US 20070239898A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2007/0239898 A1

Friend et al. (43) Pub. Date: Oct. 11, 2007

(54) SYSTEM AND METHOD FOR FULL WIRELESS SYNCHRONIZATION OF A DATA PROCESSING APPARATUS WITH A MESSAGING SERVICE

(76) Inventors: **John Friend**, Los Altos, CA (US); **Michael Belshe**, Sunnyvale, CA (US); **Roger Collins**, Novato, CA (US); **Mike Bennett**, Brisbane, CA (US)

Correspondence Address:
BLAKELY SOKOLOFF TAYLOR & ZAFMAN
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040 (US)

(21) Appl. No.: **11/807,290**

(22) Filed: **May 24, 2007**

Related U.S. Application Data

(63) Continuation of application No. 10/109,928, filed on Mar. 29, 2002, now Pat. No. 7,243,163, which is a continuation-in-part of application No. 09/924,283, filed on Aug. 7, 2001, now Pat. No. 7,155,483.

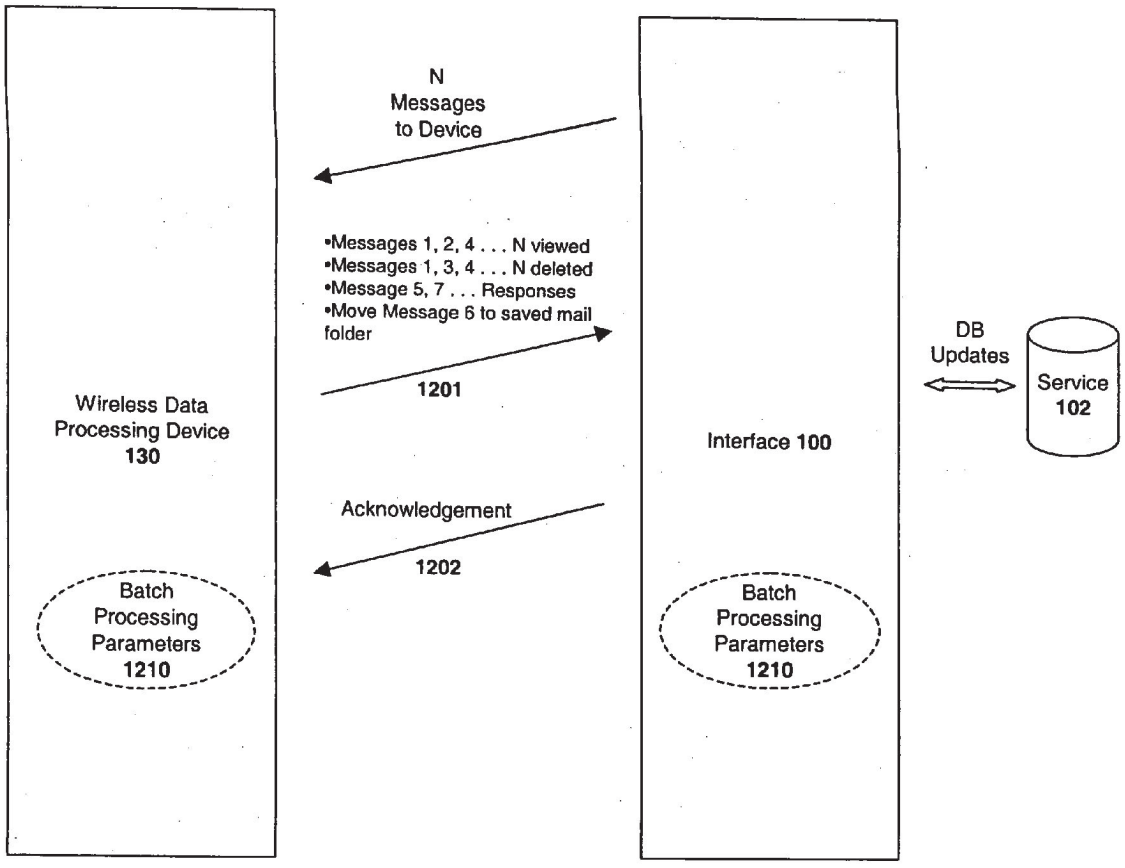
Publication Classification

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/248**

(57) **ABSTRACT**

A system is disclosed in which a data processing device is completely synchronized with a messaging service. One embodiment of the system comprises a wireless data processing device; a messaging service to maintain messages and other information on behalf of a user; and synchronization logic for maintaining synchronization of the messages and other information between the wireless device and the messaging service.



Google Exhibit 1061
Google v. Headwater

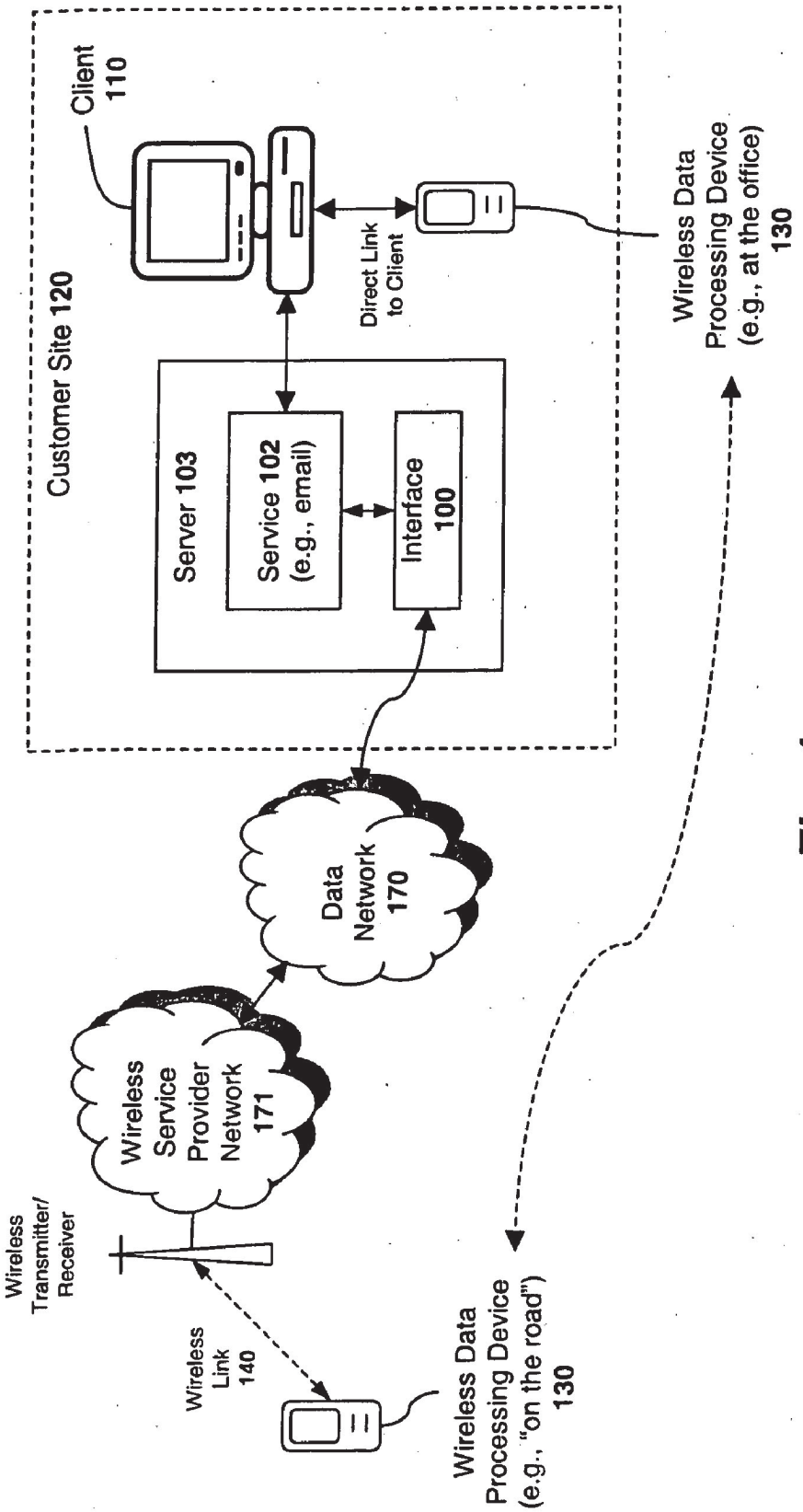

Customer Site 120
Client 110
Server 103
Service 102 (e.g., email)
Interface 100
Direct Link to Client
Wireless Data Processing Device (e.g., at the office) 130
Data Network 170
Wireless Service Provider Network 171
Wireless Transmitter/ Receiver
Wireless Link 140
Wireless Data Processing Device (e.g., "on the road") 130

