US008639267B2

(12) **United States Patent**
Johnson

(10) **Patent No.: US 8,639,267 B2**
(45) **Date of Patent: Jan. 28, 2014**

(54) **SYSTEM AND METHOD FOR LOCATION BASED EXCHANGES OF DATA FACILITATING DISTRIBUTED LOCATIONAL APPLICATIONS**

(76) Inventor: **William J. Johnson**, Flower Mound, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 971 days.

(21) Appl. No.: **12/287,064**

(22) Filed: **Oct. 3, 2008**

(65) **Prior Publication Data**

US 2009/0233623 A1 Sep. 17, 2009

Related U.S. Application Data

(63) Continuation-in-part of application No. 12/077,041, filed on Mar. 14, 2008.

(51) **Int. Cl.**
H04W 24/00 (2009.01)

(52) **U.S. Cl.**
USPC **455/456.3**; 455/404.2; 455/414.2; 370/331

(58) **Field of Classification Search**
USPC 455/404.2, 456.3; 370/331
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,636,421	A	3/1972	Barker et al.
4,021,780	A	5/1977	Narey et al.
4,255,619	A	3/1981	Saito
4,445,118	A	4/1984	Taylor et al.
4,536,647	A	8/1985	Atalla et al.
4,644,351	A	2/1987	Zabarsky et al.
4,757,267	A	7/1988	Riskin
4,841,560	A	6/1989	Chan et al.

(Continued)

FOREIGN PATENT DOCUMENTS

EP	0712227	5/1996
EP	915590	5/1999

(Continued)

OTHER PUBLICATIONS

Bill N. Schilit and Marvin M. Theimer, disseminatwing Active Map Information Mobile Hosts, IEEE Network, Sep./Oct. 1994.

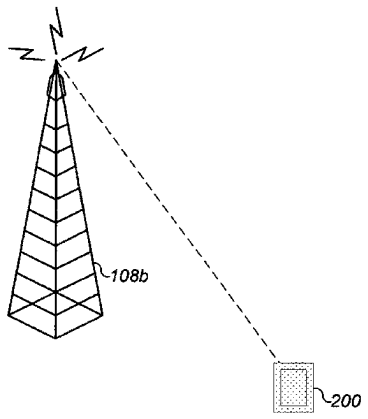
(Continued)

Primary Examiner — Liton Miah
(74) *Attorney, Agent, or Firm* — Yudell Isidore Ng Russell PLLC

(57) **ABSTRACT**

Provided is a distributed system and method for enabling new and useful location dependent features and functionality to mobile data processing systems. Mobile data processing systems (MSs) interact with each other as peers in communications and interoperability. Indirectly located mobile data processing systems are located relative other mobile data processing systems, and are automatically located using whereabouts data of directly located mobile data processing systems and/or whereabouts data of other indirectly located mobile data processing systems. A mobile data processing system may dynamically take on roles of being directly located or indirectly located, depending on the environment and capabilities available at a particular time. Data is shared between mobile data processing systems to carry out novel Location Based eXchanges (LBX) of data for new mobile applications. Information which is transmitted inbound to, transmitted outbound from, or is in process at, a mobile data processing system, is used to trigger processing of actions in accordance with user configured permissions, charters, and other configurations. In a preferred embodiment, a user configurable platform is provided for quickly building well behaving LBX applications at MSs and across a plurality of interoperating MSs.

56 Claims, 259 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

4,845,504 A 7/1989 Roberts et al.
4,922,516 A 5/1990 Butler et al.
4,973,952 A 11/1990 Malec et al.
4,974,170 A 11/1990 Bouve et al.
4,977,399 A 12/1990 Price et al.
5,089,814 A 2/1992 DeLuca et al.
5,095,532 A 3/1992 Mardus
5,121,126 A 6/1992 Clagett
5,122,795 A 6/1992 Cubley et al.
5,131,020 A 7/1992 Liebesny et al.
5,185,857 A 2/1993 Rozmanith et al.
5,195,031 A 3/1993 Ordish
5,214,793 A 5/1993 Conway et al.
5,223,844 A 6/1993 Mansell et al.
5,243,652 A 9/1993 Teare et al.
5,245,608 A 9/1993 Deaton et al.
5,264,822 A 11/1993 Vogelman et al.
5,265,070 A 11/1993 Minowa
5,303,393 A 4/1994 Noreen et al.
5,321,242 A 6/1994 Heath, Jr.
5,337,044 A 8/1994 Folger et al.
5,347,632 A 9/1994 Filepp et al.
5,363,245 A 11/1994 Borello
5,363,377 A 11/1994 Sharpe
5,365,516 A 11/1994 Jandrell
5,371,794 A 12/1994 Diffie et al.
5,390,237 A 2/1995 Hoffman et al.
5,404,505 A 4/1995 Levinson
5,432,841 A 7/1995 Rimer
5,444,444 A 8/1995 Ross
5,451,757 A 9/1995 Heath, Jr.
5,455,807 A 10/1995 Nepple
5,461,627 A 10/1995 Rypinski
5,469,362 A 11/1995 Hunt et al.
5,475,735 A 12/1995 Williams et al.
5,485,163 A 1/1996 Singer et al.
5,487,103 A 1/1996 Richardson et al.
5,493,309 A 2/1996 Bjornholt et al.
5,497,414 A 3/1996 Bartholomew
5,504,482 A 4/1996 Schreder
5,511,111 A 4/1996 Serbetcioglu et al.
5,511,233 A 4/1996 Otten
5,512,908 A 4/1996 Herrick
5,513,263 A 4/1996 White et al.
5,528,248 A 6/1996 Steiner et al.
5,539,395 A 7/1996 Buss et al.
5,544,354 A 8/1996 May et al.
5,559,520 A 9/1996 Barzegar et al.
5,561,704 A 10/1996 Salimando
5,566,235 A 10/1996 Hetz
5,581,479 A 12/1996 McLaughlin
5,583,864 A 12/1996 Lightfoot et al.
5,586,254 A 12/1996 Kondo et al.
5,588,042 A 12/1996 Comer
5,590,196 A 12/1996 Moreau
5,590,398 A 12/1996 Matthews
5,592,470 A 1/1997 Rudrapatna et al.
5,594,779 A 1/1997 Goodman
5,596,625 A 1/1997 LeBlanc
5,602,843 A 2/1997 Gray
5,608,854 A 3/1997 Labedz et al.
5,610,973 A 3/1997 Comer
5,625,364 A 4/1997 Herrick et al.
5,625,668 A 4/1997 Loomis
5,627,549 A 5/1997 Park
5,636,245 A 6/1997 Ernst et al.
5,646,632 A 7/1997 Khan et al.
5,654,959 A 8/1997 Baker et al.
5,657,375 A 8/1997 Connolly et al.
5,661,492 A 8/1997 Shoap et al.
5,663,734 A 9/1997 Krasner
5,664,948 A 9/1997 Dimitriadis et al.
5,666,481 A 9/1997 Lewis
5,677,905 A 10/1997 Bigham
5,687,212 A 11/1997 Kinser, Jr. et al.
5,689,431 A 11/1997 Rudow et al.
5,694,453 A 12/1997 Fuller et al.
5,701,301 A 12/1997 Weisser, Jr.
5,704,049 A 12/1997 Briechle
5,712,899 A 1/1998 Pace, II
5,713,075 A 1/1998 Threadgill et al.
5,714,948 A 2/1998 Farmakis et al.
5,717,688 A 2/1998 Belanger et al.
5,720,033 A 2/1998 Deo
5,724,521 A 3/1998 Dedrick
5,727,057 A 3/1998 Emery et al.
5,729,680 A 3/1998 Belanger et al.
5,758,049 A 5/1998 Johnson et al.
5,771,283 A 6/1998 Chang et al.
5,774,534 A 6/1998 Mayer
5,778,304 A 7/1998 Grube et al.
5,790,974 A 8/1998 Tognazzini
5,794,210 A 8/1998 Goldhaber et al.
5,796,727 A 8/1998 Harrison et al.
5,798,733 A 8/1998 Ethridge
5,806,018 A 9/1998 Smith et al.
5,812,763 A 9/1998 Teng
5,819,155 A 10/1998 Worthey et al.
5,826,195 A 10/1998 Westerlage et al.
5,835,061 A 11/1998 Stewart
5,838,774 A 11/1998 Weisser, Jr.
5,842,010 A 11/1998 Jain et al.
5,845,211 A 12/1998 Roach
5,852,775 A 12/1998 Hidary
5,855,007 A 12/1998 Jovicic et al.
5,870,555 A 2/1999 Pruett et al.
5,870,724 A 2/1999 Lawlor et al.
5,875,186 A 2/1999 Belanger et al.
5,875,401 A 2/1999 Rochkind
5,878,126 A 3/1999 Velamuri et al.
5,880,958 A 3/1999 Helms et al.
5,881,131 A 3/1999 Farris et al.
5,884,284 A 3/1999 Peters et al.
5,887,259 A 3/1999 Zicker et al.
5,889,953 A 3/1999 Thebaut et al.
5,892,454 A 4/1999 Schipper et al.
5,896,440 A 4/1999 Reed et al.
5,897,640 A 4/1999 Veghte et al.
5,903,636 A 5/1999 Malik
5,907,544 A 5/1999 Rypinski
5,920,846 A 7/1999 Storch et al.
5,922,040 A 7/1999 Prabhakaran
5,923,702 A 7/1999 Brenner et al.
5,933,420 A 8/1999 Jaszewski et al.
5,938,721 A 8/1999 Dussell et al.
5,949,867 A 9/1999 Sonnenberg
5,950,130 A 9/1999 Coursey
5,961,593 A 10/1999 Gabber et al.
5,963,866 A 10/1999 Palamara et al.
5,963,913 A 10/1999 Henneuse et al.
5,968,176 A 10/1999 Nessett et al.
5,969,678 A 10/1999 Stewart
5,982,867 A 11/1999 Urban et al.
5,983,091 A 11/1999 Rodriguez
5,987,381 A 11/1999 Oshizawa
5,991,287 A 11/1999 Diepstraten et al.
5,995,015 A 11/1999 DeTemple et al.
6,006,090 A 12/1999 Coleman et al.
6,009,398 A 12/1999 Mueller et al.
6,011,975 A 1/2000 Emery et al.
6,018,293 A 1/2000 Smith et al.
6,026,151 A 2/2000 Bauer et al.
6,028,921 A 2/2000 Malik et al.
6,047,327 A 4/2000 Tso et al.
6,055,637 A 4/2000 Hudson et al.
6,058,106 A 5/2000 Cudak et al.
6,067,082 A 5/2000 Enmei
6,067,297 A 5/2000 Beach
6,073,062 A 6/2000 Hoshino et al.
6,076,080 A 6/2000 Morscheck et al.
6,085,086 A 7/2000 La Porta et al.
6,091,956 A 7/2000 Hollenberg
6,101,381 A 8/2000 Tajima et al.
6,101,443 A 8/2000 Kato et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,112,186 A 8/2000 Bergh et al.
6,115,669 A 9/2000 Watanabe et al.
6,122,520 A 9/2000 Want et al.
6,133,853 A 10/2000 Obradovich et al.
6,138,003 A 10/2000 Kingdon et al.
6,138,119 A 10/2000 Hall et al.
6,141,609 A 10/2000 Herdeg et al.
6,144,645 A 11/2000 Struhsaker et al.
6,154,152 A 11/2000 Ito
6,154,637 A 11/2000 Wright et al.
6,157,829 A 12/2000 Grube et al.
6,157,946 A 12/2000 Itakura et al.
6,163,274 A 12/2000 Lindgren
6,167,255 A 12/2000 Kennedy, III et al.
6,182,226 B1 1/2001 Reid et al.
6,184,829 B1 2/2001 Stilp
6,185,426 B1 2/2001 Alperovich et al.
6,185,484 B1 2/2001 Rhinehart
6,192,314 B1 2/2001 Khavakh et al.
6,202,054 B1 3/2001 Lawlor et al.
6,205,478 B1 3/2001 Sugano et al.
6,208,854 B1 3/2001 Roberts et al.
6,208,866 B1 3/2001 Rouhollahzadeh et al.
6,226,277 B1 5/2001 Chuah
6,229,477 B1 5/2001 Chang et al.
6,229,810 B1 5/2001 Gerszberg et al.
6,233,329 B1 5/2001 Urban et al.
6,233,452 B1 5/2001 Nishino
6,236,360 B1 5/2001 Rudow et al.
6,236,365 B1 5/2001 LeBlanc et al.
6,236,940 B1 5/2001 Rudow et al.
6,246,361 B1 6/2001 Weill et al.
6,246,948 B1 6/2001 Thakker
6,252,544 B1 6/2001 Hoffberg
6,259,405 B1 7/2001 Stewart et al.
6,263,209 B1 7/2001 Reed et al.
6,266,615 B1 7/2001 Jin
6,278,938 B1 8/2001 Alumbaugh
6,285,665 B1 9/2001 Chuah et al.
6,285,931 B1 9/2001 Hattori et al.
6,298,234 B1 10/2001 Brunner
6,308,273 B1 10/2001 Goertzel et al.
6,311,069 B1 10/2001 Havinis et al.
6,317,718 B1 11/2001 Fano
6,321,092 B1 11/2001 Fitch et al.
6,324,396 B1 11/2001 Vasa et al.
6,326,918 B1 12/2001 Stewart
6,327,254 B1 12/2001 Chuah
6,327,357 B1 12/2001 Meek et al.
6,332,127 B1 12/2001 Bandera et al.
6,332,163 B1 12/2001 Bowman-Amuah
6,340,958 B1 1/2002 Cantu et al.
6,343,290 B1 1/2002 Cossins et al.
6,345,288 B1 2/2002 Reed et al.
6,353,664 B1 3/2002 Cannon et al.
6,359,880 B1 3/2002 Curry et al.
6,360,101 B1 3/2002 Irvin
6,366,561 B1 4/2002 Bender
6,377,548 B1 4/2002 Chuah et al.
6,377,810 B1 4/2002 Geiger et al.
6,377,982 B1 4/2002 Rai et al.
6,385,531 B2 5/2002 Bates et al.
6,385,591 B1 5/2002 Mankoff
6,389,426 B1 5/2002 Turnbull et al.
6,393,482 B1 5/2002 Rai et al.
6,400,722 B1 6/2002 Chuah et al.
6,405,123 B1 6/2002 Rennard et al.
6,407,673 B1 6/2002 Lane
6,408,307 B1 6/2002 Semple et al.
6,414,635 B1 7/2002 Stewart et al.
6,414,950 B1 7/2002 Rai et al.
6,415,019 B1 7/2002 Savaglio et al.
6,418,308 B1 7/2002 Heinonen et al.
6,421,441 B1 7/2002 Dzuban
6,421,714 B1 7/2002 Rai et al.
6,427,073 B1 7/2002 Kortesalmi et al.
6,427,119 B1 7/2002 Stefan et al.
6,430,276 B1 8/2002 Bouvier et al.
6,430,562 B1 8/2002 Kardos et al.
6,442,391 B1 8/2002 Johansson et al.
6,442,479 B1 8/2002 Barton
6,442,687 B1 8/2002 Savage
6,449,272 B1 9/2002 Chuah et al.
6,449,497 B1 9/2002 Kirbas et al.
6,452,498 B2 9/2002 Stewart
6,456,234 B1 9/2002 Johnson
6,463,533 B1 10/2002 Calamera et al.
6,470,378 B1 10/2002 Tracton et al.
6,470,447 B1 10/2002 Lambert et al.
6,473,626 B1 10/2002 Nevoux et al.
6,477,382 B1 11/2002 Mansfield et al.
6,477,526 B2 11/2002 Hayashi et al.
6,484,029 B2 11/2002 Hughes et al.
6,484,092 B2 11/2002 Seibel
6,484,148 B1 11/2002 Boyd
6,490,291 B1 12/2002 Lee et al.
6,496,491 B2 12/2002 Chuah et al.
6,496,931 B1 12/2002 Rajchel et al.
6,505,046 B1 1/2003 Baker
6,505,048 B1 1/2003 Moles et al.
6,505,049 B1 1/2003 Dorenbosch
6,505,120 B2 1/2003 Yamashita et al.
6,505,163 B1 1/2003 Zhang et al.
6,512,754 B2 1/2003 Feder et al.
6,516,055 B1 2/2003 Bedeski et al.
6,516,416 B2 2/2003 Gregg et al.
6,519,252 B2 2/2003 Sallberg
6,519,458 B2 2/2003 Oh et al.
6,522,876 B1 2/2003 Weiland et al.
6,526,275 B1 2/2003 Calvert
6,526,349 B2 2/2003 Bullock et al.
6,532,418 B2 3/2003 Chun et al.
6,545,596 B1 4/2003 Moon
6,546,257 B1 4/2003 Stewart
6,560,442 B1 5/2003 Yost et al.
6,560,461 B1 5/2003 Fomukong et al.
6,571,279 B1 5/2003 Herz et al.
6,577,643 B1 6/2003 Rai et al.
6,577,644 B1 6/2003 Chuah et al.
6,594,482 B1 7/2003 Findikli et al.
6,615,131 B1 9/2003 Rennard et al.
6,618,474 B1 9/2003 Reese
6,618,593 B1 9/2003 Drutman et al.
6,622,016 B1 9/2003 Sladek et al.
6,628,627 B1 9/2003 Zendle et al.
6,628,928 B1 9/2003 Crosby et al.
6,628,938 B1 9/2003 Rachabathuni et al.
6,633,633 B1 10/2003 Bedingfield
6,640,184 B1 10/2003 Rabe
6,647,257 B2 11/2003 Owensby
6,647,269 B2 11/2003 Hendrey et al.
6,650,901 B1 11/2003 Schuster et al.
6,654,610 B1 11/2003 Chen et al.
6,662,014 B1 12/2003 Walsh
6,665,536 B1 12/2003 Mahany
6,665,718 B1 12/2003 Chuah et al.
6,671,272 B2 12/2003 Vaziri et al.
6,675,017 B1 1/2004 Zellner et al.
6,675,208 B1 1/2004 Rai et al.
6,677,894 B2 1/2004 Sheynblat et al.
6,697,018 B2 2/2004 Stewart et al.
6,697,783 B1 2/2004 Brinkman et al.
6,701,160 B1 3/2004 Pinder et al.
6,701,251 B2 3/2004 Stefan et al.
6,704,311 B1 3/2004 Chuah et al.
6,716,101 B1 4/2004 Meadows et al.
6,721,406 B1 4/2004 Contractor
6,725,048 B2 4/2004 Mao et al.
6,731,238 B2 5/2004 Johnson
6,732,080 B1 5/2004 Blants
6,732,101 B1 5/2004 Cook
6,732,176 B1 5/2004 Stewart et al.
6,738,808 B1 5/2004 Zellner et al.
6,754,504 B1 6/2004 Reed

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,754,582 B1 6/2004 Smith et al.
6,759,960 B2 7/2004 Stewart et al.
6,772,064 B1 8/2004 Smith et al.
6,799,049 B1 9/2004 Zellner et al.
6,801,509 B1 10/2004 Chuah et al.
6,816,720 B2 11/2004 Hussain et al.
6,819,929 B2 11/2004 Antonucci et al.
6,820,062 B1 11/2004 Gupta et al.
6,829,475 B1 12/2004 Lee et al.
6,850,758 B1 2/2005 Paul et al.
6,867,733 B2 3/2005 Sandhu et al.
6,868,074 B1 3/2005 Hanson
6,874,011 B1 3/2005 Spielman
6,876,858 B1 4/2005 Duvall et al.
6,898,569 B1 5/2005 Bansal et al.
6,937,869 B1 8/2005 Rayburn
6,937,998 B1 8/2005 Swartz et al.
6,954,147 B1 10/2005 Cromer et al.
6,985,747 B2 1/2006 Chithambaram
6,999,572 B1 2/2006 Shaffer et al.
7,005,985 B1 2/2006 Steeves
7,009,556 B2 3/2006 Stewart et al.
7,023,995 B2 4/2006 Olsson
7,043,231 B2 5/2006 Bhatia et al.
7,058,594 B2 6/2006 Stewart et al.
7,069,319 B2 6/2006 Zellner et al.
7,085,555 B2 8/2006 Zellner et al.
7,103,368 B2 9/2006 Teshima
7,103,476 B2 9/2006 Smith et al.
7,106,843 B1 9/2006 Gainsboro et al.
7,110,749 B2 9/2006 Zellner et al.
7,116,977 B1 10/2006 Moton et al.
7,124,101 B1 10/2006 Mikurak
7,130,630 B1 10/2006 Enzmann et al.
7,139,722 B2 11/2006 Perrella et al.
7,155,199 B2 12/2006 Zalewski et al.
7,177,651 B1 * 2/2007 Almassy 455/456.1
7,181,225 B1 2/2007 Moton et al.
7,181,529 B2 2/2007 Bhatia et al.
7,188,027 B2 3/2007 Smith et al.
7,190,960 B2 3/2007 Wilson et al.
7,203,502 B2 4/2007 Wilson et al.
7,212,829 B1 5/2007 Lau et al.
7,224,978 B2 5/2007 Zellner et al.
7,236,799 B2 6/2007 Wilson et al.
RE39,717 E 7/2007 Yates et al.
7,245,925 B2 7/2007 Zellner
7,260,378 B2 8/2007 Holland et al.
7,272,493 B1 9/2007 Hamrick et al.
7,292,939 B2 11/2007 Smith et al.
7,295,924 B2 11/2007 Smith et al.
7,362,851 B2 4/2008 Contractor
7,383,052 B2 6/2008 Moton et al.
7,386,396 B2 6/2008 Johnson
7,787,887 B2 * 8/2010 Gupta et al. 455/456.1
2001/0001239 A1 5/2001 Stewart
2001/0007450 A1 7/2001 Begum
2001/0021646 A1 9/2001 Antonucci et al.
2001/0028301 A1 10/2001 Geiger et al.
2001/0034709 A1 10/2001 Stoifo et al.
2001/0049275 A1 12/2001 Pierry et al.
2001/0051911 A1 12/2001 Marks et al.
2002/0035474 A1 3/2002 Alpdemir
2002/0035493 A1 3/2002 Mozayeny et al.
2002/0037709 A1 3/2002 Bhatia et al.
2002/0037722 A1 3/2002 Hussain et al.
2002/0037731 A1 3/2002 Mao et al.
2002/0037744 A1 3/2002 Bhatia et al.
2002/0037750 A1 3/2002 Hussain et al.
2002/0038362 A1 3/2002 Bhatia et al.
2002/0038384 A1 3/2002 Khan et al.
2002/0038386 A1 3/2002 Bhatia et al.
2002/0046069 A1 4/2002 Mozayeny et al.
2002/0046077 A1 4/2002 Mozayeny et al.
2002/0046090 A1 4/2002 Stewart
2002/0052781 A1 5/2002 Aufricht et al.
2002/0077083 A1 6/2002 Zellner et al.
2002/0077084 A1 6/2002 Zellner et al.
2002/0077118 A1 6/2002 Zellner et al.
2002/0077130 A1 6/2002 Owensby
2002/0077897 A1 6/2002 Zellner et al.
2002/0087335 A1 7/2002 Meyers et al.
2002/0090932 A1 7/2002 Bhatia et al.
2002/0091991 A1 7/2002 Castro
2002/0095312 A1 7/2002 Wheat
2002/0102993 A1 8/2002 Hendrey et al.
2002/0107027 A1 8/2002 O'Neil
2002/0120713 A1 8/2002 Gupta et al.
2002/0161637 A1 10/2002 Sugaya
2002/0174147 A1 11/2002 Wang et al.
2003/0003990 A1 1/2003 Von Kohorn
2003/0016233 A1 1/2003 Charpentier
2003/0018527 A1 1/2003 Filepp et al.
2003/0140088 A1 7/2003 Robinson et al.
2003/0169151 A1 9/2003 Ebling et al.
2004/0002329 A1 1/2004 Bhatia et al.
2004/0097243 A1 5/2004 Zellner et al.
2004/0111269 A1 6/2004 Koch
2004/0116131 A1 * 6/2004 Hochrainer et al. 455/456.1
2004/0151151 A1 8/2004 Kubler et al.
2004/0164898 A1 8/2004 Stewart
2004/0186902 A1 9/2004 Stewart et al.
2004/0203903 A1 10/2004 Wilson et al.
2004/0205198 A1 10/2004 Zellner et al.
2004/0228330 A1 11/2004 Kubler et al.
2004/0246940 A1 12/2004 Kubler et al.
2004/0252051 A1 12/2004 Johnson
2004/0264442 A1 12/2004 Kubler et al.
2004/0266453 A1 12/2004 Maanoja et al.
2005/0002419 A1 1/2005 Doviak et al.
2005/0004838 A1 1/2005 Perkowski et al.
2005/0017068 A1 1/2005 Zalewski et al.
2005/0043036 A1 2/2005 Ioppe et al.
2005/0060365 A1 3/2005 Robinson et al.
2005/0096067 A1 5/2005 Martin
2005/0114777 A1 5/2005 Szeto
2005/0151655 A1 7/2005 Hamrick et al.
2005/0246097 A1 11/2005 Hamrick et al.
2005/0272445 A1 12/2005 Zellner
2006/0022048 A1 2/2006 Johnson
2006/0030335 A1 2/2006 Zellner et al.
2006/0030339 A1 2/2006 Zhovnirovsky et al.
2006/0059043 A1 3/2006 Chan et al.
2006/0089134 A1 4/2006 Moton et al.
2006/0094447 A1 5/2006 Zellner
2006/0099966 A1 5/2006 Moton et al.
2006/0105784 A1 5/2006 Zellner et al.
2006/0106537 A1 5/2006 Hamrick et al.
2006/0164302 A1 7/2006 Stewart et al.
2006/0167986 A1 7/2006 Trzyna et al.
2006/0183467 A1 8/2006 Stewart et al.
2006/0189327 A1 8/2006 Zellner et al.
2006/0189332 A1 8/2006 Benco et al.
2006/0195570 A1 8/2006 Zellner et al.
2006/0253252 A1 11/2006 Hamrick et al.
2007/0005188 A1 1/2007 Johnson
2007/0010260 A1 1/2007 Zellner et al.
2007/0042789 A1 2/2007 Moton et al.
2007/0105565 A1 5/2007 Enzmann et al.
2007/0124721 A1 5/2007 Cowing et al.
2007/0136603 A1 6/2007 Kuecuekyan
2007/0232326 A1 10/2007 Johnson
2007/0233387 A1 10/2007 Johnson
2007/0244633 A1 * 10/2007 Phillips et al. 701/207
2007/0250920 A1 10/2007 Lindsay
2007/0275730 A1 * 11/2007 Bienas et al. 455/456.1
2007/0276587 A1 11/2007 Johnson
2007/0287473 A1 * 12/2007 Dupray 455/456.1
2008/0030308 A1 2/2008 Johnson

(56) **References Cited**

U.S. PATENT DOCUMENTS

2008/0071761	A1*	3/2008	Singh et al.	707/5
2008/0096529	A1	4/2008	Zellner	
2008/0170679	A1*	7/2008	Sheha et al.	379/201.06

FOREIGN PATENT DOCUMENTS

EP	917320	5/1999
EP	935364	8/1999
EP	924914	4/2003
EP	779752	6/2004
EP	1435749	7/2004
EP	1445923	8/2004
EP	838933	4/2008
GB	2396779	6/2004
JP	01-194628	8/1989
JP	03-128540	5/1991
JP	08-87296	4/1995
JP	07-234789	9/1995
JP	07-288514	10/1995
JP	07-319706	12/1995
JP	08-44568	2/1996
JP	11-168478	6/1999
WO	WO 98/19484	5/1998
WO	WO 99/16263	4/1999
WO	WO 99/27716	6/1999
WO	WO 99/51005	10/1999
WO	WO 99/55012	10/1999
WO	WO 00/02365	10/2000
WO	WO 00/76249	12/2000
WO	WO 02/11407	2/2002
WO	WO 2004/080092	9/2004

OTHER PUBLICATIONS

Andy Harter and Andy Hooper, A Distributed Location system for the Active Office, IEEE Network, Jan./Feb. 1994.

Max J. Engenhofer, Spatial SQL: A Query and Presentation Language, IEEE Network, Feb. 1994.

Mike Spreitzer and Marvin Theimer, Providing Location Information in a Ubiquitous Computing Environment, Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, Dec. 1993.

George W. Fitzmaurice, Situated Information Spaces and Spatially Aware Palmtop Computers, Communication of the ACM, Jul. 1993.

Ronald Azuma, Tracking Requirements for Augmented Reality, Communications of the ACM, vol. 36 No. 1, Jan. 1992.

Roy Want, et al., The Active Badge Location System, ACM Transactions on Information Systems, vol. 10, No. 1, Jan. 1992.

Marvin White, Emerging Requirements for Digital Maps for In-Vehicle Pathfinding and Other Traveller Assistance, Vehicular Navigation and Information Systems Conference Proceedings, Part 1, Oct. 1991.

Fred Phail, The Power of a Personal Computer for Car Information and Communications Systems, Vehicular Navigation and Information Systems Conference Proceedings, Part 1, Oct. 1991.

Thomas A. Dingus, et al., Human Factors Engineering the TravTek Driver Interface, Vehicular Navigation and Information Systems Conference Proceedings, Part II, Oct. 1991.

Michael Muffat et al., European Cooperation on Dual Mode Route Guidance Perspectives for Advanced Research Partners, Vehicular Navigation and Information Systems Conference Proceedings, Part II, Oct. 1991.

High-Performance Wireless Access Point for the Enterprise, ORiNOCO™ AP-100 Access Point for the Enterprise, Lucent Technologies, 2000.

MobileStar Network, MobileStar Network First to Provide Business travelers with High-Speed Data Access via the Internet-Wirelessly, New York, NY, Jun. 24, 1998.

ORiNCO AP-1000—Getting Started, Lucent Technologies.

Harry Chen, et al., "Dynamic Service Discovery for Mobile Computing: Intelligent Agents Meet Jini in the Aether," Cluster Computing, Special Issue on Internet Scalability, vol. 4, No. 4, Feb. 2001.

3rd Generating Partnership Project: Technical Specification Group Services and System Aspects; Functional Stage 2 Description of Location Services in UMTS (1999).

http://www.openwave.com/us/news_room/press_releases/2001/20020320, "Open Wave Announces Availability to End-to-End Set of Location Services for Wireless Internet".

Trembly, A., "Wireless products arm road warriors," National Underwriter, vol. 105, No. 3, pp. 23-25, Dialog 02113577 67213220 (Jan. 2001).

* cited by examiner

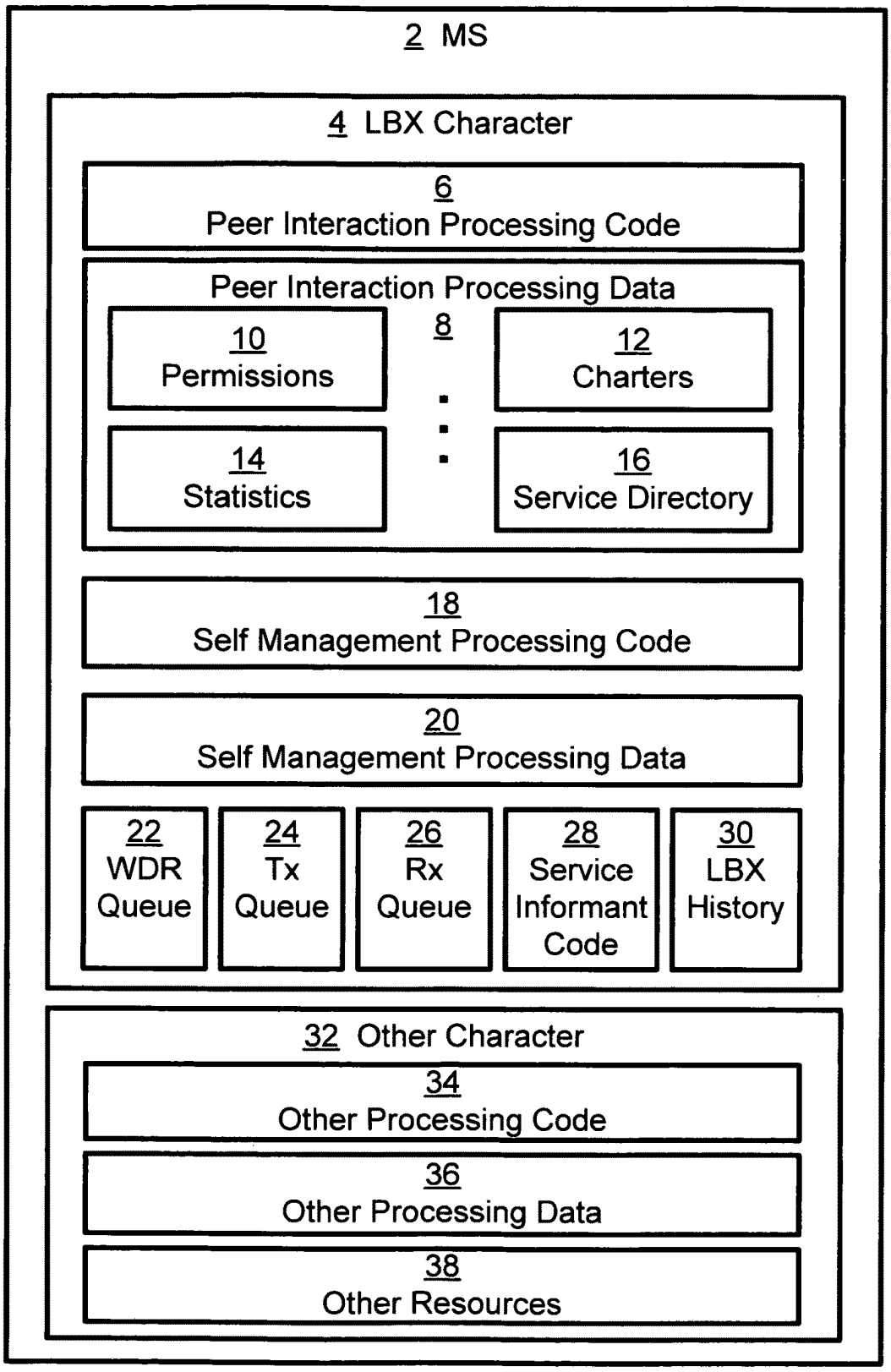

2 MS
4 LBX Character
6
Peer Interaction Processing Code
Peer Interaction Processing Data
10
Permissions
8
12
Charters
14
Statistics
16
Service Directory
18
Self Management Processing Code
20
Self Management Processing Data
22
WDR
Queue
24
Tx
Queue
26
Rx
Queue
28
Service
Informant
Code
30
LBX
History
32 Other Character
34
Other Processing Code
36
Other Processing Data
38
Other Resources

Fig. 1A

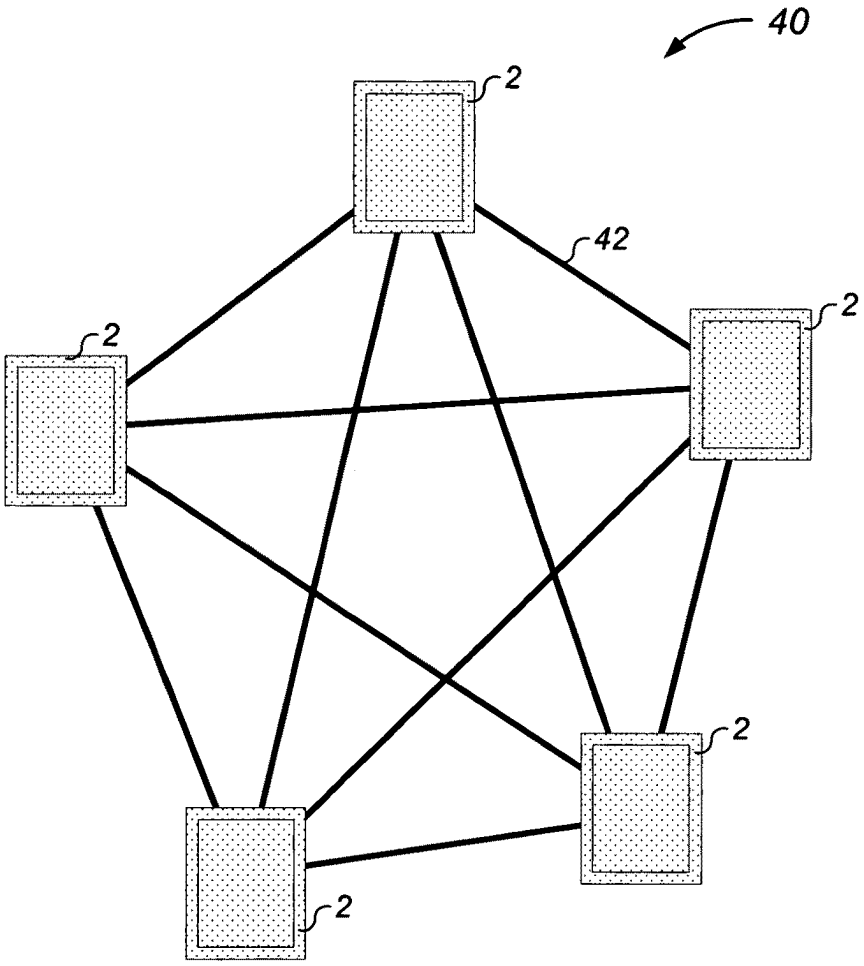


Fig. 1B

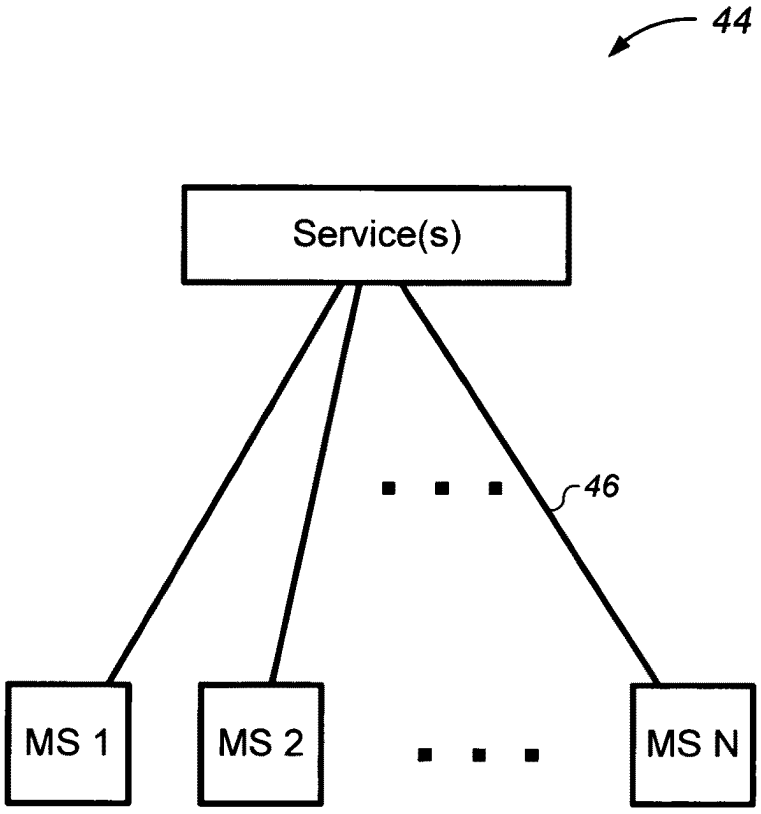


Fig. 1C

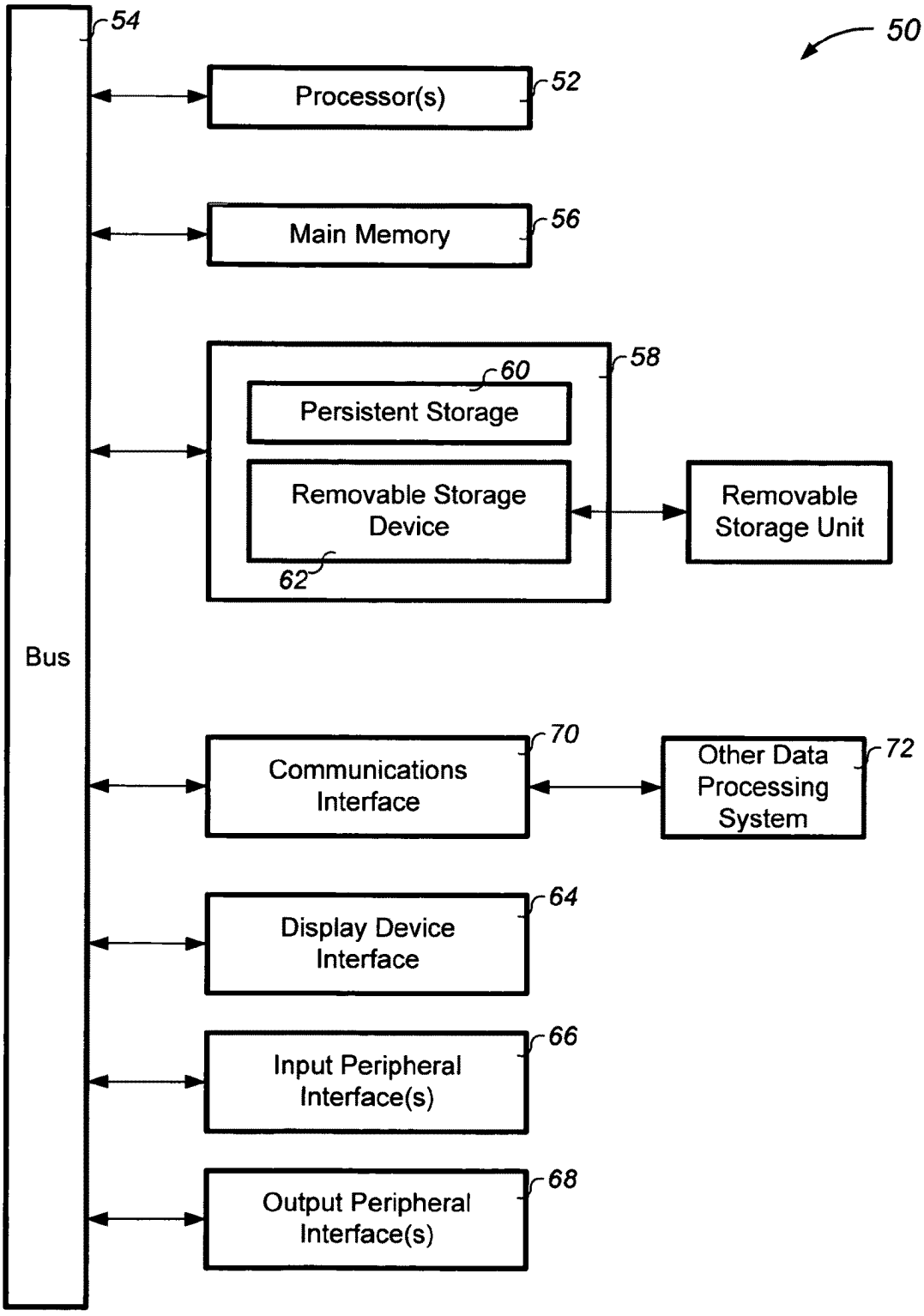

50
54
Processor(s)
52
Main Memory
56
58
60
Persistent Storage
Removable Storage Device
62
Removable Storage Unit
Bus
70
Communications Interface
Other Data Processing System
72
64
Display Device Interface
66
Input Peripheral Interface(s)
68
Output Peripheral Interface(s)

Fig. 1D

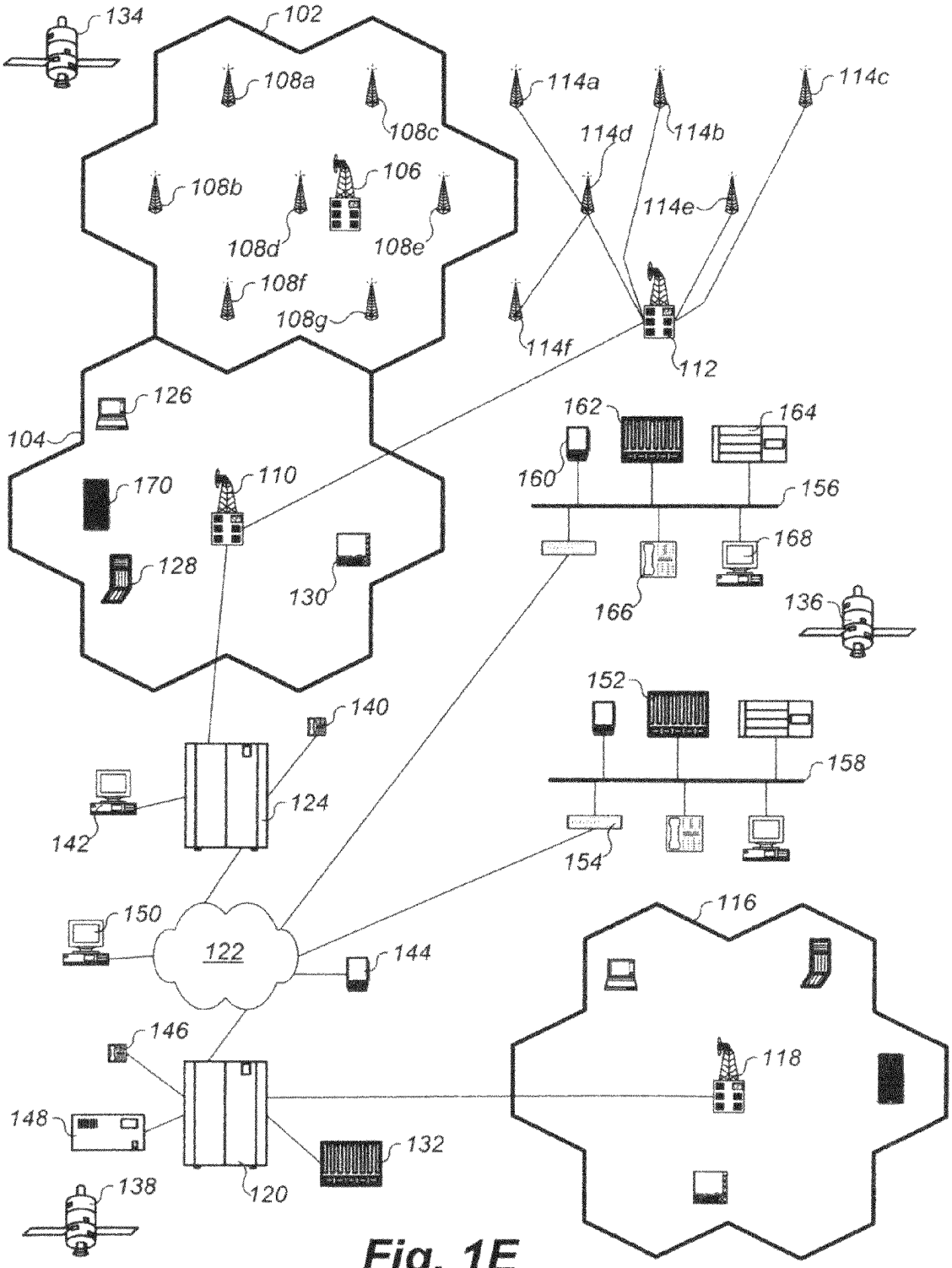

134
102
108a
108c
114a
114c
114d
114b
108b
106
114e
108d
108e
108f
108g
114f
112
126
104
162
164
160
156
170
110
168
128
130
166
136
140
152
158
124
142
154
150
122
144
116
146
118
148
132
120
138

Fig. 1E

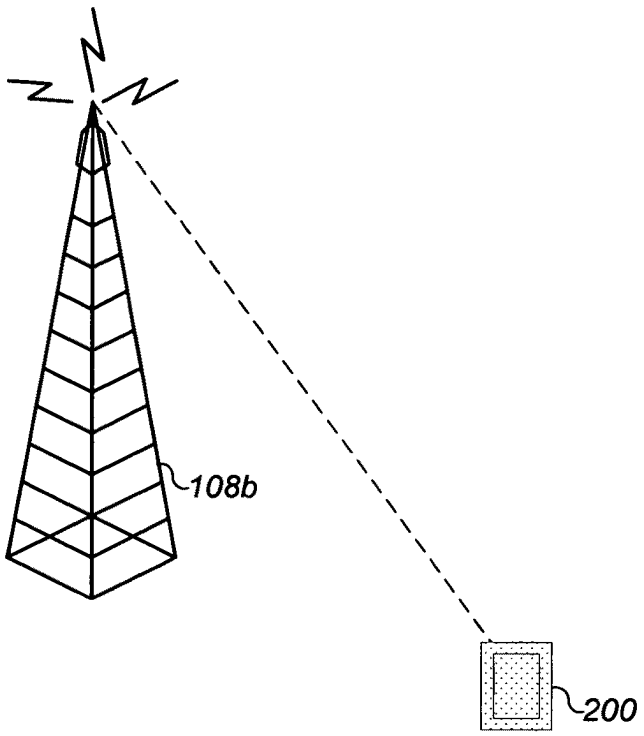


Fig. 2A

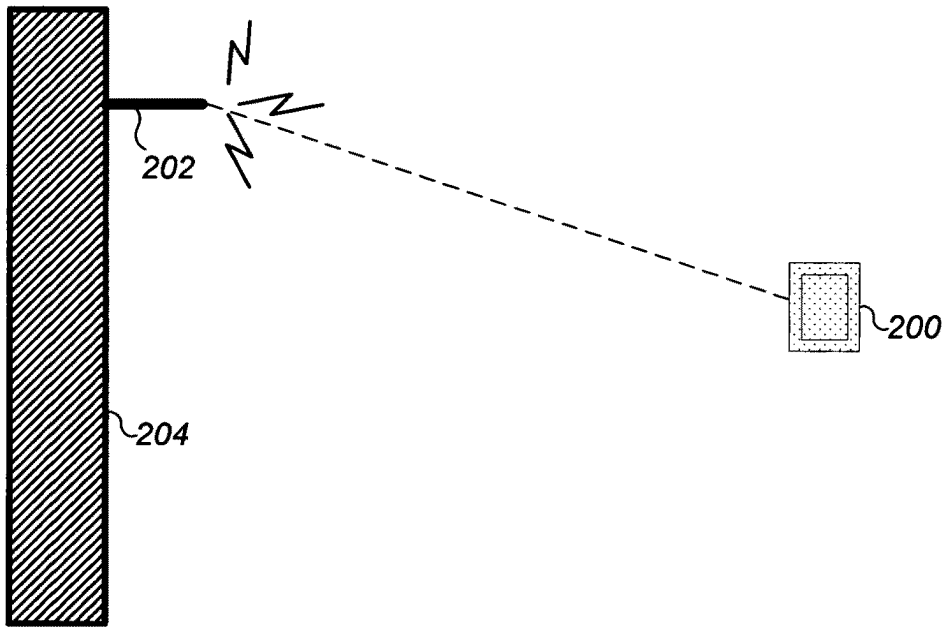


Fig. 2B

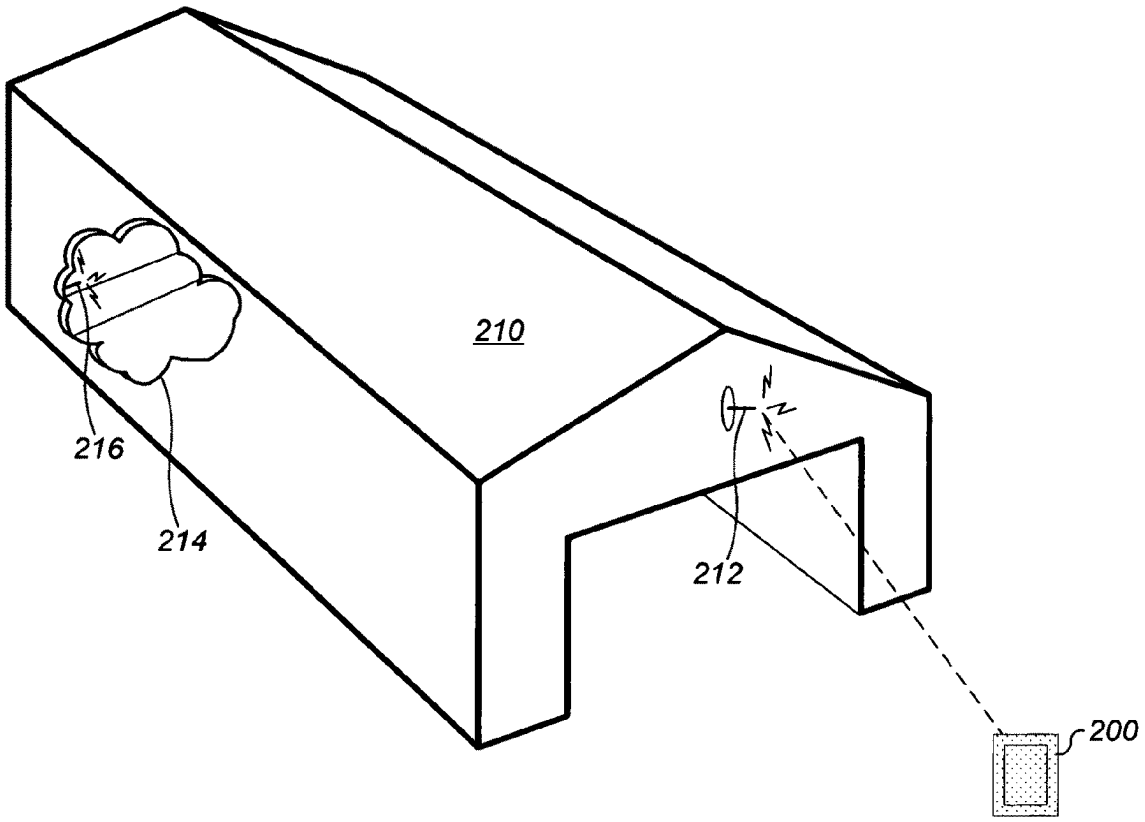


Fig. 2C

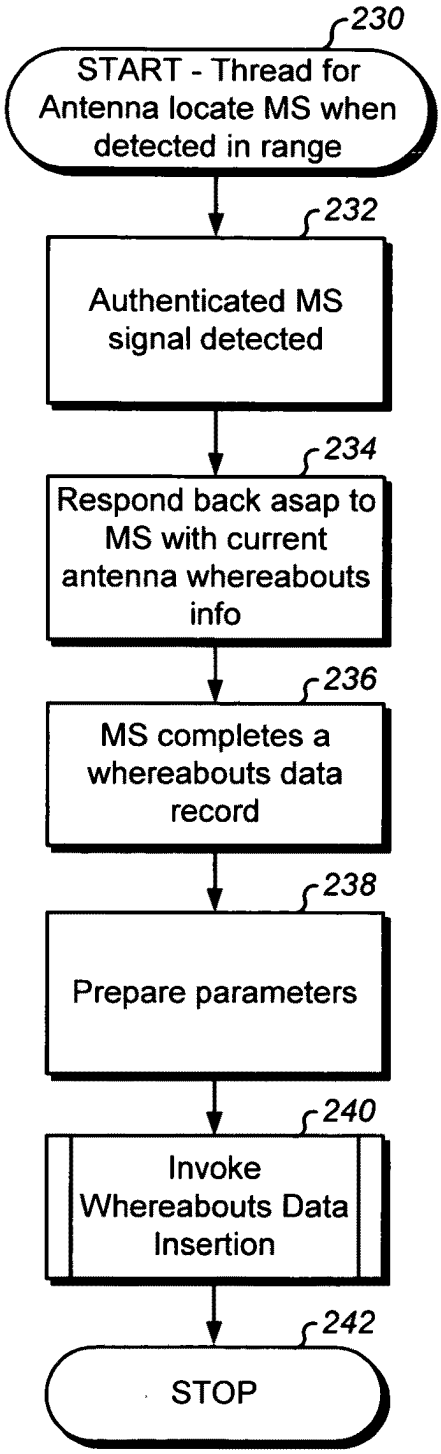

230
START - Thread for Antenna locate MS when detected in range
232
Authenticated MS signal detected
234
Respond back asap to MS with current antenna whereabouts info
236
MS completes a whereabouts data record
238
Prepare parameters
240
Invoke Whereabouts Data Insertion
242
STOP

Fig. 2D

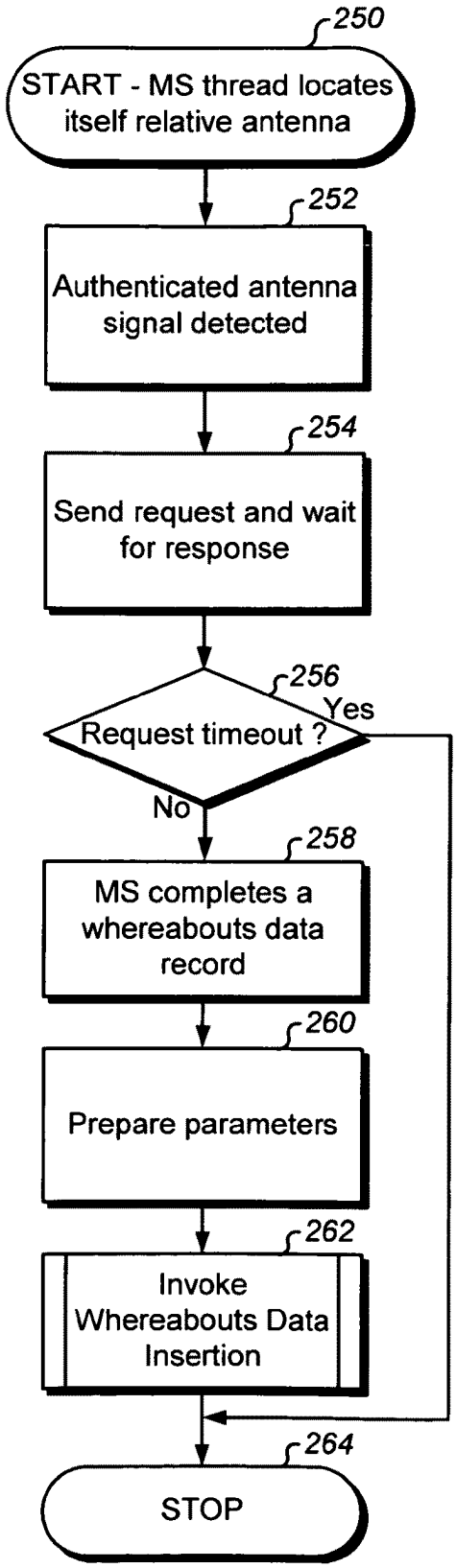

250
START - MS thread locates itself relative antenna
252
Authenticated antenna signal detected
254
Send request and wait for response
256
Request timeout ?
Yes
No
258
MS completes a whereabouts data record
260
Prepare parameters
262
Invoke Whereabouts Data Insertion
264
STOP

Fig. 2E

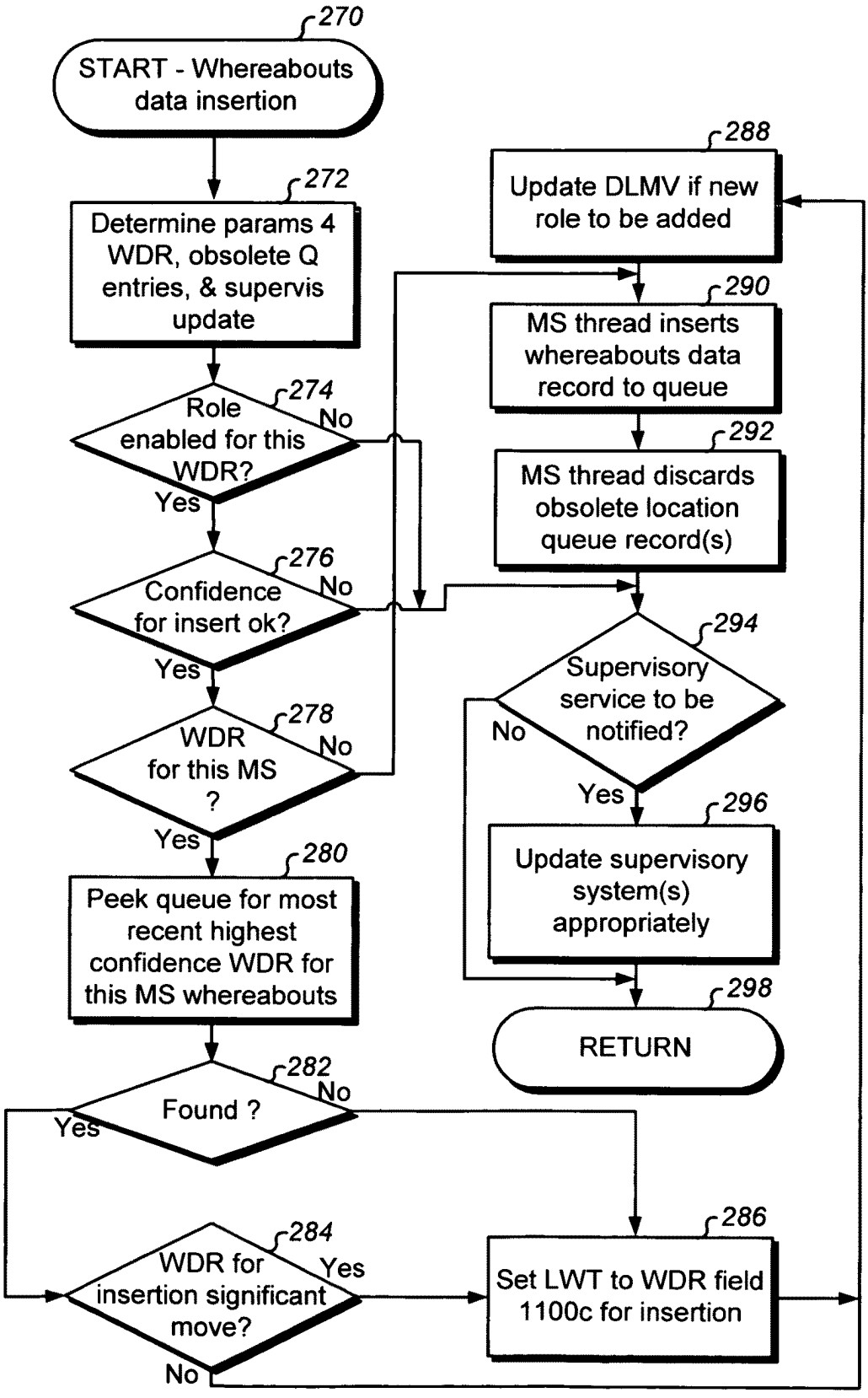

270
START - Whereabouts data insertion
272
Determine params 4 WDR, obsolete Q entries, & supervis update
274
Role enabled for this WDR?
No
Yes
276
Confidence for insert ok?
No
Yes
278
WDR for this MS ?
No
Yes
280
Peek queue for most recent highest confidence WDR for this MS whereabouts
282
Found ?
No
Yes
284
WDR for insertion significant move?
Yes
No
286
Set LWT to WDR field 1100c for insertion
288
Update DLMV if new role to be added
290
MS thread inserts whereabouts data record to queue
292
MS thread discards obsolete location queue record(s)
294
Supervisory service to be notified?
No
Yes
296
Update supervisory system(s) appropriately
298
RETURN

Fig. 2F

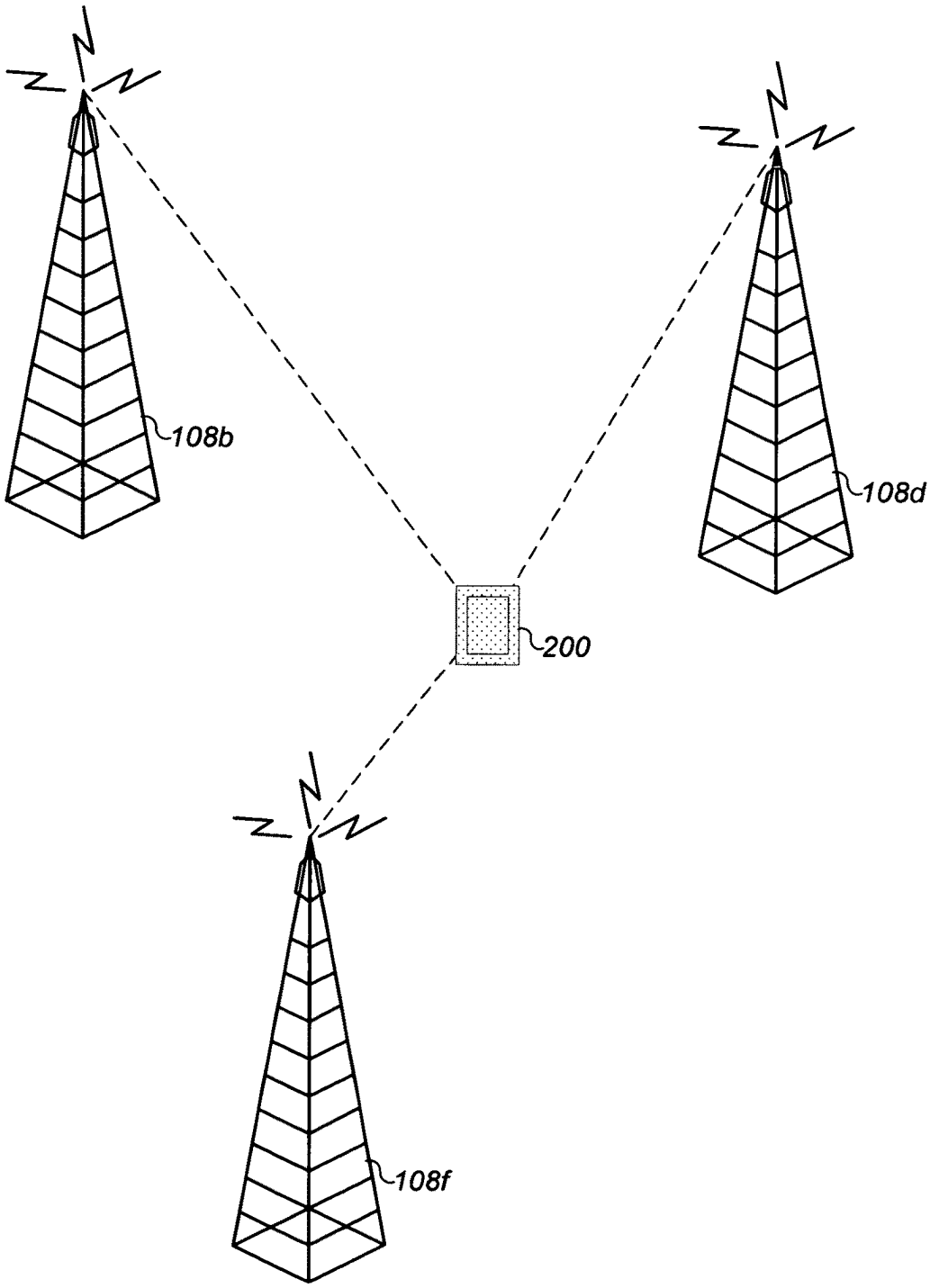


Fig. 3A

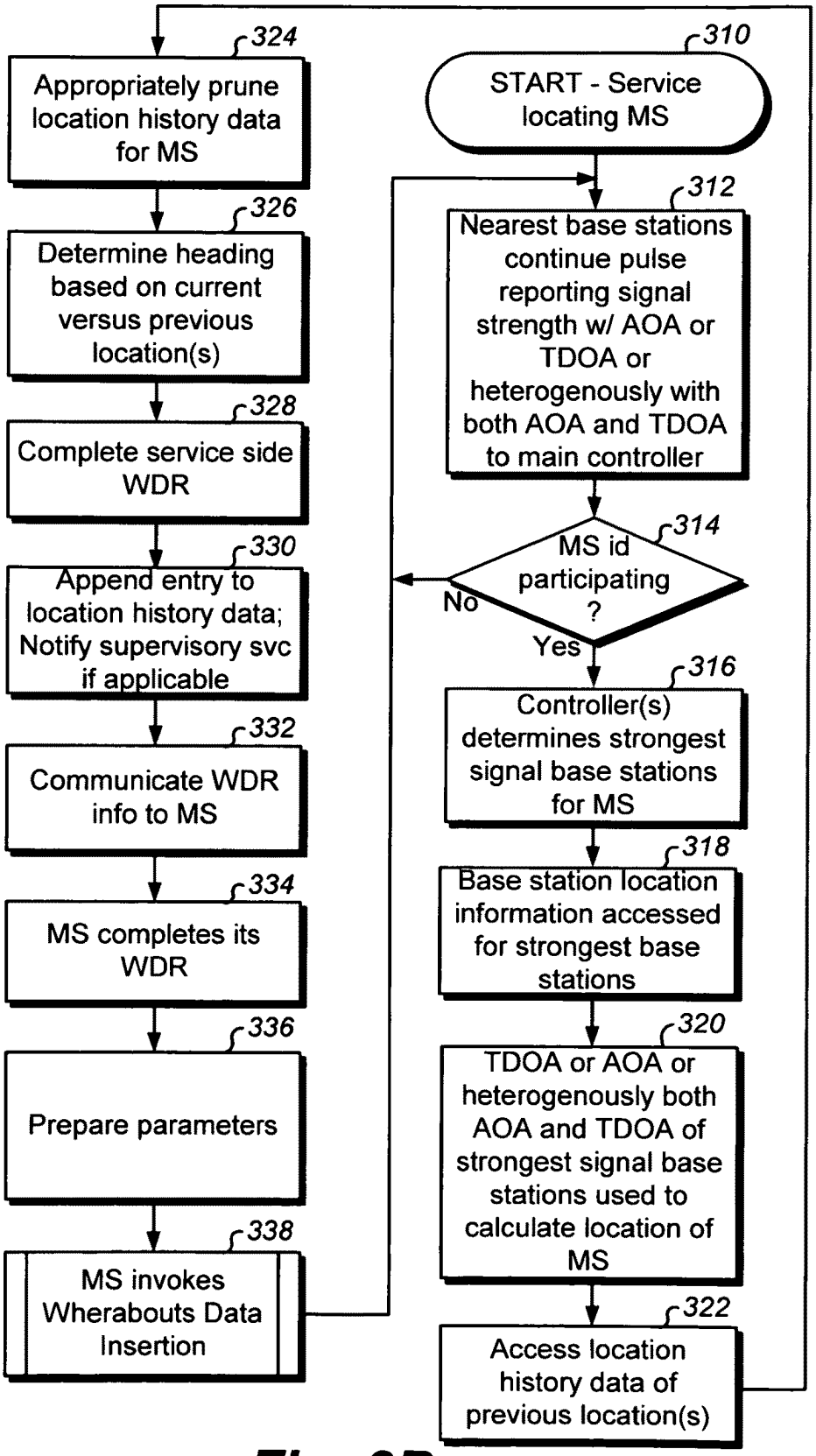

310
START - Service locating MS
312
Nearest base stations continue pulse reporting signal strength w/ AOA or TDOA or heterogenously with both AOA and TDOA to main controller
314
MS id participating ?
No
Yes
316
Controller(s) determines strongest signal base stations for MS
318
Base station location information accessed for strongest base stations
320
TDOA or AOA or heterogenously both AOA and TDOA of strongest signal base stations used to calculate location of MS
322
Access location history data of previous location(s)
324
Appropriately prune location history data for MS
326
Determine heading based on current versus previous location(s)
328
Complete service side WDR
330
Append entry to location history data; Notify supervisory svc if applicable
332
Communicate WDR info to MS
334
MS completes its WDR
336
Prepare parameters
338
MS invokes Wherabouts Data Insertion

Fig. 3B

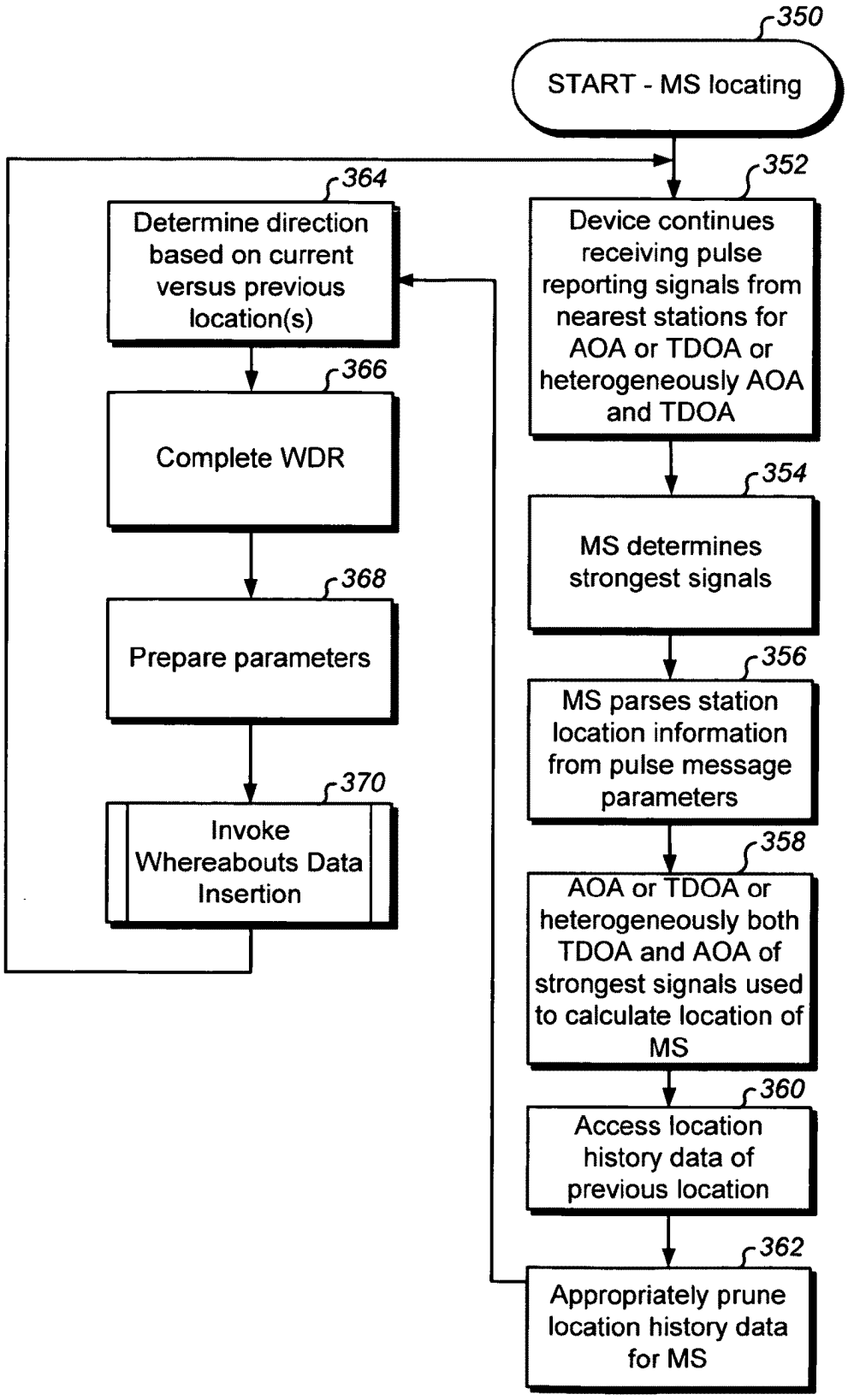

350
START - MS locating
352
Device continues receiving pulse reporting signals from nearest stations for AOA or TDOA or heterogeneously AOA and TDOA
354
MS determines strongest signals
356
MS parses station location information from pulse message parameters
358
AOA or TDOA or heterogeneously both TDOA and AOA of strongest signals used to calculate location of MS
360
Access location history data of previous location
362
Appropriately prune location history data for MS
364
Determine direction based on current versus previous location(s)
366
Complete WDR
368
Prepare parameters
370
Invoke Whereabouts Data Insertion

