURE 9:

20

Field	Description
True Name	True Name of a data item subject to license validation.

Field	Description
licensee	identity of a user authorized to have access to this object.

5 Various other data structures are employed on  
some or all of the processors 102 in the data processing  
system 100. Each processor 102 has a global freeze lock  
(GFL) 152 (FIGURE 1), which is used to prevent  
synchronization errors when a directory is frozen or  
copied. Any processor 102 may include a special archive  
directory (SAD) 154 into which directories may be copied  
for the purposes of archival. Any processor 102 may  
10 include a special media directory (SMD) 156, into which  
the directories of removable volumes are stored to form a  
media inventory. Each processor has a grooming lock 158,  
which is set during a grooming operation. During this  
period the grooming delete count of True File registry  
15 entries 140 is active, and no True Files should be  
deleted until grooming is complete. While grooming is in  
effect, grooming information includes a table of  
pathnames selected for deletion, and keeps track of the  
amount of space that would be freed if all of the files  
20 were deleted.

#### Primitive Mechanisms

The first of the mechanisms provided by the  
present invention, primitive mechanisms, are now  
described. The mechanisms described here depend on  
25 underlying data management mechanisms to create, copy,  
read, and delete data items in the True File registry  
126, as identified by a True File ID. This support may  
be provided by an underlying operating system or disk  
storage manager.

30 The following primitive mechanisms are  
described:

1. Calculate True Name;
2. Assimilate Data Item;

007040-09EE60

3. New True File;
4. Get True Name from Path;
5. Link Path to True Name;
6. Realize True File from Location;
- 5 7. Locate Remote File;
8. Make True File Local;
9. Create Scratch File;
10. Freeze Directory;
11. Expand Frozen Directory;
12. Delete True File;
13. Process Audit File Entry;
14. Begin Grooming;
15. Select For Removal; and
16. End Grooming.

15 1. Calculate True Name

A True Name is computed using a function, MD, which reduces a data block B of arbitrary length to a relatively small, fixed size identifier, the True Name of the data block, such that the True Name of the data block is virtually guaranteed to represent the data block B and only data block B.

The function MD must have the following properties:

- 25 1. The domain of the function MD is the set of all data items. The range of the function MD is the set of True Names.
2. The function MD must take a data item of arbitrary length and reduce it to an integer value in the range 0 to N-1, where  
30 N is the cardinality of the set of True Names. That is, for an arbitrary length data block B,  $0 \leq MD(B) < N$ .
3. The results of MD(B) must be evenly and randomly distributed over the range of N,  
35 in such a way that simple or regular

23

changes to B are virtually guaranteed to produce a different value of MD(B).

4. It must be computationally difficult to find a different value B' such that MD(B)=MD(B').
5. The function MD(B) must be efficiently computed.

A family of functions with the above properties are the so-called message digest functions, which are used in digital security systems as techniques for authentication of data. These functions (or algorithms) include MD4, MD5, and SHA.

In the presently preferred embodiments, either MD5 or SHA is employed as the basis for the computation of True Names. Whichever of these two message digest functions is employed, that same function must be employed on a system-wide basis.

It is impossible to define a function having a unique output for each possible input when the number of elements in the range of the function is smaller than the number of elements in its domain. However, a crucial observation is that the actual data items that will be encountered in the operation of any system embodying this invention form a very sparse subset of all the possible inputs.

A colliding set of data items is defined as a set wherein, for one or more pairs x and y in the set, MD(x) = MD(y). Since a function conforming to the requirements for MD must evenly and randomly distribute its outputs, it is possible, by making the range of the function large enough, to make the probability arbitrarily small that actual inputs encountered in the operation of an embodiment of this invention will form a colliding set.

To roughly quantify the probability of a collision, assume that there are no more than  $2^{30}$  storage devices in the world, and that each storage device has an

0982310-0972260

average of at most  $2^{20}$  different data items. Then there are at most  $2^{50}$  data items in the world. If the outputs of MD range between 0 and  $2^{128}$ , it can be demonstrated that the probability of a collision is approximately 1 in  $2^{29}$ . Details on the derivation of these probability values are found, for example, in P. Flajolet and A.M. Odlyzko, "Random Mapping Statistics," *Lecture Notes in Computer Science 434: Advances in Cryptology -- Eurocrypt '89 Proceedings*, Springer-Verlag, pp. 329-354.

Note that for some less preferred embodiments of the present invention, lower probabilities of uniqueness may be acceptable, depending on the types of applications and mechanisms used. In some embodiments it may also be useful to have more than one level of True Names, with some of the True Names having different degrees of uniqueness. If such a scheme is implemented, it is necessary to ensure that less unique True Names are not propagated in the system.

While the invention is described herein using only the True Name of a data item as the identifier for the data item, other preferred embodiments use tagged, typed, categorized or classified data items and use a combination of both the True Name and the tag, type, category or class of the data item as an identifier. Examples of such categorizations are files, directories, and segments; executable files and data files, and the like. Examples of classes are classes of objects in an object-oriented system. In such a system, a lower degree of True Name uniqueness is acceptable over the entire universe of data items, as long as sufficient uniqueness is provided per category of data items. This is because the tags provide an additional level of uniqueness.

A mechanism for calculating a True Name given a data item is now described, with reference to FIGURES 10(a) and 10(b).

A simple data item is a data item whose size is less than a particular given size (which must be defined

25



considered desirable features in the preferred embodiment.

2. Assimilate Data Item

5 A mechanism for assimilating a data item  
(scratch file or segment) into a file system, given the  
scratch file ID of the data item, is now described with  
reference to FIGURE 11. The purpose of this mechanism is  
to add a given data item to the True File registry 126.  
10 If the data item already exists in the True File registry  
126, this will be discovered and used during this  
process, and the duplicate will be eliminated.

Thereby the system stores at most one copy of  
any data item or file by content, even when multiple  
names refer to the same content.

15 First, determine the True Name of the data item  
corresponding to the given scratch File ID using the  
Calculate True Name primitive mechanism (Step S230).  
Next, look for an entry for the True Name in the True  
File registry 126 (Step S232) and determine whether a  
20 True Name entry, record 140, exists in the True File  
registry 126. If the entry record includes a  
corresponding True File ID or compressed File ID (Step  
S237), delete the file with the scratch File ID (Step  
S238). Otherwise store the given True File ID in the  
25 entry record (step S239).

If it is determined (in step S232) that no True  
Name entry exists in the True File registry 126, then, in  
Step S236, create a new entry in the True File registry  
126 for this True Name. Set the True Name of the entry  
30 to the calculated True Name, set the use count for the  
new entry to one, store the given True File ID in the  
entry and set the other fields of the entry as  
appropriate.

27

Because this procedure may take some time to compute, it is intended to run in background after a file has ceased to change. In the meantime, the file is considered an unassimilated scratch file.

5 3. New True File

The New True File process is invoked when processing the audit file 132, some time after a True File has been assimilated (using the Assimilate Data Item primitive mechanism). Given a local directory extensions table entry record 138 in the local directory extensions table 124, the New True File process can provide the following steps (with reference to FIGURE 12), depending on how the local processor is configured:

15 First, in step S238, examine the local directory extensions table entry record 138 to determine whether the file is locked by a cache server. If the file is locked, then add the ID of the cache server to the dependent processor list of the True File registry table 126, and then send a message to the cache server to update the cache of the current processor using the Update Cache remote mechanism (Step 242).

20 If desired, compress the True File (Step S246), and, if desired, mirror the True File using the Mirror True File background mechanism (Step S248).

25 4. Get True Name from Path

The True Name of a file can be used to identify a file by contents, to confirm that a file matches its original contents, or to compare two files. The mechanism to get a True Name given the pathname of a file is now described with reference to FIGURE 13.

30 First, search the local directory extensions table 124 for the entry record 138 with the given pathname (Step S250). If the pathname is not found, this process fails and no True Name corresponding to the given pathname exists. Next, determine whether the local

28



directory extensions table entry record 138 includes a True Name (Step S252), and if so, the mechanism's task is complete. Otherwise, determine whether the local directory extensions table entry record 138 identifies a  
5 directory (Step S254), and if so, freeze the directory (Step S256) (the primitive mechanism Freeze Directory is described below).

Otherwise, in step S258, assimilate the file (using the Assimilate Data Item primitive mechanism)  
10 defined by the File ID field to generate its True Name and store its True Name in the local directory extensions entry record. Then return the True Name identified by the local directory extensions table 124.

5. Link Path to True Name

15 The mechanism to link a path to a True Name provides a way of creating a new directory entry record identifying an existing, assimilated file. This basic process may be used to copy, move, and rename files without a need to copy their contents. The mechanism to  
20 link a path to a True Name is now described with reference to FIGURE 14.

First, if desired, confirm that the True Name exists locally by searching for it in the True Name registry or local directory extensions table 135 (Step  
25 S260). Most uses of this mechanism will require this form of validation. Next, search for the path in the local directory extensions table 135 (Step S262). Confirm that the directory containing the file named in the path already exists (Step S264). If the named file  
30 itself exists, delete the File using the Delete True File operating system mechanism (see below) (Step S268).

Then, create an entry record in the local directory extensions with the specified path (Step S270) and update the entry record and other data structures as  
35 follows: fill in the True Name field of the entry with the specified True Name; increment the use count for the

True File registry entry record 140 of the corresponding True Name; note whether the entry is a directory by reading the True File to see if it contains a tag (magic number) indicating that it represents a frozen directory (see also the description of the Freeze Directory primitive mechanism regarding the tag); and compute and set the other fields of the local directory extensions appropriately. For instance, search the region table 128 to identify the region of the path, and set the time of last access and time of last modification to the current time.

6. Realize True File from Location

This mechanism is used to try to make a local copy of a True File, given its True Name and the name of a source location (processor or media) that may contain the True File. This mechanism is now described with reference to FIGURE 15.

First, in step S272, determine whether the location specified is a processor. If it is determined that the location specified is a processor, then send a Request True File message (using the Request True File remote mechanism) to the remote processor and wait for a response (Step S274). If a negative response is received or no response is received after a timeout period, this mechanism fails. If a positive response is received, enter the True File returned in the True File registry 126 (Step S276). (If the file received was compressed, enter the True File ID in the compressed File ID field.)

If, on the other hand, it is determined in step S272 that the location specified is not a processor, then, if necessary, request the user or operator to mount the indicated volume (Step S278). Then (Step S280) find the indicated file on the given volume and assimilate the file using the Assimilate Data Item primitive mechanism. If the volume does not contain a True File registry 126, search the media inventory to find the path of the file

30

on the volume. If no such file can be found, this mechanism fails.

At this point, whether or not the location is determined (in step S272) to be a processor, if desired, verify the True File (in step S282).

7. Locate Remote File

D3 This mechanism allows a processor to locate a file or data item from a remote source of True Files, when a specific source is unknown or unavailable. A client processor system may ask one of several or many sources whether it can supply a data object with a given True Name. The steps to perform this mechanism are as follows (with reference to FIGURE 16).

The client processor 102 uses the source table 145 to select one or more source processors (Step S284). If no source processor can be found, the mechanism fails. Next, the client processor 102 broadcasts to the selected sources a request to locate the file with the given True Name using the Locate True File remote mechanism (Step S286). The request to locate may be augmented by asking to propagate this request to distant servers. The client processor then waits for one or more servers to respond positively (Step S288). After all servers respond negatively, or after a timeout period with no positive response, the mechanism repeats selection (Step S284) to attempt to identify alternative sources. If any selected source processor responds, its processor ID is the result of this mechanism. Store the processor ID in the source field of the True File registry entry record 140 of the given True Name (Step S290).

If the source location of the True Name is a different processor or medium than the destination (Step S290a), perform the following steps:

(i) Look up the True File registry entry record 140 for the corresponding True Name, and add the

source location ID to the list of sources for the True Name (Step S290b); and

5 (ii) If the source is a publishing system, determine the expiration date on the publishing system for the True Name and add that to the list of sources. If the source is not a publishing system, send a message to reserve the True File on the source processor (Step S290c).

10 Source selection in step S284 may be based on optimizations involving general availability of the source, access time, bandwidth, and transmission cost, and ignoring previously selected processors which did not respond in step S288.

8. Make True File Local

15 <sup>D4</sup> This mechanism is used when a True Name is known and a locally accessible copy of the corresponding file or data item is required. This mechanism makes it possible to actually read the data in a True File. The mechanism takes a True Name and returns when there is a local, accessible copy of the True File in the True File registry 126. This mechanism is described here with reference to the flow chart of FIGURE 17.

20 First, look in the True File registry 126 for a True File entry record 140 for the corresponding True Name (Step S292). If no such entry is found this mechanism fails. If there is already a True File ID for the entry (Step S294), this mechanism's task is complete. If there is a compressed file ID for the entry (Step S296), decompress the file corresponding to the file ID (Step S298) and store the decompressed file ID in the entry (Step S300). This mechanism is then complete.

30 If there is no True File ID for the entry (Step S294) and there is no compressed file ID for the entry (Step S296), then continue searching for the requested file. At this time it may be necessary to notify the user that the system is searching for the requested file.

32

If there are one or more source IDs, then select an order in which to attempt to realize the source ID (Step S304). The order may be based on optimizations involving general availability of the source, access  
5 time, bandwidth, and transmission cost. For each source in the order chosen, realize the True File from the source location (using the Realize True File from Location primitive mechanism), until the True File is realized (Step S306). If it is realized, continue with  
10 step S294. If no known source can realize the True File, use the Locate Remote File primitive mechanism to attempt to find the True File (Step S308). If this succeeds, realize the True File from the identified source location and continue with step S296.

15 9. Create Scratch File

~~15~~ A scratch copy of a file is required when a file is being created or is about to be modified. The scratch copy is stored in the file system of the underlying operating system. The scratch copy is  
20 eventually assimilated when the audit file record entry 146 is processed by the Process Audit File Entry primitive mechanism. This Create Scratch File mechanism requires a local directory extensions table entry record 138. When it succeeds, the local directory extensions  
25 table entry record 138 contains the scratch file ID of a scratch file that is not contained in the True File registry 126 and that may be modified. This mechanism is now described with reference to FIGURE 18.

First determine whether the scratch file should  
30 be a copy of the existing True File (Step S310). If so, continue with step S312. Otherwise, determine whether the local directory extensions table entry record 138 identifies an existing True File (Step S316), and if so, delete the True File using the Delete True File primitive  
35 mechanism (Step S318). Then create a new, empty scratch file and store its scratch file ID in the local directory

33

extensions table entry record 138 (Step S320). This mechanism is then complete.

5 If the local directory extensions table entry record 138 identifies a scratch file ID (Step S312), then the entry already has a scratch file, so this mechanism succeeds.

10 If the local directory extensions table entry record 138 identifies a True File (S316), and there is no True File ID for the True File (S312), then make the True File local using the Make True File Local primitive mechanism (Step S322). If there is still no True File ID, this mechanism fails.

15 There is now a local True File for this file. If the use count in the corresponding True File registry entry record 140 is one (Step S326), save the True File ID in the scratch file ID of the local directory extensions table entry record 138, and remove the True File registry entry record 140 (Step S328). (This step makes the True File into a scratch file.) This  
20 mechanism's task is complete.

25 Otherwise, if the use count in the corresponding True File registry entry record 140 is not one (in step S326), copy the file with the given True File ID to a new scratch file, using the Read File OS mechanism and store its file ID in the local directory extensions table entry record 138 (Step S330), and reduce the use count for the True File by one. If there is insufficient space to make a copy, this mechanism fails.

30 10. Freeze Directory  
~~Do~~ This mechanism freezes a directory in order to calculate its True Name. Since the True Name of a directory is a function of the files within the directory, they must not change during the computation of the True Name of the directory. This mechanism requires  
35 the pathname of a directory to freeze. This mechanism is described with reference to FIGURE 19.

In step S332, add one to the global freeze lock. Then search the local directory extensions table 124 to find each subordinate data file and directory of the given directory, and freeze each subordinate  
5 directory found using the Freeze Directory primitive mechanism (Step S334). Assimilate each unassimilated data file in the directory using the Assimilate Data Item primitive mechanism (Step S336). Then create a data item which begins with a tag or marker (a "magic number")  
10 being a unique data item indicating that this data item is a frozen directory (Step S337). Then list the file name and True Name for each file in the current directory (Step S338). Record any additional information required, such as the type, time of last access and modification,  
15 and size (Step S340). Next, in step S342, using the Assimilate Data Item primitive mechanism, assimilate the data item created in step S338. The resulting True Name is the True Name of the frozen directory. Finally, subtract one from the global freeze lock (Step S344).

20 11. Expand Frozen Directory

This mechanism expands a frozen directory in a given location. It requires a given pathname into which to expand the directory, and the True Name of the directory and is described with reference to FIGURE 20.

25 First, in step S346, make the True File with the given True Name local using the Make True File Local primitive mechanism. Then read each directory entry in the local file created in step S346 (Step S348). For each such directory entry, do the following:

30 Create a full pathname using the given pathname and the file name of the entry (Step S350); and

link the created path to the True Name (Step S352) using the Link Path to True Name primitive mechanism.