Fig. 1

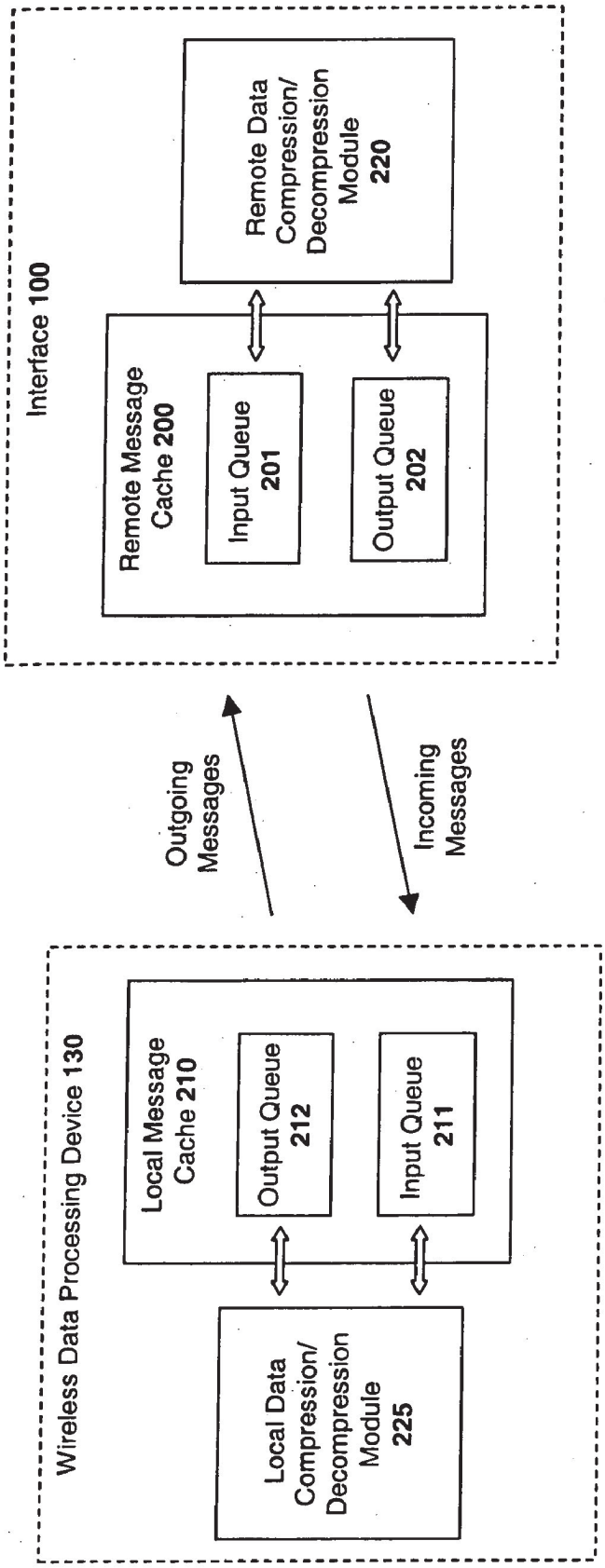

Interface 100
Remote Data Compression/ Decompression Module 220
Remote Message Cache 200
Input Queue 201
Output Queue 202
Outgoing Messages
Incoming Messages
Wireless Data Processing Device 130
Local Message Cache 210
Output Queue 212
Input Queue 211
Local Data Compression/ Decompression Module 225

Fig. 2

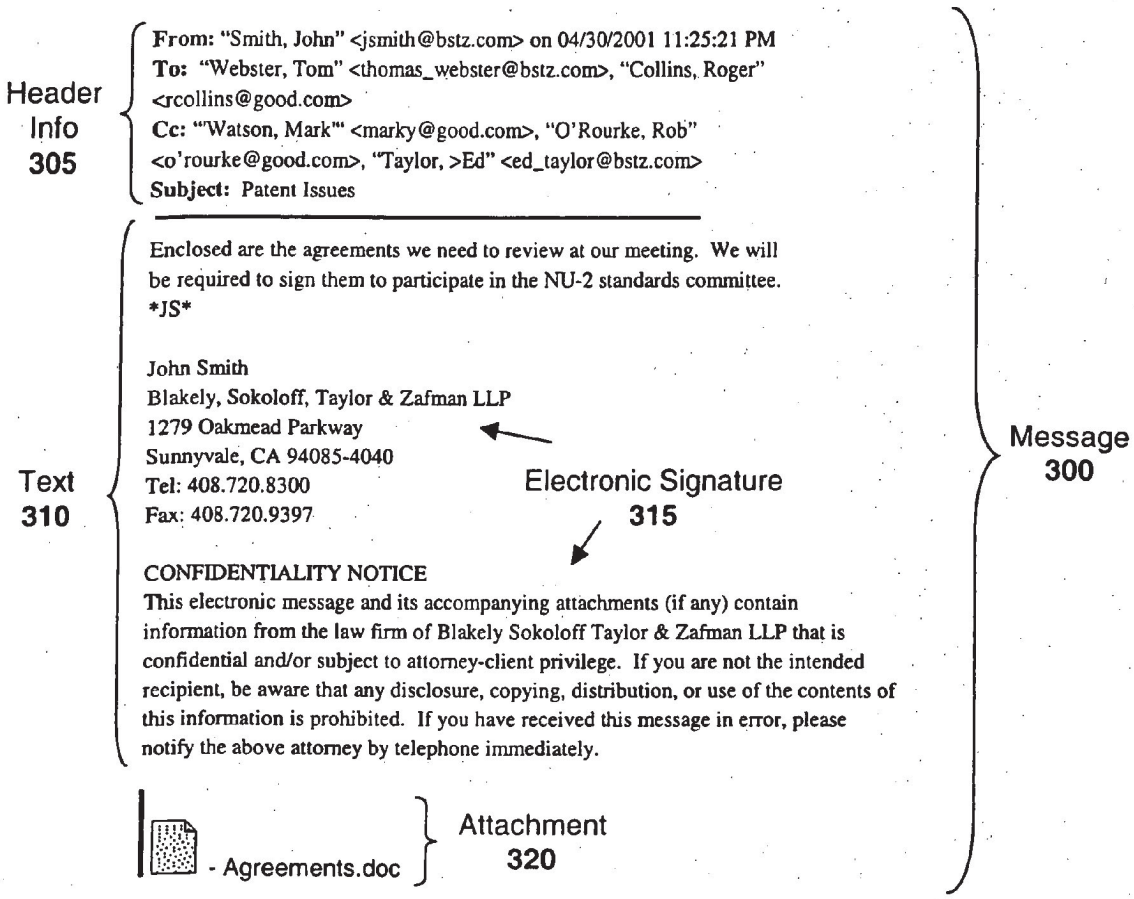

Header Info 305
From: "Smith, John" <jsmith@bstz.com> on 04/30/2001 11:25:21 PM
To: "Webster, Tom" <thomas_webster@bstz.com>, "Collins, Roger" <rcollins@good.com>
Cc: "'Watson, Mark'" <marky@good.com>, "O'Rourke, Rob" <o'rourke@good.com>, "Taylor, >Ed" <ed_taylor@bstz.com>
Subject: Patent Issues
Text 310
Enclosed are the agreements we need to review at our meeting. We will be required to sign them to participate in the NU-2 standards committee.
JS
John Smith
Blakely, Sokoloff, Taylor & Zafman LLP
1279 Oakmead Parkway
Sunnyvale, CA 94085-4040
Tel: 408.720.8300
Fax: 408.720.9397
Electronic Signature 315
CONFIDENTIALITY NOTICE
This electronic message and its accompanying attachments (if any) contain information from the law firm of Blakely Sokoloff Taylor & Zafman LLP that is confidential and/or subject to attorney-client privilege. If you are not the intended recipient, be aware that any disclosure, copying, distribution, or use of the contents of this information is prohibited. If you have received this message in error, please notify the above attorney by telephone immediately.
- Agreements.doc
Attachment 320
Message 300