Fig. 3C

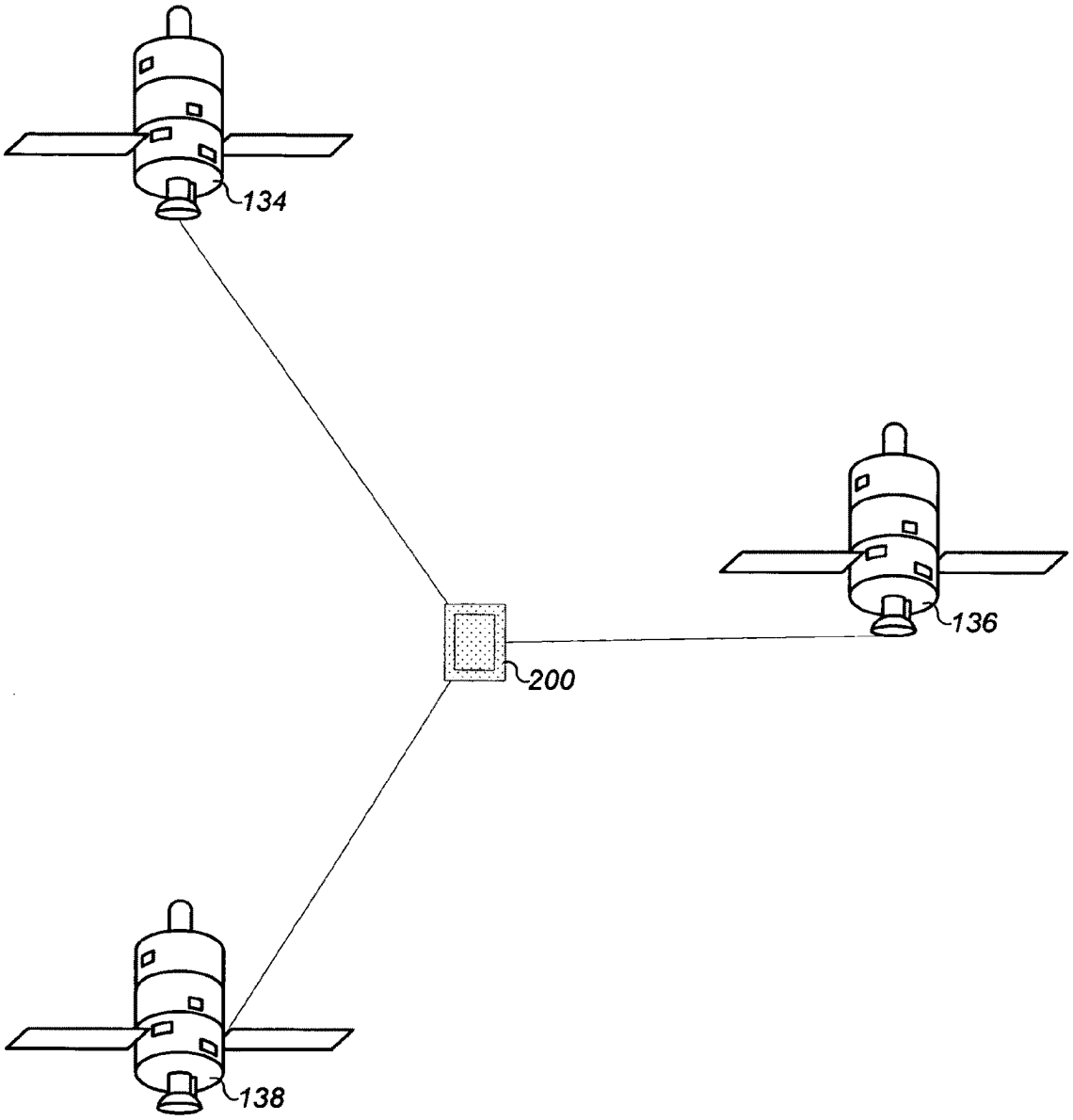


Fig. 4A

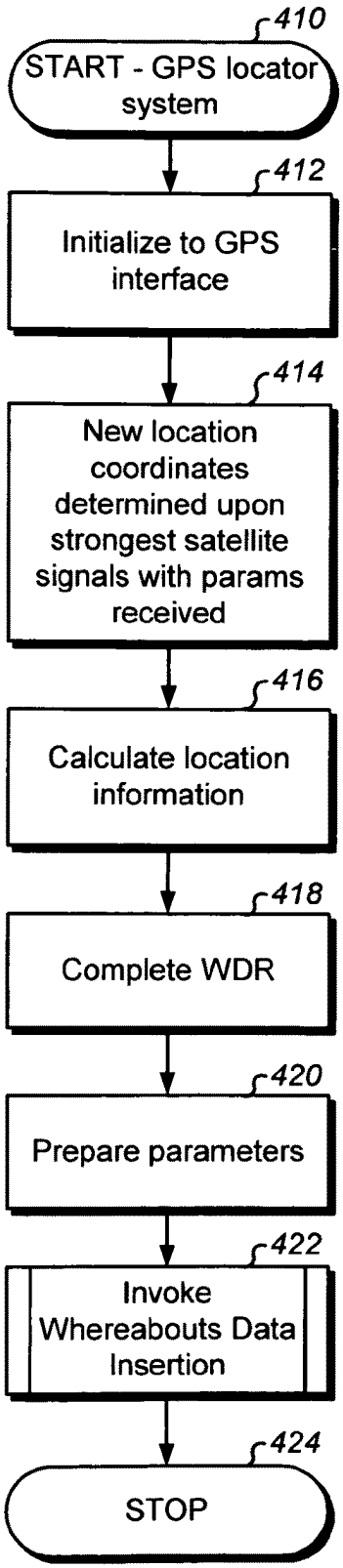

410
START - GPS locator system
412
Initialize to GPS interface
414
New location coordinates determined upon strongest satellite signals with params received
416
Calculate location information
418
Complete WDR
420
Prepare parameters
422
Invoke Whereabouts Data Insertion
424
STOP

Fig. 4B

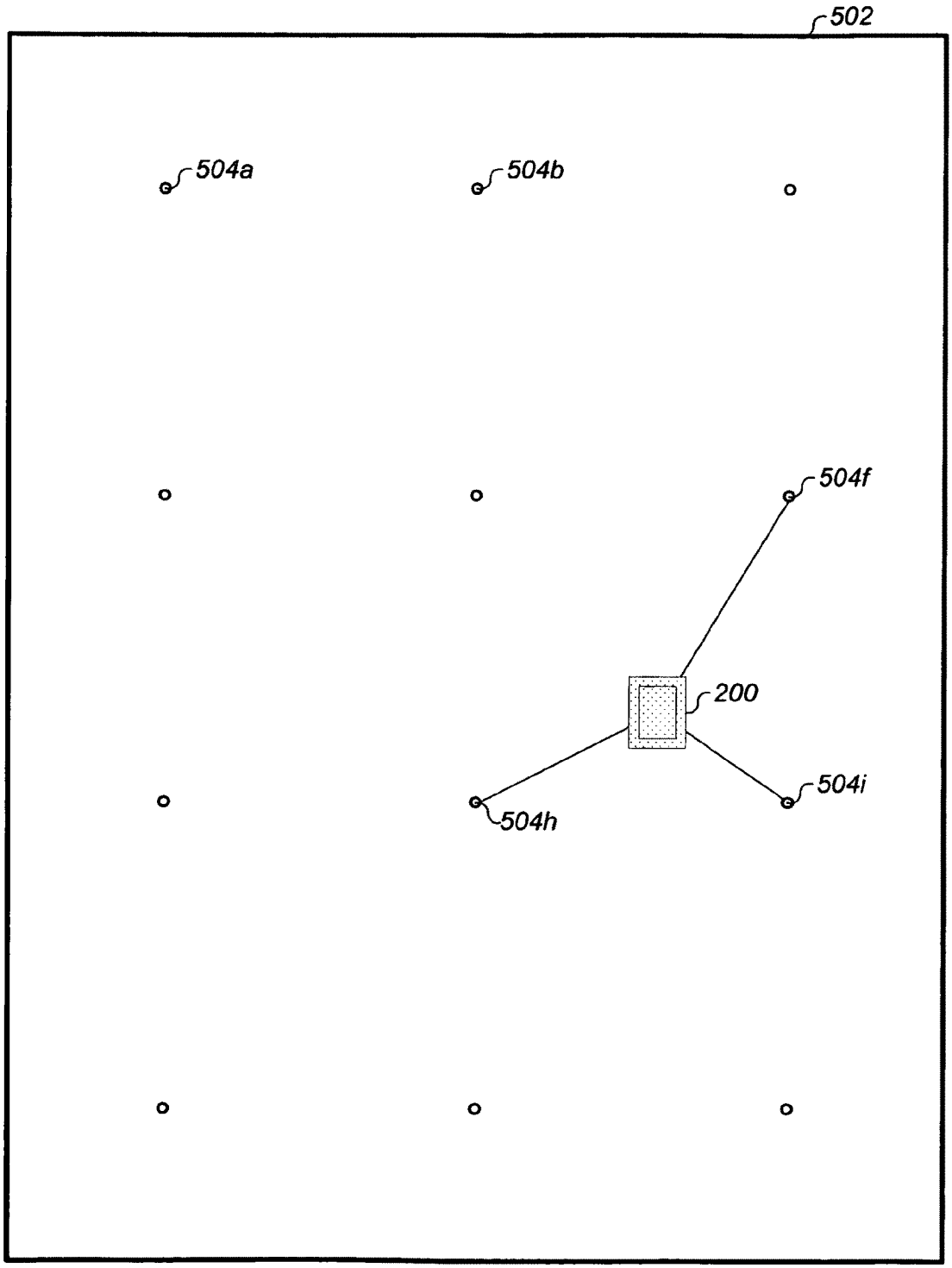


Fig. 5A

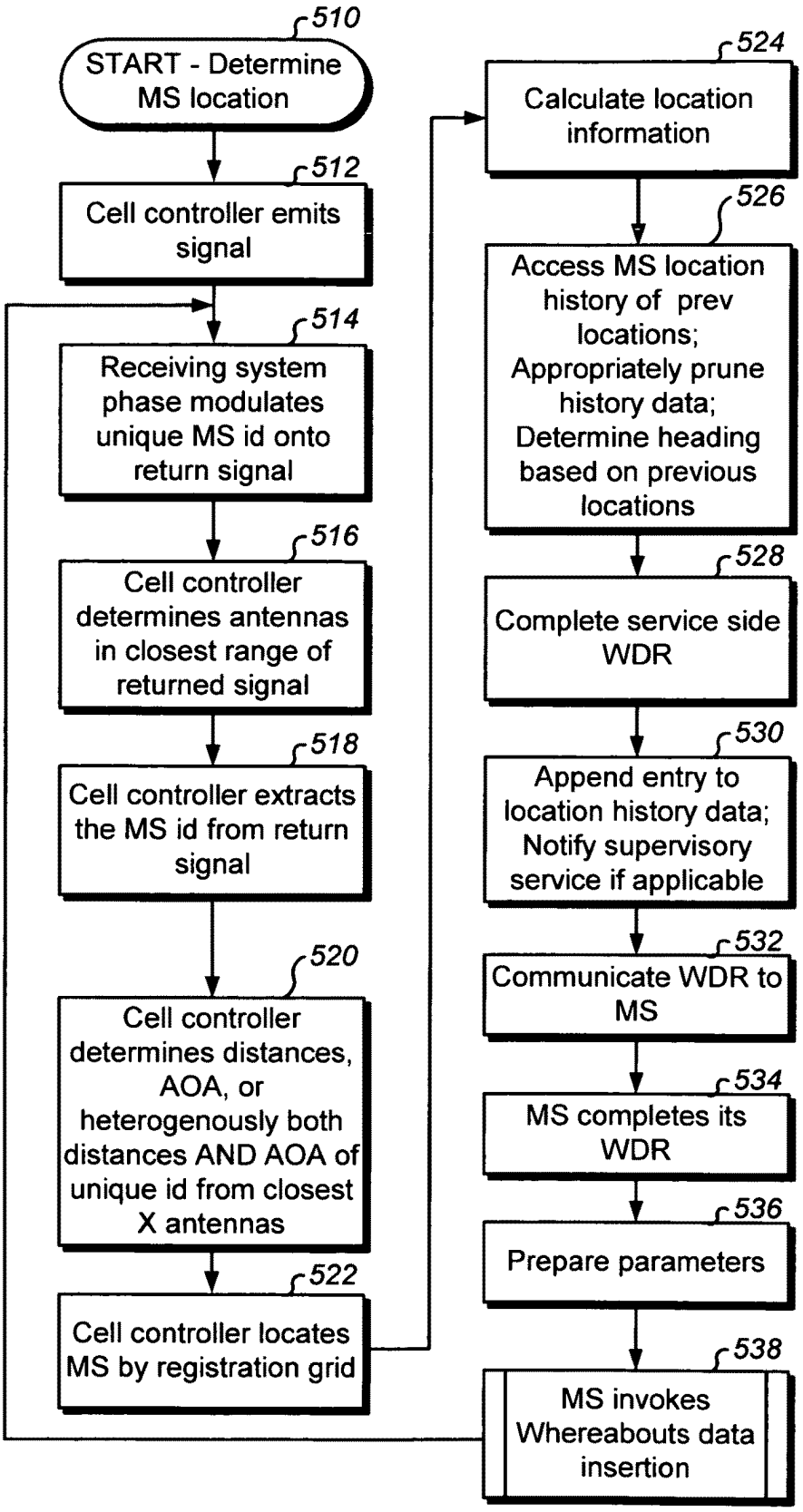

510
START - Determine MS location
512
Cell controller emits signal
514
Receiving system phase modulates unique MS id onto return signal
516
Cell controller determines antennas in closest range of returned signal
518
Cell controller extracts the MS id from return signal
520
Cell controller determines distances, AOA, or heterogenously both distances AND AOA of unique id from closest X antennas
522
Cell controller locates MS by registration grid
524
Calculate location information
526
Access MS location history of prev locations; Appropriately prune history data; Determine heading based on previous locations
528
Complete service side WDR
530
Append entry to location history data; Notify supervisory service if applicable
532
Communicate WDR to MS
534
MS completes its WDR
536
Prepare parameters
538
MS invokes Whereabouts data insertion

Fig. 5B

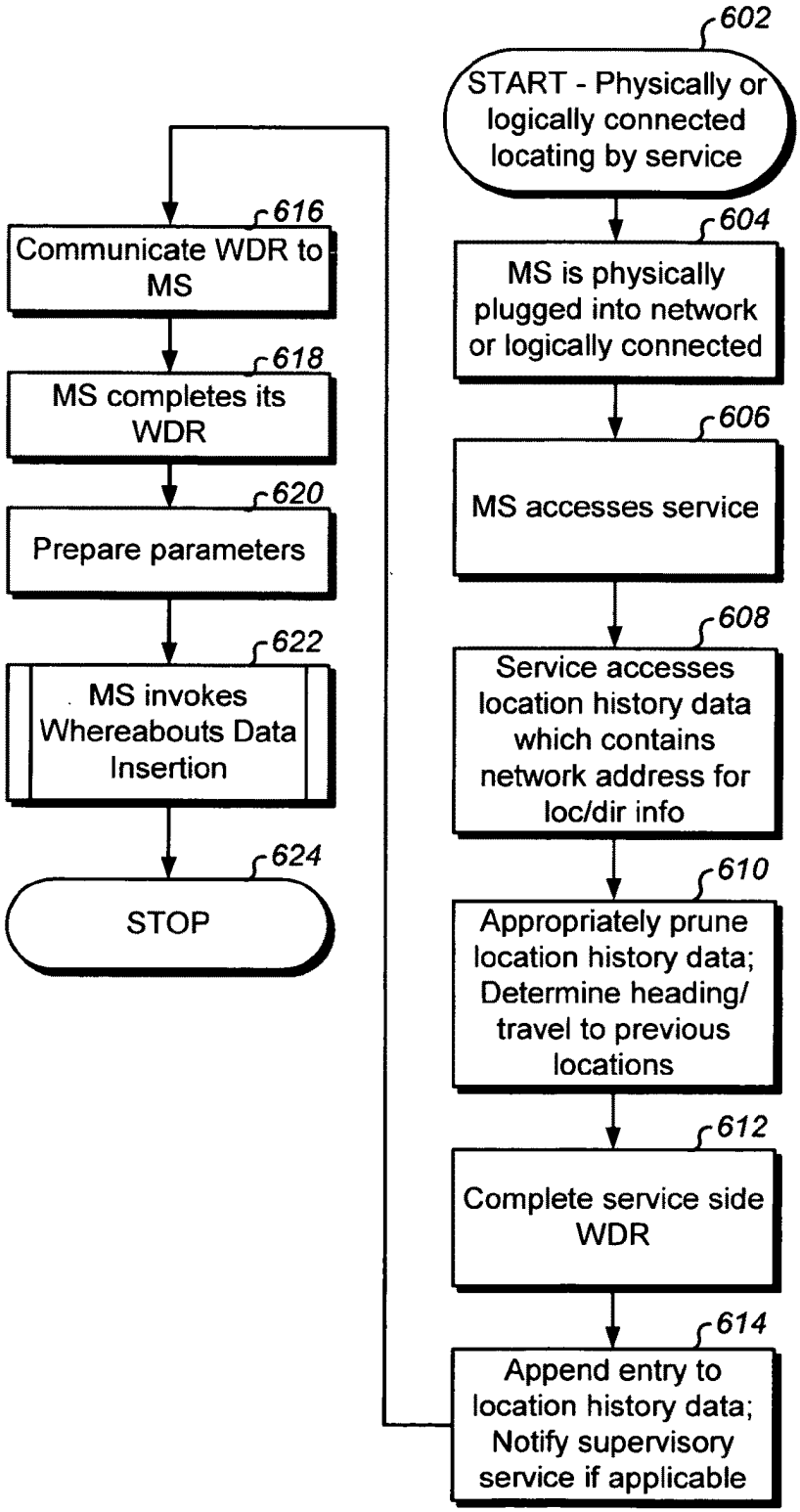

602
START - Physically or logically connected locating by service
604
MS is physically plugged into network or logically connected
606
MS accesses service
608
Service accesses location history data which contains network address for loc/dir info
610
Appropriately prune location history data; Determine heading/ travel to previous locations
612
Complete service side WDR
614
Append entry to location history data; Notify supervisory service if applicable
616
Communicate WDR to MS
618
MS completes its WDR
620
Prepare parameters
622
MS invokes Whereabouts Data Insertion
624
STOP

Fig. 6A

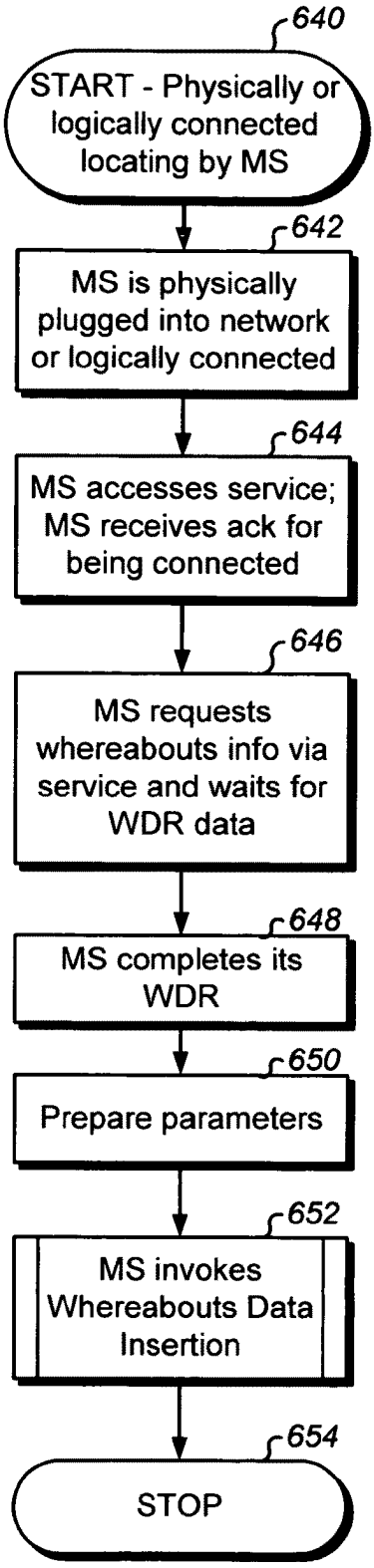

640
START - Physically or logically connected locating by MS
642
MS is physically plugged into network or logically connected
644
MS accesses service; MS receives ack for being connected
646
MS requests whereabouts info via service and waits for WDR data
648
MS completes its WDR
650
Prepare parameters
652
MS invokes Whereabouts Data Insertion
654
STOP

Fig. 6B

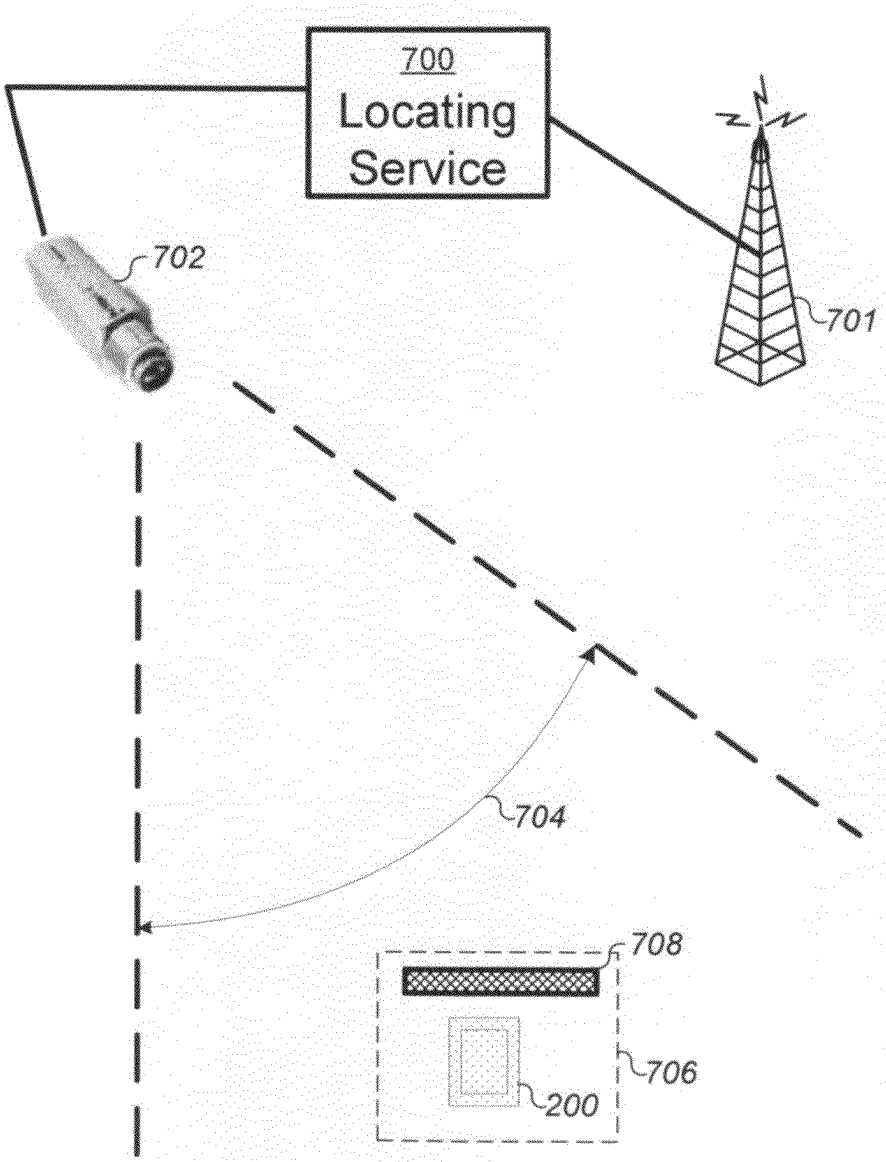


Fig. 7A

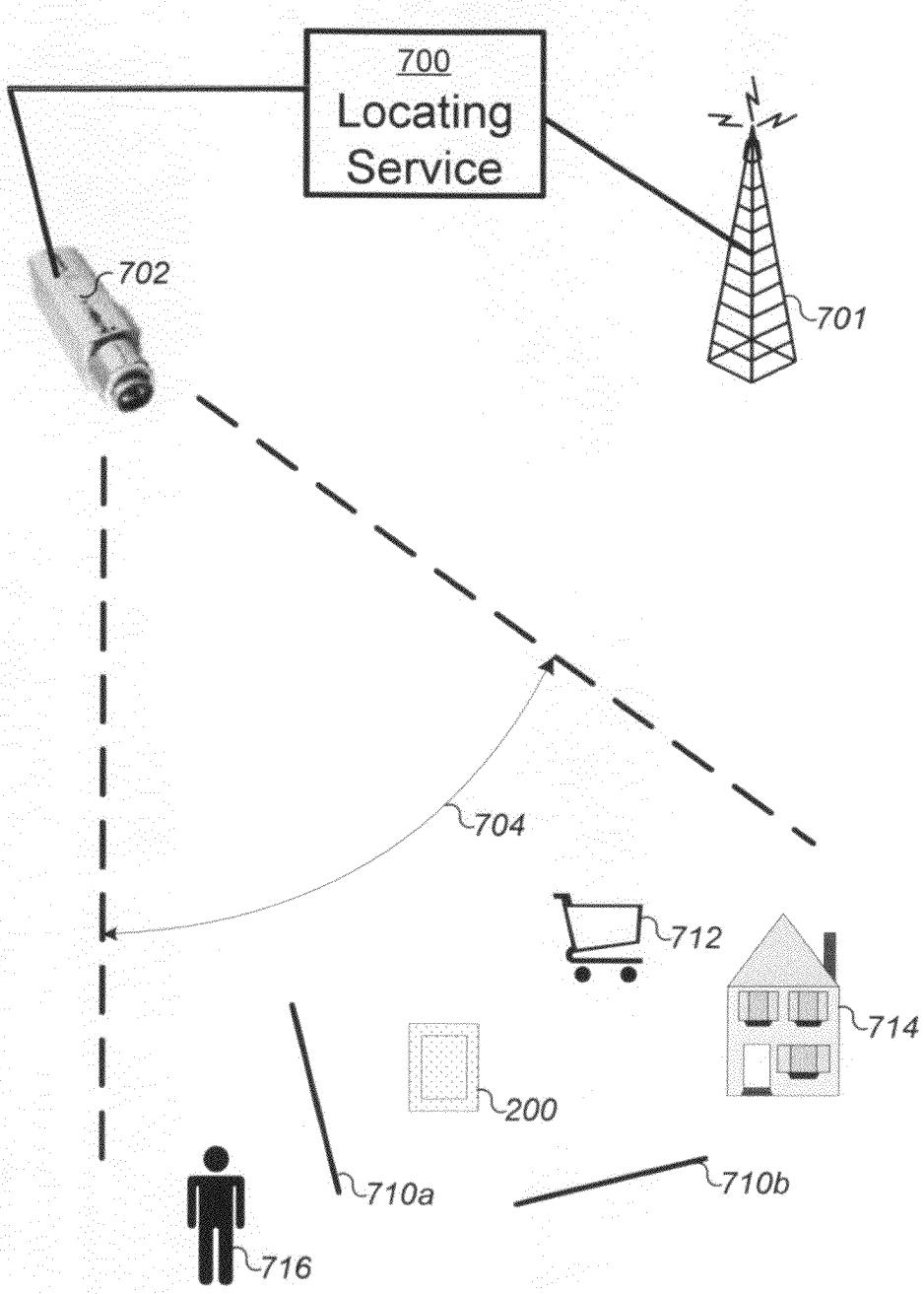


Fig. 7B

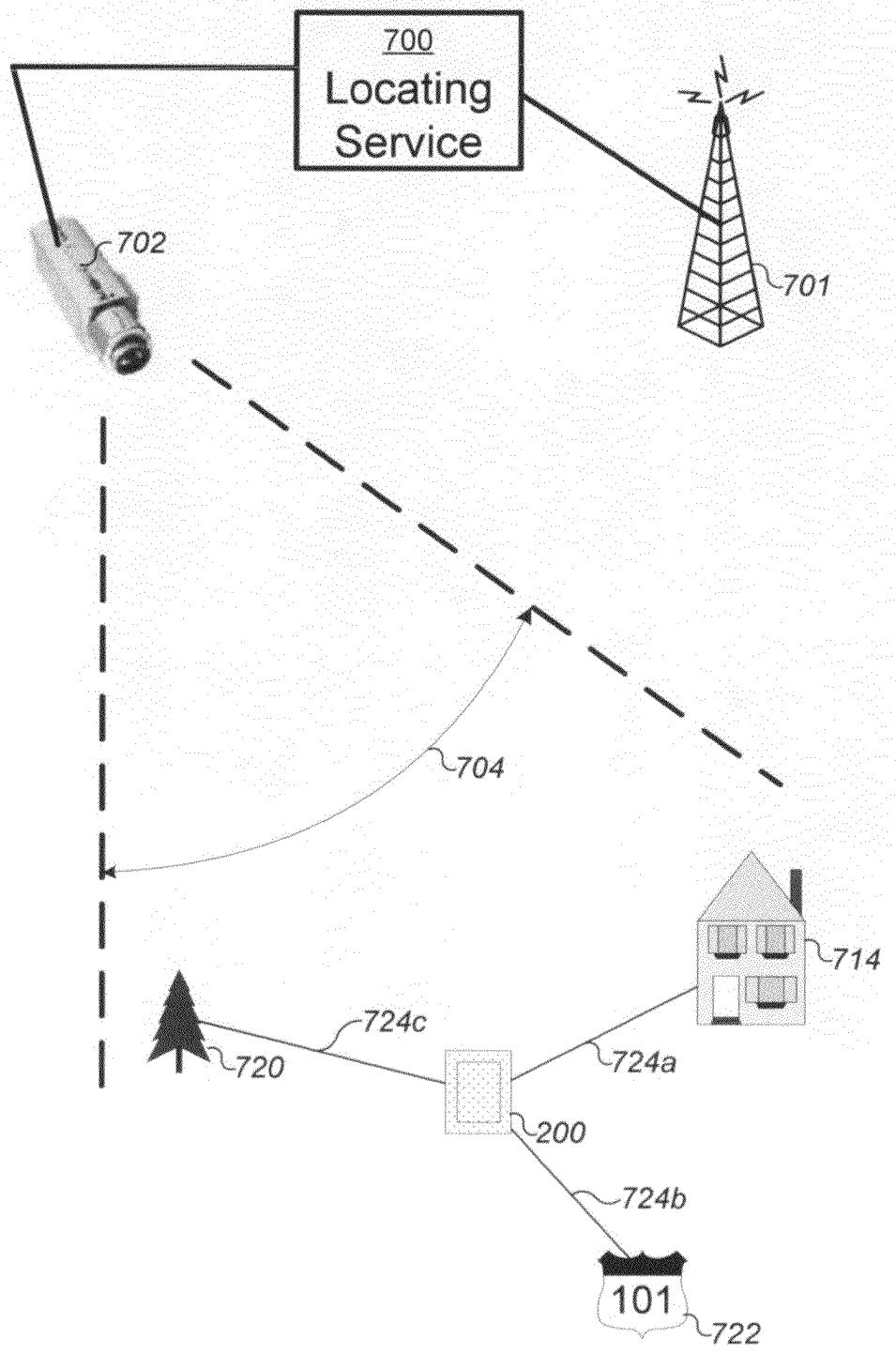

700
Locating Service
702
701
704
714
724c
720
724a
200
724b
101
722

Fig. 7C

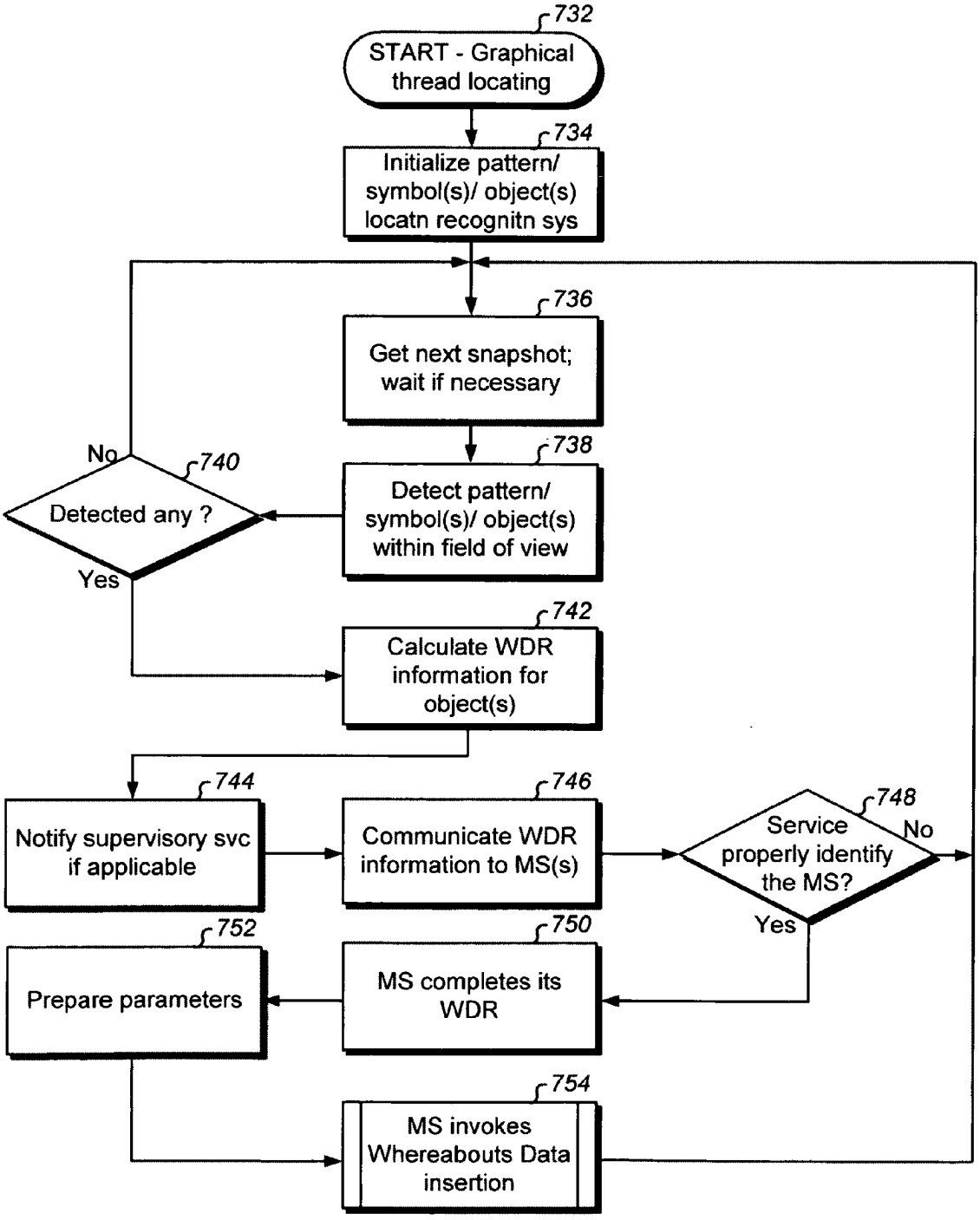

732
START - Graphical thread locating
734
Initialize pattern/ symbol(s)/ object(s) locatn recognitn sys
736
Get next snapshot; wait if necessary
738
Detect pattern/ symbol(s)/ object(s) within field of view
740
Detected any ?
No
Yes
742
Calculate WDR information for object(s)
744
Notify supervisory svc if applicable
746
Communicate WDR information to MS(s)
748
Service properly identify the MS?
No
Yes
750
MS completes its WDR
752
Prepare parameters
754
MS invokes Whereabouts Data insertion

Fig. 7D

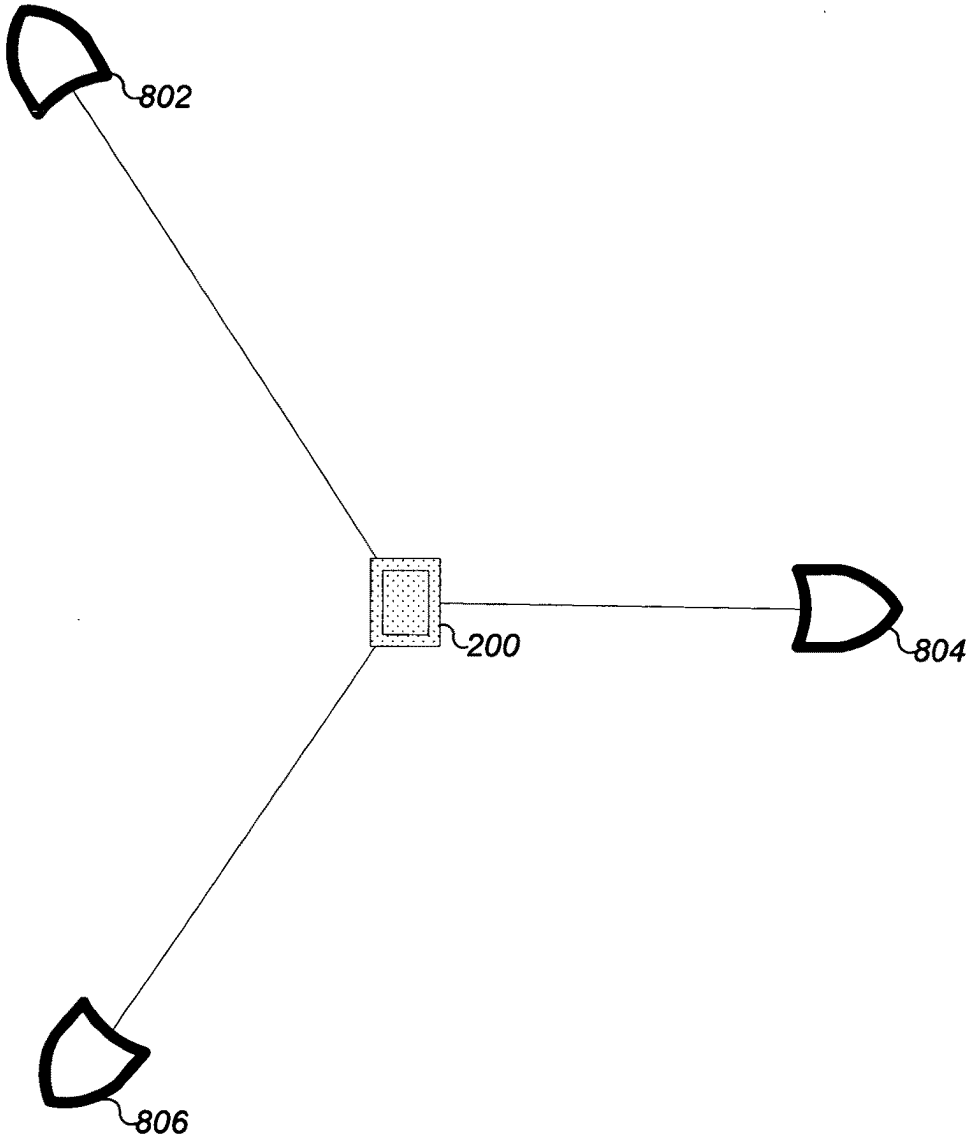


Fig. 8A

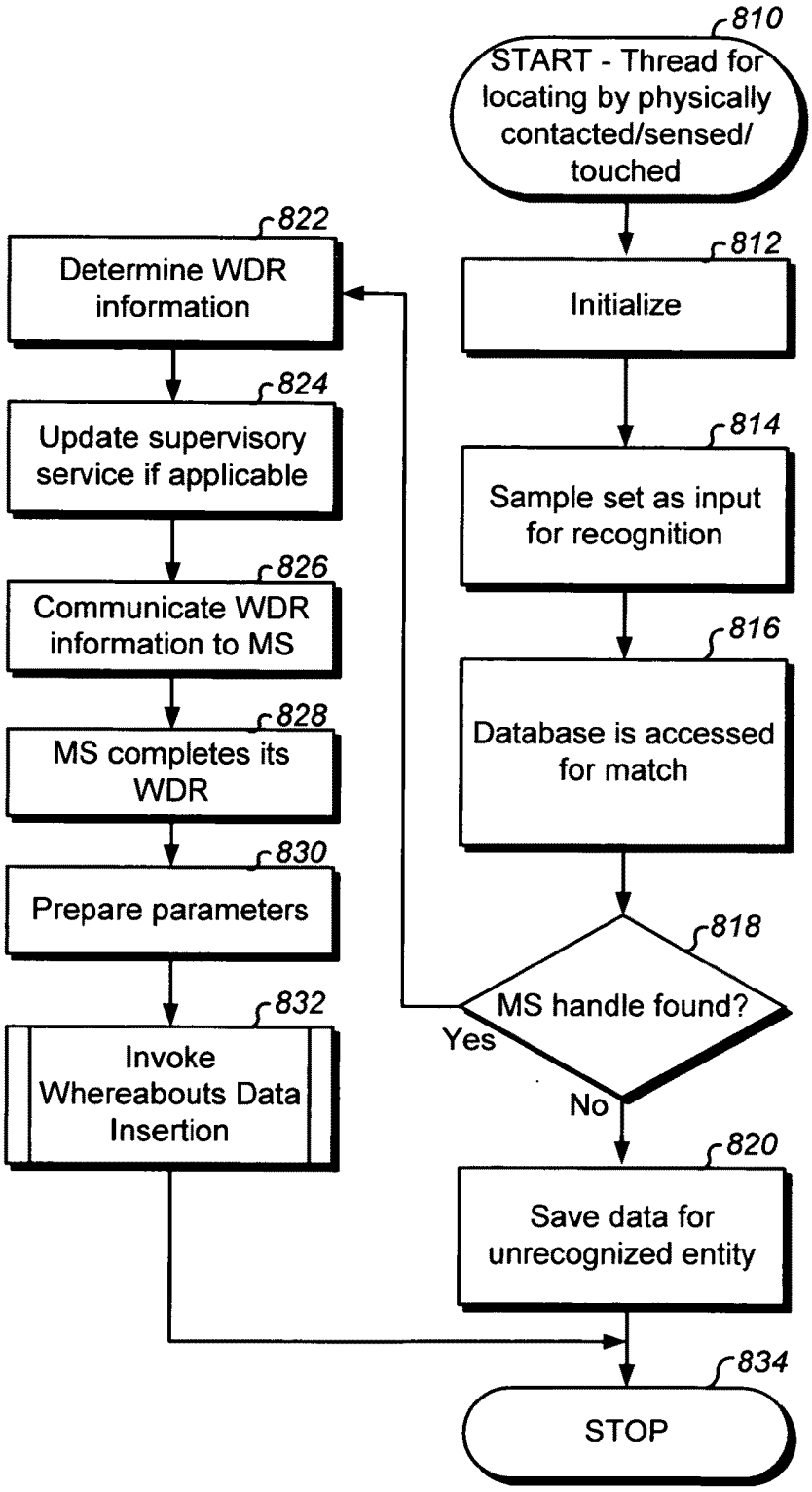

810
START - Thread for locating by physically contacted/sensed/ touched
812
Initialize
814
Sample set as input for recognition
816
Database is accessed for match
818
MS handle found?
Yes
No
820
Save data for unrecognized entity
822
Determine WDR information
824
Update supervisory service if applicable
826
Communicate WDR information to MS
828
MS completes its WDR
830
Prepare parameters
832
Invoke Whereabouts Data Insertion
834
STOP

Fig. 8B

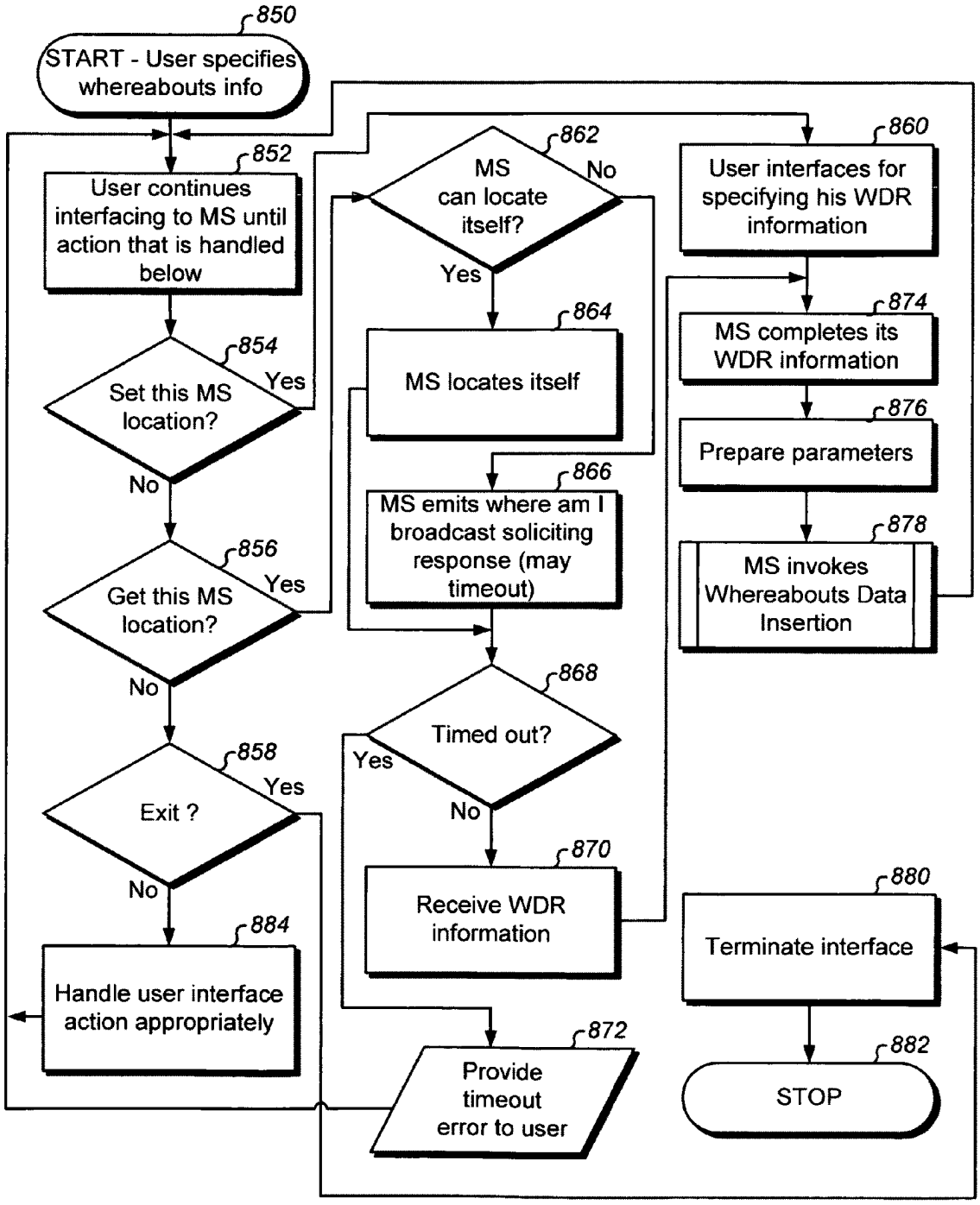

850
START - User specifies whereabouts info
852
User continues interfacing to MS until action that is handled below
854
Set this MS location?
Yes
No
856
Get this MS location?
Yes
No
858
Exit ?
Yes
No
884
Handle user interface action appropriately
862
MS can locate itself?
No
Yes
864
MS locates itself
866
MS emits where am I broadcast soliciting response (may timeout)
868
Timed out?
Yes
No
870
Receive WDR information
872
Provide timeout error to user
860
User interfaces for specifying his WDR information
874
MS completes its WDR information
876
Prepare parameters
878
MS invokes Whereabouts Data Insertion
880
Terminate interface
882
STOP

Fig. 8C

		MS (id 0A12:43EF:985B:012F)
GPS	C	x
	S	
A-GPS	C	
	S	
D-GPS	C	
	S	
Graphic-Pattern(s)	C	
	S	
Graphic-Distances	C	
	S	
Graphic-Triangulate	C	
	S	
Artificial Intelligence	C	
	S	
Cell Range	C	
	S	x
Cell AOA	C	
	S	
Cell TDOA	C	
	S	x
Cell MPT	C	
	S	x
Antenna Range	C	
	S	x
Antenna AOA	C	
	S	x
Antenna TDOA	C	
	S	x
Antenna MPT	C	
	S	x
LIDAR/optics	C	
	S	
Manual	C	
	S	
Contact	C	
	S	x
MPT	C	
	S	x
Client Logical Connect	C	
	S	
Server Logical Connect	C	
	S	
Client Physical Connect	C	
	S	
Server Physical Connect	C	
	S	
Sound/Acoustics	C	
	S	
Microdot/ RFI	C	
	S	
Transponder	C	
	S	
Others	C	
	S	
. . .	C	
	S	

Fig. 9A

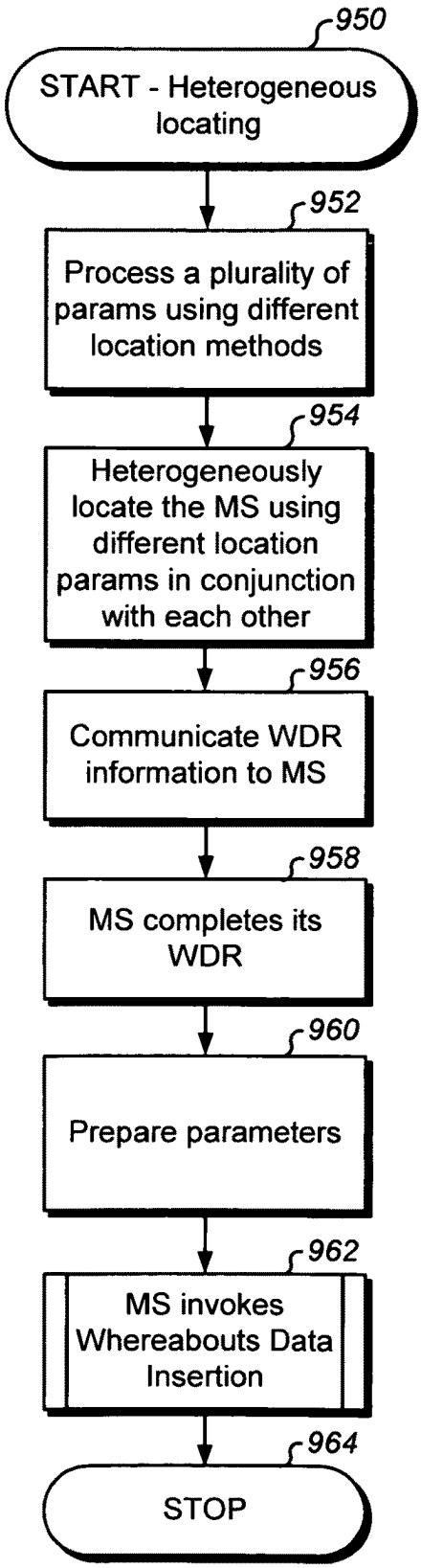

950
START - Heterogeneous locating
952
Process a plurality of params using different location methods
954
Heterogeneously locate the MS using different location params in conjunction with each other
956
Communicate WDR information to MS
958
MS completes its WDR
960
Prepare parameters
962
MS invokes Whereabouts Data Insertion
964
STOP

Fig. 9B

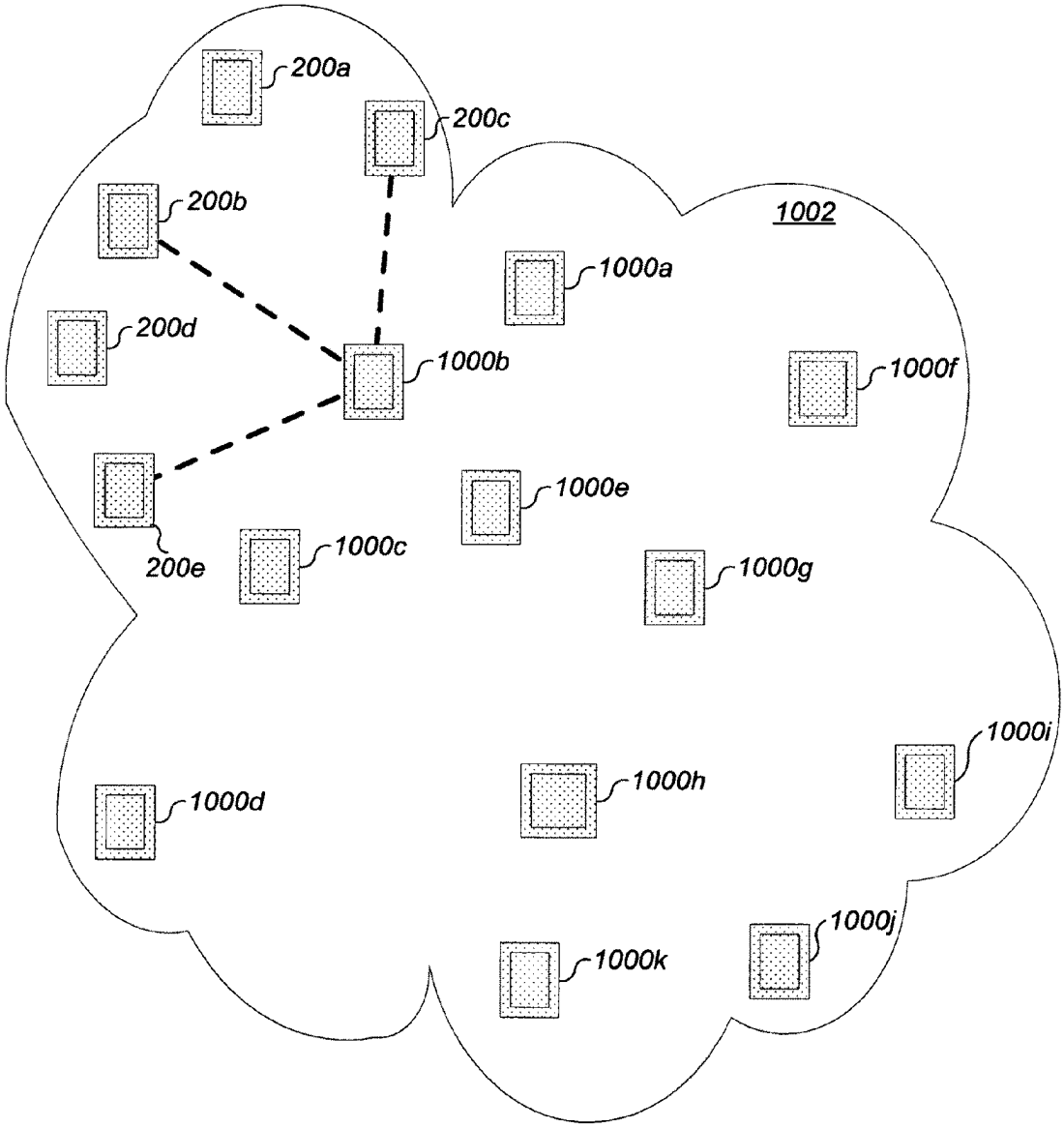

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000d
1000h
1000j
1000k

Fig. 10A

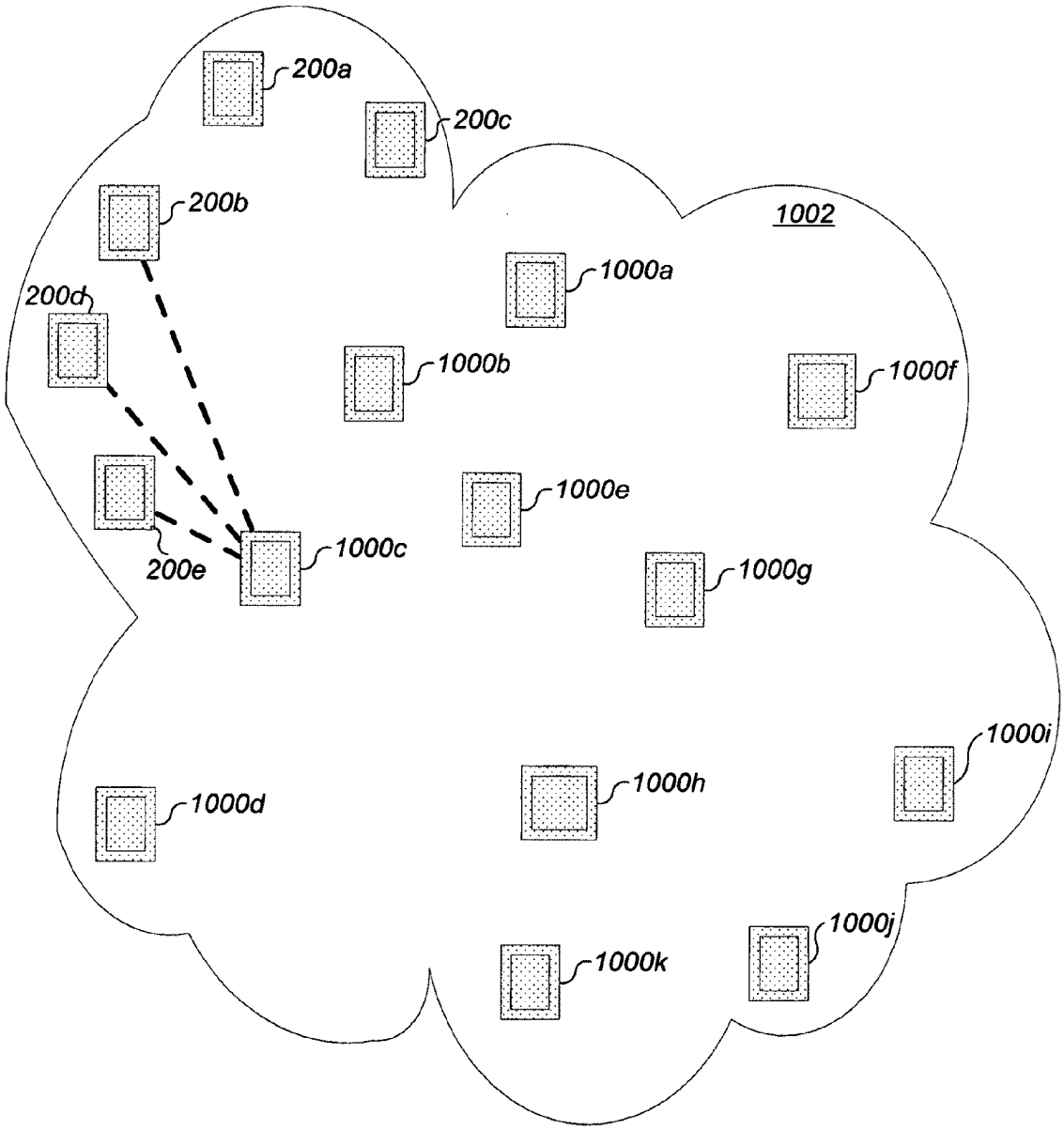

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10B

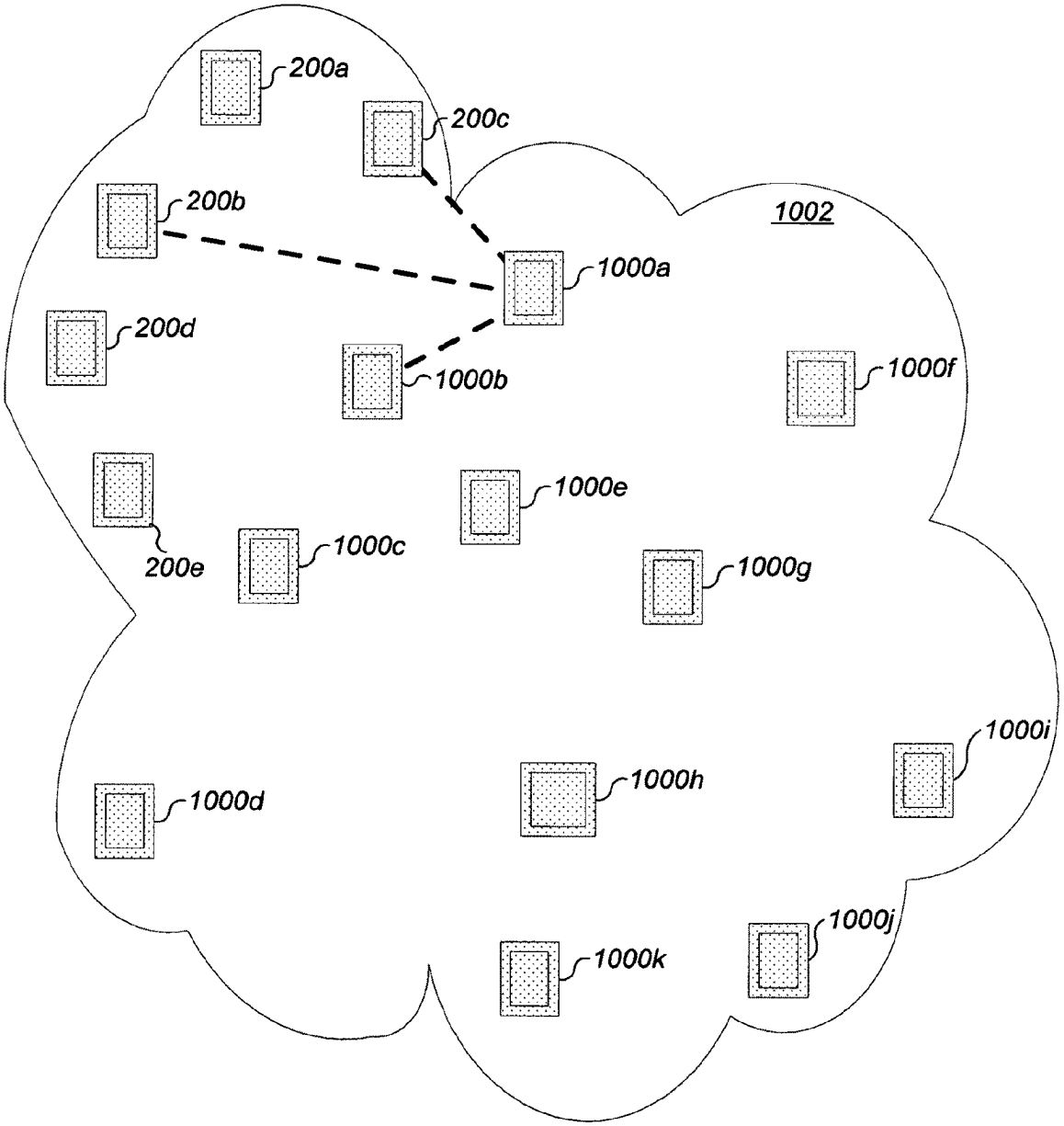

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10C

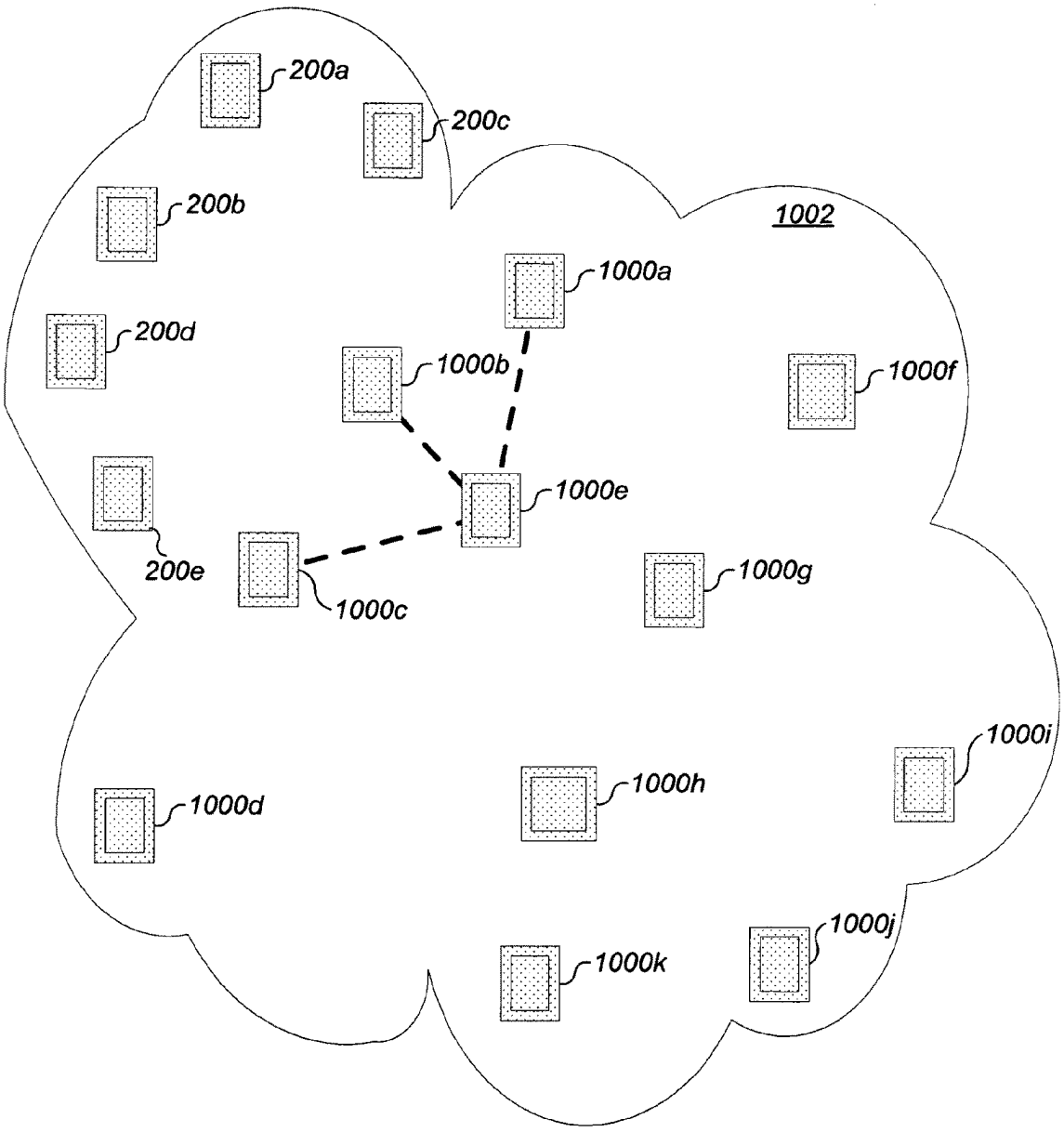

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10D

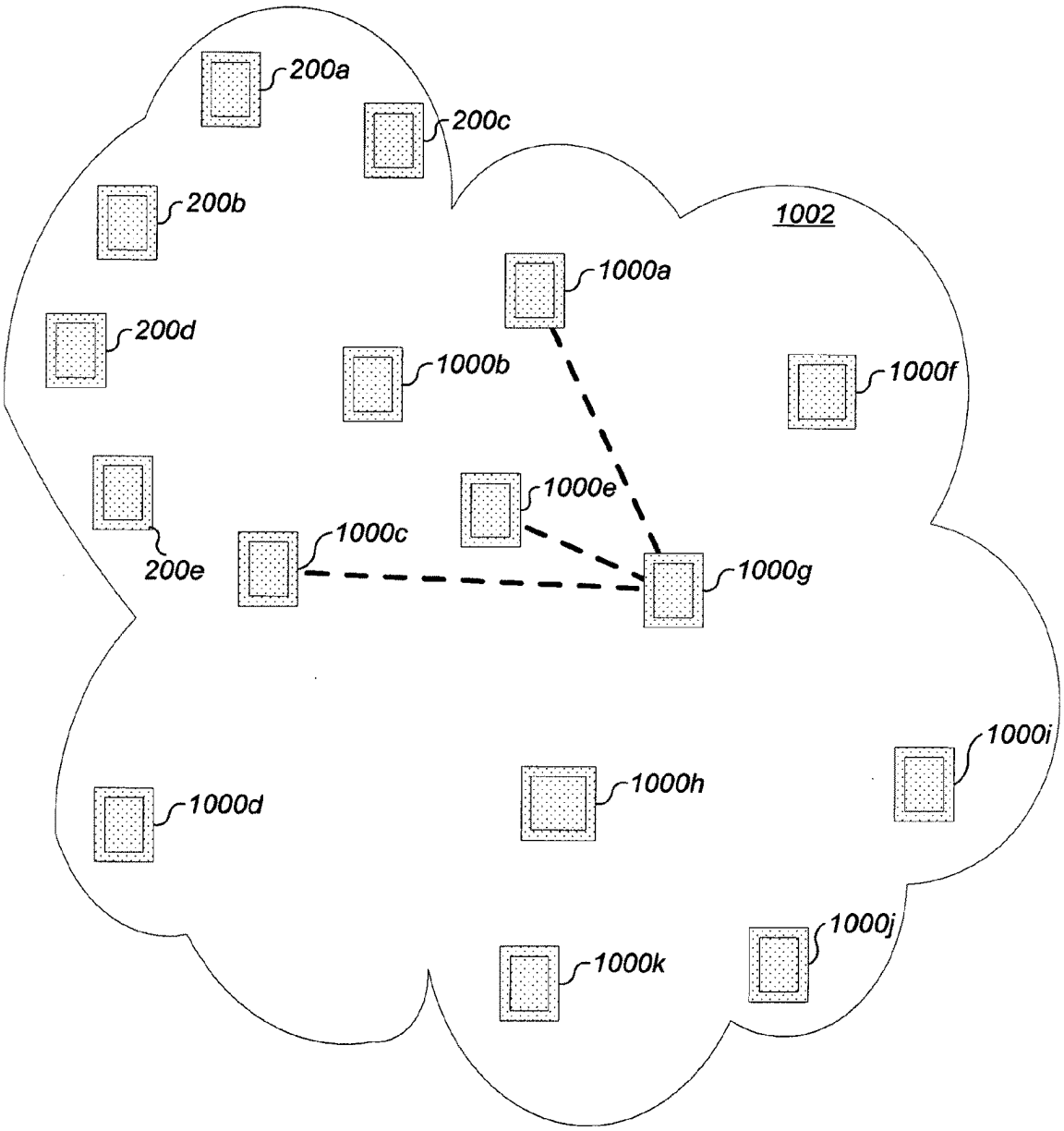

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
1000c
200e
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10E

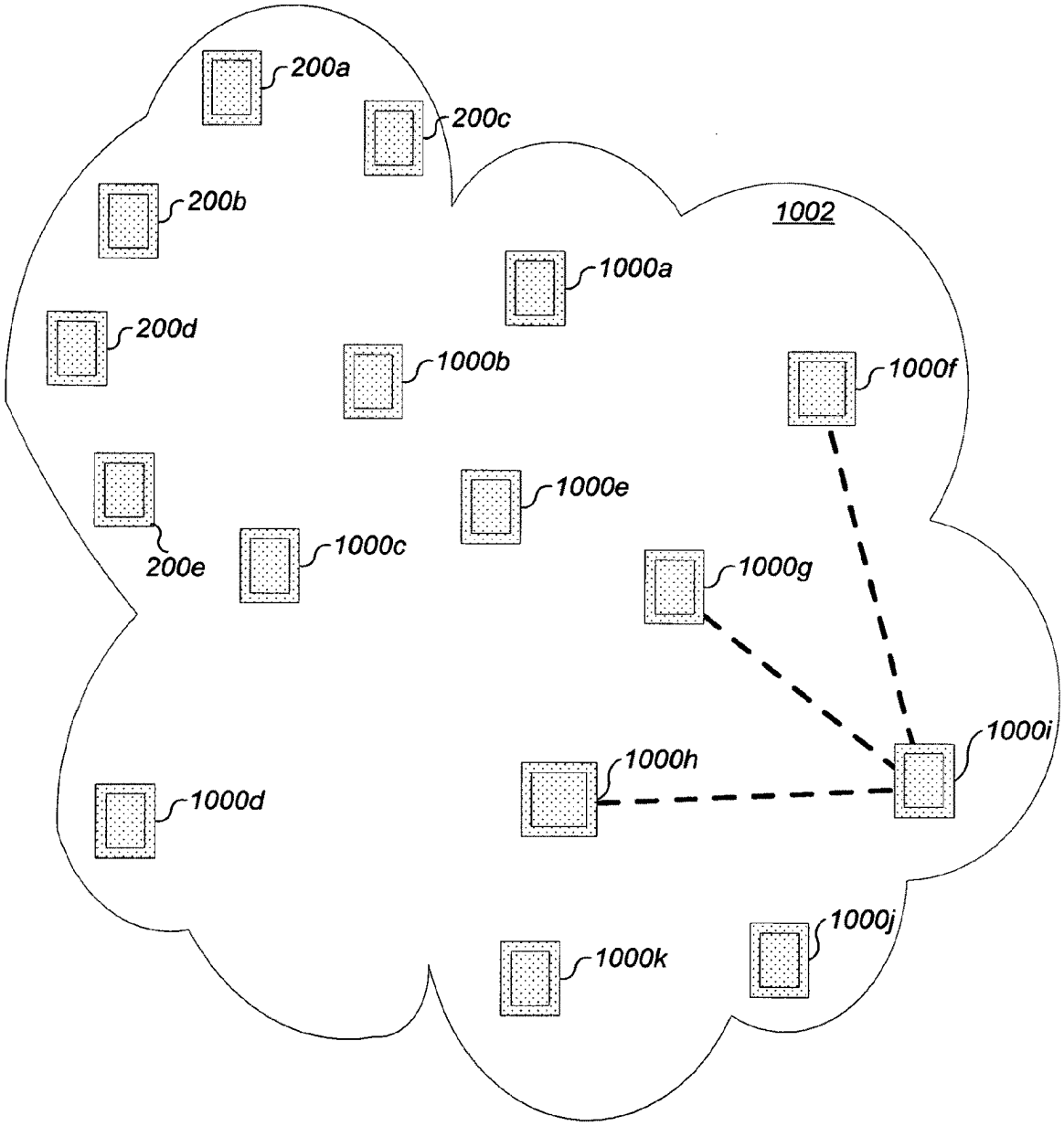

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10F

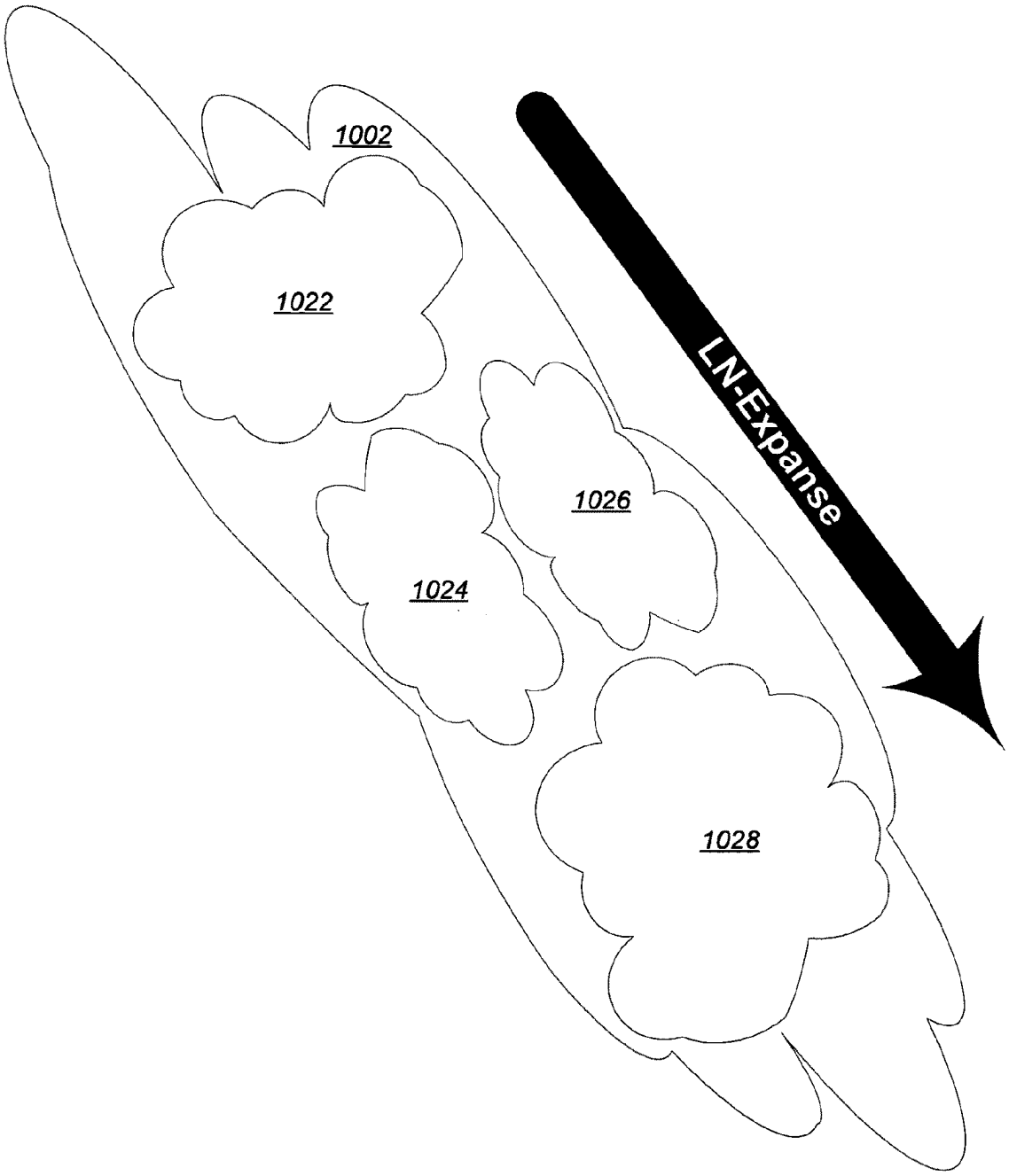


Fig. 10G

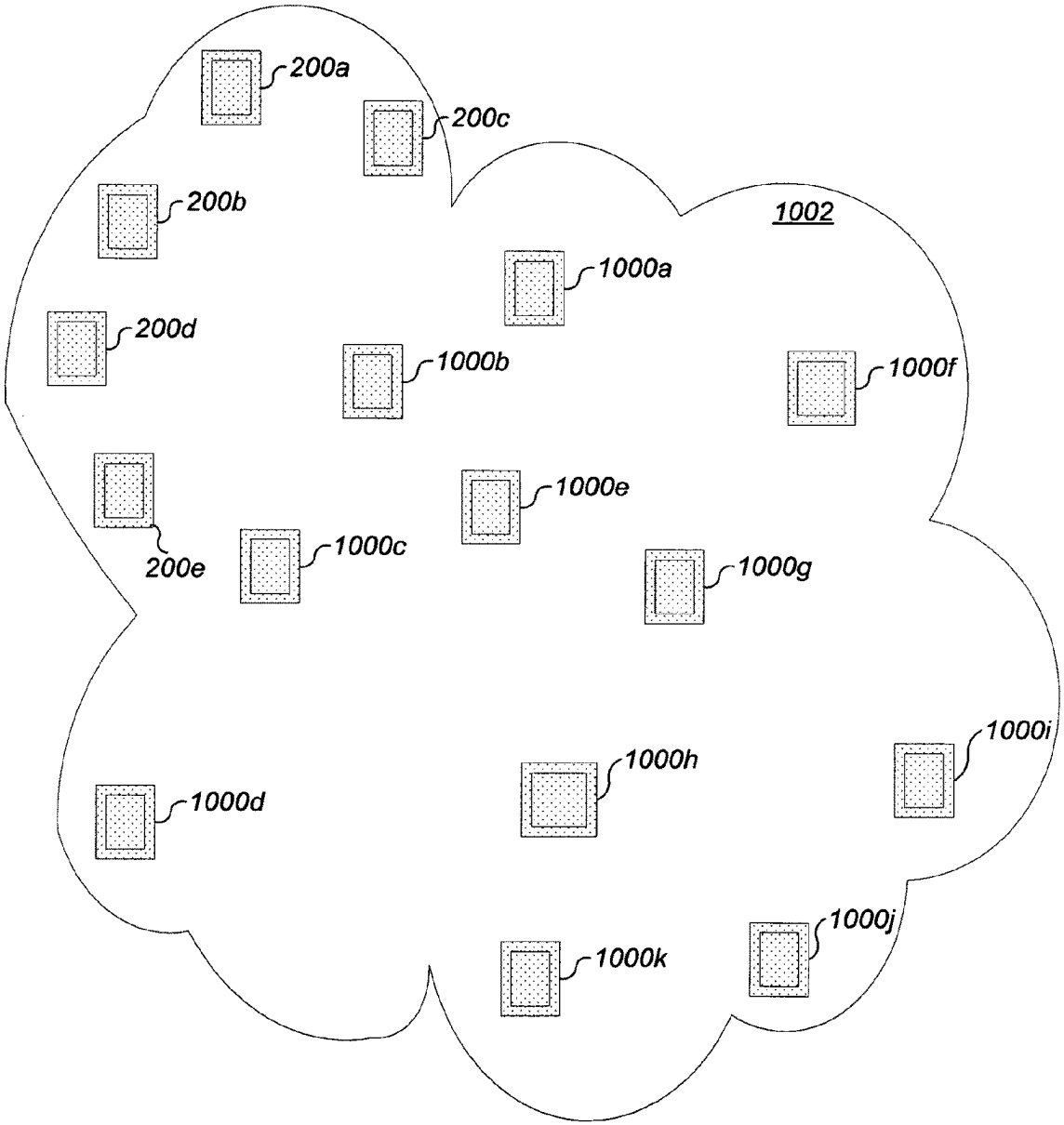

200a
200c
200b
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10H

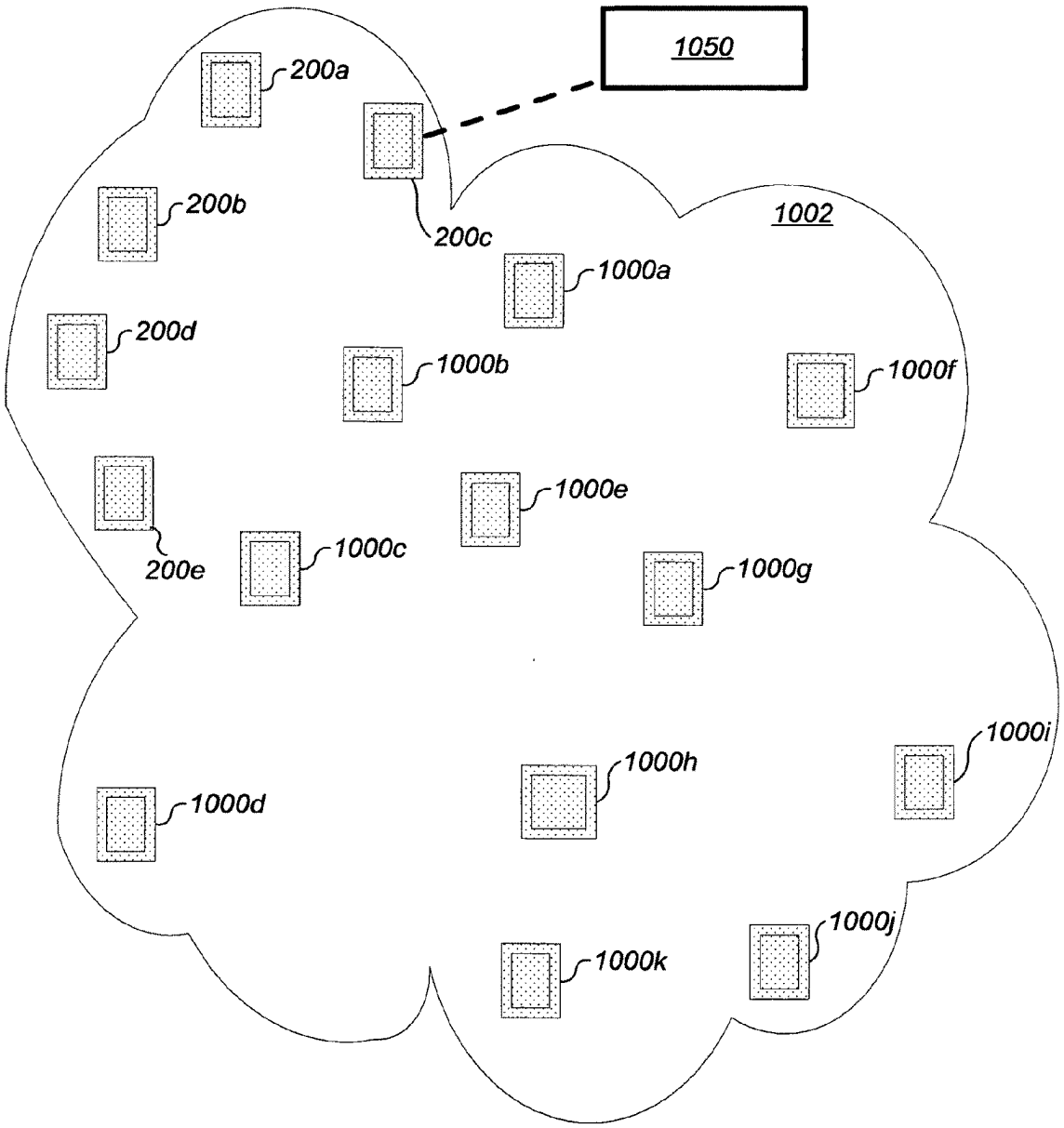

1050
200a
200b
200c
1002
1000a
200d
1000b
1000f
1000e
200e
1000c
1000g
1000i
1000h
1000d
1000j
1000k