FIGURE 22, the steps for processing an entry are as follows:

5 Determine the operation in the entry 142 currently being processed (Step S365). If the operation indicates that a file was created or written (Step S366), then assimilate the file using the Assimilate Data Item primitive mechanism (Step S368), use the New True File primitive mechanism to do additional desired processing (such as cache update, compression, and mirroring) (Step 10 S369), and record the newly computed True Name for the file in the audit file record entry (Step S370).

15 Otherwise, if the entry being processed indicates that a compound data item or directory was copied (or deleted) (Step S376), then for each component True Name in the compound data item or directory, add (or subtract) one to the use count of the True File registry entry record 140 corresponding to the component True Name (Step S378).

20 In all cases, for each parent directory of the given file, update the size, time of last access, and time of last modification, according to the operation in the audit record (Step S379).

25 Note that the audit record is not removed after processing, but is retained for some reasonable period so that it may be used by the Synchronize Directory extended mechanism to allow a disconnected remote processor to update its representation of the local system.

14. Begin Grooming

30 This mechanism makes it possible to select a set of files for removal and determine the overall amount of space to be recovered. With reference to FIGURE 23, first verify that the global grooming lock is currently unlocked (Step S382). Then set the global grooming lock, set the total amount of space freed during grooming to 35 zero and empty the list of files selected for deletion

(Step S384). For each True File in the True File registry 126, set the delete count to zero (Step S386).

15. Select For Removal

5 This grooming mechanism tentatively selects a  
pathname to allow its corresponding True File to be  
removed. With reference to FIGURE 24, first find the  
local directory extensions table entry record 138  
corresponding to the given pathname (Step S388). Then  
find the True File registry entry record 140  
10 corresponding to the True File name in the local  
directory extensions table entry record 138 (Step S390).  
Add one to the grooming delete count in the True File  
registry entry record 140 and add the pathname to a list  
of files selected for deletion (Step S392). If the  
15 grooming delete count of the True File registry entry  
record 140 is equal to the use count of the True File  
registry entry record 140, and if there are no  
entries in the dependency list of the True File registry  
entry record 140, then add the size of the file indicated  
20 by the True File ID and or compressed file ID to the  
total amount of space freed during grooming (Step S394).

16. End Grooming

This grooming mechanism ends the grooming phase  
and removes all files selected for removal. With  
25 reference to FIGURE 25, for each file in the list of  
files selected for deletion, delete the file (Step S396)  
and then unlock the global grooming lock (Step S398).

Operating System Mechanisms

30 The next of the mechanisms provided by the  
present invention, operating system mechanisms, are now  
described.

The following operating system mechanisms are  
described:

1. Open File;

0928130 0101000  
00101010 00101010

- 2. Close File;
- 3. Read File;
- 4. Write File;
- 5. Delete File or Directory;
- 6. Copy File or Directory;
- 7. Move File or Directory;
- 8. Get File Status; and
- 9. Get Files in Directory.

1. Open File

D1 A mechanism to open a file is described with reference to FIGURE 26. This mechanism is given as input a pathname and the type of access required for the file (for example, read, write, read/write, create, etc.) and produces either the File ID of the file to be opened or an indication that no file should be opened. The local directory extensions table record 138 and region table record 142 associated with the opened file are associated with the open file for later use in other processing functions which refer to the file, such as read, write, and close.

First, determine whether or not the named file exists locally by examining the local directory extensions table 124 to determine whether there is an entry corresponding to the given pathname (Step S400). If it is determined that the file name does not exist locally, then, using the access type, determine whether or not the file is being created by this opening process (Step S402). If the file is not being created, prohibit the open (Step S404). If the file is being created, create a zero-length scratch file using an entry in local directory extensions table 124 and produce the scratch file ID of this scratch file as the result (Step S406).

If, on the other hand, it is determined in step S400 that the file name does exist locally, then determine the region in which the file is located by searching the region table 128 to find the record 142

39

with the longest region path which is a prefix of the file pathname (Step S408). This record identifies the region of the specified file.

5 Next, determine using the access type, whether the file is being opened for writing or whether it is being opened only for reading (Step S410). If the file is being opened for reading only, then, if the file is a scratch file (Step S419), return the scratch File ID of the file (Step S424). Otherwise get the True Name from  
10 the local directory extensions table 124 and make a local version of the True File associated with the True Name using the Make True File Local primitive mechanism, and then return the True File ID associated with the True Name (Step S420).

15 If the file is not being opened for reading only (Step S410), then, if it is determined by inspecting the region table entry record 142 that the file is in a read-only directory (Step S416), then prohibit the opening (Step S422).

20 If it is determined by inspecting the region table 128 that the file is in a cached region (Step S423), then send a Lock Cache message to the corresponding cache server, and wait for a return message (Step S418). If the return message says the file is  
25 already locked, prohibit the opening.

If the access type indicates that the file being modified is being rewritten completely (Step S419), so that the original data will not be required, then  
30 Delete the File using the Delete File OS mechanism (Step S421) and perform step S406. Otherwise, make a scratch copy of the file (Step S417) and produce the scratch file ID of the scratch file as the result (Step S424).

## 2. Close File

35 This mechanism takes as input the local directory extensions table entry record 138 of an open file and the data maintained for the open file. To close

a file, add an entry to the audit file indicating the time and operation (create, read or write). The audit file processing (using the Process Audit File Entry primitive mechanism) will take care of assimilating the file and thereby updating the other records.

3. Read File

To read a file, a program must provide the offset and length of the data to be read, and the location of a buffer into which to copy the data read.

The file to be read from is identified by an open file descriptor which includes a File ID as computed by the Open File operating system mechanism defined above. The File ID may identify either a scratch file or a True File (or True File segment). If the File ID identifies a True File, it may be either a simple or a compound True File. Reading a file is accomplished by the following steps:

In the case where the File ID identifies a scratch file or a simple True File, use the read capabilities of the underlying operating system.

In the case where the File ID identifies a compound file, break the read operation into one or more read operations on component segments as follows:

A. Identify the segment(s) to be read by dividing the specified file offset and length each by the fixed size of a segment (a system dependent parameter), to determine the segment number and number of segments that must be read.

B. For each segment number computed above, do the following:

i. Read the compound True File index block to determine the True Name of the segment to be read.

ii. Use the Realize True File from Location primitive mechanism to make the True File

41

segment available locally. (If that mechanism fails, the Read File mechanism fails).

5           iii. Determine the File ID of the True File specified by the True Name corresponding to this segment.

          iv. Use the Read File mechanism (recursively) to read from this segment into the corresponding location in the specified buffer.

4.           Write File

10           File writing uses the file ID and data management capabilities of the underlying operating system. File access (Make File Local described above) can be deferred until the first read or write.

5.           Delete File or Directory

15           38 > The process of deleting a file, for a given path name, is described here with reference to FIGURE 27.

          First, determine the local directory extensions table entry record 138 and region table entry record 142 for the file (Step S422). If the file has no local directory extensions table entry record 138 or is locked or is in a read-only region, prohibit the deletion.

20           Identify the corresponding True File given the True Name of the file being deleted using the True File registry 126 (Step S424). If the file has no True Name, (Step S426) then delete the scratch copy of the file based on its scratch file ID in the local directory extensions table 124 (Step S427), and continue with step S428.

30           If the file has a True Name and the True File's use count is one (Step S429), then delete the True File (Step S430), and continue with step S428.

          If the file has a True Name and the True File's use count is greater than one, reduce its use count by one (Step S431). Then proceed with step S428.

In Step S428, delete the local directory extensions table entry record, and add an entry to the audit file 132 indicating the time and the operation performed (delete).

5     6.         Copy File or Directory

A mechanism is provided to copy a file or directory given a source and destination processor and pathname. The Copy File mechanism does not actually copy the data in the file, only the True Name of the file.

10    This mechanism is performed as follows:

(A) Given the source path, get the True Name from the path. If this step fails, the mechanism fails.

(B) Given the True Name and the destination path, link the destination path to the True Name.

15         (C) If the source and destination processors have different True File registries, find (or, if necessary, create) an entry for the True Name in the True File registry table 126 of the destination processor. Enter into the source ID field of this new entry the source processor identity.

20         (D) Add an entry to the audit file 132 indicating the time and operation performed (copy).

This mechanism addresses capability of the system to avoid copying data from a source location to a destination location when the destination already has the data. In addition, because of the ability to freeze a directory, this mechanism also addresses capability of the system immediately to make a copy of any collection of files, thereby to support an efficient version control mechanisms for groups of files.

25     7.         Move File or Directory

A mechanism is described which moves (or renames) a file from a source path to a destination path. The move operation, like the copy operation, requires no actual transfer of data, and is performed as follows:

43

(A) Copy the file from the source path to the destination path.

(B) If the source path is different from the destination path, delete the source path.

5 8. Get File Status

This mechanism takes a file pathname and provides information about the pathname. First the local directory extensions table entry record 138 corresponding to the pathname given is found. If no such entry exists, then this mechanism fails, otherwise, gather information about the file and its corresponding True File from the local directory extensions table 124. The information can include any information shown in the data structures, including the size, type, owner, True Name, sources, time of last access, time of last modification, state (local or not, assimilated or not, compressed or not), use count, expiration date, and reservations.

9. Get Files in Directory

This mechanism enumerates the files in a directory. It is used (implicitly) whenever it is necessary to determine whether a file exists (is present) in a directory. For instance, it is implicitly used in the Open File, Delete File, Copy File or Directory, and Move File operating system mechanisms, because the files operated on are referred to by pathnames containing directory names. The mechanism works as follows:

The local directory extensions table 124 is searched for an entry 138 with the given directory pathname. If no such entry is found, or if the entry found is not a directory, then this mechanism fails.

If there is a corresponding True File field in the local directory extensions table record, then it is assumed that the True File represents a frozen directory. The Expand Frozen Directory primitive mechanism is used

44



to expand the existing True File into directory entries  
in the local directory extensions table.

5 Finally, the local directory extensions table  
124 is again searched, this time to find each directory  
subordinate to the given directory. The names found are  
provided as the result.

#### Remote Mechanisms

10 The remote mechanisms provided by the present  
invention are now described. Recall that remote  
mechanisms are used by the operating system in responding  
to requests from other processors. These mechanisms  
enable the capabilities of the present invention in a  
peer-to-peer network mode of operation.

15 In a presently preferred embodiment, processors  
communicate with each other using a remote procedure call  
(RPC) style interface, running over one of any number of  
communication protocols such as IPX/SPX or TCP/IP. Each  
peer processor which provides access to its True File  
20 registry 126 or file regions, or which depends on another  
peer processor, provides a number of mechanisms which can  
be used by its peers.

The following remote mechanisms are described:

1. Locate True File;
2. Reserve True File;
- 25 3. Request True File;
4. Retire True File;
5. Cancel Reservation;
6. Acquire True File;
7. Lock Cache;
- 30 8. Update Cache; and
9. Check Expiration Date.

#### 1. Locate True File

This mechanism allows a remote processor to  
determine whether the local processor contains a copy of

a specific True File. The mechanism begins with a True Name and a flag indicating whether to forward requests for this file to other servers. This mechanism is now described with reference to FIGURE 28.

5           First determine if the True File is available locally or if there is some indication of where the True File is located (for example, in the Source IDs field). Look up the requested True Name in the True File registry 126 (Step S432).

10           If a True File registry entry record 140 is not found for this True Name (Step S434), and the flag indicates that the request is not to be forwarded (Step S436), respond negatively (Step S438). That is, respond to the effect that the True File is not available.

15           One the other hand, if a True File registry entry record 140 is not found (Step S434), and the flag indicates that the request for this True File is to be forwarded (Step S436), then forward a request for this True File to some other processors in the system (Step  
20 S442). If the source table for the current processor identifies one or more publishing servers which should have a copy of this True File, then forward the request to each of those publishing servers (Step S436).

25           If a True File registry entry record 140 is found for the required True File (Step S434), and if the entry includes a True File ID or Compressed File ID (Step S440), respond positively (Step S444). If the entry includes a True File ID then this provides the identity or disk location of the actual physical representation of  
30 the file or file segment required. If the entry include a Compressed File ID, then a compressed version of the True File may be stored instead of, or in addition to, an uncompressed version. This field provides the identity of the actual representation of the compressed version of  
35 the file.

          If the True File registry entry record 140 is found (Step S434) but does not include a True File ID

46

(the File ID is absent if the actual file is not currently present at the current location) (Step S440), and if the True File registry entry record 140 includes one or more source processors, and if the request can be forwarded, then forward the request for this True File to one or more of the source processors (Step S444).

2. Reserve True File

This mechanism allows a remote processor to indicate that it depends on the local processor for access to a specific True File. It takes a True Name as input. This mechanism is described here.

(A) Find the True File registry entry record 140 associated with the given True File. If no entry exists, reply negatively.

(B) If the True File registry entry record 140 does not include a True File ID or compressed File ID, and if the True File registry entry record 140 includes no source IDs for removable storage volumes, then this processor does not have access to a copy of the given file. Reply negatively.

(C) Add the ID of the sending processor to the list of dependent processors for the True File registry entry record 140. Reply positively, with an indication of whether the reserved True File is on line or off line.

3. Request True File

This mechanism allows a remote processor to request a copy of a True File from the local processor. It requires a True Name and responds positively by sending a True File back to the requesting processor.

The mechanism operates as follows:

(A) Find the True File registry entry record 140 associated with the given True Name. If there is no such True File registry entry record 140, reply negatively.

(B) Make the True File local using the Make True File Local primitive mechanism. If this mechanism fails, the Request True File mechanism also fails.

5 (C) Send the local True File in either it is uncompressed or compressed form to the requesting remote processor. Note that if the True File is a compound file, the components are not sent.

(D) If the remote file is listed in the dependent process list of the True File registry entry  
10 record 140, remove it.

#### 4. Retire True File

This mechanism allows a remote processor to indicate that it no longer plans to maintain a copy of a given True File. An alternate source of the True File  
15 can be specified, if, for instance, the True File is being moved from one server to another. It begins with a True Name, a requesting processor ID, and an optional alternate source. This mechanism operates as follows:

(A) Find a True Name entry in the True File registry 126. If there is no entry for this True Name,  
20 this mechanism's task is complete.

(B) Find the requesting processor on the source list and, if it is there, remove it.

(C) If an alternate source is provided, add it  
25 to the source list for the True File registry entry record 140.

(D) If the source list of the True File registry entry record 140 has no items in it, use the Locate Remote File primitive mechanism to search for  
30 another copy of the file. If it fails, raise a serious error.

#### 5. Cancel Reservation

This mechanism allows a remote processor to indicate that it no longer requires access to a True File  
35 stored on the local processor. It begins with a True

Name and a requesting processor ID and proceeds as follows:

5 (A) Find the True Name entry in the True File registry 126. If there is no entry for this True Name, this mechanism's task is complete.

(B) Remove the identity of the requesting processor from the list of dependent processors, if it appears.

10 (C) If the list of dependent processors becomes zero and the use count is also zero, delete the True File.

6. Acquire True File

15 This mechanism allows a remote processor to insist that a local processor make a copy of a specified True File. It is used, for example, when a cache client wants to write through a new version of a file. The Acquire True File mechanism begins with a data item and an optional True Name for the data item and proceeds as follows:

20 (A) Confirm that the requesting processor has the right to require the local processor to acquire data items. If not, send a negative reply.

(B) Make a local copy of the data item transmitted by the remote processor.

25 (C) Assimilate the data item into the True File registry of the local processor.

(D) If a True Name was provided with the file, the True Name calculation can be avoided, or the mechanism can verify that the file received matches the True Name sent.

30 (E) Add an entry in the dependent processor list of the true file registry record indicating that the requesting processor depends on this copy of the given True File.

35 (F) Send a positive reply.

7. Lock Cache

This mechanism allows a remote cache client to lock a local file so that local users or other cache clients cannot change it while the remote processor is using it. The mechanism begins with a pathname and proceeds as follows:

(A) Find the local directory extensions table entry record 138 of the specified pathname. If no such entry exists, reply negatively.

(B) If an local directory extensions table entry record 138 exists and is already locked, reply negatively that the file is already locked.

(C) If an local directory extensions table entry record 138 exists and is not locked, lock the entry. Reply positively.

8. Update Cache

This mechanism allows a remote cache client to unlock a local file and update it with new contents. It begins with a pathname and a True Name. The file corresponding to the True Name must be accessible from the remote processor. This mechanism operates as follows:

Find the local directory extensions table entry record 138 corresponding to the given pathname. Reply negatively if no such entry exists or if the entry is not locked.

Link the given pathname to the given True Name using the Link Path to True Name primitive mechanism.

Unlock the local directory extensions table entry record 138 and return positively.

9. Check Expiration Date

Return current or new expiration date and possible alternative source to caller.