FIG. 3a

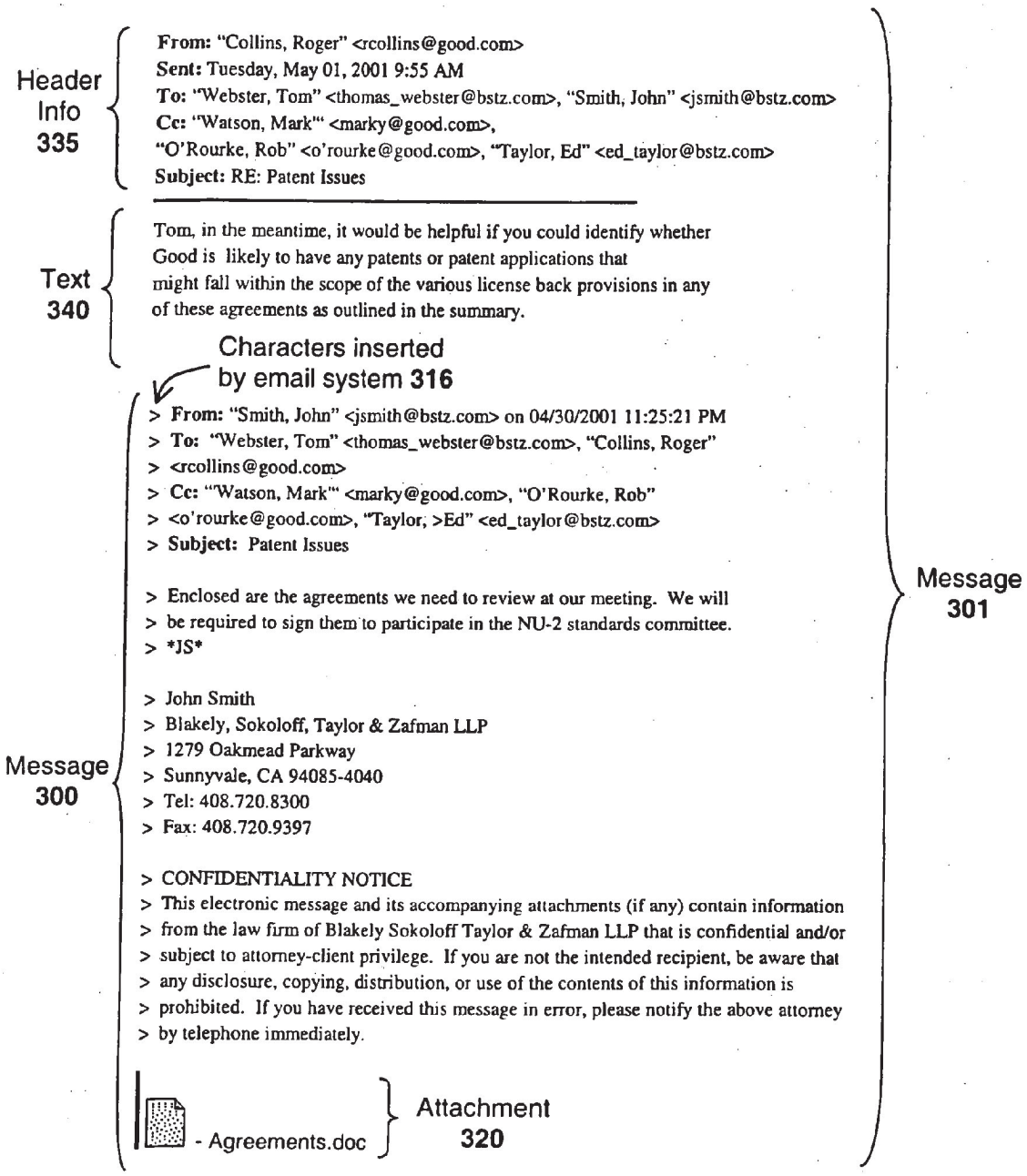

Header Info 335
From: "Collins, Roger" <rcollins@good.com>
Sent: Tuesday, May 01, 2001 9:55 AM
To: "Webster, Tom" <thomas_webster@bstz.com>, "Smith, John" <jsmith@bstz.com>
Cc: "Watson, Mark"' <marky@good.com>,
"O'Rourke, Rob" <o'rourke@good.com>, "Taylor, Ed" <ed_taylor@bstz.com>
Subject: RE: Patent Issues
Text 340
Tom, in the meantime, it would be helpful if you could identify whether Good is likely to have any patents or patent applications that might fall within the scope of the various license back provisions in any of these agreements as outlined in the summary.
Characters inserted by email system 316
Message 300
> From: "Smith, John" <jsmith@bstz.com> on 04/30/2001 11:25:21 PM
> To: "Webster, Tom" <thomas_webster@bstz.com>, "Collins, Roger"
> <rcollins@good.com>
> Cc: "Watson, Mark"' <marky@good.com>, "O'Rourke, Rob"
> <o'rourke@good.com>, "Taylor, >Ed" <ed_taylor@bstz.com>
> Subject: Patent Issues
> Enclosed are the agreements we need to review at our meeting. We will
> be required to sign them to participate in the NU-2 standards committee.
> *JS*
> John Smith
> Blakely, Sokoloff, Taylor & Zafman LLP
> 1279 Oakmead Parkway
> Sunnyvale, CA 94085-4040
> Tel: 408.720.8300
> Fax: 408.720.9397
> CONFIDENTIALITY NOTICE
> This electronic message and its accompanying attachments (if any) contain information
> from the law firm of Blakely Sokoloff Taylor & Zafman LLP that is confidential and/or
> subject to attorney-client privilege. If you are not the intended recipient, be aware that
> any disclosure, copying, distribution, or use of the contents of this information is
> prohibited. If you have received this message in error, please notify the above attorney
> by telephone immediately.
- Agreements.doc
Attachment 320
Message 301

FIG. 3b

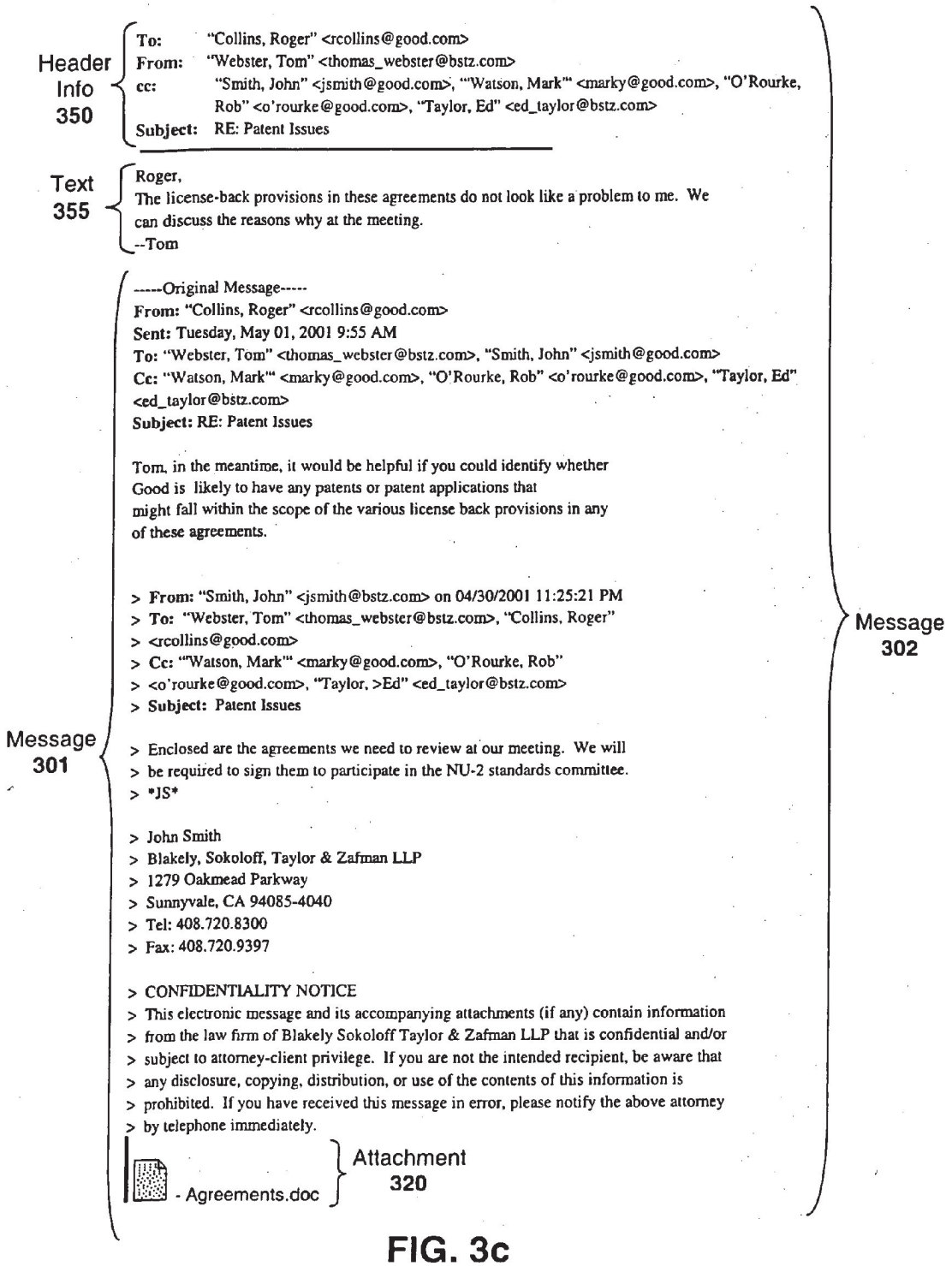

Header Info 350
To: "Collins, Roger" <rcollins@good.com>
From: "Webster, Tom" <thomas_webster@bstz.com>
cc: "Smith, John" <jsmith@good.com>, "'Watson, Mark'" <marky@good.com>, "O'Rourke, Rob" <o'rourke@good.com>, "Taylor, Ed" <ed_taylor@bstz.com>
Subject: RE: Patent Issues
Text 355
Roger,
The license-back provisions in these agreements do not look like a problem to me. We can discuss the reasons why at the meeting.
--Tom
Message 301
-----Original Message-----
From: "Collins, Roger" <rcollins@good.com>
Sent: Tuesday, May 01, 2001 9:55 AM
To: "Webster, Tom" <thomas_webster@bstz.com>, "Smith, John" <jsmith@good.com>
Cc: "Watson, Mark'" <marky@good.com>, "O'Rourke, Rob" <o'rourke@good.com>, "Taylor, Ed" <ed_taylor@bstz.com>
Subject: RE: Patent Issues
Tom, in the meantime, it would be helpful if you could identify whether Good is likely to have any patents or patent applications that might fall within the scope of the various license back provisions in any of these agreements.
> From: "Smith, John" <jsmith@bstz.com> on 04/30/2001 11:25:21 PM
> To: "Webster, Tom" <thomas_webster@bstz.com>, "Collins, Roger" <rcollins@good.com>
> Cc: "'Watson, Mark'" <marky@good.com>, "O'Rourke, Rob" <o'rourke@good.com>, "Taylor, >Ed" <ed_taylor@bstz.com>
> Subject: Patent Issues
> Enclosed are the agreements we need to review at our meeting. We will be required to sign them to participate in the NU-2 standards committee.
> *JS*
> John Smith
> Blakely, Sokoloff, Taylor & Zafman LLP
> 1279 Oakmead Parkway
> Sunnyvale, CA 94085-4040
> Tel: 408.720.8300
> Fax: 408.720.9397
> CONFIDENTIALITY NOTICE
> This electronic message and its accompanying attachments (if any) contain information from the law firm of Blakely Sokoloff Taylor & Zafman LLP that is confidential and/or subject to attorney-client privilege. If you are not the intended recipient, be aware that any disclosure, copying, distribution, or use of the contents of this information is prohibited. If you have received this message in error, please notify the above attorney by telephone immediately.
- Agreements.doc
Attachment 320
Message 302