Fig. 10I

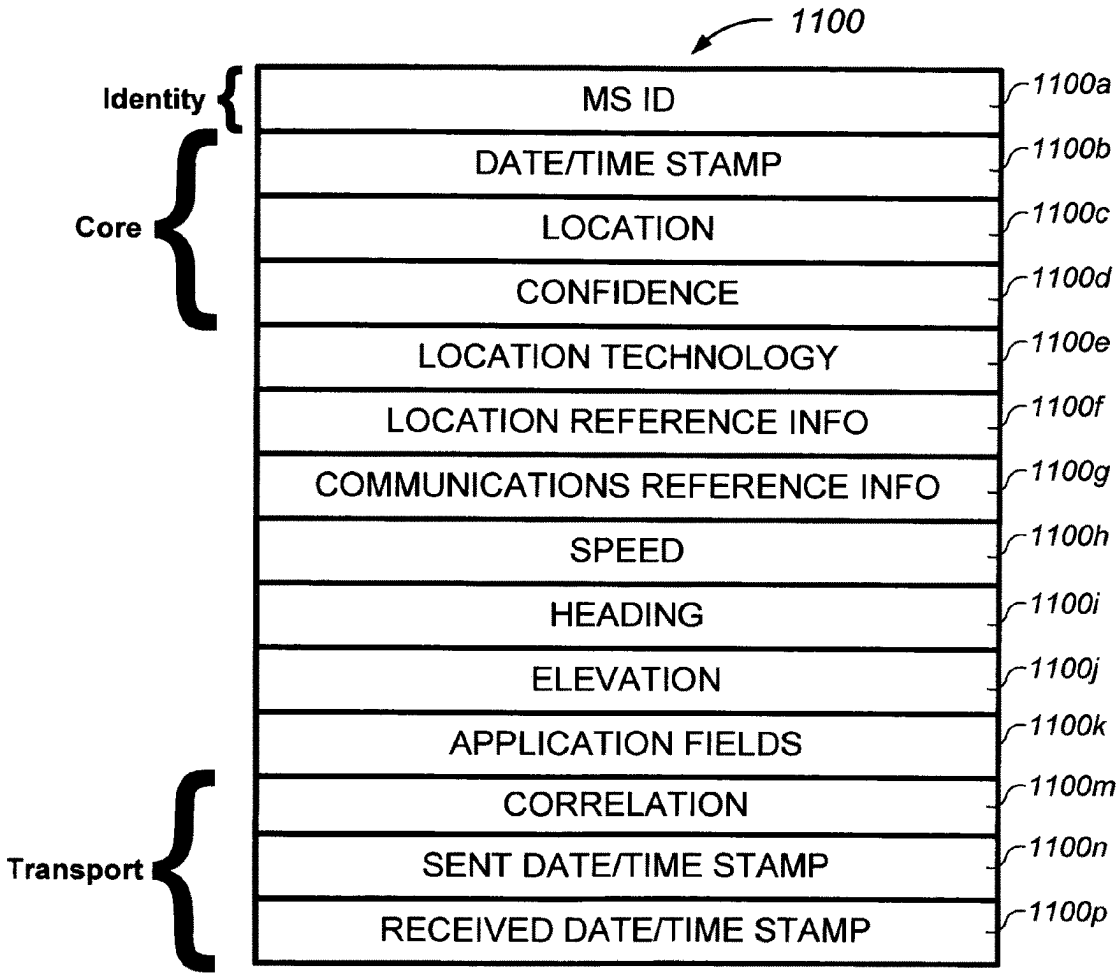

1100
Identity
Core
Transport
MS ID
DATE/TIME STAMP
LOCATION
CONFIDENCE
LOCATION TECHNOLOGY
LOCATION REFERENCE INFO
COMMUNICATIONS REFERENCE INFO
SPEED
HEADING
ELEVATION
APPLICATION FIELDS
CORRELATION
SENT DATE/TIME STAMP
RECEIVED DATE/TIME STAMP
1100a
1100b
1100c
1100d
1100e
1100f
1100g
1100h
1100i
1100j
1100k
1100m
1100n
1100p

Fig. 11A

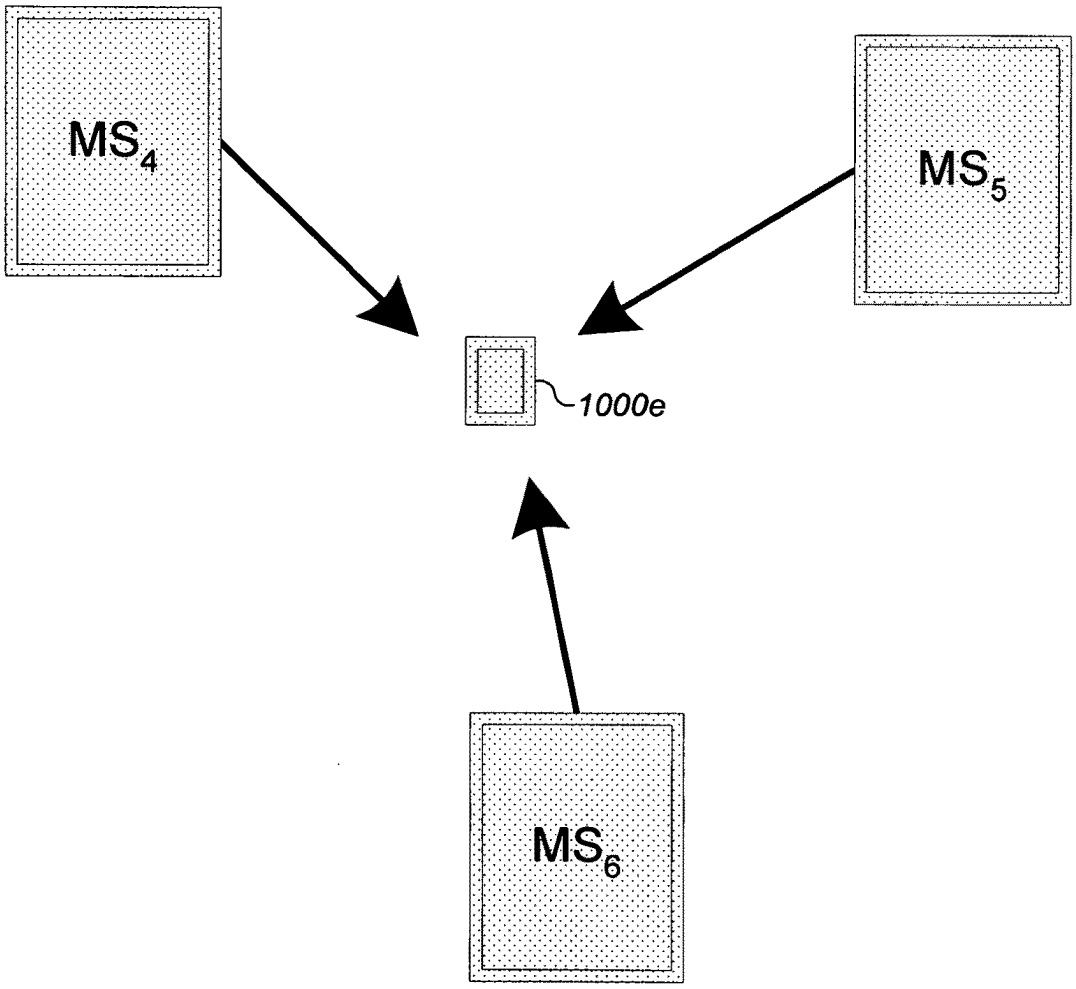


Fig. 11B

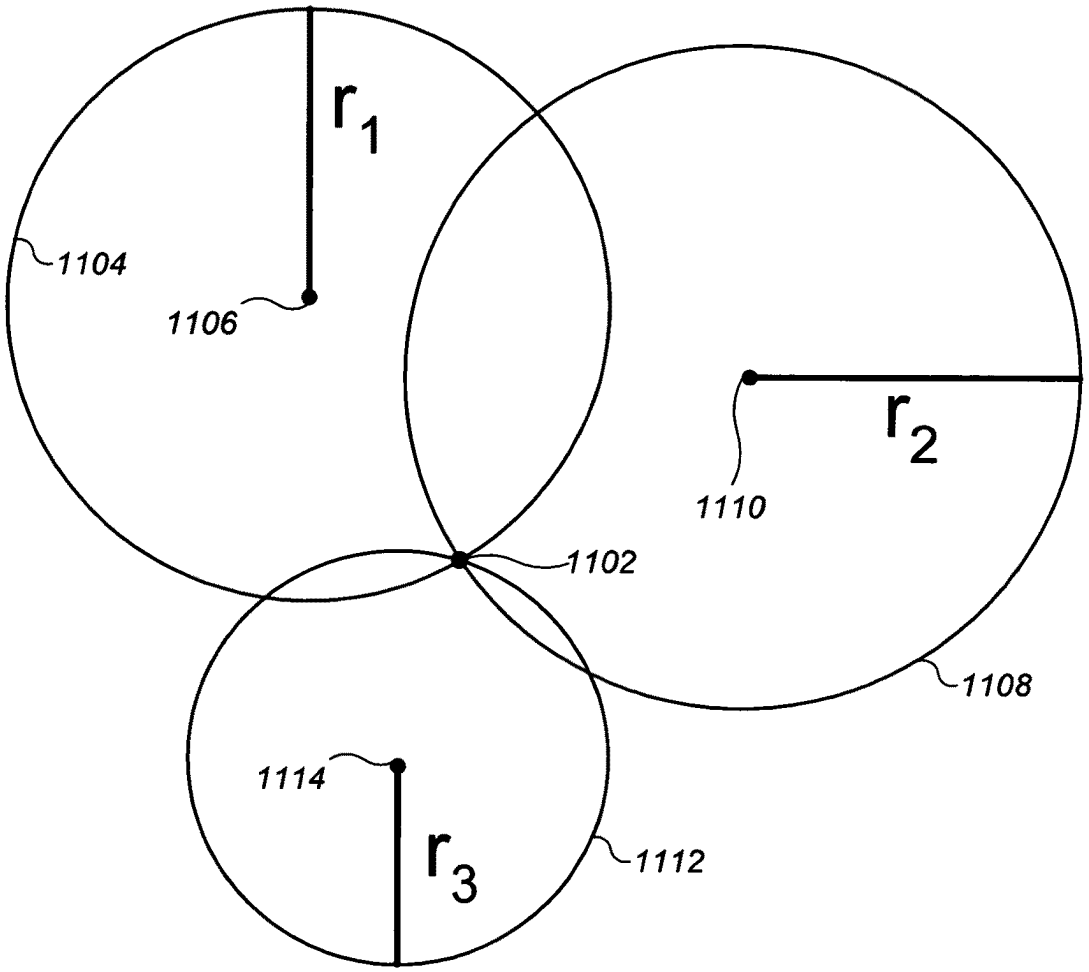


Fig. 11C

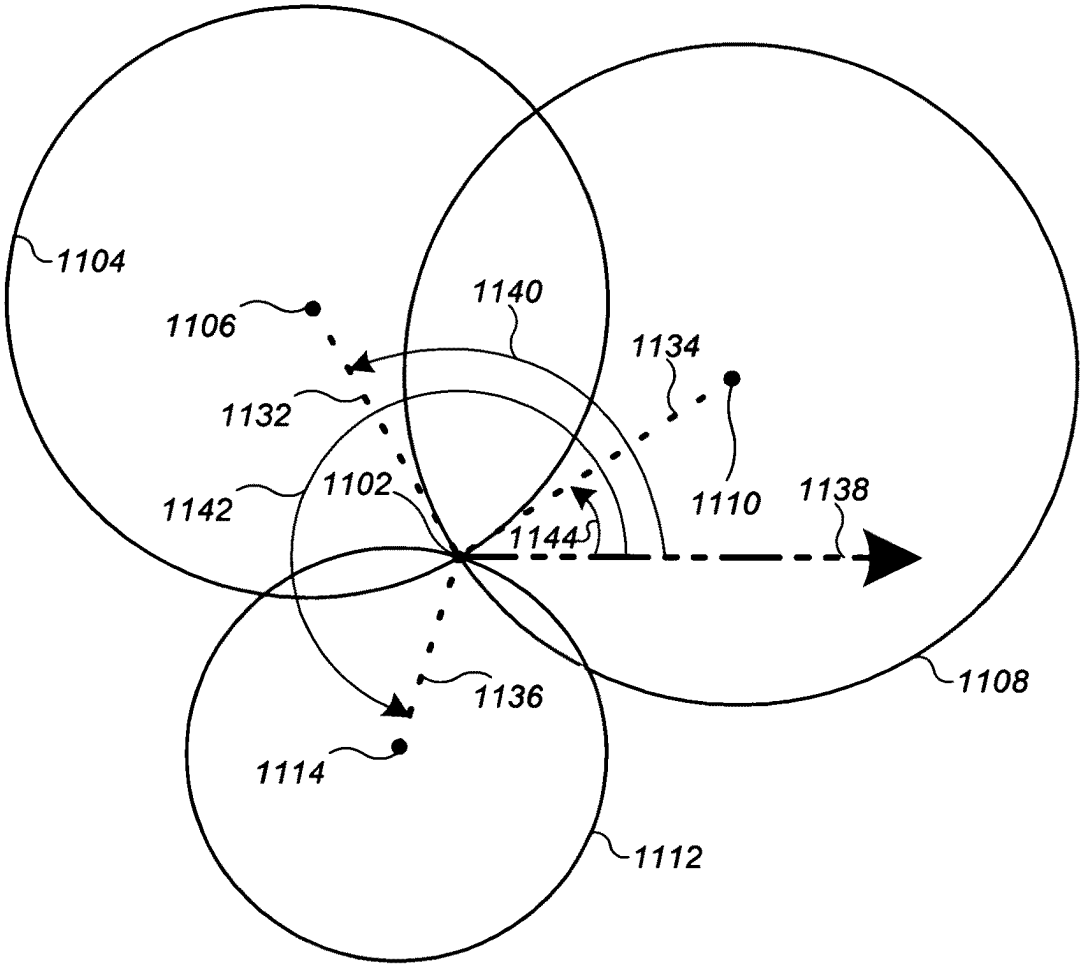

1104
1106
1140
1134
1132
1142
1102
1110
1138
1144
1136
1108
1114
1112

Fig. 11D

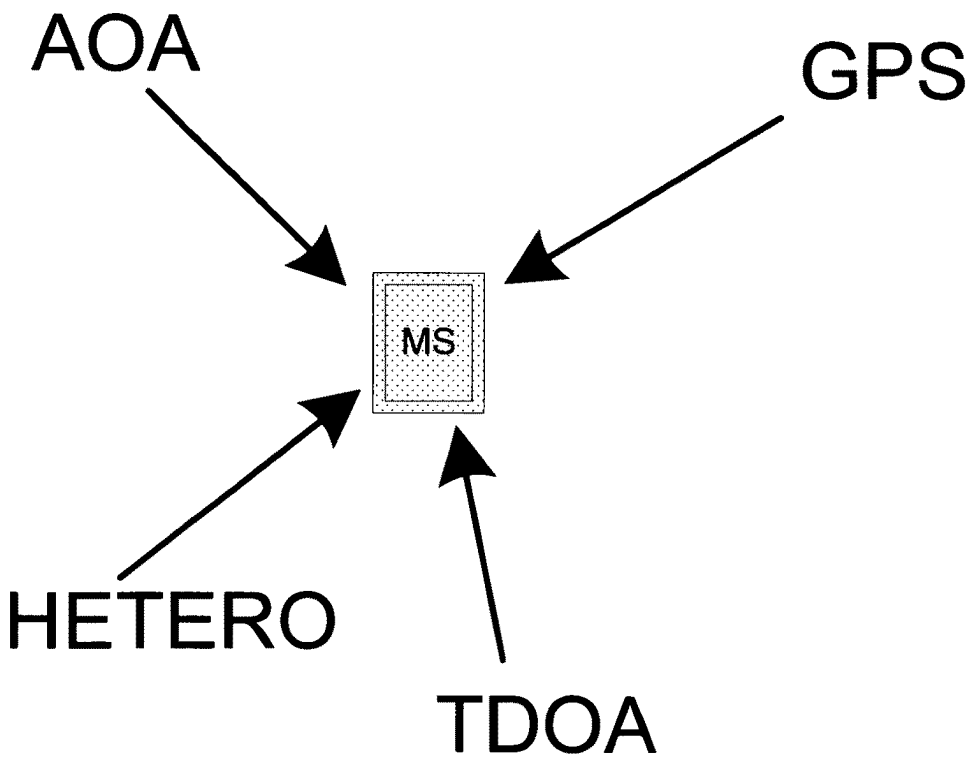


Fig. 11E

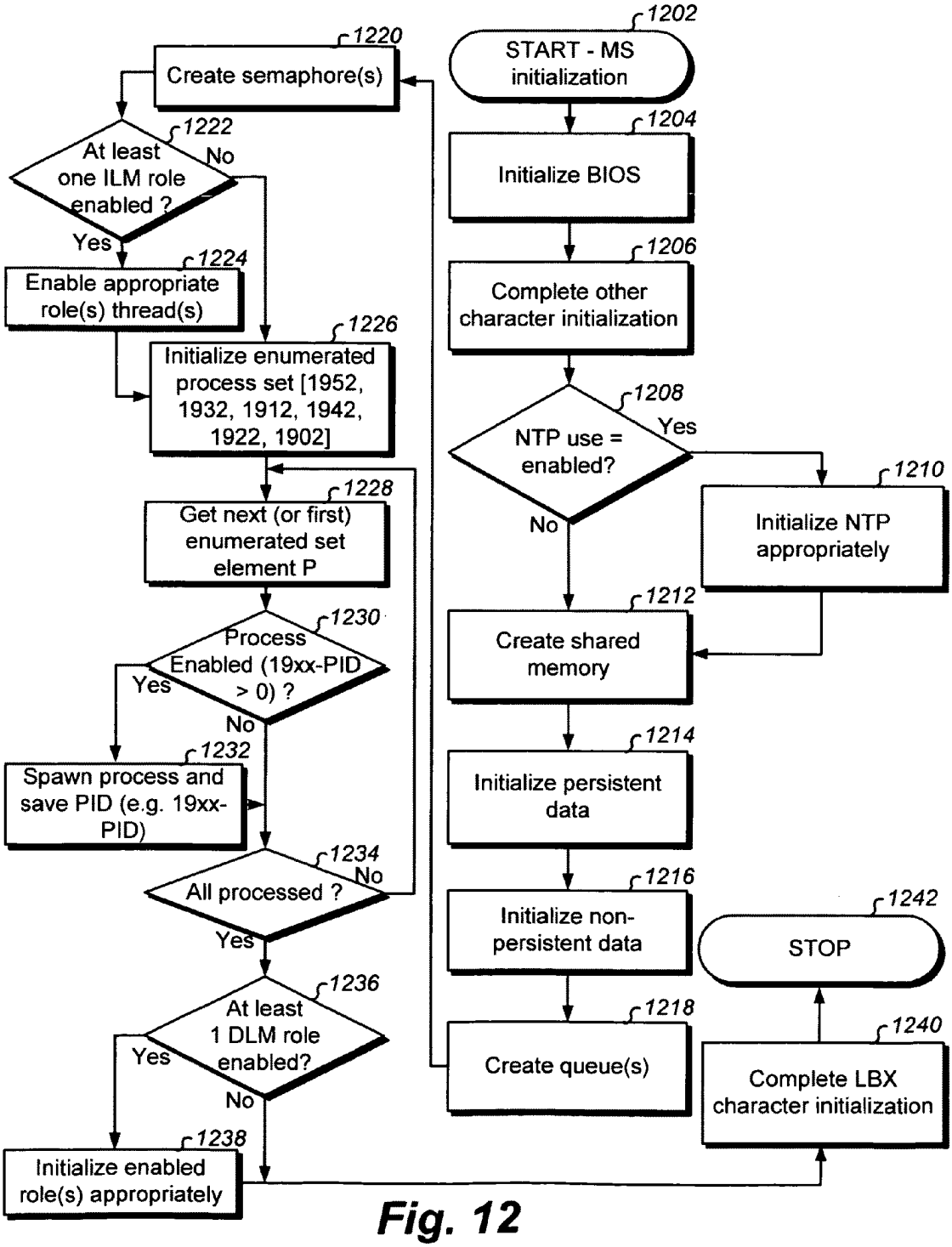

1202
START - MS initialization
1204
Initialize BIOS
1206
Complete other character initialization
1208
NTP use = enabled?
Yes
No
1210
Initialize NTP appropriately
1212
Create shared memory
1214
Initialize persistent data
1216
Initialize non-persistent data
1218
Create queue(s)
1220
Create semaphore(s)
1222
At least one ILM role enabled ?
No
Yes
1224
Enable appropriate role(s) thread(s)
1226
Initialize enumerated process set [1952, 1932, 1912, 1942, 1922, 1902]
1228
Get next (or first) enumerated set element P
1230
Process Enabled (19xx-PID > 0) ?
Yes
No
1232
Spawn process and save PID (e.g. 19xx-PID)
1234
All processed ?
No
Yes
1236
At least 1 DLM role enabled?
Yes
No
1238
Initialize enabled role(s) appropriately
1240
Complete LBX character initialization
1242
STOP

Fig. 12

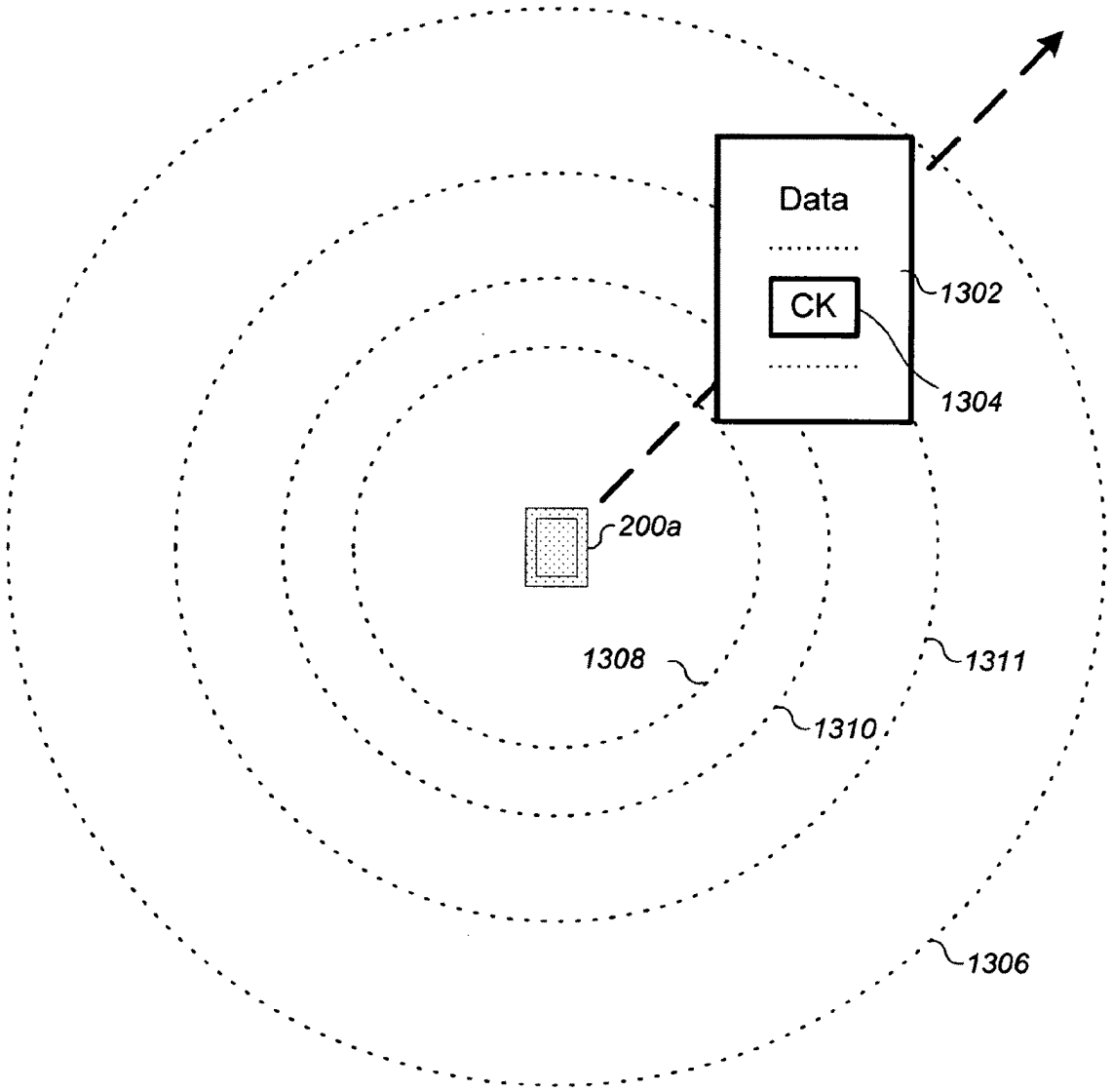


Fig. 13A

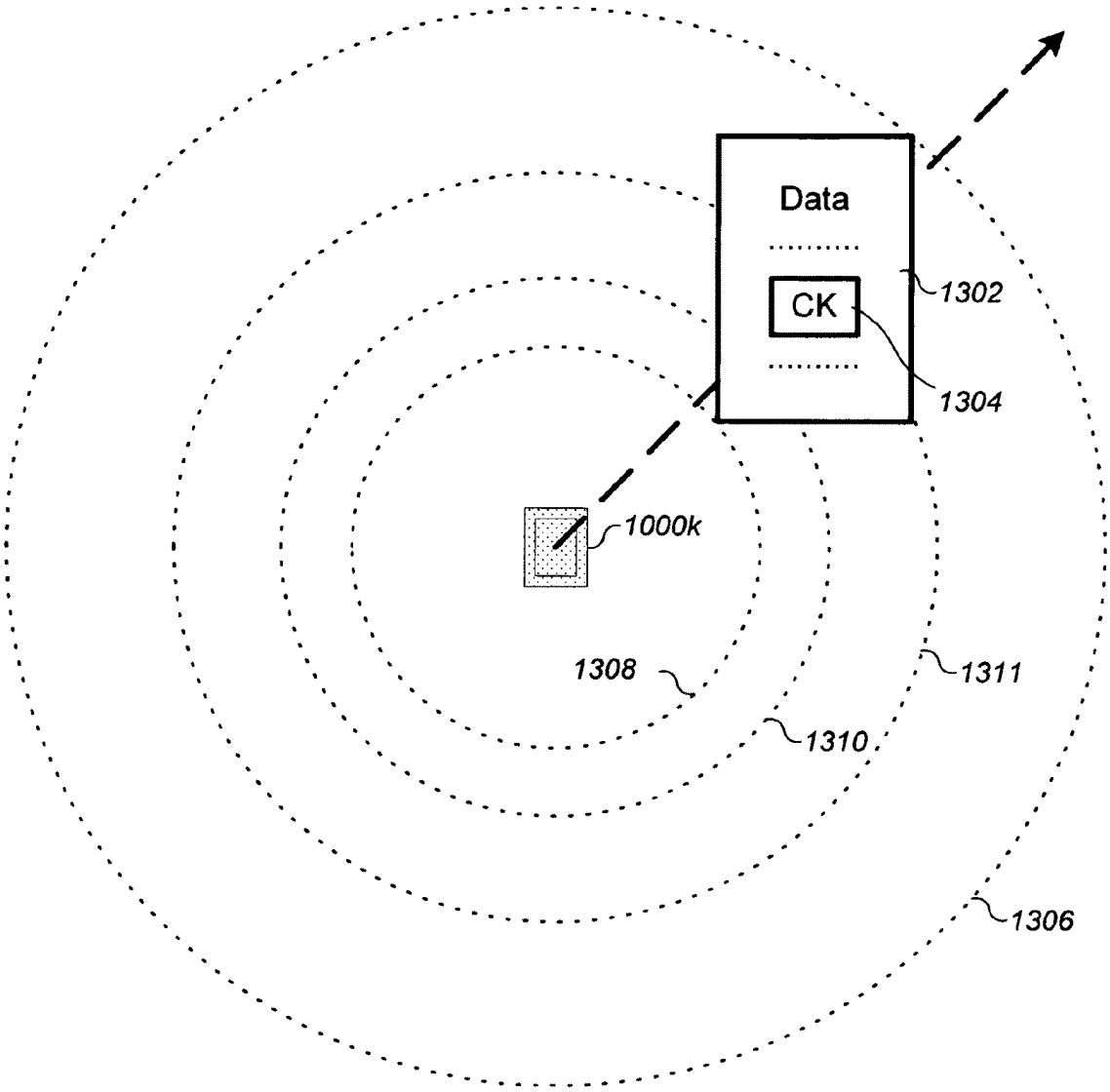


Fig. 13B

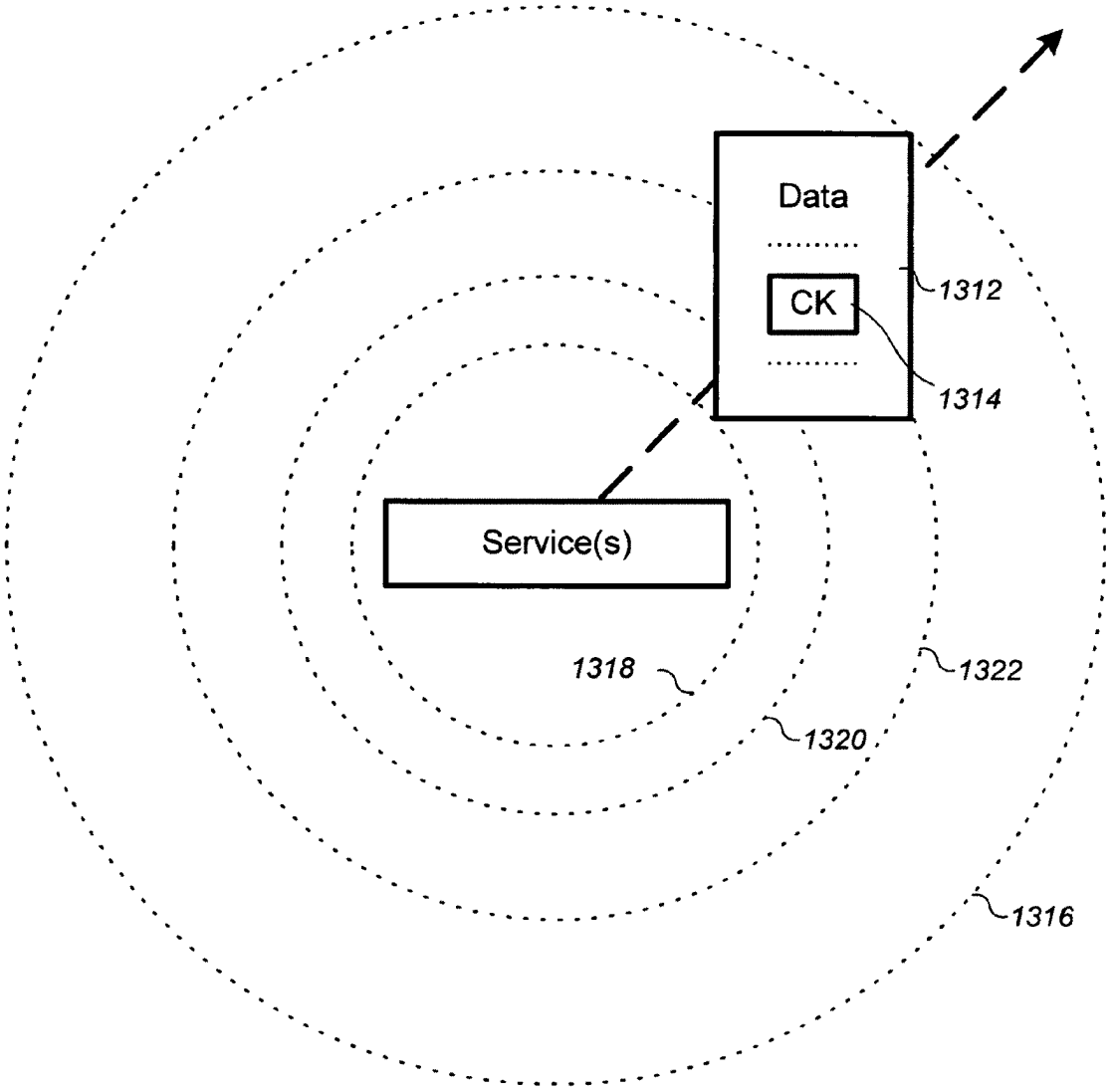


Fig. 13C

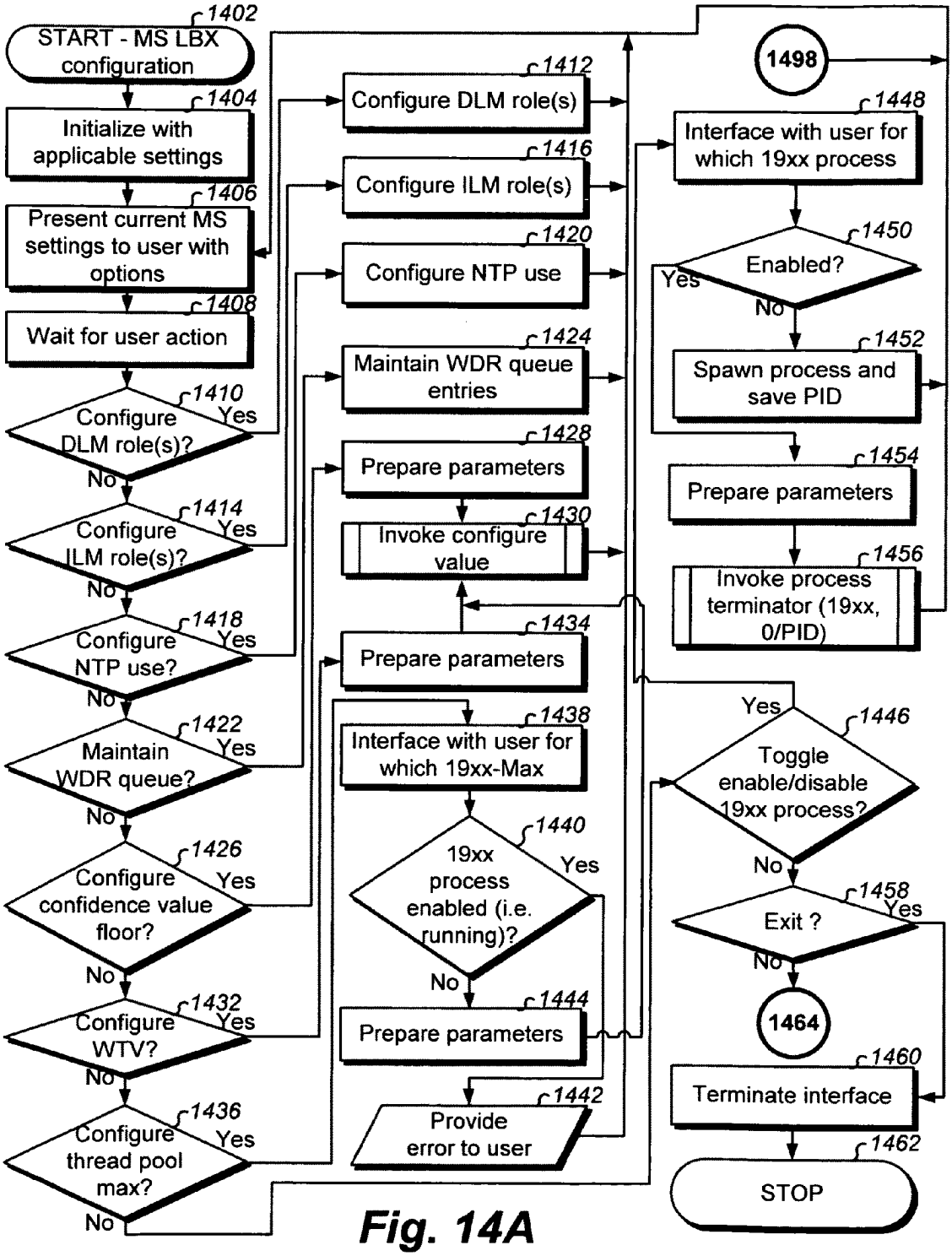

1402
START - MS LBX configuration
1404
Initialize with applicable settings
1406
Present current MS settings to user with options
1408
Wait for user action
1410
Configure DLM role(s)?
Yes
No
1414
Configure ILM role(s)?
Yes
No
1418
Configure NTP use?
Yes
No
1422
Maintain WDR queue?
Yes
No
1426
Configure confidence value floor?
Yes
No
1432
Configure WTV?
Yes
No
1436
Configure thread pool max?
Yes
No
1412
Configure DLM role(s)
1416
Configure ILM role(s)
1420
Configure NTP use
1424
Maintain WDR queue entries
1428
Prepare parameters
1430
Invoke configure value
1434
Prepare parameters
1438
Interface with user for which 19xx-Max
1440
19xx process enabled (i.e. running)?
Yes
No
1444
Prepare parameters
1442
Provide error to user
1498
1448
Interface with user for which 19xx process
1450
Enabled?
Yes
No
1452
Spawn process and save PID
1454
Prepare parameters
1456
Invoke process terminator (19xx, 0/PID)
Yes
1446
Toggle enable/disable 19xx process?
No
1458
Exit ?
Yes
No
1464
1460
Terminate interface
1462
STOP

Fig. 14A

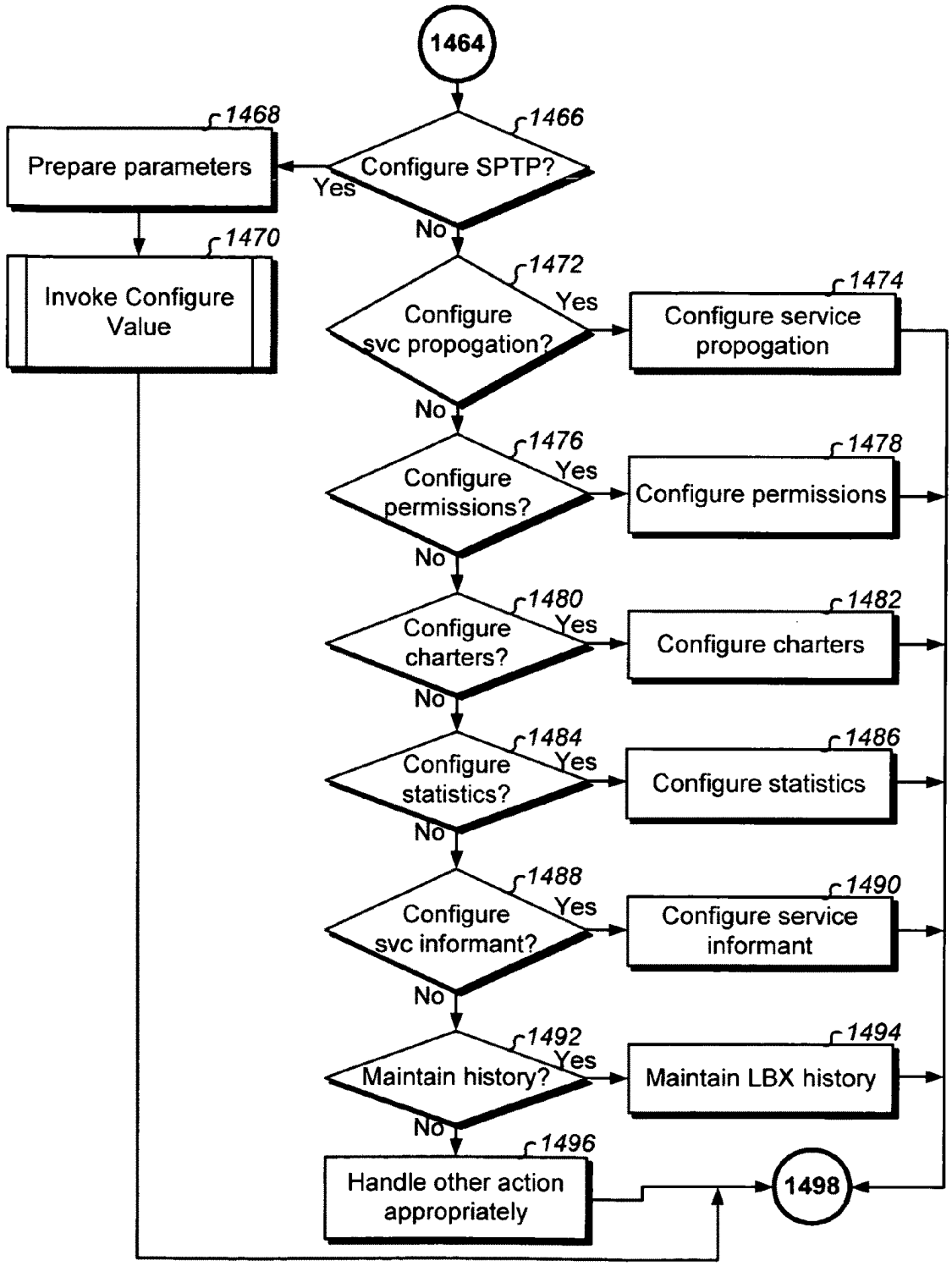

1464
1466
Configure SPTP?
Yes
1468
Prepare parameters
No
1470
Invoke Configure Value
1472
Configure svc propogation?
Yes
1474
Configure service propogation
No
1476
Configure permissions?
Yes
1478
Configure permissions
No
1480
Configure charters?
Yes
1482
Configure charters
No
1484
Configure statistics?
Yes
1486
Configure statistics
No
1488
Configure svc informant?
Yes
1490
Configure service informant
No
1492
Maintain history?
Yes
1494
Maintain LBX history
No
1496
Handle other action appropriately
1498

Fig. 14B

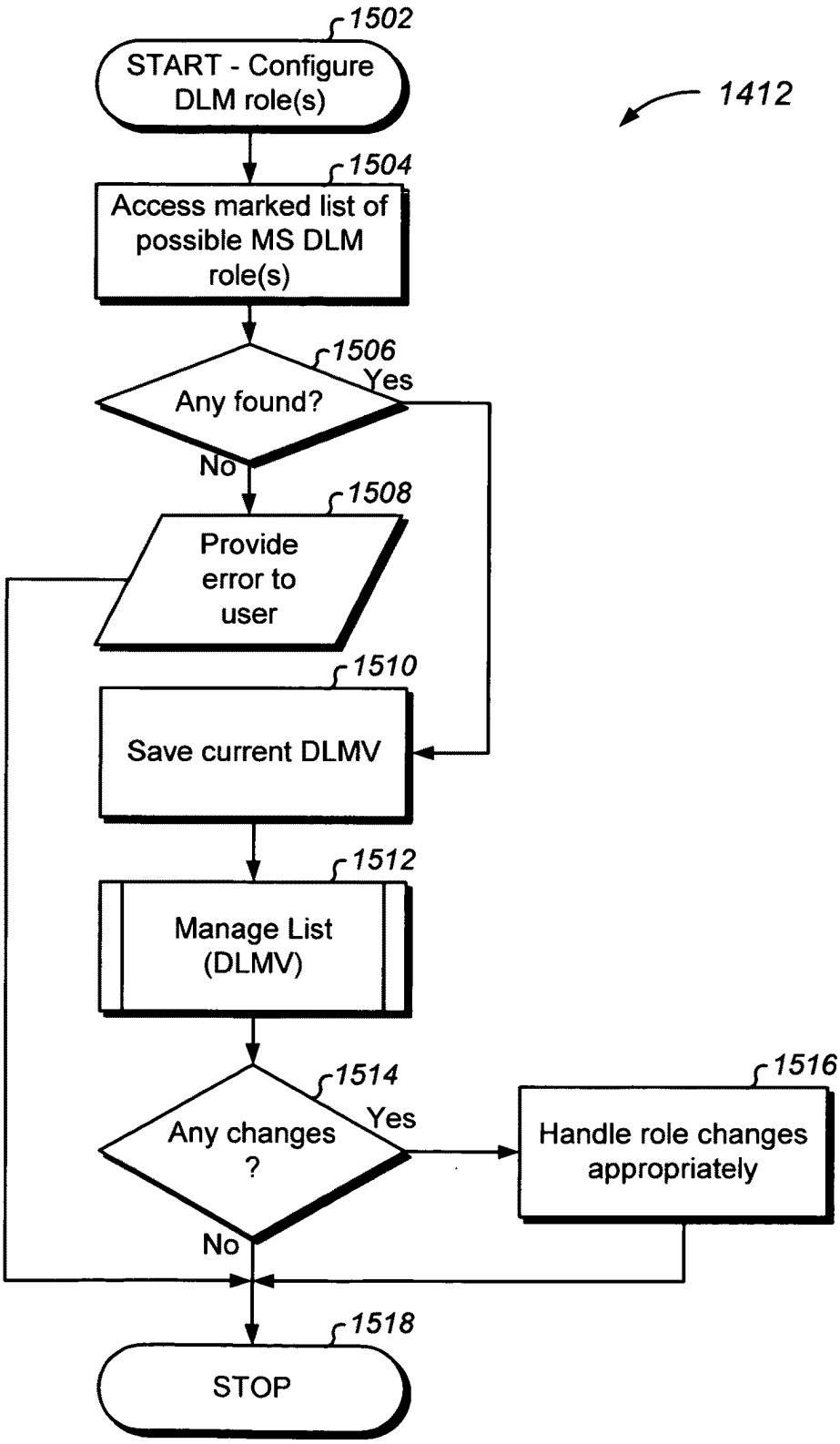

1502
START - Configure DLM role(s)
1412
1504
Access marked list of possible MS DLM role(s)
1506
Any found?
Yes
No
1508
Provide error to user
1510
Save current DLMV
1512
Manage List (DLMV)
1514
Any changes ?
Yes
No
1516
Handle role changes appropriately
1518
STOP

Fig. 15A

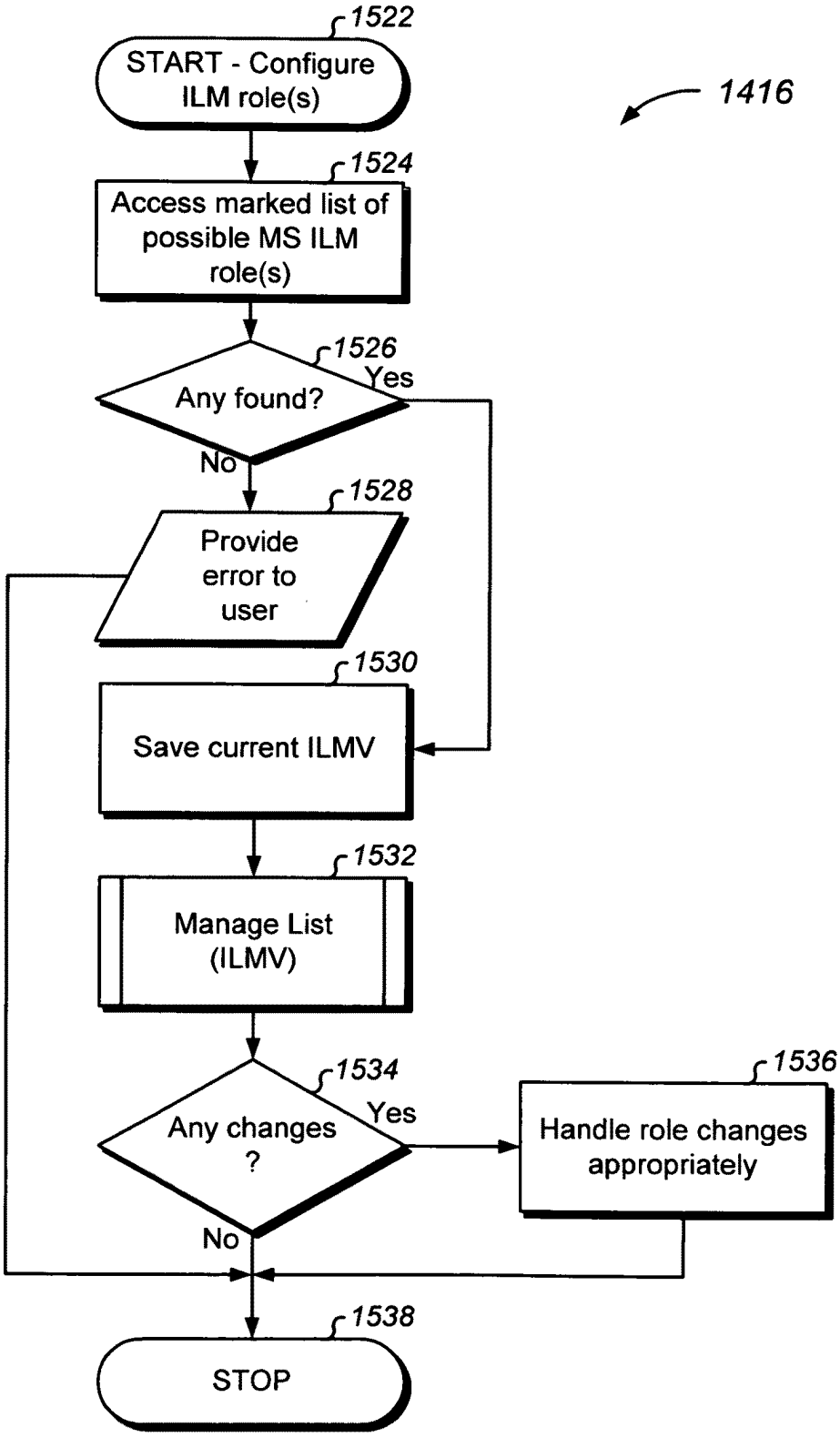

1522
START - Configure ILM role(s)
1416
1524
Access marked list of possible MS ILM role(s)
1526
Any found?
Yes
No
1528
Provide error to user
1530
Save current ILMV
1532
Manage List (ILMV)
1534
Any changes ?
Yes
No
1536
Handle role changes appropriately
1538
STOP

Fig. 15B

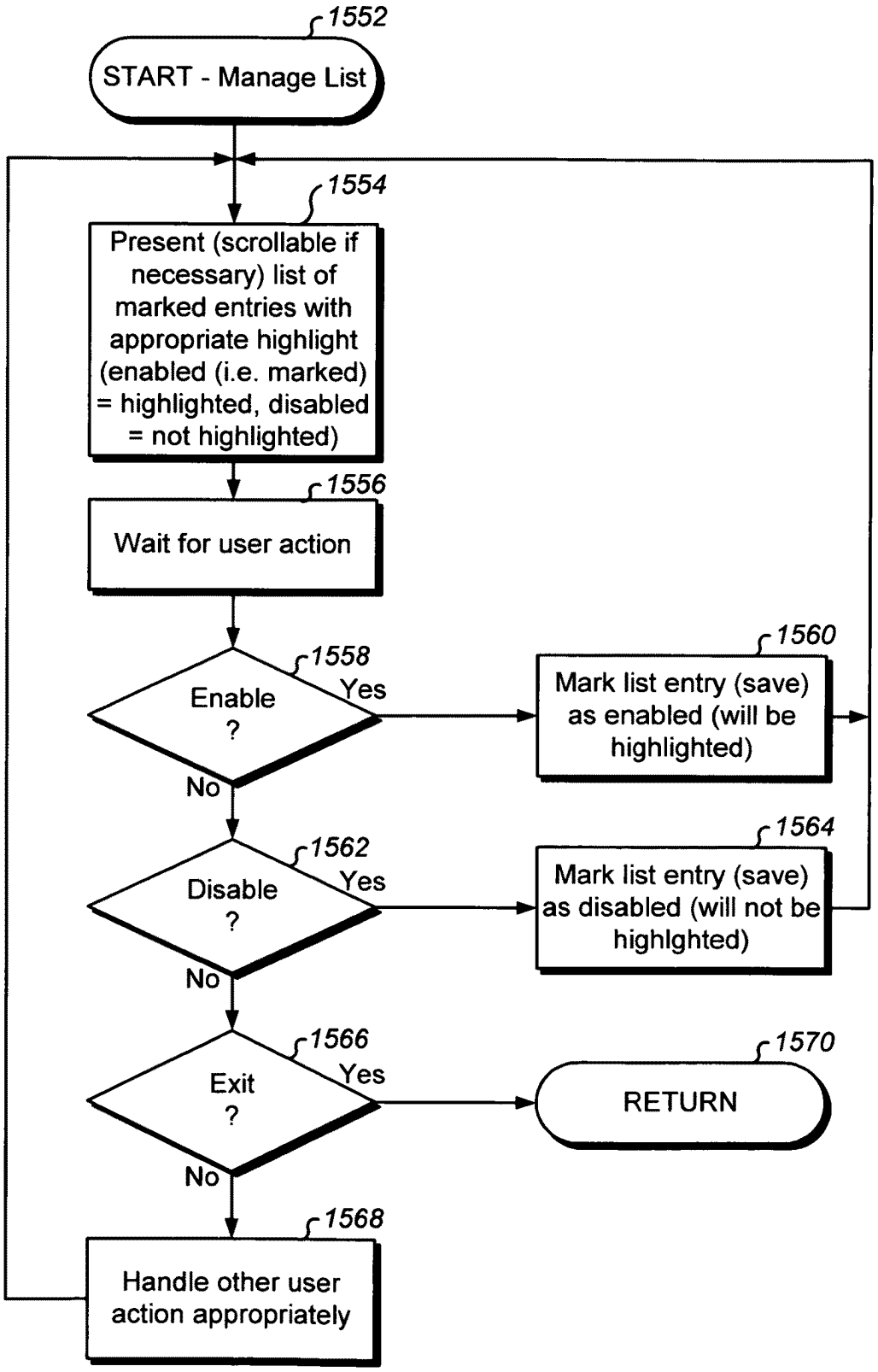

1552
START - Manage List
1554
Present (scrollable if necessary) list of marked entries with appropriate highlight (enabled (i.e. marked) = highlighted, disabled = not highlighted)
1556
Wait for user action
1558
Enable ?
Yes
No
1560
Mark list entry (save) as enabled (will be highlighted)
1562
Disable ?
Yes
No
1564
Mark list entry (save) as disabled (will not be highlghted)
1566
Exit ?
Yes
No
1570
RETURN
1568
Handle other user action appropriately

Fig. 15C

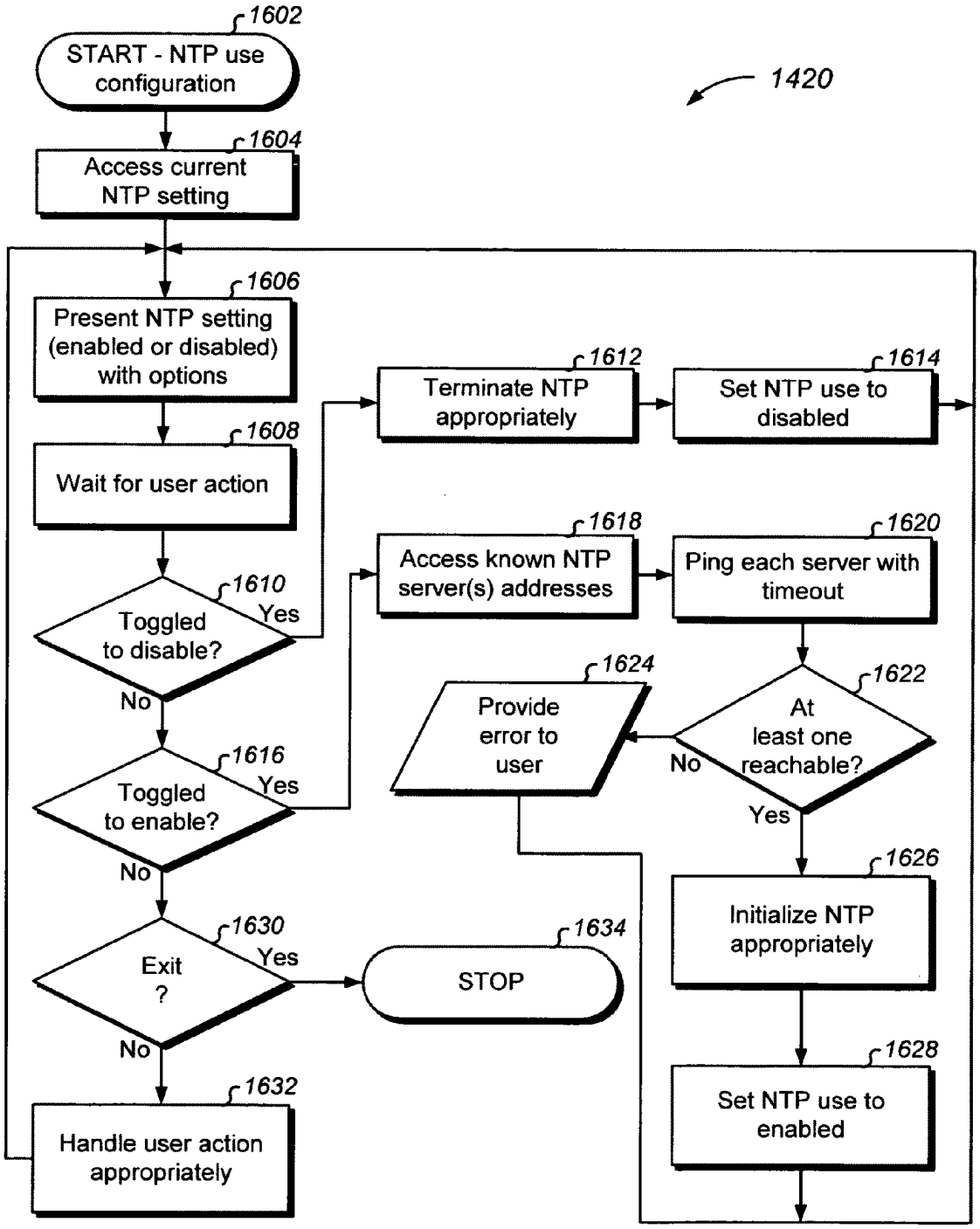

1602
START - NTP use configuration
1420
1604
Access current NTP setting
1606
Present NTP setting (enabled or disabled) with options
1608
Wait for user action
1610
Toggled to disable?
Yes
No
1612
Terminate NTP appropriately
1614
Set NTP use to disabled
1616
Toggled to enable?
Yes
No
1618
Access known NTP server(s) addresses
1620
Ping each server with timeout
1622
At least one reachable?
No
Yes
1624
Provide error to user
1626
Initialize NTP appropriately
1628
Set NTP use to enabled
1630
Exit ?
Yes
No
1634
STOP
1632
Handle user action appropriately

Fig. 16

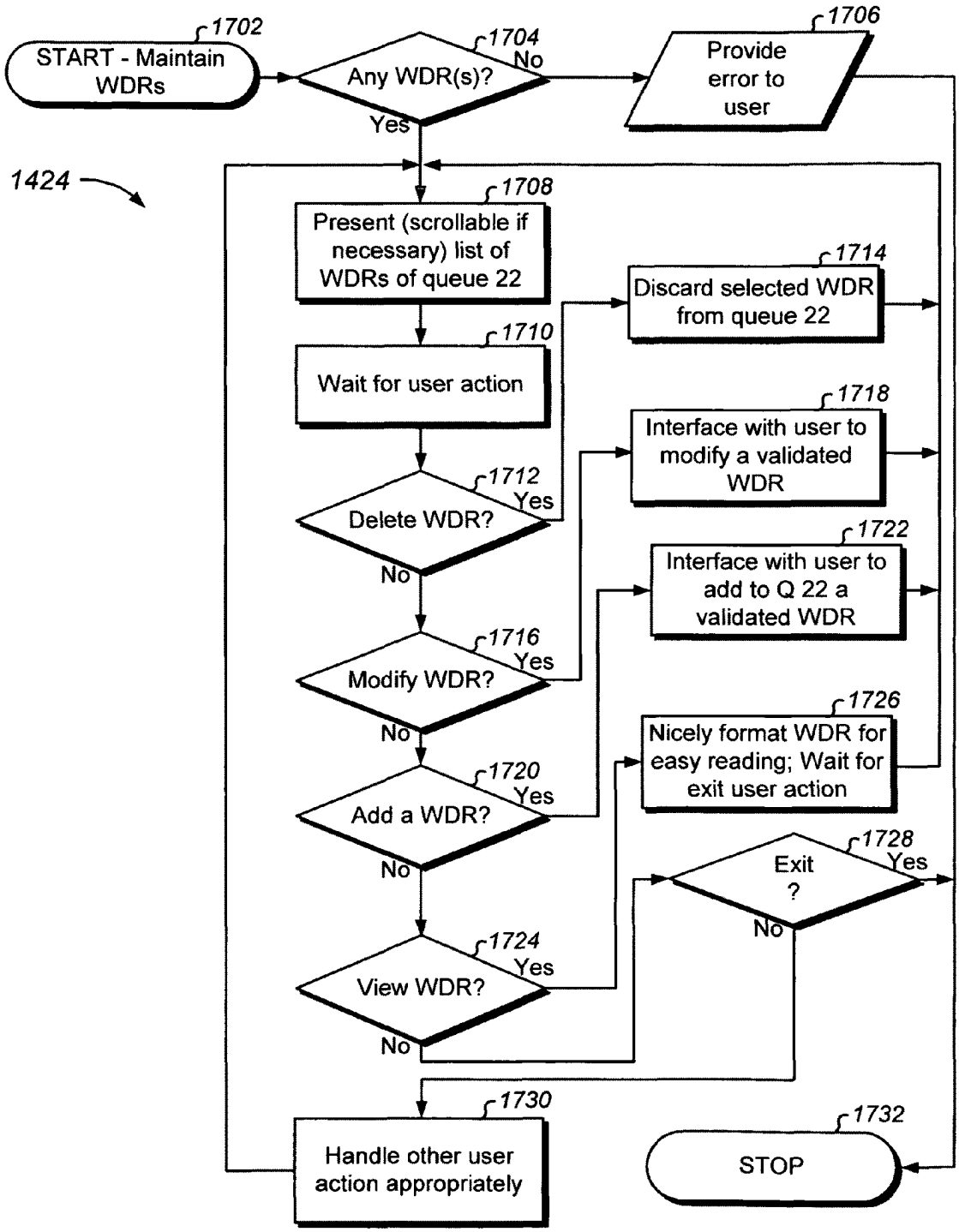

1702
START - Maintain WDRs
1704
Any WDR(s)?
No
Yes
1706
Provide error to user
1424
1708
Present (scrollable if necessary) list of WDRs of queue 22
1710
Wait for user action
1712
Delete WDR?
Yes
No
1714
Discard selected WDR from queue 22
1716
Modify WDR?
Yes
No
1718
Interface with user to modify a validated WDR
1720
Add a WDR?
Yes
No
1722
Interface with user to add to Q 22 a validated WDR
1724
View WDR?
Yes
No
1726
Nicely format WDR for easy reading; Wait for exit user action
1728
Exit ?
Yes
No
1730
Handle other user action appropriately
1732
STOP

Fig. 17

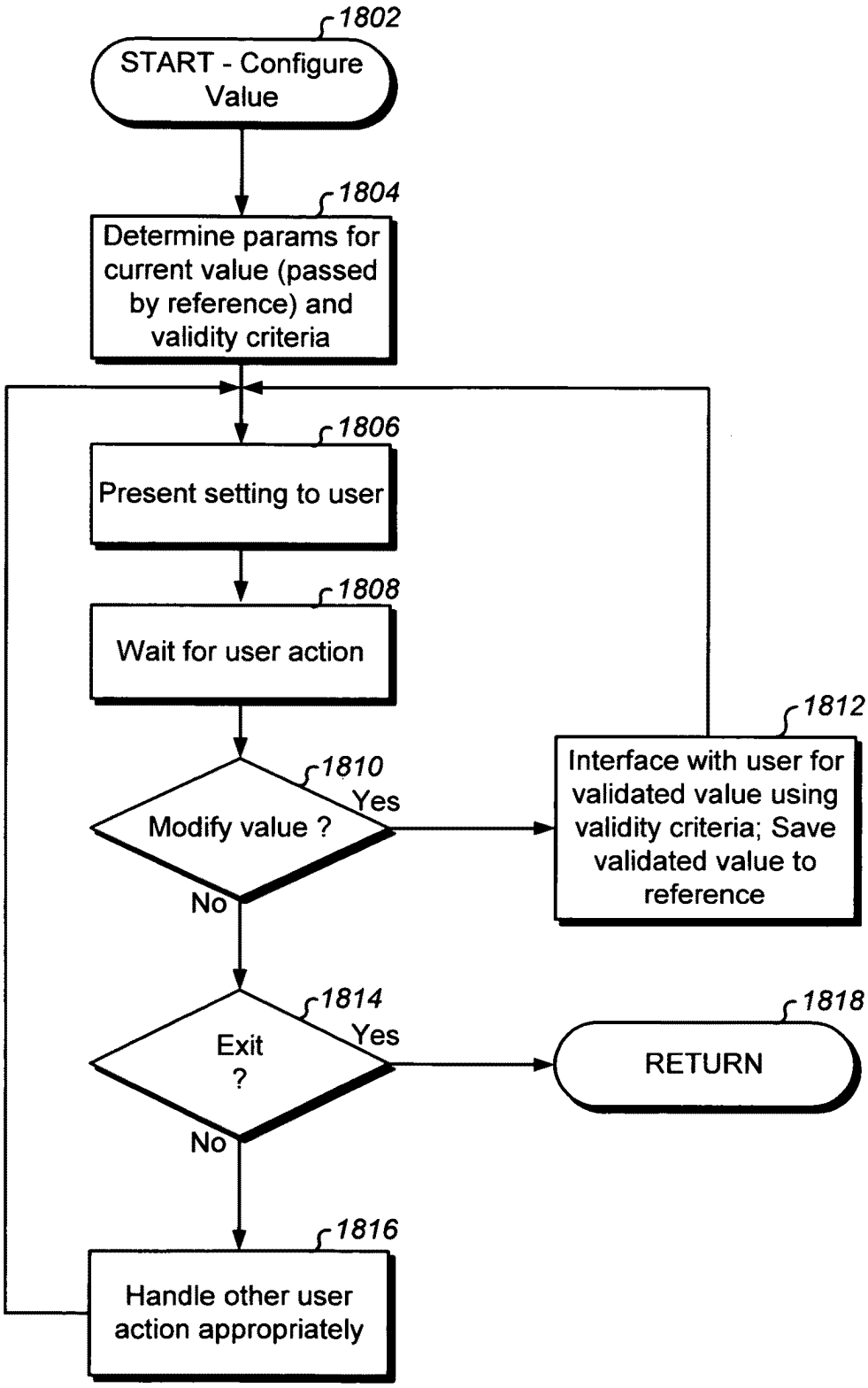

1802
START - Configure Value
1804
Determine params for current value (passed by reference) and validity criteria
1806
Present setting to user
1808
Wait for user action
1810
Modify value ?
Yes
No
1812
Interface with user for validated value using validity criteria; Save validated value to reference
1814
Exit ?
Yes
No
1818
RETURN
1816
Handle other user action appropriately

Fig. 18

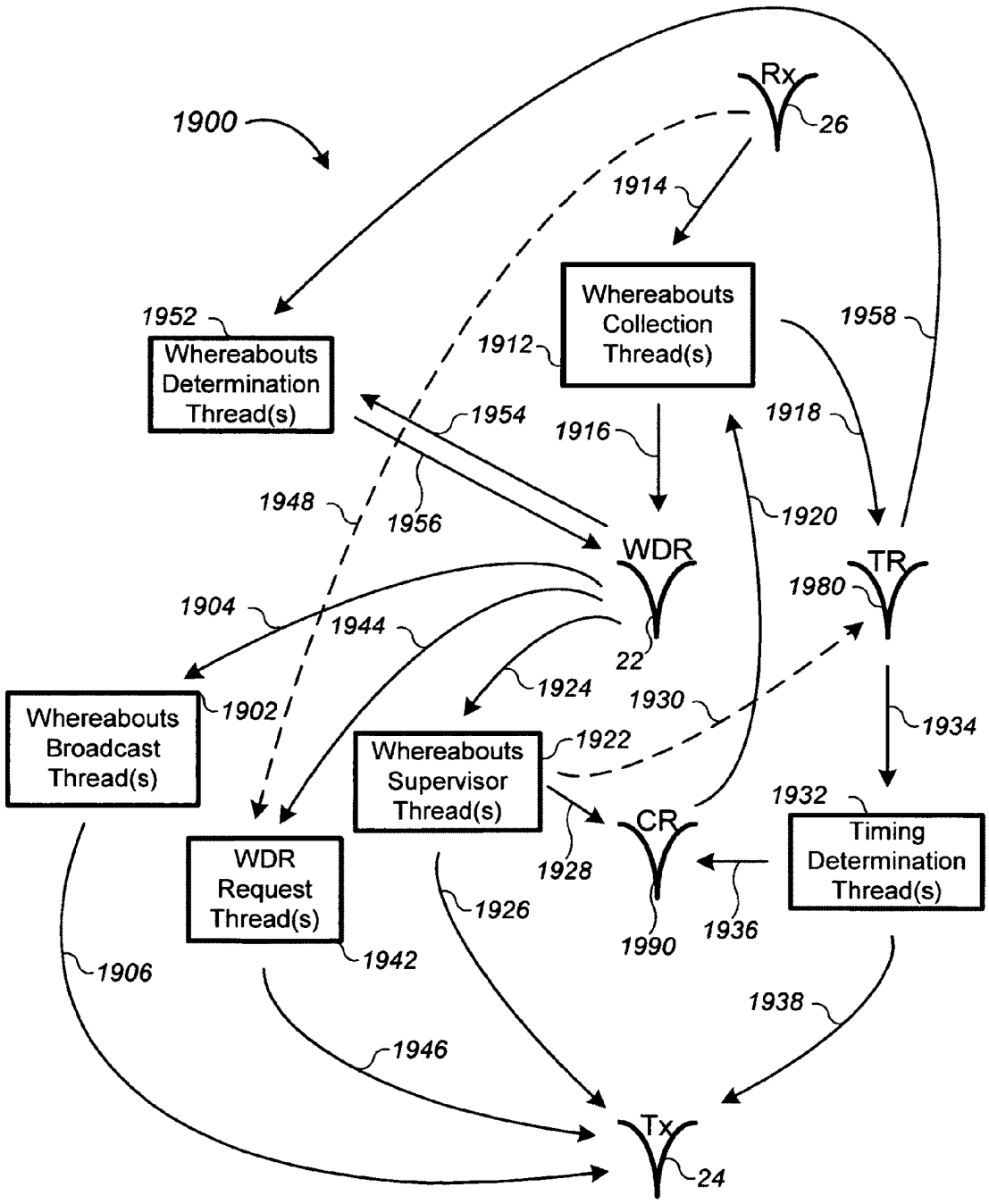

1900
Rx
26
1914
Whereabouts Collection Thread(s)
1912
1958
1952
Whereabouts Determination Thread(s)
1954
1916
1918
1948
1956
1920
WDR
TR
1980
1904
1944
22
1924
1930
1934
Whereabouts Broadcast Thread(s)
1902
Whereabouts Supervisor Thread(s)
1922
1932
CR
Timing Determination Thread(s)
WDR Request Thread(s)
1928
1926
1990
1936
1906
1942
1938
1946
Tx
24