50

Background Processes and Mechanisms

The background processes and mechanisms provided by the present invention are now described. Recall that background mechanisms are intended to run occasionally and at a low priority to provide automated management capabilities with respect to the present invention.

The following background mechanisms are described:

1. Mirror True File;
2. Groom Region;
3. Check for Expired Links;
4. Verify Region; and
5. Groom Source List.

1. Mirror True File

This mechanism is used to ensure that files are available in alternate locations in mirror groups or archived on archival servers. The mechanism depends on application-specific migration/archival criteria (size, time since last access, number of copies required, number of existing alternative sources) which determine under what conditions a file should be moved. The Mirror True File mechanism operates as follows, using the True File specified, perform the following steps:

(A) Count the number of available locations of the True File by inspecting the source list of the True File registry entry record 140 for the True File. This step determines how many copies of the True File are available in the system.

(B) If the True File meets the specified migration criteria, select a mirror group server to which a copy of the file should be sent. Use the Acquire True File remote mechanism to copy the True File to the selected mirror group server. Add the identity of the selected system to the source list for the True File.

2. Groom Region

This mechanism is used to automatically free up space in a processor by deleting data items that may be available elsewhere. The mechanism depends on application-specific grooming criteria (for instance, a file may be removed if there is an alternate online source for it, it has not been accessed in a given number of days, and it is larger than a given size). This mechanism operates as follows:

Repeat the following steps (i) to (iii) with more aggressive grooming criteria until sufficient space is freed or until all grooming criteria have been exercised. Use grooming information to determine how much space has been freed. Recall that, while grooming is in effect, grooming information includes a table of pathnames selected for deletion, and keeps track of the amount of space that would be freed if all of the files were deleted.

(i) Begin Grooming (using the primitive mechanism).

(ii) For each pathname in the specified region, for the True File corresponding to the pathname, if the True File is present, has at least one alternative source, and meets application specific grooming criteria for the region, select the file for removal (using the primitive mechanism).

(iii) End Grooming (using the primitive mechanism).

If the region is used as a cache, no other processors are dependent on True Files to which it refers, and all such True Files are mirrored elsewhere. In this case, True Files can be removed with impunity. For a cache region, the grooming criteria would ordinarily eliminate the least recently accessed True Files first. This is best done by sorting the True Files in the region by the most recent access time before performing step (ii) above. The application specific

52



criteria would thus be to select for removal every True File encountered (beginning with the least recently used) until the required amount of free space is reached.

3. Check for Expired Links

5 This mechanism is used to determine whether dependencies on published files should be refreshed. The following steps describe the operation of this mechanism:

For each pathname in the specified region, for each True File corresponding to the pathname, perform the following step:

10 If the True File registry entry record 140 corresponding to the True File contains at least one source which is a publishing server, and if the expiration date on the dependency is past or close, then perform the following steps:

15 (A) Determine whether the True File registry entry record contains other sources which have not expired.

20 (B) Check the True Name expiration of the server. If the expiration date has been extended, or an alternate source is suggested, add the source to the True File registry entry record 140.

(C) If no acceptable alternate source was found in steps (A) or (B) above, make a local copy of the True File.

25 (D) Remove the expired source.

4. Verify Region

This mechanism can be used to ensure that the data items in the True File registry 126 have not been damaged accidentally or maliciously. The operation of this mechanism is described by the following steps:

30 (A) Search the local directory extensions table 124 for each pathname in the specified region and then perform the following steps:

(i) Get the True File name corresponding to the pathname;

(ii) If the True File registry entry 140 for the True File does not have a True File ID or compressed file ID, ignore it.

(iii) Use the Verify True File mechanism (see extended mechanisms below) to confirm that the True File specified is correct.

5. Groom Source List

The source list in a True File entry should be groomed sometimes to ensure there are not too many mirror or archive copies. When a file is deleted or when a region definition or its mirror criteria are changed, it may be necessary to inspect the affected True Files to determine whether there are too many mirror copies. This can be done with the following steps:

For each affected True File,

(A) Search the local directory extensions table to find each region that refers to the True File.

(B) Create a set of "required sources", initially empty.

(C) For each region found,  
(a) determine the mirroring criteria for that region,

(b) determine which sources for the True File satisfy the mirroring criteria, and

(c) add these sources to the set of required sources.

(D) For each source in the True File registry entry, if the source identifies a remote processor (as opposed to removable media), and if the source is not a publisher, and if the source is not in the set of required sources, then eliminate the source, and use the Cancel Reservation remote mechanism to eliminate the given processor from the list of dependent processors

recorded at the remote processor identified by the source.

Extended Mechanisms

5 The extended mechanisms provided by the present invention are now described. Recall that extended mechanisms run within application programs over the operating system to provide solutions to specific problems and applications.

10 The following extended mechanisms are described:

1. Inventory Existing Directory;
2. Inventory Removable, Read-only Files;
3. Synchronize Directories;
4. Publish Region;
- 15 5. Retire Directory;
6. Realize Directory at Location;
7. Verify True File;
8. Track for Accounting Purposes; and
9. Track for Licensing Purposes.

20 1. Inventory Existing Directory

This mechanism determines the True Names of files in an existing on-line directory in the underlying operating system. One purpose of this mechanism is to install True Name mechanisms in an existing file system.

25 An effect of such an installation is to eliminate immediately all duplicate files from the file system being traversed. If several file systems are inventoried in a single True File registry, duplicates across the volumes are also eliminated.

30 (A) Traverse the underlying file system in the operating system. For each file encountered, excluding directories, perform the following:

(i) Assimilate the file encountered (using the Assimilate File primitive mechanism). This

process computes its True Name and moves its data into the True File registry 126.

(ii) Create a pathname consisting of the path to the volume directory and the relative path of the file on the media. Link this path to the computed True Name using the Link Path to True Name primitive mechanism.

## 2. Inventory Removable, Read-only Files

A system with access to removable, read-only media volumes (such as WORM disks and CD-ROMs) can create a usable inventory of the files on these disks without having to make online copies. These objects can then be used for archival purposes, directory overlays, or other needs. An operator must request that an inventory be created for such a volume.

This mechanism allows for maintaining inventories of the contents of files and data items on removable media, such as diskettes and CD-ROMs, independent of other properties of the files such as name, location, and date of creation.

The mechanism creates an online inventory of the files on one or more removable volumes, such as a floppy disk or CD-ROM, when the data on the volume is represented as a directory. The inventory service uses a True Name to identify each file, providing a way to locate the data independent of its name, date of creation, or location.

The inventory can be used for archival of data (making it possible to avoid archiving data when that data is already on a separate volume), for grooming (making it possible to delete infrequently accessed files if they can be retrieved from removable volumes), for version control (making it possible to generate a new version of a CD-ROM without having to copy the old version), and for other purposes.



as a table of contents for the volume. It can be copied using the Copy File or Directory primitive mechanism to create an "overlay" directory which can then be modified, making it possible to edit a virtual copy of a read-only medium.

3. Synchronize Directories

Given two versions of a directory derived from the same starting point, this mechanism creates a new, synchronized version which includes the changes from each. Where a file is changed in both versions, this mechanism provides a user exit for handling the discrepancy. By using True Names, comparisons are instantaneous, and no copies of files are necessary.

This mechanism lets a local processor synchronize a directory to account for changes made at a remote processor. Its purpose is to bring a local copy of a directory up to date after a period of no communication between the local and remote processor. Such a period might occur if the local processor were a mobile processor detached from its server, or if two distant processors were run independently and updated nightly.

An advantage of the described synchronization process is that it does not depend on synchronizing the clocks of the local and remote processors. However, it does require that the local processor track its position in the remote processor's audit file.

This mechanism does not resolve changes made simultaneously to the same file at several sites. If that occurs, an external resolution mechanism such as, for example, operator intervention, is required.

The mechanism takes as input a start time, a local directory pathname, a remote processor name, and a remote directory pathname name, and it operates by the following steps:

(A) Request a copy of the audit file 132 from the remote processor using the Request True File remote mechanism.

5 (B) For each entry 146 in the audit file 132 after the start time, if the entry indicates a change to a file in the remote directory, perform the following steps:

(i) Compute the pathname of the corresponding file in the local directory. Determine the  
10 True Name of the corresponding file.

(ii) If the True Name of the local file is the same as the old True Name in the audit file, or if there is no local file and the audit entry indicates a new file is being created, link the new True Name in the  
15 audit file to the local pathname using the Link Path to True Name primitive mechanism.

(iii) Otherwise, note that there is a problem with the synchronization by sending a message to the operator or to a problem resolution program,  
20 indicating the local pathname, remote pathname, remote processor, and time of change.

(C) After synchronization is complete, record the time of the final change. This time is to be used as the new start time the next time this directory is  
25 synchronized with the same remote processor.

#### 4. Publish Region

The publish region mechanism allows a processor to offer the files in a region to any client processors for a limited period of time.

30 The purpose of the service is to eliminate any need for client processors to make reservations with the publishing processor. This in turn makes it possible for the publishing processor to service a much larger number of clients.

35 When a region is published, an expiration date is defined for all files in the region, and is propagated

into the publishing system's True File registry entry record 140 for each file.

5 When a remote file is copied, for instance using the Copy File operating system mechanism, the expiration date is copied into the source field of the client's True File registry entry record 140. When the source is a publishing system, no dependency need be created.

10 The client processor must occasionally and in background, check for expired links, to make sure it still has access to these files. This is described in the background mechanism Check for Expired Links.

#### 5. Retire Directory

15 This mechanism makes it possible to eliminate safely the True Files in a directory, or at least dependencies on them, after ensuring that any client processors depending on those files remove their dependencies. The files in the directory are not actually deleted by this process. The directory can be  
20 deleted with the Delete File operating system mechanism.

The mechanism takes the pathname of a given directory, and optionally, the identification of a preferred alternate source processor for clients to use. The mechanism performs the following steps:

25 (A) Traverse the directory. For each file in the directory, perform the following steps:

(i) Get the True Name of the file from its path and find the True File registry entry 140 associated with the True Name.

30 (ii) Determine an alternate source for the True File. If the source IDs field of the TFR entry includes the preferred alternate source, that is the alternate source. If it does not, but includes some other source, that is the alternate source. If it  
35 contains no alternate sources, there is no alternate source.



(iii) For each dependent processor in the True File registry entry 140, ask that processor to retire the True File, specifying an alternate source if one was determined, using the remote mechanism.

5 6. Realize Directory at Location

This mechanism allows the user or operating system to force copies of files from some source location to the True File registry 126 at a given location. The purpose of the mechanism is to ensure that files are accessible in the event the source location becomes  
10 inaccessible. This can happen for instance if the source or given location are on mobile computers, or are on removable media, or if the network connection to the source is expected to become unavailable, or if the  
15 source is being retired.

This mechanism is provided in the following steps for each file in the given directory, with the exception of subdirectories:

(A) Get the local directory extensions table entry record 138 given the pathname of the file. Get the True Name of the local directory extensions table entry record 138. This service assimilates the file if it has not already been assimilated.  
20

(B) Realize the corresponding True File at the given location. This service causes it to be copied to the given location from a remote system or removable media.  
25

7. Verify True File

This mechanism is used to verify that the data item in a True File registry 126 is indeed the correct data item given its True Name. Its purpose is to guard against device errors, malicious changes, or other problems.  
30

If an error is found, the system has the ability to "heal" itself by finding another source for  
35

61

the True File with the given name. It may also be desirable to verify that the error has not propagated to other systems, and to log the problem or indicate it to the computer operator. These details are not described here.

To verify a data item that is not in a True File registry 126, use the Calculate True Name primitive mechanism described above.

The basic mechanism begins with a True Name, and operates in the following steps:

(A) Find the True File registry entry record 140 corresponding to the given True Name.

(B) If there is a True File ID for the True File registry entry record 140 then use it. Otherwise, indicate that no file exists to verify.

(C) Calculate the True Name of the data item given the file ID of the data item.

(D) Confirm that the calculated True Name is equal to the given True Name.

(E) If the True Names are not equal, there is an error in the True File registry 126. Remove the True File ID from the True File registry entry record 140 and place it somewhere else. Indicate that the True File registry entry record 140 contained an error.

#### 8. Track for Accounting Purposes

This mechanism provides a way to know reliably which files have been stored on a system or transmitted from one system to another. The mechanism can be used as a basis for a value-based accounting system in which charges are based on the identity of the data stored or transmitted, rather than simply on the number of bits.

This mechanism allows the system to track possession of specific data items according to content by owner, independent of the name, date, or other properties of the data item, and tracks the uses of specific data items and files by content for accounting purposes. True

62

names make it possible to identify each file briefly yet uniquely for this purpose.

Tracking the identities of files requires maintaining an accounting log 134 and processing it for accounting or billing purposes. The mechanism operates in the following steps:

(A) Note every time a file is created or deleted, for instance by monitoring audit entries in the Process Audit File Entry primitive mechanism. When such an event is encountered, create an entry 148 in the accounting log 134 that shows the responsible party and the identity of the file created or deleted.

(B) Every time a file is transmitted, for instance when a file is copied with a Request True File remote mechanism or an Acquire True File remote mechanism, create an entry in the accounting log 134 that shows the responsible party, the identity of the file, and the source and destination processors.

(C) Occasionally run an accounting program to process the accounting log 134, distributing the events to the account records of each responsible party. The account records can eventually be summarized for billing purposes.

9. Track for Licensing Purposes

This mechanism ensures that licensed files are not used by unauthorized parties. The True Name provides a safe way to identify licensed material. This service allows proof of possession of specific files according to their contents without disclosing their contents.

Enforcing use of valid licenses can be active (for example, by refusing to provide access to a file without authorization) or passive (for example, by creating a report of users who do not have proper authorization).

One possible way to perform license validation is to perform occasional audits of employee systems. The



09282160 010100

5 The primitive mechanisms Calculate True Name and Assimilate Data Item support this property. For any given data item, using the Calculate True Name primitive mechanism, a substantially unique identifier or True Name for that data item can be determined.

10 Further, in operation of a DP system incorporating the present invention, multiple copies of data items are avoided (unless they are required for some reason such as backups or mirror copies in a fault-tolerant system). Multiple copies of data items are avoided even when multiple names refer to the same data item. The primitive mechanisms Assimilate Data Items and New True File support this property. Using the Assimilate Data Item primitive mechanism, if a data item already exists in the system, as indicated by an entry in the True File registry 126, this existence will be discovered by this mechanism, and the duplicate data item (the new data item) will be eliminated (or not added). Thus, for example, if a data file is being copied onto a system from a floppy disk, if, based on the True Name of the data file, it is determined that the data file already exists in the system (by the same or some other name), then the duplicate copy will not be installed. If the data item was being installed on the system by some name other than its current name, then, using the Link Path to True Name primitive mechanism, the other (or new) name can be linked to the already existing data item.

25 In general, the mechanisms of the present invention operate in such a way as to avoid recreating an actual data item at a location when a copy of that data item is already present at that location. In the case of a copy from a floppy disk, the data item (file) may have to be copied (into a scratch file) before it can be determined that it is a duplicate. This is because only one processor is involved. On the other hand, in a multiprocessor environment or DP system, each processor has a record of the True Names of the data items on that

65

processor. When a data item is to be copied to another location (another processor) in the DP system, all that is necessary is to examine the True Name of the data item prior to the copying. If a data item with the same True Name already exists at the destination location (processor), then there is no need to copy the data item. Note that if a data item which already exists locally at a destination location is still copied to the destination location (for example, because the remote system did not have a True Name for the data item or because it arrives as a stream of un-named data), the Assimilate Data Item primitive mechanism will prevent multiple copies of the data item from being created.

Since the True Name of a large data item (a compound data item) is derived from and based on the True Names of components of the data item, copying of an entire data item can be avoided. Since some (or all) of the components of a large data item may already be present at a destination location, only those components which are not present there need be copied. This property derives from the manner in which True Names are determined.

When a file is copied by the Copy File or Directory operating system mechanism, only the True Name of the file is actually replicated.

When a file is opened (using the Open File operating system mechanism), it uses the Make True File Local primitive mechanism (either directly or indirectly through the Create Scratch File primitive mechanism) to create a local copy of the file. The Open File operating system mechanism uses the Make True File Local primitive mechanism, which uses the Realize True File from Location primitive mechanism, which, in turn uses the Request True File remote mechanism.

The Request True File remote mechanism copies only a single data item from one processor to another. If the data item is a compound file, its component

66

segments are not copied, only the indirect block is copied. The segments are copied only when they are read (or otherwise needed).

5 The Read File operating system mechanism actually reads data. The Read File mechanism is aware of compound files and indirect blocks, and it uses the Realize True File from Location primitive mechanism to make sure that component segments are locally available, and then uses the operating system file mechanisms to  
10 read data from the local file.

Thus, when a compound file is copied from a remote system, only its True Name is copied. When it is opened, only its indirect block is copied. When the corresponding file is read, the required component  
15 segments are realized and therefore copied.