FIG. 3c

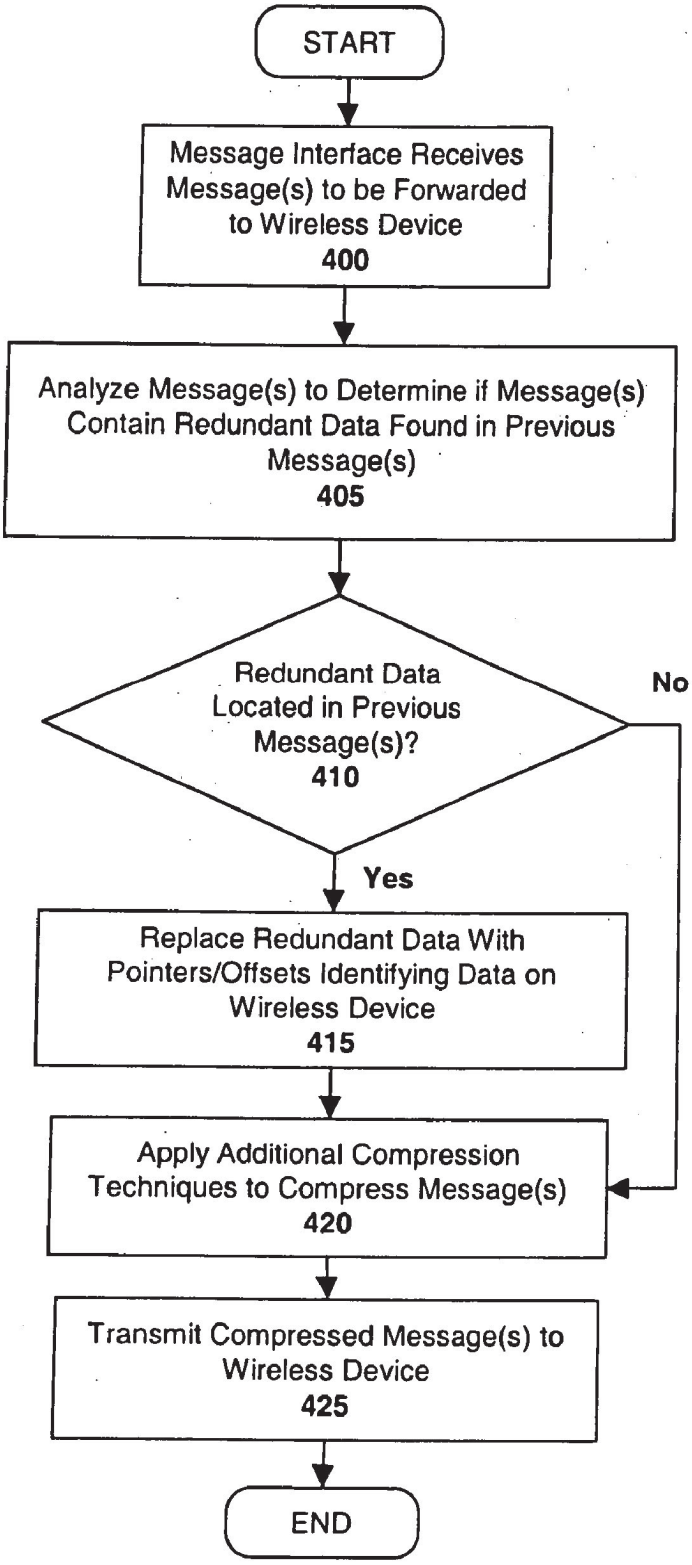

START
Message Interface Receives Message(s) to be Forwarded to Wireless Device
400
Analyze Message(s) to Determine if Message(s) Contain Redundant Data Found in Previous Message(s)
405
Redundant Data Located in Previous Message(s)?
410
No
Yes
Replace Redundant Data With Pointers/Offsets Identifying Data on Wireless Device
415
Apply Additional Compression Techniques to Compress Message(s)
420
Transmit Compressed Message(s) to Wireless Device
425
END

Fig. 4

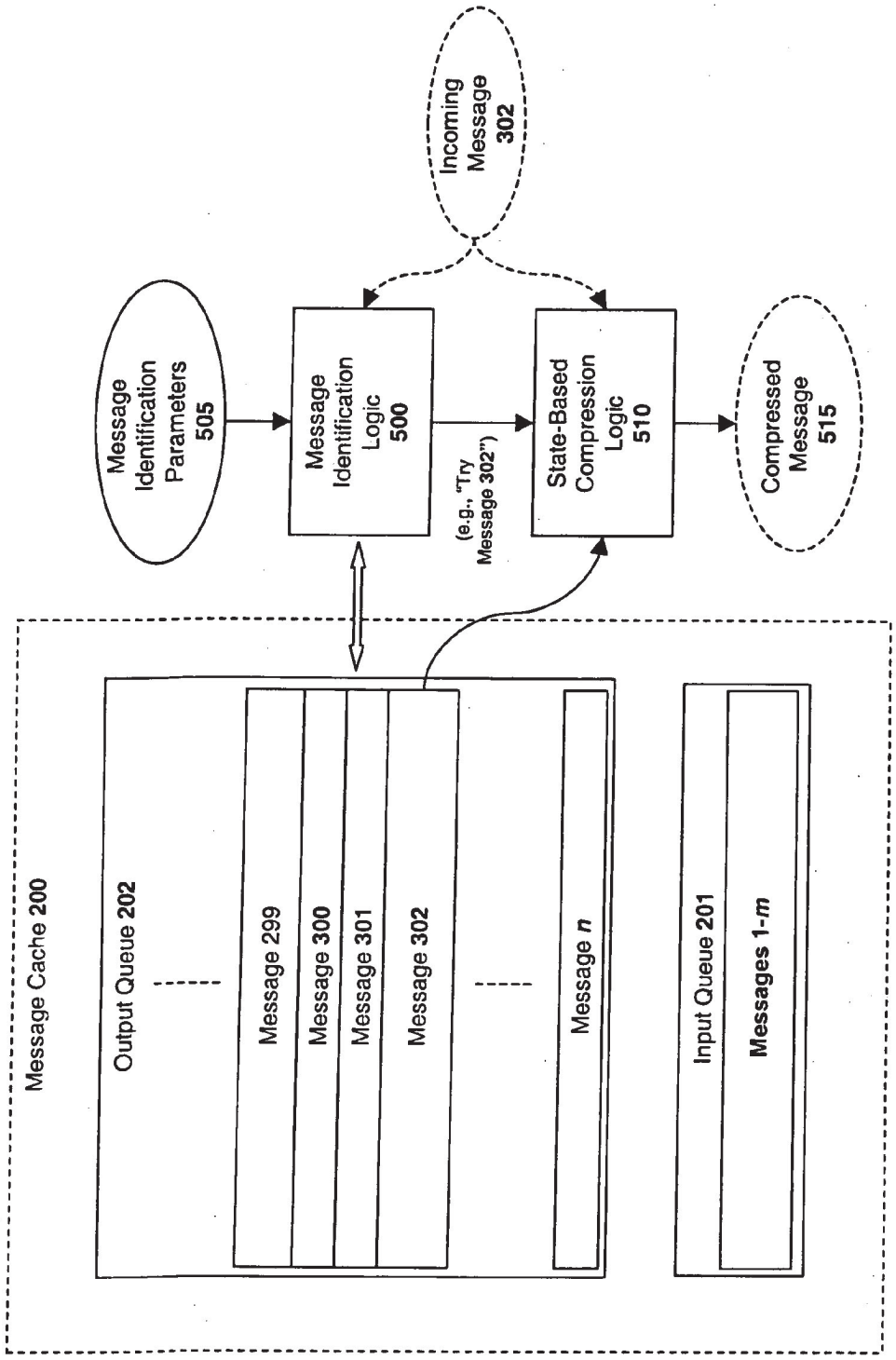

Incoming Message 302
Message Identification Parameters 505
Message Identification Logic 500
(e.g., "Try Message 302")
State-Based Compression Logic 510
Compressed Message 515
Message Cache 200
Output Queue 202
Message 299
Message 300
Message 301
Message 302
Message n
Input Queue 201
Messages 1-m