Fig. 19

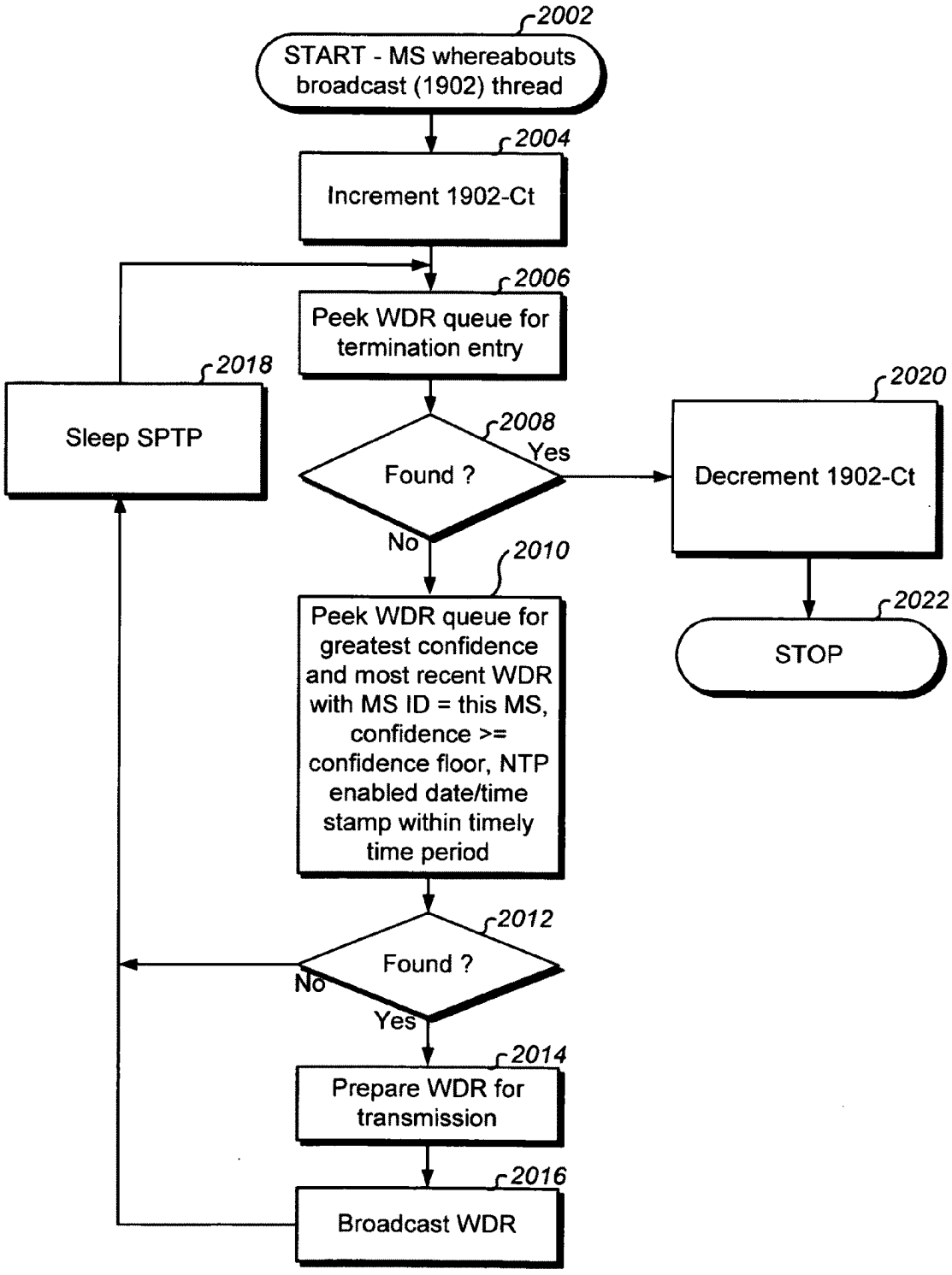

2002
START - MS whereabouts broadcast (1902) thread
2004
Increment 1902-Ct
2006
Peek WDR queue for termination entry
2008
Found ?
Yes
No
2018
Sleep SPTP
2020
Decrement 1902-Ct
2022
STOP
2010
Peek WDR queue for greatest confidence and most recent WDR with MS ID = this MS, confidence >= confidence floor, NTP enabled date/time stamp within timely time period
2012
Found ?
No
Yes
2014
Prepare WDR for transmission
2016
Broadcast WDR

Fig. 20

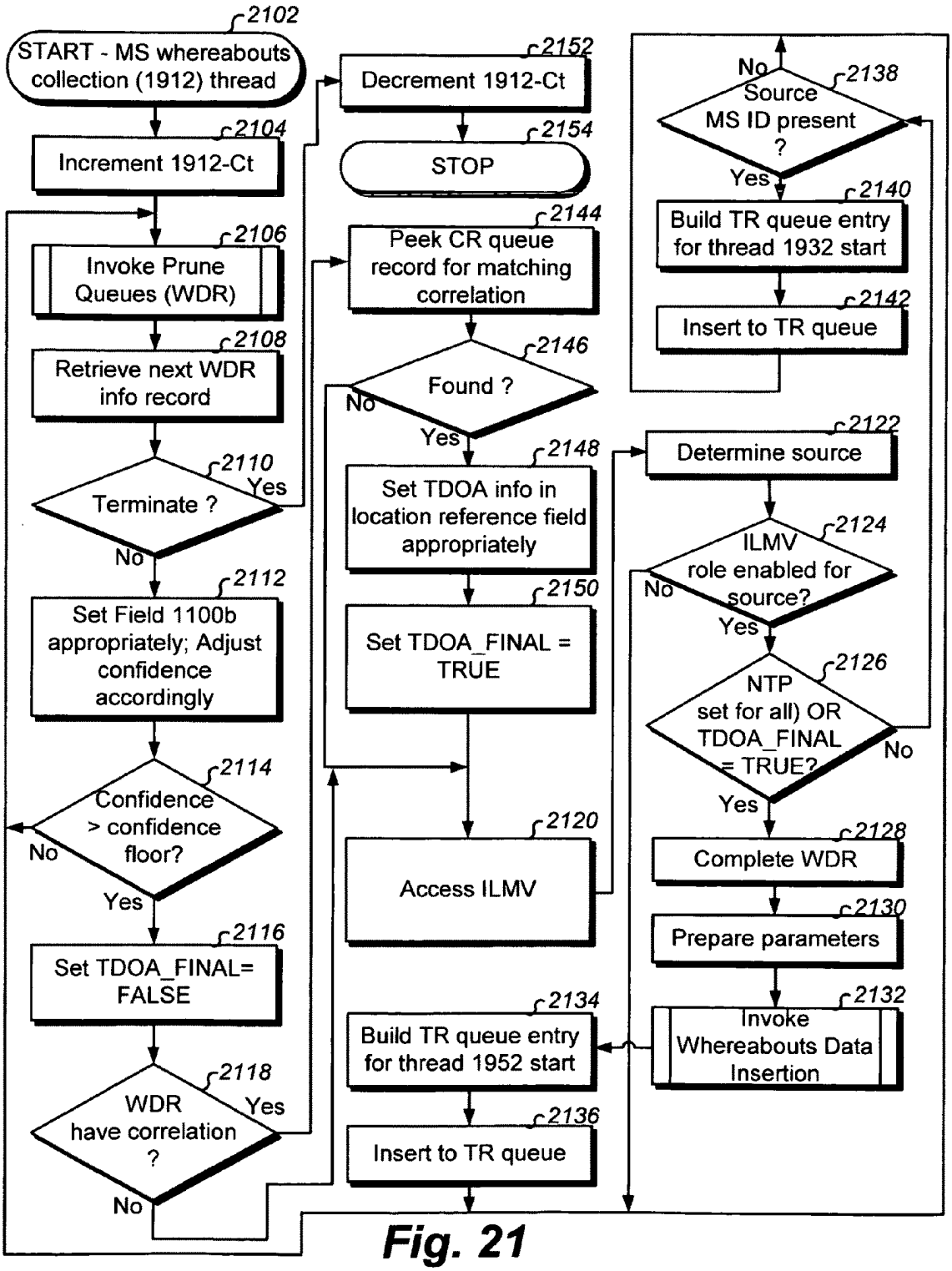

2102
START - MS whereabouts collection (1912) thread
2104
Increment 1912-Ct
2106
Invoke Prune Queues (WDR)
2108
Retrieve next WDR info record
2110
Terminate ?
Yes
No
2112
Set Field 1100b appropriately; Adjust confidence accordingly
2114
Confidence > confidence floor?
No
Yes
2116
Set TDOA_FINAL= FALSE
2118
WDR have correlation ?
Yes
No
2152
Decrement 1912-Ct
2154
STOP
2144
Peek CR queue record for matching correlation
2146
Found ?
No
Yes
2148
Set TDOA info in location reference field appropriately
2150
Set TDOA_FINAL = TRUE
2120
Access ILMV
2134
Build TR queue entry for thread 1952 start
2136
Insert to TR queue
2122
Determine source
2124
ILMV role enabled for source?
No
Yes
2126
NTP set for all) OR TDOA_FINAL = TRUE?
No
Yes
2128
Complete WDR
2130
Prepare parameters
2132
Invoke Whereabouts Data Insertion
2138
Source MS ID present ?
No
Yes
2140
Build TR queue entry for thread 1932 start
2142
Insert to TR queue

Fig. 21

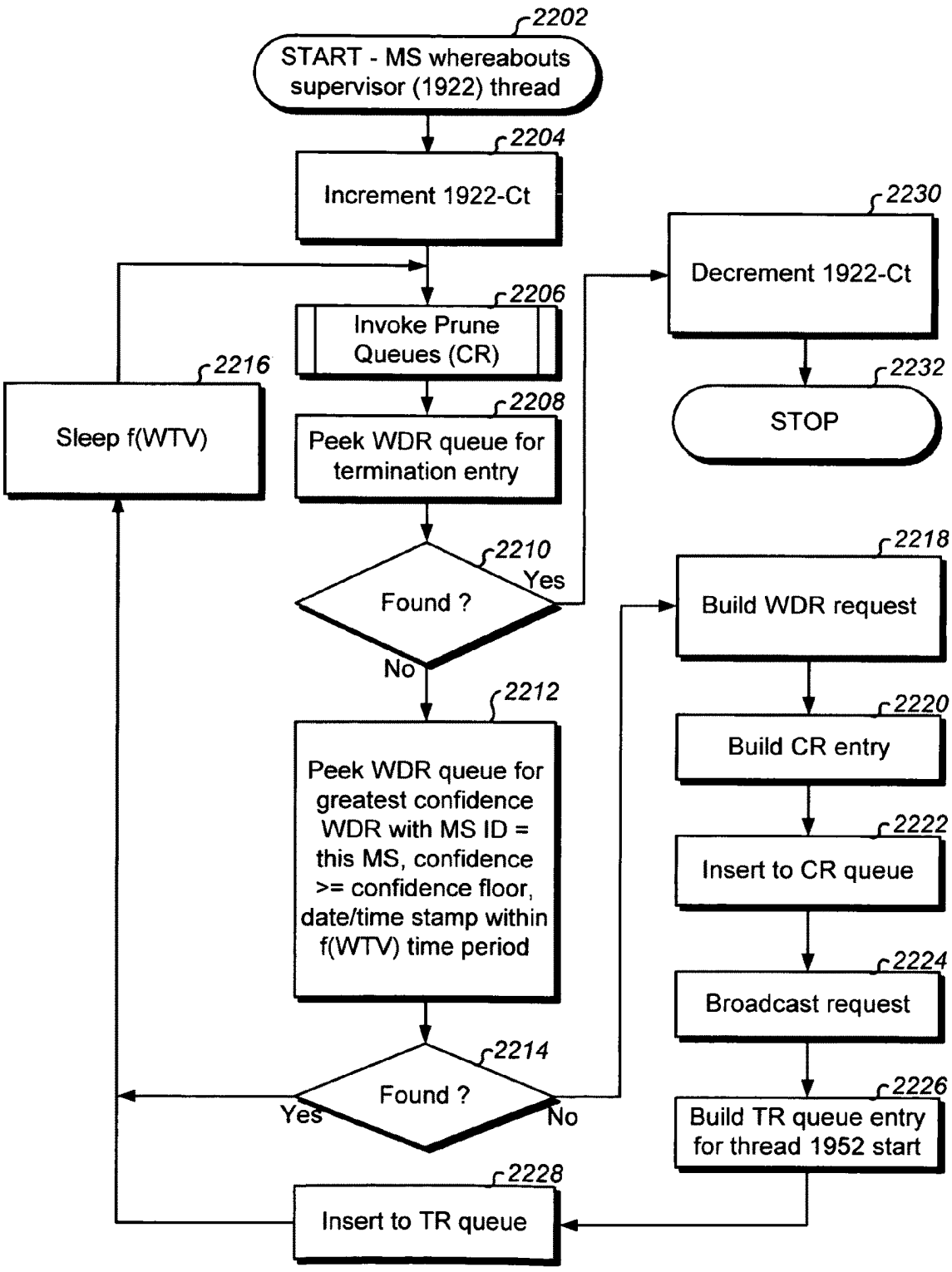

2202
START - MS whereabouts supervisor (1922) thread
2204
Increment 1922-Ct
2206
Invoke Prune Queues (CR)
2208
Peek WDR queue for termination entry
2210
Found ?
Yes
No
2212
Peek WDR queue for greatest confidence WDR with MS ID = this MS, confidence >= confidence floor, date/time stamp within f(WTV) time period
2214
Found ?
Yes
No
2216
Sleep f(WTV)
2218
Build WDR request
2220
Build CR entry
2222
Insert to CR queue
2224
Broadcast request
2226
Build TR queue entry for thread 1952 start
2228
Insert to TR queue
2230
Decrement 1922-Ct
2232
STOP

Fig. 22

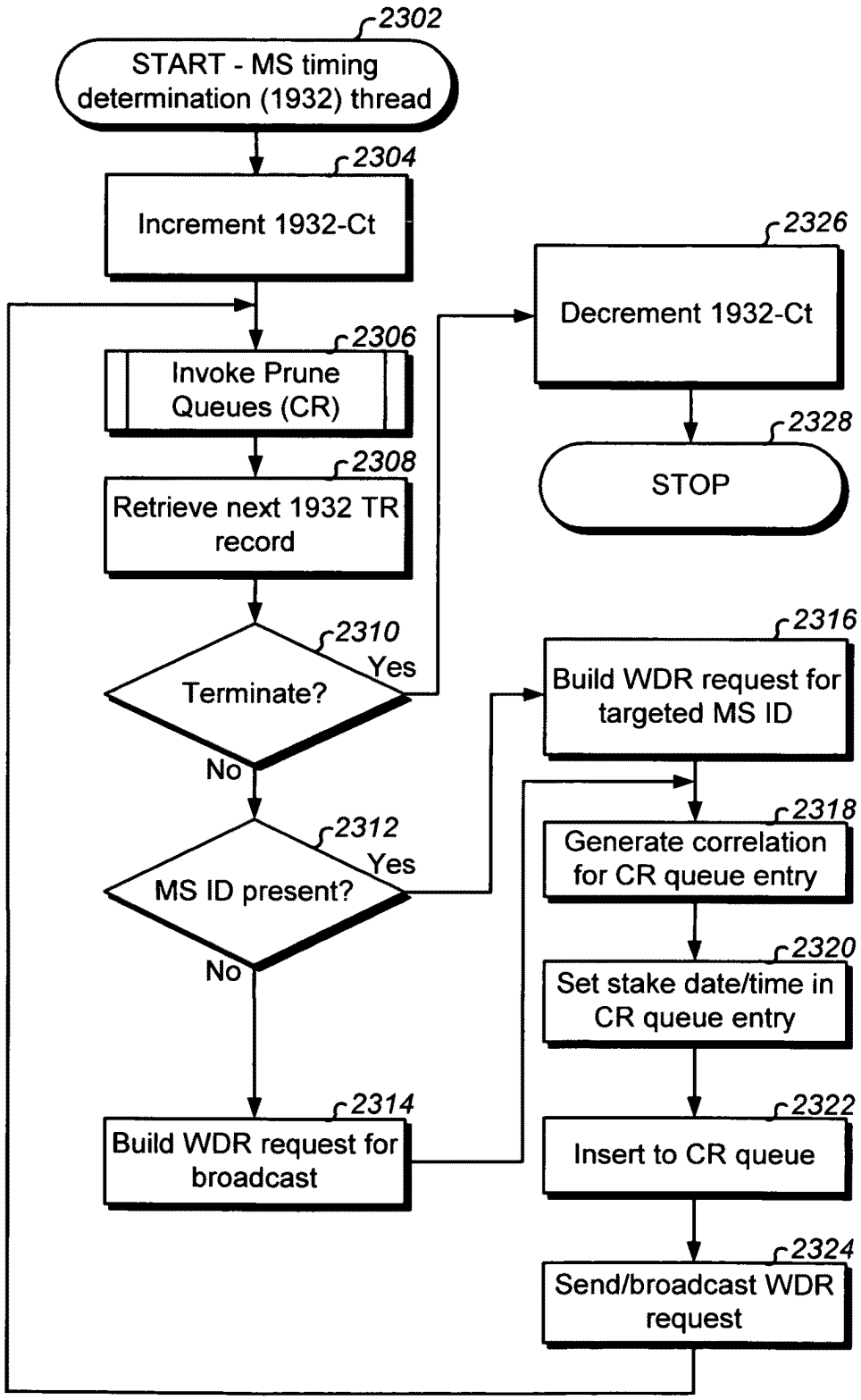

2302
START - MS timing determination (1932) thread
2304
Increment 1932-Ct
2306
Invoke Prune Queues (CR)
2308
Retrieve next 1932 TR record
2310
Terminate?
Yes
No
2312
MS ID present?
Yes
No
2314
Build WDR request for broadcast
2316
Build WDR request for targeted MS ID
2318
Generate correlation for CR queue entry
2320
Set stake date/time in CR queue entry
2322
Insert to CR queue
2324
Send/broadcast WDR request
2326
Decrement 1932-Ct
2328
STOP

Fig. 23

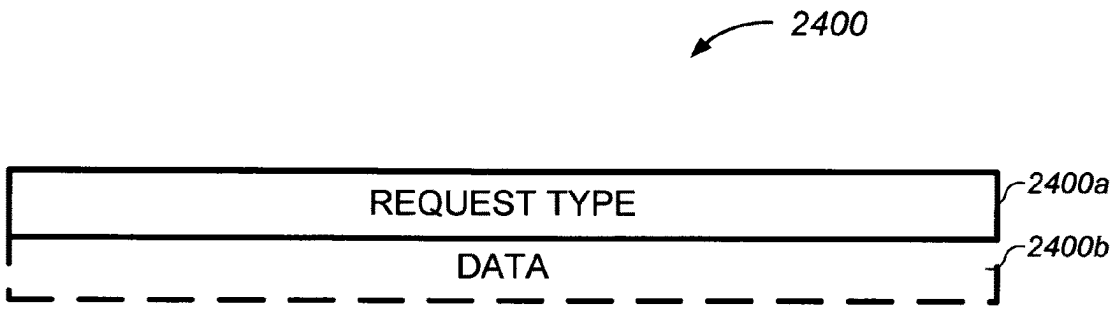


Fig. 24A

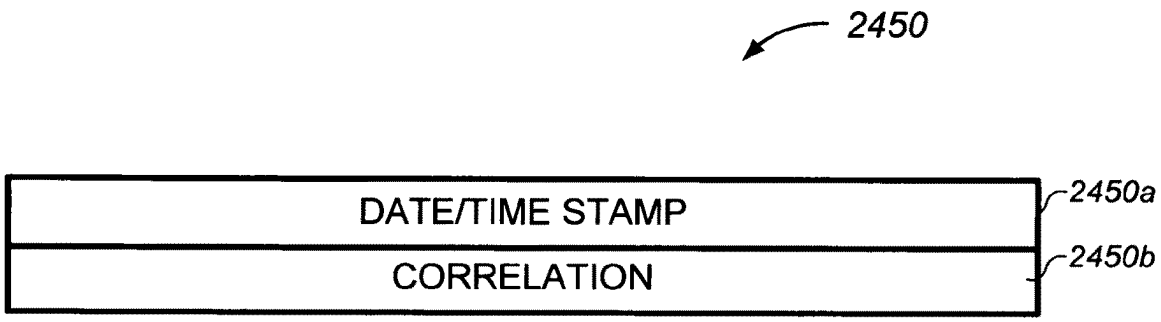


Fig. 24B

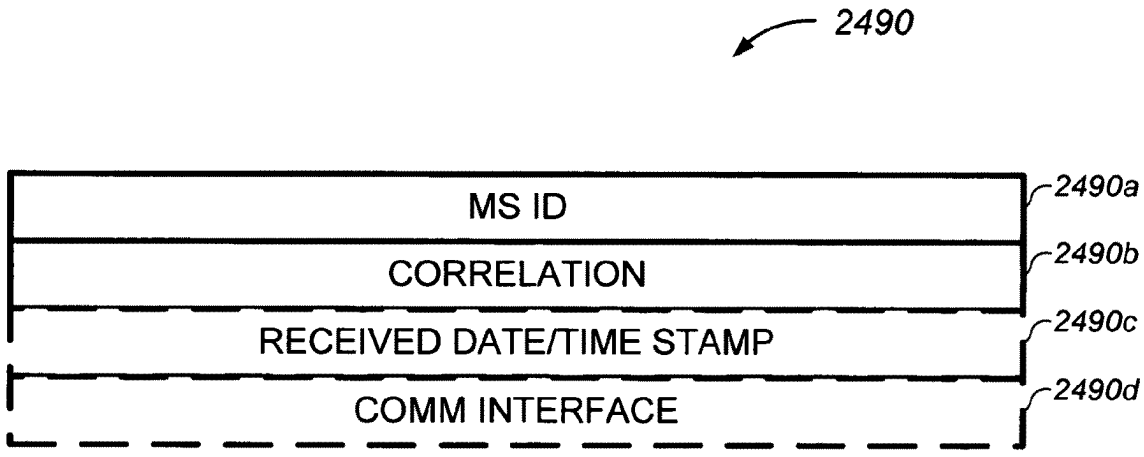

2490
MS ID
2490a
CORRELATION
2490b
RECEIVED DATE/TIME STAMP
2490c
COMM INTERFACE
2490d

Fig. 24C

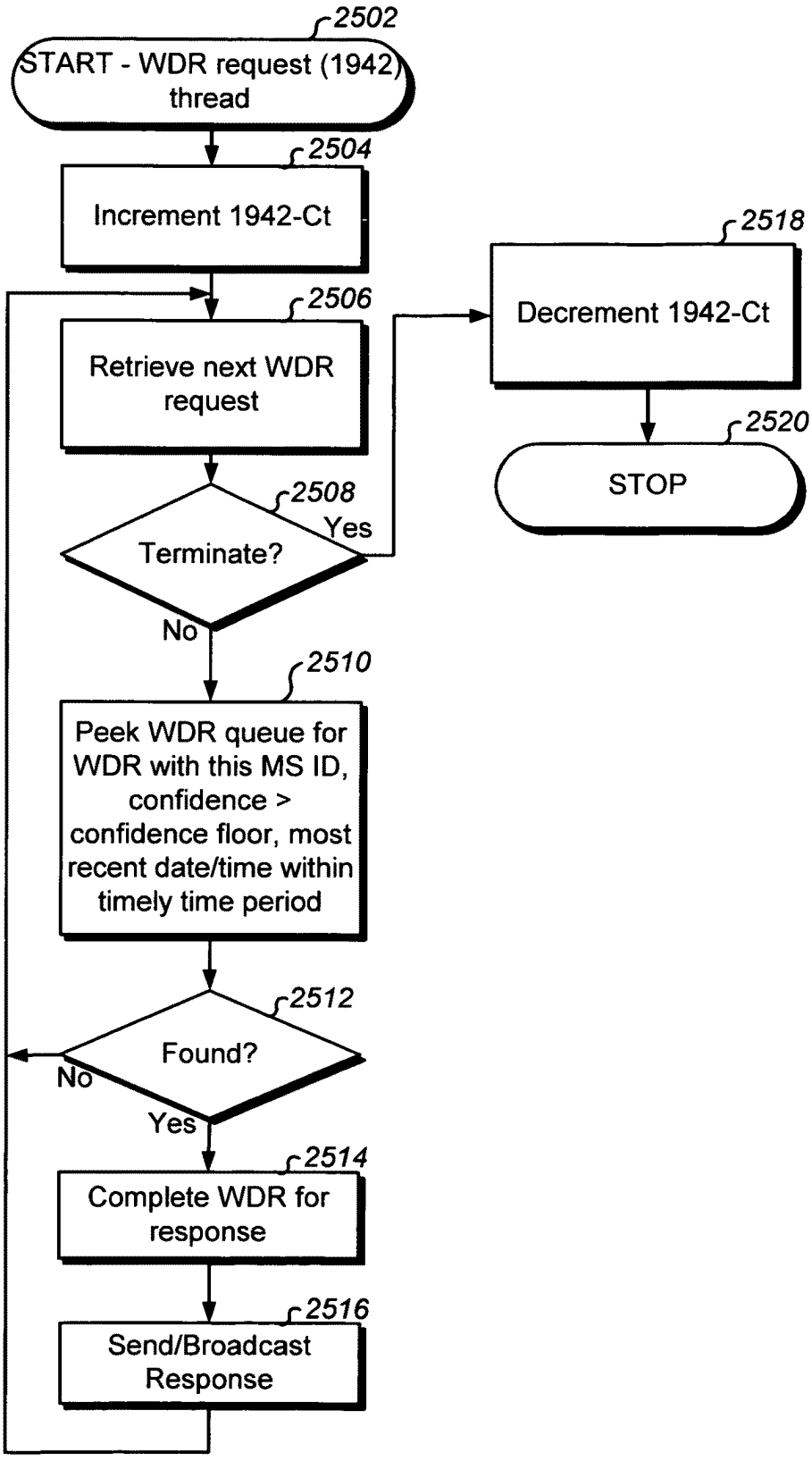

2502
START - WDR request (1942) thread
2504
Increment 1942-Ct
2506
Retrieve next WDR request
2508
Terminate?
Yes
No
2518
Decrement 1942-Ct
2520
STOP
2510
Peek WDR queue for WDR with this MS ID, confidence > confidence floor, most recent date/time within timely time period
2512
Found?
No
Yes
2514
Complete WDR for response
2516
Send/Broadcast Response

Fig. 25

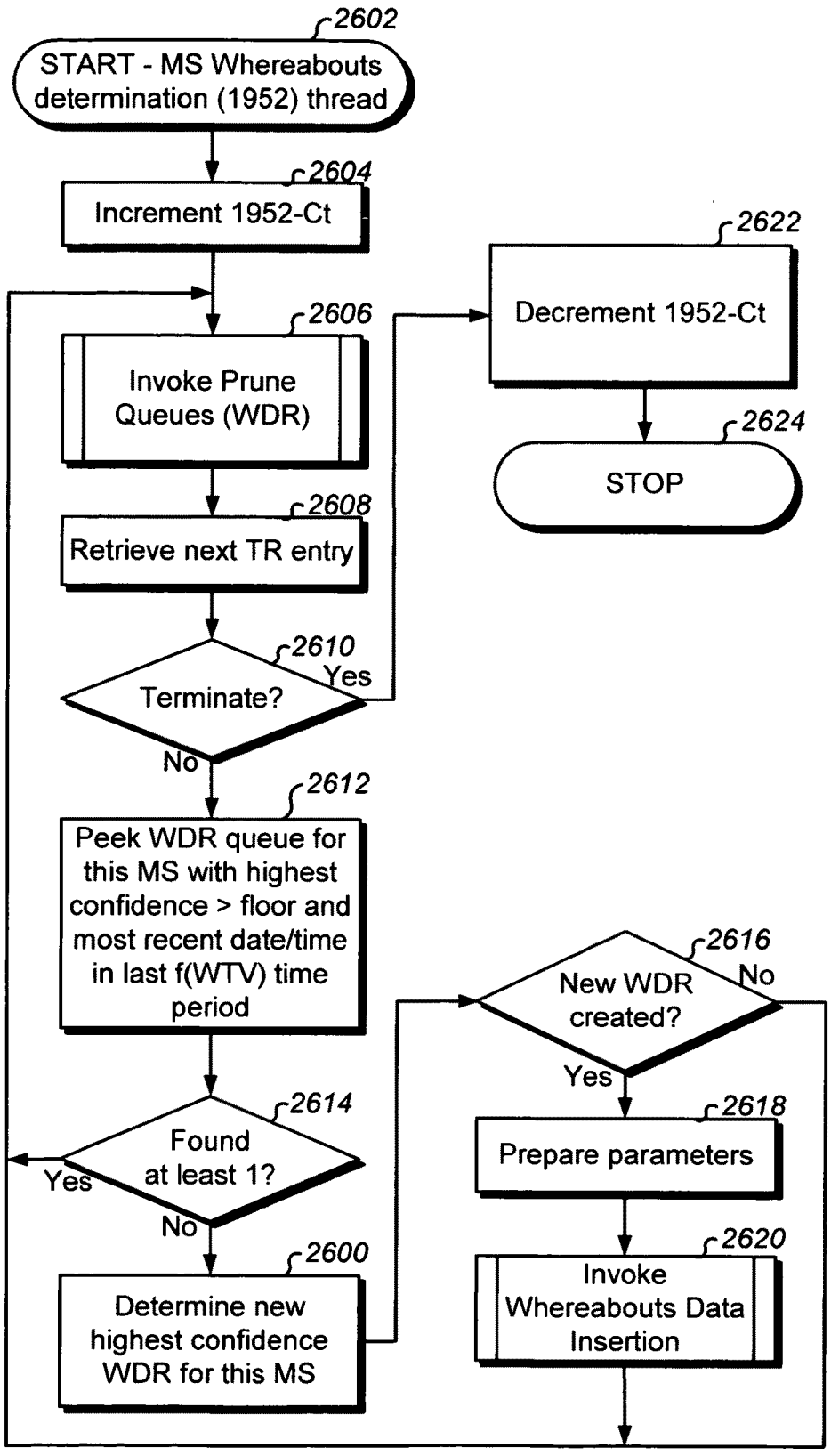

2602
START - MS Whereabouts determination (1952) thread
2604
Increment 1952-Ct
2606
Invoke Prune Queues (WDR)
2608
Retrieve next TR entry
2610
Terminate?
Yes
No
2612
Peek WDR queue for this MS with highest confidence > floor and most recent date/time in last f(WTV) time period
2614
Found at least 1?
Yes
No
2600
Determine new highest confidence WDR for this MS
2616
New WDR created?
No
Yes
2618
Prepare parameters
2620
Invoke Whereabouts Data Insertion
2622
Decrement 1952-Ct
2624
STOP

Fig. 26A

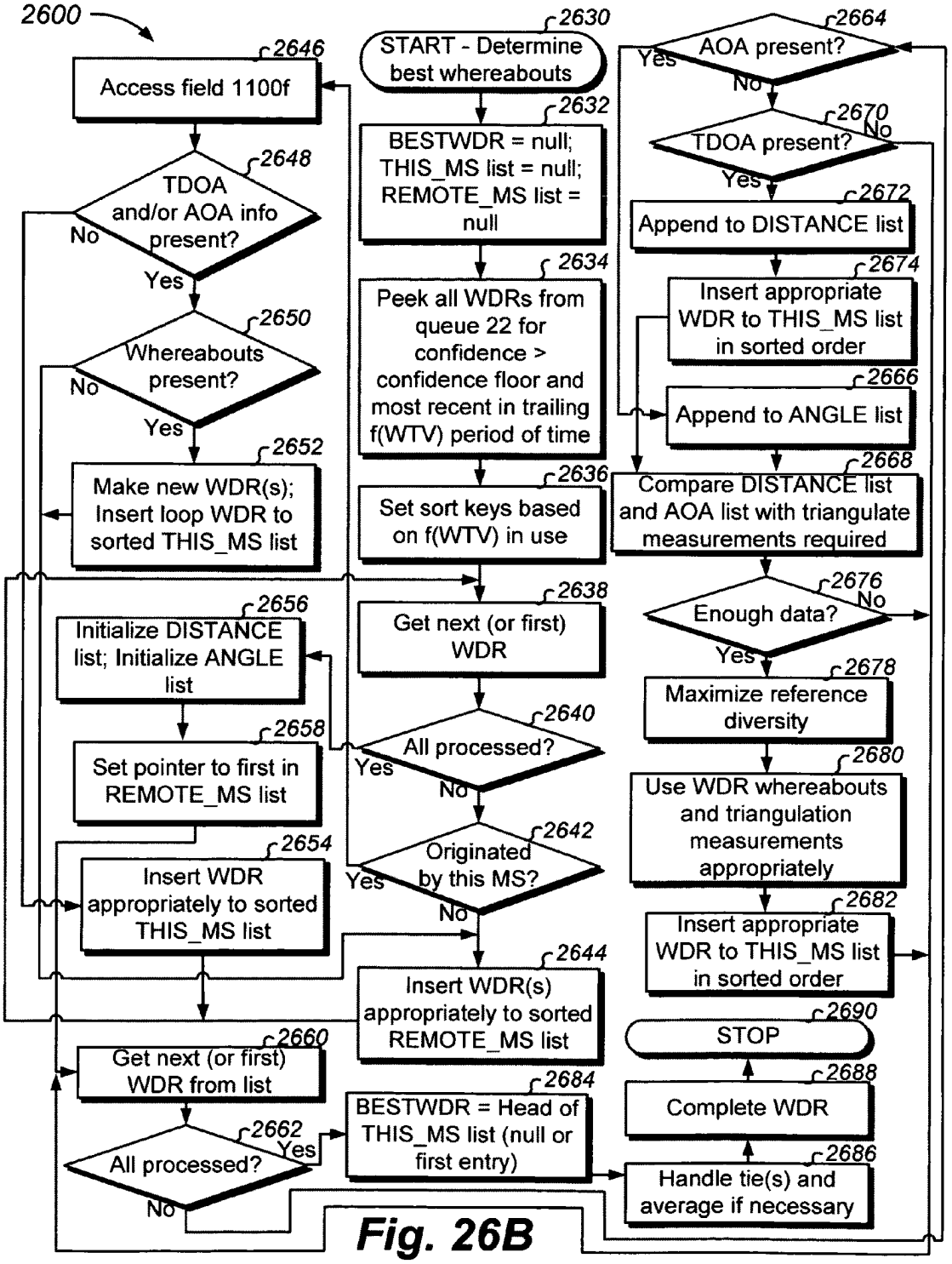

2600
2630
START - Determine best whereabouts
2632
BESTWDR = null; THIS_MS list = null; REMOTE_MS list = null
2634
Peek all WDRs from queue 22 for confidence > confidence floor and most recent in trailing f(WTV) period of time
2636
Set sort keys based on f(WTV) in use
2638
Get next (or first) WDR
2640
All processed?
Yes
No
2642
Originated by this MS?
Yes
No
2644
Insert WDR(s) appropriately to sorted REMOTE_MS list
2646
Access field 1100f
2648
TDOA and/or AOA info present?
No
Yes
2650
Whereabouts present?
No
Yes
2652
Make new WDR(s); Insert loop WDR to sorted THIS_MS list
2656
Initialize DISTANCE list; Initialize ANGLE list
2658
Set pointer to first in REMOTE_MS list
2654
Insert WDR appropriately to sorted THIS_MS list
2660
Get next (or first) WDR from list
2662
All processed?
Yes
No
2684
BESTWDR = Head of THIS_MS list (null or first entry)
2664
AOA present?
Yes
No
2670
TDOA present?
No
Yes
2672
Append to DISTANCE list
2674
Insert appropriate WDR to THIS_MS list in sorted order
2666
Append to ANGLE list
2668
Compare DISTANCE list and AOA list with triangulate measurements required
2676
Enough data?
No
Yes
2678
Maximize reference diversity
2680
Use WDR whereabouts and triangulation measurements appropriately
2682
Insert appropriate WDR to THIS_MS list in sorted order
2690
STOP
2688
Complete WDR
2686
Handle tie(s) and average if necessary

Fig. 26B

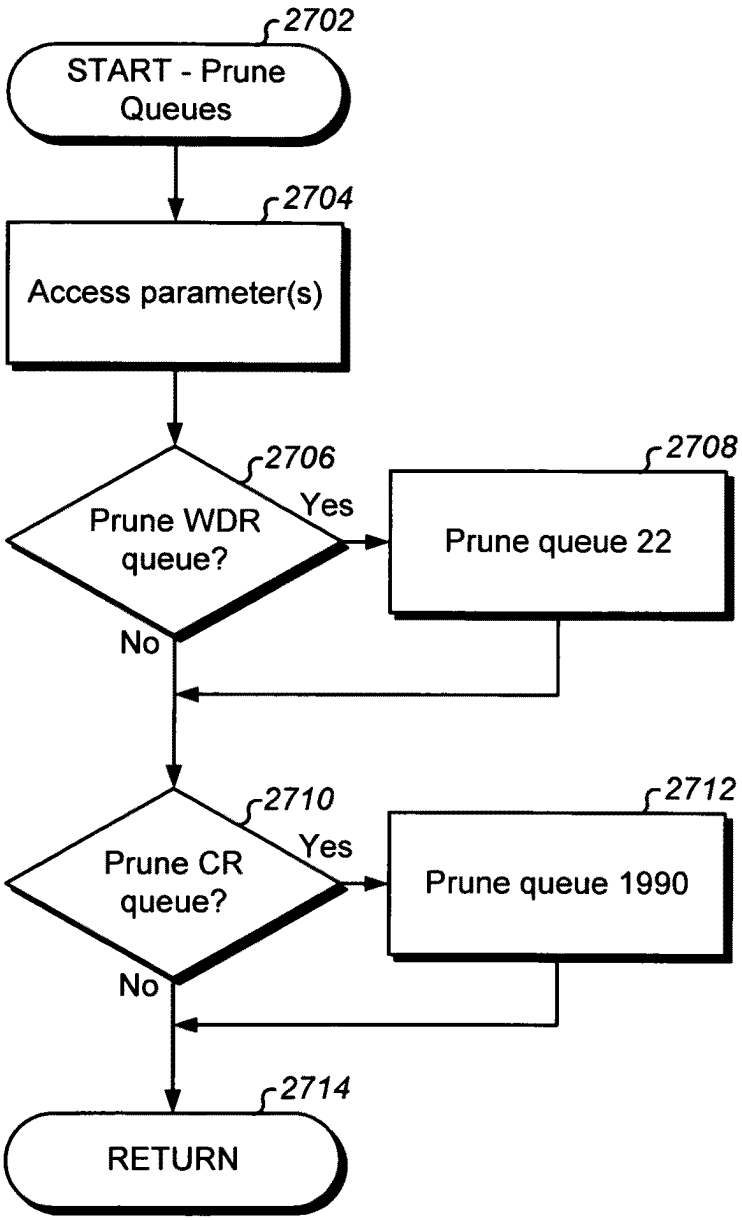

2702
START - Prune Queues
2704
Access parameter(s)
2706
Prune WDR queue?
Yes
No
2708
Prune queue 22
2710
Prune CR queue?
Yes
No
2712
Prune queue 1990
2714
RETURN

Fig. 27

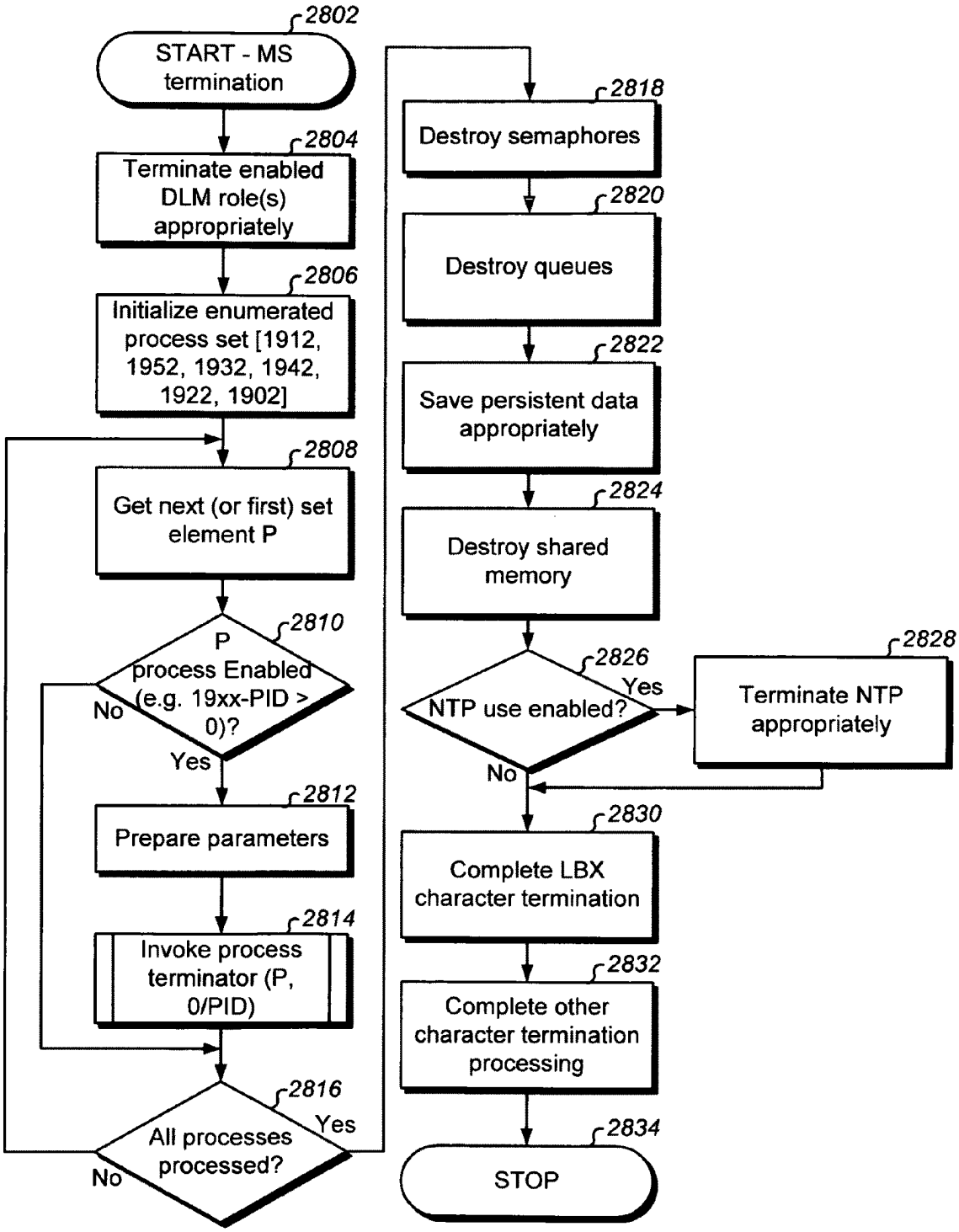

2802
START - MS termination
2804
Terminate enabled DLM role(s) appropriately
2806
Initialize enumerated process set [1912, 1952, 1932, 1942, 1922, 1902]
2808
Get next (or first) set element P
2810
P process Enabled (e.g. 19xx-PID > 0)?
No
Yes
2812
Prepare parameters
2814
Invoke process terminator (P, 0/PID)
2816
All processes processed?
Yes
No
2818
Destroy semaphores
2820
Destroy queues
2822
Save persistent data appropriately
2824
Destroy shared memory
2826
NTP use enabled?
Yes
No
2828
Terminate NTP appropriately
2830
Complete LBX character termination
2832
Complete other character termination processing
2834
STOP

Fig. 28

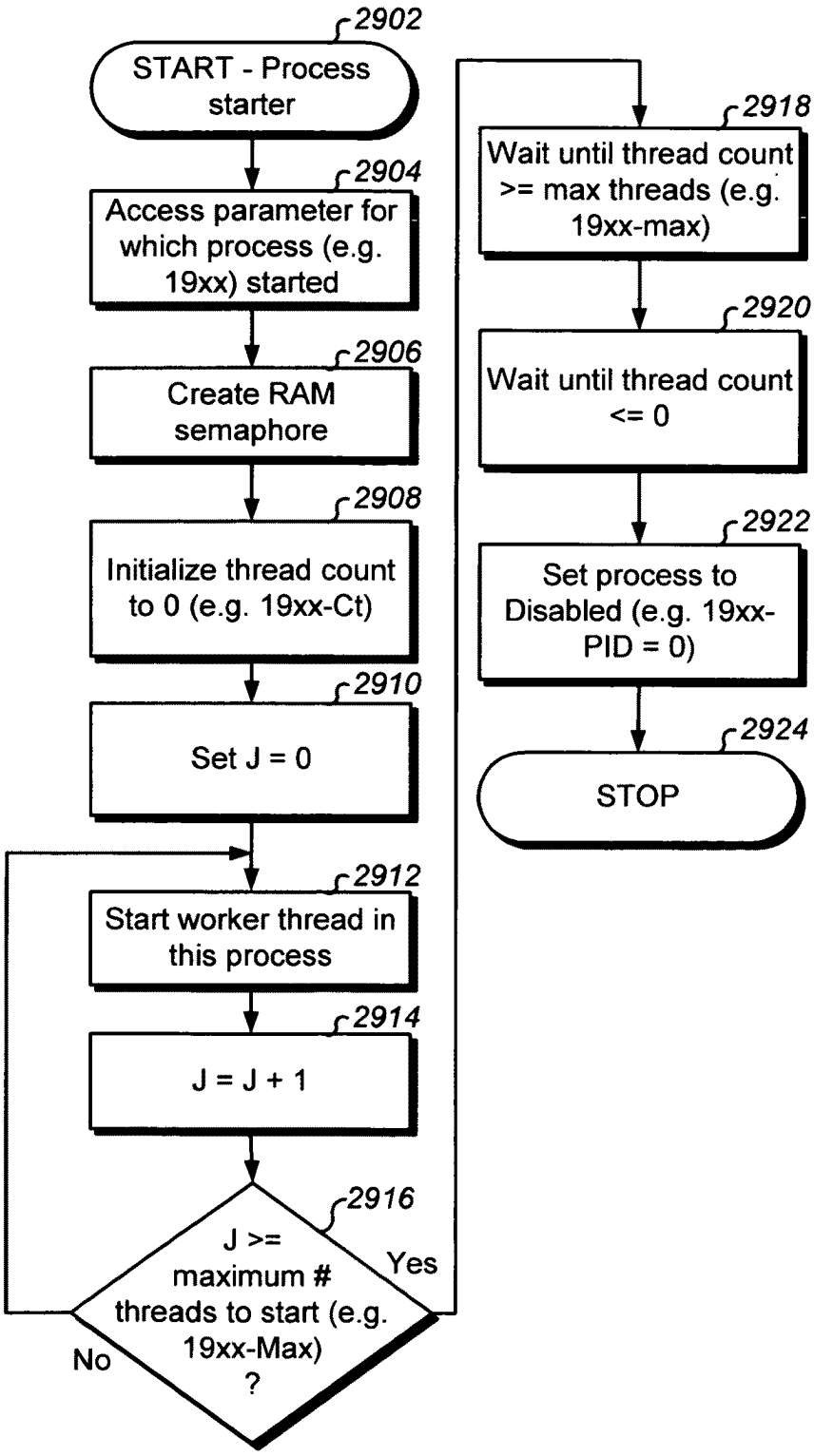

2902
START - Process starter
2904
Access parameter for which process (e.g. 19xx) started
2906
Create RAM semaphore
2908
Initialize thread count to 0 (e.g. 19xx-Ct)
2910
Set J = 0
2912
Start worker thread in this process
2914
J = J + 1
2916
J >= maximum # threads to start (e.g. 19xx-Max) ?
Yes
No
2918
Wait until thread count >= max threads (e.g. 19xx-max)
2920
Wait until thread count <= 0
2922
Set process to Disabled (e.g. 19xx-PID = 0)
2924
STOP

Fig. 29A

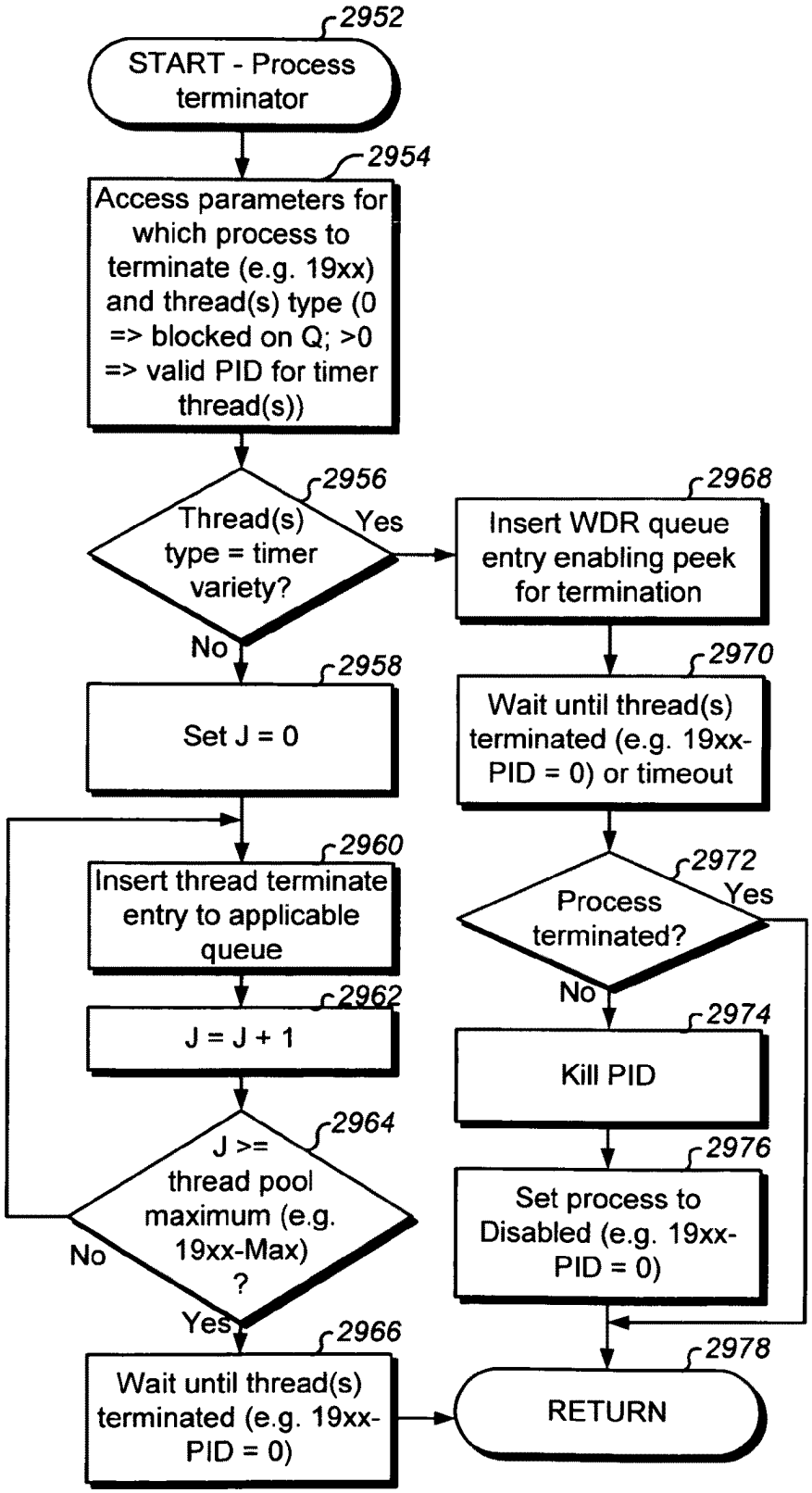

2952
START - Process terminator
2954
Access parameters for which process to terminate (e.g. 19xx) and thread(s) type (0 => blocked on Q; >0 => valid PID for timer thread(s))
2956
Thread(s) type = timer variety?
Yes
No
2958
Set J = 0
2960
Insert thread terminate entry to applicable queue
2962
J = J + 1
2964
J >= thread pool maximum (e.g. 19xx-Max) ?
No
Yes
2966
Wait until thread(s) terminated (e.g. 19xx-PID = 0)
2968
Insert WDR queue entry enabling peek for termination
2970
Wait until thread(s) terminated (e.g. 19xx-PID = 0) or timeout
2972
Process terminated?
Yes
No
2974
Kill PID
2976
Set process to Disabled (e.g. 19xx-PID = 0)
2978
RETURN

Fig. 29B

3002a

```
// Figs. 30A through 30E syntaxes (e.g. delimeters, etc) used should enforce
// appropriate unambiguous grammar parsability for Lex&Yacc, top down
// recursive parsing, XML encoding, other syntactic embodiments, applicable semantic
// representations, and any other syntactic/semantic embodiments. Figs. 30A through 30E BNF
// grammar elaborates for a corresponding interpreter, recommended syntaxes, programming
// language stuctures and/or objects, DB schemas, ANSI datastream encoding (e.g. X.409),
// flowchart processing blocks and locations in parent application flowhcharts, and any other
// analogous implementation embodiments or subsets thereof.

// ***** Common BNF grammar (e.g. in Data 8): *****

Variables          = "null" | Variables Variable
      // Variables are placed anywhere; Can be used for referencing (a="..." b=a  c=b)

Variable           = VarType(VarName) = "null" | VarType(VarName) = ...value(s)... |
                         VarType(VarName) = [ Variables ] [ VarInstantiations ] |
                         VarType(VarName) = [ VarInstantiations ]  [ Variables ]
      // Variables scope to following & descending nesting; "value" has appropriate syntax
      // per VarType; VarName can be set to other variables (e.g. indirect tree structure)

VarInstantiations  = "null" | VarInstantiations VarInstantiate

VarInstantiate     = *VarName(Param1="x1", Param2="x2", ... ParamN="xN")  for N >= 0
      // Parameters allow optionally substituting occurrences in VarName with new values
      // prior to instantiation.

VarName            = "text string"

Description        = "null" | "text string" | VarInstantiate

History            = [ CreatorInfo ] [ ModifierInfo ] | VarInstantiations

CreatorInfo        = "null" | [ CreateDateTime ] [ CreatorID] [ CreatorIDType ]
                         [ CreatorAddr ] [ CreatorSysID ] [ CreatorSysType ]
                         [ CreatorSysAddr ] | VarInstantiations

ModifierInfo       = "null" | [ LastModifyDateTime ] [ LastModifyID ]
                [ LastModifyIDType ] [ LastModifyAddr ] [ LastModifySysID ]
                [ LastModifySysType ] [ LastModifySysAddr ] | VarInstantiations

CreateDateTime     = "date/time stamp" | VarInstantiate

CreatorID          = ID

CreatorIDType      = IDType
```

Fig. 30A

3002b

```
CreatorAddr          = Address

CreatorSysID         = "text string" | VarInstantiate

CreatorSysType       = "system type" | VarInstantiate   // e.g. type of MS

CreatorSysAddr       = Address

LastModifyDateTime=  "date/time stamp" | VarInstantiate

LastModifyID         = ID

LastModifyIDType     = IDType

LastModifyAddr       = Address

LastModifySysID      = "text string" | VarInstantiate

LastModifySysType  = "system type" | VarInstantiate

LastModifySysAddr  = Address

ID                   = "MS ID" [ Description ] [ History ] |
        "MS Group ID" [ Description ] [ History ] | "User ID" [ Description ] [ History ] |
        "User Group ID" [ Description ] [ History ] | "logical handle" [ Description ] [ History ] |
        "physical handle" [ Description ] [ History ] | VarInstantiations

IDType               = "MS_ID" | "MS_Group_ID" | "User_ID" | "User_Group_ID" |
                         "logical_handle" | "physical_handle" | VarInstantiate

Address              = "ip address" | "SNA address" | "Postal address" |
                       "point" | "logical address" | "physical address" | "situational location" |
                       "2 dimensional area" | "3 dimensional area" | VarInstantiate

TimeSpec             = "Xdate/time stamp" | "Xdate/time period" | VarInstantiate

VarType              = Description | History | ID | IDType | CreatorInfo | ModifierInfo |
        CreateDateTime | CreatorID | CreatorIDType | CreatorAddr | CreatorSysID |
        CreatorSysType | CreatorSysAddr | LastModifyDateTime | LastModifyID |
        LastModifyIDType | LastModifyAddr | LastModifySysID | LastModifySysType |
        LastModifySysAddr | Address | "Xdate/time stamp" |  "Xdate/time period" | "text string" |
        "system type" | TimeSpec | "MS ID" | "MS Group ID" | "User ID" | "User Group ID" |
        "logical handle" | physical handle" | "...Address elaborations..." |
        "...IDType elaborations..." |  Variable  // | VarInstantiate here as well (but elaborates)
```

Fig. 30B

3034

```
// ***** BNF grammar for Permissions 10: *****

PermissionBody     = "null" | [ Variables ] [ Permissions ]
        // [ Variables ] placed anywhere (not shown in constructs below to enhance readability)

Permissions        = "null" | Permissions Permission | VarInstantiations

Permission         = Grantor Grantee [ Grants ] [ TimeSpec ] [ Description ] [ History ] |
                          VarInstantiations
        // No Grants implies granting all permissions; This embodiment ensures non-null
        // Grantor and Grantee, but "null" could be used (e.g. for placeholder entries).

Grantor            = ID [ IDType ] | VarInstantiations
        // ID defaults (e.g. MS ID) when IDType not present

Grantee            = ID [ IDType ] | VarInstantiations

Grants             = "null" | Grants Grant | Privileges | VarInstantiations

Grant              = "grant name" AND (Privileges [ TimeSpec ] [ Description ] [ History ] |
                          Grants [ TimeSpec ] [ Description ] [ History ] |
                          VarInstantiations)

Privileges         = "null" | Privileges Privilege | VarInstantiations

Privilege          = "atomic privilege for assignment" [ MSRelevance ]
                          [ TimeSpec ] [ Description ] [ History ] | VarInstantiations

MSRelevance        = "MS relevance descriptor"

Groups             = "null" | Groups Group | VarInstantiations

Group              = "group name" AND (IDs [ Description ] [ History ] |
                          Groups [ Description ] [ History ] |
                          VarInstantiations)

IDs                = "null" | IDs ID [ IDType ] | VarInstantiations

VarType            = *VarType | Permissions | Permission | Grantor | Grantee | Grants |
                          Grant | Privileges | Privilege | MSRelevance | Groups | Group |
                          IDs
```

Fig. 30C

3068a

```
// ***** BNF grammar for Charters 12: *****

CharterBody        = "null" | [ Variables ] [ Charters ]
      // [ Variables ] placed anywhere (not shown in constructs below to enhance readability)

Charters           = "null" | Charters Charter | VarInstantiations

Charter            = Grantee Grantor Expression Actions [ TimeSpec ] [ Description ]
                        [ History ] | VarInstantiations

Expression         = Conditions [ TimeSpec ] | VarInstantiations
      // This embodiment ensures at least one condition to a Charter, but "null" could be
      // used (e.g. for placeholder entries).

Conditions         = Condition | Conditions CondOp Condition] | VarInstantiations

CondOp             = "and" | "or" | VarInstantiations

Condition          = Term Op Term [ TimeSpec ] [ Description ] [ History ] |
      Value [ TimeSpec ] [ Description ] [ History ] |
      Invocation [ TimeSpec ] [ Description ] [ History ] | VarInstantiations
      // Another embodiment allows unary operators (e.g. "not"), for example for boolean
      // WDR fields (e.g. Applications field(s)). Current boolean tests for "True" or "False",
      // or non-zero = "True" and zero = "False". Value & Invocation result in a boolean.

Term               = WDRTerm [ TimeSpec ] [ Description ] [ History ] |
                     AppTerm [ TimeSpec ] [ Description ] [ History ] |
                     Value [ TimeSpec ] [ Description ] [ History ] |
                     Invocation [ TimeSpec ] [ Description ] [ History ] |
                      VarInstantiate

WDRTerm            = "Any WDR 1100 field, or any subset thereof" [ Description ]
                        [ History ] | VarInstantiate

AppTerm            = "Any Application data field, or any subset thereof" [ Description ]
                        [ History ] | VarInstantiate

Value              = Data | "number" | "text string" | "value" | "True" | "False" |
                        "atomic term" | ID [ IDType ] | "null" | VarInstantiate

Data               = "typed memory pointer" | "typed memory value" | "typed file path" |
                        "typed file path and offset" | "typed DB qualifier" | VarInstantiate
      // i.e. pointer or value from stack, globals, shared memory, file data location, DB
      // pointer, DB value, or any other data.
```

Fig. 30D

3068b

```
Invocation          = "DLL interface(optional params...)" |
                          "Linked interface(optional params...)" |
                          "executable path(optional params...)" | VarInstantiate
      // Invocation can return any value of any type, except will be converted to a boolean
      // when used as a Term (0 = False, else = True). Best to return boolean when Term use.

Op                  = [ "atomic not operator" ] "atomic operator" | ProfileMatch |
                          VarInstantiate

ProfileMatch        = "atomic profile match operator" | VarInstantiate

Actions             = "null" | Actions Action

Action              = [ Host ] Command Operand [Parameters]
                          [ TimeSpec ] [ Description ] [ History ] | VarInstantiations

Host                = "null" | ID [IDType] | VarInstantiations

Command             = "atomic command" | VarInstantiations
      // Command may map to translation member entry of natural language map

Operand             = "atomic operand" | VarInstantiations
      // Some embodiments have no need for an operand in this grammar (e.g. command file
      // reference, DLL call, self contained command, invocation callout, etc).

Parameters          = "null" | Parameters Parameter | VarInstantiations

Parameter           = WDRTerm [ Description ] [ History ] |
                          AppTerm [ Description ] [ History ] |
                          Value [ Description ] [ History ] |
                          Invocation [ Description ] [ History ] |
                          ID [ IDType ] [ Description ] [ History ] |
                          VarInstantiate [ Description ] [ History ]

VarType             = *VarType | Charters | Charter | Expression | Conditions | Condition |
                      CondOp | WDRTerm | Term | Value | Data | Invocation | Op | Actions
                      ProfileMatch | Action | Command | Operand | Parameters | Parameter |
                      Host
```

Fig. 30E

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	...
201	#, sender, msg/subj, attribs, recip(s)	#, sender, msg/subj, attribs, recip(s)	#	#	#, system(s)	#, system(s)	#, ack, source, system(s)	#, ack, system(s)	#, ack, source, system(s)	#, system(s)	
203	link, sender, msg/subj, attribs, recip(s)	link, sender, msg/subj, attribs, recip(s)	link, params	link, params	link, params, system(s)	link, params, system(s)	link, ack, source, system(s)	link, ack, system(s)	link, ack, source, system(s)	link, params, system(s)	
205	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	email, system(s)	body, sender, msg/subj, attribs, recip(s)	email, ack, source, system(s)	email, ack, system(s)	email, ack, source, system(s)	email, system(s)	
207	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	msg, system(s)	body, sender, msg/subj, attribs, recip(s)	msg, ack, source, system(s)	msg, ack, system(s)	msg, ack, source, system(s)	msg, system(s)	
209	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	body, sender, msg/subj, attribs, recip(s)	email, system(s)	body, sender, msg/subj, attribs, recip(s)	email, ack, source, system(s)	email, ack, system(s)	email, ack, source, system(s)	email, system(s)	
211	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, sender, msg/subj, attribs, recip(s)	msg, system(s)	body, sender, msg/subj, attribs, recip(s)	msg, ack, source, system(s)	msg, ack, system(s)	msg, ack, source, system(s)	msg, system(s)	
213	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, sender, msg/subj, attribs, recip(s)	indicator, system(s)	indicator, system(s)	indicator, ack, source, system(s)	indicator, ack, system(s)	indicator, ack, source, system(s)	indicator, system(s)	

Fig. 31A

Operand ↓	Command									
	101	103	105	119	107	109	111	113	115	117	...
215	app, sender, msg/subj, attribs, recip(s)	app, sender, msg/subj, attribs, recip(s)	app, params	app, params	app, params, system(s)	app, params, system(s)	app, params, ack, source, system(s)	app, params, ack, system(s)	app, params, ack, source, system(s)	app, params, system(s)	
217	doc, sender, msg/subj, attribs, recip(s)	doc, sender, msg/subj, attribs, recip(s)	doc	doc	doc, system(s)	doc, system(s)	doc, ack, source, system(s)	doc, ack, system(s)	doc, ack, source, system(s)	doc, system(s)	
219	path, sender, msg/subj, attribs, recip(s)	path, sender, msg/subj, attribs, recip(s)	path	path	path, system(s)	path, system(s)	path, ack, source, system(s)	path, ack, system(s)	path, ack, source, system(s)	path, system(s)	
221	content, sender, msg/subj, attribs, recip(s)	content, sender, msg/subj, attribs, recip(s)	content	content	content, system(s)	content, system(s)	content, ack, source, system(s)	content, ack, system(s)	content, ack, source, system(s)	content, system(s)	
223	DB-obj, sender, msg/subj, attribs, recip(s)	DB-obj, query, sender, msg/subj, attribs, recip(s)	DB-obj	DB-obj, query	DB-obj, system(s)	DB-obj, query, system(s)	DB-obj, ack, source, system(s)	DB-obj, ack, system(s)	DB-obj, ack, source, system(s)	DB-obj, query, system(s)	
225	data, sender, msg/subj, attribs, recip(s)	data, value, sender, msg/subj, attribs, recip(s)	data, value	data, value	data, system(s)	data, value, system(s)	data, ack, source, system(s)	data, ack, system(s)	data, ack, source, system(s)	data, value, system(s)	

Fig. 31B

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	...
227	sem, sender, msg/subj, attribs, recip(s)	sem, cmd, sender, msg/subj, attribs, recip(s)	sem, cmd	sem, cmd	sem, system(s)	sem, cmd, system(s)	sem, ack, source, system(s)	sem, ack, system(s)	sem, ack, source, system(s)	sem, cmd, system(s)	
229	path, sender, msg/subj, attribs, recip(s)	path, sender, msg/subj, attribs, recip(s)	path	path	path, system(s)	path, system(s)	path, ack, source, system(s)	path, ack, system(s)	path, ack, source, system(s)	path, system(s)	
231	app, macro, sender, msg/subj, attribs, recip(s)	app, macro, sender, msg/subj, attribs, recip(s)	app, macro	app, macro	app, macro, system(s)	app, macro, system(s)	app, params, ack, source, system(s)	app, params, ack, system(s)	app, params, ack, source, system(s)	app, macro, system(s)	
233	"<alt> <prtscr>", sender, msg/subj, attribs, recip(s)	"<alt> <prtscr>", sender, msg/subj, attribs, recip(s)	"<alt> <prtscr>"	"<alt> <prtscr>"	objtxt, system(s)	cmds, system(s)	"<alt> <prtscr>", ack, source, system(s)	objtxt, ack, system(s)	"<alt> <prtscr>", ack, source, system(s)	"<alt> <prtscr>", system(s)	
235	macro, sender, msg/subj, attribs, recip(s)	macro, sender, msg/subj, attribs, recip(s)	macro	macro	macro, system(s)	macro, system(s)	macro, ack, system(s)	app, params, ack, system(s)	macro, ack, system(s)	macro, system(s)	
237	iodev, input, sender, msg/subj, attribs, recip(s)	iodev, input, sender, msg/subj, attribs, recip(s)	iodev, input	iodev, input	iodev, input, system(s)	iodev, input, system(s)	iodev, input, ack, system(s)	iodev, ack, system(s)	iodev, input, ack, system(s)	iodev, input, system(s)	

Fig. 31C

Operand ↓	Command									
	101	103	105	119	107	109	111	113	115	117	...
239	iodev, output, sender, subj, attribs, recip(s)	iodev, output, sender, subj, attribs, recip(s)	iodev, output	iodev, output	iodev, output, system(s)	iodev, output, system(s)	iodev, output, ack, system(s)	iodev, ack, system(s)	iodev, output, ack, system(s)	iodev, output, system(s)	
241	alert, sender, msg/subj, attribs, recip(s)	alert, sender, msg/subj, attribs, recip(s)	alert	alert	alert, system(s)	alert, system(s)	alert, ack, source, system(s)	alert, ack, system(s)	alert, ack, source, system(s)	alert, system(s)	
243	pid, signal, sender, msg/subj, attribs, recip(s)	pid, signal, sender, msg/subj, attribs, recip(s)	pid, signal	pid, signal	prname, system(s)	pid, signal, system(s)	prname, ack, source, system(s)	prname, ack, system(s)	prname, ack, source, system(s)	prname, signal system(s)	
245	container, sender, msg/subj, attribs, recip(s)	container, sender, msg/subj, attribs, recip(s)	container	container	container, system(s)	container, system(s)	container, ack, source, system(s)	container, ack, system(s)	container, ack, source, system(s)	container, system(s)	
247	progobj, data, sender, msg/subj, attribs, recip(s)	progobj, data, sender, msg/subj, attribs, recip(s)	progobj, data	progobj, data, sender, msg/subj, attribs, recip(s)	progobj, data, system(s)	progobj, data, system(s)	progobj, ack, source, system(s)	progobj, ack, system(s)	progobj, ack, source, system(s)	progobj, data, system(s)	
249	cursor, sender, msg/subj, attribs, recip(s)	cursor, sender, msg/subj, attribs, recip(s)	cursor	cursor, sender, msg/subj, attribs, recip(s)	cursor, system(s)	cursor, attribs, system(s)	ack, source, system(s)	ack, system(s)	ack, source, system(s)	cursor, attribs, system(s)	

Fig. 31D

Operand ↓	Command										
	101	103	105	119	107	109	111	113	115	117	...
251	calobj, sender, msg/subj, attribs, recip(s)	calobj, sender, msg/subj, attribs, recip(s)	calobj, sender, msg/subj, attribs, recip(s)	calobj, sender, msg/subj, attribs, recip(s)	calobj, system(s)	calobj, attribs, system(s)	calobj, ack, source, system(s)	calobj, ack, system(s)	calobj, ack, source, system(s)	calobj, attribs, system(s)	
253	ABobj, sender, msg/subj, attribs, recip(s)	ABobj, sender, msg/subj, attribs, recip(s)	ABobj, sender, msg/subj, attribs, recip(s)	ABobj, sender, msg/subj, attribs, recip(s)	ABobj, system(s)	ABobj, attribs, system(s)	ABobj, ack, source, system(s)	ABobj, ack, system(s)	ABobj, ack, source, system(s)	ABobj, attribs, system(s)	
...											