In operation data items can be accessed by reference to their identities (True Names) independent of their present location. The actual data item or True File corresponding to a given data identifier or True  
20 Name may reside anywhere in the system (that is, locally, remotely, offline, etc). If a required True File is present locally, then the data in the file can be accessed. If the data item is not present locally, there are a number of ways in which it can be obtained from  
25 wherever it is present. Using the source IDs field of the True File registry table, the location(s) of copies of the True File corresponding to a given True Name can be determined. The Realize True File from Location primitive mechanism tries to make a local copy of a True  
30 File, given its True Name and the name of a source location (processor or media) that may contain the True File. If, on the other hand, for some reason it is not known where there is a copy of the True File, or if the processors identified in the source IDs field do not  
35 respond with the required True File, the processor requiring the data item can make a general request for the data item using the Request True File remote

mechanism from all processors in the system that it can contact.

As a result, the system provides transparent access to any data item by reference to its data identity, and independent of its present location.

In operation, data items in the system can be verified and have their integrity checked. This is from the manner in which True Names are determined. This can be used for security purposes, for instance, to check for viruses and to verify that data retrieved from another location is the desired and requested data. For example, the system might store the True Names of all executable applications on the system and then periodically redetermine the True Names of each of these applications to ensure that they match the stored True Names. Any change in a True Name potentially signals corruption in the system and can be further investigated. The Verify Region background mechanism and the Verify True File extended mechanisms provide direct support for this mode of operation. The Verify Region mechanism is used to ensure that the data items in the True File registry have not been damaged accidentally or maliciously. The Verify True File mechanism verifies that a data item in a True File registry is indeed the correct data item given its True Name.

Once a processor has determined where (that is, at which other processor or location) a copy of a data item is in the DP system, that processor might need that other processor or location to keep a copy of that data item. For example, a processor might want to delete local copies of data items to make space available locally while knowing that it can rely on retrieving the data from somewhere else when needed. To this end the system allows a processor to Reserve (and cancel the reservation of) True Files at remote locations (using the remote mechanism). In this way the remote locations are

68



put on notice that another location is relying on the presence of the True File at their location.

5 A DP system employing the present invention can be made into a fault-tolerant system by providing a certain amount of redundancy of data items at multiple locations in the system. Using the Acquire True File and Reserve True File remote mechanisms, a particular processor can implement its own form of fault-tolerance by copying data items to other processors and then  
10 reserving them there. However, the system also provides the Mirror True File background mechanism to mirror (make copies) of the True File available elsewhere in the system. Any degree of redundancy (limited by the number of processors or locations in the system) can be  
15 implemented. As a result, this invention maintains a desired degree or level of redundancy in a network of processors, to protect against failure of any particular processor by ensuring that multiple copies of data items exist at different locations.

20 The data structures used to implement various features and mechanisms of this invention store a variety of useful information which can be used, in conjunction with the various mechanisms, to implement storage schemes and policies in a DP system employing the invention. For  
25 example, the size, age and location of a data item (or of groups of data items) is provided. This information can be used to decide how the data items should be treated. For example, a processor may implement a policy of deleting local copies of all data items over a certain  
30 age if other copies of those data items are present elsewhere in the system. The age (or variations on the age) can be determined using the time of last access or modification in the local directory extensions table, and the presence of other copies of the data item can be  
35 determined either from the Safe Flag or the source IDs, or by checking which other processors in the system have

copies of the data item and then reserving at least one of those copies.

In operation, the system can keep track of data items regardless of how those items are named by users (or regardless of whether the data items even have names). The system can also track data items that have different names (in different or the same location) as well as different data items that have the same name. Since a data item is identified by the data in the item, without regard for the context of the data, the problems of inconsistent naming in a DP system are overcome.

In operation, the system can publish data items, allowing other, possibly anonymous, systems in a network to gain access to the data items and to rely on the availability of these data items. True Names are globally unique identifiers which can be published simply by copying them. For example, a user might create a textual representation of a file on system A with True Name N (for instance as a hexadecimal string), and post it on a computer bulletin board. Another user on system B could create a directory entry F for this True Name N by using the Link Path to True Name primitive mechanism. (Alternatively, an application could be developed which hides the True Name from the users, but provides the same public transfer service.)

When a program on system B attempts to open pathname F linked to True Name N, the Locate Remote File primitive mechanism would be used, and would use the Locate True File remote mechanism to search for True Name N on one or more remote processors, such as system A. If system B has access to system A, it would be able to realize the True File (using the Realize True File from Location primitive mechanism) and use it locally. Alternatively, system B could find True Name N by accessing any publicly available True Name server, if the server could eventually forward the request to system A.

70

0922160 040390  
06F0D0 07E260

Clients of a local server can indicate that they depend on a given True File (using the Reserve True File remote mechanism) so that the True File is not deleted from the server registry as long as some client  
5 requires access to it. (The Retire True File remote mechanism is used to indicate that a client no longer needs a given True File.)

A publishing server, on the other hand, may want to provide access to many clients, and possibly  
10 anonymous ones, without incurring the overhead of tracking dependencies for each client. Therefore, a public server can provide expiration dates for True Files in its registry. This allows client systems to safely maintain references to a True File on the public server.  
15 The Check For Expired Links background mechanism allows the client of a publishing server to occasionally confirm that its dependencies on the publishing server are safe.

In a variation of this aspect of the invention, a processor that is newly connected (or reconnected after  
20 some absence) to the system can obtain a current version of all (or of needed) data in the system by requesting it from a server processor. Any such processor can send a request to update or resynchronize all of its directories (starting at a root directory), simply by using the  
25 Synchronize Directories extended mechanism on the needed directories.

Using the accounting log or some other user provided mechanism, a user can prove the existence of certain data items at certain times. By publishing (in a  
30 public place) a list of all True Names in the system on a given day (or at some given time), a user can later refer back to that list to show that a particular data item was present in the system at the time that list was published. Such a mechanism is useful in tracking, for  
35 example, laboratory notebooks or the like to prove dates of conception of inventions. Such a mechanism also

permits proof of possession of a data item at a particular date and time.

5 The accounting log file can also track the use of specific data items and files by content for accounting purposes. For instance, an information utility company can determine the data identities of data items that are stored and transmitted through its computer systems, and use these identities to provide bills to its customers based on the identities of the data items being transmitted (as defined by the substantially unique identifier). The assignment of prices for storing and transmitting specific True Files would be made by the information utility and/or its data suppliers; this information would be joined periodically with the information in the accounting log file to produce customer statements.

10 Backing up data items in a DP system employing the present invention can be done based on the True Names of the data items. By tracking backups using True Names, duplication in the backups is prevented. In operation, the system maintains a backup record of data identifiers of data items already backed up, and invokes the Copy File or Directory operating system mechanism to copy only those data items whose data identifiers are not recorded in the backup record. Once a data item has been backed up, it can be restored by retrieving it from its backup location, based on the identifier of the data item. Using the backup record produced by the backup to identify the data item, the data item can be obtained using, for example, the Make True File Local primitive mechanism.

15 In operation, the system can be used to cache data items from a server, so that only the most recently accessed data items need be retained. To operate in this way, a cache client is configured to have a local registry (its cache) with a remote Local Directory Extensions table (from the cache server). Whenever a

001010 0978260

file is opened (or read), the Local Directory Extensions table is used to identify the True Name, and the Make True File Local primitive mechanism inspects the local registry. When the local registry already has a copy,  
5 the file is already cached. Otherwise, the Locate True File remote mechanism is used to get a copy of the file. This mechanism consults the cache server and uses the Request True File remote mechanism to make a local copy, effectively loading the cache.

10 The Groom Cache background mechanism flushes the cache, removing the least-recently-used files from the cache client's True File registry. While a file is being modified on a cache client, the Lock Cache and Update Cache remote mechanisms prevent other clients from  
15 trying to modify the same file.

In operation, when the system is being used to cache data items, the problems of maintaining cache consistency are avoided.

20 To access a cache and to fill it from its server, a key is required to identify the data item desired. Ordinarily, the key is a name or address (in this case, it would be the pathname of a file). If the data associated with such a key is changed, the client's cache becomes inconsistent; when the cache client refers  
25 to that name, it will retrieve the wrong data. In order to maintain cache consistency it is necessary to notify every client immediately whenever a change occurs on the server.

30 By using an embodiment of the present invention, the cache key uniquely identifies the data it represents. When the data associated with a name changes, the key itself changes. Thus, when a cache client wishes to access the modified data associated with a given file name, it will use a new key (the True Name  
35 of the new file) rather than the key to the old file contents in its cache. The client will always request the correct data, and the old data in its cache will be

73

eventually aged and flushed by the Groom Cache background mechanism.

5 Because it is not necessary to immediately notify clients when changes on the cache server occur, the present invention makes it possible for a single server to support a much larger number of clients than is otherwise possible.

10 In operation, the system automatically archives data items as they are created or modified. After a file is created or modified, the Close File operating system mechanism creates an audit file record, which is eventually processed by the Process Audit File Entry primitive mechanism. This mechanism uses the New True File primitive mechanism for any file which is newly  
15 created, which in turn uses the Mirror True File background mechanism if the True File is in a mirrored or archived region. This mechanism causes one or more copies of the new file to be made on remote processors.

20 In operation, the system can efficiently record and preserve any collection of data items. The Freeze Directory primitive mechanism creates a True File which identifies all of the files in the directory and its subordinates. Because this True File includes the True Names of its constituents, it represents the exact  
25 contents of the directory tree at the time it was frozen. The frozen directory can be copied with its components preserved.

30 The Acquire True File remote mechanism (used in mirroring and archiving) preserves the directory tree structure by ensuring that all of the component segments and True Files in a compound data item are actually copied to a remote system. Of course, no transfer is necessary for data items already in the registry of the remote system.

35 In operation, the system can efficiently make a copy of any collection of data items, to support a version control mechanism for groups of the data items.

00223150-040390  
00223150-040390

The Freeze Directory primitive mechanism is used to create a collection of data items. The constituent files and segments referred to by the frozen directory are maintained in the registry, without any  
5 need to make copies of the constituents each time the directory is frozen.

Whenever a pathname is traversed, the Get Files in Directory operating system mechanism is used, and when it encounters a frozen directory, it uses the Expand  
10 Frozen Directory primitive mechanism.

A frozen directory can be copied from one pathname to another efficiently, merely by copying its True Name. The Copy File operating system mechanism is used to copy a frozen directory.

15 Thus it is possible to efficiently create copies of different versions of a directory, thereby creating a record of its history (hence a version control system).

In operation, the system can maintain a local  
20 inventory of all the data items located on a given removable medium, such as a diskette or CD-ROM. The inventory is independent of other properties of the data items such as their name, location, and date of creation.

The Inventory Existing Directory extended  
25 mechanism provides a way to create True File Registry entries for all of the files in a directory. One use of this inventory is as a way to pre-load a True File registry with backup record information. Those files in the registry (such as previously installed software)  
30 which are on the volumes inventoried need not be backed up onto other volumes.

The Inventory Removable, Read-only Files  
35 extended mechanism not only determines the True Names for the files on the medium, but also records directory entries for each file in a frozen directory structure. By copying and modifying this directory, it is possible to create an on line patch, or small modification of an

75

existing read-only file. For example, it is possible to create an online representation of a modified CD-ROM, such that the unmodified files are actually on the CD-ROM, and only the modified files are online.

5 In operation, the system tracks possession of specific data items according to content by owner, independent of the name, date, or other properties of the data item, and tracks the uses of specific data items and files by content for accounting purposes. Using the  
10 Track for Accounting Purposes extended mechanism provides a way to know reliably which files have been stored on a system or transmitted from one system to another.

#### True Names in Relational and Object-Oriented Databases

15 Although the preferred embodiment of this invention has been presented in the context of a file system, the invention of True Names would be equally valuable in a relational or object-oriented database. A relational or object-oriented database system using True Names would have similar benefits to those of the file  
20 system employing the invention. For instance, such a database would permit efficient elimination of duplicate records, support a cache for records, simplify the process of maintaining cache consistency, provide location-independent access to records, maintain archives  
25 and histories of records, and synchronize with distant or disconnected systems or databases.

The mechanisms described above can be easily modified to serve in such a database environment. The True Name registry would be used as a repository of  
30 database records. All references to records would be via the True Name of the record. (The Local Directory Extensions table is an example of a primary index that uses the True Name as the unique identifier of the desired records.)



In such a database, the operations of inserting, updating, and deleting records would be implemented by first assimilating records into the registry, and then updating a primary key index to map  
5 the key of the record to its contents by using the True Name as a pointer to the contents.

The mechanisms described in the preferred embodiment, or similar mechanisms, would be employed in such a system. These mechanisms could include, for  
10 example, the mechanisms for calculating true names, assimilating, locating, realizing, deleting, copying, and moving True Files, for mirroring True Files, for maintaining a cache of True Files, for grooming True Files, and other mechanisms based on the use of  
15 substantially unique identifiers.

While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiments, it is to be understood that the invention is not to be limited to the  
20 disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

77

WHAT IS CLAIMED IS:

1. In a data processing system, an apparatus comprising:

identity means for determining, for any of a plurality of data items in the system, a substantially unique identifier, said identifier depending on all of the data in the data item and only on the data in the data item; and

existence means for determining whether a particular data item is present in the system, by examining the identifiers of the plurality of data items.

2. An apparatus as in claim 1, further comprising:

local existence means for determining whether an instance of a particular data item is present at a particular location in the system, based on the identifier of the data item.

3. An apparatus as in claim 2, wherein each location contains a distinct plurality of data items, and wherein said local existence means determines whether a particular data item is present at a particular location in the system by examining the identifiers of the plurality of data items at said particular location in the system.

4. An apparatus as in claim 2, further comprising:

data associating means for making and maintaining, for a data item in the system, an association between the data item and the identifier of the data item; and

access means for accessing a particular data item using the identifier of the data item.

5. An apparatus as in claim 2, further comprising:

duplication means for copying a data item from a source to a destination in the data processing system, by providing said destination with the data item only if it is determined using the data identifier that the data item is not present at the destination.

6. An apparatus as in claim 4, further comprising:

assimilation means for assimilating a new data item into the system, said assimilation means invoking said identity means to determine the identifier of the new data item and invoking said data associating means to associate the new data item with its identifier.

7. An apparatus as in claim 4, further comprising:

duplication means for duplicating a data item from a source location to a destination location in the data processing system, based on the identifier of the data item, said duplication means invoking said local existence means to determine whether an instance of the data item is present at the destination location, and invoking said access means to provide said destination with the data item only if said local existence means determines that no instance of the data item is present at the destination.

8. An apparatus as in claim 7, further comprising:

backup means for making copies of data items in the system, said backup means maintaining a backup record of identifiers of data items backed up, and invoking duplication means to copy only those data items whose data identifiers are not recorded in the backup record.

9. An apparatus as in claim 8, further comprising:

recovery means for retrieving a data item previously backed up by said backup means, based on the identifier of the data item, said recovery means using the backup record to identify the data item, and invoking access means to retrieve the data item.

10. An apparatus as in claim 2, wherein a location is a computer among a network of computers, the apparatus further comprising:

remote existence means for determining whether a data item is present at a remote location in the system from a current location in the system, based on the identifier of the data item, said remote location using local existence means at the remote location to determine whether the data item is present at the remote location, and providing the current location with an indication of the presence of the data item at the remote location.

11. An apparatus as in claim 4, wherein a location is a computer among a network of computers, the apparatus further comprising:

requesting means for requesting a data item at a current location in the system from a remote location in the system, based on the identifier of the data item, said remote location using access means at the remote location to obtain the data item and to send it to the current location if it is present.

12. An apparatus as in claim 1, further comprising:

context means for making and maintaining a context association between at least one contextual name of a data item in the system and the identifier of the data item; and

referencing means for obtaining the identifier of a data item in the system given a contextual name for the data item, using said context association.

5 13. An apparatus as in claim 12, further comprising:

assignment means for assigning a data item to a contextual name, invoking said identity means to determine the identifier of the data item, and invoking said context means to make or modify the context association between the contextual name of the data item and the identifier of the data item.

14. An apparatus as in claim 12, further comprising:

15 data associating means for making and maintaining, for a data item in the system, an association between the data item and the identifier of the data item;

20 access means for accessing a particular data item using the identifier of the particular data item; and

contextual name access means for accessing a data item in the system for a given context name of the data item, determining the data identifier associated with the given context name, and invoking said access means to access the data item using the data identifier.

15. An apparatus as in claim 11, further comprising:

30 transparent access means for accessing a data item from one of several locations, using the identifier of the data item, said transparent access means invoking said local existence means to determine if the particular data item is present at the current location, and, in the case when the particular data item is not present at the

current location, invoking said requesting means to obtain the data item from a remote location.

16. An apparatus as in claim 15, further comprising:

5 identifier copy means for copying an identifier of a data item from a source location to a destination location.