Fig. 5

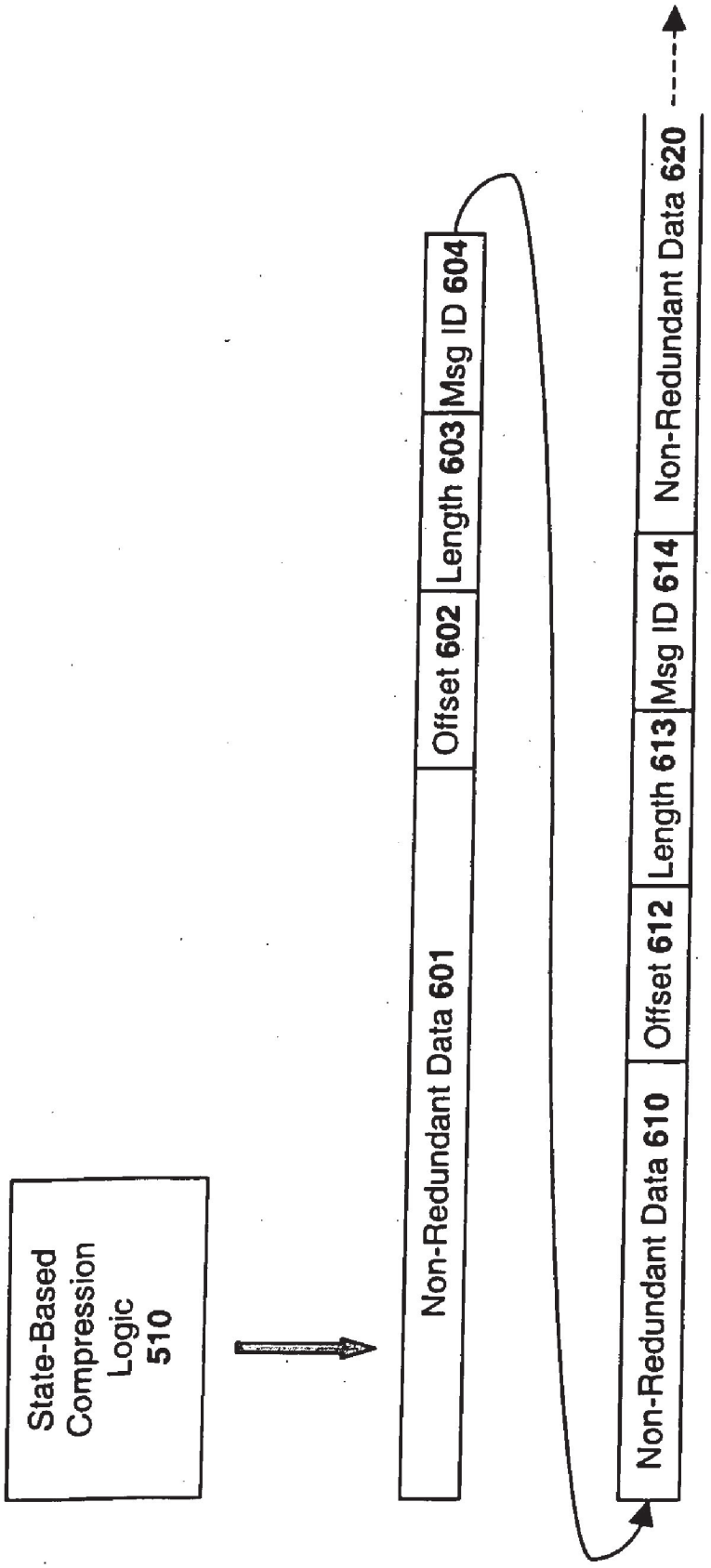
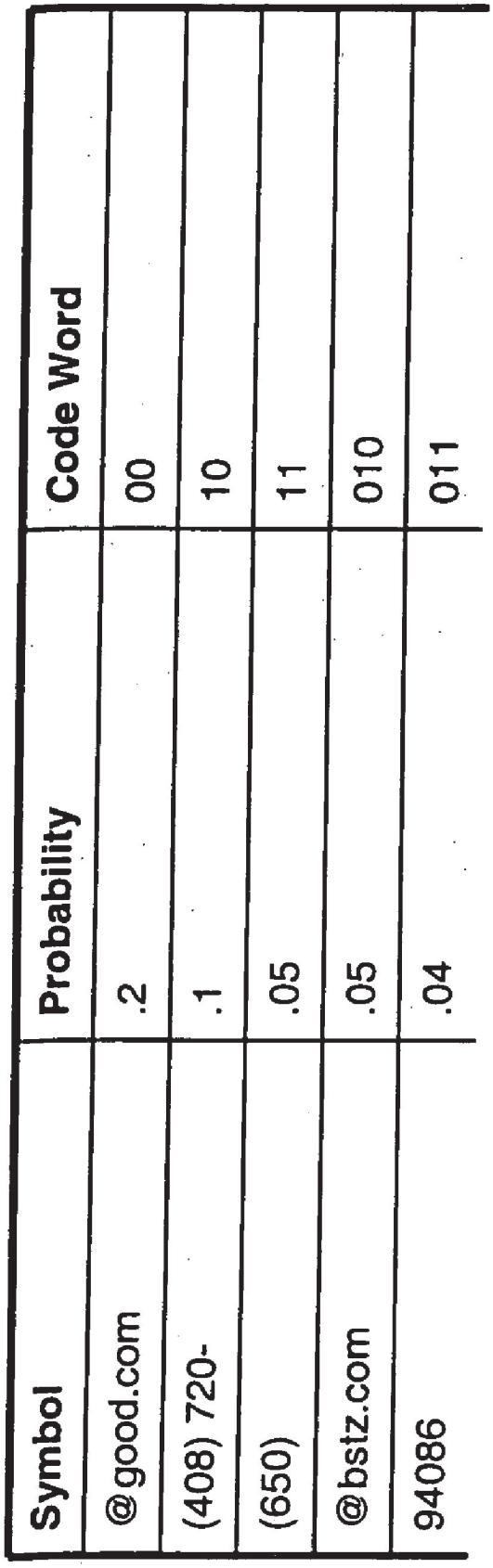

State-Based Compression Logic 510
Non-Redundant Data 601
Offset 602
Length 603
Msg ID 604
Non-Redundant Data 610
Offset 612
Length 613
Msg ID 614
Non-Redundant Data 620
State-Based Compression Format

Fig. 6



Symbol	Probability	Code Word
@good.com	.2	00
(408) 720-	.1	10
(650)	.05	11
@bstz.com	.05	010
94086	.04	011

Code Word Table

Fig. 7

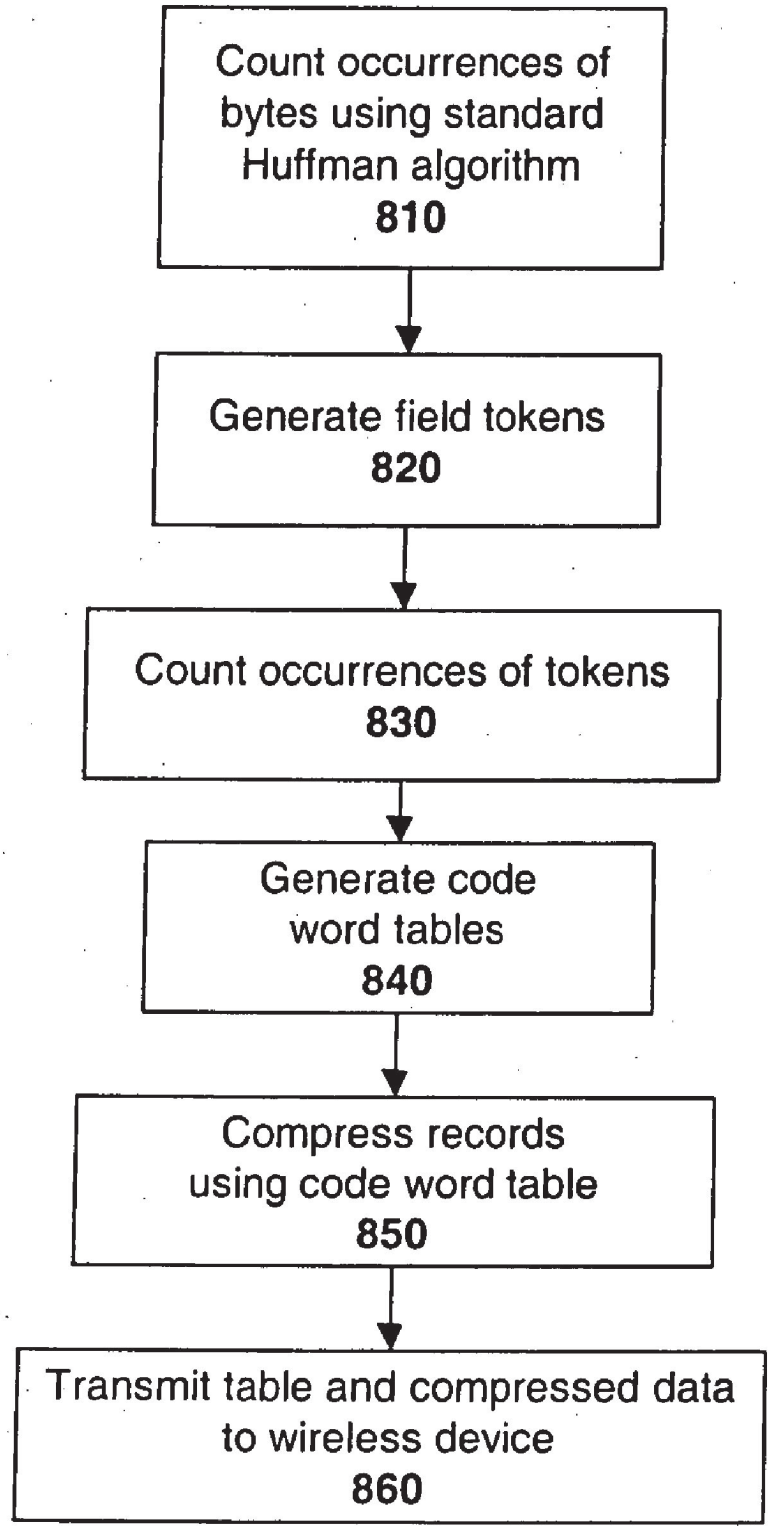

Count occurrences of bytes using standard Huffman algorithm
810
Generate field tokens
820
Count occurrences of tokens
830
Generate code word tables
840
Compress records using code word table
850
Transmit table and compressed data to wireless device
860