Fig. 31E

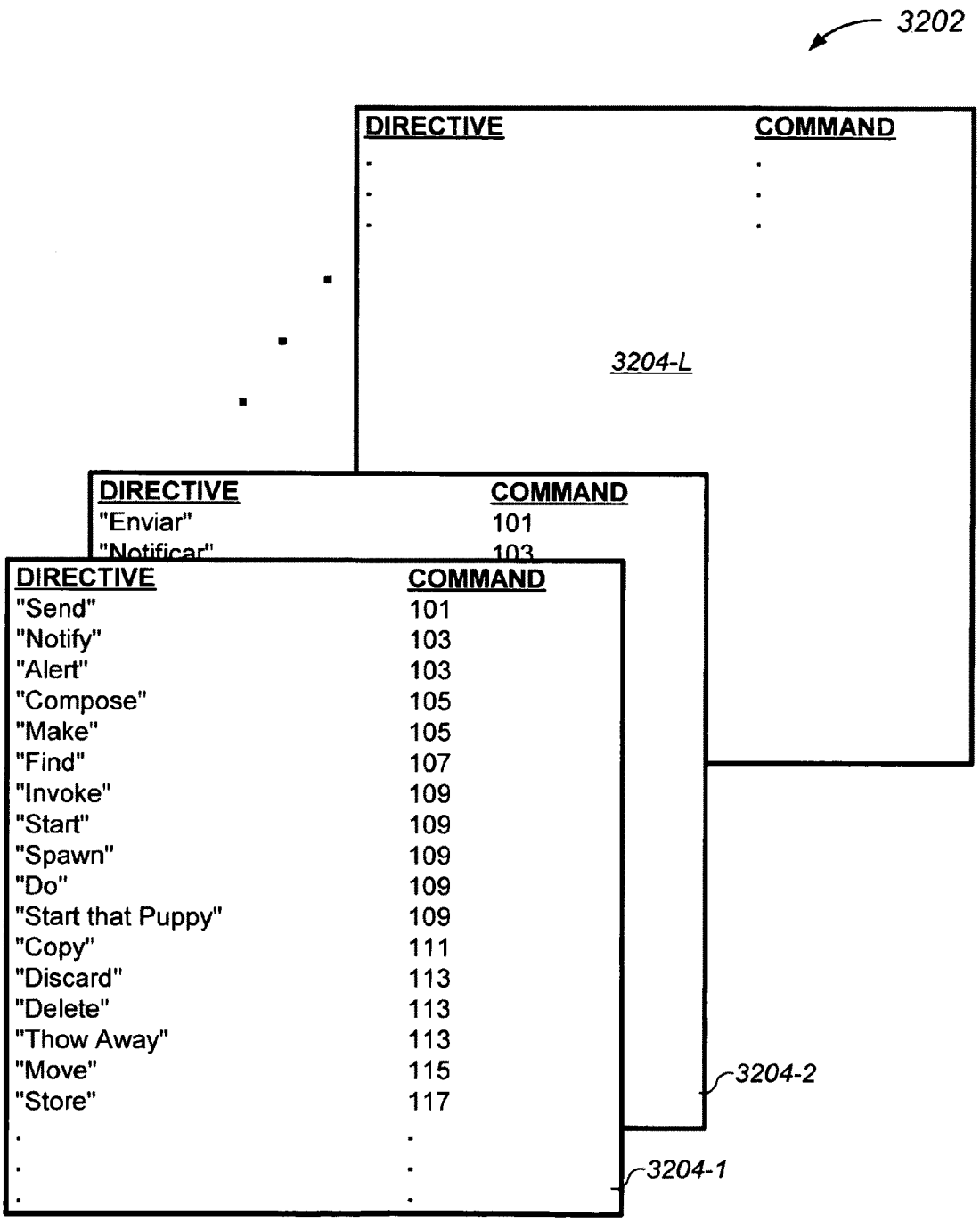

3202
DIRECTIVE
COMMAND
3204-L
DIRECTIVE
COMMAND
"Enviar"
101
DIRECTIVE
COMMAND
"Send" 101
"Notify" 103
"Alert" 103
"Compose" 105
"Make" 105
"Find" 107
"Invoke" 109
"Start" 109
"Spawn" 109
"Do" 109
"Start that Puppy" 109
"Copy" 111
"Discard" 113
"Delete" 113
"Thow Away" 113
"Move" 115
"Store" 117
3204-2
3204-1

Fig. 32A

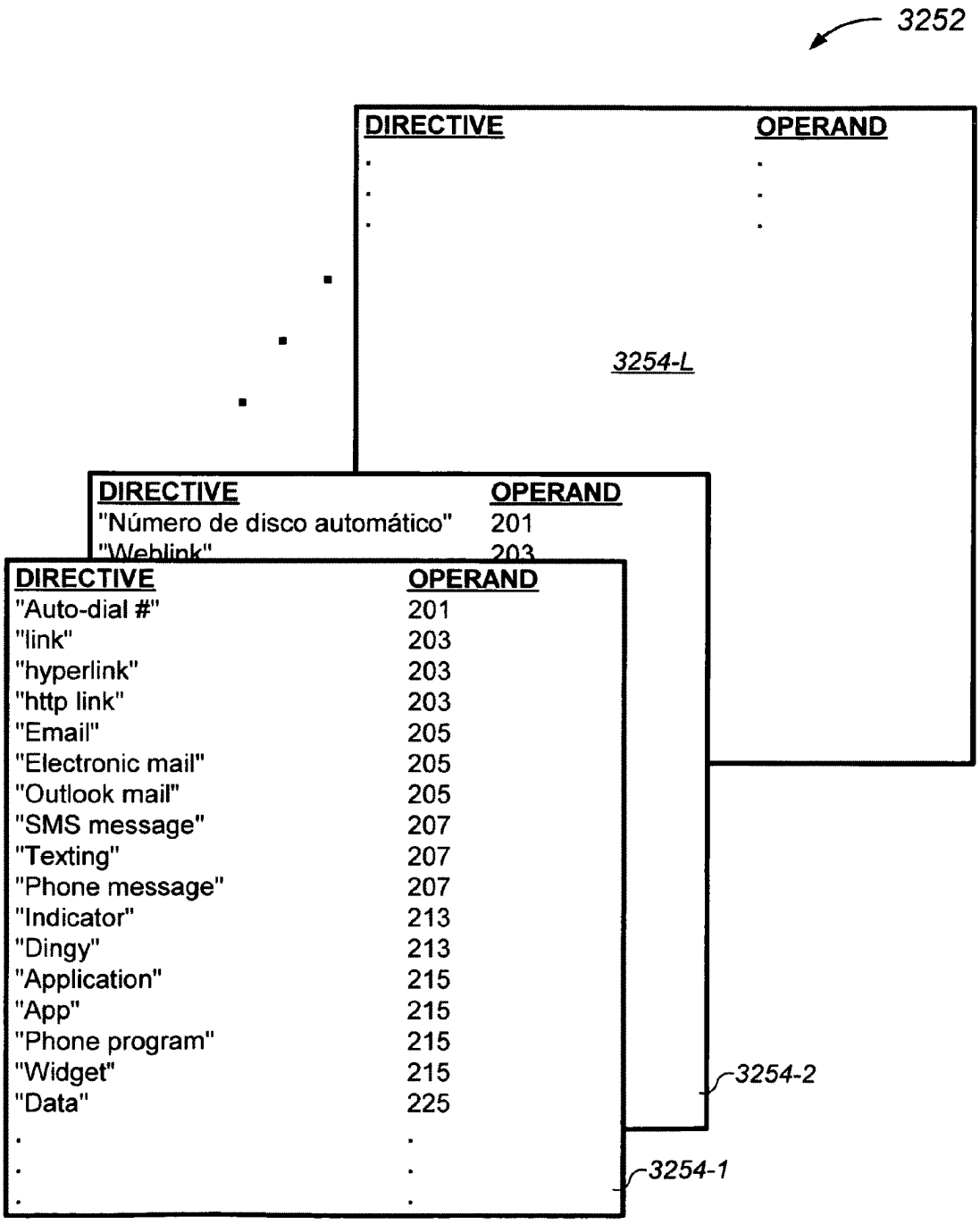

3252
DIRECTIVE
OPERAND
3254-L
DIRECTIVE
OPERAND
"Número de disco automático" 201
"Weblink" 203
DIRECTIVE
OPERAND
"Auto-dial #" 201
"link" 203
"hyperlink" 203
"http link" 203
"Email" 205
"Electronic mail" 205
"Outlook mail" 205
"SMS message" 207
"Texting" 207
"Phone message" 207
"Indicator" 213
"Dingy" 213
"Application" 215
"App" 215
"Phone program" 215
"Widget" 215
"Data" 225
3254-2
3254-1

Fig. 32B

Token	Length	Value
Variables[1]	L	complex ([Variable] ... [Variable]).
Variable	L	complex (First 2 bytes = VarType; VarName; value(s)); this can be present in any complex datastream for scope within current complex datastream thereafter and all descending contructs to it.
VarInstantiations[1]	L	complex ([VarInstatiate] ... [VarInstantiate]).
VarInstantiate	L	instantiation variable name and optional parameters; this can be subordinate to any other construct (e.g. {Description,8,{VarInstantiate,2,x}} where x is variable of string type (e.g. x = "Very long description text here"). Note the savings in TLV datastream size by using variables defined in 1 place for multiple subsequent instantiations thereafter).
VarName	L	text string for variable name.
Description	L	text string for description.
History	L	complex ([CreatorInfo] [ModifierInfo])
CreatorInfo	L	complex ([CreateDateTime] [CreatorID] [CreatorIDType] [CreatorAddr] [CreatorSysID] [CreatorSysType] [CreatorSysAddr]).
ModifierInfo	L	complex ([LastModifyDateTime] [LastModifyID] [LastModifyIDType] [LastModifyAddr] [LastModifySysID] [LastModifySysType] [LastModifySysAddr]).
CreateDateTime	L	date/time stamp (i.e. YYYYMMDDHHMMSS.12..J).
CreatorID	L	complex (ID).
CreatorIDType	1	Atomic element of IDType.
CreatorAddr	L	complex (Address).
CreatorSysID	L	Text string for system ID.
CreatorSysType	L	Text string for system type.
CreatorSysAddr	L	complex (Address).
LastModifyDateTime	L	date/time stamp.
LastModifyID	L	complex (ID).
LastModifyIDType	L	Atomic element of IDType.
LastModifyAddr	L	complex (Address).
LastModifySysID	L	Text string for system ID.
LastModifySysType	L	Text string for system type.
LastModifySysAddr	L	complex (Address).
ID	L	complex (first 2 bytes = length of the identifier;followed by identifier;[Description] [History]).
IDType	1	Atomic element of IDType.
Address	L	First 2 bytes = address type; next L-2 bytes = address info.
TimeSpec	L	First byte = spec type (stamp, period); Next bytes = the time spec(s) (e.g. preferably in syntax described).

Fig. 33A

Token	Length	Value
PermissionBody	L	complex ([Variables] [Permissions]).
Permissions[1]	L	complex ([Permission] ... [Permisson]).
Permission	L	complex (Grantor Grantee [Grants] [TimeSpec] [Description] [History]).
Grantor	L	complex (ID [IDType]).
Grantee	L	complex (ID [IDType]).
Grants[1]	L	complex ([Grant] ... [Grant] | [Privileges]).
Grant	L	First 2 bytes = length of Grant name; following bytes = grant name string; then complex: (Privileges [TimeSpec] [Description] [History] | Grants [TimeSpec] [Description] [History]).
Privileges[1]	L	complex ([Privilege] ... [Privilege]).
Privilege	L	first 4 bytes = unsigned integer for atomic privilege assigned; following bytes (L-4) are complex: ([MSRelevance] [TimeSpec] [Description] [History]).
MSRelevance	8	64 bits for up to 64 different MS types of different capabilities.
Groups[1]	L	complex ([Group] ... [Group]).
Group	L	First 2 bytes = length of Group name; following bytes = group name string; then complex: (IDs [Description] [History] | Groups [Description] [History]).
IDs[1]	L	complex (ID [IDType] ... ID [IDType]).

Fig. 33B

Token	Length	Value
CharterBody	L	complex ([Variables] [Charters]).
Charters[1]	L	complex ([Charter] ... [Charter]).
Charter	L	complex (Grantee Grantor Expression Actions [TimeSpec] [Description] [History]).
Expression	L	complex (Conditions [TimeSpec]).
Conditions	L	complex (Condition | Conditions CondOp Condition]).
CondOp	1	"&" or "|".
Condition	L	complex (Term Op Term [TimeSpec] [Description] [History] | Value [TimeSpec] [Description] [History] | Invocation [TimeSpec] [Description] [History]).
Term	L	complex (WDRTerm [TimeSpec] [Description] [History] | AppTerm [TimeSpec] [Description] [History] | Value [TimeSpec] [Description] [History] | Invocation [TimeSpec] [Description] [History]).
WDRTerm	L	first 2 bytes for WDR 1100 field/subfield length; following bytes are __name syntactical reference; then, if any, is complex = [Description] [History]).
AppTerm	L	first 2 bytes for Application field length; following bytes are Prefix_name syntactical reference; then, if any, is complex = [Description] [History]).
Value	L	first byte indicates Value type; following bytes (L-1), if any, is the "number", "text string", "value", "Boolean", "null", "atomic term" or complex = (Data | ID [IDType]).
Data	L	first byte = atomic element data type; L-1 following bytes are the data syntactical reference.
Invocation	L	first byte = atomic element data type; L-1 following bytes = atomic element invocation with optional parameters.
Op	2	the operator reference (not clarifier simply provides unique operator (e.g. = and != are two operators; ProfileMatch here too). Numeric values used instead of characters here.
Actions[1]	L	complex ([Action] ... [Action]).
Action	L	complex ([Host] Command Operand [Parameters] [TimeSpec] [Description] [History]).
Host	L	complex (ID [IDType]).
Command	2	the command reference.
Operand	2	the operand reference.
Parameters[1]	L	complex ([Parameter] ... [Parameter]).
Parameter	L	complex (WDRTerm [Description] [History] | AppTerm [Description] [History] | Value [Description] [History] | Invocation [Description] [History] ID [IDType] [Description] [History]).

Fig. 33C

```
//********* Grammar Common Definitions: *********
//
#define     TOKEN_LENGTH                    2
#define     LENGTH_LENGTH                   4


// #define  VARTYPE_x       Use Token Definitions for VarType

#define     IDTYPE_MSID                     11
#define     IDTYPE_MSGRPID                  12
#define     IDTYPE_USERID                   13
#define     IDTYPE_USERGRPID                14
#define     IDTYPE_LOGICAL                  15
            // e.g. ip address and socket; e.g. inetd.cfg invocation (e.g. 23.56.232.2:34002)
#define     IDTYPE_PHYSICAL                 16 // MS serial #

#define     ADDRTYPE_LOGICAL                21 // e.g. ip address
#define     ADDRTYPE_PHYSICAL               22 // e.g. MS serial #
#define     ADDRTYPE_POSTAL                 23
#define     ADDRTYPE_POINT                  24
#define     ADDRTYPE_SL                     25
#define     ADDRTYPE_2D                     26
#define     ADDRTYPE_3D                     27

#define     TIMESPECTYPE_STAMP              31
#define     TIMESPECTYPE_PERIOD             32
```

Fig. 34A

```
//********* Grammar Common Construct Token Definitions: *********
//
// #define     VARIABLES                   10001
#define        VARIABLE                    10002
// #define     VARINSTANTIATIONS           10003
#define        VARINSTANTIATE              10004
#define        VARNAME                     10005
#define        DESCRIPTION                 10006
#define        HISTORY                     10007
#define        CREATORINFO                 10008
#define        MODIFIERINFO                10009
#define        CREATEDATETIME              10010
#define        CREATORID                   10011
#define        CREATORIDTYPE               10012
#define        CREATORADDR                 10013
#define        CREATORSYSID                10014
#define        CREATORSYSTYPE              10015
#define        CREATORSYSADDR              10016
#define        LASTMODIFYDATETIME          10017
#define        LASTMODIFYID                10018
#define        LASTMODIFYIDTYPE            10019
#define        LASTMODIFYADDR              10020
#define        LASTMODIFYSYSID             10021
#define        LASTMODIFYSYSTYPE           10022
#define        LASTMODIFYSYSADDR           10023
#define        ID                          10024
#define        IDTYPE                      10025
#define        ADDRESS                     10026
#define        TIMESPEC                    10027

//********* Grammar Permission Construct Token Definitions: *********
//
#define        PERMISSIONBODY              12001
// #define     PERMISSIONS                 12002
#define        PERMISSION                  12003
#define        GRANTOR                     12004
#define        GRANTEE                     12005
#define        GRANTS                      12006
#define        GRANT                       12007
// #define     PRIVILEGES                  12008
#define        PRIVILEGE                   12009
#define        MSRELEVANCE                 12010
```

Fig. 34B

```
#define     GROUPS                          12011
#define     GROUP                           12012
#define     IDS                             12013

//********* Grammar Charter Construct Token Definitions: *********
//
#define     CHARTERBODY                     14001
// #define  CHARTERS                        14002
#define     CHARTER                         14003
#define     EXPRESSION                      14004
#define     CONDITIONS                      14005
#define     CONDOP                          14006
#define     CONDITION                       14007
#define     TERM                            14008
#define     WDRTERM                         14009
#define     APPTERM                         14010
#define     VALUE                           14011
#define     DATA                            14012
#define     INVOCATION                      14013
#define     OP                              14014
// #define  ACTIONS                         14015
#define     ACTION                          14016
#define     HOST                            14017
#define     COMMAND                         14018
#define     OPERAND                         14019
// #define  PARAMETERS                      14020
#define     PARAMETER                       14021

//********* Grammar Charter Definitions: *********
//
// atomic terms (e.g. \loc_my), WDR field terms (e.g. __location),
// Application terms (e.g. M_source), Invocation (e.g. fcn(p1,p2)), CondOp (e.g. "&") and
// Data atomic elements (e.g. c:\dir1\fname:58/LONGINT) are recognized syntaxes.
#define     VALUE_NUMBER                    41
#define     VALUE_TEXT                      42
#define     VALUE_ENUM                      43 // "value"
#define     VALUE_BOOLEAN                   44 // 1 = True, 0 = False
#define     VALUE_ID                        45
```

Fig. 34C

```
//********* Atomic Commands : *********
//
#define     CMD_SEND                    101
#define     CMD_NOTIFY                  103
#define     CMD_COMPOSE                 105
#define     CMD_FIND                    107
#define     CMD_INVOKE                  109
#define     CMD_COPY                    111
#define     CMD_DISCARD                 113
#define     CMD_MOVE                    115
#define     CMD_STORE                   117
#define     CMD_CONNECT                 119
#define     CMD_ADMINISTRATE            121
#define     CMD_CHANGE                  123

//********* Atomic Operands : *********
//
#define     OPERAND_AUTODIALNUMBER      201
#define     OPERAND_WEBLINK             203
#define     OPERAND_EMAIL               205
#define     OPERAND_SMSMSG              207
#define     OPERAND_BRDEMAIL            209
#define     OPERAND_BRDSMSMSG           211
#define     OPERAND_INDICATOR           213
#define     OPERAND_APP                 215
#define     OPERAND_DOCUMENT            217
#define     OPERAND_FILE                219
#define     OPERAND_CONTENT             221
#define     OPERAND_DBOBJ               223
#define     OPERAND_DATA                225
#define     OPERAND_SEMAPHORE           227
#define     OPERAND_DIRECTORY           229
#define     OPERAND_APPCONTEXT          231
#define     OPERAND_UIFOBJ              233
#define     OPERAND_UIFCTL              235
#define     OPERAND_INPUT               237
#define     OPERAND_OUTPUT              239
#define     OPERAND_ALERT               241
#define     OPERAND_PROC                243
#define     OPERAND_CONTAINER           245
#define     OPERAND_PROGOBJ             247
#define     OPERAND_CURSOR              249
#define     OPERAND_CALENDAR            251
#define     OPERAND_ADDRESSBOOK         253
```

Fig. 34D

```
// TIMESPEC date/time stamps for open ended periods are set with no start/end spec:
//  >=YYYMMDDHHMMSS.1..J ==> set x.endDT to DT_NOENDSPEC;
//  <=YYYMMDDHHMMSS.1..J ==> set x.startDT to DT_NOSTARTSPEC;
// <YYYMMDDHHMMSS.1..J and >YYYMMDDHHMMSS.1..J subtracts/adds min precision
// from specified date/time stamp (i.e. TIMESPEC periods are preferably inclusive).
#define DT_NOENDSPEC            -1.0
#define DT_NOSTARTSPEC          -2.0

typedef struct timespec { // specifications converted to a Julian period form
        double          startDT;    // converted to Julian format (1ms precision)
        double          endDT;      // converted to Julian format (1ms precision)
        struct timespec *nextTS;    // linked list of sibling timespecs
        } TIMESPEC;

typedef struct {
        double          *dt;        // Julian date/time
        unsigned char   id[MAX_IDLENGTH];
        unsigned short  idtype;     // for IDTYPE_x values
        // unsigned short cadr_type; // Assume 1 format here
        unsigned char   *c_address;
        char            *sysid;
        char            *systype;
        // unsigned short sysadr_type; // Assume 1 format here
        unsigned char   *sys_address;
        } BOOKKEEP;

typedef struct {
        BOOKKEEP *creation;
        BOOKKEEP *modify;
        } HISTRY;

typedef struct {
        unsigned short  vartype;
        char            name[MAX_VARNAME];
        unsigned char   *value;     // may be complex or series of complex
        } VAR;
```

Fig. 34E

```
typedef struct privilege {
        unsigned long       priv;          // constant value of known privilege
        unsigned char       relevance[MAX_RELEVANCEMASK];
        TIMESPEC            *tspec;
        char                *desc;
        HISTRY              *hist;
        struct privilege    *nextPriv;     // linked list of sibling privileges
        } PRIVILEGE;

typedef struct grant {
        char                name[MAX_GRNAMLENGTH];
        char                permtype;      // 'P' = Privilege(s), 'G' = Grant(s)
        union {
              struct grant  *grants;       // linked list subordinate/descending grant(s)
              PRIVILEGE     *privileges;   // linked list of privilege(s)
              } assigned;
        TIMESPEC            *tspec;
        char                *desc;
        HISTRY              *hist;
        struct grant        *nextGrant;    // linked list of sibling grants
        } GRANT;

typedef struct permission {
        unsigned char       grantor[MAX_IDLENGTH];
        unsigned short      gortype;              // for IDTYPE_x values
        unsigned char       grantee[MAX_IDLENGTH];
        unsigned short      geetype;              // for IDTYPE_x values
        char                permtype;             // 'P' = Privilege(s), 'G' = Grant(s)
        union {
              GRANT         *grants;              // linked list of grant(s)
              PRIVILEGE     *privileges;          // linked list of privilege(s)
              } assigned;
        TIMESPEC            *tspec;
        char                *desc;
        HISTRY              *hist;
        struct permission   *nextPerm;            // linked list of permissions
        } PERMISSION;
```

Fig. 34F

```
typedef struct identity {
        unsigned char       id[MAX_IDLENGTH];
        unsigned short      idtype;       // for IDTYPE_x values
        struct identity     *nextID;      // linked list of sibling IDs
        } IDENTITY;

typedef struct group {
        char                name[MAX_GRPNAMELENGTH];
        char                grptype;      // 'B' = Branch, 'L' = Leaf
        union {
              struct group  *groups;      // linked list subordinate/descending group(s)
              IDENTITY      *ids;         // linked list of IDs in this group
              } assigned;
        char                *desc;
        HISTRY              *hist;
        struct grant        *nextGroup;   // linked list of sibling groups
        } GROUP;

typedef struct action {
        IDENTITY            host;  // .idtype = 0 = no host spec)
        unsigned short      cmd;
        unsigned short      operand;
        unsigned char       *params;            // maintained in syntax for flexibility
                                                // and for stack processing
        TIMESPEC            *tspec;
        char                *desc;
        HISTRY              *hist;
        struct action       *nextActn;          // linked list of sibling actions
        } ACTION;

typedef struct charter {
        unsigned char       grantee[MAX_IDLENGTH];
        unsigned short      geetype;            // for IDTYPE_x values
        unsigned char       grantor[MAX_IDLENGTH];
        unsigned short      gortype;            // for IDTYPE_x values
        TIMESPEC            *exprTS;
        unsigned char       *cond;              // at least 1 condition maintained in
                                                // syntax for proper stack processing
        ACTION              *actn;
        char                *desc;
        HISTRY              *hist;
        struct charter      *nextCharter;       // linked list of charters
        } CHARTER;
```

Fig. 34G

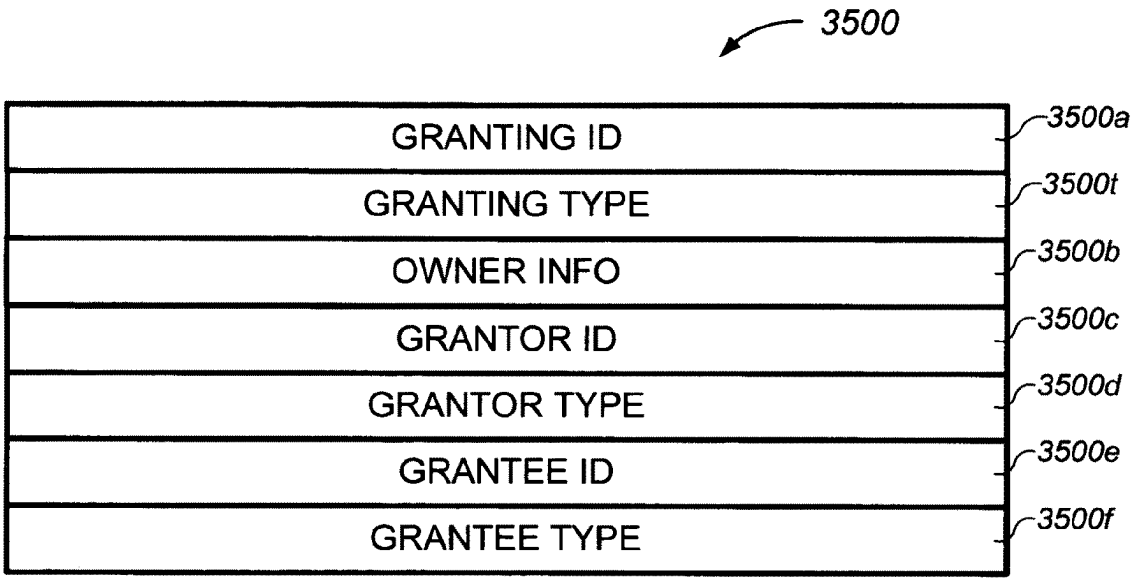

3500
GRANTING ID
3500a
GRANTING TYPE
3500t
OWNER INFO
3500b
GRANTOR ID
3500c
GRANTOR TYPE
3500d
GRANTEE ID
3500e
GRANTEE TYPE
3500f

Fig. 35A

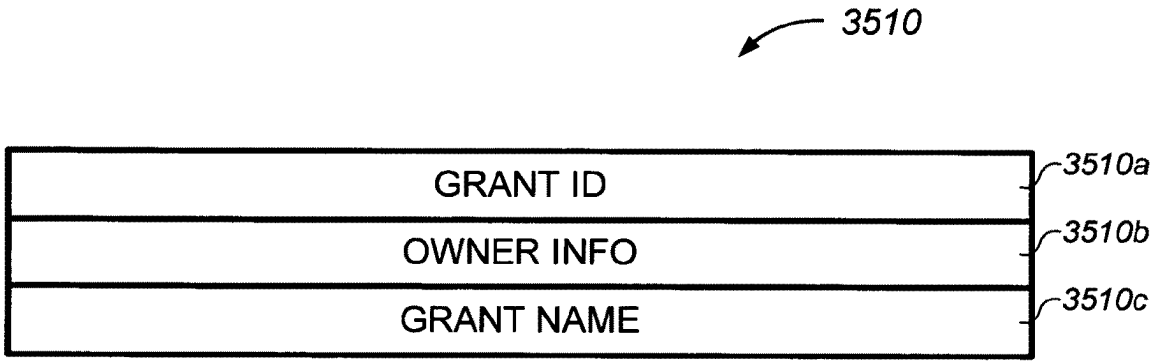


Fig. 35B

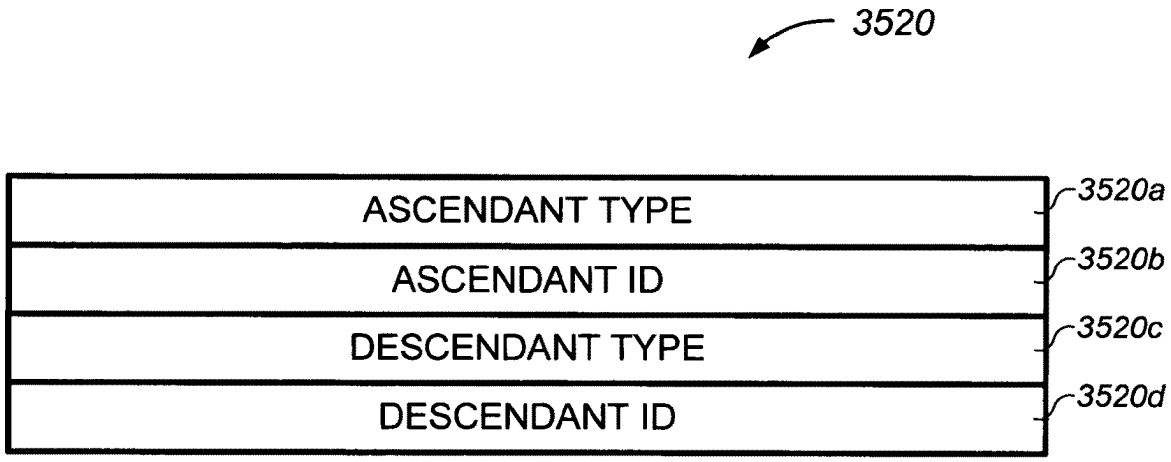

3520
ASCENDANT TYPE
3520a
ASCENDANT ID
3520b
DESCENDANT TYPE
3520c
DESCENDANT ID
3520d

Fig. 35C

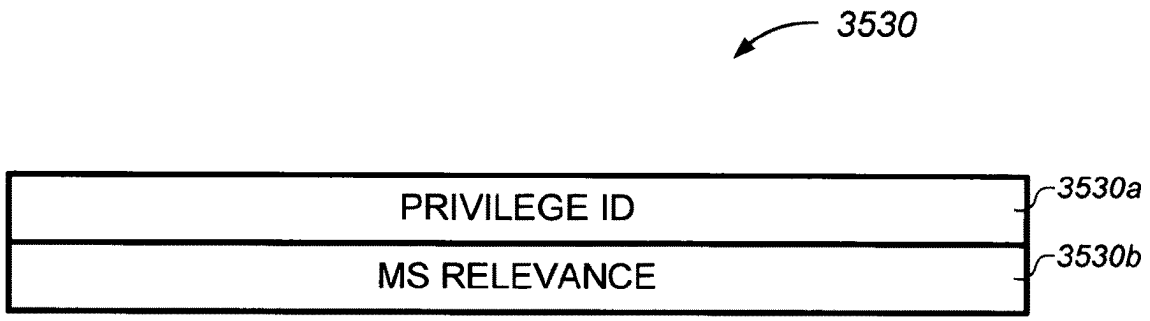


Fig. 35D

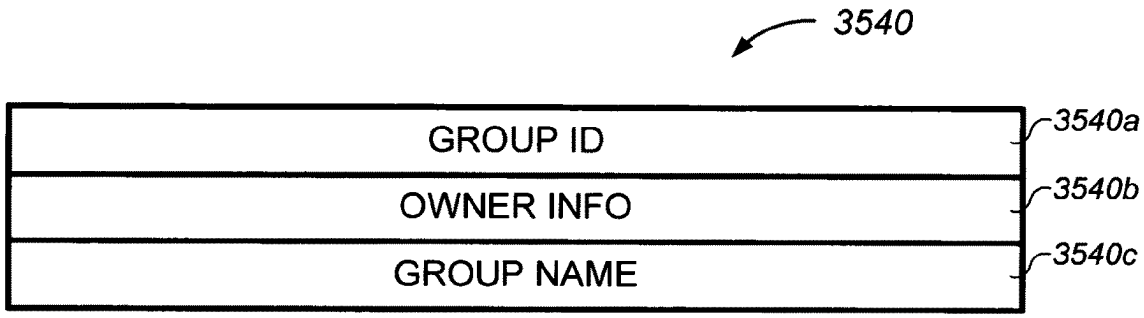


Fig. 35E

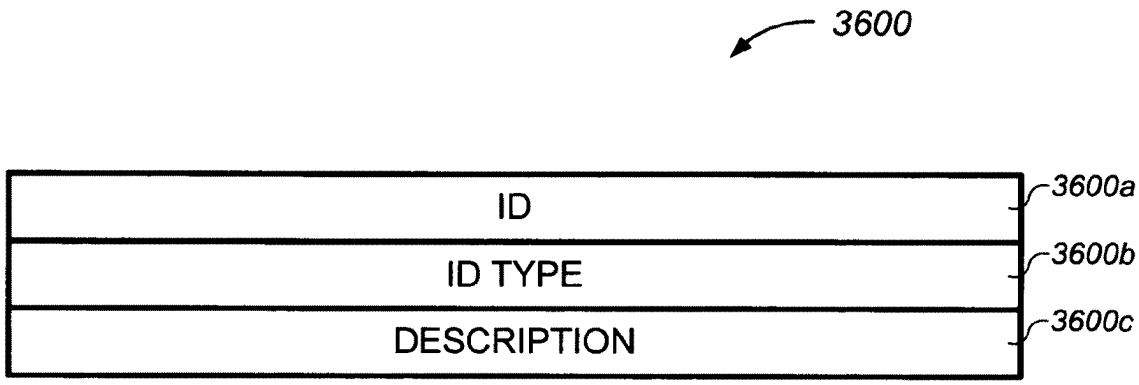


Fig. 36A

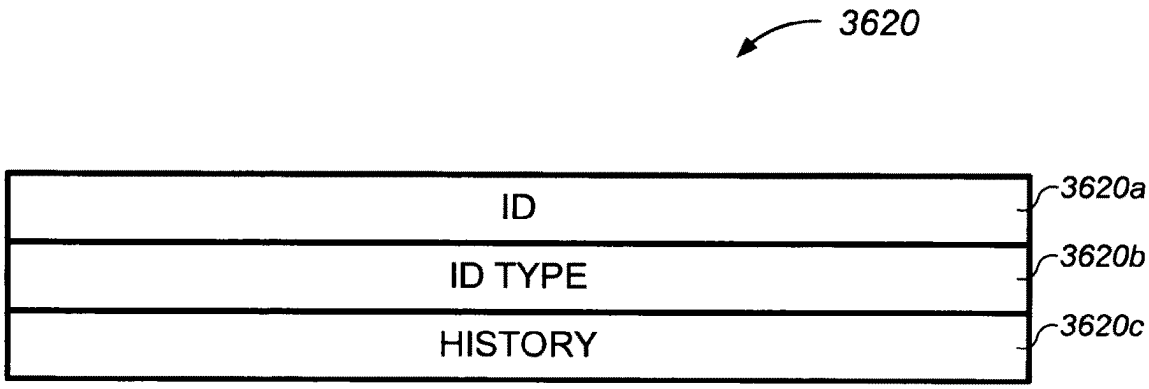


Fig. 36B

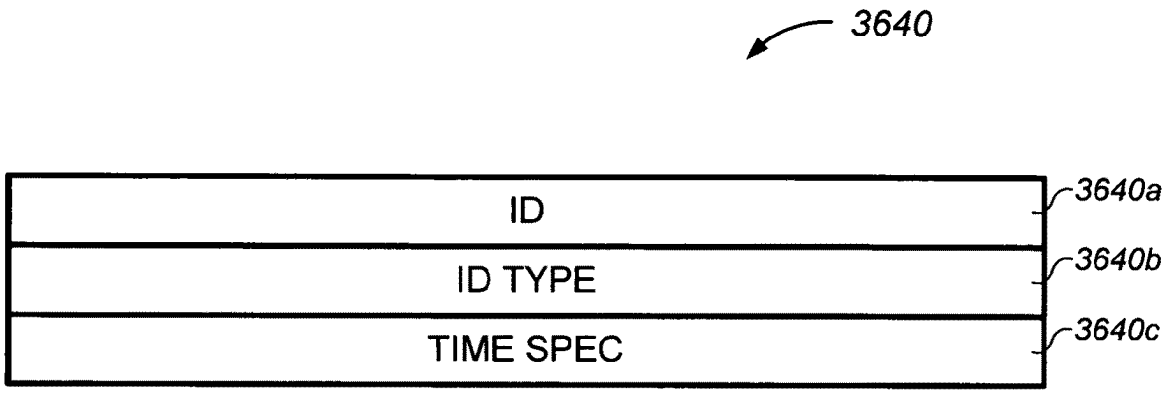


Fig. 36C

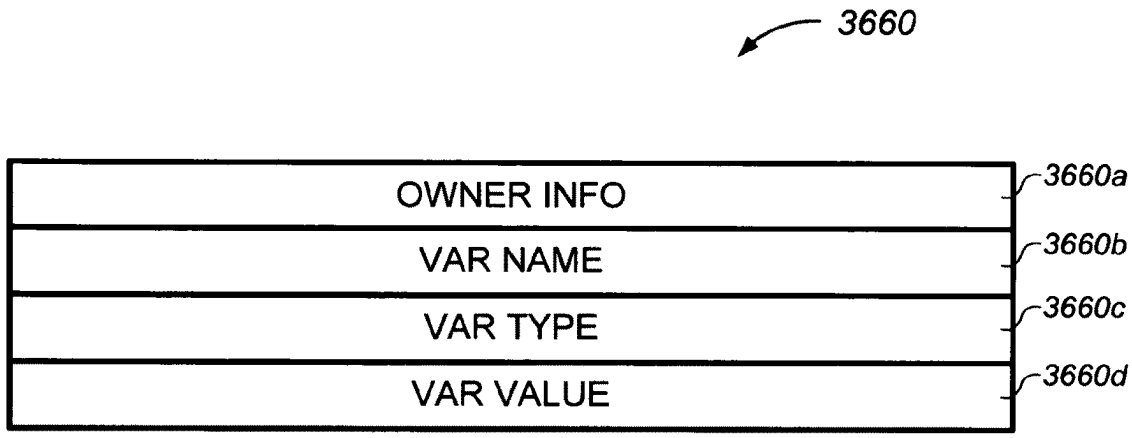

3660
OWNER INFO
VAR NAME
VAR TYPE
VAR VALUE
3660a
3660b
3660c
3660d

Fig. 36D

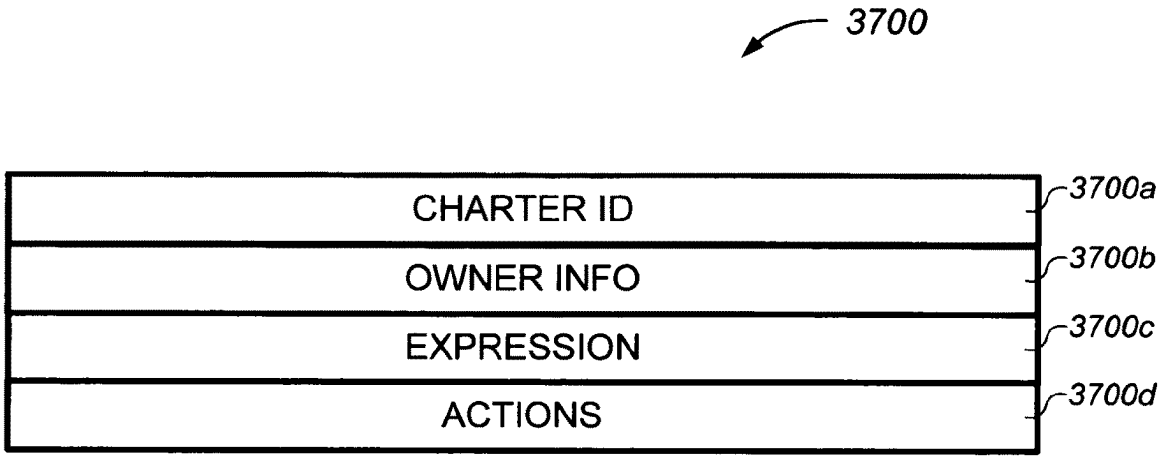

3700
CHARTER ID
OWNER INFO
EXPRESSION
ACTIONS
3700a
3700b
3700c
3700d

Fig. 37A

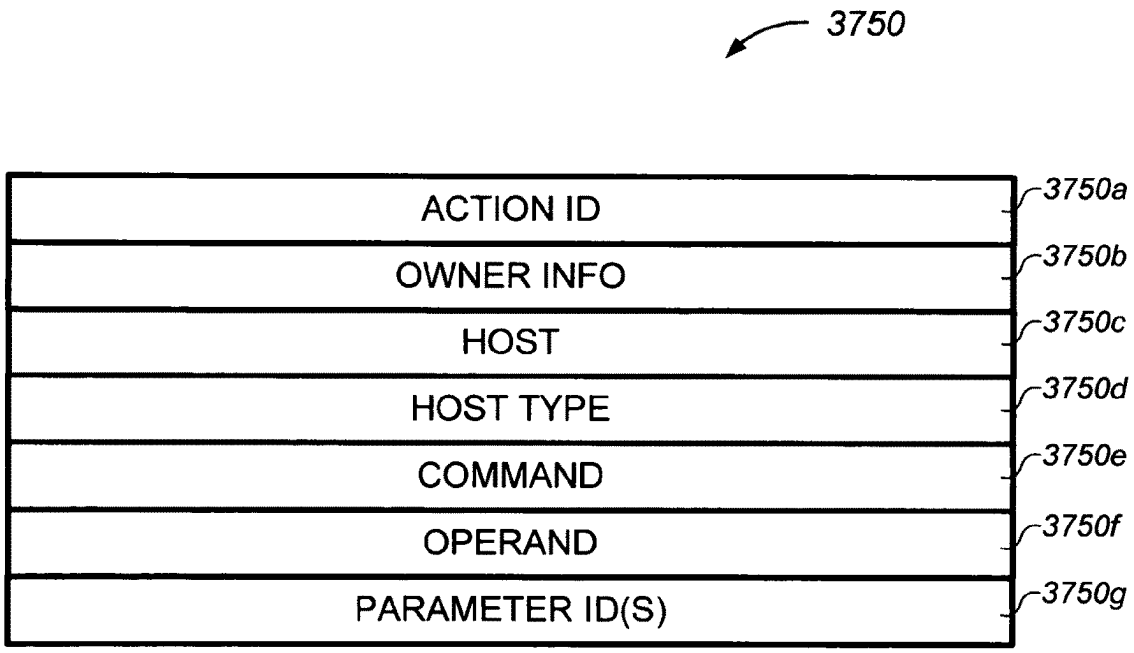

3750
ACTION ID
3750a
OWNER INFO
3750b
HOST
3750c
HOST TYPE
3750d
COMMAND
3750e
OPERAND
3750f
PARAMETER ID(S)
3750g

Fig. 37B

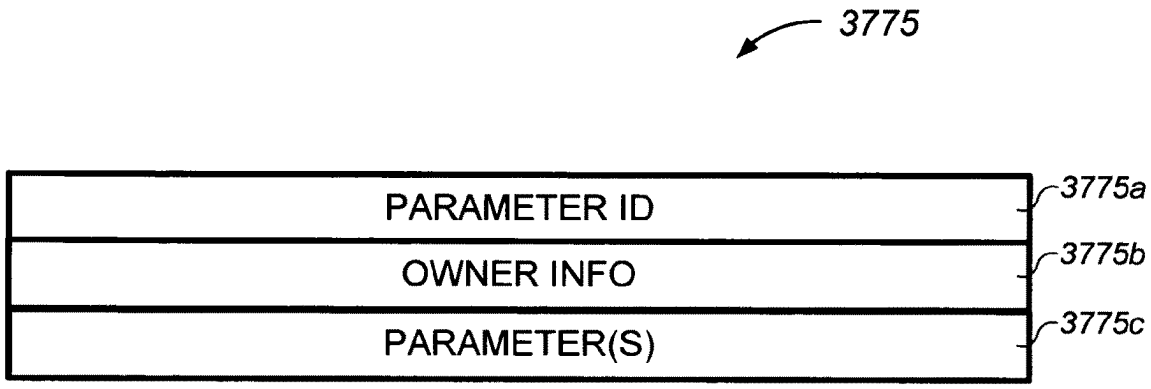


Fig. 37C

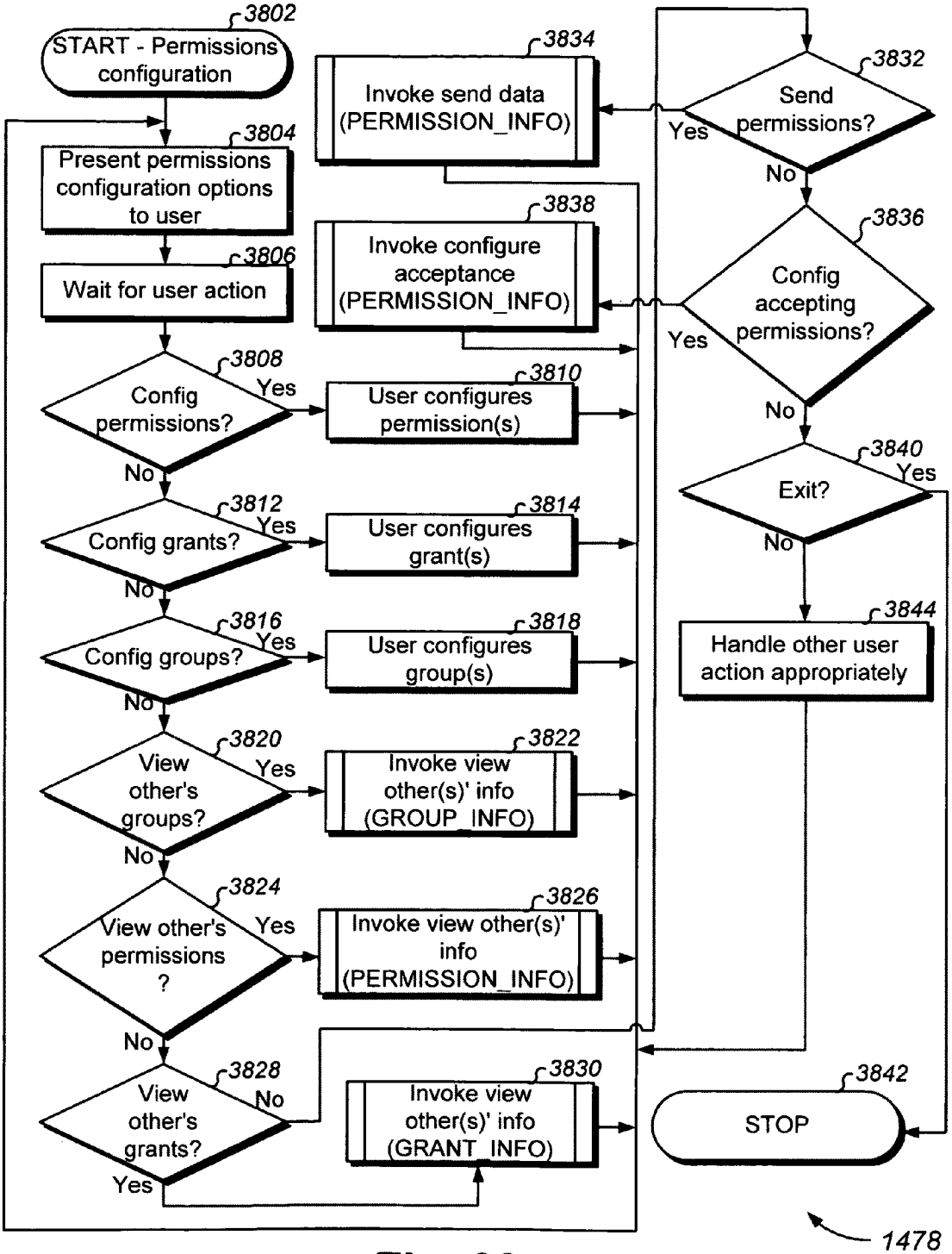

3802
START - Permissions configuration
3804
Present permissions configuration options to user
3806
Wait for user action
3808
Config permissions?
Yes
No
3810
User configures permission(s)
3812
Config grants?
Yes
No
3814
User configures grant(s)
3816
Config groups?
Yes
No
3818
User configures group(s)
3820
View other's groups?
Yes
No
3822
Invoke view other(s)' info (GROUP_INFO)
3824
View other's permissions ?
Yes
No
3826
Invoke view other(s)' info (PERMISSION_INFO)
3828
View other's grants?
No
Yes
3830
Invoke view other(s)' info (GRANT_INFO)
3834
Invoke send data (PERMISSION_INFO)
3832
Send permissions?
Yes
No
3838
Invoke configure acceptance (PERMISSION_INFO)
3836
Config accepting permissions?
Yes
No
3840
Exit?
Yes
No
3844
Handle other user action appropriately
3842
STOP
1478

Fig. 38

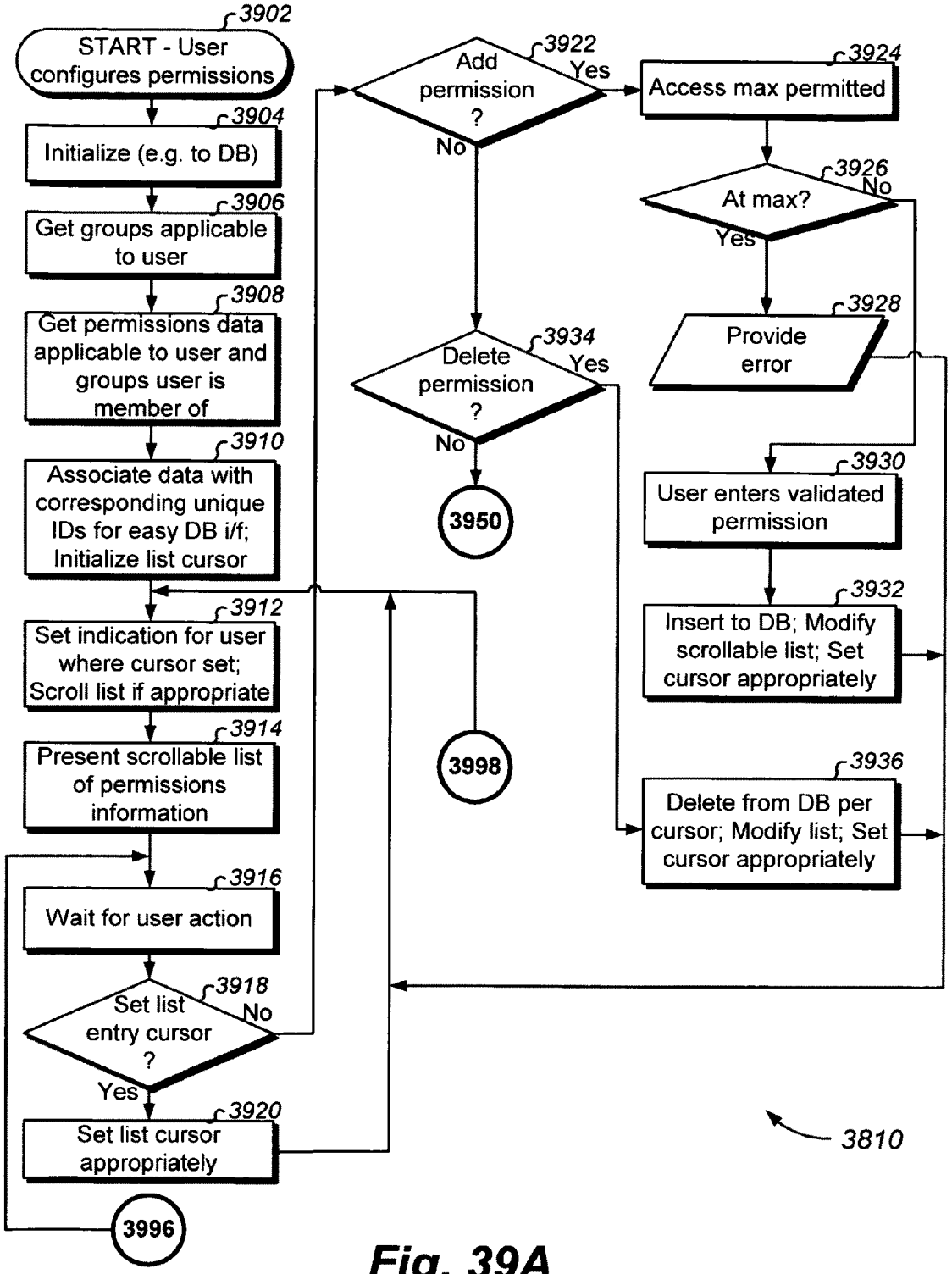

3902
START - User configures permissions
3904
Initialize (e.g. to DB)
3906
Get groups applicable to user
3908
Get permissions data applicable to user and groups user is member of
3910
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
3912
Set indication for user where cursor set; Scroll list if appropriate
3914
Present scrollable list of permissions information
3916
Wait for user action
3918
Set list entry cursor ?
No
Yes
3920
Set list cursor appropriately
3996
3922
Add permission ?
Yes
No
3924
Access max permitted
3926
At max?
No
Yes
3928
Provide error
3934
Delete permission ?
Yes
No
3950
3930
User enters validated permission
3932
Insert to DB; Modify scrollable list; Set cursor appropriately
3998
3936
Delete from DB per cursor; Modify list; Set cursor appropriately
3810

Fig. 39A

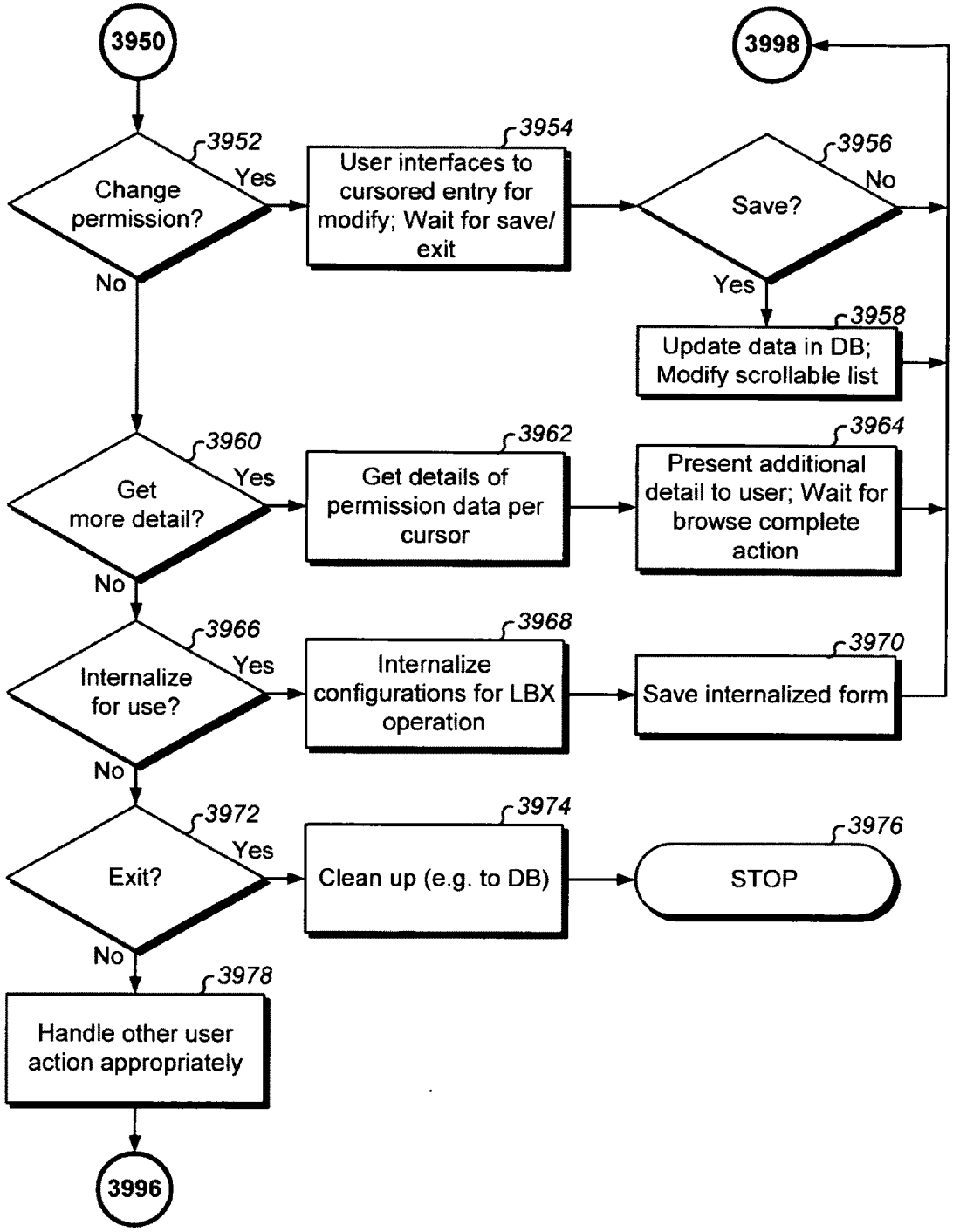

3950
3998
3952
Change permission?
Yes
No
3954
User interfaces to cursored entry for modify; Wait for save/ exit
3956
Save?
No
Yes
3958
Update data in DB; Modify scrollable list
3960
Get more detail?
Yes
No
3962
Get details of permission data per cursor
3964
Present additional detail to user; Wait for browse complete action
3966
Internalize for use?
Yes
No
3968
Internalize configurations for LBX operation
3970
Save internalized form
3972
Exit?
Yes
No
3974
Clean up (e.g. to DB)
3976
STOP
3978
Handle other user action appropriately
3996

Fig. 39B

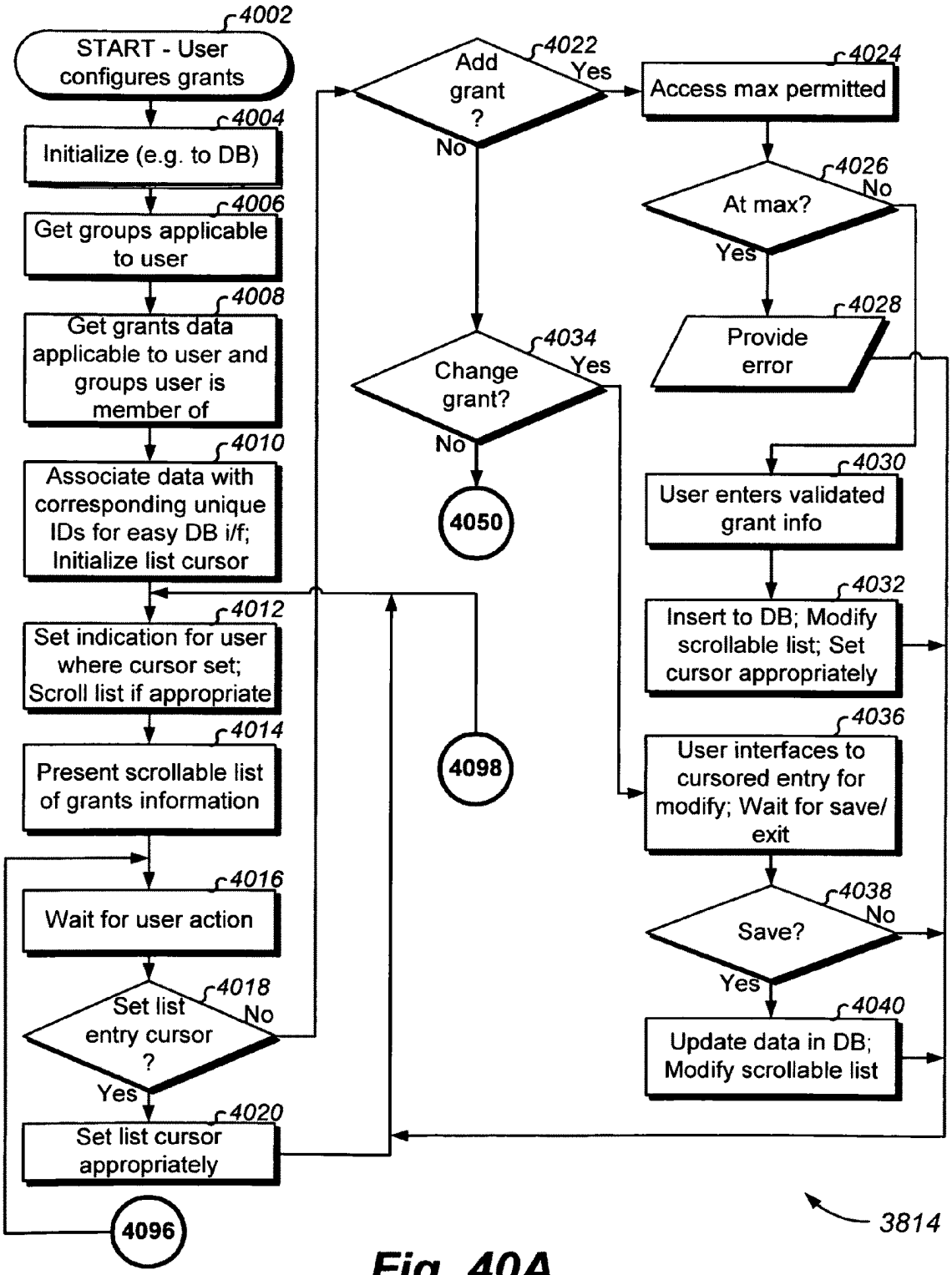

4002
START - User configures grants
4004
Initialize (e.g. to DB)
4006
Get groups applicable to user
4008
Get grants data applicable to user and groups user is member of
4010
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4012
Set indication for user where cursor set; Scroll list if appropriate
4014
Present scrollable list of grants information
4016
Wait for user action
4018
Set list entry cursor ?
No
Yes
4020
Set list cursor appropriately
4096
4022
Add grant ?
Yes
No
4024
Access max permitted
4026
At max?
No
Yes
4028
Provide error
4034
Change grant?
Yes
No
4050
4030
User enters validated grant info
4032
Insert to DB; Modify scrollable list; Set cursor appropriately
4036
User interfaces to cursored entry for modify; Wait for save/ exit
4038
Save?
No
Yes
4040
Update data in DB; Modify scrollable list
4098
3814

Fig. 40A

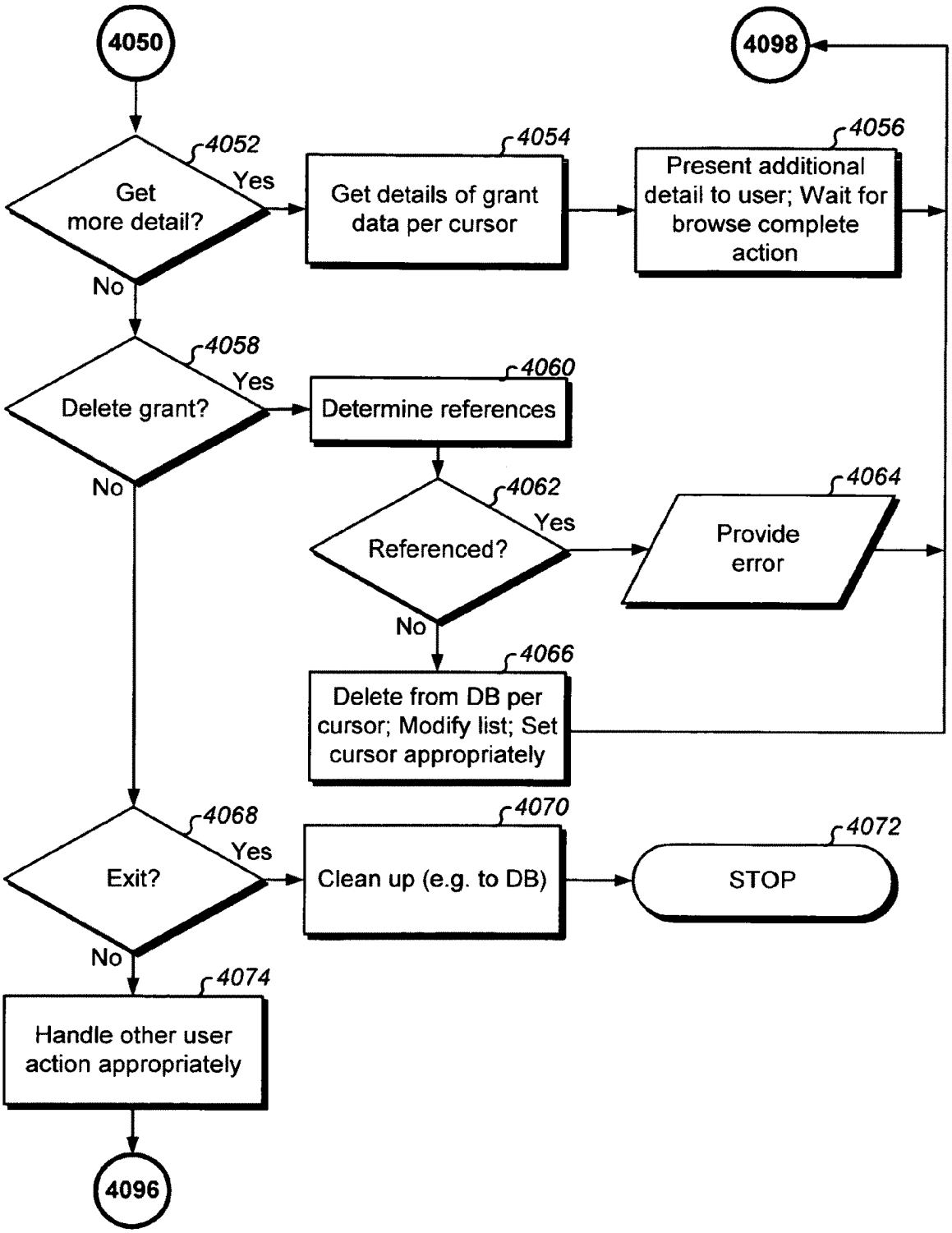

4050
4098
4052
Get more detail?
Yes
No
4054
Get details of grant data per cursor
4056
Present additional detail to user; Wait for browse complete action
4058
Delete grant?
Yes
No
4060
Determine references
4062
Referenced?
Yes
No
4064
Provide error
4066
Delete from DB per cursor; Modify list; Set cursor appropriately
4068
Exit?
Yes
No
4070
Clean up (e.g. to DB)
4072
STOP
4074
Handle other user action appropriately
4096

Fig. 40B

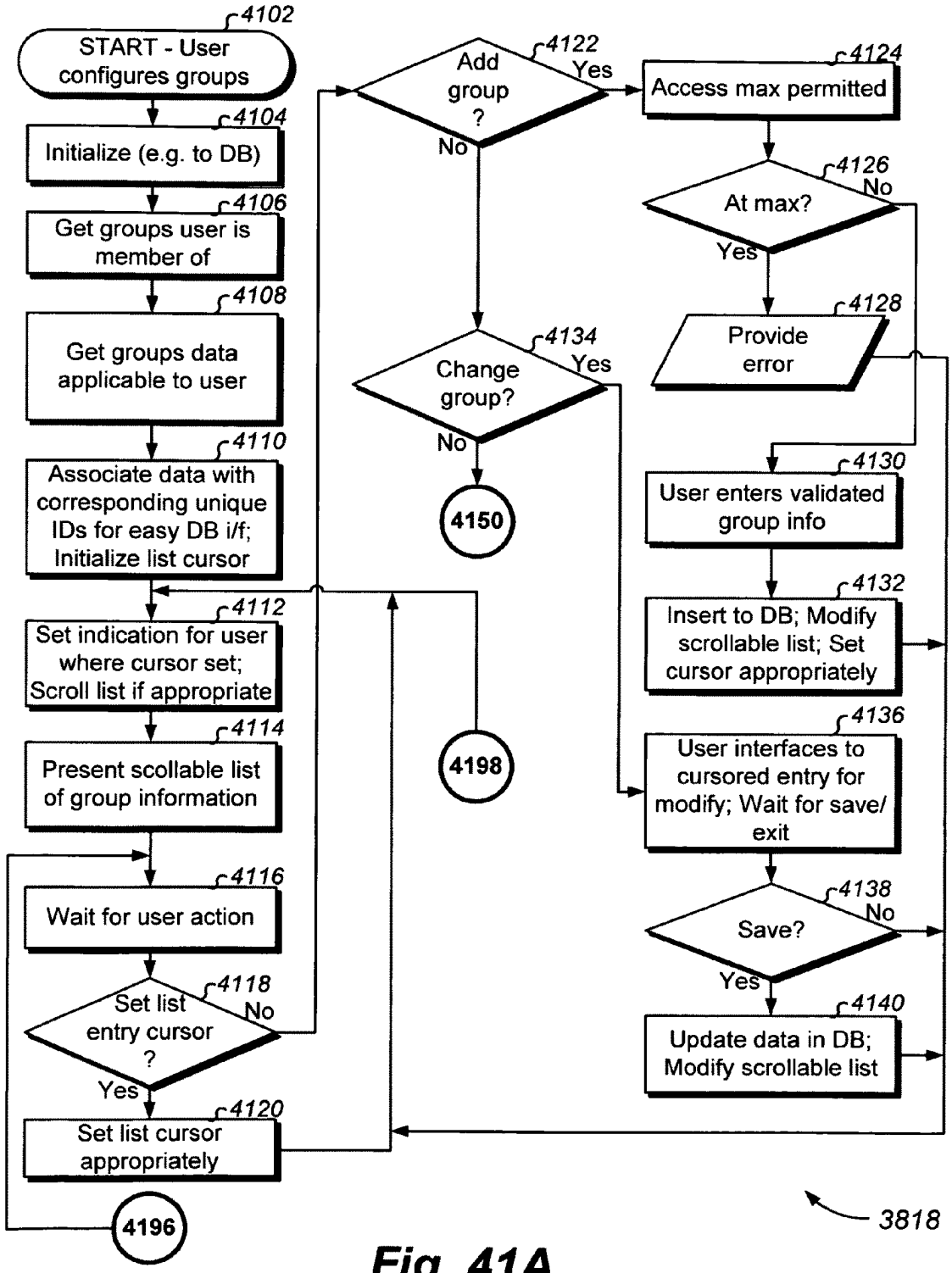

4102
START - User configures groups
4104
Initialize (e.g. to DB)
4106
Get groups user is member of
4108
Get groups data applicable to user
4110
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4112
Set indication for user where cursor set; Scroll list if appropriate
4114
Present scollable list of group information
4116
Wait for user action
4118
Set list entry cursor ?
No
Yes
4120
Set list cursor appropriately
4196
4122
Add group ?
Yes
No
4124
Access max permitted
4126
At max?
No
Yes
4128
Provide error
4134
Change group?
Yes
No
4150
4130
User enters validated group info
4132
Insert to DB; Modify scrollable list; Set cursor appropriately
4198
4136
User interfaces to cursored entry for modify; Wait for save/ exit
4138
Save?
No
Yes
4140
Update data in DB; Modify scrollable list
3818

Fig. 41A

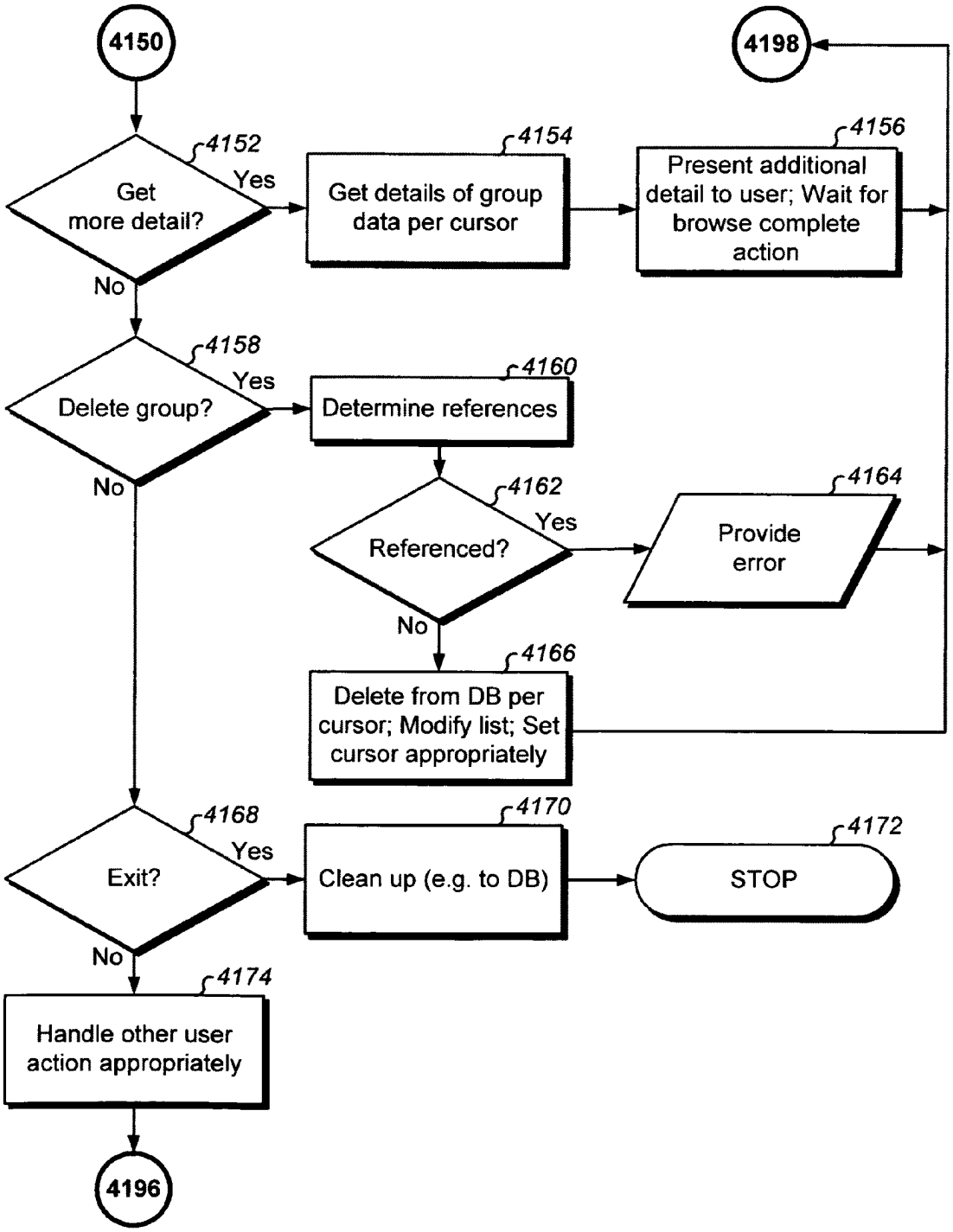

4150
4198
4152
Get more detail?
Yes
No
4154
Get details of group data per cursor
4156
Present additional detail to user; Wait for browse complete action
4158
Delete group?
Yes
No
4160
Determine references
4162
Referenced?
Yes
No
4164
Provide error
4166
Delete from DB per cursor; Modify list; Set cursor appropriately
4168
Exit?
Yes
No
4170
Clean up (e.g. to DB)
4172
STOP
4174
Handle other user action appropriately
4196

Fig. 41B

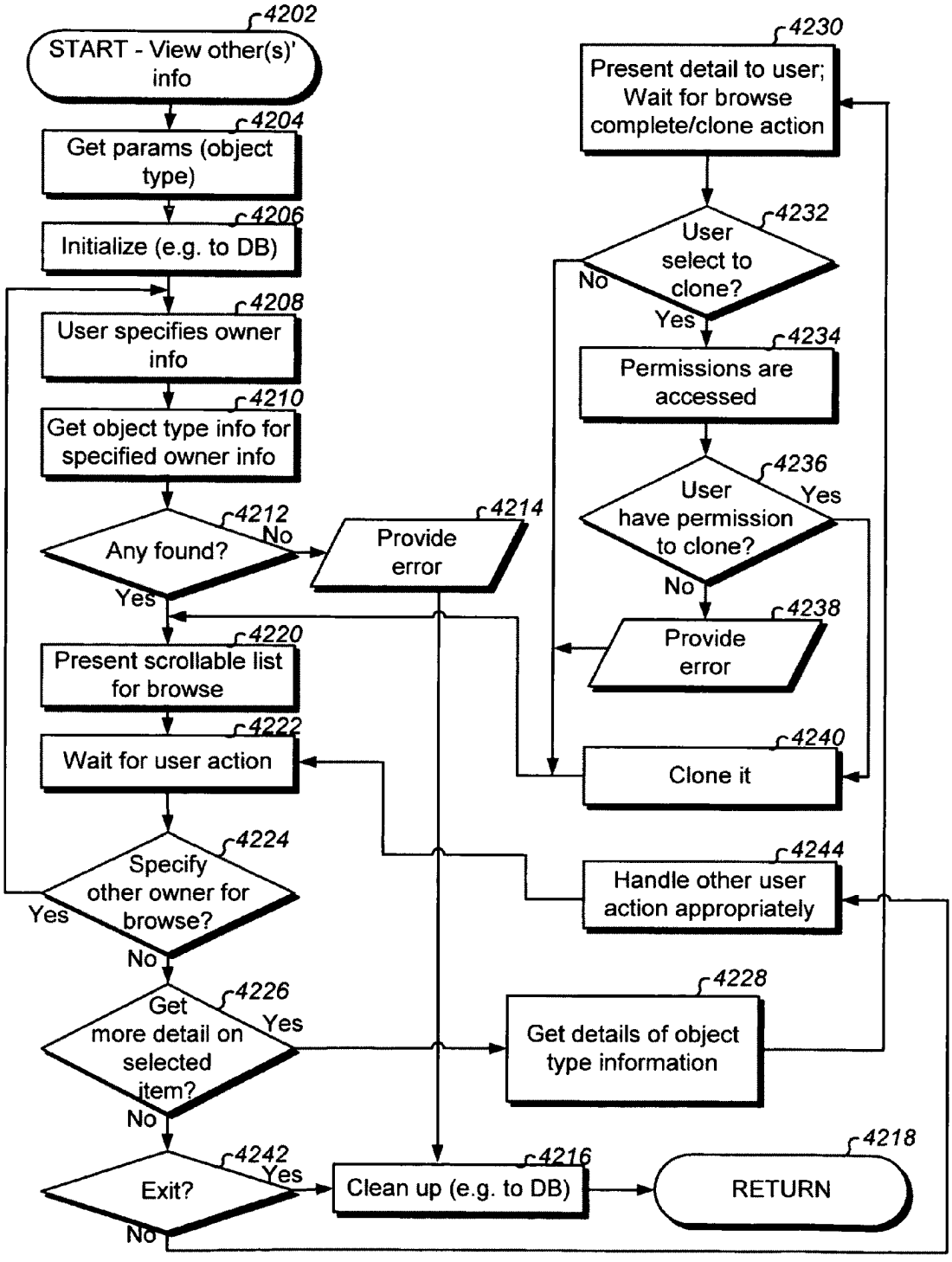

4202
START - View other(s)' info
4204
Get params (object type)
4206
Initialize (e.g. to DB)
4208
User specifies owner info
4210
Get object type info for specified owner info
4212
Any found?
No
Yes
4214
Provide error
4220
Present scrollable list for browse
4222
Wait for user action
4224
Specify other owner for browse?
Yes
No
4226
Get more detail on selected item?
Yes
No
4228
Get details of object type information
4242
Exit?
Yes
No
4216
Clean up (e.g. to DB)
4218
RETURN
4230
Present detail to user; Wait for browse complete/clone action
4232
User select to clone?
No
Yes
4234
Permissions are accessed
4236
User have permission to clone?
Yes
No
4238
Provide error
4240
Clone it
4244
Handle other user action appropriately

Fig. 42

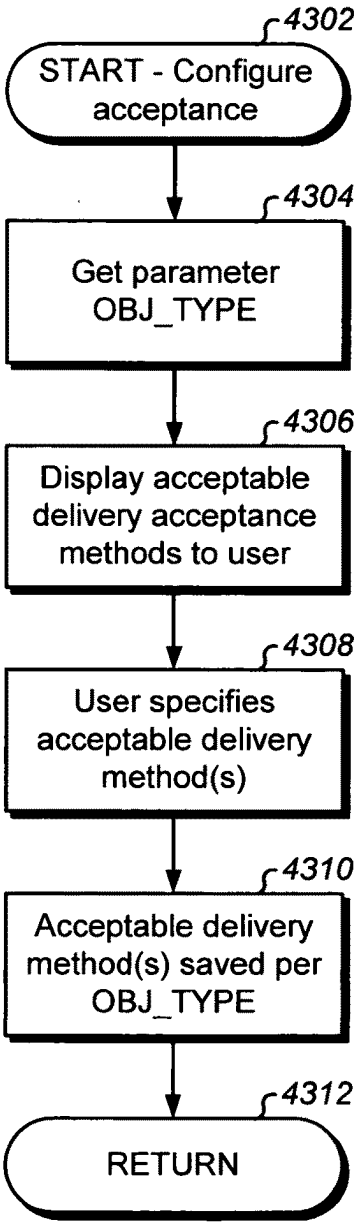

4302
START - Configure acceptance
4304
Get parameter OBJ_TYPE
4306
Display acceptable delivery acceptance methods to user
4308
User specifies acceptable delivery method(s)
4310
Acceptable delivery method(s) saved per OBJ_TYPE
4312
RETURN