17. An apparatus as in claim 15, further comprising:

10 context means for making and maintaining a context association between a contextual name of a data item in the system and the identifier of the data item;  
context copy means for copying a data item from a source location to a destination location, given the  
15 contextual name of the data item, by copying only the context association between the contextual identifier and the data identifier from the source location to the destination location; and

20 transparent referencing means for obtaining a data item from one of several locations the system given a contextual name for the data item, said transparent referencing means invoking said context association to determine the data identifier of a data item given a  
25 contextual name, and invoking said transparent access means to access the data item from one of several locations given the identifier of the data item.

18. An apparatus as in claim 1, wherein at least some of said data items are compound data items, each compound data item including at least some component  
30 data items in a fixed sequence, and wherein the identity means determines the identifier of a compound data item based on each component data item of the compound data item.

00000-092260

5 19. An apparatus as in claim 18, wherein said compound data items are files and said component data items are segments, and wherein the identity means determines the identifier of a file based on the identifier of each data segment of the file.

10 20. An apparatus as in claim 18, wherein said compound data items are directories and said component data items are files or subordinate directories, and wherein the identity means determines the identifier of a given directory based on each file and subordinate directory within the given directory.

15 21. An apparatus as in claim 11, further comprising:  
means for advertising a data item from a location in the system to at least one other location in the system, said means for advertising providing each of said at least one other location with the data identifier of the data item, and providing the data item to only those locations of said other locations that request said data item in response to said providing.

25 22. An apparatus as in claim 18, further comprising:  
local existence means for determining whether a particular data item is present at a particular location in the system, based on the identifier of the data item; and

30 compound copy means for copying a data item from a source to a destination in the data processing system, said compound copy means invoking said local existence means to determine whether the data item is present at the destination, and to determine, when the data item is a compound data item, whether the component data items of the compound data item are present at the destination, and providing said destination with the data





context associating means for making and  
maintaining a context association, for any data item in  
the system, between the identifier of the data item and  
at least one contextual name of the data item at a  
5 particular location in the system;

means for obtaining the identifier of a data  
item in the system given a contextual name for the data  
item at a particular location in the system; and

10 logical copy means for associating the data  
identifier corresponding to a contextual name at a source  
location with a contextual name at a destination location  
in the data processing system.

27. An apparatus as in claim 25, wherein said  
15 compound data items are files and said component data  
items are segments, and wherein the identity means  
determines the identifier of a file based on the  
identifier of each data segment of the file.

28. An apparatus as in claim 25, further  
comprising:

20 compound copy means for copying a data item  
from a source location to a destination location in the  
data processing system, said compound copy means invoking  
said local existence means to determine whether the data  
item is present at the destination, and to determine,  
25 when the data item is a compound data item, whether the  
component data items of the compound data item are  
present at the destination, and providing said  
destination with the data item only if said local  
existence means determines that the data item is not  
30 present at the destination, and providing said  
destination with each component data item only if said  
local existence means determines that the component data  
item is not present at the destination.

29. An apparatus as in any of claims 1-28, wherein a data item is at least one of a file, a database record, a message, a data segment, a data block, a directory, and an instance an object class.

5 30. A method of identifying a data item in a data processing system for subsequent access to the data item, the method comprising the steps of:  
determining a substantially unique identifier for the data item, said identifier depending on all of  
10 the data in the data item and on the data in the data item; and  
accessing a data item in the system using the identifier of the data item.

15 31. A method as in claim 30, further comprising the step of:  
making and maintaining, for a plurality of data items in the system, an association between each of the data items and the identifier of each of the data items,  
20 wherein said accessing step accesses a data item via the association.

25 32. A method as in claim 31, further comprising the step of:  
assimilating a new data item into the system, by determining the identifier of the new data item and associating the new data item with its identifier.

30 33. A method for duplicating a given data item from a source location to a destination location in a data processing system, the method comprising the steps of:  
determining a substantially unique identifier for the given data item, said identifier depending on all of the data in the data item and only on the data in the data item;

determining, using said data identifier,  
whether said data item is present at said destination  
location; and

5 based on said determining, providing said  
destination location with said data item only if said  
data item is not present at said destination.

10 34. A method as in claim 33, wherein said  
given data item is a compound data item having a  
plurality of component data items, the method further  
comprising the steps of:

for each data item of said component data  
items,

15 obtaining the component data  
identifier of the data item by determining a  
substantially unique identifier for the data  
item, said identifier depending on all of the  
data in the data item and only on the data in  
the data item;

20 determining, using said obtained  
component data identifier, whether said data  
item is present at said destination; and  
based on said determining, providing  
said destination with said data item only if  
said data item is not present at said  
25 destination.

35. A method for determining whether a  
particular data item is present in a data processing  
system, the method comprising the steps of:

30 (A) for each data item of a plurality of data  
items in the system,

(i) determining a substantially unique  
identifier for the data item, said identifier  
depending on all of the data in the data item  
and only on the data in the data item; and

(ii) making and maintaining a set of identifiers of said plurality of data items; and

(B) for the particular data item,

(i) determining a particular substantially unique identifier for the data item, said identifier depending on all of the data in the data item and only on the data in the data item; and

(ii) determining whether said particular identifier is in said set of data items.

36. A method of backing up, of a plurality of data items, data items modified since a previous backup time in a data processing system, the method comprising the steps of:

(A) maintaining a backup record of identifiers of data items backed up at the previous backup time; and

(B) for each of said plurality of data items, (i) determining a substantially unique identifier for the data item, said identifier depending on all of the data in the data item and only on the data in the data item;

(ii) determining those data items of the plurality of data items whose identifiers are not in the backup record; and

(iii) based on said determining, copying only those data items whose data identities are not recorded in the backup record.

37. A method as in claim 36, further comprising the step of:

recording in the backup record the identifiers of those data items copied in said step of copying.

38. A method of locating a particular data item at a location in a data processing system, the method comprising the steps of:

(A) determining a substantially unique identifier for the data item, said identifier depending on all of the data in the data item and only on the data in the data item;

(B) requesting the particular data item by sending the data identifier of the data item from the requestor location to at least one location of a plurality of provider locations in the system; and

(C) on at least some of said provider locations,

(a) for each data item of a plurality of data items at said provider locations,

(i) determining a substantially unique identifier for the data item, said identifier depending on all of the data in the data item and only on the data in the data item; and

(ii) making and maintaining a set of identifiers of data items,

(b) determining, based on said set of identifiers, whether the data item corresponding to the requested data identifier is present at said provider location; and

(c) based on said determining, when said provider location determines that the particular data item is present at the provider location, notifying said requestor that the provider has a copy of the given data item.

39. The method of claim 38, further comprising the steps of:

5 (a) for each data item of a plurality of data items at said provider locations, making and maintaining an association between the data item and the identifier of the data item,

10 (b) in response to said notifying, said client location copying said data item from one of said responding remote locations, using said association to access the data item given the data identifier.

40. A method of locating a particular data item among a plurality of locations, each of said locations having a plurality of data items, the method comprising the steps of:

15 determining, for the particular data item and for each data item of the plurality of data items, a substantially unique identifier for the data item, said identifier depending on all of the data in the data item and only on the data in the data item; and

20 determining the presence of the particular data item in each of said plurality of locations by determining whether the identifier of the particular data item is present at each of said locations.

41. The method of claim 30, wherein said step of accessing further comprises the steps of, for a given data identifier and for a given current location and a remote location in the system:

30 determining whether the data item corresponding to the given data identifier is present at the current location, and

based on said determining, if said data item is not present at the current location, fetching the data

item from a remote location in the system to the current location.

42. The method of claim 41, further comprising the steps of:

5 for each contextual name at a location,  
making and maintaining a context  
association between the context name of a data item and  
the identifier of said data item, and when some context  
association changes at said current location, and  
10 notifying said remote location of a  
modification to the context association.

43. The method of claim 42, further comprising the step of:

15 at said remote location, updating the  
association between the contextual identifier of the data  
item and the identifier of the data item.

44. The method of claim 43, further comprising the step of:

20 from said remote location, notifying all other  
locations that said data item has been modified, by  
providing the contextual identifier and data identifier  
of said data item to said other locations.

45. The method of claim 44, further comprising the step of, at each location notified that the data item  
25 has been modified:

modifying an association between the contextual  
identifier of the data item and the data identifier of  
the data item, to record that the data item has been  
modified.

30 46. A method of eliminating a data item at a  
given location in a data processing system when said data

item can be obtained from another location in the system,  
the method comprising the steps of:

5           determining a substantially unique identifier  
for the data, said identifier depending on all of the  
data in the data item and only on the data in the data  
item;

          making and maintaining a source association  
between the data identifier and at least one location at  
which said data item is known to be present; and

10           based on said source association, if said data  
item is present at said other location, removing the data  
item from the given location.

47. A method of deleting a data item from a  
location in a data processing system, the method  
15           comprising the steps of:

          for each of a plurality of data items in the  
system:

20           determining a substantially unique identifier  
for the data, said identifier depending on all of the  
data in the data item and only on the data in the data  
item; and

          making and maintaining, an association between  
each of the data items and the unique identifier of the  
data items; and

25           for a given data item:

          determining a substantially unique identifier  
for the data, said identifier depending on all of the  
data in the data item and only on the data in the data  
item; and

30           determining whether a contextual identifier or  
a compound data item or a remote processor in the system  
refers to the unique identifier of the data item, and  
based on said determining, deleting said data item and  
its association if no other contextual identifier or  
35           compound data item or remote processor refers to said  
data item.



00000516250

5           48. The method of claim 47, wherein said determining is based on a use count for the data item, and wherein said data item is deleted only if said use count indicates that no other contextual identifier or compound data item or remote processor in the system refers to the data item.

10           49. A method of substantially synchronizing data items at a client location in a data processing system after a period of independent changes on the client and another location in the system, given a context, the method comprising the steps of:  
          making and maintaining a list of changes to the context association between each context name of a data item and the identifier of said data item, in the given context and during the period of independent change;  
15           obtaining the list of changes from the other location for the given context; and,  
          for each context name in the list of changes updating the context identifier  
20           associations at the client whenever it is determined that the context association of the given context name changed either only at the client or only at the other location during the period of independent changes; and  
          performing a conflict-resolution task such  
25           as notifying an operator of the client location, whenever it is determined that the context association changed at both the client and the other location.

30           50. A method as in claim 49, wherein said lists are maintained as queues based on a temporal order, and wherein, at said client location, said replacing is based on said temporal order.

          51. A method of maintaining at least a predetermined number of copies of a given data item in a data processing system, at different locations in the

00704D-057E8E00

data processing system, said data processing system being one wherein data is identified by a substantially unique identifier, said identifier depending on all of the data in the data item and only on the data in the data item, and wherein any data item in the system may be accessed using only the identifier of the data item, the method comprising the steps of:

- (i) sending, from a first location in the system, the data identifier of the given data item to other locations in the system; and
- (ii) in response to said sending, at each of said other locations,
  - (A) determining whether the data item corresponding to the data identifier is present at the other location, and based on said determining, and
  - (B) informing said first location whether said data item is present at the other location; and
  - (iii) in response to said informing from said other locations, at said first location,
    - (A) determining whether said data item is present in at least the predetermined number of other locations, and based on said determining,
    - (B) when less than the predetermined number of other locations have a copy of the data item, requesting some locations that do not have a copy of the data item make a copy of the data item.

52. A method as in claim 51, wherein said step (iii) further comprises the step of:

- (C) when more than the predetermined number of other locations have a copy of the data item, requesting some locations that do have a copy of the data item delete the copy of the data item.

~~53. A method as in any of claims 30-52,  
wherein said data items are at least one of a file, a  
database record, a message, a data segment, a data block,  
a directory, and an instance of an object class.~~

ADD B17

ADD C27

66T040D\*05T02260

INS. C4

ABSTRACT OF THE DISCLOSURE

IDENTIFYING DATA IN A DATA PROCESSING SYSTEM

5 In a data processing system, a mechanism identifies data items by substantially unique identifiers which depend on all of the data in the data items and only on the data in the data items. Existence means determine whether a particular data item is present in the system, by examining the identifiers of the plurality of data items.

001070-01E250

FOR UTILITY/DESIGN  
CIP/PCT NATIONAL/PLANT  
ORIGINAL/SUBSTITUTE/SUPPLEMENTAL  
DECLARATIONS

RULE 63 (37 C.F.R. 1.63)  
DECLARATION AND POWER OF ATTORNEY  
FOR PATENT APPLICATION  
IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

CUSHMAN  
FORM

As a below named inventor, I hereby declare that my residence, post office address and citizenship are as stated below next to my name, and I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the INVENTION ENTITLED

IDENTIFYING DATA IN A DATA PROCESSING SYSTEM

the specification of which (CHECK applicable BOX(ES))

- X  
-> [ ] is attached hereto.  
-> [ x ] was filed on April 11, 1995 as U.S. Application No. 08 / 425,160  
BOX(ES) -> [ ] was filed as PCT International Application No. PCT/ / on  
-> -> and (if applicable to U.S. or PCT application) was amended on

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose all information known to me to be material to patentability as defined in 37 C.F.R. 1.56. I hereby claim foreign priority benefits under 35 U.S.C. 119/365 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate filed by me or my assignee disclosing the subject matter claimed in this application and having a filing date (1) before that of the application on which priority is claimed, or (2) if no priority claimed, before the filing date of this application:

PRIOR FOREIGN APPLICATION(S)		Date first Laid-	Date Patented	Priority Claimed	
Number	Country	open or Published	or Granted	Yes	No

I hereby claim the benefit under 35 U.S.C. 120/365 of all United States applications listed below and PCT international applications listed above or below and, if this is a continuation-in-part (CIP) application, insofar as the subject matter disclosed and claimed in this application is in addition to that disclosed in such prior applications, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in 37 C.F.R. 1.56 which became available between the filing date of each such prior application and the national or PCT international filing date of this application:

PRIOR U.S. OR PCT APPLICATION(S)	Day/MONTH/Year Filed	Status
Application No. (series code/serial no.)		pending, abandoned, patented

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

And I hereby appoint Cushman Darby & Cushman, L.L.P. 1100 New York Avenue, N.W., Ninth Floor, East Tower Washington, D.C. 20005-3918, telephone number 861-3000 (to whom all communications are to be directed), and the below-named persons (of the same address) individually and collectively my attorneys to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith and with the resulting patent, and I hereby authorize them to act and rely on instructions from and communicate directly with the person/assignee/attorney/firm/ organization who/which first sends/sent this case to them and by whom/which I hereby declare that I have consented after full disclosure to be represented unless/until I instruct Cushman, Darby & Cushman in writing to the contrary.

Paul N. Kokulis	16773	Edward M. Prince	22429	Dale S. Lazar	28872	Michelle N. Lester	32331
Raymond F. Lippitt	17519	Donald B. Deaver	23048	Glenn J. Perry	28458	Jeffrey A. Simenauer	31933
G. Lloyd Knight	17698	David W. Brinkman	20817	Kendrew H. Colton	30368	Robert A. Molan	29834
Carl G. Love	18781	George M. Sirilla	18221	Chris Comuntzis	31097	G. Paul Edgell	24238
Edgar H. Martin	20534	Donald J. Bird	25323	Wallace G. Walter	27843	Lynn E. Eccleston	35861
William K. West, Jr.	22057	W. Warren Taltavull	25647	Lawrence Harbin	27644	Frederick S. Frei	27105
Kevin E. Joyce	20508	Peter W. Gowdey	25872	Paul E. White, Jr.	32011	David A. Jakopin	32995
1. INVENTOR'S SIGNATURE: <u>David A. Farber</u>						Mark G. Paulson	30793

1. INVENTOR'S SIGNATURE: David A. Farber Date 6/7/95  
Inventor's Name (typed) David A. FARBER U.S.A.  
Residence (City) Ojai First Middle Initial A. Family Name FARBER Country of Citizenship  
(State/Foreign Country) CA  
Post Office Address (Include Zip Code) 2201 Canillo Rd. Ojai, CA 93023

2. INVENTOR'S SIGNATURE: Ronald D. Lachman Date 6/13/95  
Inventor's Name (typed) Ronald D. LACHMAN U.S.A.  
Residence (City) Northbrook First Middle Initial D. Family Name LACHMAN Country of Citizenship  
(State/Foreign Country) IL  
Post Office Address (Include Zip Code) 3140 Whisperwoods Court, Northbrook, IL 60062

3. INVENTOR'S SIGNATURE: \_\_\_\_\_ Date \_\_\_\_\_  
Inventor's Name (typed) \_\_\_\_\_  
Residence (City) \_\_\_\_\_ First Middle Initial \_\_\_\_\_ Family Name \_\_\_\_\_ Country of Citizenship  
(State/Foreign Country) \_\_\_\_\_  
Post Office Address (Include Zip Code) \_\_\_\_\_

OR ADDITIONAL INVENTORS, check box [ ] and attach sheet (CDC-116.2) for same information for each re signature, name, date, citizenship, residence and address.)