Fig. 8

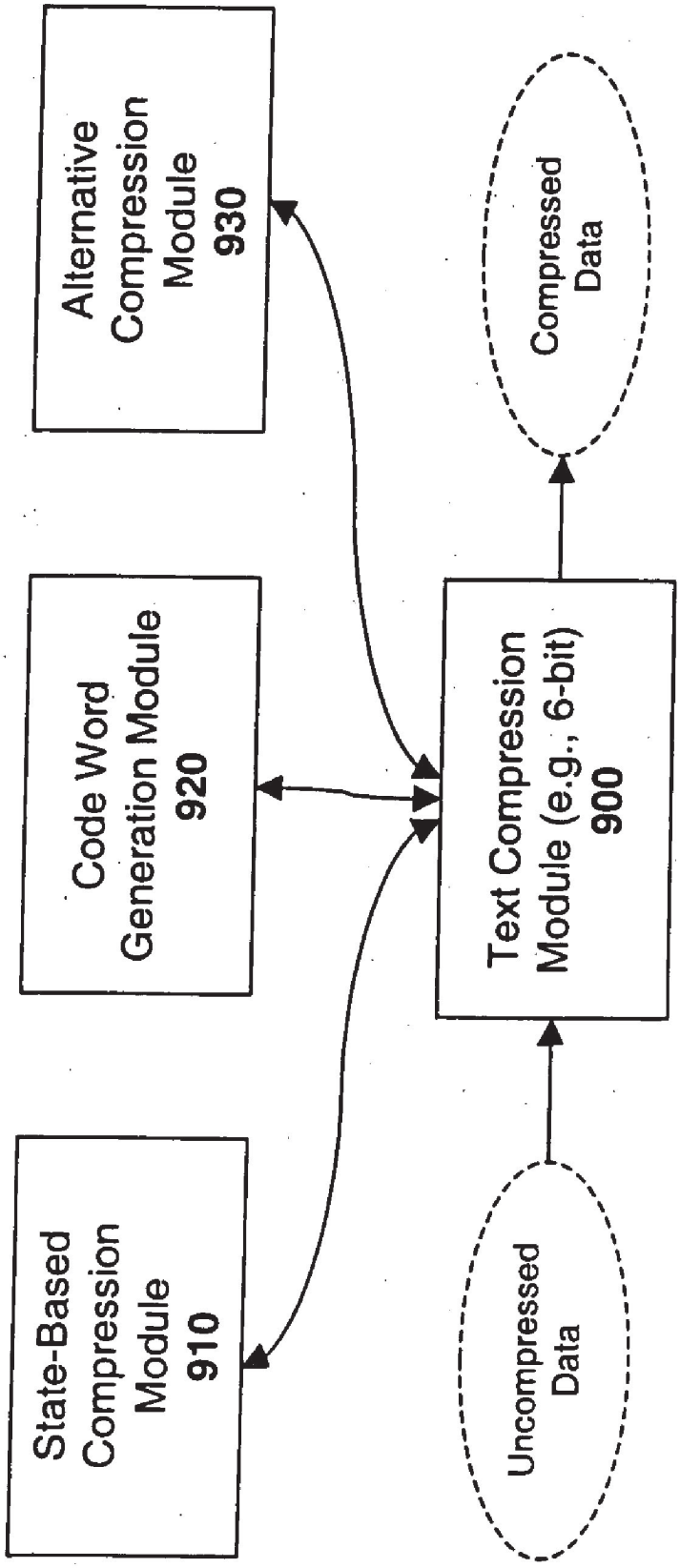

Alternative Compression Module 930
Compressed Data
Code Word Generation Module 920
Text Compression Module (e.g., 6-bit) 900
State-Based Compression Module 910
Uncompressed Data

Fig. 9

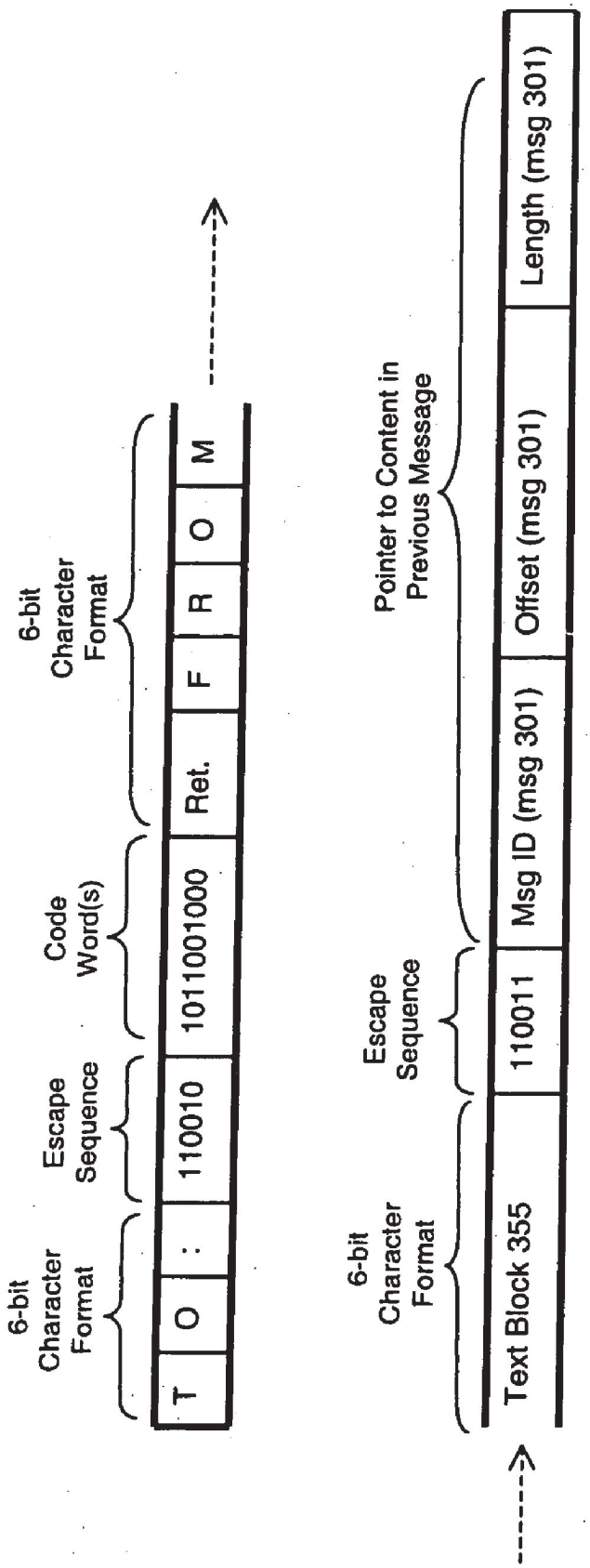

6-bit Character Format
Escape Sequence
Code Word(s)
6-bit Character Format
T
O
:
110010
1011001000
Ret.
F
R
O
M
Pointer to Content in Previous Message
6-bit Character Format
Escape Sequence
Text Block 355
110011
Msg ID (msg 301)
Offset (msg 301)
Length (msg 301)
Encoded Message 302

Fig. 10

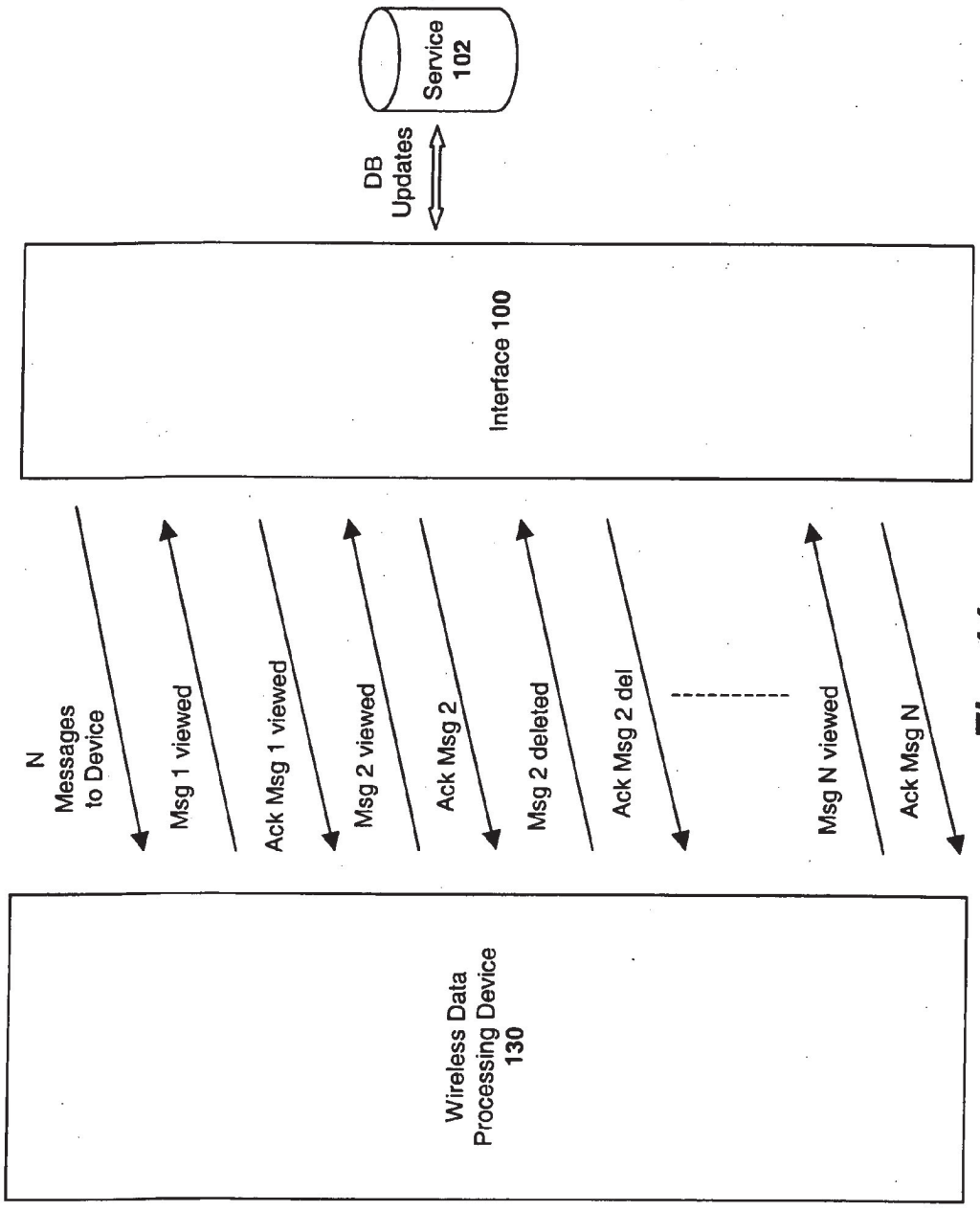

Service 102
DB Updates
Interface 100
N Messages to Device
Msg 1 viewed
Ack Msg 1 viewed
Msg 2 viewed
Ack Msg 2
Msg 2 deleted
Ack Msg 2 del
Msg N viewed
Ack Msg N
Wireless Data Processing Device 130