Fig. 43

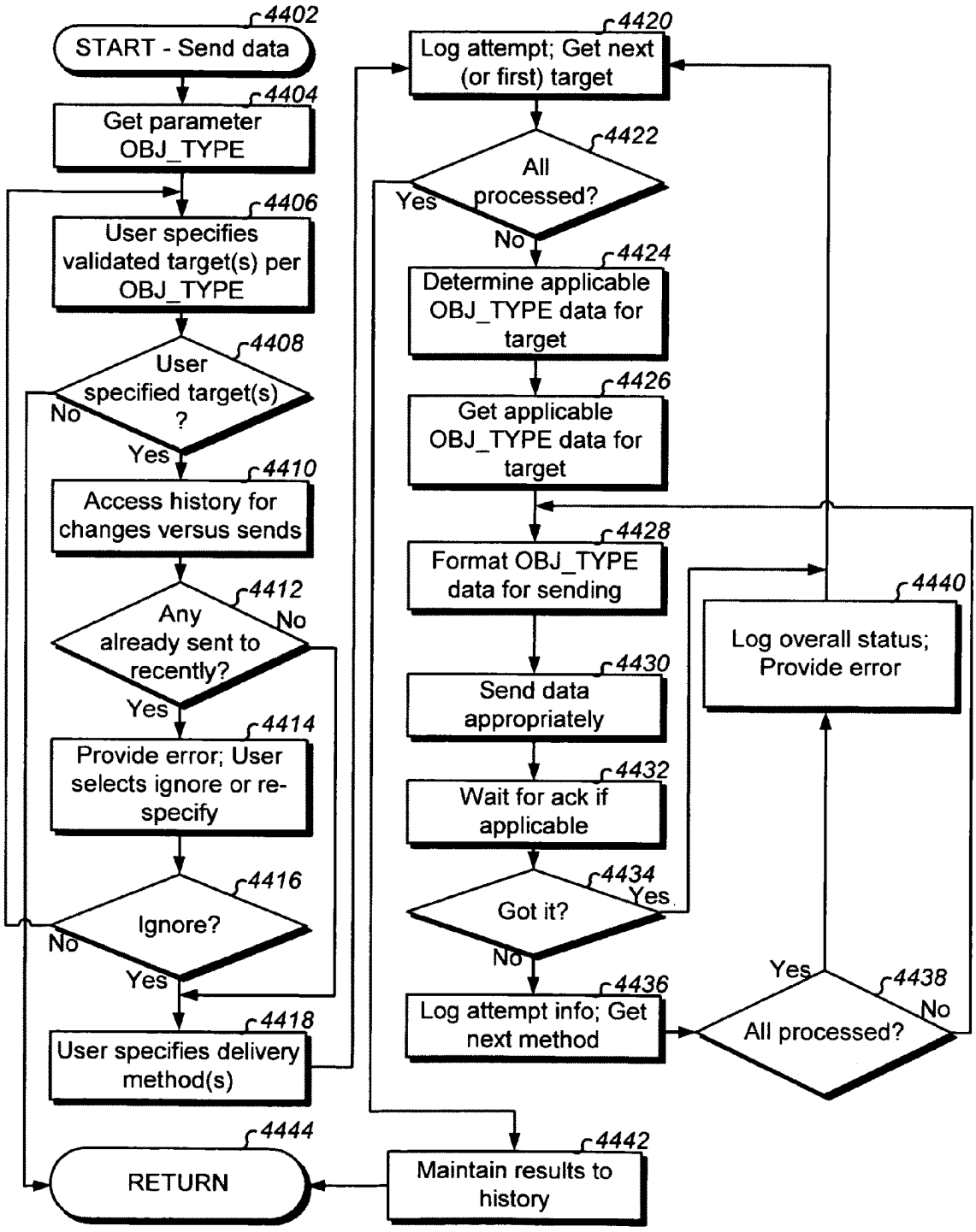

4402
START - Send data
4404
Get parameter OBJ_TYPE
4406
User specifies validated target(s) per OBJ_TYPE
4408
User specified target(s) ?
No
Yes
4410
Access history for changes versus sends
4412
Any already sent to recently?
No
Yes
4414
Provide error; User selects ignore or re-specify
4416
Ignore?
No
Yes
4418
User specifies delivery method(s)
4420
Log attempt; Get next (or first) target
4422
All processed?
Yes
No
4424
Determine applicable OBJ_TYPE data for target
4426
Get applicable OBJ_TYPE data for target
4428
Format OBJ_TYPE data for sending
4430
Send data appropriately
4432
Wait for ack if applicable
4434
Got it?
Yes
No
4436
Log attempt info; Get next method
4438
All processed?
Yes
No
4440
Log overall status; Provide error
4442
Maintain results to history
4444
RETURN

Fig. 44A

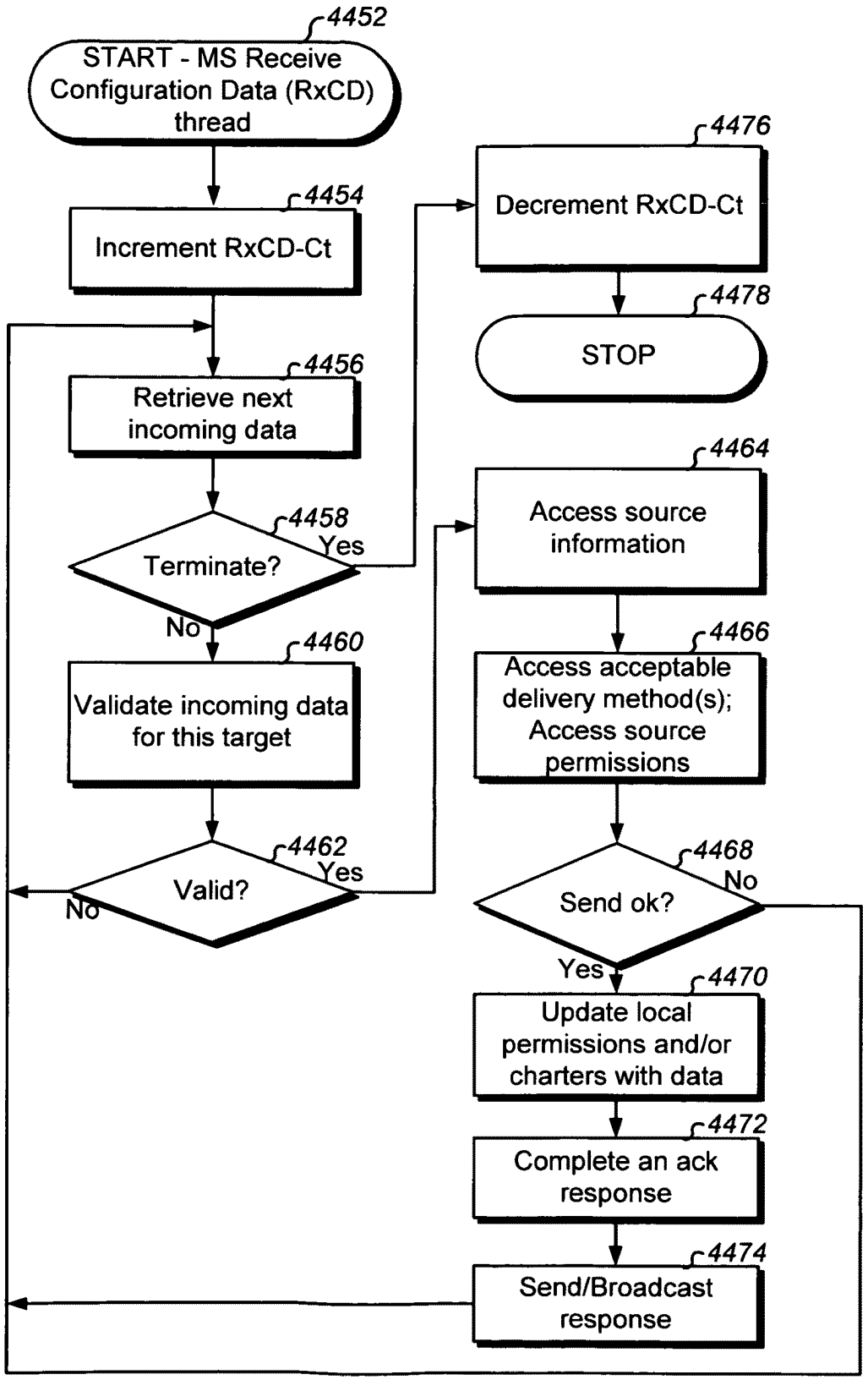

4452
START - MS Receive Configuration Data (RxCD) thread
4454
Increment RxCD-Ct
4456
Retrieve next incoming data
4458
Terminate?
Yes
No
4460
Validate incoming data for this target
4462
Valid?
Yes
No
4464
Access source information
4466
Access acceptable delivery method(s); Access source permissions
4468
Send ok?
No
Yes
4470
Update local permissions and/or charters with data
4472
Complete an ack response
4474
Send/Broadcast response
4476
Decrement RxCD-Ct
4478
STOP

Fig. 44B

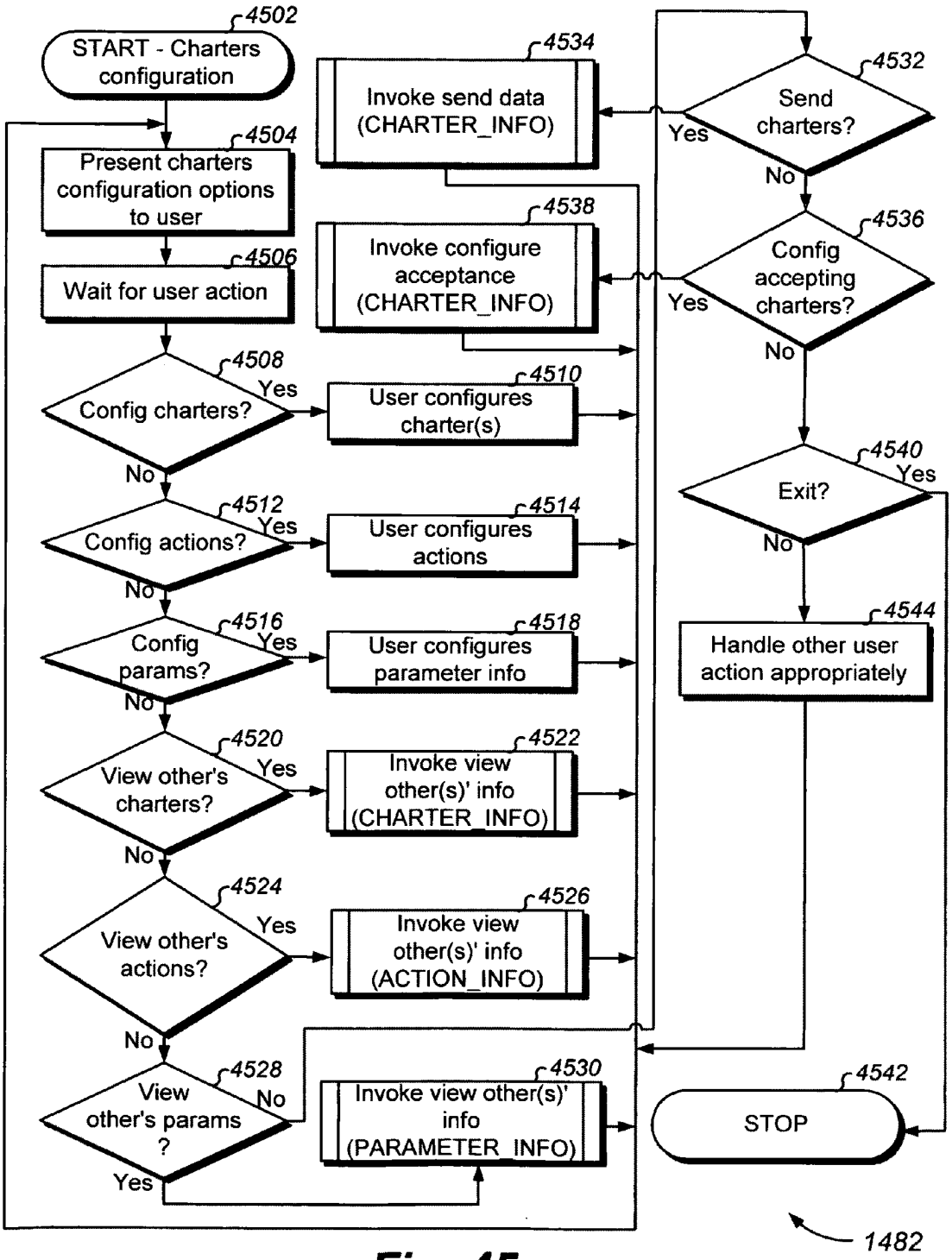

4502
START - Charters configuration
4504
Present charters configuration options to user
4506
Wait for user action
4508
Config charters?
Yes
No
4510
User configures charter(s)
4512
Config actions?
Yes
No
4514
User configures actions
4516
Config params?
Yes
No
4518
User configures parameter info
4520
View other's charters?
Yes
No
4522
Invoke view other(s)' info (CHARTER_INFO)
4524
View other's actions?
Yes
No
4526
Invoke view other(s)' info (ACTION_INFO)
4528
View other's params ?
No
Yes
4530
Invoke view other(s)' info (PARAMETER_INFO)
4534
Invoke send data (CHARTER_INFO)
4532
Send charters?
Yes
No
4538
Invoke configure acceptance (CHARTER_INFO)
4536
Config accepting charters?
Yes
No
4540
Exit?
Yes
No
4544
Handle other user action appropriately
4542
STOP
1482

Fig. 45

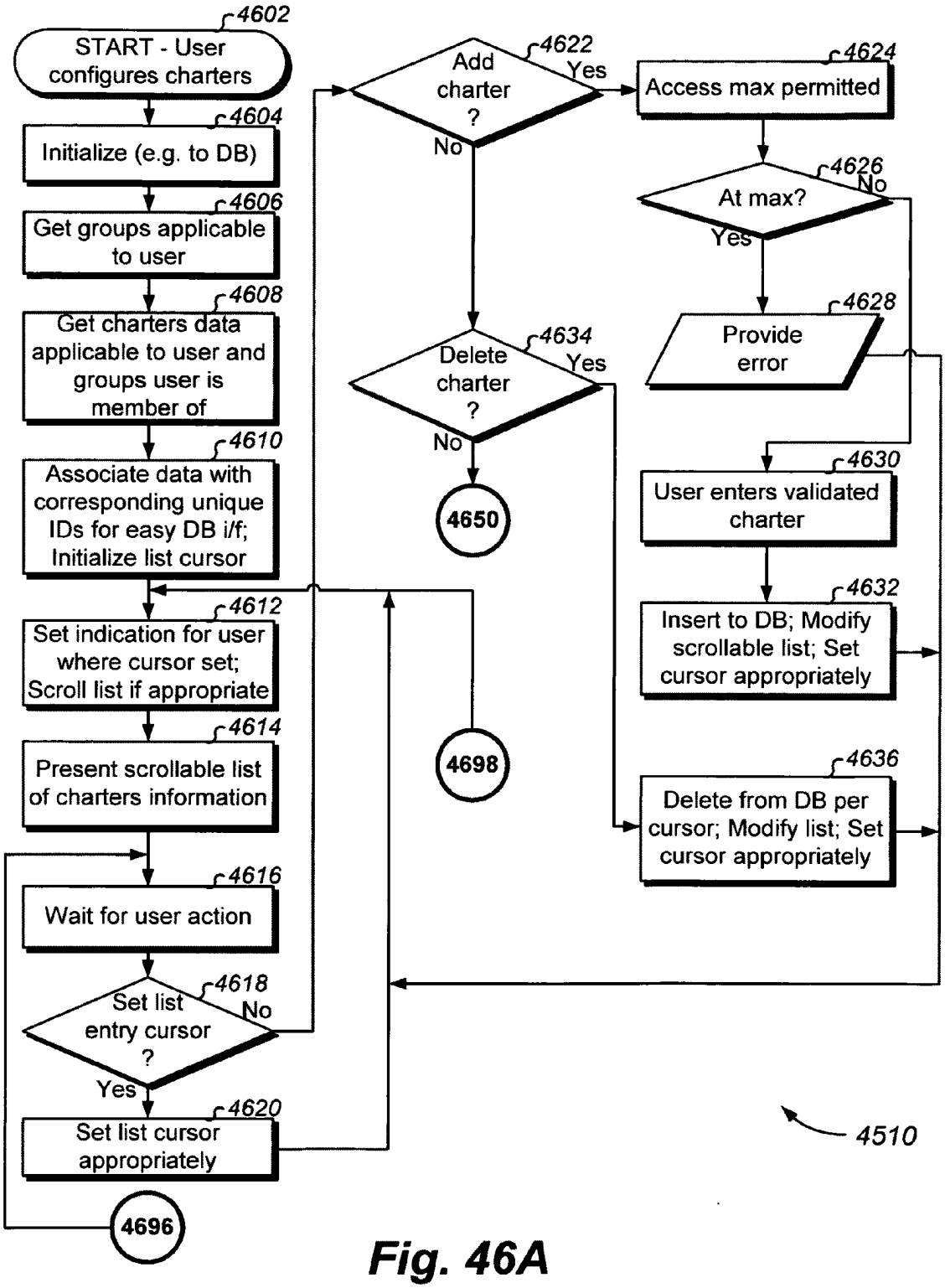

4602
START - User configures charters
4604
Initialize (e.g. to DB)
4606
Get groups applicable to user
4608
Get charters data applicable to user and groups user is member of
4610
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4612
Set indication for user where cursor set; Scroll list if appropriate
4614
Present scrollable list of charters information
4616
Wait for user action
4618
Set list entry cursor ?
No
Yes
4620
Set list cursor appropriately
4696
4622
Add charter ?
Yes
No
4624
Access max permitted
4626
At max?
No
Yes
4628
Provide error
4634
Delete charter ?
Yes
No
4650
4630
User enters validated charter
4632
Insert to DB; Modify scrollable list; Set cursor appropriately
4636
Delete from DB per cursor; Modify list; Set cursor appropriately
4698
4510

Fig. 46A

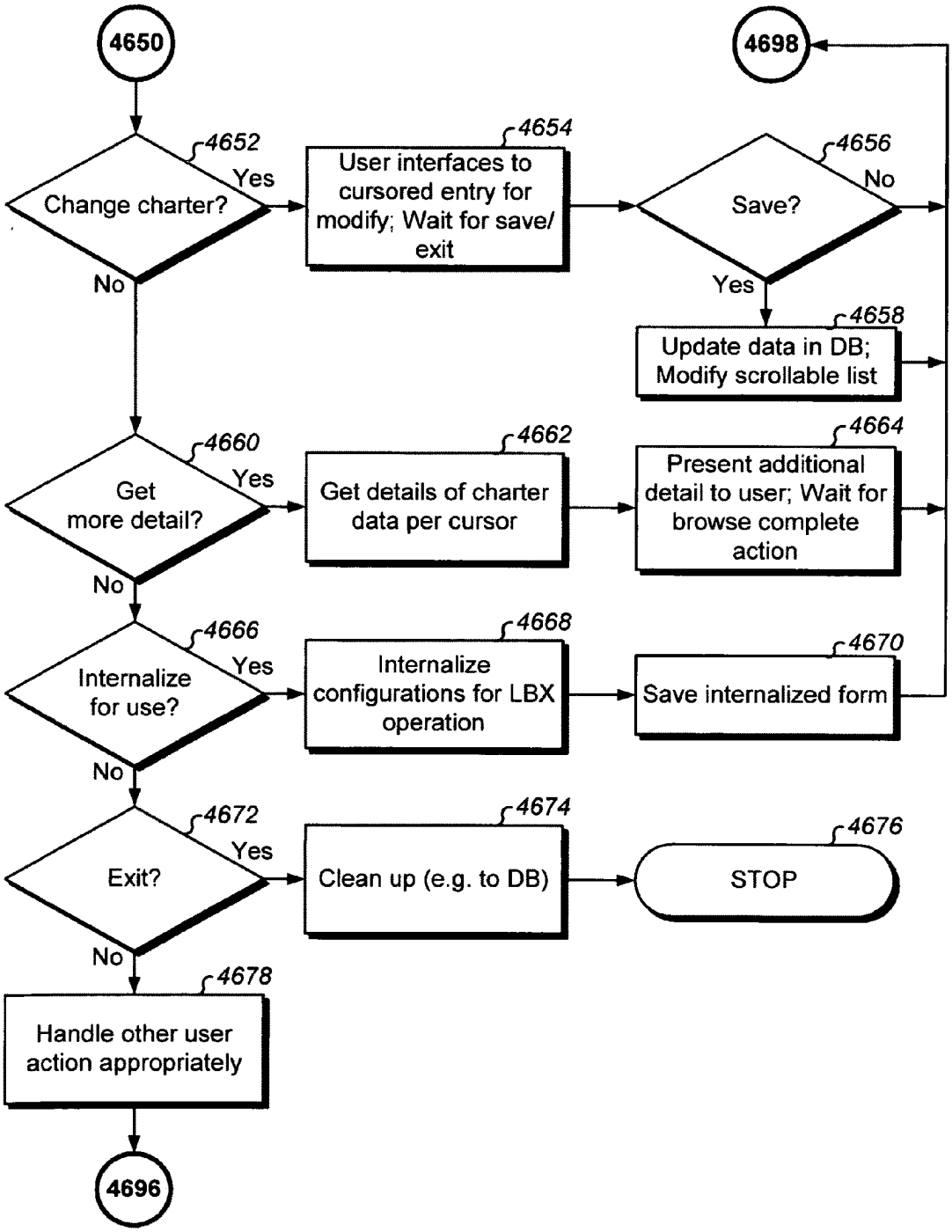

4650
4652
Change charter?
Yes
No
4654
User interfaces to cursored entry for modify; Wait for save/ exit
4656
Save?
No
Yes
4658
Update data in DB; Modify scrollable list
4698
4660
Get more detail?
Yes
No
4662
Get details of charter data per cursor
4664
Present additional detail to user; Wait for browse complete action
4666
Internalize for use?
Yes
No
4668
Internalize configurations for LBX operation
4670
Save internalized form
4672
Exit?
Yes
No
4674
Clean up (e.g. to DB)
4676
STOP
4678
Handle other user action appropriately
4696

Fig. 46B

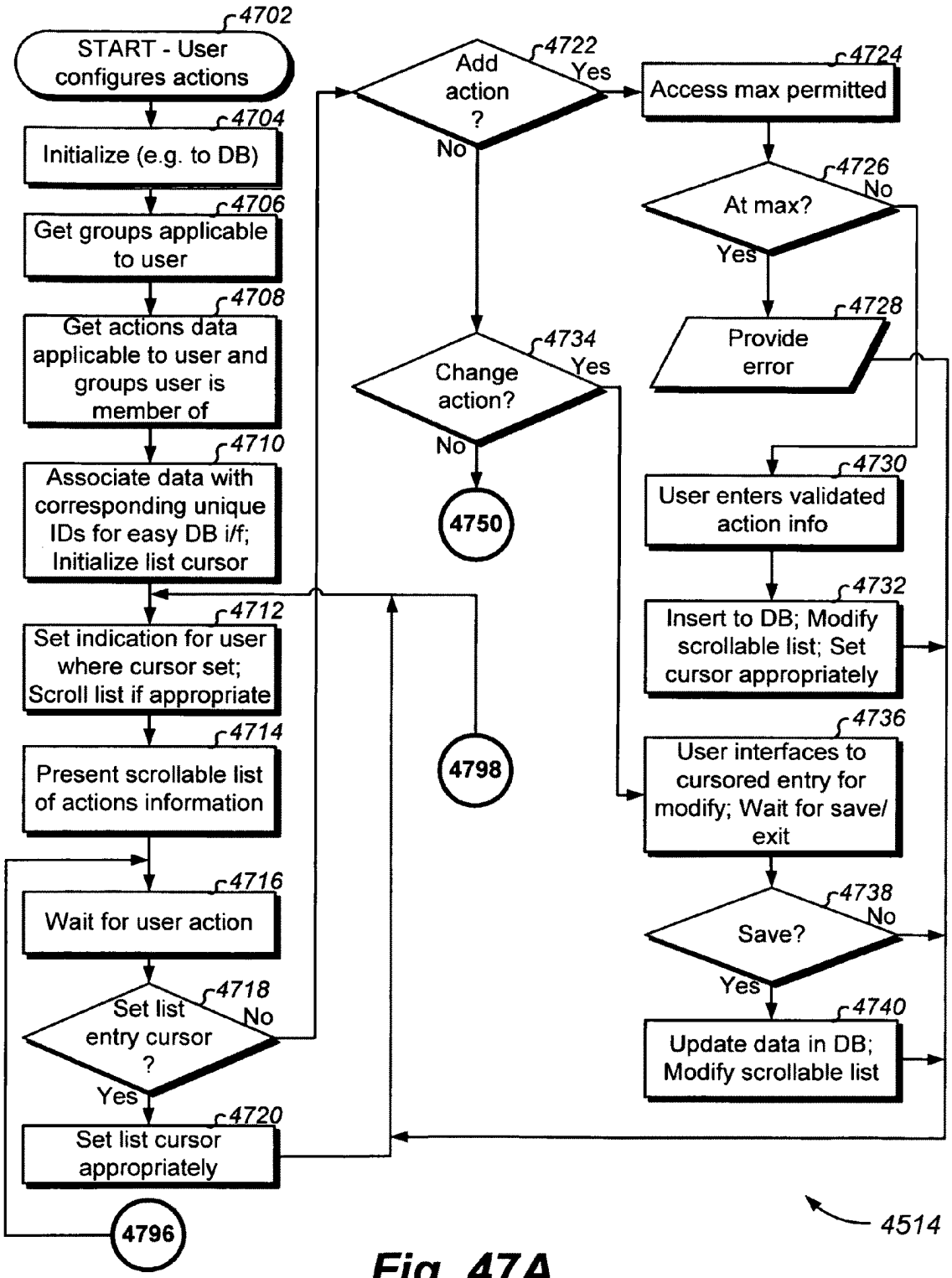

4702
START - User configures actions
4704
Initialize (e.g. to DB)
4706
Get groups applicable to user
4708
Get actions data applicable to user and groups user is member of
4710
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4712
Set indication for user where cursor set; Scroll list if appropriate
4714
Present scrollable list of actions information
4716
Wait for user action
4718
Set list entry cursor ?
No
Yes
4720
Set list cursor appropriately
4796
4722
Add action ?
Yes
No
4724
Access max permitted
4726
At max?
No
Yes
4728
Provide error
4734
Change action?
Yes
No
4750
4730
User enters validated action info
4732
Insert to DB; Modify scrollable list; Set cursor appropriately
4736
User interfaces to cursored entry for modify; Wait for save/ exit
4738
Save?
No
Yes
4740
Update data in DB; Modify scrollable list
4798
4514

Fig. 47A

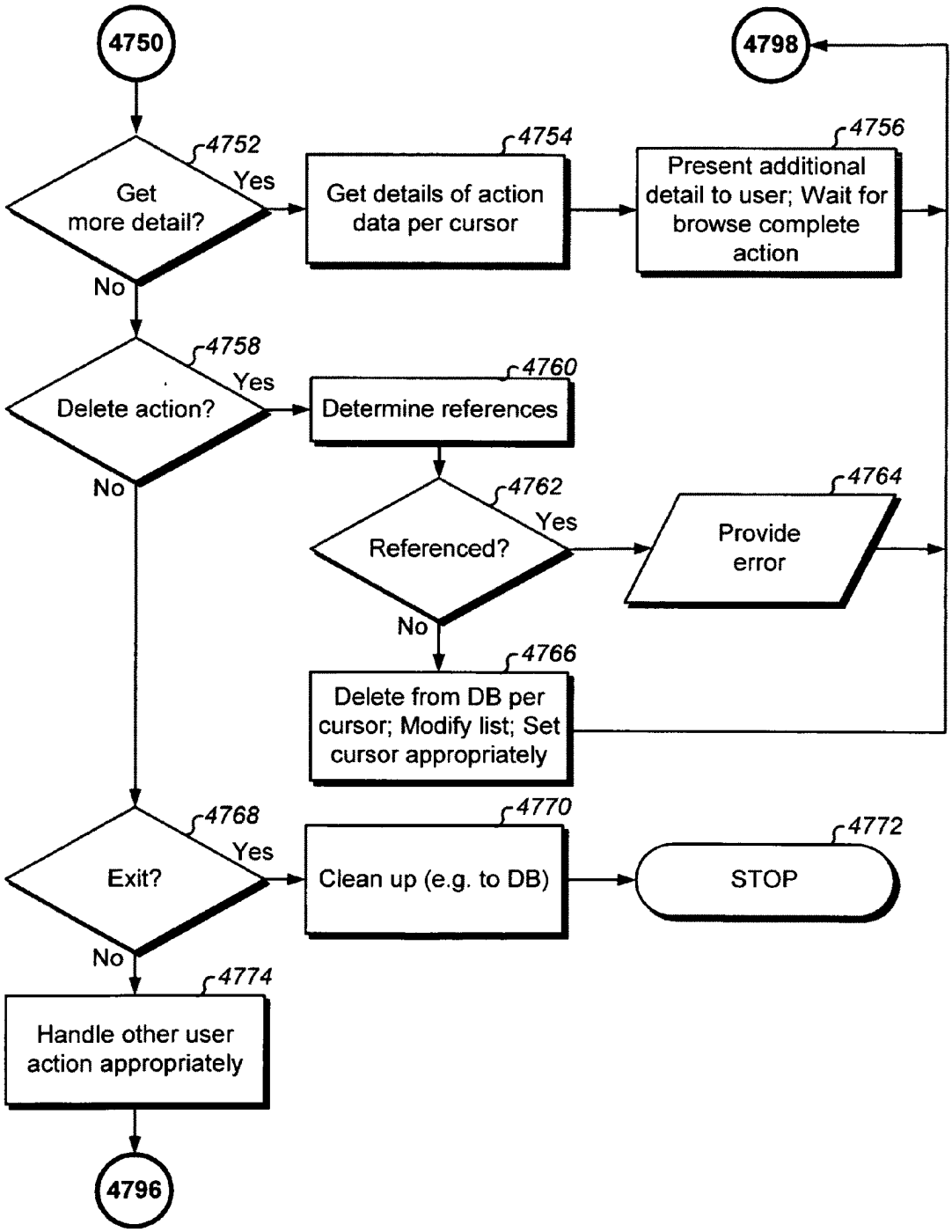

4750
4798
4752
Get more detail?
Yes
No
4754
Get details of action data per cursor
4756
Present additional detail to user; Wait for browse complete action
4758
Delete action?
Yes
No
4760
Determine references
4762
Referenced?
Yes
No
4764
Provide error
4766
Delete from DB per cursor; Modify list; Set cursor appropriately
4768
Exit?
Yes
No
4770
Clean up (e.g. to DB)
4772
STOP
4774
Handle other user action appropriately
4796

Fig. 47B

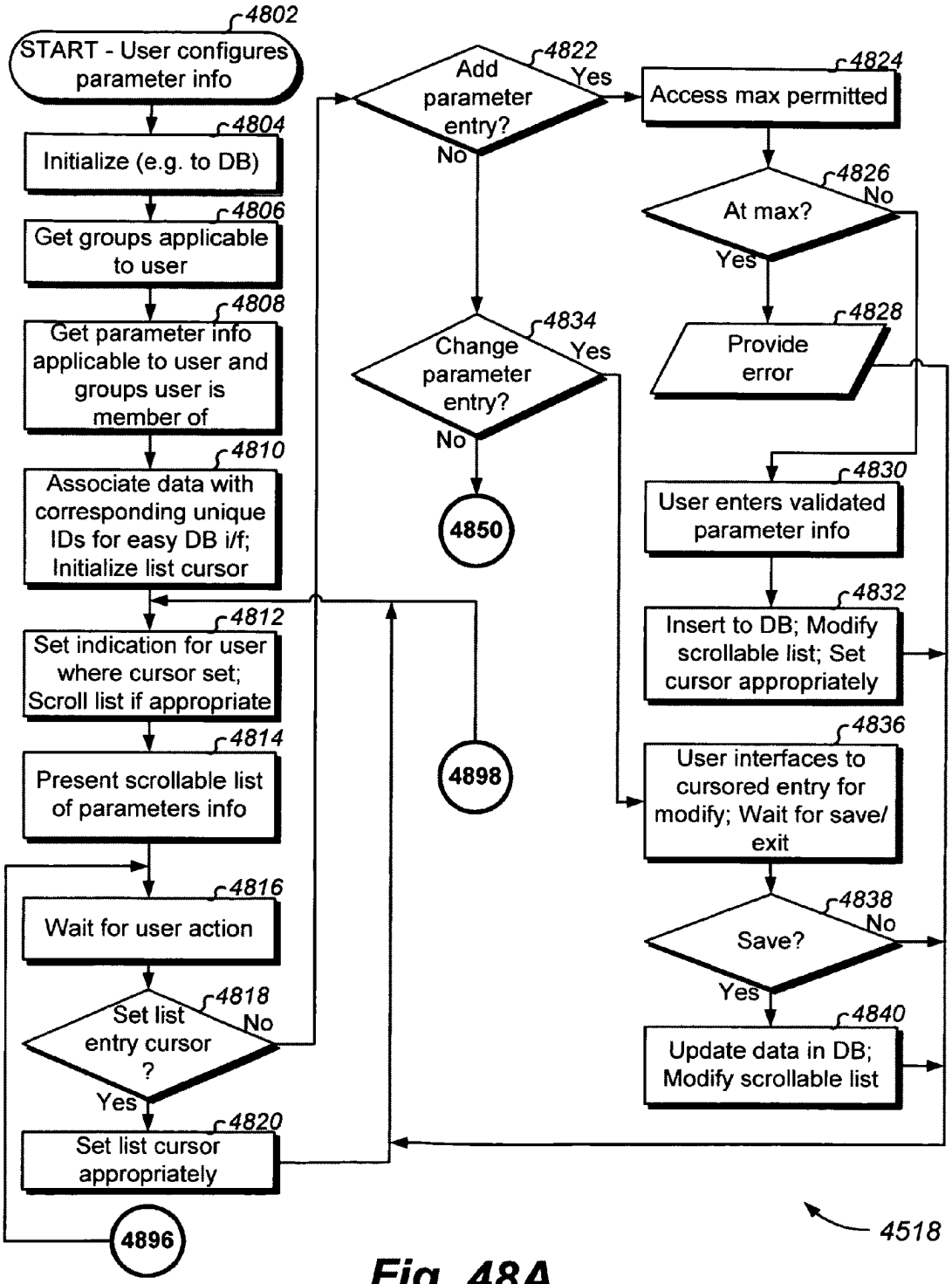

4802
START - User configures parameter info
4804
Initialize (e.g. to DB)
4806
Get groups applicable to user
4808
Get parameter info applicable to user and groups user is member of
4810
Associate data with corresponding unique IDs for easy DB i/f; Initialize list cursor
4812
Set indication for user where cursor set; Scroll list if appropriate
4814
Present scrollable list of parameters info
4816
Wait for user action
4818
Set list entry cursor ?
No
Yes
4820
Set list cursor appropriately
4896
4822
Add parameter entry?
Yes
No
4824
Access max permitted
4826
At max?
No
Yes
4828
Provide error
4834
Change parameter entry?
Yes
No
4850
4830
User enters validated parameter info
4832
Insert to DB; Modify scrollable list; Set cursor appropriately
4836
User interfaces to cursored entry for modify; Wait for save/ exit
4838
Save?
No
Yes
4840
Update data in DB; Modify scrollable list
4898
4518

Fig. 48A

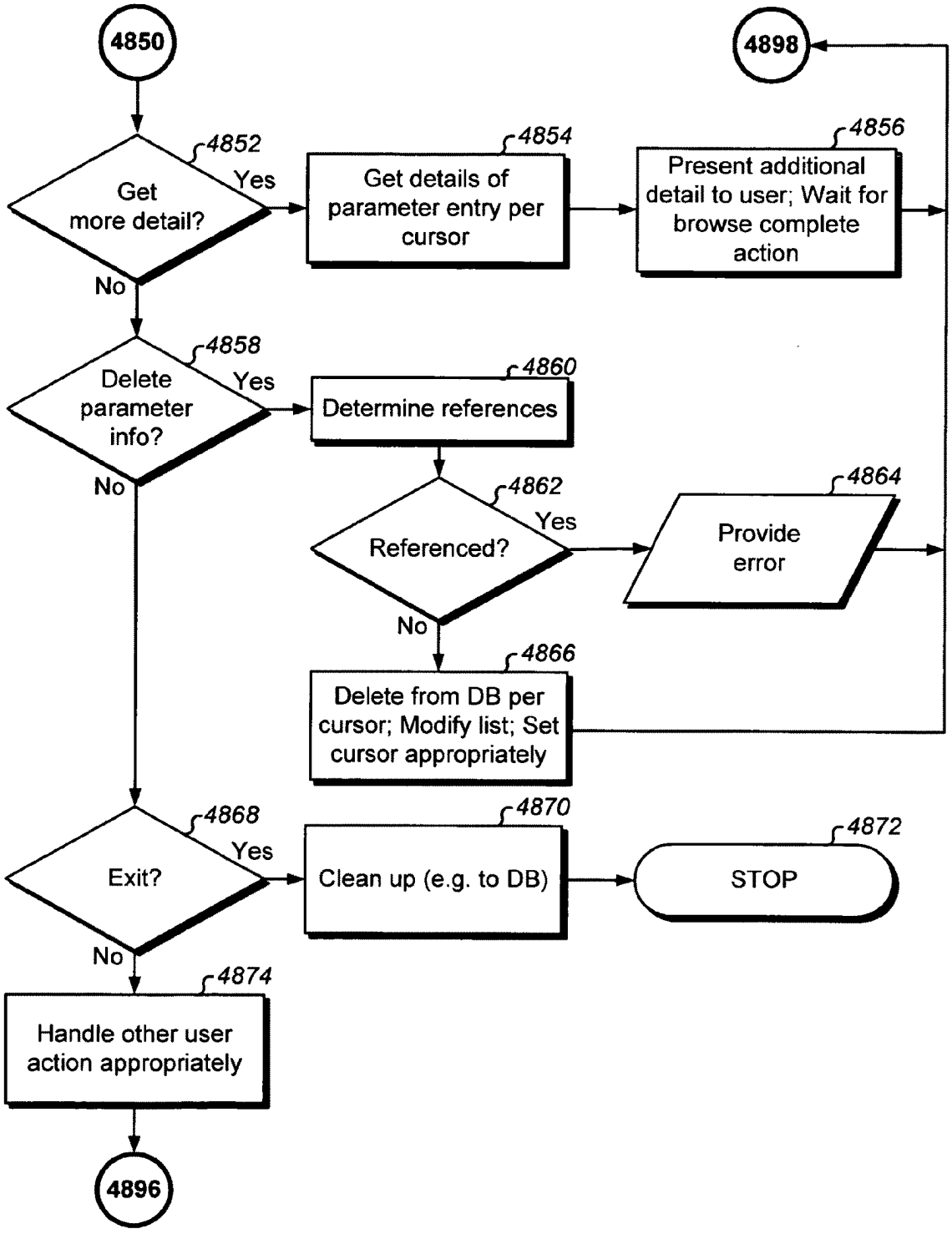

4850
4898
4852
Get more detail?
Yes
No
4854
Get details of parameter entry per cursor
4856
Present additional detail to user; Wait for browse complete action
4858
Delete parameter info?
Yes
No
4860
Determine references
4862
Referenced?
Yes
No
4864
Provide error
4866
Delete from DB per cursor; Modify list; Set cursor appropriately
4868
Exit?
Yes
No
4870
Clean up (e.g. to DB)
4872
STOP
4874
Handle other user action appropriately
4896

Fig. 48B

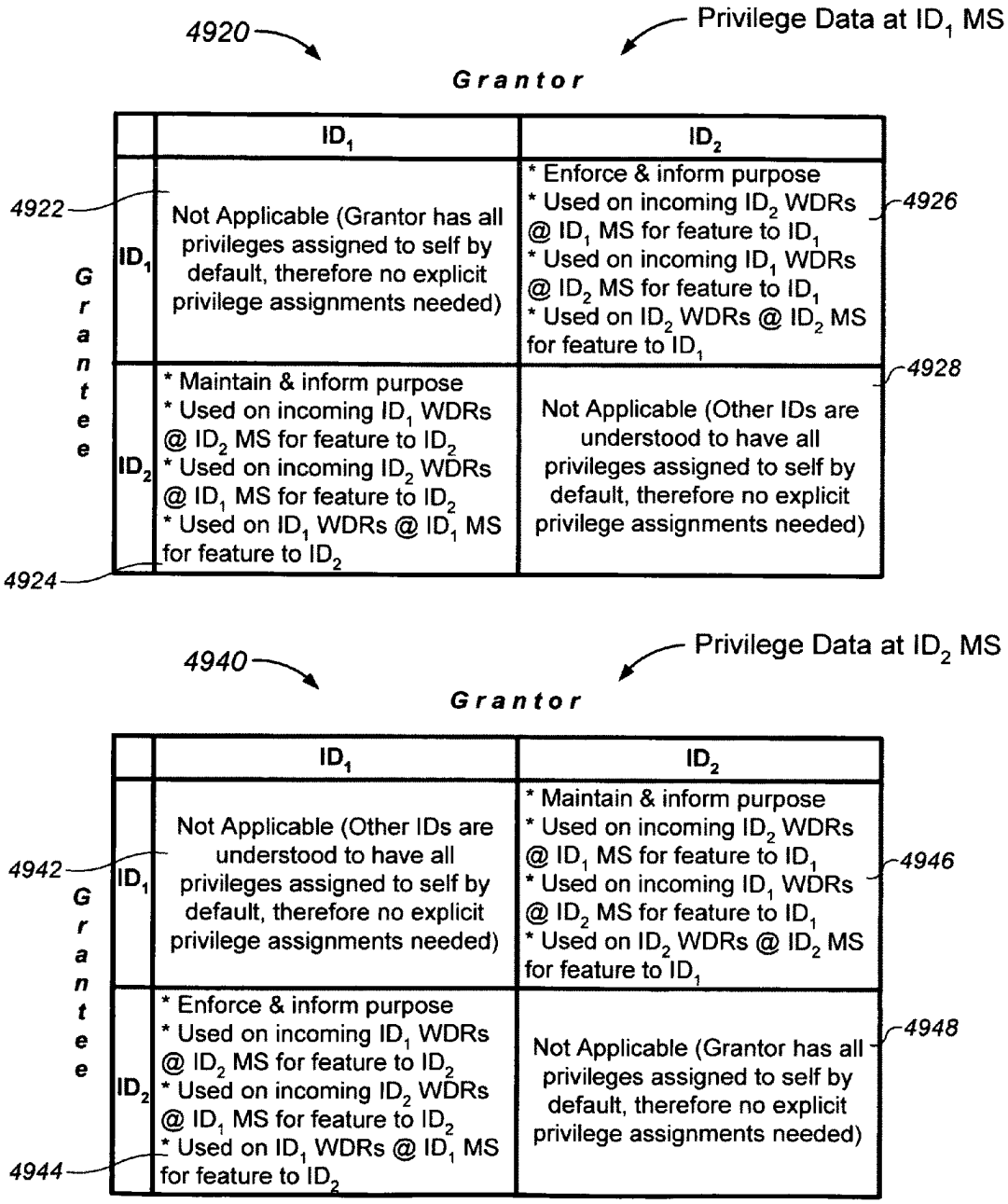

4920
Privilege Data at ID1 MS
Grantor
ID1
ID2
4922
4926
4928
4924
Grantee
ID1
ID2
Not Applicable (Grantor has all privileges assigned to self by default, therefore no explicit privilege assignments needed)
* Enforce & inform purpose
* Used on incoming ID2 WDRs @ ID1 MS for feature to ID1
* Used on incoming ID1 WDRs @ ID2 MS for feature to ID1
* Used on ID2 WDRs @ ID2 MS for feature to ID1
* Maintain & inform purpose
* Used on incoming ID1 WDRs @ ID2 MS for feature to ID2
* Used on incoming ID2 WDRs @ ID1 MS for feature to ID2
* Used on ID1 WDRs @ ID1 MS for feature to ID2
Not Applicable (Other IDs are understood to have all privileges assigned to self by default, therefore no explicit privilege assignments needed)
4940
Privilege Data at ID2 MS
Grantor
ID1
ID2
4942
4946
4948
4944
Grantee
ID1
ID2
Not Applicable (Other IDs are understood to have all privileges assigned to self by default, therefore no explicit privilege assignments needed)
* Maintain & inform purpose
* Used on incoming ID2 WDRs @ ID1 MS for feature to ID1
* Used on incoming ID1 WDRs @ ID2 MS for feature to ID1
* Used on ID2 WDRs @ ID2 MS for feature to ID1
* Enforce & inform purpose
* Used on incoming ID1 WDRs @ ID2 MS for feature to ID2
* Used on incoming ID2 WDRs @ ID1 MS for feature to ID2
* Used on ID1 WDRs @ ID1 MS for feature to ID2
Not Applicable (Grantor has all privileges assigned to self by default, therefore no explicit privilege assignments needed)

Fig. 49A

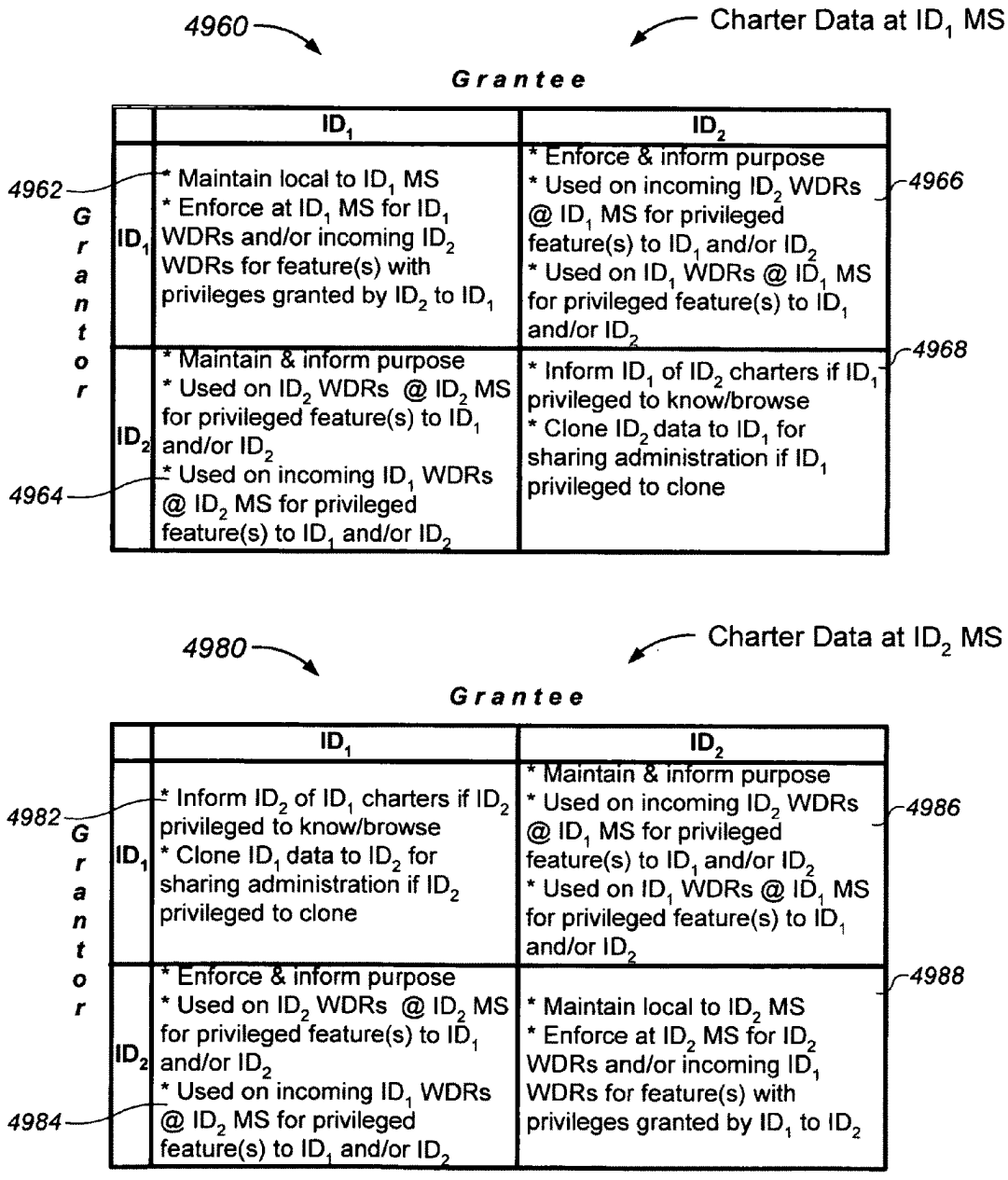

4960
Charter Data at ID1 MS
Grantee
Grantor
ID1
ID2
4962
4964
4966
4968
* Maintain local to ID1 MS
* Enforce at ID1 MS for ID1 WDRs and/or incoming ID2 WDRs for feature(s) with privileges granted by ID2 to ID1
* Enforce & inform purpose
* Used on incoming ID2 WDRs @ ID1 MS for privileged feature(s) to ID1 and/or ID2
* Used on ID1 WDRs @ ID1 MS for privileged feature(s) to ID1 and/or ID2
* Maintain & inform purpose
* Used on ID2 WDRs @ ID2 MS for privileged feature(s) to ID1 and/or ID2
* Used on incoming ID1 WDRs @ ID2 MS for privileged feature(s) to ID1 and/or ID2
* Inform ID1 of ID2 charters if ID1 privileged to know/browse
* Clone ID2 data to ID1 for sharing administration if ID1 privileged to clone
4980
Charter Data at ID2 MS
Grantee
Grantor
ID1
ID2
4982
4984
4986
4988
* Inform ID2 of ID1 charters if ID2 privileged to know/browse
* Clone ID1 data to ID2 for sharing administration if ID2 privileged to clone
* Maintain & inform purpose
* Used on incoming ID2 WDRs @ ID1 MS for privileged feature(s) to ID1 and/or ID2
* Used on ID1 WDRs @ ID1 MS for privileged feature(s) to ID1 and/or ID2
* Enforce & inform purpose
* Used on ID2 WDRs @ ID2 MS for privileged feature(s) to ID1 and/or ID2
* Used on incoming ID1 WDRs @ ID2 MS for privileged feature(s) to ID1 and/or ID2
* Maintain local to ID2 MS
* Enforce at ID2 MS for ID2 WDRs and/or incoming ID1 WDRs for feature(s) with privileges granted by ID1 to ID2

Fig. 49B

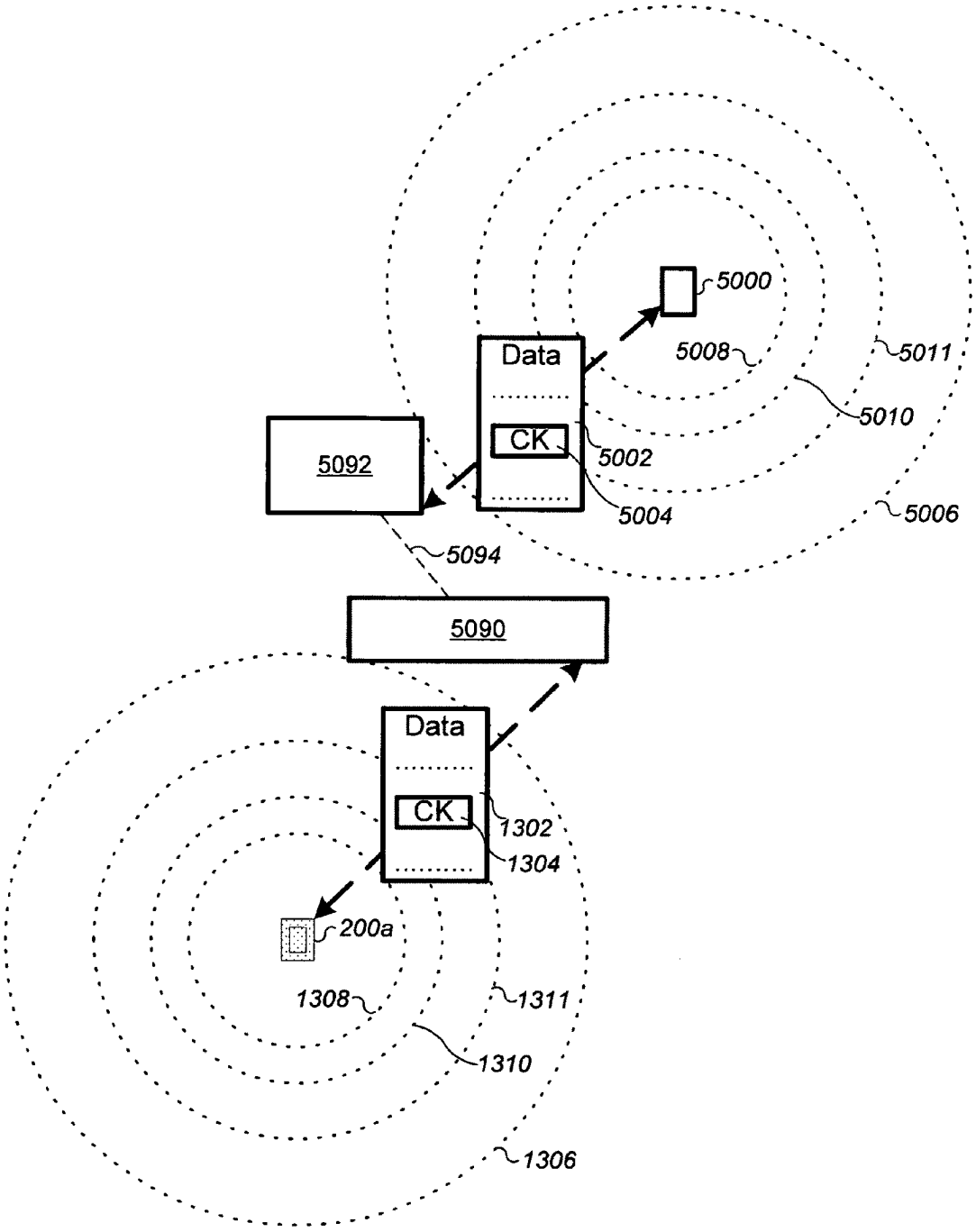

5000
5008
5011
Data
5010
CK
5002
5092
5004
5006
5094
5090
Data
CK
1302
1304
200a
1308
1311
1310
1306

Fig. 50A

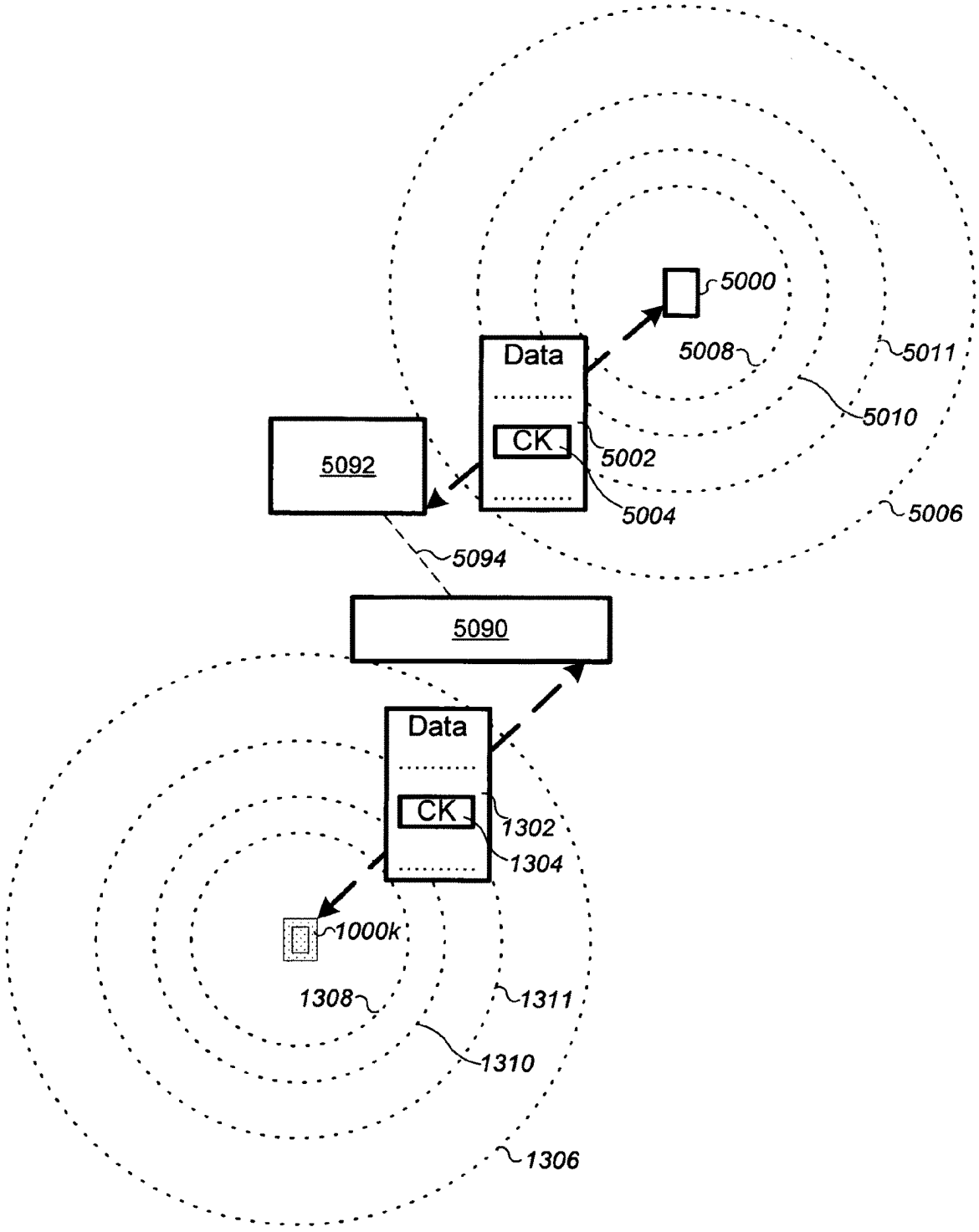

5000
5008
5011
Data
5010
CK
5002
5092
5004
5006
5094
5090
Data
CK
1302
1304
1000k
1308
1311
1310
1306

Fig. 50B

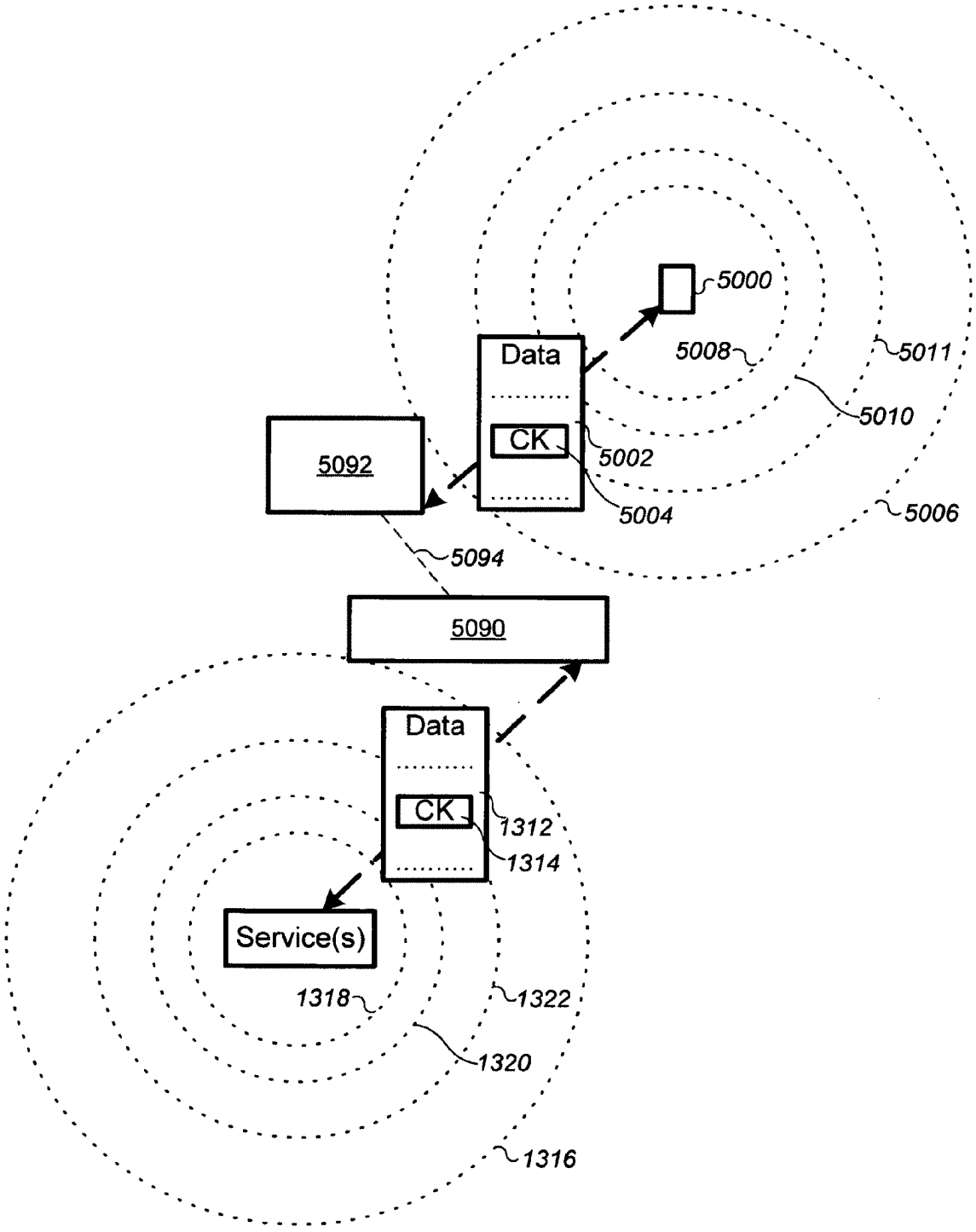

5000
5008
5011
Data
5010
CK
5002
5092
5004
5006
5094
5090
Data
CK
1312
1314
Service(s)
1318
1322
1320
1316

Fig. 50C

```
Permissions {

        Text(str) = "Test Case #106729 (context)";
        Generic(assignPrivs) = "G=Family,Work,\vuloc [T=>20080402000130.24,<20080428;
                D=*str; H;]";
        Groups {
                LBXPHONE_USERS = Austin, Davood, Jane, Kris, Mark, Ravi, Sam, Tim;
                "SW Components" = "SM 1.0", "PIP 1.0", "PIPGUI 1.0", "SMGUI 1.0",
                                "COMM 1.0", "KERNEL 1.1";
        }

        Grants /* Can define Grant structure(s) prior to assignment */ {
                Family= \lbxall[R=0xFFFFFFFF;] [D=*str(context="Family")];
                Work = [T=YYYYMMDD08:YYYYMMDD17;D=*str(context="Work");H;] {
                        "Department 232"=\geoar,\geode,\nearar,\nearde;
                        "Department 458" = [D="Davood Iyadi's mgt scope";] {
                                "Server Development Team" = ;
                                "lbxPhone Development Team" = {
                                        "Comm Layer Guys" = \mssys;\msbios;
                                        "GUI girls" = \msguiload;
                                        "Mark and Tim" = \msapps;
                                };
                        };
                        "Accounting Department" [H;] = \track;
                };
                Parents = { Mom=\lbxall; Dad=\lbxall; };
                Michael-Friends=\geoarr;\geode;
                Jason-Friends=\nearar;\nearde;
        }

        // Permissions are granted here:
        Bill: LBXPHONE_USERS [G=\caller;\callee;\trkall;];
        LBXPHONE_USERS: Bill [G=\callee;\caller;];
        Bill:Sophia;
        Bill:Brian [*assignPrivs];
        Bill:George [G=\geoall,\nearall;];
        Michael : Bill [G=Parents,Michael-Friends;];
        Jason: Bill [G=Parents,Jason-Friends;];

}
```

Fig. 51A

```
Charters {

        Condition(cond1) = "(__location @@ \loc_my) [D="Test Case #104223 (v)";]";
        "ms group"  =  { "Jane", "George", "Sally" };

        (  ((__msid = "Michael") & *cond1(v="Michael")) |
                ((__msid = "Jason") & *cond1(v="'Jason"))  ):
               Invoke App myscript.cmd ("S"), Notify Autodial 214-405-6733;

        ((_I_msid = "Brian") & (_I_location @ \loc_my) [D="multi-cond text";H;]):
                        Invoke App (myscript.cmd ("B")) [T=20080302;],
                        Notify Autodial (214-405-5422);

        (M_sender  = ~emailAddrVar [T=<YYYYMMDD18]):
                Notify Indicator (M_sender, \thisMS) [D="Test Case #104223"; H;];

        (B_srchSubj ^ M_subject) & !(_fcnTest(B_srchSubj)) :
                "ms group"[G].Store DBobject(JOESDB.LBXTABS.TEST,
                        "INSERT INTO TABLESAV (" && \thisMS && ", " && \timestamp &&
                                ", 9);", \thisMS);

        ( _I_msid = "Sophia" & \loc_my (30M)$$(25M) _I_location ) :
                "ms group".Invoke App (alert.cmd);

        (%c:\myprofs\interests.chk > 90):
                Send Email ("Howdy " && _I_msid && " !!\n\nOur profiles matched > 90%.\n\n"
                        && "Call me at " && \appfld.phone.id && ". We are " &&
                        (_I_location - \loc_my)F && " feet apart\n", \appfld.source.id, "Call Me!",
                        , _I_appfld.email.source);

}
```

Fig. 51B

```
typedef struct privilege {
        unsigned long       priv;
        unsigned char       relevance[MAX_RELEVANCEMASK];
        TIMESPEC            *tspec;      // merged with permission level (if permission
                                         // level was present)
        struct privilege    *nextPriv;
        } PRIVILEGE;

typedef struct permission {
        unsigned char       grantor[MAX_IDLENGTH];
        unsigned short      grantor_idtype;
        unsigned char       grantee[MAX_IDLENGTH];
        unsigned short      grantee_idtype;
        PRIVILEGE           *privileges;
        struct permission   *nextPerm;
        } PERMISSION;

typedef struct action {
        IDENTITY            host;
        unsigned short      cmd;
        unsigned short      operand;
        unsigned char       *params;
        TIMESPEC            *tspec;          // merged with charter level (if charter
                                             // level was present)
        struct action       *nextActn;
        } ACTION;

typedef struct charter {
        unsigned char       grantee[MAX_IDLENGTH];
        unsigned short      grantee_idtype;
        unsigned char       grantor[MAX_IDLENGTH];
        unsigned short      grantor_idtype;
        unsigned char       *expression;
        ACTION              *actn;
        struct charter      *nextCharter;
        } CHARTER;
```

Fig. 52

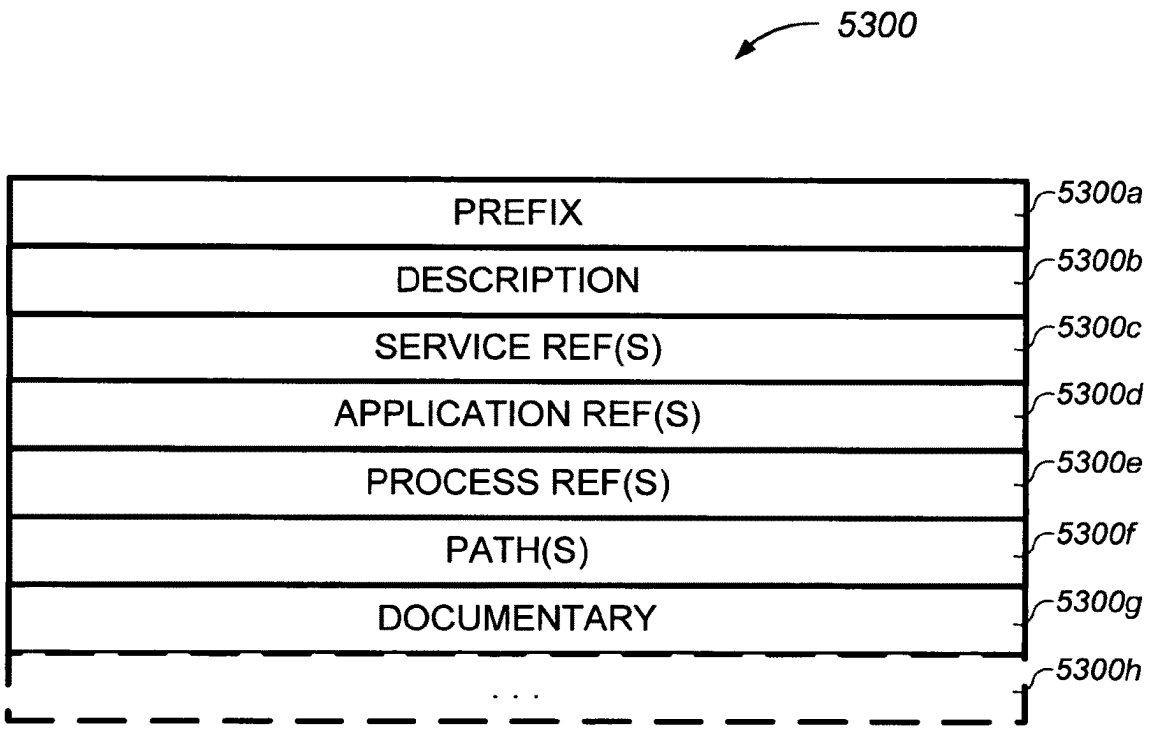

5300
PREFIX
DESCRIPTION
SERVICE REF(S)
APPLICATION REF(S)
PROCESS REF(S)
PATH(S)
DOCUMENTARY
. . .
5300a
5300b
5300c
5300d
5300e
5300f
5300g
5300h

Fig. 53

```
...
x = "this is a textual description"
...
z="timespec=""200802030000:200812312359"" description=""test98341;Permission"""
...
<permission grantor="Jimbob" grantee="Henry" <%=z%> >
        <grant name="grant1" >
                <privilege id="\lbxcpy" relevance="FFFFFFFF"
                        timespec="YYYMMDD09:YYMMDD17" description="<%=x%>" />
                <privilege id="\lbxflt" />
                ...
        </grant>
        ...
</permission>
...
<group name="group123" >
        <member="Jim" />
        <member="Sue" />
        ...
</group>
...
<charter grantee="Henry" grantor="Jimbob" timespec="200802030000:200812312359"
        description="test98341;Charter" >
        <expression>
                <condition trigger="true"
                        specification="(__msid = ""Michael"") & __location $(300M) \loc_my" />
                <condition trigger="true"
                        specification="(__msid = "Jason") & __location $(300M) \loc_my" />
        </expression>
        <action host="George" cmd="Invoke" operand="App" param="alert.cmd" />
        <action host="George" cmd="Notify" operand="Indicator" param="test98341 Fired!" />
</charter>
...
```

Fig. 54

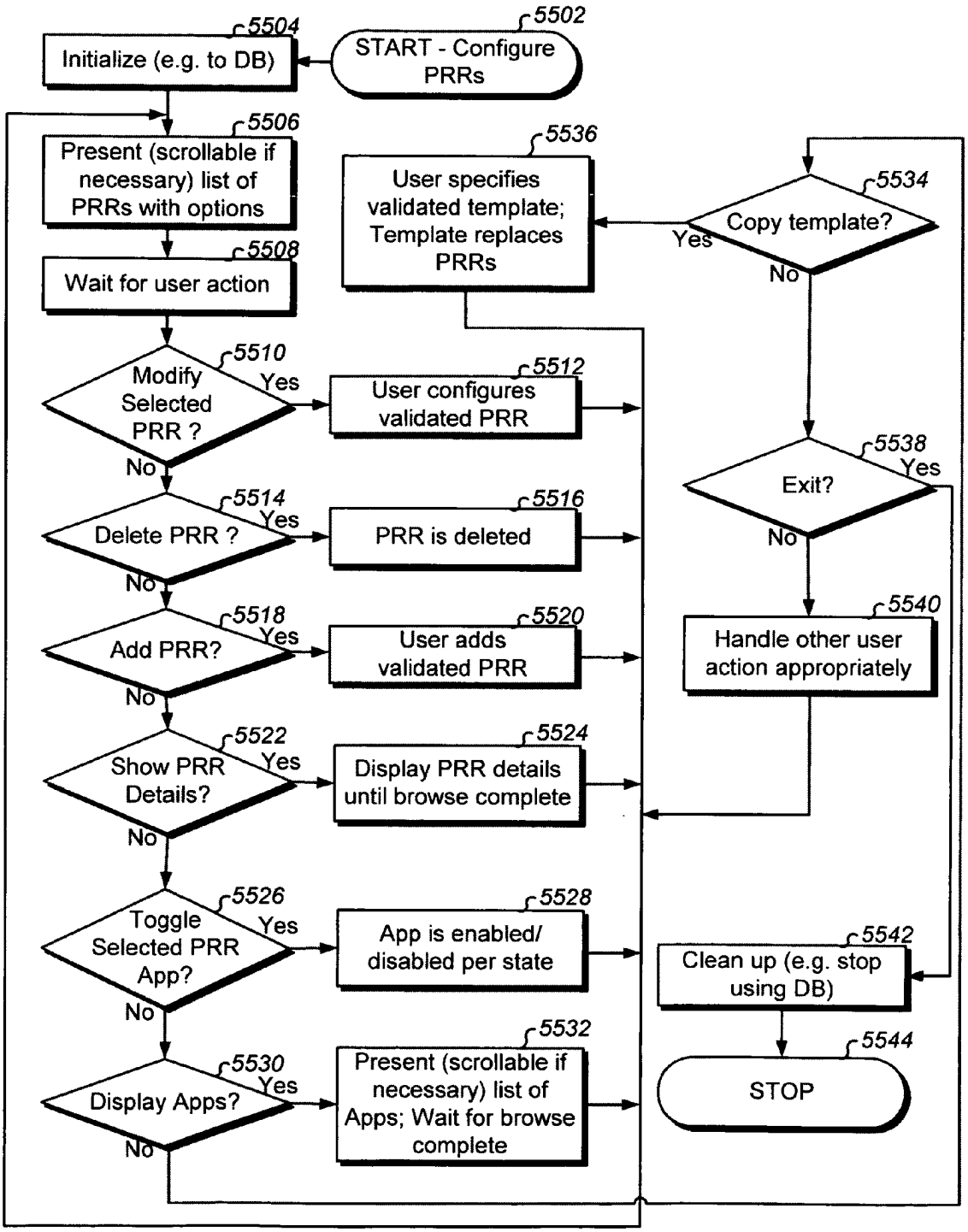

5502
START - Configure PRRs
5504
Initialize (e.g. to DB)
5506
Present (scrollable if necessary) list of PRRs with options
5508
Wait for user action
5510
Modify Selected PRR ?
Yes
No
5512
User configures validated PRR
5514
Delete PRR ?
Yes
No
5516
PRR is deleted
5518
Add PRR?
Yes
No
5520
User adds validated PRR
5522
Show PRR Details?
Yes
No
5524
Display PRR details until browse complete
5526
Toggle Selected PRR App?
Yes
No
5528
App is enabled/ disabled per state
5530
Display Apps?
Yes
No
5532
Present (scrollable if necessary) list of Apps; Wait for browse complete
5534
Copy template?
Yes
No
5536
User specifies validated template; Template replaces PRRs
5538
Exit?
Yes
No
5540
Handle other user action appropriately
5542
Clean up (e.g. stop using DB)
5544
STOP

Fig. 55A

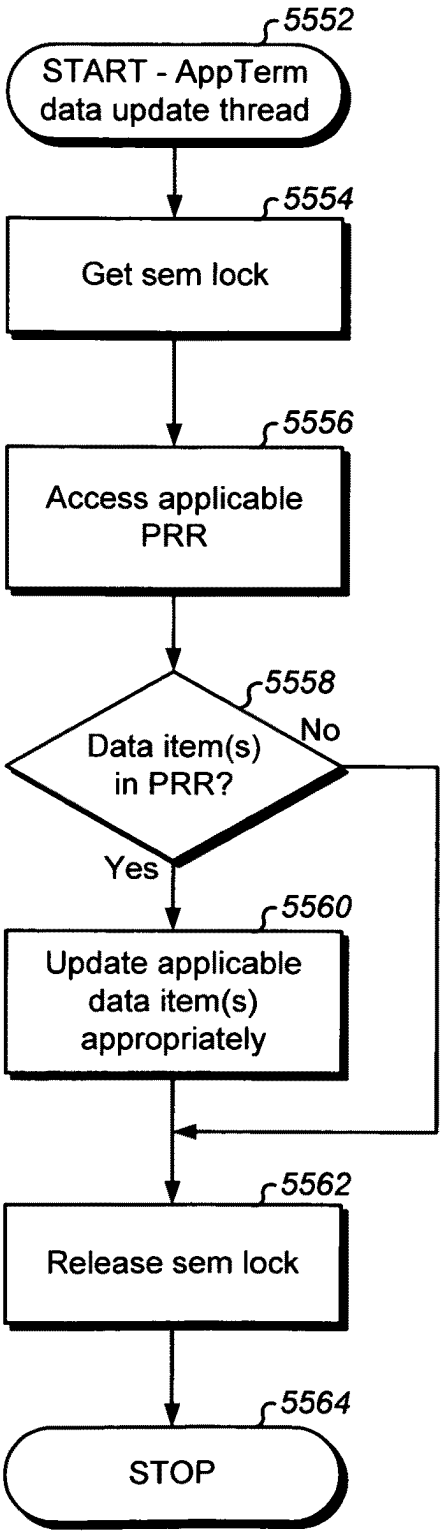

5552
START - AppTerm data update thread
5554
Get sem lock
5556
Access applicable PRR
5558
Data item(s) in PRR?
No
Yes
5560
Update applicable data item(s) appropriately
5562
Release sem lock
5564
STOP