Inventor(s): David A. Farber and Ronald D. Lachman (Atty. Dkt.  
Appln. No.: 0 8 /425,160 or Patent No.: ( 213987 /  
Filed: April 11, 1995 or Issued: \_\_\_\_\_ M# / Client Ref.  
Title: IDENTIFYING DATA IN A DATA PROCESSING SYSTEM

VERIFIED STATEMENT (DECLARATION) CLAIMING SMALL ENTITY  
STATUS (37 CFR 1.9(d) and 1.27(c)) - **SMALL BUSINESS CONCERN**

I hereby declare that I am

- the owner of the small business concern identified below:  
 an official of the small business concern empowered to act on behalf of the concern identified below:

NAME OF CONCERN KINETECH INC.  
ADDRESS OF CONCERN 292N Carillo Rd., Ojai, California 93023 DE 6/9/95  
3140 Whisperwoods Ct., Northbrook Illinois 60062 6/11

**I hereby declare** that the above identified small business concern qualifies as a small business concern as defined in 13 CFR 121.12, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third party or parties controls or has the power to control both.

**I hereby declare** that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention entitled: IDENTIFYING DATA IN A DATA PROCESSING SYSTEM

by inventor(s) David A. FARBER and Ronald D. LACHMAN  
described in

->  the Specification filed herewith,  
 ->  Application No. 0 8 /425,160, filed April 11, 1995  
 ->  Patent No. \_\_\_\_\_, issued \_\_\_\_\_

If the rights held by the above identified small business concern are not exclusive, each small entity individual, concern or organization having rights to the invention is listed in (A) and (B) below and no rights to the invention are held by any person, other than the inventor, who could not qualify under 37 CFR 1.9(c) as an independent inventor if that person had made the invention, or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

(A) FULL NAME of assignee/licensee/grantee/conveyee\* \_\_\_\_\_

ADDRESS \_\_\_\_\_

X proper box:  INDIVIDUAL  SMALL BUSINESS CONCERN  NONPROFIT ORGANIZATION

(B) FULL NAME of assignee/licensee/grantee/conveyee\* \_\_\_\_\_

ADDRESS \_\_\_\_\_

X proper box:  INDIVIDUAL  SMALL BUSINESS CONCERN  NONPROFIT ORGANIZATION

\*NOTE: Separate verified statement is required from each person, concern or organization named in (A) and (B) above having rights to the invention, averring to his/her/its status as a small entity. (37 CFR 1.27)

I acknowledge the duty to file, in this case, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b))

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

NAME OF PERSON SIGNING Ronald Lachman  
T OF PERSON OTHER THAN OWNER \_\_\_\_\_  
A. SS OF PERSON SIGNING 3140 WHISPERWOODS CT  
NORTHBROOK IL 60062

SIGNATURE [Signature] DATE 6-14-95

PRINT OF DRAWINGS  
AS ORIGINALLY FILE

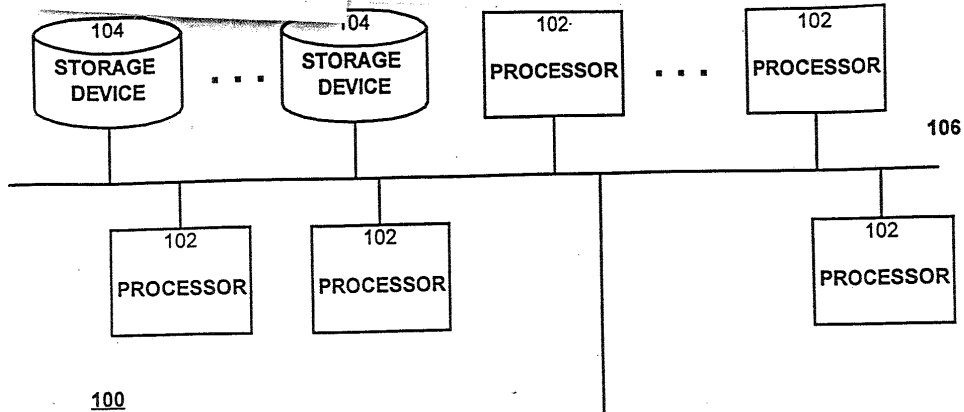
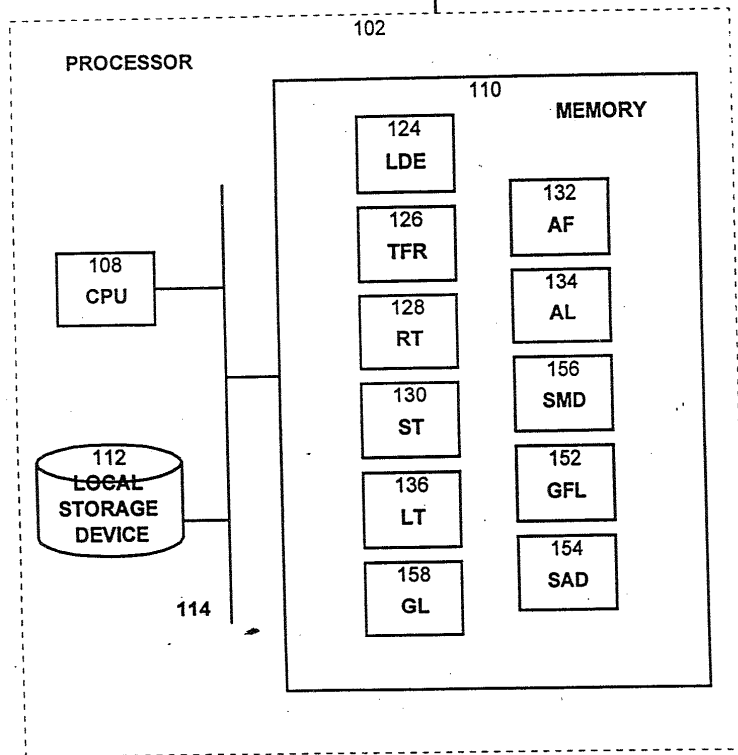
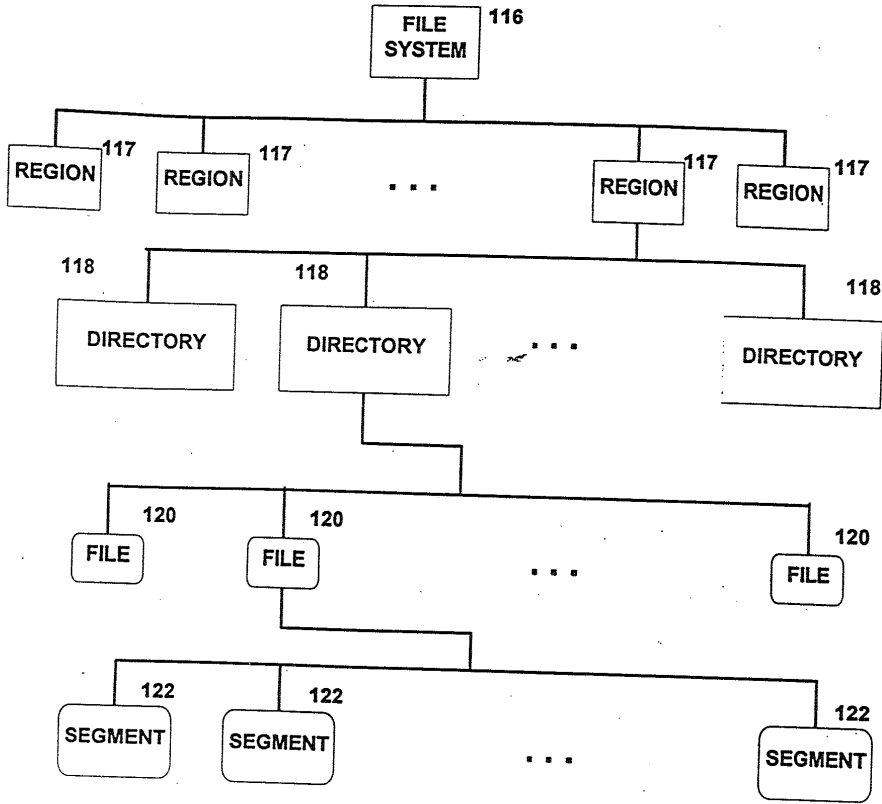


FIG. 1



667099-091E260

FIG. 2



097470-097660





PRINT OF DRAWINGS  
AS ORIGINALLY FILED

source ID	144
source type	
source rights	
source availability	
source location	

FIG. 6

Original Name	146
Operation	
Type	
Processor ID	
Timestamp	
Pathname	
True Name	

FIG. 7

date of entry	148
type of entry	
True Name	

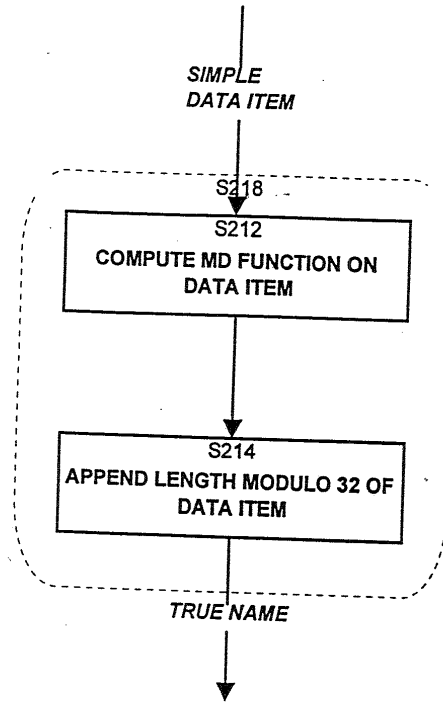
FIG. 8

True Name	150
licensee	

FIG. 9

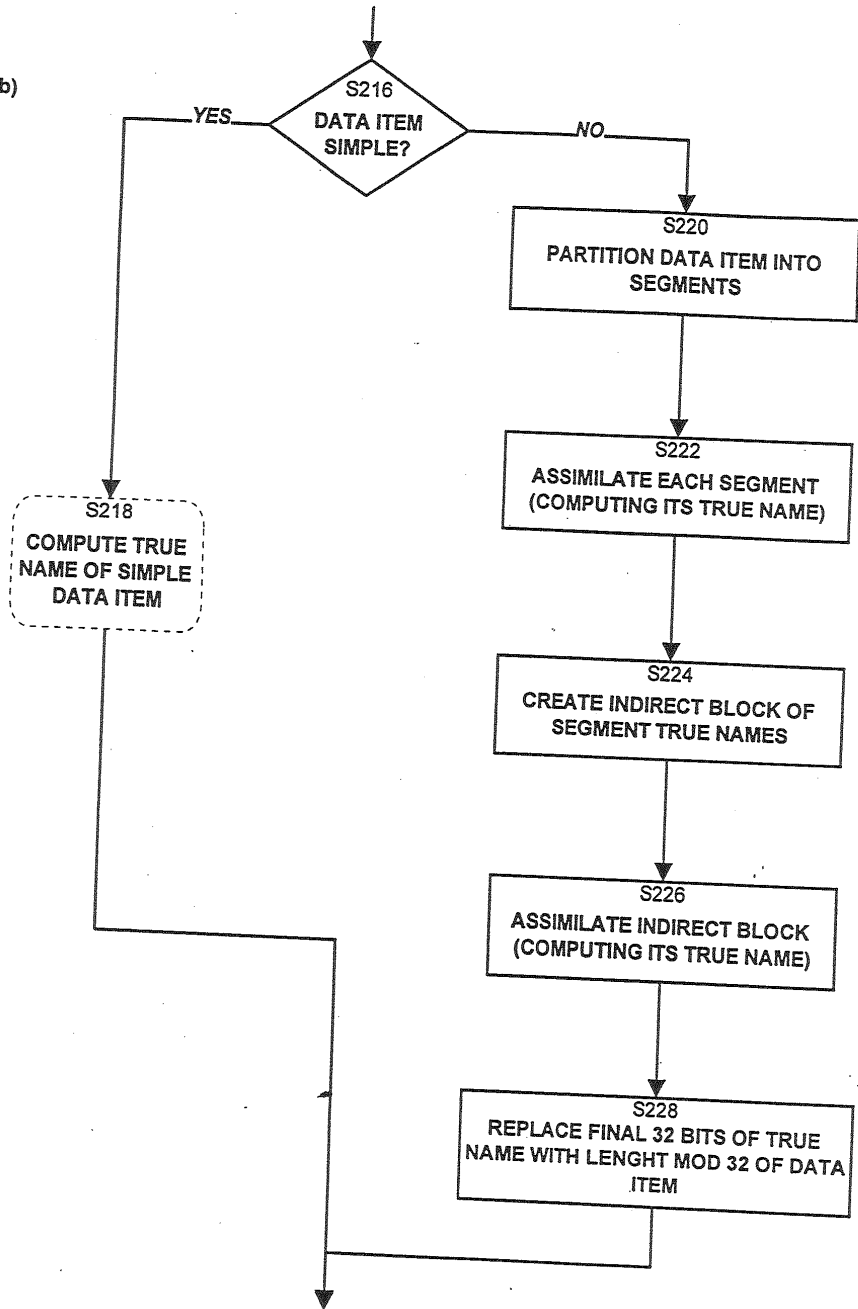
001010-0916260

FIG. 10(a)



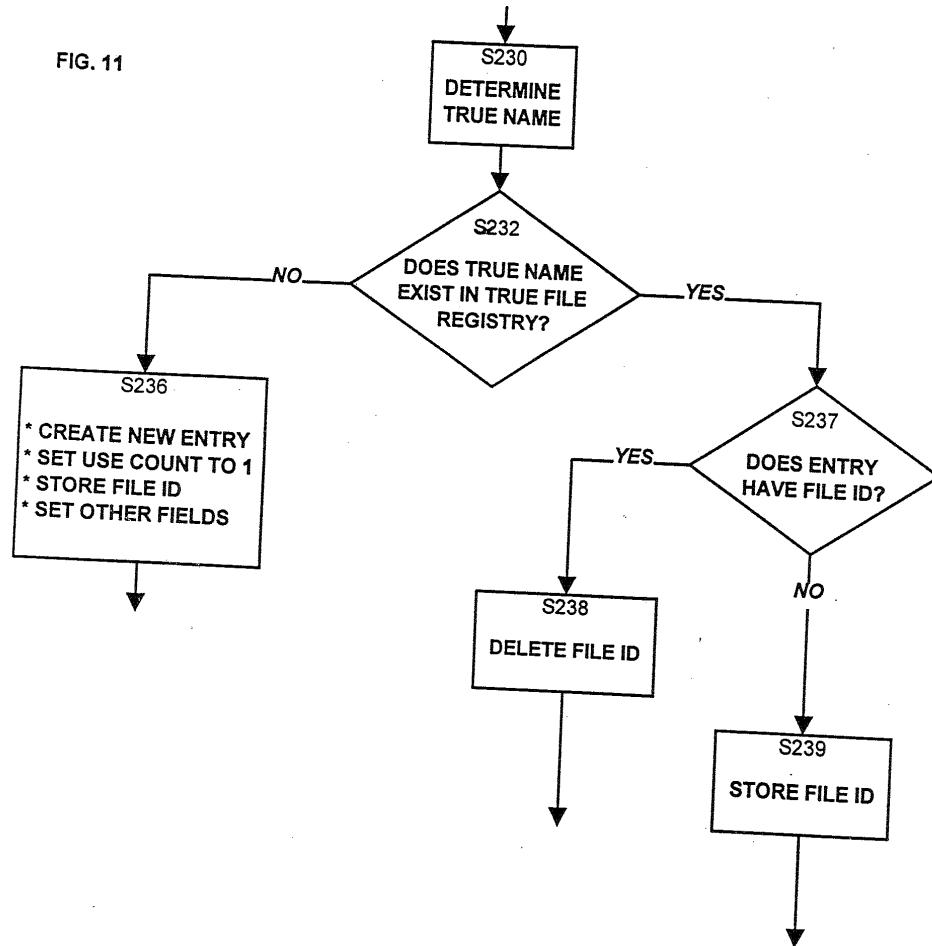
067010-097E260

FIG. 10(b)