Fig. 11

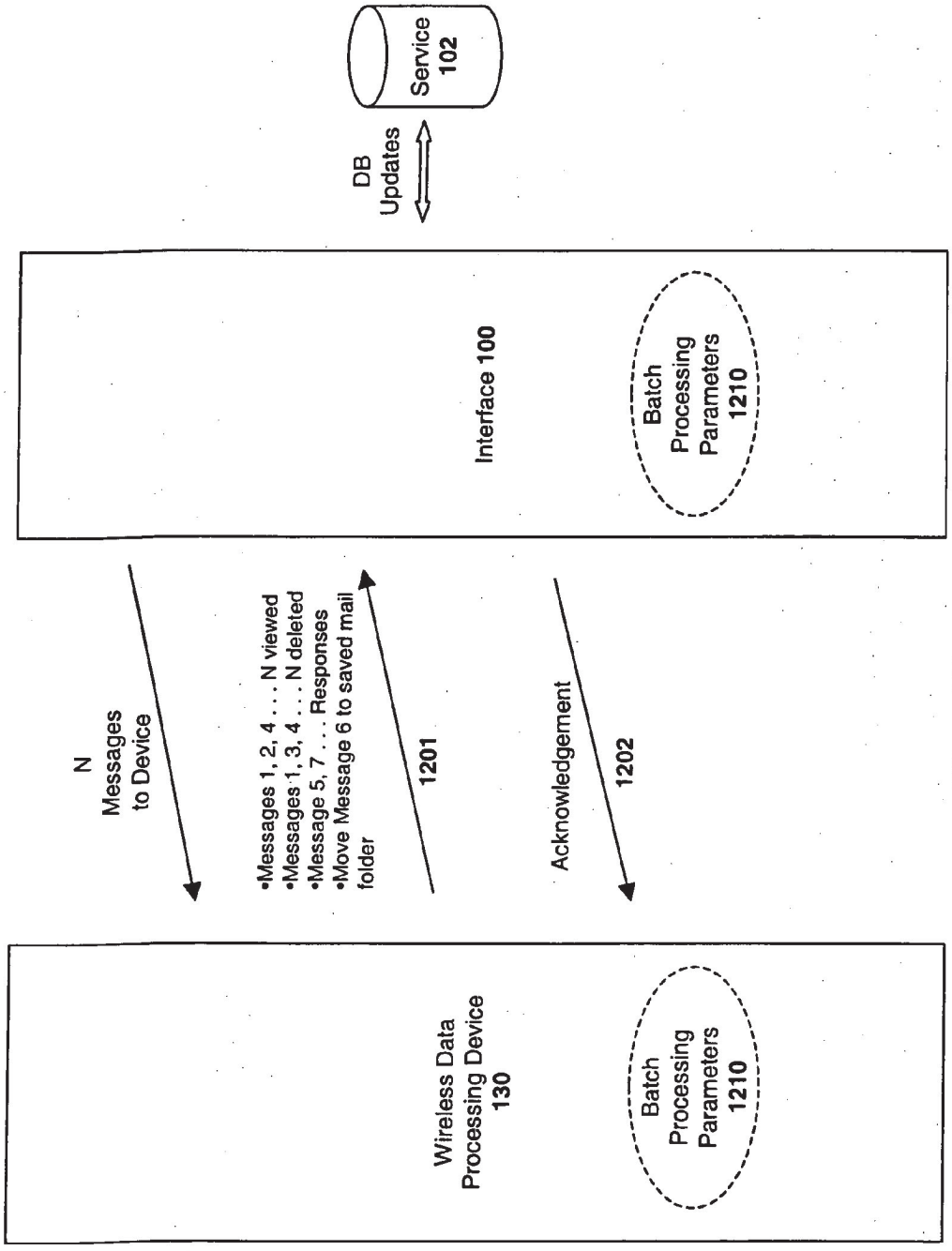

Service 102
DB Updates
Interface 100
Batch Processing Parameters 1210
N Messages to Device
•Messages 1, 2, 4 . . . N viewed
•Messages 1, 3, 4 . . . N deleted
•Message 5, 7 . . . Responses
•Move Message 6 to saved mail folder
1201
Acknowledgement
1202
Wireless Data Processing Device 130
Batch Processing Parameters 1210

Fig. 12

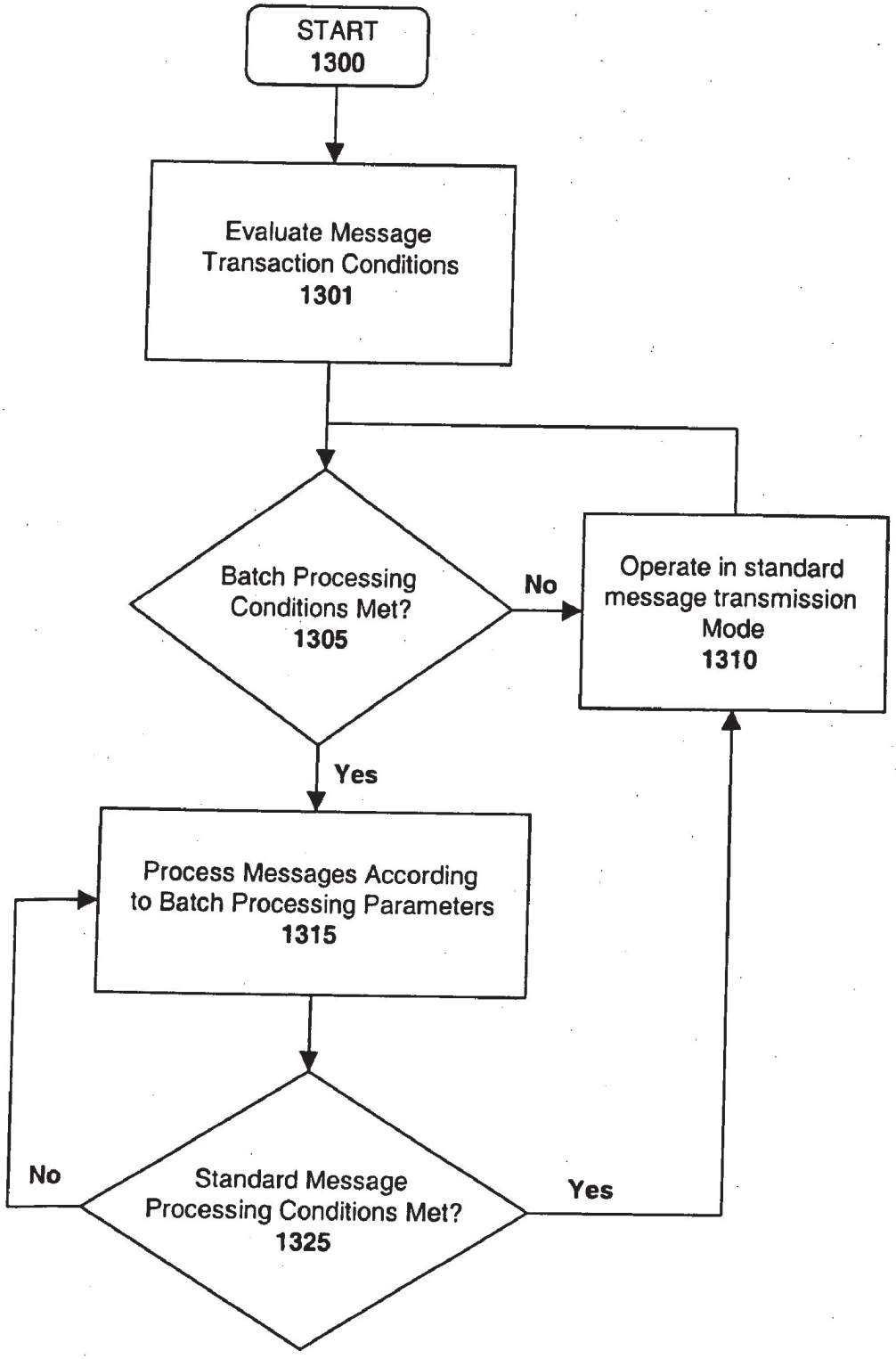

START
1300
Evaluate Message Transaction Conditions
1301
Batch Processing Conditions Met?
1305
No
Operate in standard message transmission Mode
1310
Yes
Process Messages According to Batch Processing Parameters
1315
Standard Message Processing Conditions Met?
1325
No
Yes