Fig. 55B

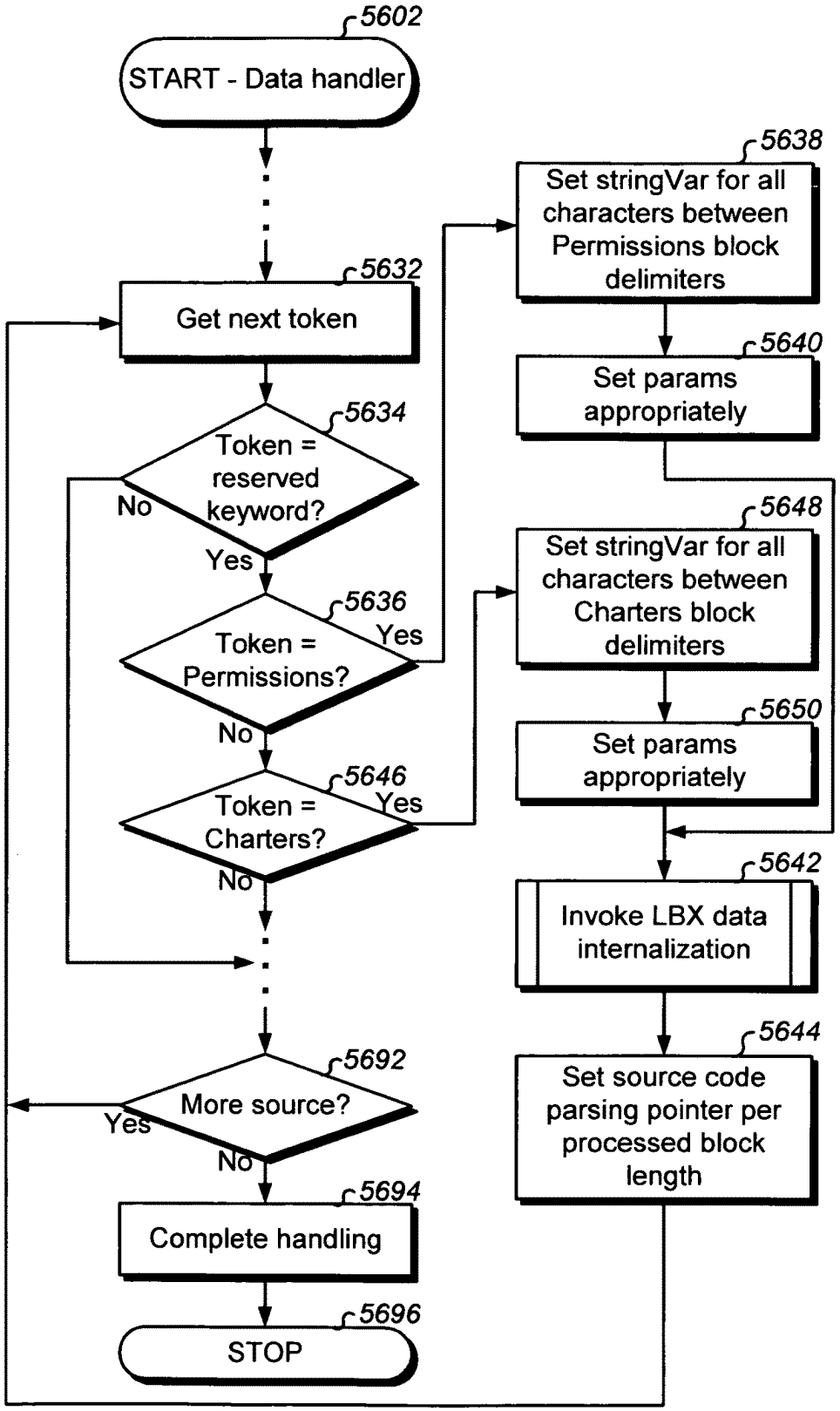

5602
START - Data handler
5632
Get next token
5634
Token = reserved keyword?
No
Yes
5636
Token = Permissions?
Yes
No
5646
Token = Charters?
Yes
No
5692
More source?
Yes
No
5694
Complete handling
5696
STOP
5638
Set stringVar for all characters between Permissions block delimiters
5640
Set params appropriately
5648
Set stringVar for all characters between Charters block delimiters
5650
Set params appropriately
5642
Invoke LBX data internalization
5644
Set source code parsing pointer per processed block length

Fig. 56

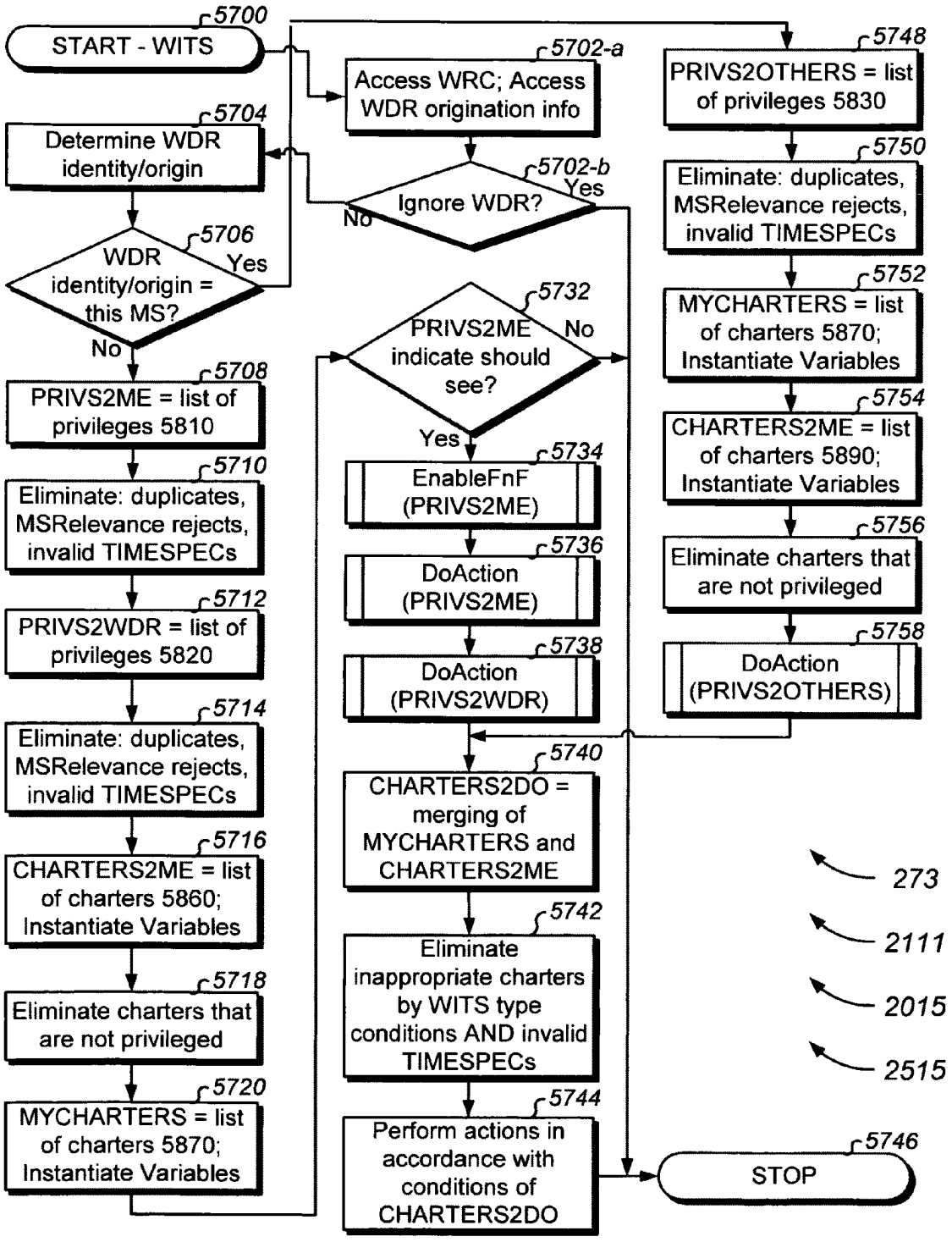

5700
START - WITS
5702-a
Access WRC; Access WDR origination info
5748
PRIVS2OTHERS = list of privileges 5830
5704
Determine WDR identity/origin
5702-b
Ignore WDR?
Yes
No
5750
Eliminate: duplicates, MSRelevance rejects, invalid TIMESPECs
5706
WDR identity/origin = this MS?
Yes
No
5732
PRIVS2ME indicate should see?
No
Yes
5752
MYCHARTERS = list of charters 5870; Instantiate Variables
5754
CHARTERS2ME = list of charters 5890; Instantiate Variables
5708
PRIVS2ME = list of privileges 5810
5734
EnableFnF (PRIVS2ME)
5710
Eliminate: duplicates, MSRelevance rejects, invalid TIMESPECs
5736
DoAction (PRIVS2ME)
5756
Eliminate charters that are not privileged
5712
PRIVS2WDR = list of privileges 5820
5738
DoAction (PRIVS2WDR)
5758
DoAction (PRIVS2OTHERS)
5714
Eliminate: duplicates, MSRelevance rejects, invalid TIMESPECs
5740
CHARTERS2DO = merging of MYCHARTERS and CHARTERS2ME
5716
CHARTERS2ME = list of charters 5860; Instantiate Variables
5742
Eliminate inappropriate charters by WITS type conditions AND invalid TIMESPECs
5718
Eliminate charters that are not privileged
5744
Perform actions in accordance with conditions of CHARTERS2DO
5746
STOP
5720
MYCHARTERS = list of charters 5870; Instantiate Variables
273
2111
2015
2515

Fig. 57

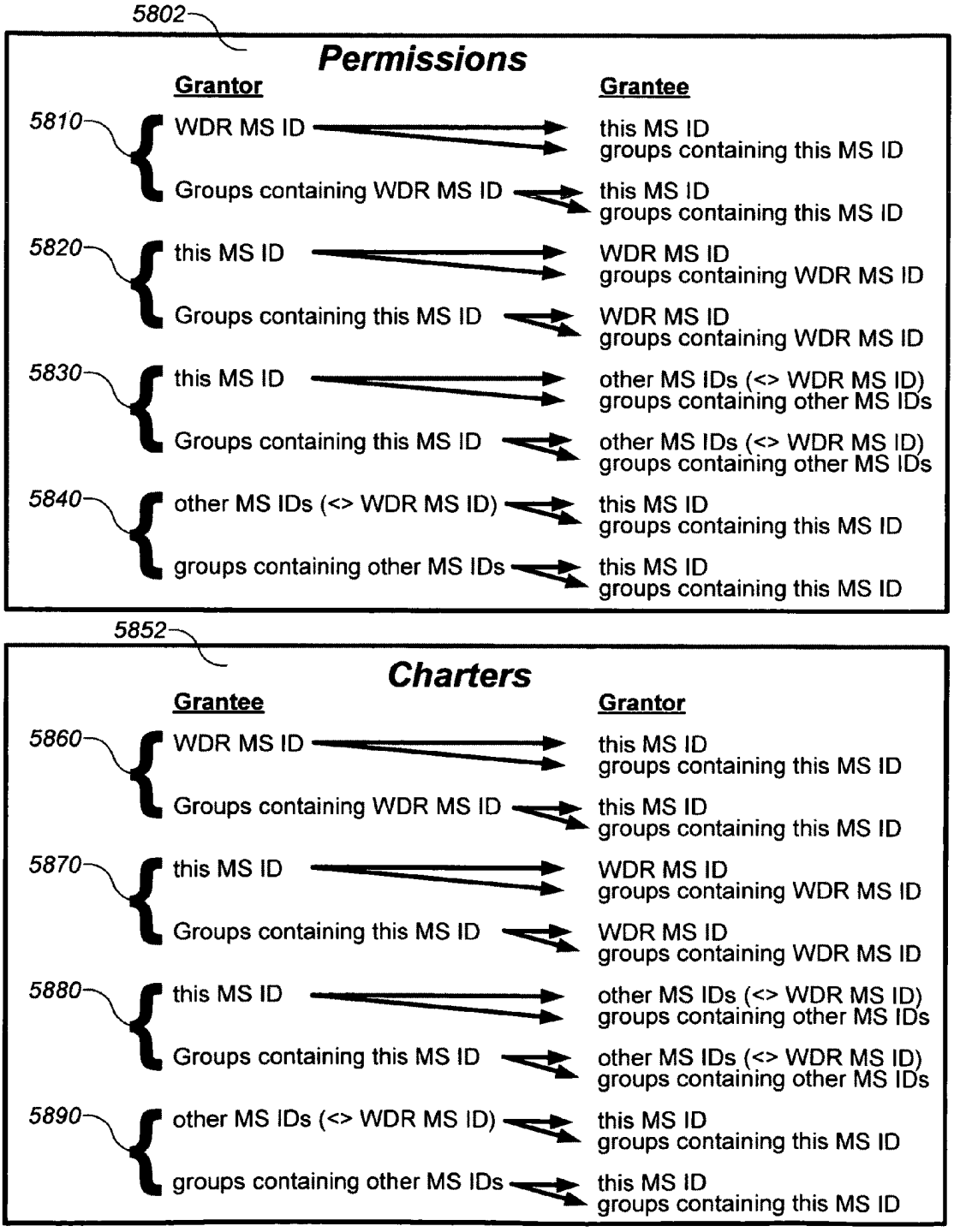

5802
Permissions
Grantor
Grantee
5810
WDR MS ID
this MS ID
groups containing this MS ID
Groups containing WDR MS ID
this MS ID
groups containing this MS ID
5820
this MS ID
WDR MS ID
groups containing WDR MS ID
Groups containing this MS ID
WDR MS ID
groups containing WDR MS ID
5830
this MS ID
other MS IDs (<> WDR MS ID)
groups containing other MS IDs
Groups containing this MS ID
other MS IDs (<> WDR MS ID)
groups containing other MS IDs
5840
other MS IDs (<> WDR MS ID)
this MS ID
groups containing this MS ID
groups containing other MS IDs
this MS ID
groups containing this MS ID
5852
Charters
Grantee
Grantor
5860
WDR MS ID
this MS ID
groups containing this MS ID
Groups containing WDR MS ID
this MS ID
groups containing this MS ID
5870
this MS ID
WDR MS ID
groups containing WDR MS ID
Groups containing this MS ID
WDR MS ID
groups containing WDR MS ID
5880
this MS ID
other MS IDs (<> WDR MS ID)
groups containing other MS IDs
Groups containing this MS ID
other MS IDs (<> WDR MS ID)
groups containing other MS IDs
5890
other MS IDs (<> WDR MS ID)
this MS ID
groups containing this MS ID
groups containing other MS IDs
this MS ID
groups containing this MS ID

Fig. 58

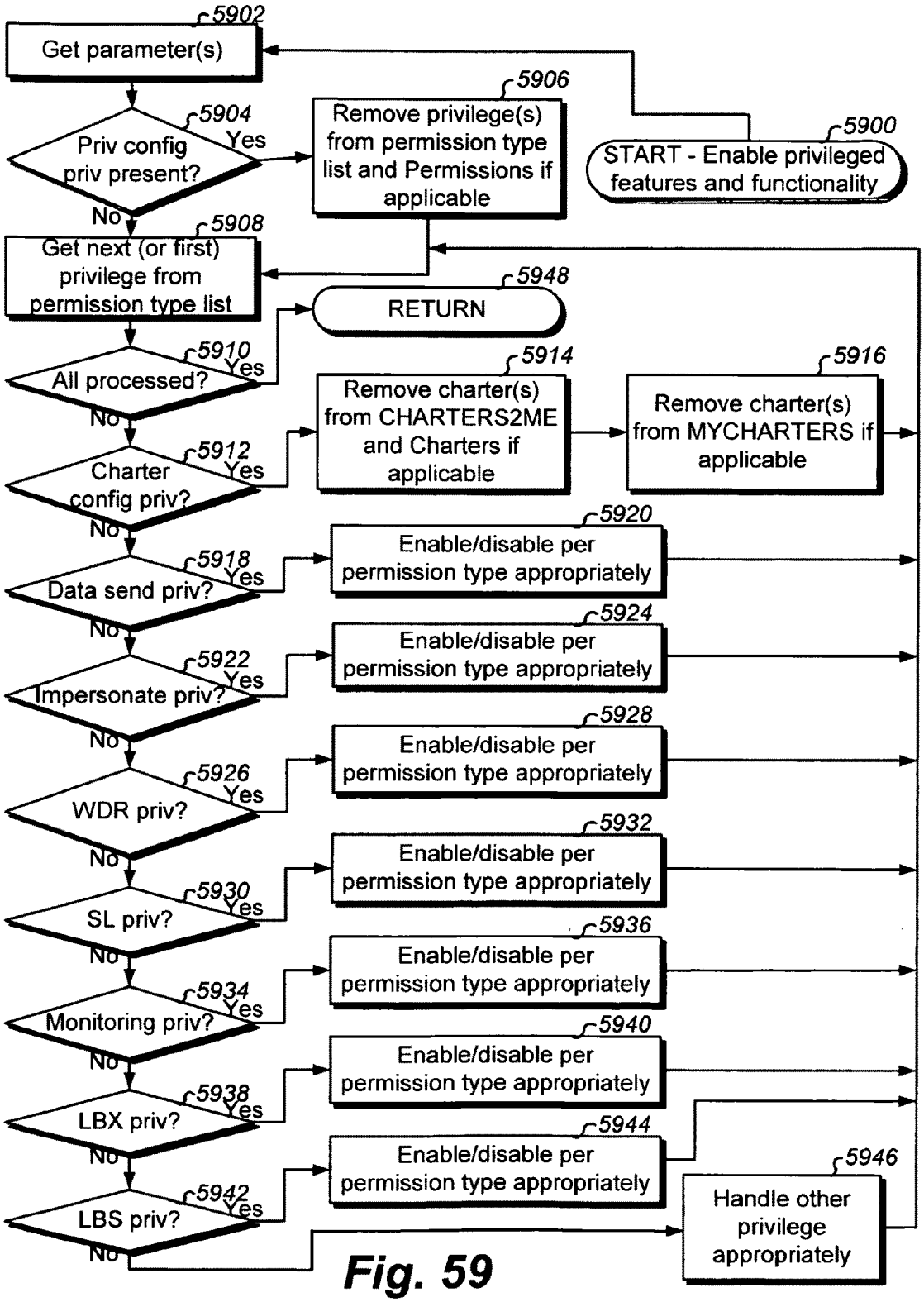

5902
Get parameter(s)
5904
Priv config priv present?
Yes
No
5906
Remove privilege(s) from permission type list and Permissions if applicable
5900
START - Enable privileged features and functionality
5908
Get next (or first) privilege from permission type list
5948
RETURN
5910
All processed?
Yes
No
5914
Remove charter(s) from CHARTERS2ME and Charters if applicable
5916
Remove charter(s) from MYCHARTERS if applicable
5912
Charter config priv?
Yes
No
5920
Enable/disable per permission type appropriately
5918
Data send priv?
Yes
No
5924
Enable/disable per permission type appropriately
5922
Impersonate priv?
Yes
No
5928
Enable/disable per permission type appropriately
5926
WDR priv?
Yes
No
5932
Enable/disable per permission type appropriately
5930
SL priv?
Yes
No
5936
Enable/disable per permission type appropriately
5934
Monitoring priv?
Yes
No
5940
Enable/disable per permission type appropriately
5938
LBX priv?
Yes
No
5944
Enable/disable per permission type appropriately
5946
Handle other privilege appropriately
5942
LBS priv?
Yes
No

Fig. 59

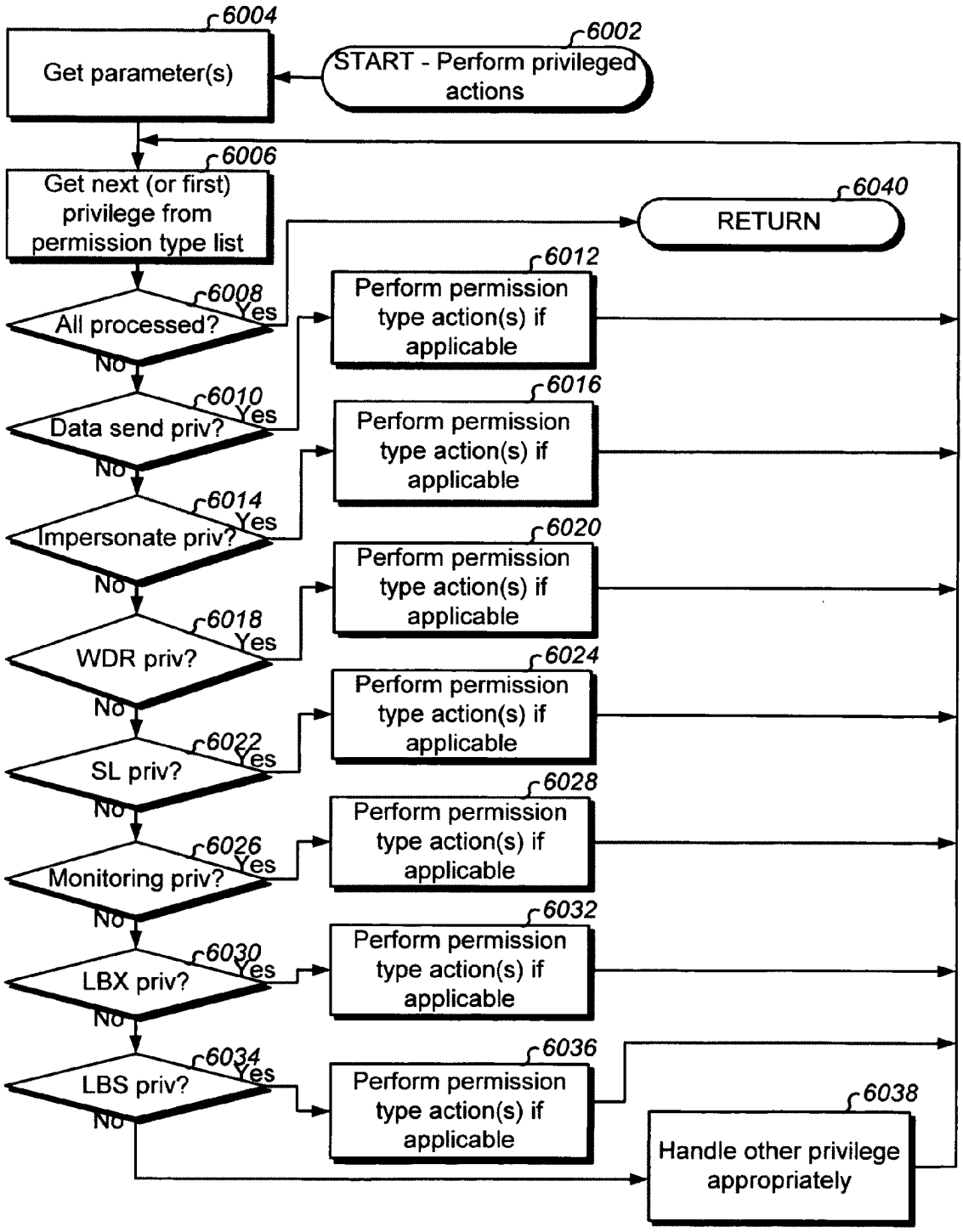

6004
Get parameter(s)
6002
START - Perform privileged actions
6006
Get next (or first) privilege from permission type list
6040
RETURN
6008
All processed?
Yes
No
6012
Perform permission type action(s) if applicable
6010
Data send priv?
Yes
No
6016
Perform permission type action(s) if applicable
6014
Impersonate priv?
Yes
No
6020
Perform permission type action(s) if applicable
6018
WDR priv?
Yes
No
6024
Perform permission type action(s) if applicable
6022
SL priv?
Yes
No
6028
Perform permission type action(s) if applicable
6026
Monitoring priv?
Yes
No
6032
Perform permission type action(s) if applicable
6030
LBX priv?
Yes
No
6036
Perform permission type action(s) if applicable
6034
LBS priv?
Yes
No
6038
Handle other privilege appropriately

Fig. 60

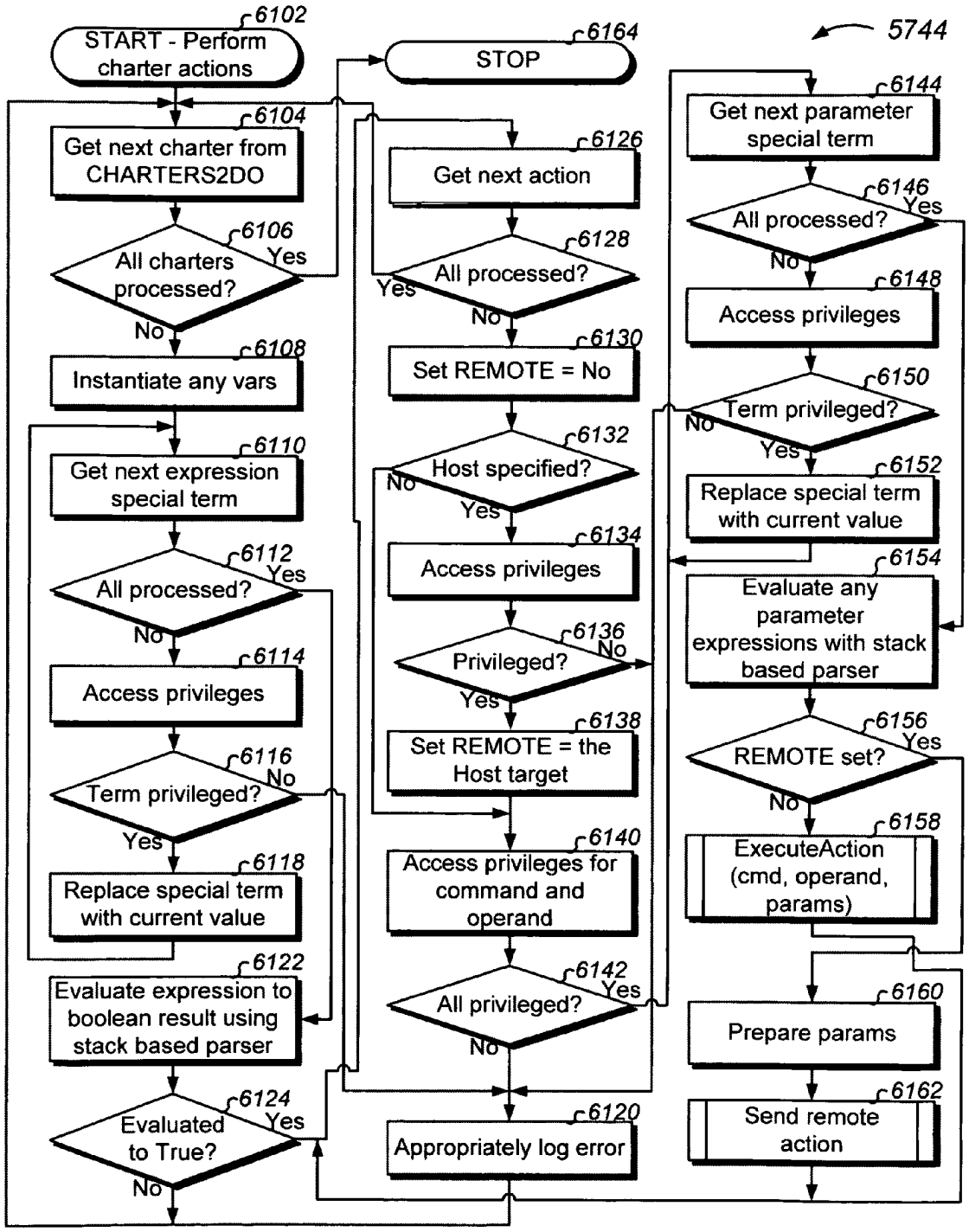

6102
START - Perform charter actions
6104
Get next charter from CHARTERS2DO
6106
All charters processed?
Yes
No
6108
Instantiate any vars
6110
Get next expression special term
6112
All processed?
Yes
No
6114
Access privileges
6116
Term privileged?
No
Yes
6118
Replace special term with current value
6122
Evaluate expression to boolean result using stack based parser
6124
Evaluated to True?
Yes
No
6164
STOP
6126
Get next action
6128
All processed?
Yes
No
6130
Set REMOTE = No
6132
Host specified?
No
Yes
6134
Access privileges
6136
Privileged?
No
Yes
6138
Set REMOTE = the Host target
6140
Access privileges for command and operand
6142
All privileged?
Yes
No
6120
Appropriately log error
5744
6144
Get next parameter special term
6146
All processed?
Yes
No
6148
Access privileges
6150
Term privileged?
No
Yes
6152
Replace special term with current value
6154
Evaluate any parameter expressions with stack based parser
6156
REMOTE set?
Yes
No
6158
ExecuteAction (cmd, operand, params)
6160
Prepare params
6162
Send remote action

Fig. 61

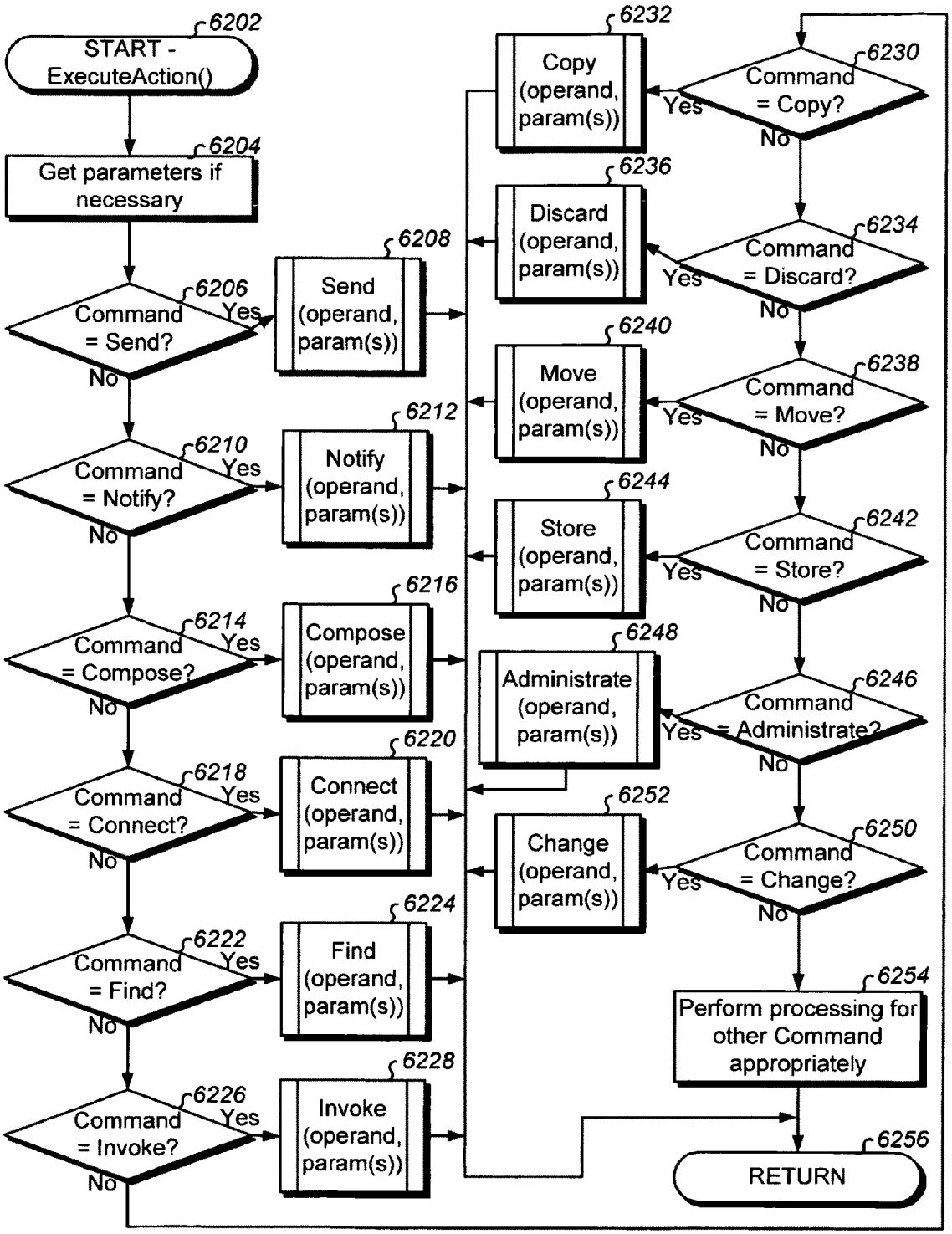

6202
START - ExecuteAction()
6204
Get parameters if necessary
6206
Command = Send?
Yes
No
6208
Send (operand, param(s))
6210
Command = Notify?
Yes
No
6212
Notify (operand, param(s))
6214
Command = Compose?
Yes
No
6216
Compose (operand, param(s))
6218
Command = Connect?
Yes
No
6220
Connect (operand, param(s))
6222
Command = Find?
Yes
No
6224
Find (operand, param(s))
6226
Command = Invoke?
Yes
No
6228
Invoke (operand, param(s))
6230
Command = Copy?
Yes
No
6232
Copy (operand, param(s))
6234
Command = Discard?
Yes
No
6236
Discard (operand, param(s))
6238
Command = Move?
Yes
No
6240
Move (operand, param(s))
6242
Command = Store?
Yes
No
6244
Store (operand, param(s))
6246
Command = Administrate?
Yes
No
6248
Administrate (operand, param(s))
6250
Command = Change?
Yes
No
6252
Change (operand, param(s))
6254
Perform processing for other Command appropriately
6256
RETURN

Fig. 62

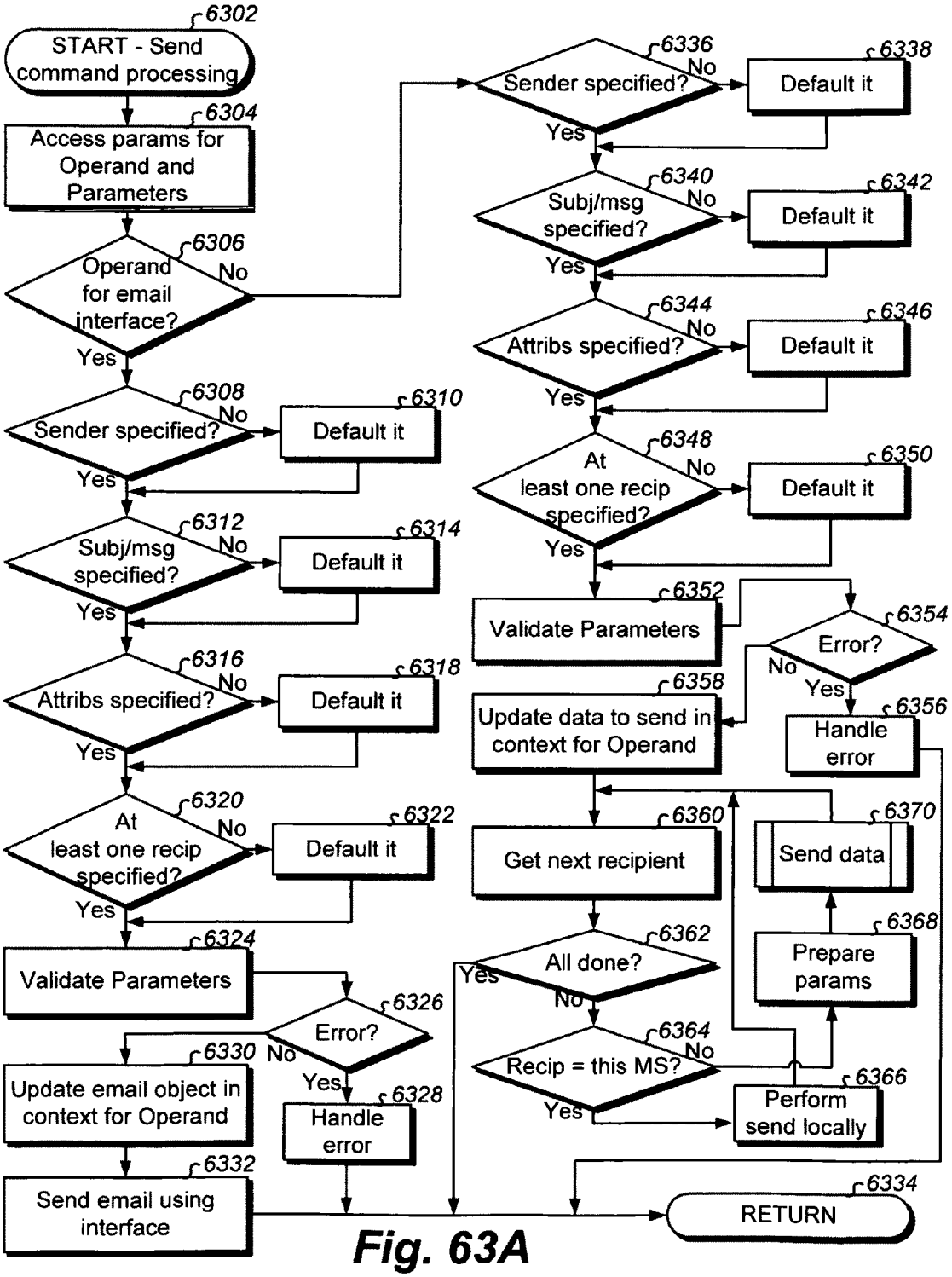

6302
START - Send command processing
6304
Access params for Operand and Parameters
6306
Operand for email interface?
No
Yes
6308
Sender specified?
No
6310
Default it
Yes
6312
Subj/msg specified?
No
6314
Default it
Yes
6316
Attribs specified?
No
6318
Default it
Yes
6320
At least one recip specified?
No
6322
Default it
Yes
6324
Validate Parameters
6326
Error?
No
Yes
6328
Handle error
6330
Update email object in context for Operand
6332
Send email using interface
6334
RETURN
6336
Sender specified?
No
6338
Default it
Yes
6340
Subj/msg specified?
No
6342
Default it
Yes
6344
Attribs specified?
No
6346
Default it
Yes
6348
At least one recip specified?
No
6350
Default it
Yes
6352
Validate Parameters
6354
Error?
No
Yes
6356
Handle error
6358
Update data to send in context for Operand
6360
Get next recipient
6362
All done?
Yes
No
6364
Recip = this MS?
No
Yes
6366
Perform send locally
6368
Prepare params
6370
Send data

Fig. 63A

Operand ↓	PM	Preferred embodiment Send processing
201	O	Sending an auto-dial # updates appropriate recipient MS storage so that a recipient user can subsequently auto-dial the auto-dial # with a minimal user interface action. Preferably, the recipient MS user is appropriately and immediately notified of the receipt. Preferably, the send command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent user browse of the accompanying message and a date/time stamp of when sent, and for automated speed dialing of the # in response to a user action to auto-dial. Various embodiments will save to LBX History how many times, and when, the auto-dial # was used to perform automated speed dialing.
203	O	Sending a web link updates appropriate recipient MS storage so a recipient user can subsequently invoke (transpose to) the link, for example in a browser, with a minimal user interface action. Preferably, the recipient MS user is appropriately and immediately notified of the receipt. Preferably, the send command data is maintained to LBX History, a historical call log (e.g. incoming), browser history data, browser favorites, or other useful storage for subsequent user browse of the accompanying message and a date/time stamp of when sent, and for invocation of the link within a MS browser in response to a user action to use the link. Various embodiments will save to LBX History how many times, and when, the weblink was invoked.
205	E	Sending an email causes interface to the email delivery system (e.g. SMTP API) for sending the email body parameter. In one embodiment, the body is assumed to be the body of the email. In another embodiment, the body is attached with or without attachment(s). Attachments are preferably referenced with an appropriate syntax in the body specified. In another embodiment, the body is parsed for determining and using the best delivery options. The email will arrive to a recipient like other emails. Attributes can be set as is customary for email attributes (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
207	E	Sending an SMS message causes interface to the sms message delivery system (e.g. SMTP API) for sending the sms message. The email interface can be used provided the sms message length maximum is observed. In one embodiment, the message parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the sms message. In another embodiment, only a null subj/msg is supported. The message will arrive to a recipient like other sms messages. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

Fig. 63B-1

Operand ↓	PM	Preferred embodiment Send processing
209	E	Sending a broadcast email causes interface to the email delivery system (e.g. SMTP API) for sending the email body parameter. In one embodiment, the body is assumed to be the body of the email. In another embodiment, the body is attached with or without attachment(s). Attachments are preferably referenced with an appropriate syntax in the body specified. In another embodiment, the body is parsed for determining and using the best delivery options. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery is set since this is a broadcast.
211	E	Sending an SMS broadcast message causes interface to the sms message delivery system (e.g. SMTP API) for sending the sms message parameter. The email interface can be used provided the sms message length maximum is observed. In one embodiment, the message parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the sms message. In another embodiment, only a null subj/msg is supported. The message will arrive to a recipient like other messages. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is set since this is a broadcast.
213	O	Sending an indicator updates appropriate recipient MS storage so that the currently focused user interface object (e.g. window titlebar) of the MS user interface is modified with the indicator. If there are no active user interface objects in the current MS user interface, then an appropriate alert area of the currently focused interface is to display the indicator. The user can clear (remove) the indicator when desired. Preferably, the indicator is used for modifying other focused objects (e.g. titlebars) or other focused areas in the user interface so as to not get overlooked. For example, as the user navigates and surfaces/focuses new user interface objects, the indicator remains visible on the newly focused object. Preferably, the indicator is selectable by the user of the MS for showing all other send command parameters associated, as well as a date/time stamp of when sent. In other embodiments, the most recently displayed indicator is displayed in the appropriate focused area, but the user can conveniently select any indicators which were sent in history at some point in time for sought indicator information by selecting the currently displayed indicator and then requesting to browse/scroll history of previously delivered indicators (with options to see details). Preferably, the send command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use. Some title bar management methods include various IBM Technical Disclosure Bulletins from 1991 through 1995 (e.g. DA8-92-0910 "Originator Identified Direct Access Mail Basket Title Bar Mechanism", DA8-93-0061 "Roving Title Bar", DA8-93-0223 "Roving Title Bar Status", etc).

Fig. 63B-2

Operand ↓	PM	Preferred embodiment Send processing
215	O	Sending an application causes invocation of the application at the recipient MS. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The attributes parameter can be used for how to start the application, for example to flag whether to start an additional instance if the application is already running at the MS (provided multiple instances are supported). The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the application invocation reference. An error is logged if the app parameter is not found for launch. Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
217	E	Sending a document causes interface to the email delivery system (e.g. SMTP API) for appropriately sending the document. The doc parameter is preferably a fully qualified path name, or suitable reference, to the document which may have a document type (e.g. by file extension, document parse, or document location). The document type is used for setting proper email attachment settings and perhaps the attributes parameter. Depending on the document type, the document may form the email body or be an attachment. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
219	E	Sending a file causes interface to the email delivery system (e.g. SMTP API) for appropriately sending the file. The path parameter is preferably a fully qualified path name, or suitable reference, to the file which should have a file type (e.g. by file extension, file parse, or file location). The file type is used for setting proper email attachment settings and perhaps the attributes parameter. Depending on the file type, the file may form the email body or be an attachment. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
221	O	Sending content causes the content to be sent to the recipient MS in a manner which is appropriate for where the content is stored and how it is to be subsequently presented. The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the send command data is maintained to LBX History, a historical content log, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content in response to an applicable user action. Attributes may be set for special content handling.

Fig. 63B-3

Operand →	PM	Preferred embodiment Send processing
223	O	Sending a Database (DB) object causes the DB object to be sent to the recipient MS in a manner which is appropriate for subsequent import DB or table(s). The DB-obj parameter takes on many syntaxes for sending any subset of a database object, such as an entire database, table(s), certain rows, certain columns, etc. In one embodiment, a qualified database form is used such as: Owner:DatabaseName:TableName for sending the entire table (can use table name wildcard for multiple tables). In another embodiment, Owner:DatabaseName:"...SQL query... shall return the data that is to be sent, preferably in a comma delimited or tab delimited form (as specified in the attributes parameter). Preferably, the send command data is maintained to LBX History, a database log, or other useful storage (subj/msg to document the transaction) for subsequent user browse of the accompanying data and a date/time stamp of when sent, and for DB query manager browse of the data in response to a user action for browse. One preferred embodiment enables the data for easy import to a variety of database destinations, preferably via the same DB query mgr interface(s) used for browsing.
225	O	Sending data causes reading the current value of the data at the MS where the send command action is being executed and then sending the current value to the recipient MS for informative purposes. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname"). In the preferred embodiment, all occurrences found on the MS and information including its value about the occurrence is presented to the user. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. Preferably, the data value is maintained to LBX History, a historical log, or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its value and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	O	Sending a semaphore causes reading the current value of the semaphore at the MS where the send command action is being executed and then sending the current value to the recipient MS for informative purposes. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value (cleared or set) is maintained to LBX History, a historical log, or other useful storage for subsequent user browse, or programmatic access, of the semaphore name, its value and date/time stamp of when sent, and for presentation of the semaphore value in response to a user action to show it.

Fig. 63B-4

Operand 1	PM	Preferred embodiment Send processing
229	E	Sending a directory causes interface to the email delivery system (e.g. SMTP API) for sending the directory. In one embodiment, the directory is assumed to be the body of the email (e.g. when attributes parameter indicates it is to be a description only of the directory) for sending information about the directory such as # files, nesting of folders, sizes, and any useful file system characteristic(s) or statistics of the directory. In another embodiment, or as specified with an additional parameter (or in attributes), the directory is compressed and encoded as an attachment. In another embodiment, the directory is sent as individually attached files (as indicated to send that way by new or attributes parameter). The email will arrive to a recipient like other emails. The attribute parameter can be used for conventional email attributes as well as new attributes which affect directory data processing.
231	O	Sending an application context causes invocation of the application at the recipient MS and then executing a macro within the application context. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. The attributes parameter can be used for how to start the application, for example to flag whether to start an additional instance if the application is already running at the MS (provided multiple instances are supported), and to specify the type of macro parameter being specified, or to specify a speed for processing individuals of the macro. The msg/subject parameter may be useful for maintaining to LBX history useful information with record of the application context invocation reference. An error is logged if the app parameter is not found for launch.
233	E	Sending the focused user interface object causes interface to the email delivery system (e.g. SMTP API) for sending an image (preferably .JPG) of the currently focused user interface object as an attachment. The "<alt><prtscrn>" constant string parameter is a syntactical string representation for the keystroke sequence for performing the MS focused user interface capture action. A similar syntax can be used to specify a different keystroke sequence (1st parameter) for the same functionality. The email will arrive to a recipient like other emails. The attributes parameter can be set for which format to send, in which case a conversion may take place prior to sending (depends on embodiment). Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

Fig. 63B-5

Operand ↓	PM	Preferred embodiment Send processing
235	O	Sending user interface control causes redirecting the keystroke macro to input of the recipient MS as if it were entered by the MS user. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard user-interface/input. In another embodiment, the macro is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement, or cancellation, prior to executing the macro at the MS. This allows the user time to get the MS user interface in a desirable state if necessary for running the macro, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of user interface control sent.
237	O	Sending input causes redirecting the input to the iodev parameter input device stream of the recipient MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process. The input parameter is preferably a file, path, or accessible variable name containing a datastream (e.g. macro) recognizable by the iodev connected device. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement, or cancellation, prior to redirecting the input parameter datastream at the MS, or to specify a speed for processing individuals of the input. This allows the user time to get the MS user interface, and any iodev devices, in a desirable state if necessary for running the input, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of the user interface control having been sent.
239	O	Sending output causes redirecting the output to the iodev parameter output device stream of the recipient MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process. The output parameter is preferably a file, path, or accessible variable name containing a datastream (e.g. macro) recognizable by the iodev connected device. The attributes parameter can be used for whether or not to first display the subj/msg to the recipient MS user for user acknowledgement prior to redirecting the output parameter datastream at the MS, or to specify a speed for processing individuals of the output.. This allows the user time to get the MS user interface, and any iodev devices, in a desirable state if necessary for running the output, and to see information of the origination (i.e. Parameters). The msg/subject parameter may be useful for maintaining to LBX history information with a record of the user interface control having been sent.

Fig. 63B-6

Operand ↓	PM	Preferred embodiment Send processing
241	O	Sending an alert updates appropriate recipient MS storage so that a recipient MS alerter process can pick up the alert and then alert the user. There are a variety of alert processes, the most basic of which monitors incoming messages and posts them to the user in an alerting manner. In one embodiment, the alert parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the alert message. In another embodiment, only a null subj/msg is supported. The attributes parameter can be for special treatment of the alert by an alerter process.
243	O	See Notify Command for identical processing.
245	O	See Notify Command for identical processing.
247	O	See Notify Command for identical processing.
249	O	See Notify Command for identical processing.
251	O	Sending a calendar object causes interface to the recipient MS calendar/scheduling system for sending/scheduling the calendar object parameter. The calobj parameter contains the date/time stamp of when to schedule the object, or a special syntax constant for "now", "first available per recipient and sender availability", "by end of the week pending availability", or other reasonable constants for when to schedule the calendar object. In one embodiment, the calendar object is assumed to be a newly scheduled calendar item for placement to the calendar of recipients. In another embodiment, the calendar object (e.g. data or file containing parsable syntax) contains directives for what actions exactly to perform to the calendar application interface. In another embodiment, the email system is the transport to deliver the calendar object or calendar actions to recipients. Attributes can be set as is customary for calendar entries (attendance required, emergency meeting, recurring/weekly/monthly meeting, etc). The attributes parameter may be used for performing other actions/functions in the calendaring interface.
253	O	Sending an address book (AB) object causes interface to the AB system for sending/entering the AB object parameter at the recipient MS. In one embodiment, the AB object is assumed to be a newly entered AB entry (e.g. contact reference name) for creation in the AB of recipients. In another embodiment, the AB object parameter contains directives (e.g. data or file containing parsable syntax) for what actions exactly to perform to the AB application interface. Attributes can be set as may be customary for AB entries (customer, peer, manager, friend, family, etc). The attributes parameter may be used for performing other actions/functions in the AB interface.
. . .		

Fig. 63B-7

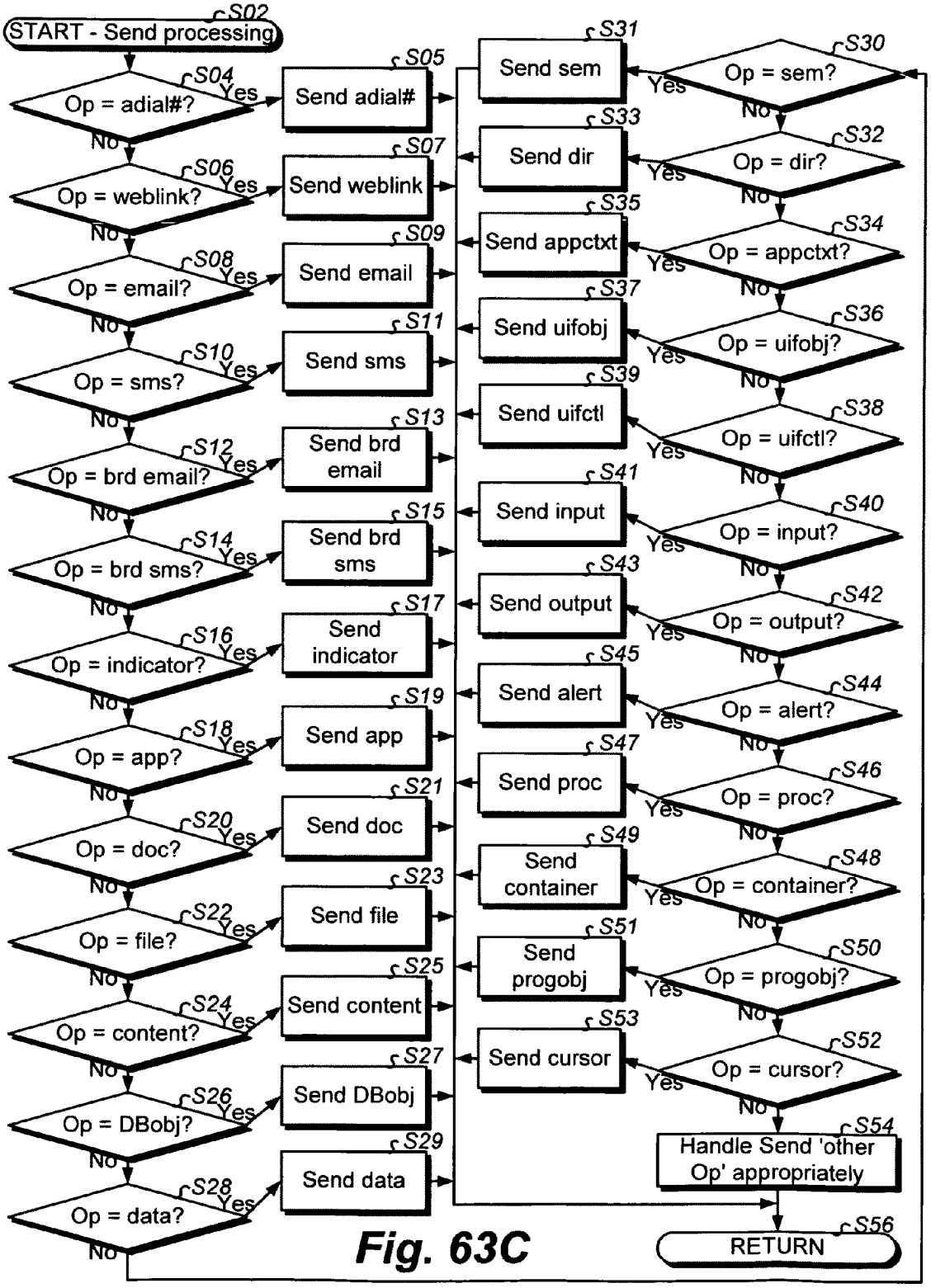

S02
START - Send processing
S04
Op = adial#?
Yes
No
S05
Send adial#
S06
Op = weblink?
Yes
No
S07
Send weblink
S08
Op = email?
Yes
No
S09
Send email
S10
Op = sms?
Yes
No
S11
Send sms
S12
Op = brd email?
Yes
No
S13
Send brd email
S14
Op = brd sms?
Yes
No
S15
Send brd sms
S16
Op = indicator?
Yes
No
S17
Send indicator
S18
Op = app?
Yes
No
S19
Send app
S20
Op = doc?
Yes
No
S21
Send doc
S22
Op = file?
Yes
No
S23
Send file
S24
Op = content?
Yes
No
S25
Send content
S26
Op = DBobj?
Yes
No
S27
Send DBobj
S28
Op = data?
Yes
No
S29
Send data
S31
Send sem
S33
Send dir
S35
Send appctxt
S37
Send uifobj
S39
Send uifctl
S41
Send input
S43
Send output
S45
Send alert
S47
Send proc
S49
Send container
S51
Send progobj
S53
Send cursor
S30
Op = sem?
Yes
No
S32
Op = dir?
Yes
No
S34
Op = appctxt?
Yes
No
S36
Op = uifobj?
Yes
No
S38
Op = uifctl?
Yes
No
S40
Op = input?
Yes
No
S42
Op = output?
Yes
No
S44
Op = alert?
Yes
No
S46
Op = proc?
Yes
No
S48
Op = container?
Yes
No
S50
Op = progobj?
Yes
No
S52
Op = cursor?
Yes
No
S54
Handle Send 'other Op' appropriately
S56
RETURN

Fig. 63C

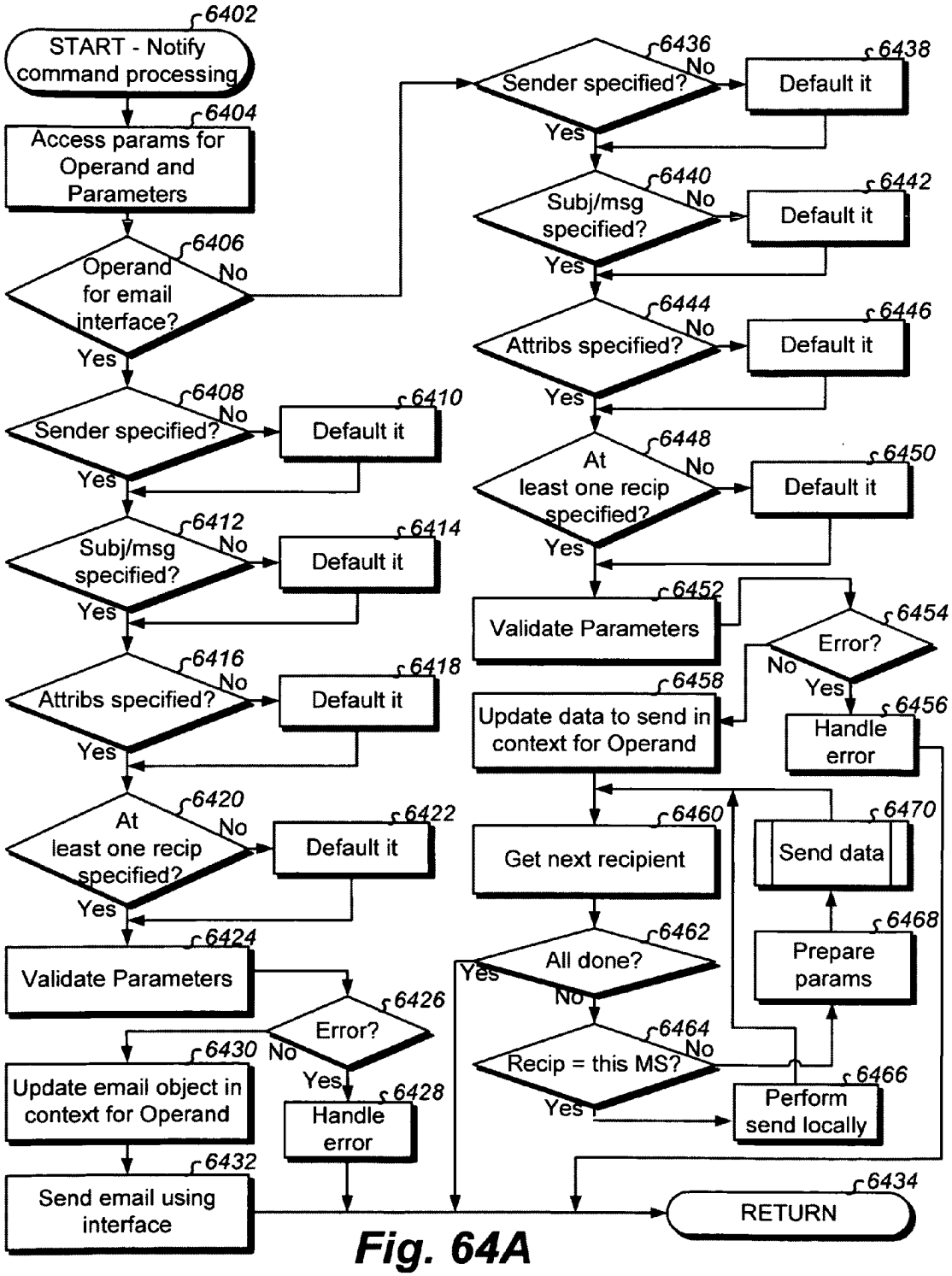

6402
START - Notify command processing
6404
Access params for Operand and Parameters
6406
Operand for email interface?
No
Yes
6408
Sender specified?
No
Yes
6410
Default it
6412
Subj/msg specified?
No
Yes
6414
Default it
6416
Attribs specified?
No
Yes
6418
Default it
6420
At least one recip specified?
No
Yes
6422
Default it
6424
Validate Parameters
6426
Error?
No
Yes
6428
Handle error
6430
Update email object in context for Operand
6432
Send email using interface
6434
RETURN
6436
Sender specified?
No
Yes
6438
Default it
6440
Subj/msg specified?
No
Yes
6442
Default it
6444
Attribs specified?
No
Yes
6446
Default it
6448
At least one recip specified?
No
Yes
6450
Default it
6452
Validate Parameters
6454
Error?
No
Yes
6456
Handle error
6458
Update data to send in context for Operand
6460
Get next recipient
6462
All done?
Yes
No
6464
Recip = this MS?
No
Yes
6466
Perform send locally
6468
Prepare params
6470
Send data

Fig. 64A

Operand	PM	Preferred embodiment Notify processing
201	O	Notifying with an auto-dial # automatically performs call processing to auto-dial the auto-dial #. Preferably, the recipient MS user is called by the MS as a normal phone call would be made. In one embodiment, multiple recipients are called "back to back" after the previous recipient call terminates. In another embodiment, a multiple line party call is made with an automated manner with all recipients. The attributes parameter can indicate which embodiment to use, and can be used for specialized call processing (collect, prepaid account check, hide caller id, etc). Preferably, the notify command data and call data is maintained to LBX History, a historical call log (e.g. incoming), with the accompanying subj/msg and a date/time stamp of when sent, and for future repeated automated speed dialing of the # in response to a user action to auto-dial. Various embodiments will save to LBX History how many times, and when, the auto-dial # was used to perform automated speed dialing, along with call details such as direction of call, parties to the call, features of the call, or other call characteristics. In one embodiment, the recipient ID is one to one with the called #. In another embodiment, the recipient ID is used to find the associated called number. Preferably, an existing API is used to accomplish processing. Automatic dialing through a variety of interfaces is well known in the art, and depends on the software development environment. Conventional processing side affects of automated calling should occur like the action was manual (e.g. log update).
203	O	Notifying with a web link automatically invokes (transposes to) at the recipient MS the link, for example in a browser, with a minimal (if any) user interface action. In one embodiment, the link includes URL parameter(s) (e.g. ?p1=xyz). In another embodiment, all recipients are passed to the link with appended URL parameter(s) (e.g. ?ids=Recip1;Recip2;...RecipN). An alternate embodiment fires form variables to the loaded page with the same URL variables. The attributes parameter can indicate which embodiment to use, and how to invoke the link (e.g. use currently focused window, use an active browser window, spawn new browser window, etc). Preferably, the notify command data is maintained to LBX History, a historical call log (e.g. incoming), browser history data, browser favorites, or other useful storage for subsequent user browse of the accompanying subj/msg and a date/time stamp of when sent, and for invocation of the link within a MS browser in response to a user action to use the link again in the future. Various embodiments will save to LBX History how many times, and when, the weblink was invoked.
205	E	See Send Command for identical processing.
207	E	See Send Command for identical processing.
209	E	See Send Command for identical processing.
211	E	See Send Command for identical processing.
213	O	See Send Command for identical processing.
215	O	See Send Command for identical processing.

Fig. 64B-1

Operand ↓	PM	Preferred embodiment Notify processing
217	E	See Send Command for identical processing.
219	E	See Send Command for identical processing.
221	O	Notifying with content causes the content to be presented at the recipient MS upon delivery in a manner which is appropriate for the content type. The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the notify command data is maintained to LBX History, a historical content log (e.g. incoming), browser history data, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content. Various embodiments will save to LBX History how many times, and when, the content was presented.
223	O	Notifying with a Database (DB) object causes the DB object (i.e. qualified database with access query string) to be modified with the query parameter. The query parameter is used to perform any query against the specified DB-database (DB-obj), preferably a query that only returns a return code (e.g. causes alteration). Preferably, the notify command data is maintained to LBX History, a database log (e.g. incoming), or other useful storage for subsequent query use and a date/time stamp of when sent, and for DB query manager browse/use of the query in response to an applicable user action. Other params are for documentary purposes when information is saved. In some embodiments, an appropriate SQL client interface (e.g. SQLNET API) is used to carry out processing, or a suitable DB API is used.
225	O	Notifying data causes modifying the value of the data at the recipient MS (set data to value). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	O	Notifying a semaphore causes modifying the value of the semaphore at the recipient MS. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.

Fig. 64B-2

Operand ↓	PM	Preferred embodiment Notify processing
229	E	See Send Command for identical processing.
231	O	See Send Command for identical processing.
233	E	See Send Command for identical processing.
235	O	See Send Command for identical processing.
237	O	See Send Command for identical processing.
239	O	See Send Command for identical processing.
241	O	Notifying with an alert presents the alert to each recipient MS. Depending on attributes parameter settings, the alert may be asynchronously presented to an alert area, synchronously alerted and requiring a user action to acknowledge, logged to special file, or other reasonable alert method(s). Fig. 75B processing will cause the alert to be presented to the MS user. In one embodiment, the alert parameter is identical to the msg/subj parameter. In another embodiment, the two parameters are concatenated, or formed in a complimentary manner, to highlight the subj/msg parameter from the alert message. In another embodiment, only a null subj/msg is supported. Various embodiments will support different alert content types and applicable processing as indicated by the attributes parameter. Preferably, an appropriate API is made available for processing.
243	O	Notifying a process causes sending an operating system signal (see UNIX signaling) to the process with Process ID (PID) of the pid parameter. A numeric value parameter (e.g. 0 or 1) may be communicated with the signal. Depending on attributes parameter settings, another embodiment accesses the pid parameter as a process identifier parameter which is used to lookup the operating system PID prior to signaling. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the application invocation reference. An error can be logged if the process is not found for signaling.
245	O	Notifying a container causes launch of a MS file manager to examine the contents of a MS system container having the path in the container parameter (e.g. c:\dir1\subdir3). The attributes parameter can be used for how to start the file manager, for example to flag whether to start an additional instance if the file manager is already running at the MS (provided multiple instances are supported), otherwise an existing instance is updated for the container, or a new instance is started for the container. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a sent date/time stamp, with record of the application invocation reference. An error can be logged if the file manager is not found for launch, or if the container is invalid.

Fig. 64B-3

Operand ↓	PM	Preferred embodiment Notify processing
247	O	Notifying a program object causes acting on a specified program object (per attribute parameter) with the specified data at the recipient MS. The progobj parameter is the linked run time symbolic name accessible to charter processing of the present disclosure for third party plug-in processing. The progobj parameter can be a variable name, function name, object name, queue name, procedure name, or semaphore name accessed at run time by the symbolic name evaluation during charter processing. The binary data parameter is used to modify the program object (variable name set Least Significant Bit (LSB) to Most significant Bit (MSB) right to left intuitive Motorola processing byte/bit order until bits set or unmatched, function name invoked with respective data bytes pushed to the stack prior to invocation, object name data public data area initialized with the data parameter on a byte to byte basis, queue name entry inserted using the data parameter as a typecast data record of bits, procedure name invoked with respective data bytes pushed to the stack prior to invocation, or semaphore name set with clear for a null data parameter, else a set action). Alternately, an Intel reverse byte order can be used to apply the data Parameter. The attributes parameter indicates which variety of progobj is specified, and can be used to indicate a byte order data mapping method to use. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent, with record of the program object invocation. An error can be logged if the progobj parameter is not resolvable. Appropriate MS O/S interfaces are used.
249	O	Notifying a cursor causes modifying a recipient MS user interface cursor in accordance with direction by the attributes parameter. The cursor parameter can be a suitable cursor bitmap file reference, suitable animated cursor file, predefined appearance type, or predefined behaving cursor. The attributes parameter further distinguishes which cursor modification is being requested. The msg/subject parameter may be useful for maintaining to LBX history useful information, along with a date/time stamp when sent. An error is logged if there is no active cursor in the user interface. An appropriate MS API is used, depending on the development environment.
251	O	See Send Command for identical processing.
253	O	See Send Command for identical processing.
. . .		

Fig. 64B-4

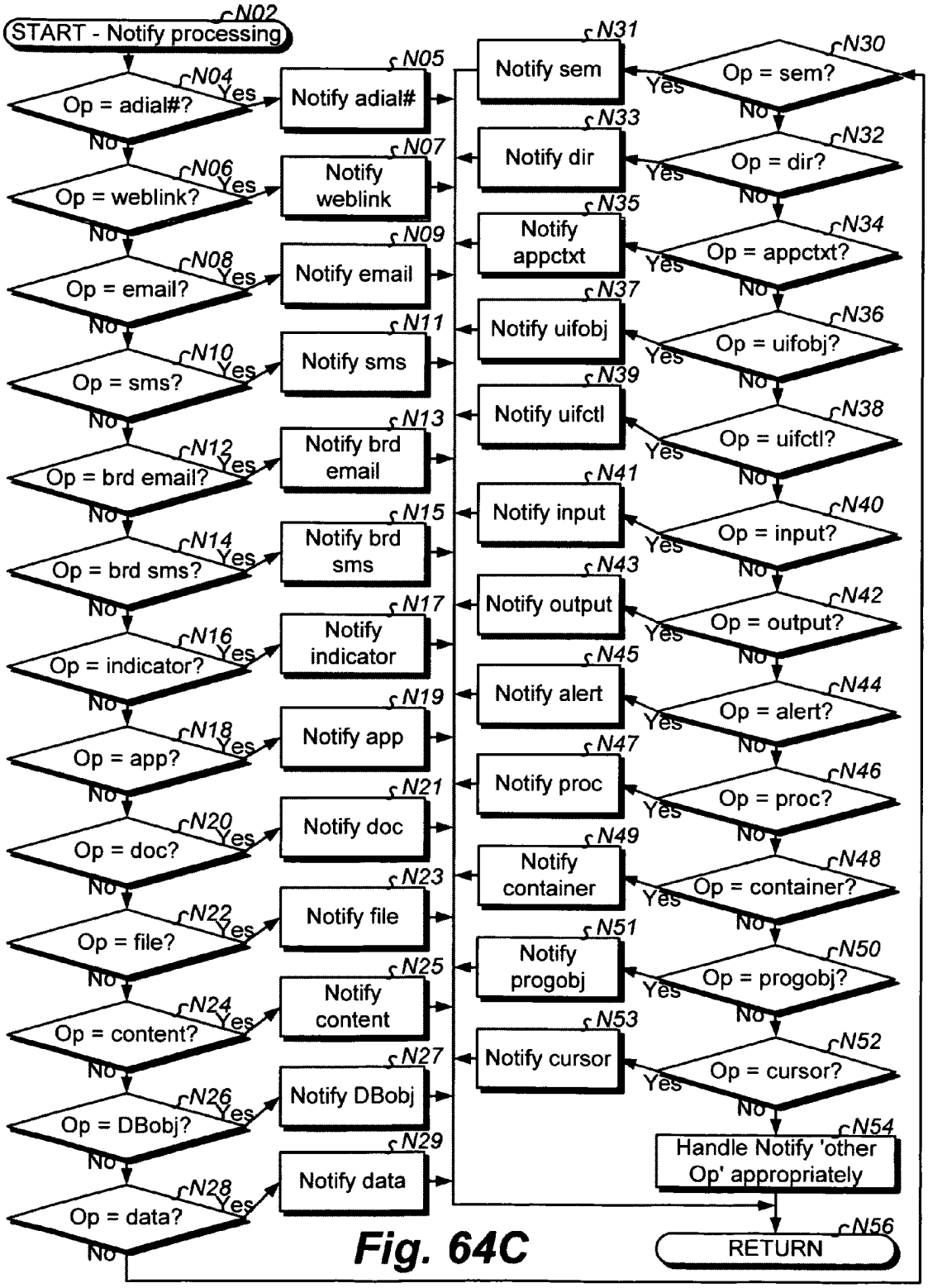

N02
START - Notify processing
N04
Op = adial#?
Yes
No
N05
Notify adial#
N06
Op = weblink?
Yes
No
N07
Notify weblink
N08
Op = email?
Yes
No
N09
Notify email
N10
Op = sms?
Yes
No
N11
Notify sms
N12
Op = brd email?
Yes
No
N13
Notify brd email
N14
Op = brd sms?
Yes
No
N15
Notify brd sms
N16
Op = indicator?
Yes
No
N17
Notify indicator
N18
Op = app?
Yes
No
N19
Notify app
N20
Op = doc?
Yes
No
N21
Notify doc
N22
Op = file?
Yes
No
N23
Notify file
N24
Op = content?
Yes
No
N25
Notify content
N26
Op = DBobj?
Yes
No
N27
Notify DBobj
N28
Op = data?
Yes
No
N29
Notify data
N31
Notify sem
N33
Notify dir
N35
Notify appctxt
N37
Notify uifobj
N39
Notify uifctl
N41
Notify input
N43
Notify output
N45
Notify alert
N47
Notify proc
N49
Notify container
N51
Notify progobj
N53
Notify cursor
N30
Op = sem?
Yes
No
N32
Op = dir?
Yes
No
N34
Op = appctxt?
Yes
No
N36
Op = uifobj?
Yes
No
N38
Op = uifctl?
Yes
No
N40
Op = input?
Yes
No
N42
Op = output?
Yes
No
N44
Op = alert?
Yes
No
N46
Op = proc?
Yes
No
N48
Op = container?
Yes
No
N50
Op = progobj?
Yes
No
N52
Op = cursor?
Yes
No
N54
Handle Notify 'other Op' appropriately
N56
RETURN

Fig. 64C

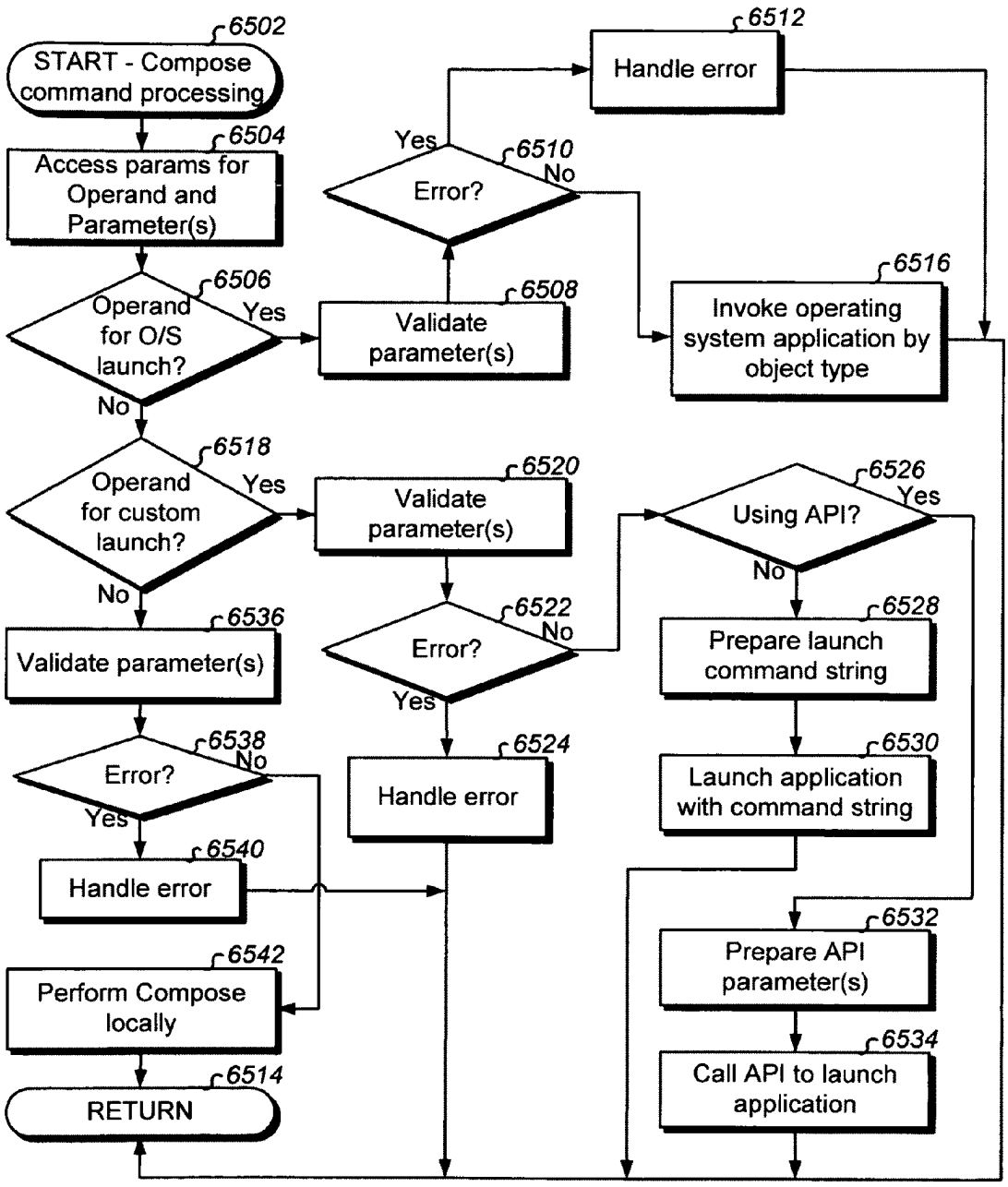

6502
START - Compose command processing
6504
Access params for Operand and Parameter(s)
6506
Operand for O/S launch?
Yes
No
6508
Validate parameter(s)
6510
Error?
Yes
No
6512
Handle error
6516
Invoke operating system application by object type
6518
Operand for custom launch?
Yes
No
6520
Validate parameter(s)
6522
Error?
Yes
No
6524
Handle error
6526
Using API?
Yes
No
6528
Prepare launch command string
6530
Launch application with command string
6532
Prepare API parameter(s)
6534
Call API to launch application
6536
Validate parameter(s)
6538
Error?
Yes
No
6540
Handle error
6542
Perform Compose locally
6514
RETURN

Fig. 65A

Operand →	PM	Preferred embodiment Compose processing
201	C	Composing an auto-dial # launches the MS phone number calling interface with the auto-dial # parameter defaulted for making the call. Once launched, the user can make a very simple confirmation action for placing the call to the auto-dial #. Call processing takes place as though the user manually launched the dialing application, entered the auto-dial # and then is ready to decide if the call should be placed. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. outgoing when call placed), or other useful storage for subsequent use.
203	S	Composing a web link launches a MS browser and defaults the link as though the user had entered it manually. The user can subsequently invoke (transpose to) the link if desired with a minimal action (e.g. click ok). The link may include appended URL parameters (e.g. "?v=yes&T=go" for customized web page processing). An alternate embodiment can fire form variables for active web page processing using URL parameters specified or using the params parameter. Processing takes place as though the user manually launched the browser application, entered the weblink and then is ready to decide if the link should be invoked. Appropriate MS storage is updated and subsequently processed as though the user had entered the weblink in the browser manually. Preferably, the compose command data is maintained to LBX History, a historical log (web page load history when invoked), or other useful storage for subsequent use.
205	C	Composing an email causes interface to the email delivery system for invoking the create email interface and defaulting the appropriate email fields with the passed parameters. The user can subsequently send the email with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create email application, entered the fields of the email form with passed parameters, and then is ready to decide if the email should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create email interface and entered the email information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface is preferably used. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).