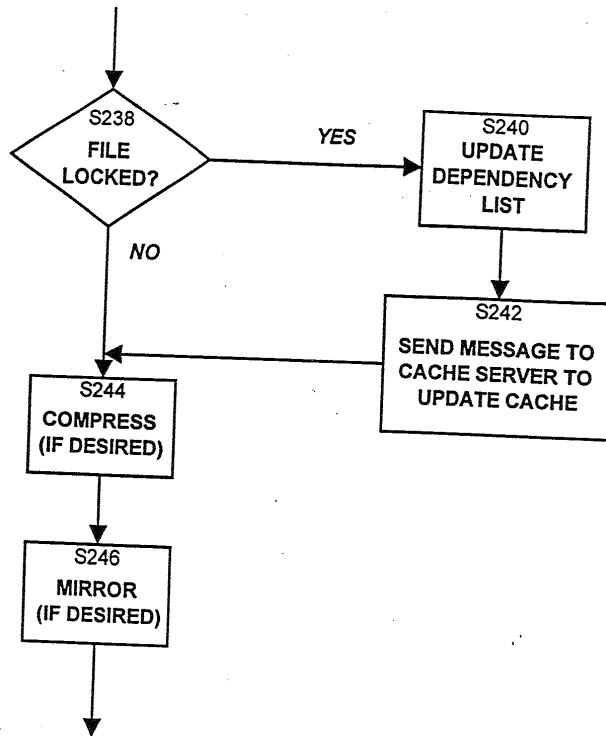
061010 097E8260

FIG. 11



001010 097E260

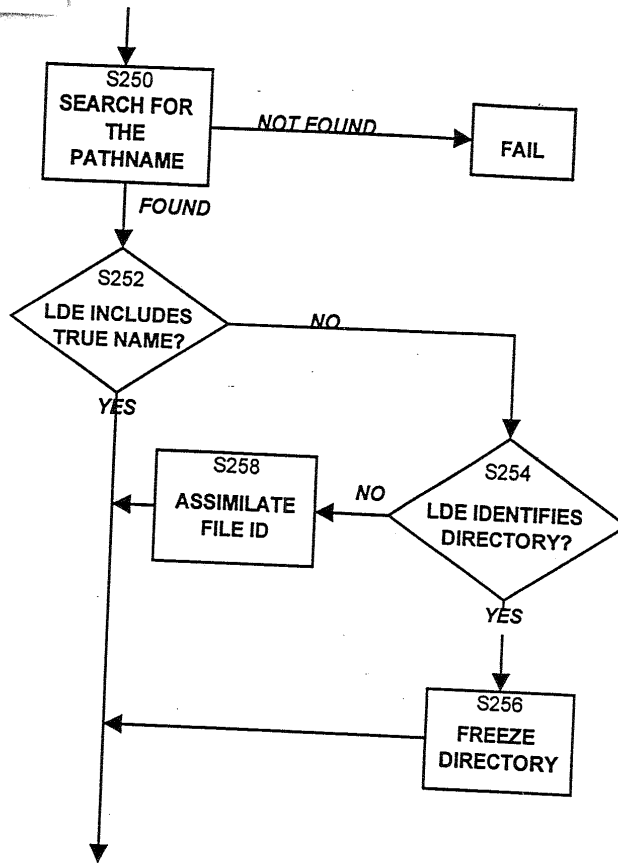
FIG. 12



667010-09E260

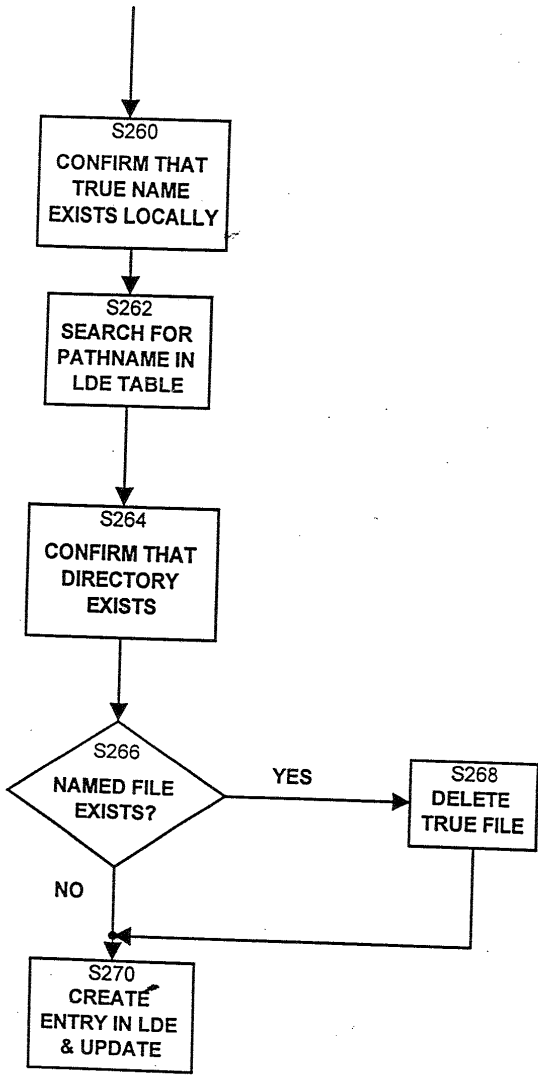
PRINT OF DRAWINGS  
AS ORIGINALLY FILED

FIG. 13



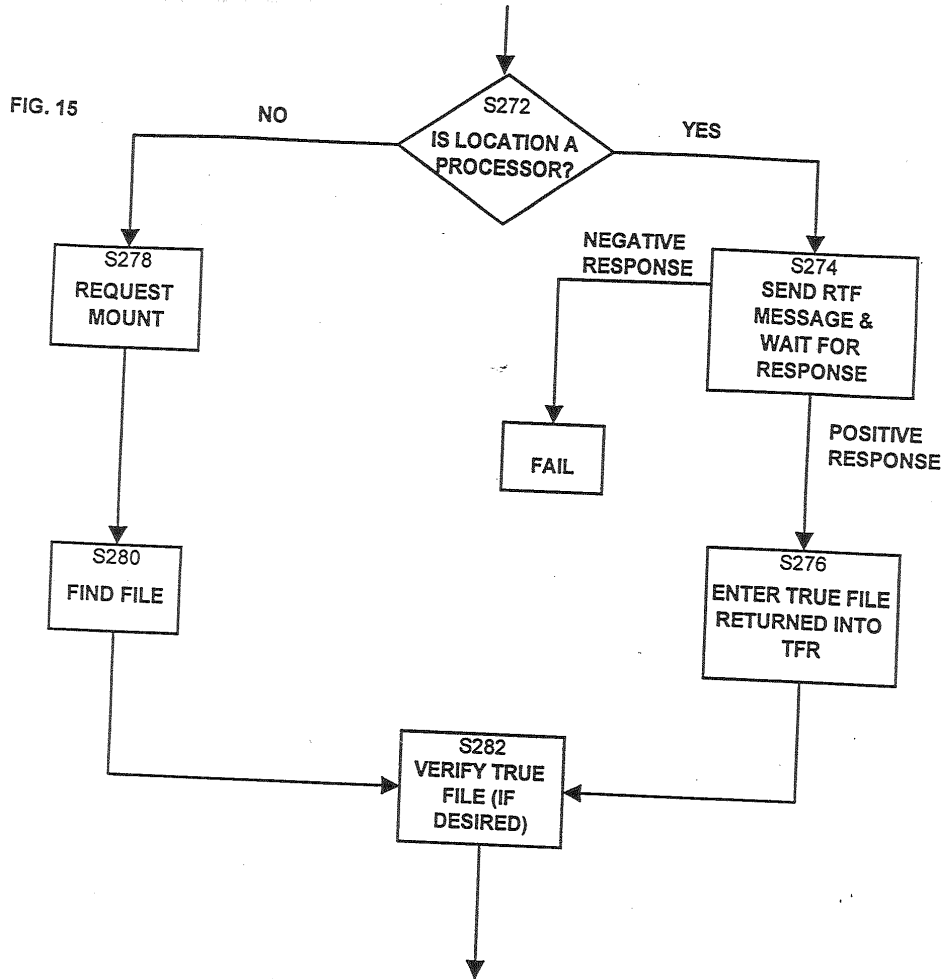
66700-0948200

FIG. 14



607070 0972220

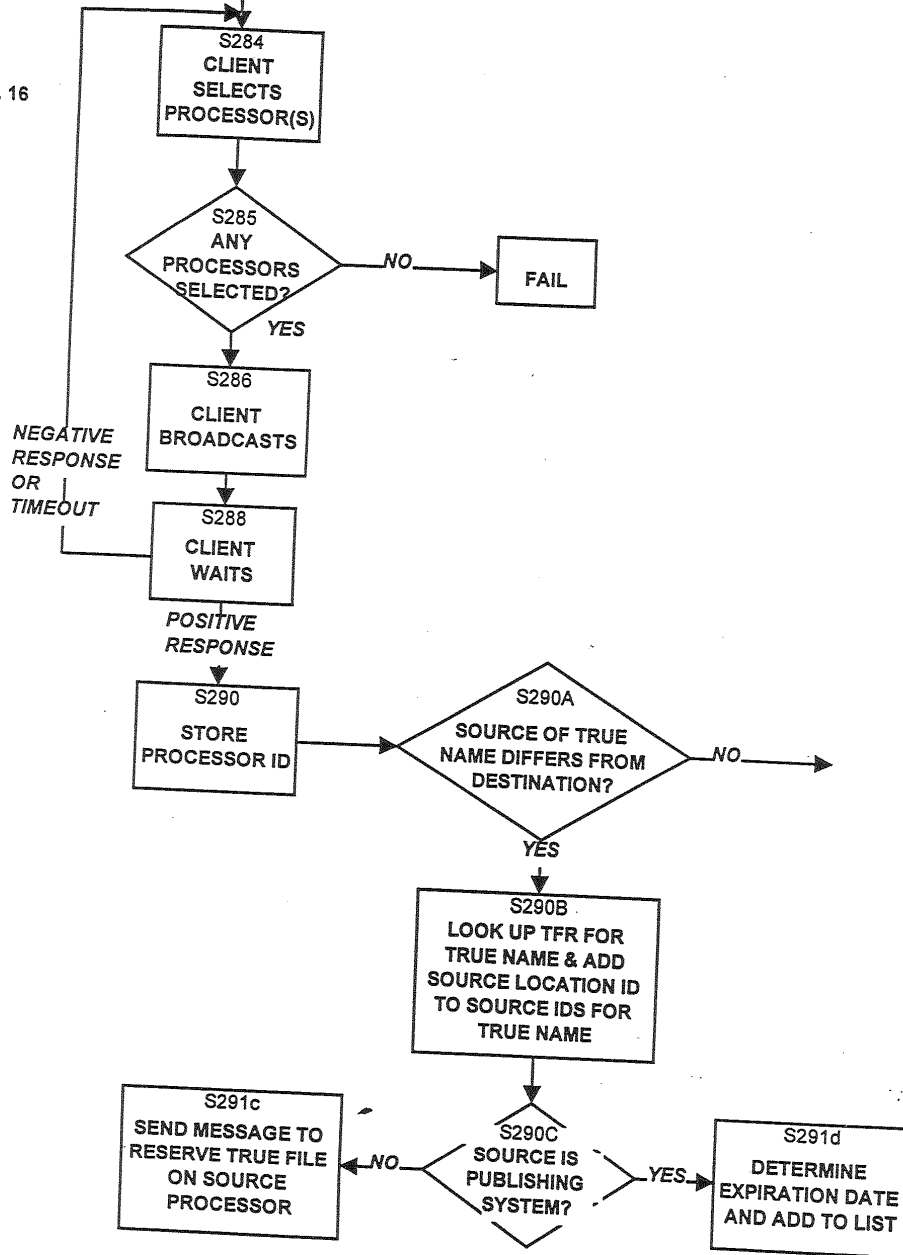




667010-0972200

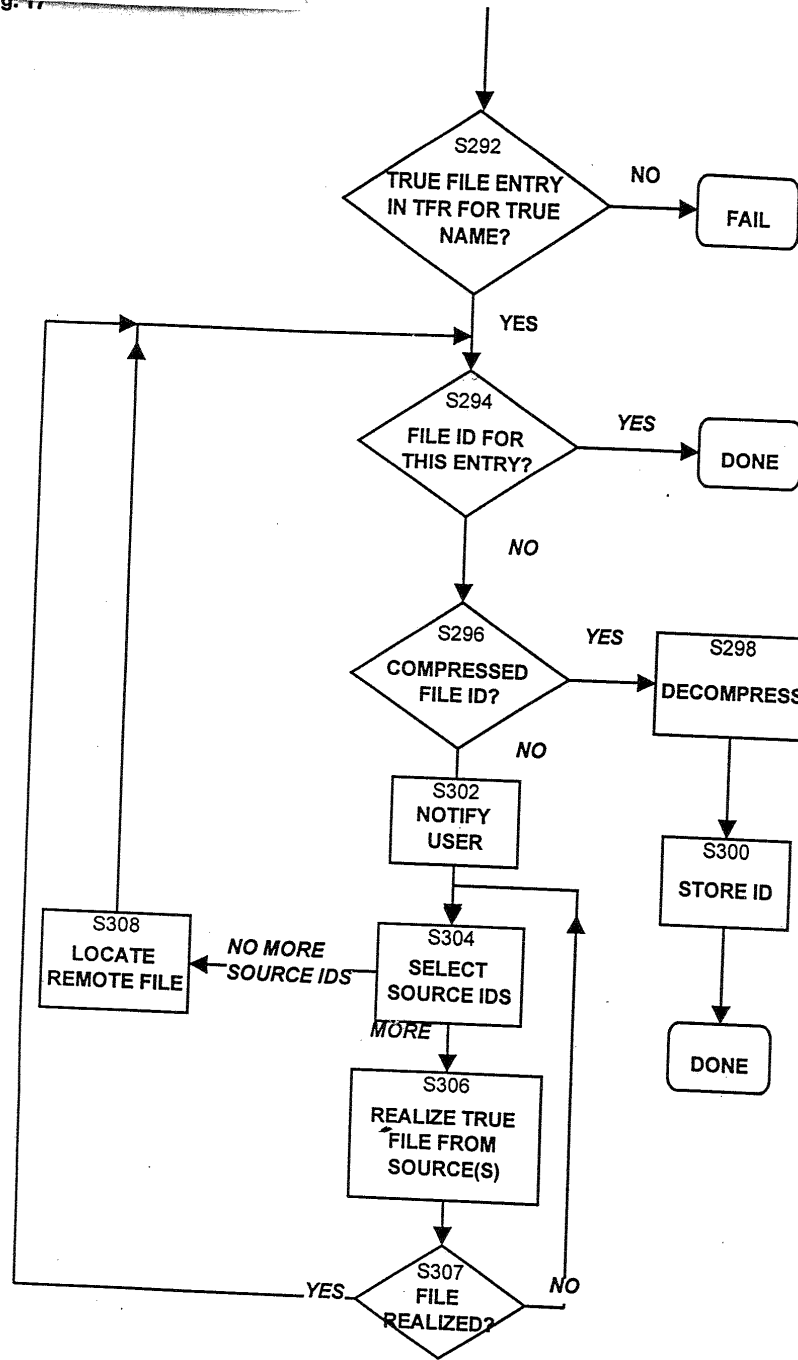
PRINT OF DRAWINGS  
AS ORIGINALLY FILED

FIG. 16



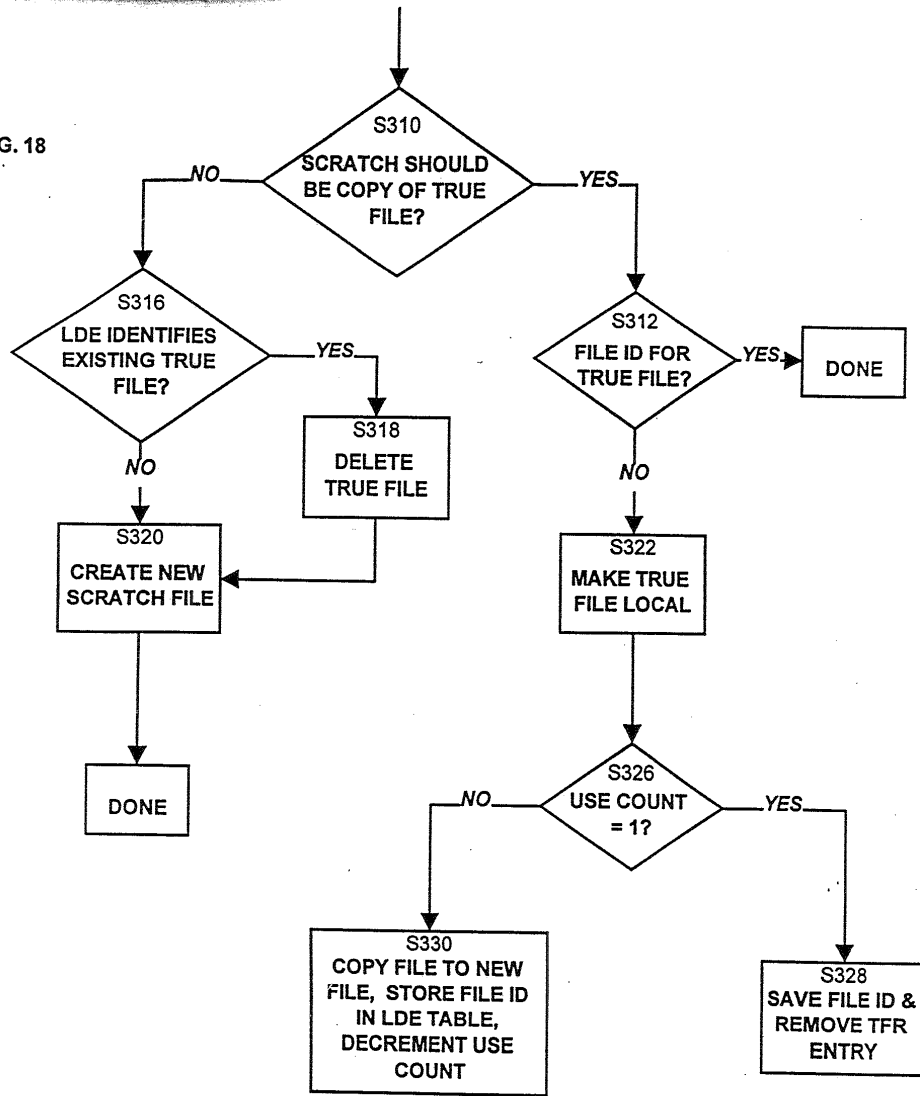
657010-0972260

Fig. 17



667010 0972260

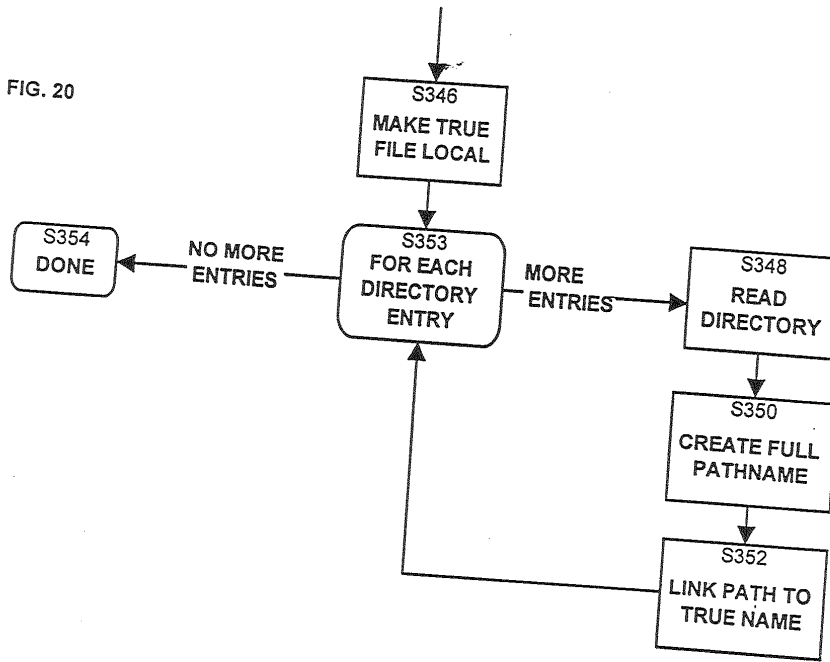
FIG. 18



6070707 0972220

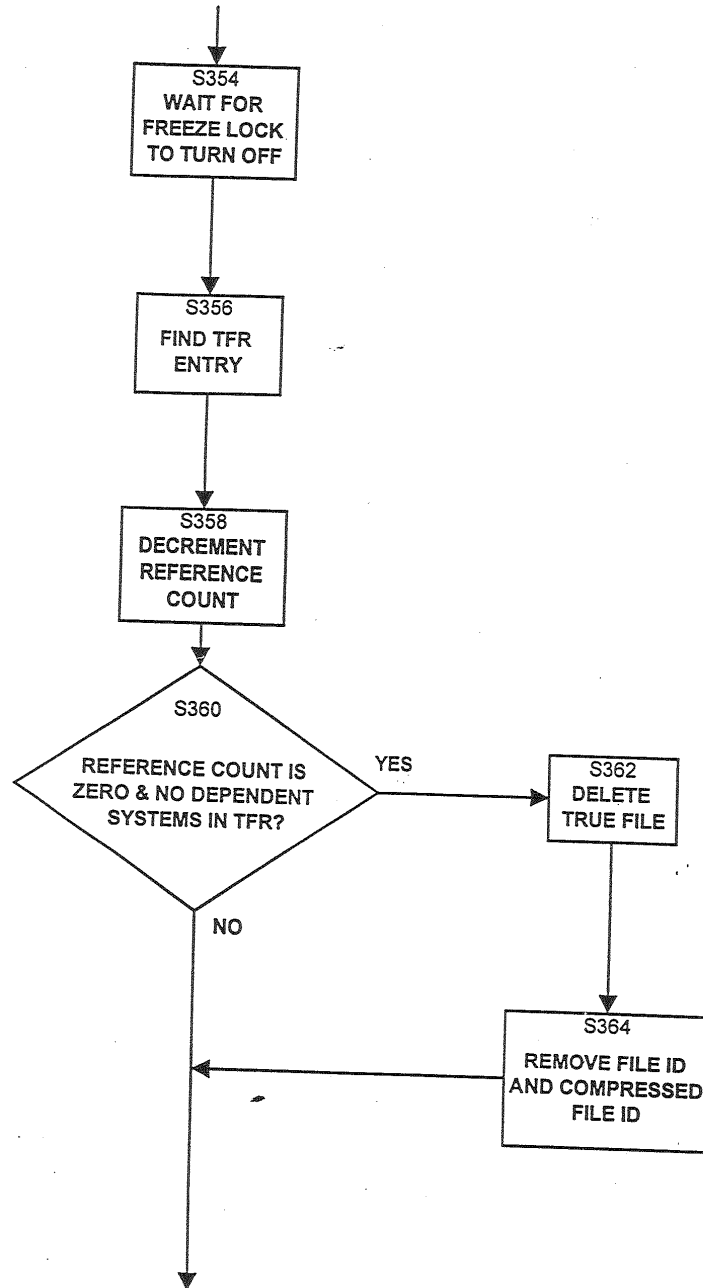


FIG. 20



667010-09E860

FIG. 21

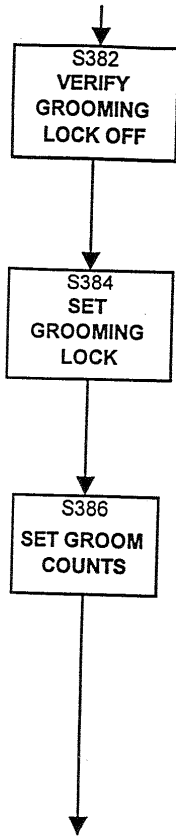


0923150 0478  
007070 0512260





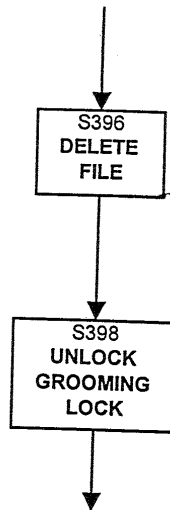
FIG. 23



66740-01E220

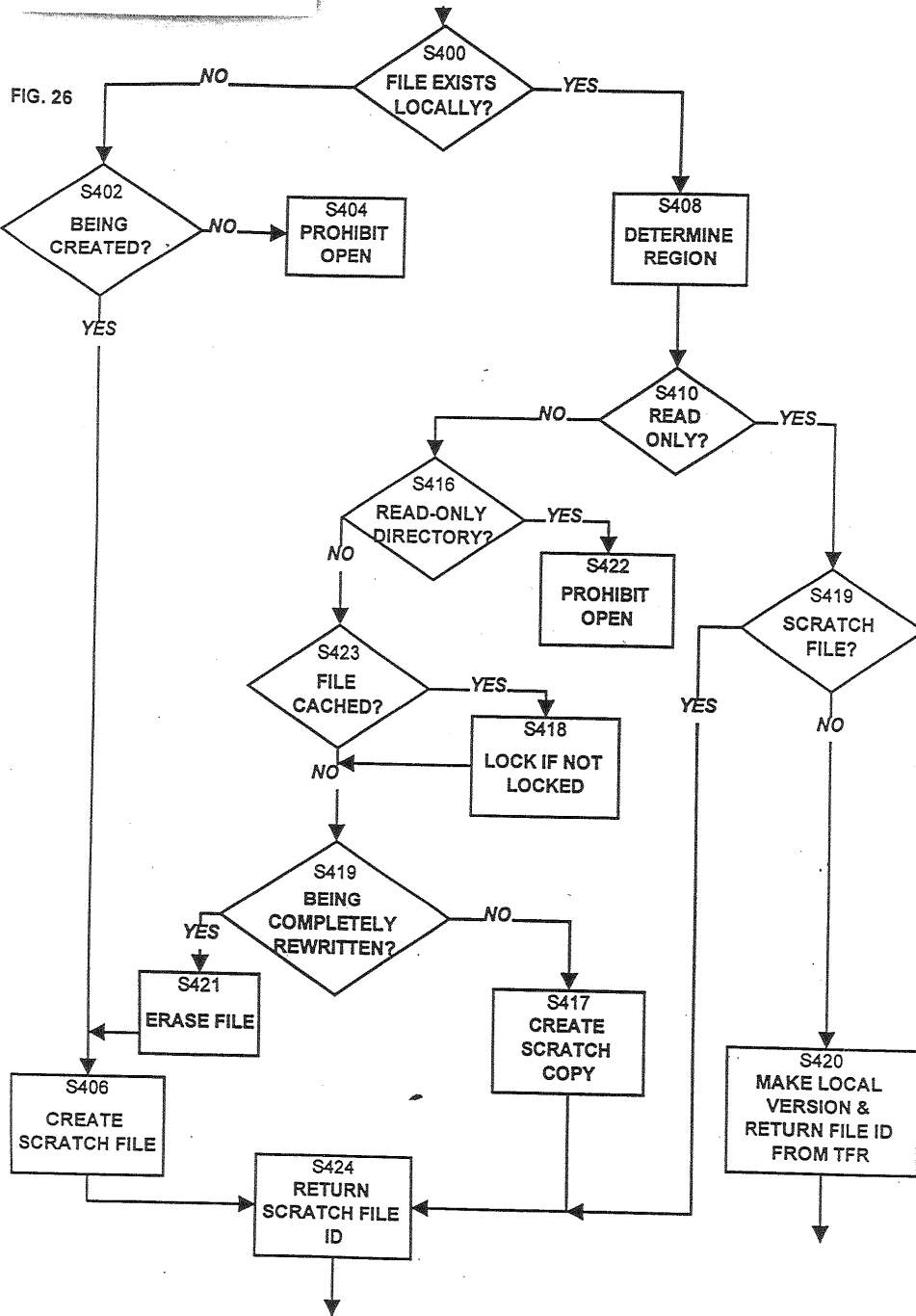


FIG. 25



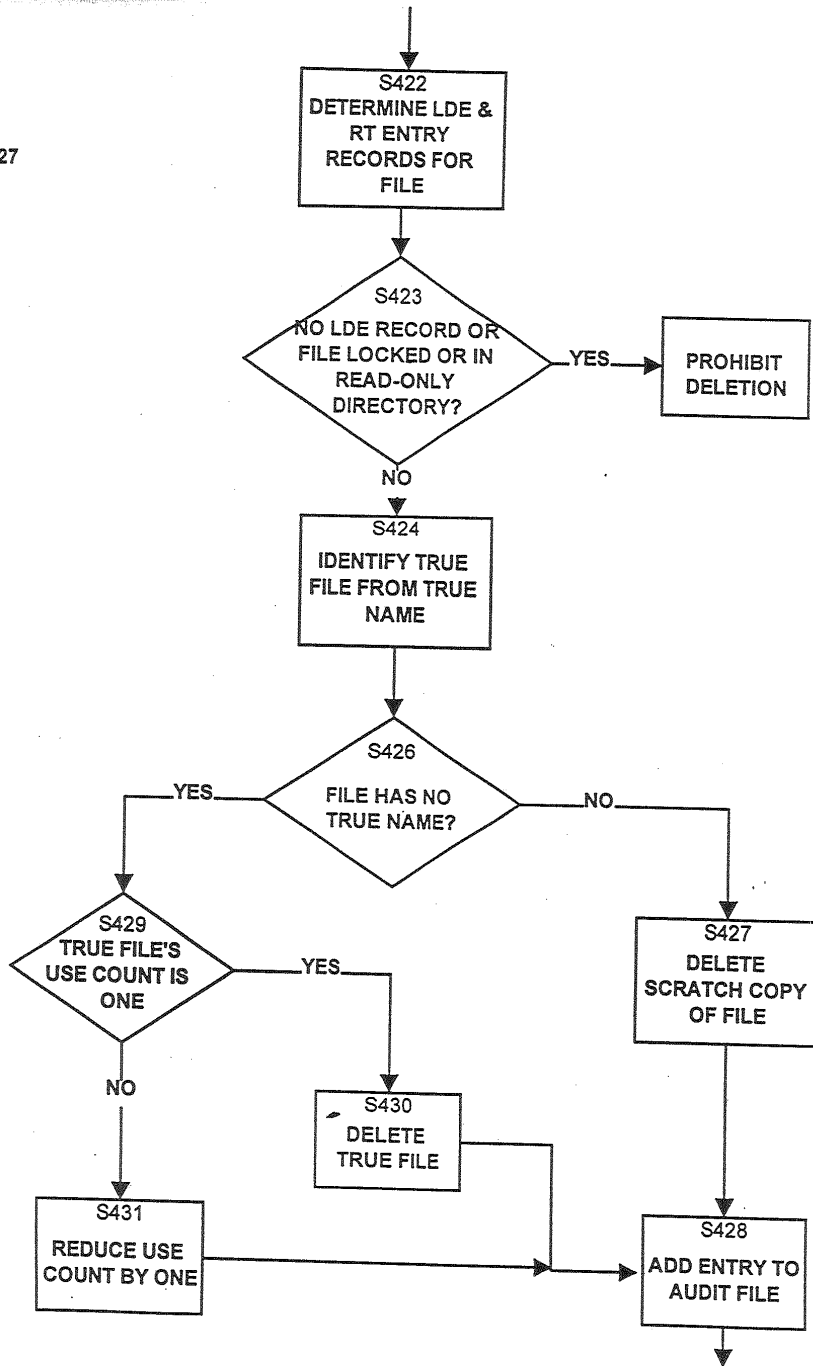
016F010" 091E0200

PRINT OF DRAWINGS  
AS ORIGINALLY FILE



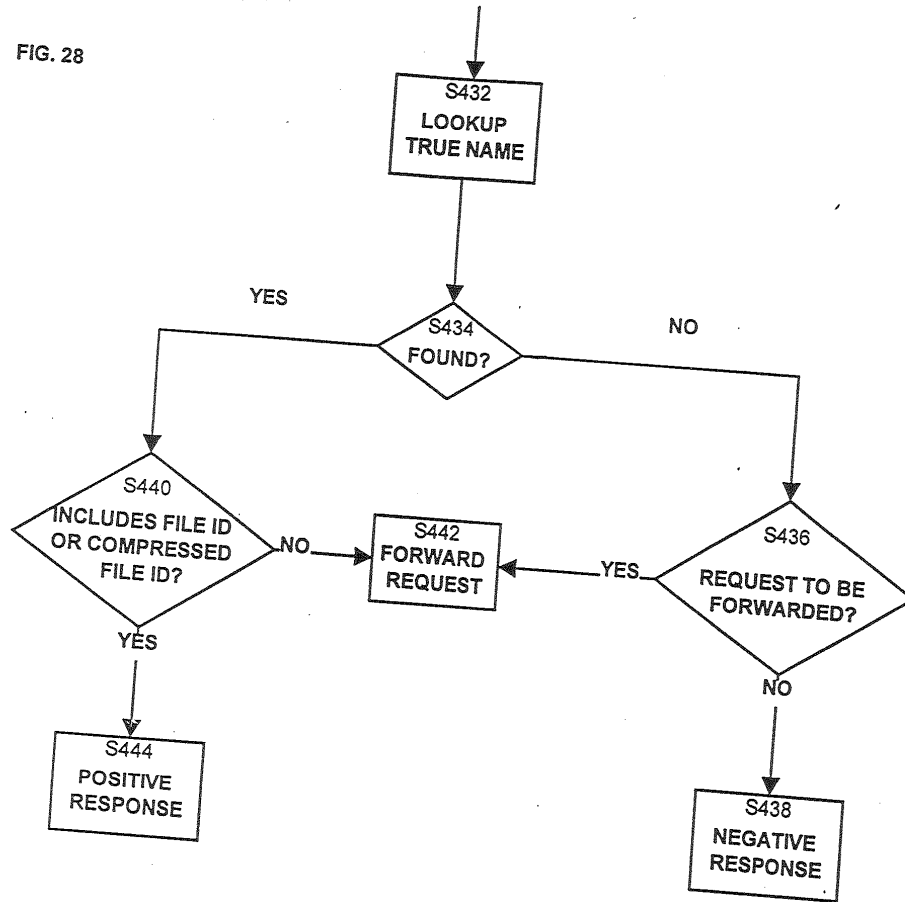
667410 09 FEB 80

FIG. 27



667010-0572260

FIG. 28



607070 097E8E00