Fig. 65B-1

Operand ↓	PM	Preferred embodiment Compose processing
207	C	Composing an sms message causes interface to an appropriate messaging delivery system for invoking the create message interface and defaulting the appropriate message fields with the passed parameters. The user can subsequently send the message with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create message application, entered the fields of the messaging form with passed parameters, and then is ready to decide if the message should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create message interface and entered message information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface, or a similar messaging interface, can be used. The message will arrive to a recipient like other sms messages. Various sms message attributes may be set (e.g. confirmation of delivery status, special handling, NLS considerations, etc).
209	C	Composing a broadcast email causes interface to the email delivery system for invoking the create email interface and defaulting the appropriate email fields with the passed parameters. The user can subsequently send the email with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create email application, entered the fields of the email form with passed parameters, and then is ready to decide if the email should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create email interface and entered the email information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface is preferably used. The email will arrive to a recipient like other emails. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is requested in attributes since this is a broadcast.

Fig. 65B-2

Operand ↓	PM	Preferred embodiment Compose processing
211	C	Composing a broadcast sms message causes interface to an appropriate messaging delivery system for invoking the create message interface and defaulting the appropriate message fields with the passed parameters. The user can subsequently send the message with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the create message application, entered the fields of the messaging form with passed parameters, and then is ready to decide if the message should be sent, or further edited, or possibly cancelled. Appropriate MS storage is updated and subsequently processed as though the user manually started the create message interface and entered message information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard email POP or mailbox interface, or a similar messaging interface, can be used. The message will arrive to a recipient like other messages. Various attributes can be set (e.g. special handling, NLS considerations, etc), but preferably, no confirmation of delivery status is requested in attributes since this is a broadcast.
213	O	See Send Command for identical processing.
215	C	Composing an application causes invocation of the application at the MS. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The params parameter can be used for command line, or string to append, or pass, to the app/path parameter, for how to start the application (e.g. with parameters). Processing takes place as though the user manually launched the application (and with any optional params). Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
217	S	Composing a document causes invocation of the appropriate application at the MS in accordance with the object type as though the user selected the document for automatically being associated to the correct application when opening the document. The doc parameter may be preferably a fully qualified path name to the document. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
219	S	Composing a file causes invocation of the appropriate application at the MS in accordance with the file type of the fully qualified path name of the file as though the user selected the file for automatically being associated to the correct application when opening the document. Processing takes place as though the user manually launched the application for the specified file. The path parameter is preferably a fully qualified path name to the file. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 65B-3

Operand →	PM	Preferred embodiment Compose processing
221	O	Composing content causes invocation of the appropriate application at the MS in accordance with the content as though the user selected the content for automatically being associated to the correct application when opening the content. Processing takes place as though the user manually launched the applicable application for the content. The path parameter is preferably a fully qualified specification to the content. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
223	C	Composing a DB object causes invocation of the appropriate database query manager DB object creation interface to a context complementary to the type of DB object as though the user started the query manager and manually entered the DB object for creation. Processing takes place as though the user manually launched the query manager, entered the fields of the database object form, and then is ready to further work with the starting template of DB object information. In one embodiment, the DB-obj parameter contains directives for automatically populating specified data to a particular Query Manager create object interface. In another embodiment, the DB-obj parameter is specified for an existing DB object for then being opened by the query manager for further review or work. Appropriate MS storage is updated and subsequently processed as though the user had entered information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use.
225	O	Composing data causes modifying the value of the data at the MS (analogous to a Notify data action -- set data to value). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.
227	O	Composing a semaphore causes modifying the value of the semaphore at the MS (analogous to a Notify sem action -- set sem to value). An error can result if the semaphore is not resolvable for the attempt. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (i.e. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.

Fig. 65B-4

Operand 1	PM	Preferred embodiment Compose processing
229	S	Composing a directory causes invocation of the appropriate application (e.g. file system manager) at the MS as though the user selected the directory for automatically being associated to the correct file management application when opening the directory. Processing takes place as though the user manually launched the application for working with the directory. The path parameter is preferably a fully qualified path name to the directory. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
231	C	Composing an application context causes invocation of the application at the MS and then executing a macro within the application context (analogous to a Send app context action). The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
233	S	Composing a focused user interface object causes invocation of the appropriate application (e.g. graphic application by file type embodiment .jpg, .gif, etc) at the MS as though the user manually captured the focused user interface object (e.g. Alt-Prtscrn) using the first command string syntax parameter, invoked the correct graphical application to open for the captured image, and is ready for save of the file, or for further editing. Processing takes place as though the user manually launched the application for the specified file. The embodiment's file type preference may influence which application is to be launched. The first parameter can be used to change the keystroke sequence for capture. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
235	O	Composing user interface control causes redirecting the keystroke macro to user interface input of the MS as if it were entered by the MS user (analogous to a Send user interface control action). The macro parameter is preferably a file, path, or accessible variable name containing a set of keystrokes that can be directed to standard input. In another embodiment, the macro is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 65B-5

Operand ↓	PM	Preferred embodiment Compose processing
237	O	Composing input causes redirecting the input to the iodev parameter input device stream of the MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process (analogous to a Send input action). The input parameter is preferably a file or accessible variable name containing a datastream recognizable by the iodev connected device. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
239	O	Composing output causes redirecting the output to the iodev parameter output device stream of the MS as if it were entered by the MS user, or programmatically specified to the iodev I/O device parameter by a data processing system process (analogous to a Send output action). The output parameter is preferably a file or accessible variable name containing a datastream recognizable by the iodev connected device. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
241	S	Composing an alert causes invocation of the appropriate alerter application at the MS as though the user selected the alert application, manually entered the alert parameter, and is ready to decide what to do with the alert, for example send it with a minimal action (e.g. ok), edit it, or cancel it. Processing takes place as though the user manually launched the application for creating the specified alert. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
243	O	Composing a process causes sending an operating system signal (see UNIX signaling) to the process with Process ID (PID) of the pid parameter (analogous to a Notify process action). A numeric value parameter (e.g. 0 or 1) may be communicated with the signal. An error can be logged if the process is not found for signaling. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
245	S	Composing a container causes launch of a MS container manager (e.g. file manager) to examine the contents of the container having the path in the container parameter (e.g. c:\dir1\subdir3) (analogous to a Notify container action). An error is logged if the file manager is not found for launch, or if the container is invalid. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
247	O	Composing a program object causes launch of a MS development environment application (e.g. Microsoft Visual Studio or IBM C-Set development consoles, etc), performing a search for the progobj parameter symbol, and producing search results of all occurrences for the current development working directory, mount point, or last used development repository. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 65B-6

Operand	PM	Preferred embodiment Compose processing
249	O	Composing a cursor causes invocation of the appropriate application (e.g. graphic application by file type embodiment .bmp, .ico, etc) at the MS as though the user manually launched the application for the cursor parameter, and is ready for save of the file, or for further editing, or for cancellation. Depending on the cursor parameter referenced, an appropriate application will be launched for graphics, animation, etc. The cursor parameter is preferably a fully qualified path to determine the cursor (e.g. file). Processing takes place as though the user manually launched the application for the specified file. The embodiment's file type preference will influence which application is to be launched. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
251	S	Composing a calendar object causes interface to the calendar system for invoking the create calendar object interface and defaulting the appropriate calendar interface fields with the passed parameters. The calendar object parameter may be as described for Send calendar object, except for defaulting calendar interface create object interface(s). The user can subsequently create the scheduled event with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the calendar application, entered the fields of the calendar form with passed parameters, and then is ready to decide what to do with it. Appropriate MS storage is updated and subsequently processed as though the user had manually started the calendar application and entered the calendar information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard calendaring interface is preferably used. Attributes can be set as is customary for a calendar object.
253	S	Composing an address book (AB) object causes interface to the AB system for invoking the create AB object interface and defaulting the appropriate AB interface fields with the passed parameters. The AB object parameter may be as described for Send AB object, except for defaulting AB interface create object interface(s). The user can subsequently create the AB entry with little effort, or after optional modification, with a minimal action (e.g. click ok). Processing takes place as though the user manually launched the AB application, entered the fields of the AB form with passed parameters, and then is ready to decide what to do with it. Appropriate MS storage is updated and subsequently processed as though the user manually started the AB application and entered the AB information manually. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. incoming), or other useful storage for subsequent use. A standard AB interface is preferably used. Attributes can be set as may be customary for an AB entry.
...		

Fig. 65B-7

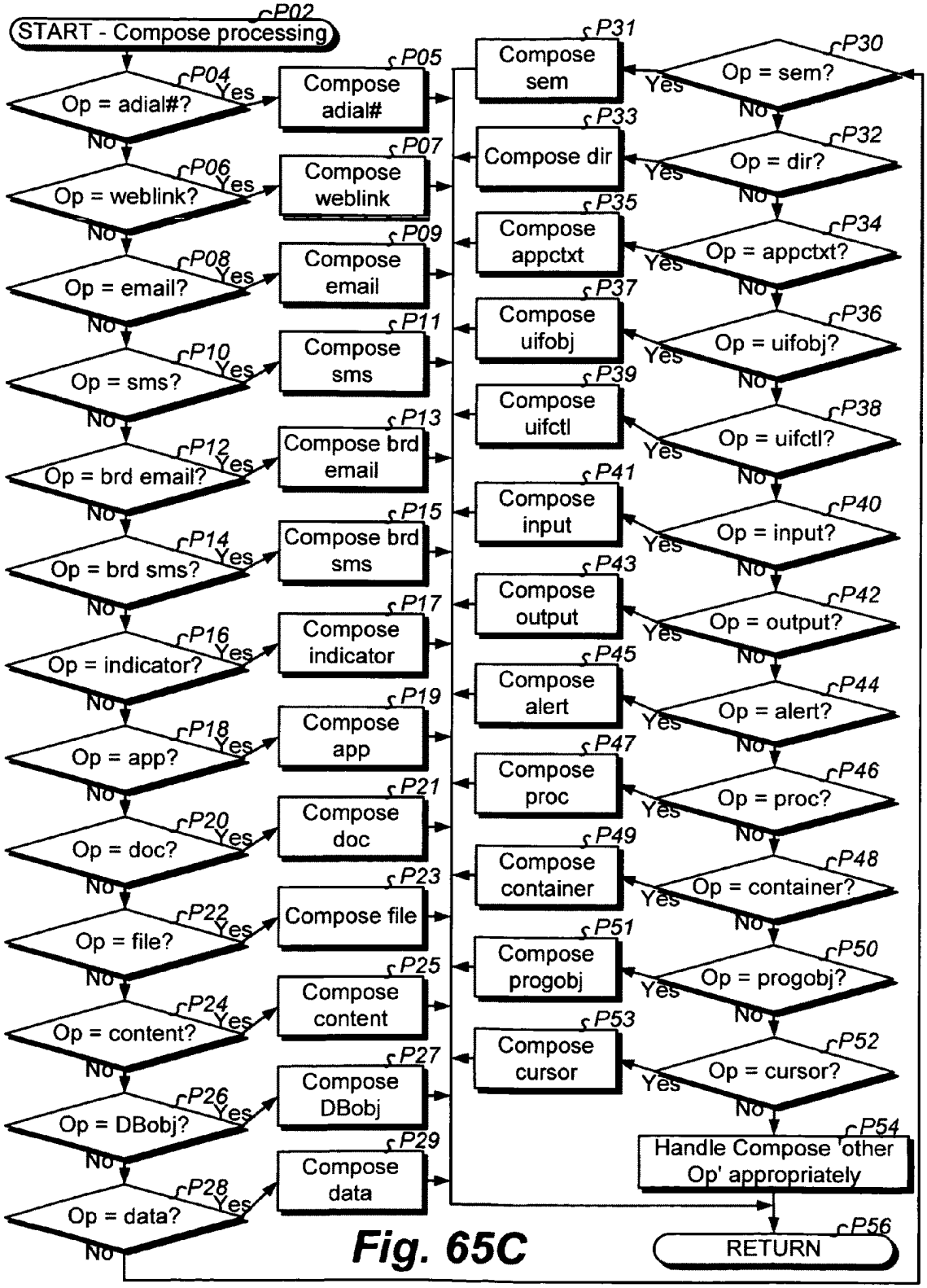

P02
START - Compose processing
P04
Op = adial#?
Yes
No
P05
Compose adial#
P06
Op = weblink?
Yes
No
P07
Compose weblink
P08
Op = email?
Yes
No
P09
Compose email
P10
Op = sms?
Yes
No
P11
Compose sms
P12
Op = brd email?
Yes
No
P13
Compose brd email
P14
Op = brd sms?
Yes
No
P15
Compose brd sms
P16
Op = indicator?
Yes
No
P17
Compose indicator
P18
Op = app?
Yes
No
P19
Compose app
P20
Op = doc?
Yes
No
P21
Compose doc
P22
Op = file?
Yes
No
P23
Compose file
P24
Op = content?
Yes
No
P25
Compose content
P26
Op = DBobj?
Yes
No
P27
Compose DBobj
P28
Op = data?
Yes
No
P29
Compose data
P31
Compose sem
P33
Compose dir
P35
Compose appctxt
P37
Compose uifobj
P39
Compose uifctl
P41
Compose input
P43
Compose output
P45
Compose alert
P47
Compose proc
P49
Compose container
P51
Compose progobj
P53
Compose cursor
P30
Op = sem?
Yes
No
P32
Op = dir?
Yes
No
P34
Op = appctxt?
Yes
No
P36
Op = uifobj?
Yes
No
P38
Op = uifctl?
Yes
No
P40
Op = input?
Yes
No
P42
Op = output?
Yes
No
P44
Op = alert?
Yes
No
P46
Op = proc?
Yes
No
P48
Op = container?
Yes
No
P50
Op = progobj?
Yes
No
P52
Op = cursor?
Yes
No
P54
Handle Compose 'other Op' appropriately
P56
RETURN

Fig. 65C

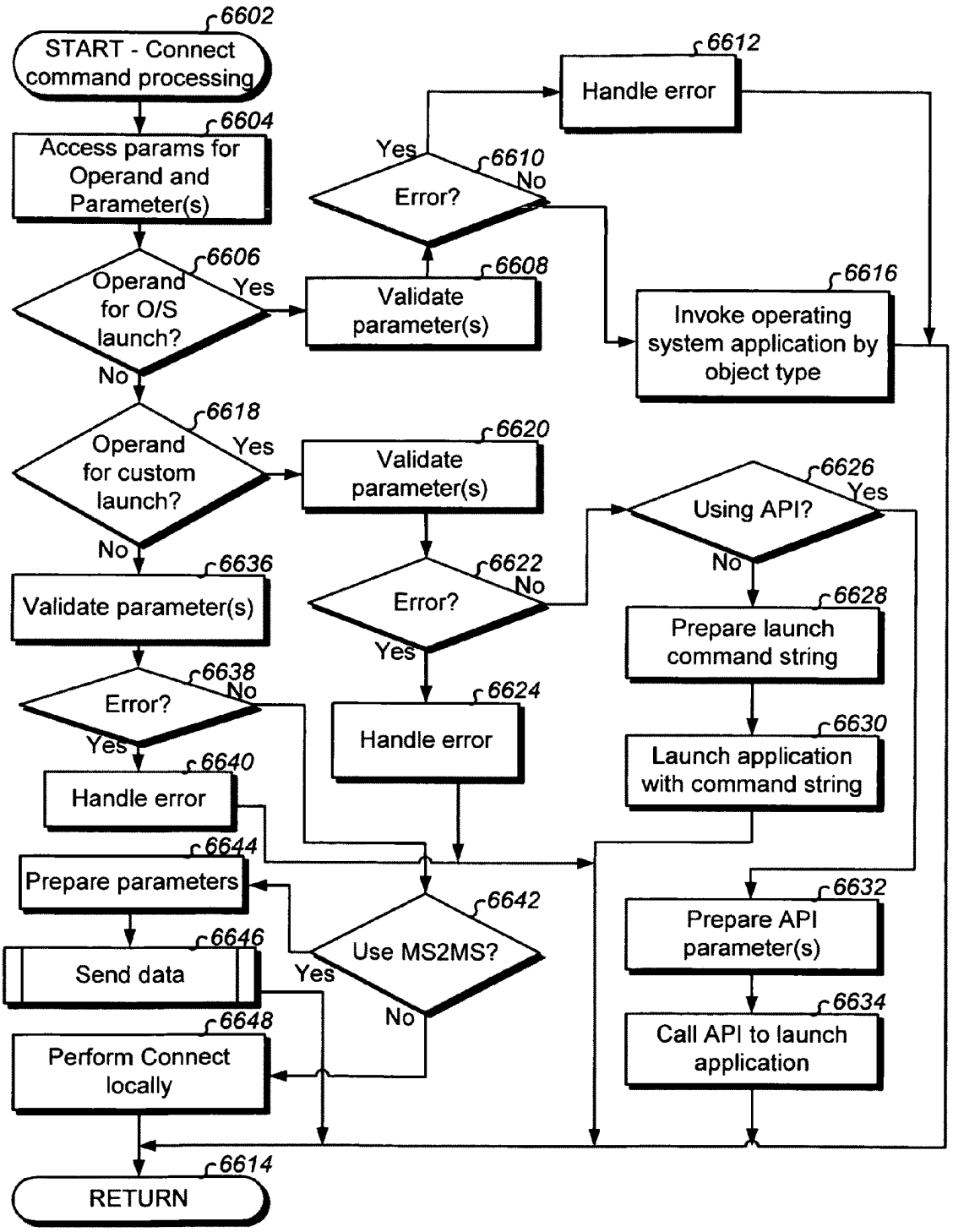

6602
START - Connect command processing
6604
Access params for Operand and Parameter(s)
6606
Operand for O/S launch?
Yes
No
6608
Validate parameter(s)
6610
Error?
Yes
No
6612
Handle error
6616
Invoke operating system application by object type
6618
Operand for custom launch?
Yes
No
6620
Validate parameter(s)
6622
Error?
Yes
No
6624
Handle error
6626
Using API?
Yes
No
6628
Prepare launch command string
6630
Launch application with command string
6632
Prepare API parameter(s)
6634
Call API to launch application
6636
Validate parameter(s)
6638
Error?
Yes
No
6640
Handle error
6642
Use MS2MS?
Yes
No
6644
Prepare parameters
6646
Send data
6648
Perform Connect locally
6614
RETURN

Fig. 66A

Operand ↓	PM	Preferred embodiment Connect processing
201	C	Connecting with an auto-dial # launches the MS phone number calling interface with the auto-dial # parameter defaulted for placing a call (like Notify autodial #). The call is actually made as though the user manually launched the dialing application, entered the auto-dial # and then chose to make the call with it. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually and then made the call. Conventional call processing takes place thereafter. Preferably, the compose command data is maintained to LBX History, a historical call log (e.g. outgoing), or other useful storage for subsequent use.
203	S	Connecting with a web link launches a MS browser and invokes (transposes to) the link as though the user had entered it manually and went to the weblink page (like Notify weblink). In one embodiment, the weblink parameter includes URL parameter(s). In another embodiment, the params parameter supports a URL command string for appending to the weblink (e.g. "?v=yes&T=go") for customized web page processing. An alternate embodiment can fire form variables for active web page processing using the params parameter. Processing takes place as though the user manually launched the browser application, entered the weblink and then loaded the weblink webpage. Appropriate MS storage is updated and subsequently processed as though the user had entered the weblink in the browser manually. Preferably, the compose command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
205	C	See Send Command for identical processing.
207	C	See Send Command for identical processing.
209	C	See Send Command for identical processing.
211	C	See Send Command for identical processing.
213	O	See Send Command for identical processing.
215	C	See Compose command for identical processing.
217	S	See Compose command for identical processing.
219	S	See Compose command for identical processing.
221	O	See Compose command for identical processing.
223	C	See Notify command for identical processing, except some applicable parameters not used.
225	O	See Compose command for identical processing.
227	O	See Compose command for identical processing.
229	O	See Compose command for identical processing.
231	C	See Compose command for identical processing.
233	S	See Compose command for identical processing.
235	O	See Compose command for identical processing.

Fig. 66B-1

Operand ↓	PM	Preferred embodiment Connect processing
237	O	See Compose command for identical processing.
239	O	See Compose command for identical processing.
241	C	Connecting with an alert causes interfacing to the alert subsystem for instantly producing the alert at the MS. Preferably, the connect command data is maintained to LBX History, a log, or other useful storage for subsequent use. The alert parameter of Notify processing is identical.
243	O	See Compose command for identical processing.
245	S	See Compose command for identical processing.
247	O	See Notify command for identical processing.
249	O	See Notify command for identical processing.
251	C	See Send Command for identical processing.
253	C	See Send Command for identical processing.
. . .		

Fig. 66B-2

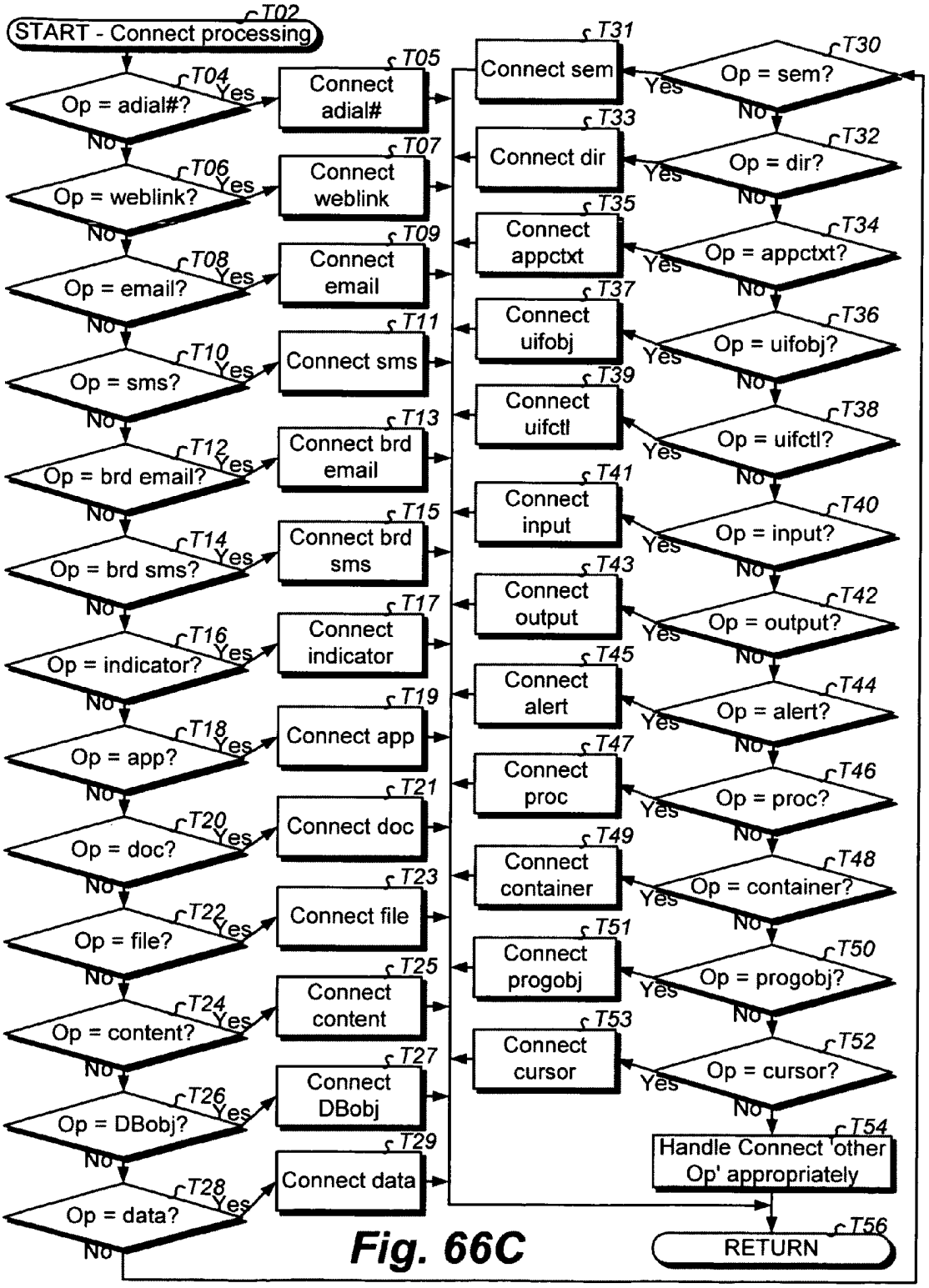

T02
START - Connect processing
T04
Op = adial#?
Yes
No
T05
Connect adial#
T06
Op = weblink?
Yes
No
T07
Connect weblink
T08
Op = email?
Yes
No
T09
Connect email
T10
Op = sms?
Yes
No
T11
Connect sms
T12
Op = brd email?
Yes
No
T13
Connect brd email
T14
Op = brd sms?
Yes
No
T15
Connect brd sms
T16
Op = indicator?
Yes
No
T17
Connect indicator
T18
Op = app?
Yes
No
T19
Connect app
T20
Op = doc?
Yes
No
T21
Connect doc
T22
Op = file?
Yes
No
T23
Connect file
T24
Op = content?
Yes
No
T25
Connect content
T26
Op = DBobj?
Yes
No
T27
Connect DBobj
T28
Op = data?
Yes
No
T29
Connect data
T31
Connect sem
T33
Connect dir
T35
Connect appctxt
T37
Connect uifobj
T39
Connect uifctl
T41
Connect input
T43
Connect output
T45
Connect alert
T47
Connect proc
T49
Connect container
T51
Connect progobj
T53
Connect cursor
T30
Op = sem?
Yes
No
T32
Op = dir?
Yes
No
T34
Op = appctxt?
Yes
No
T36
Op = uifobj?
Yes
No
T38
Op = uifctl?
Yes
No
T40
Op = input?
Yes
No
T42
Op = output?
Yes
No
T44
Op = alert?
Yes
No
T46
Op = proc?
Yes
No
T48
Op = container?
Yes
No
T50
Op = progobj?
Yes
No
T52
Op = cursor?
Yes
No
T54
Handle Connect 'other Op' appropriately
T56
RETURN

Fig. 66C

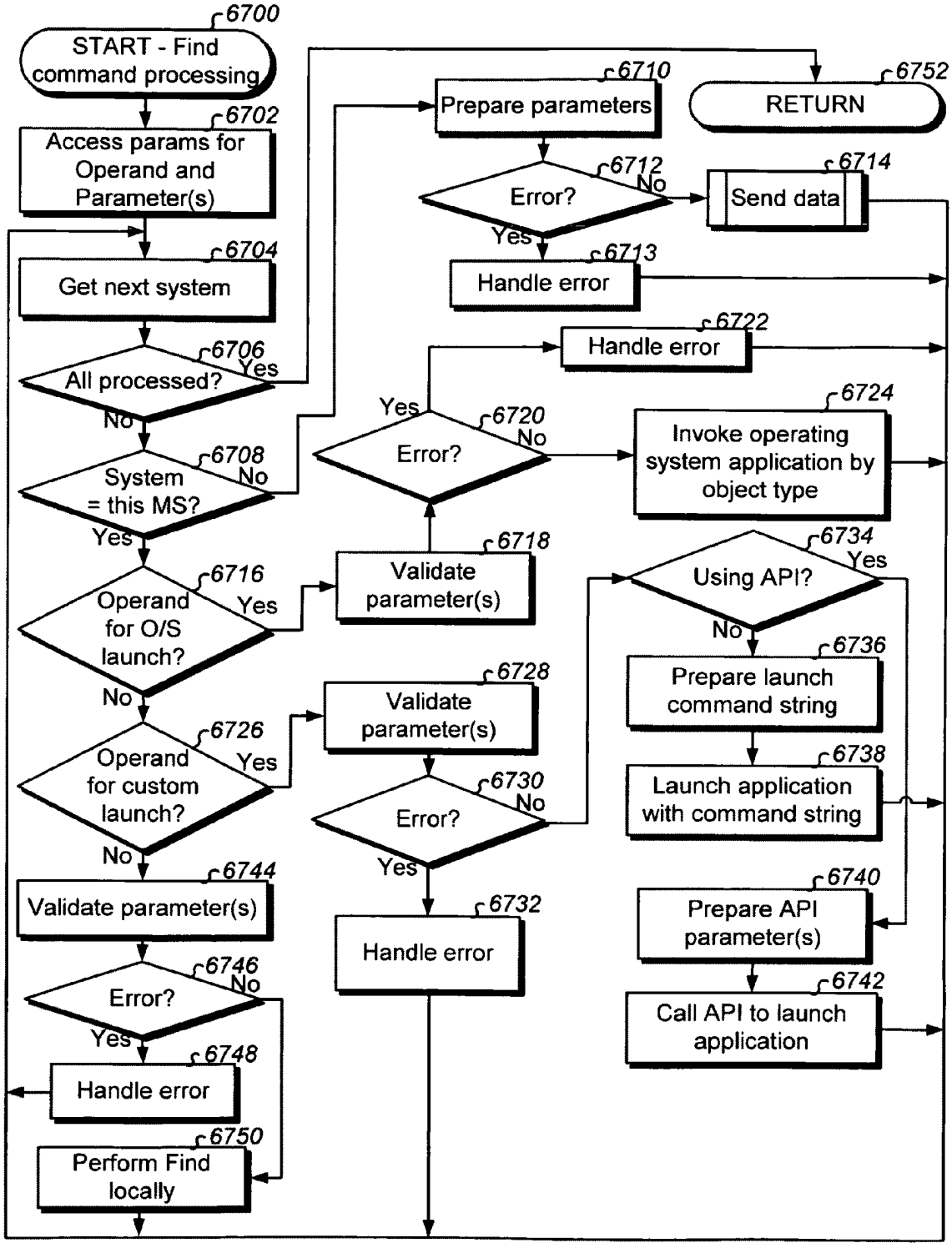

6700
START - Find command processing
6702
Access params for Operand and Parameter(s)
6704
Get next system
6706
All processed?
Yes
No
6708
System = this MS?
No
Yes
6716
Operand for O/S launch?
Yes
No
6726
Operand for custom launch?
Yes
No
6744
Validate parameter(s)
6746
Error?
No
Yes
6748
Handle error
6750
Perform Find locally
6710
Prepare parameters
6712
Error?
No
Yes
6713
Handle error
6718
Validate parameter(s)
6720
Error?
Yes
No
6722
Handle error
6728
Validate parameter(s)
6730
Error?
No
Yes
6732
Handle error
6752
RETURN
6714
Send data
6724
Invoke operating system application by object type
6734
Using API?
Yes
No
6736
Prepare launch command string
6738
Launch application with command string
6740
Prepare API parameter(s)
6742
Call API to launch application

Fig. 67A

Operand →	PM	Preferred embodiment Find processing
201	C	Finding an auto-dial # launches a system (e.g. MS) phone number log interface with the auto-dial # parameter for searching. Preferably, both the outgoing and incoming logs are searched. The auto-dial # parameter can be a wildcard (pattern) for matching. In one embodiment, all matching occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place. In another embodiment, the most recent occurrence from a particular log is presented, and perhaps in an interface which enables calling the # with a minimal user action. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search. Preferably, the find cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. A new parameter can be specified for which log to search.
203	S	Finding a weblink launches a search to system (e.g. MS) browser history with the weblink parameter (and with the params parameter if specified) for searching. The weblink parameter can be a wildcard (pattern), and may include URL parameters, for matching. In one embodiment, all matching occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked. In another embodiment, the most recent occurrence from a particular invocation is presented, and perhaps in an interface which enables invoking (transposing to) the weblink with a minimal user action. In a preferred embodiment, the params parameter tells find processing how and where to search. The search takes place as though the user manually launched the search, entered the weblink for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. A new parameter can be specified for which folder to search.

Fig. 67B-1

Operand →	PM	Preferred embodiment Find processing
205	C	Finding an email causes searching a system (e.g. MS) email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by seaching for substrings. All occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-2

Operand ↓	PM	Preferred embodiment Find processing
207	C	Finding an sms message causes searching a system (e.g. MS) sms messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-3

Operand →	PM	Preferred embodiment Find processing
209	C	Finding a broadcast email causes searching a system (e.g. MS) email system with search criteria of the email param string. The email param string can specify searching any email fields for any values including wildcarding. Each field is referenced with a predefined name and associated with a search criteria. For example, the param of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'" searches emails with a subject containing "personnel" and sent to "george@alltell.com" and has an email body containing "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders). Those skilled in the art recognize useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other info, of the email and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS email system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-4

Operand	PM	Preferred embodiment Find processing
211	C	Finding a broadcast sms message causes searching a system (e.g. MS) sms messaging system with search criteria of the sms message param string. The message param string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com'," causes searching messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders). Those skilled in the art recognize useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place. In another embodiment, the most recent occurrence from searched folders is presented, and perhaps in an interface which enables appropriate MS messaging system processing from that point forward (e.g. when processed at local MS). Alternatively, an additional parameter indicates how to search. The search takes place as though the user manually launched the search, entered search criteria, and then was presented with result(s). Appropriate MS storage is updated and processed as though the user manually performed the search. Preferably, find cmd data is maintained to LBX History, a historical log, or other useful storage for later use.
213	O	Finding an indicator searches appropriate system (e.g. MS) storage for the indicator (e.g. storage/memory used for indicators by other commands). The indicator parameter string specifies the indicator (e.g. string) being sought and wildcarding is supported. Any active user interface object containing the indicator is surfaced. If more than one user interface object contains the indicator, then all objects are appropriately tiled with the most recent in the priority position(s). In another embodiment, appropriate MS storage/memory which contains the history of indicators sent is searched and all occurrences found in history are presented with at least their date/time stamps, the indicator, and perhaps other information. In yet another embodiment, a new parameter tells processing whether to surface/prioritize active objects, or to search history for when indicators were sent. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-5

Operand ↓	PM	Preferred embodiment Find processing
215	C	Finding an application causes searching the system (e.g. MS) for the application (and with the params parameter if specified). The app parameter is preferably an executable name, optionally with parameters that were passed. Providing a partial or full path to the application parameter will validate that it is found there. The app parameter string preferably supports wildcarding. Embodiment (or as specified with params and/or new parameters) include file system searching, invocation history (e.g. Microsoft Windows up/down arrow command line recall) searching for what had been invoked (perhaps within a trailing time period), what is currently running, what has been terminated (perhaps within a trailing time period), or any of these for a particular invoked identity, credentials, or owner. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. In another embodiment, all parts which are linked to the executable are identified with their paths, date/time stamps, size, and perhaps attributes when a symbol file is specified with a new parameter. The symbol file is output from a link process and can be used to identify all executable parts such as dynamic link libraries, linked binaries, and any other executable binary file involved with the application. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and processed as though the user had manually performed the search. Preferably, find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
217	S	Finding a document causes searching the system (e.g. MS) for the document. The doc parameter is a document name. The document parameter can be a wildcard (pattern) for matching. Providing a partial or full path to the document name will validate that it is found there. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
219	S	Finding a file causes searching the system (e.g. MS) for the file. The path parameter is a file name. Providing a partial or full path to the file will validate that it is found there. The path parameter can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-6

Operand ↓	PM	Preferred embodiment Find processing
221	O	Finding content causes searching the system (e.g. MS) for the content. The content parameter can be a wildcard (pattern) for matching. The content parameter can be a handle to the content, or a search criteria for the content. In the preferred embodiment, all occurrences found on the MS and where the content is located is presented to the user, perhaps with other content description information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). There are various embodiments for how and where content is maintained. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
223	C	Finding a DB object causes searching the system (e.g. MS) database(s) for the database object. The database object parameter is provided with a variety of syntaxes depending on the type of database object sought. For example, the DB-obj parameters is "T:tablename" to seek a table, "S:schemaname" to seek a particular schema, "C:columnname" to seek a particular column name, "D:DBname" to seek a particular DB name, "R:rolename" to seek a particular role set, "P:procname" to search for particular stored procedure, etc. There are unique syntaxes for every type of DB object being sought. Those skilled in the art know how to query system tables for particular DB object(s) sought. An appropriate SQL client API should be used. If necessary, an additional parameter is specified for authentication credentials (may be specified with DB-obj string syntax). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and information about the occurrence is presented to the user. The search takes place as though the user manually launched the search, entered the criteria or query for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-7

Operand →	PM	Preferred embodiment Find processing
225	O	Finding data causes searching the system (e.g. MS) for the data. In the preferred embodiment, the data is a global system variable visible to processes of a MS O/S. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Depending on the embodiment, data may be that which is contained in a program data segment, stack segment, and/or extra segment. There can be unique syntaxes for specifying which type of data is being sought (e.g. "S:dataname"). The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, find command data is maintained to LBX History, a historical log, or other useful storage for later use.
227	O	Finding a semaphore causes reading the current value of the semaphore at the system (e.g. MS) where the find command action is being executed and then presenting the current value along with any other useful information for the semaphore. The semaphore param can be a wildcard (pattern) for matching. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
229	S	Finding a directory causes searching the system (e.g. MS) for the directory (path). The path parameter is a directory name. Providing a more defined partial or full path to the path parameter will narrow down the results if the directory exists in more than one place. The path parameter can be a wildcard (pattern) for matching. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with at least their date/time stamps, size, and perhaps attributes information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-8

Operand ↓	PM	Preferred embodiment Find processing
231	C	Finding an application context causes invocation of the application at the system (e.g. MS) and then executing a macro within the application context (similar to Compose app object processing).. The app parameter is preferably a fully qualified path name to the executable to start. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The search criteria can be a wildcard (pattern) for matching. The macro parameter is preferably a file, or path, or accessible variable name containing a set of keystrokes that can be directed to standard/user-interface input. In another embodiment, the macro parameter is a prerecorded user input scenario (for play after application launched -- pulldown selections, mouse droppings, clicks, etc) captured to a file or stored in an accessible variable name. Preferably, the compose command data is maintained to LBX History, a log, or other useful storage for subsequent use.
233	S	Finding a user interface object causes finding and focusing the user interface object at the system (e.g. MS) which contains the object text (objtxt) parameter. In a preferred embodiment, there is a unique syntax for which places of user interface objects that are currently active are to be search (e.g. title bar, entry fields, radio button options, window text, combinations thereof, etc). The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
235	O	Finding user interface control causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the keystroke macro to standard input of the MS as if it were entered by the MS user. The search criteria can be a wildcard (pattern) for matching. The macro parameter is the same as was used by the command and is to be matched. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the macro at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.

Fig. 67B-9

Operand →	PM	Preferred embodiment Find processing
237	O	Finding input causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the input to the iodev device of the MS. The iodev and input parameters are the same as was used by a previous command and is to be matched. The search criteria can be a wildcard (pattern) for matching. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the iodev and input at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
239	O	Finding output causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to redirect the output to the iodev device of the MS. The search criteria can be a wildcard (pattern) for matching. The iodev and output parameters are the same as was used by a previous command and is to be matched. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which used the iodev and output at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
241	S	Finding an alert causes searching the system (e.g. MS) for the alert. The alert parameter is the same parameter used to generate an alert (e.g. using another command). In the preferred embodiment, all occurrences found on the MS which is associated to the alerter application in use at the MS, and which is used for other commands disclosed, are presented to the user with at least their date/time stamps, and perhaps other information. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). The search criteria can be a wildcard (pattern) for matching. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-10

Operand →	PM	Preferred embodiment Find processing
243	O	Finding a process causes finding all process names running at the system (e.g. MS) which contain the prname string parameter (e.g. in UNIX: "ps -ef | grep prname"). In the preferred embodiment, all occurrences found running at the MS are presented with interesting programmatic information such as when started, its size, etc (see UNIX ps command for other information that can be presented here in various embodiments; an additional parameter (like ps parameters) can specify what info to provide to the user). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.
245	S	Finding a container causes searching the system (e.g. MS) for the container. The container parameter is a container name (e.g. file system directory) depending on the MS or environment. Unique syntaxes can be used for which type of container is being searched. In the preferred embodiment, all occurrences found on the MS and their paths are presented to the user with information of interest. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, historical log, or other useful storage for subsequent use.
247	O	Finding a program object causes searching the system (e.g. MS) for the program object. In the preferred embodiment, a unique syntax is used for which type of program object is being sought (e.g. Q:queuename has a queue qualifier prefix). There can be unique syntaxes for specifying which type of program object is being sought (e.g. "S:dataname"). In the preferred embodiment, all occurrences found on the MS and information about the occurrence including its current value is presented to the user. A null data parameter returns all occurrences found. A non-null data parameter returns these objects having the data value. Objects must be programmatically accessible. In one embodiment, a well known location of link symbol information files are consulted, and in another embodiment a new parameter specifies where to look, or which symbol file of information to use. In another embodiment, MS storage and/or memory is searched which recorded a previous atomic command action, and the search takes place for a previous command(s) (e.g. Notify) for when performed and what was performed. The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-11

Operand →	PM	Preferred embodiment Find processing
249	O	Finding a cursor causes searching the system (e.g. MS) storage and/or memory which was used for processing another command (e.g. Compose) to view or alter a cursor. In the preferred embodiment, presented in the search results are all occurrences of previous command actions, which viewed or altered the cursor at the MS, including the command, date/time stamp, and other information recorded (e.g. to LBX History, a historical log, or other useful storage for subsequent use). The search criteria can be a wildcard (pattern) for matching. The search takes place as though the user manually launched a search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
251	C	Finding a calendar object causes searching a system (e.g. MS) calendar system with search criteria of the calendar object parameter string. The calendar object parameter string can specify searching any calendar entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize useful syntaxes for searching any characteristics of calendar objects with an appropriate syntax. The calendar object parameter is at least a string with a syntax for querying any combination of calendar object fields. In the preferred embodiment, all occurrences found in history are presented with at least their date/time stamps, attendees, and perhaps other information, of the calendar object and when it was scheduled. In another embodiment, the most recent occurrence from the calendaring system is presented, and perhaps in an interface which enables appropriate MS calendar system processing from that point forward (e.g. when processed at local MS), or alternatively an additional parameter can specify how to search. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.

Fig. 67B-12

Operand →	PM	Preferred embodiment Find processing
253	C	Finding an address book (AB) object causes searching a system (e.g. MS) AB system with search criteria of the AB object parameter string. The AB object parameter string can specify searching any AB entry fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria (similar to email above). Those skilled in the art recognize many useful syntaxes for searching any characteristics of AB objects/entries with an appropriate syntax. The AB object parameter is at least a string with a syntax for querying any combination of AB object fields. In the preferred embodiment, all occurrences found are presented with appropriate AB information. In another embodiment, the most recent occurrence from the AB system is presented, and perhaps in an interface which enables appropriate MS AB system processing from that point forward (e.g. when processed at local MS), or alternatively an additional parameter can specify how to search. Wildcarding (pattern matching) is preferably inherent by searching for substrings. The search takes place as though the user manually launched the search, entered the criteria for the search, and then was presented with the result(s). Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search. Preferably, the find command data is maintained to LBX History, a historical log, or other useful storage for subsequent use.
...		

Fig. 67B-13

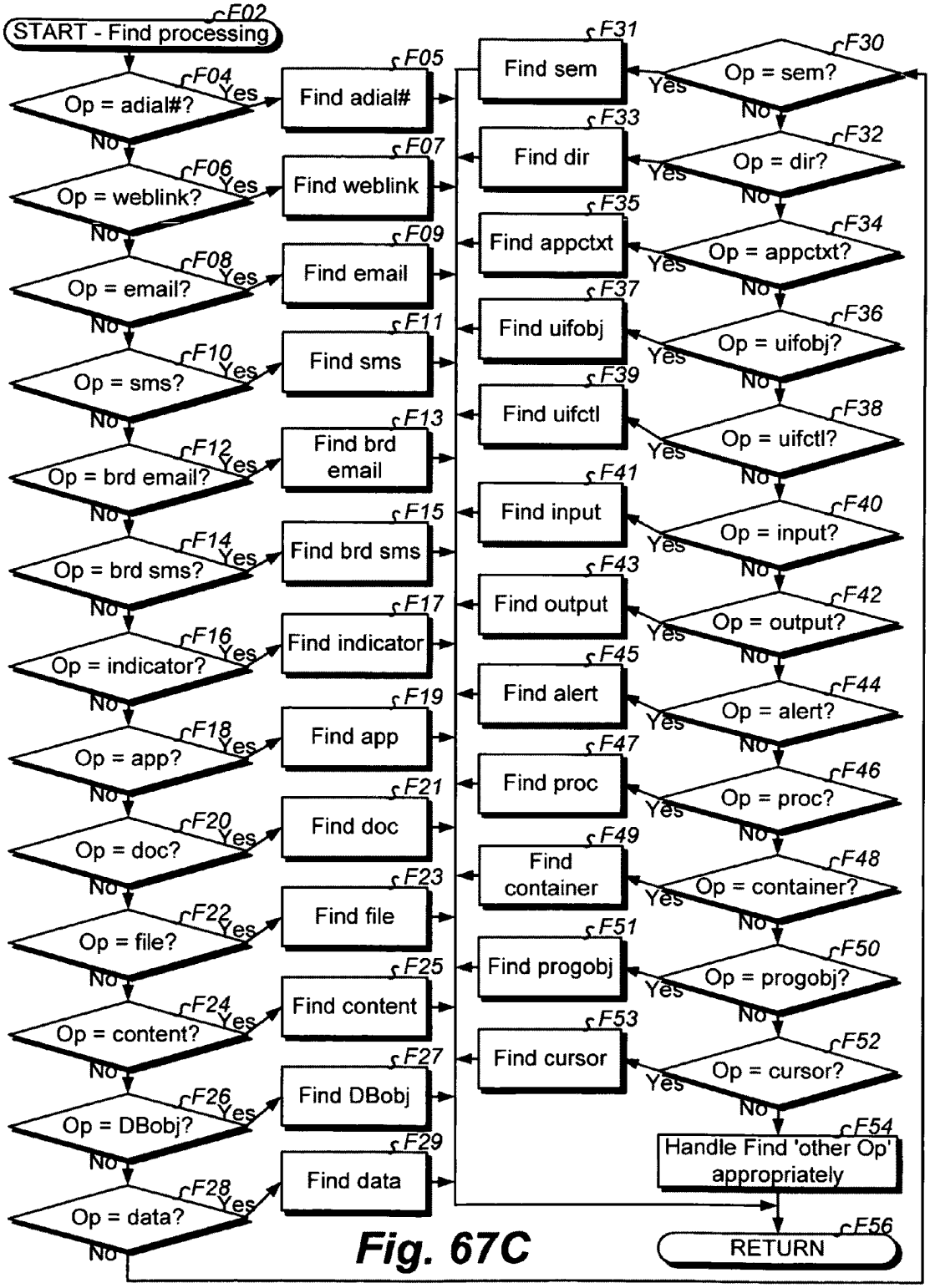

F02
START - Find processing
F04
Op = adial#?
Yes
No
F05
Find adial#
F06
Op = weblink?
Yes
No
F07
Find weblink
F08
Op = email?
Yes
No
F09
Find email
F10
Op = sms?
Yes
No
F11
Find sms
F12
Op = brd email?
Yes
No
F13
Find brd email
F14
Op = brd sms?
Yes
No
F15
Find brd sms
F16
Op = indicator?
Yes
No
F17
Find indicator
F18
Op = app?
Yes
No
F19
Find app
F20
Op = doc?
Yes
No
F21
Find doc
F22
Op = file?
Yes
No
F23
Find file
F24
Op = content?
Yes
No
F25
Find content
F26
Op = DBobj?
Yes
No
F27
Find DBobj
F28
Op = data?
Yes
No
F29
Find data
F31
Find sem
F33
Find dir
F35
Find appctxt
F37
Find uifobj
F39
Find uifctl
F41
Find input
F43
Find output
F45
Find alert
F47
Find proc
F49
Find container
F51
Find progobj
F53
Find cursor
F30
Op = sem?
Yes
No
F32
Op = dir?
Yes
No
F34
Op = appctxt?
Yes
No
F36
Op = uifobj?
Yes
No
F38
Op = uifctl?
Yes
No
F40
Op = input?
Yes
No
F42
Op = output?
Yes
No
F44
Op = alert?
Yes
No
F46
Op = proc?
Yes
No
F48
Op = container?
Yes
No
F50
Op = progobj?
Yes
No
F52
Op = cursor?
Yes
No
F54
Handle Find 'other Op' appropriately
F56
RETURN

Fig. 67C

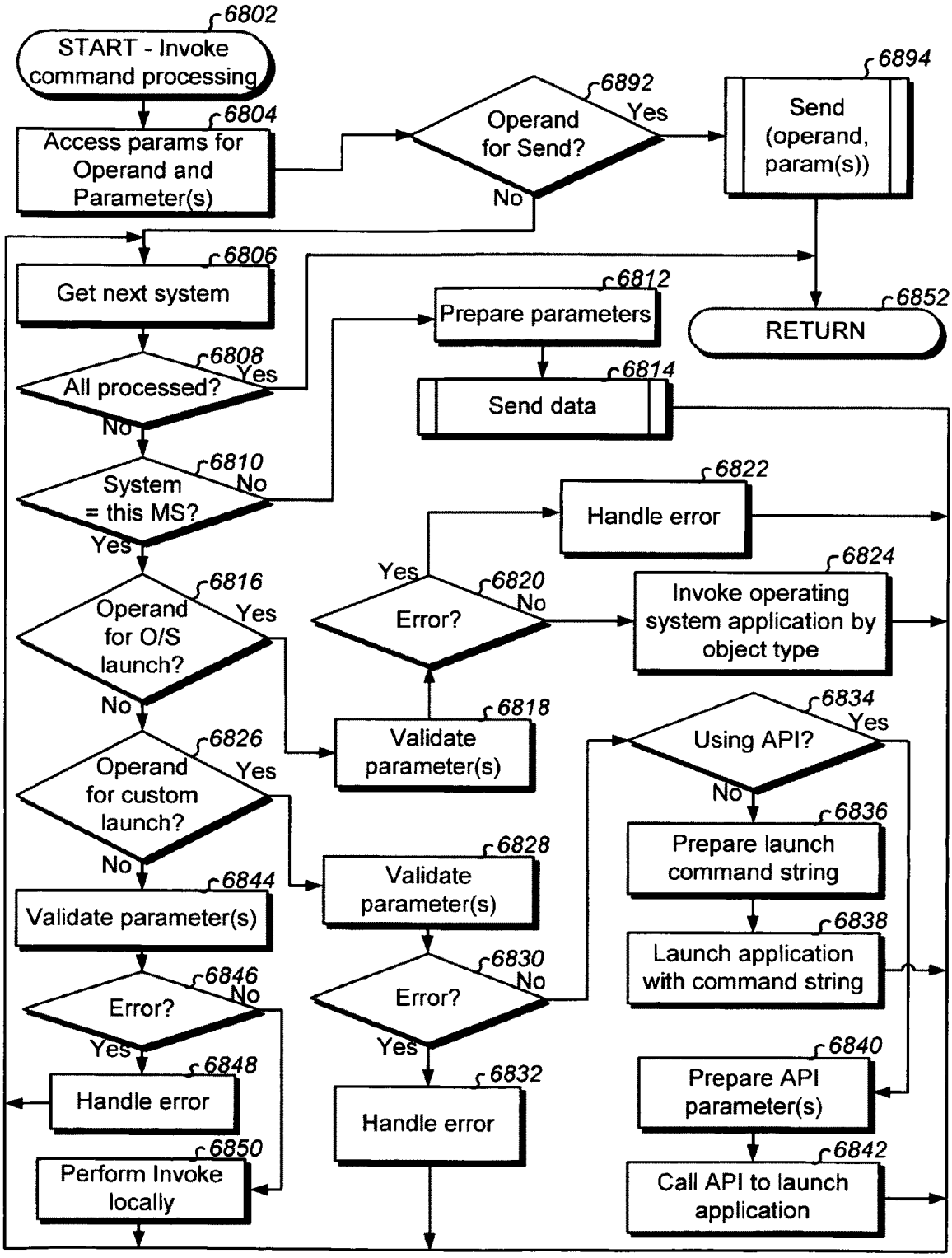

6802
START - Invoke command processing
6804
Access params for Operand and Parameter(s)
6892
Operand for Send?
Yes
No
6894
Send (operand, param(s))
6806
Get next system
6812
Prepare parameters
6852
RETURN
6808
All processed?
Yes
No
6814
Send data
6810
System = this MS?
No
Yes
6822
Handle error
6816
Operand for O/S launch?
Yes
No
Yes
6820
Error?
No
6824
Invoke operating system application by object type
6818
Validate parameter(s)
6834
Using API?
Yes
No
6826
Operand for custom launch?
Yes
No
6828
Validate parameter(s)
6836
Prepare launch command string
6844
Validate parameter(s)
6830
Error?
No
Yes
6838
Launch application with command string
6846
Error?
No
Yes
6840
Prepare API parameter(s)
6848
Handle error
6832
Handle error
6850
Perform Invoke locally
6842
Call API to launch application

Fig. 68A

Operand ↓	PM	Preferred embodiment Invoke processing
201	C	Invoking with an auto-dial # launches the MS phone number calling interface at the MS (local or remote) with the auto-dial # parameter for placing a call (like/see Notify / Connect autodial # processing). The call is actually made as though a user manually launched the dialing application at the particular MS, entered the auto-dial # and then chose to make the call with it. Appropriate MS storage is updated and subsequently processed as though the user had entered the # for calling manually, and then make the call. Conventional call processing takes place thereafter. Preferably, the invoke command data is maintained to LBX History, a historical call log (e.g. outgoing), or other useful storage for subsequent use. A system parameter provides means for placing the call from another system (e.g. another MS) – like a Host specification.
203	S	Invoking with a web link launches the MS browser at the particular MS and invokes (transposes to) the link as though the user had entered it manually and went to the weblink page (like/see Notify / Connect weblink processing). In one embodiment, the weblink parameter includes URL parameter(s). In another embodiment, the params parameter supports a URL command string for appending to the weblink (e.g. "?v=yes&T=go") for customized web page processing. An alternate embodiment can fire form variables for active web page processing using the params parameter, or URL parameter(s). Processing takes place as though the user manually launched the browser application at the particular MS, entered the weblink and then loaded the weblink webpage. Appropriate MS storage is updated and subsequently processed as though the user had entered and invoked the weblink in the browser manually. Preferably, the compose command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
205	E	See Send Command for identical processing.
207	E	See Send Command for identical processing.
209	E	See Send Command for identical processing.
211	E	See Send Command for identical processing.

Fig. 68B-1

Operand →	PM	Preferred embodiment invoke processing
213	O	Invoking an indicator updates the appropriate MS storage so that the currently focused user interface object (e.g. window titlebar) of the particular MS user interface is modified with the indicator (like/see Send indicator processing).. If there are no active user interface objects in the MS user interface, then an appropriate alert area of the currently focused interface is to display the indicator. The user can clear (remove) the indicator when desired. Preferably, the indicator is used for modifying other focused objects (e.g. titlebars) or other focused areas in the user interface so as to not get overlooked. For example, as the user navigates and surfaces/focuses new user interface objects, the indicator remains visible on the newly focused object. Preferably, the indicator is selectable by the user of the MS for showing all other send command parameters associated, as well as a date/time stamp of when sent. In other embodiments, the most recently displayed indicator is displayed in the appropriate focused area, but the user can conveniently select any indicators which were sent in history at some point in time for sought indicator information by selecting the currently displayed indicator and then requesting to browse/scroll history of previously delivered indicators (with options to see details). Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
215	C	Invoking an application causes invocation of the application at the particular MS (like/see Send app processing).. The app parameter is preferably a fully qualified path name to the executable to start, and may already include parameters. In another embodiment, the app parameter is indirect: a path name to a "shortcut" (like a MS Windows shortcut). In another embodiment, the app parameter is an identifier string for the underlying operating system to know which application to start. The params parameter may specify the executable parameters, or may be used for how to start the application (like attributes of Send app processing). An error is logged if the app parameter is not found for launch. Preferably, the invoke command data is maintained to LBX History, a historical log (web page load history), or other useful storage for subsequent use.
217	S	Invoking a document causes invocation of an appropriate application at the particular MS in accordance with the object type as though the user selected the document for automatically being associated to the correct application when opening the document. The user can then decide what to do with the document once it is opened in the appropriate application. In an alternate embodiment, an additional parameter is provided for exactly what to do with the document, in which case an appropriate API is invoked with the document (i.e. PM = C). The doc parameter is preferably a fully qualified path name to the document. Preferably, the invoke command data is maintained to LBX History, a log, or other useful storage for subsequent use.

Fig. 68B-2

Operand ↓	PM	Preferred embodiment Invoke processing
219	S	Invoking a file causes invocation of an appropriate application at the particular MS in accordance with the file type as though the user selected the file for automatically being associated to the correct application when opening the document. Processing takes place as though the user manually launched the application for the specified file. The user can then decide what to do with the file once it is opened in the appropriate application. In an alternate embodiment, an additional parameter is provided for exactly what to do with the file, in which case an appropriate API is invoked with the file (i.e. PM = C). The path parameter is preferably a fully qualified path name to the file. Preferably, the invoke command data is maintained to LBX History, a log, or other useful storage for subsequent use.
221	O	Invoking content causes the content to be presented at the particular MS in a manner which is appropriate for the content type (like/see Notify content processing).. The content parameter is one that cannot be classified in the other operands, but is content for presentation nevertheless. Examples include special data records (e.g. extern variable name), content data memory locations (e.g. programmatic variable), or files containing a customizably processed format. Methods of displaying the content include audio and/or visual using applicable MS capabilities. Preferably, the invoke command data is maintained to LBX History, a historical content log (e.g. incoming), browser history data, or other useful storage for subsequent user browse of the accompanying content and a date/time stamp of when sent, and for presentation of the content. Various embodiments will save to LBX History how many times, and when, the content was presented.
223	O	Invoking a Database (DB) object causes the DB object (i.e. qualified database with access query string) to be modified with the query parameter at the particular MS (like/see Notify DB-obj processing without certain parameters). The query parameter is used to perform any query against the specified DB-database (DB-obj), preferably a query that only returns a return code. Preferably, the invoke command data is maintained to LBX History, a database log (e.g. incoming), or other useful storage for subsequent query use and a date/time stamp of when sent, and for DB query manager browse/use of the query in response to an applicable user action.
225	O	Invoking data causes modifying the value of the data at the particular MS (i.e. set data to value – like/see Notify data processing without certain parameters). An error can result if the data is not resolvable for the attempt. In the preferred embodiment, the data is a global system variable visible to all processes of a MS operating system. In other embodiments, the data may have limited scope which is made accessible to present disclosure processing (e.g. with extern). Preferably, the data affected is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, of the data variable name, its before and after values and date/time stamp of when sent, and for presentation of the data value in response to a user action to show it.

Fig. 68B-3

Operand	PM	Preferred embodiment Invoke processing
227	O	Invoking a semaphore causes modifying the value of the semaphore at the particular MS where the action is being executed (like/see Notify semaphore processing).. In the preferred embodiment, the semaphore is a global system semaphore visible to all processes of a MS operating system. In other embodiments, the semaphore may have limited scope which is made accessible to present disclosure processing (e.g. RAM semaphore). Preferably, the semaphore value before and after setting is maintained to LBX History, a historical log (e.g. incoming), or other useful storage for subsequent user browse, or programmatic access, and for presentation of the semaphore information in response to a user action to show it.
229	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
231	C	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
233	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter). Also, the cmds parameter replaces the "capture focused object" command string with one or more semicolon delimited "capture focused object" command strings for each target system in the system(s) parameters. This enables a plurality of different types of MSs to participate even though they have different commands (e.g. keystroke capture actions) to accomplish capturing the focused user interface object. Based on the file type at the particular MS, the appropriate application opens the file.
235	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
237	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
239	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
241	C	See Connect Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
243	O	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
245	S	See Compose Command for identical processing, except processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
247	O	See Notify Command for identical processing, except sender is forced to requesting MS, no documentary subj/msg parameter, and system(s) used instead of recipient(s). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).

Fig. 68B-4

Operand →	PM	Preferred embodiment Invoke processing
249	O	See Notify Command for identical processing, except sender is forced to requesting MS, no documentary subj/msg parameter, and system(s) used instead of recipient(s). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
251	O	See Send Command for identical processing, except sender is forced to requesting MS, and system(s) used instead of recipient(s) for calendar alteration without regard for the owner (this embodiment). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
253	O	See Send Command for identical processing, except sender is forced to requesting MS, and system(s) used instead of recipient(s) for AB alteration without regard for the owner (this embodiment). Processing may take place locally and/or at privilege-providing remote MS(s) (system(s) parameter).
. . .		

Fig. 68B-5

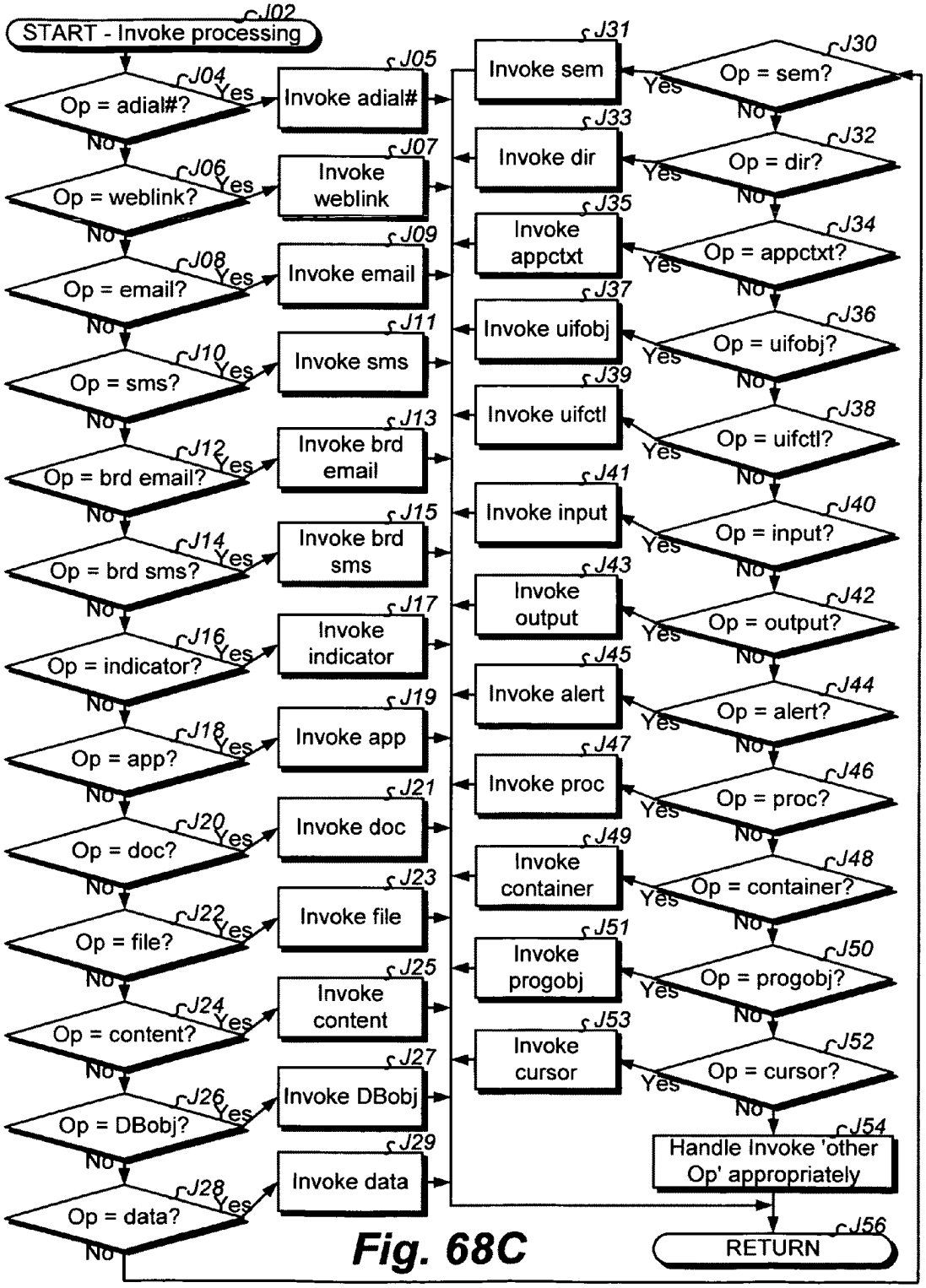

J02
START - Invoke processing
J04
Op = adial#?
Yes
J05
Invoke adial#
No
J06
Op = weblink?
Yes
J07
Invoke weblink
No
J08
Op = email?
Yes
J09
Invoke email
No
J10
Op = sms?
Yes
J11
Invoke sms
No
J12
Op = brd email?
Yes
J13
Invoke brd email
No
J14
Op = brd sms?
Yes
J15
Invoke brd sms
No
J16
Op = indicator?
Yes
J17
Invoke indicator
No
J18
Op = app?
Yes
J19
Invoke app
No
J20
Op = doc?
Yes
J21
Invoke doc
No
J22
Op = file?
Yes
J23
Invoke file
No
J24
Op = content?
Yes
J25
Invoke content
No
J26
Op = DBobj?
Yes
J27
Invoke DBobj
No
J28
Op = data?
Yes
J29
Invoke data
No
J31
Invoke sem
J33
Invoke dir
J35
Invoke appctxt
J37
Invoke uifobj
J39
Invoke uifctl
J41
Invoke input
J43
Invoke output
J45
Invoke alert
J47
Invoke proc
J49
Invoke container
J51
Invoke progobj
J53
Invoke cursor
J30
Op = sem?
Yes
No
J32
Op = dir?
Yes
No
J34
Op = appctxt?
Yes
No
J36
Op = uifobj?
Yes
No
J38
Op = uifctl?
Yes
No
J40
Op = input?
Yes
No
J42
Op = output?
Yes
No
J44
Op = alert?
Yes
No
J46
Op = proc?
Yes
No
J48
Op = container?
Yes
No
J50
Op = progobj?
Yes
No
J52
Op = cursor?
Yes
No
J54
Handle Invoke 'other Op' appropriately
J56
RETURN

Fig. 68C

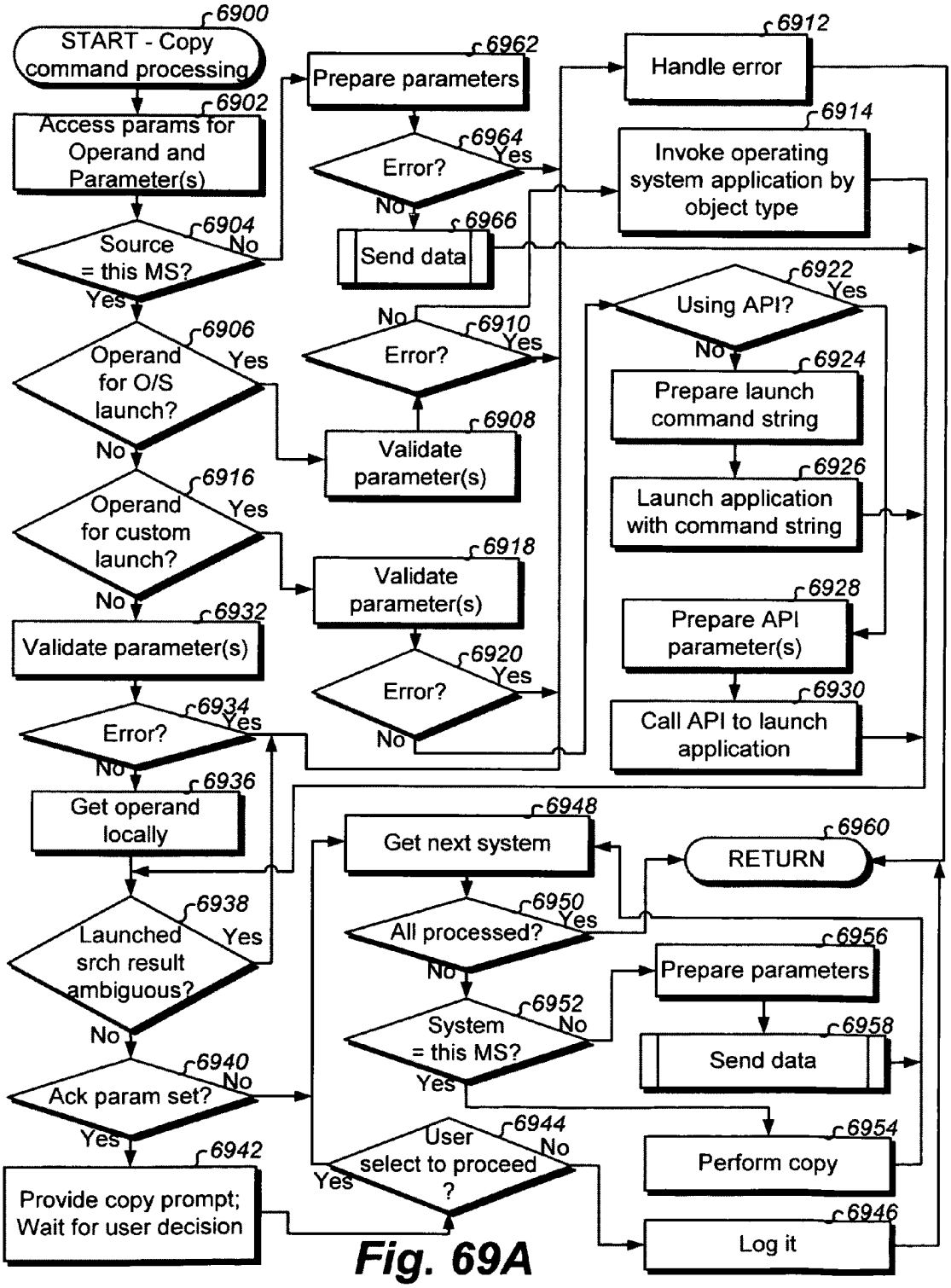

6900
START - Copy command processing
6902
Access params for Operand and Parameter(s)
6904
Source = this MS?
No
Yes
6906
Operand for O/S launch?
Yes
No
6908
Validate parameter(s)
6910
Error?
Yes
No
6912
Handle error
6914
Invoke operating system application by object type
6916
Operand for custom launch?
Yes
No
6918
Validate parameter(s)
6920
Error?
Yes
No
6922
Using API?
Yes
No
6924
Prepare launch command string
6926
Launch application with command string
6928
Prepare API parameter(s)
6930
Call API to launch application
6932
Validate parameter(s)
6934
Error?
Yes
No
6936
Get operand locally
6938
Launched srch result ambiguous?
Yes
No
6940
Ack param set?
No
Yes
6942
Provide copy prompt; Wait for user decision
6944
User select to proceed ?
No
Yes
6946
Log it
6948
Get next system
6950
All processed?
Yes
No
6952
System = this MS?
No
Yes
6954
Perform copy
6956
Prepare parameters
6958
Send data
6960
RETURN
6962
Prepare parameters
6964
Error?
Yes
No
6966
Send data

Fig. 69A

Operand →	PM	Preferred embodiment Copy processing
201	C	Copying an auto-dial # launches a phone number log interface with the auto-dial # parameter (can be wildcarded) for searching the source system. Preferably, both the outgoing and incoming logs are searched. In an alternate embodiment, the log is specified with a parameter. In the preferred embodiment, the most recent occurrence from a particular log is provided. In another embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the call and when it took place (e.g. when the ack parameter is set) and the user browses the results prior to accepting the copy of multiple items. The search takes place as though the user manually launched the search, entered the auto-dial # for the search, and then was provided with result(s) for the copy. Preferably, the copy shall take place if there are no ambiguities (e.g. more than one phone number returned per search criteria). An additional parameter may be specified for the target (different log) of the copy, otherwise the object is copied to an assumed location (e.g. same folder to more recent position). Appropriate MS storage is updated and subsequently processed as though the user manually performed the search and copy of the result. Preferably, the copy cmd data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to system(s) logs, preferably with identifying information of the source and who did the copy.
203	C	Copying a weblink launches a search to MS browser history with the weblink parameter (can be wildcarded) for searching the source system. In one embodiment, all occurrences found in history are presented with their date/time stamps, and perhaps other information, of the link and when it was invoked (e.g. when the ack parameter is specified to true) for presentation to the user prior to doing the copy. In the preferred embodiment, the most recent occurrence from a particular invocation is provided for the copy. An additional parameter may be specified for the target (specified favorites folder), otherwise the object is copied to an assumed location (e.g. highest level favorites folder or special named folder). The search takes place as though the user manually launched the search, entered the weblink for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy of the result. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special browser favorites folder, or another designated folder configured ahead of time, preferably with identifying information of the source and who did the copy.

Fig. 69B-1

Operand →	PM	Preferred embodiment Copy processing
205	C	Copying an email causes searching the source email system with search criteria of the email parameter string. The email parameter string can specify searching any email fields for any values including wildcarding (patterns for matching). Each field is referenced with a predefined name and then associated with a search criteria. For example, the email string of "subj:'personnel';recip:'george@alltell.com';body:'reduction in force'' causes searching all emails with a subject containing "personnel" and was sent to "george@alltell.com" and has a message body containing the string "reduction in force". To search for certain email containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:sent,inbox,company;" indicates to only search the email folders of sent, inbox, and company (no specification preferably indicates to search all folders). Those skilled in the art recognize many useful syntaxes for searching any characteristics of email. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In one embodiment, all occurrences found in history are presented with at least their date/time stamps, subject line, sender and recipient, and perhaps other information, of the email and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for the copy. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. drafts, inbox, or special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for doing the copy. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special email folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter for copy processing, preferably with identifying information of the source and who did the copy (if supported in email application).

Fig. 69B-2

Operand →	PM	Preferred embodiment Copy processing
207	C	Copying an sms message causes searching the source messaging system with search criteria of the sms message parameter string. The message parameter string can specify searching any message fields for any values including wildcarding. Each field is referenced with a predefined name and then associated with a search criteria. For example, the sms message string of "recip:'2144034071@nextel.com,9725397137@lbxsrv.com';" causes searching all messages to the sought recipients. To search for certain messaging containers/folders, a sub-search criteria of "folders" is used (e.g. "folders:outgoing" indicates to only search the outgoing folder (no specification preferably indicates to search all folders). Those skilled in the art recognize many useful syntaxes for searching any characteristics of messages. Wildcarding (pattern matching) is preferably inherent by searching for substrings. In the one embodiment, all occurrences found in history are presented with at least their date/time stamps, message, sender and recipient, and perhaps other information, of the message and when it took place, when the ack parameter is set to true for user reconciliation. In a preferred embodiment, the most recent occurrence from searched folders is provided for copying. An additional parameter may be specified for the target (specified folder), otherwise the object is copied to an assumed location (e.g. inbox, drafts, special named folder). The search takes place as though the user manually launched the search, entered the criteria for the search, and then was provided with the result for copying it. Appropriate MS storage is updated and subsequently processed as though the user had manually performed the search and copy. Preferably, the copy command data is maintained to LBX History, a historical log, or other useful storage for subsequent use. The copy is made to a special messaging folder of the target system, or another designated folder configured ahead of time, or as specified with a new parameter to copy processing, preferably with identifying information of the source and who did the copy.

Fig. 69B-3