Fig. 13

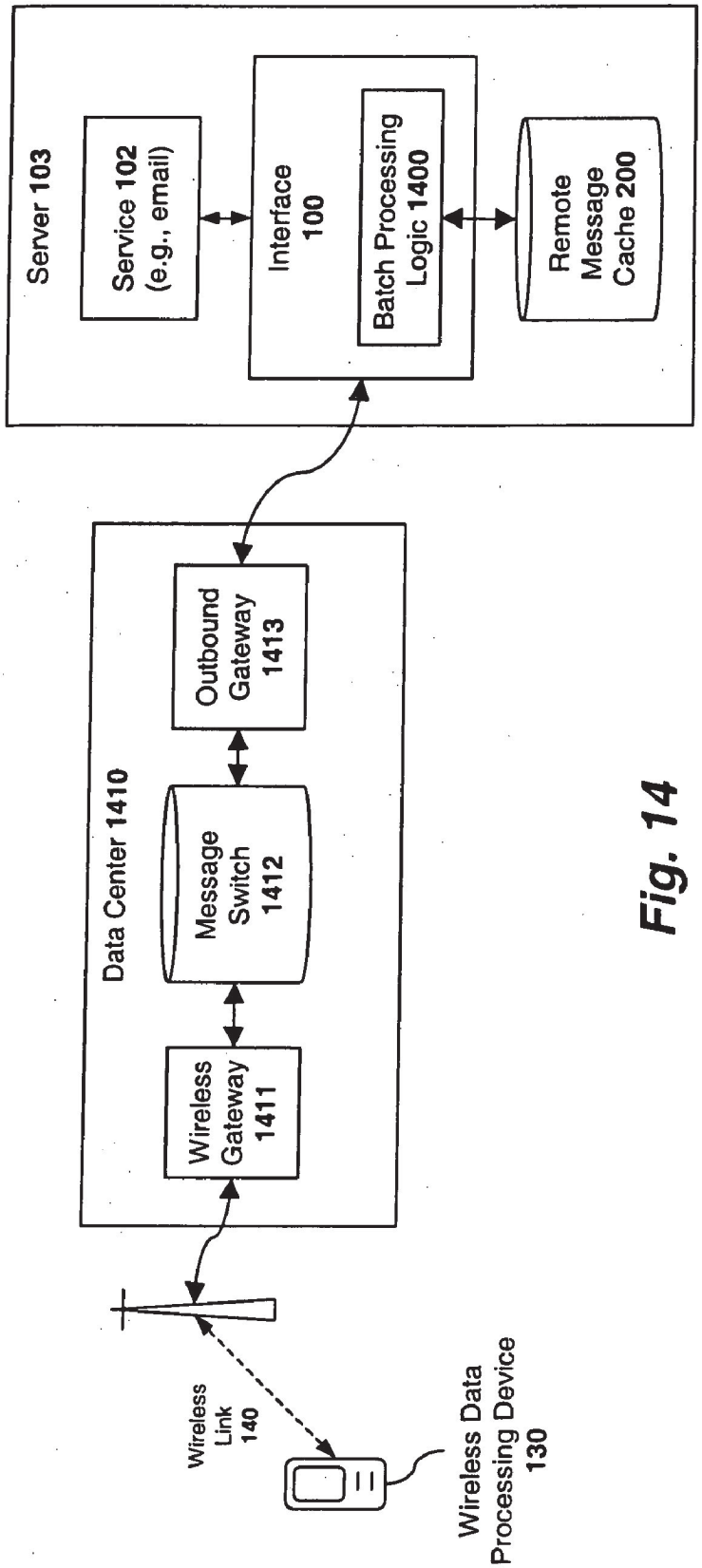

Server 103
Service 102 (e.g., email)
Interface 100
Batch Processing Logic 1400
Remote Message Cache 200
Data Center 1410
Outbound Gateway 1413
Message Switch 1412
Wireless Gateway 1411
Wireless Link 140
Wireless Data Processing Device 130

Fig. 14

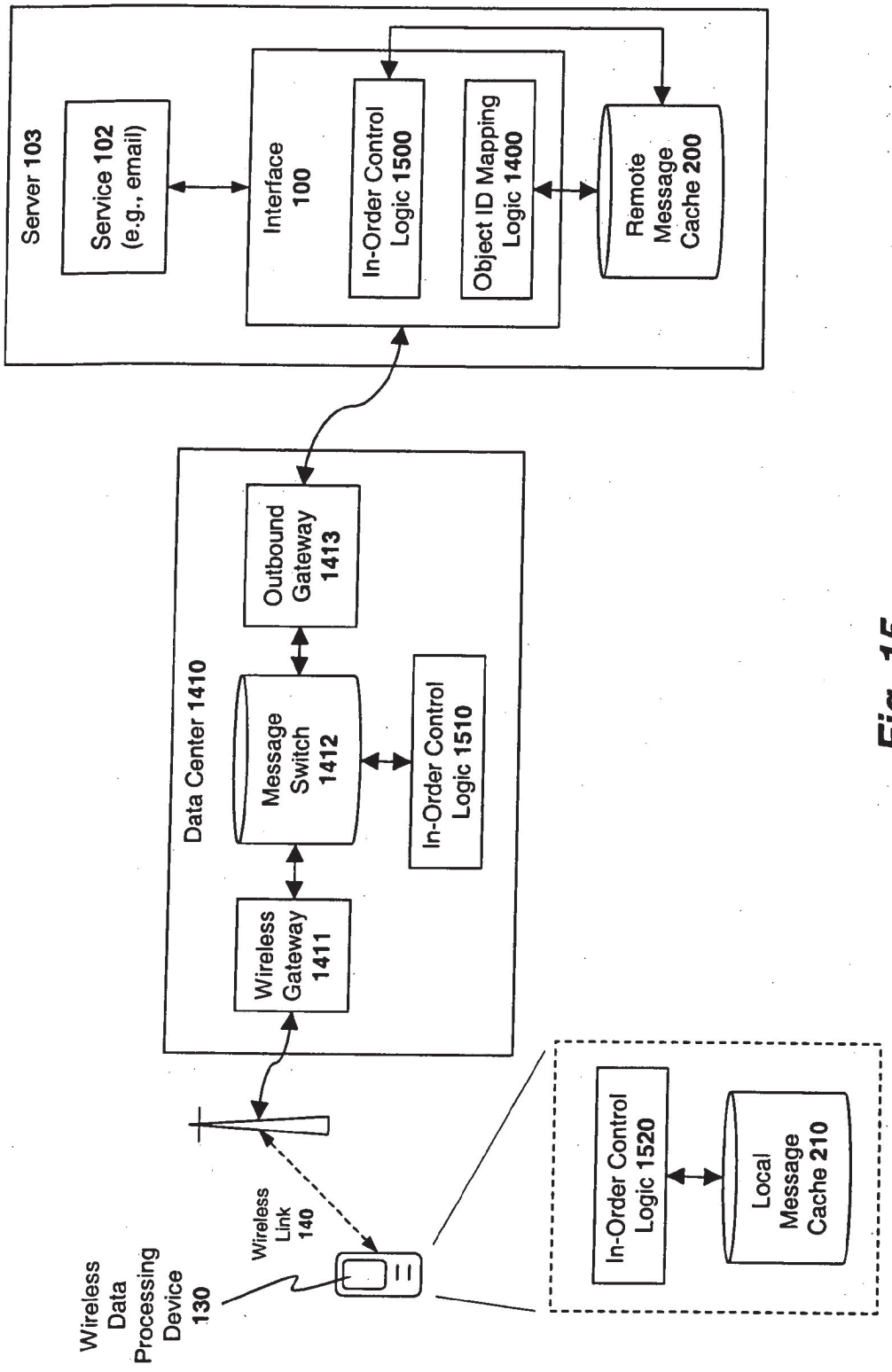

Server 103
Service 102 (e.g., email)
Interface 100
In-Order Control Logic 1500
Object ID Mapping Logic 1400
Remote Message Cache 200
Data Center 1410
Outbound Gateway 1413
Message Switch 1412
In-Order Control Logic 1510
Wireless Gateway 1411
Wireless Link 140
Wireless Data Processing Device 130
In-Order Control Logic 1520
Local Message Cache 210

Fig. 15

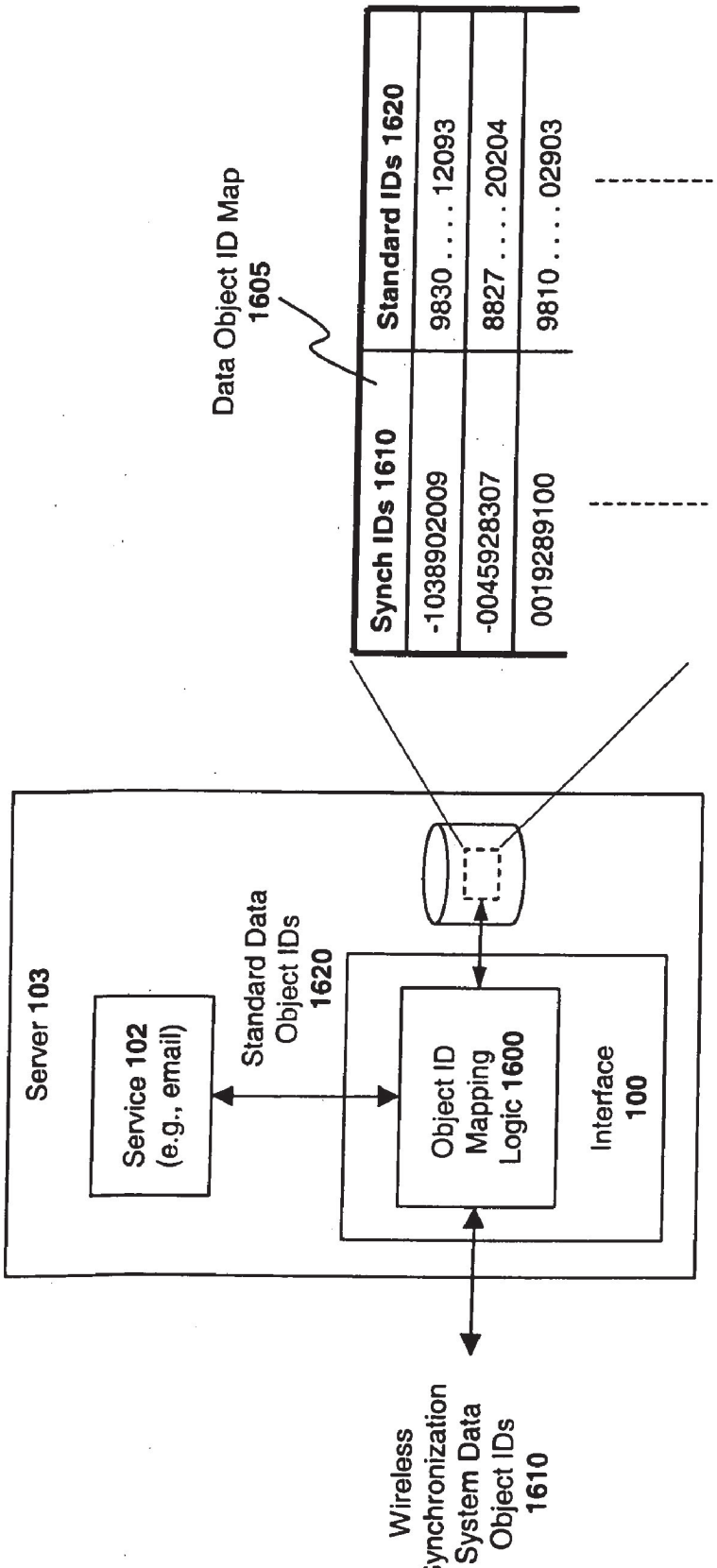

Data Object ID Map
1605
Synch IDs 1610
Standard IDs 1620
-1038902009
9830 12093
-0045928307
8827 20204
0019289100
9810 02903
Server 103
Service 102
(e.g., email)
Standard Data
Object IDs
1620
Object ID
Mapping
Logic 1600
Interface
100
Wireless
Synchronization
System Data
Object IDs
1610

Fig. 16

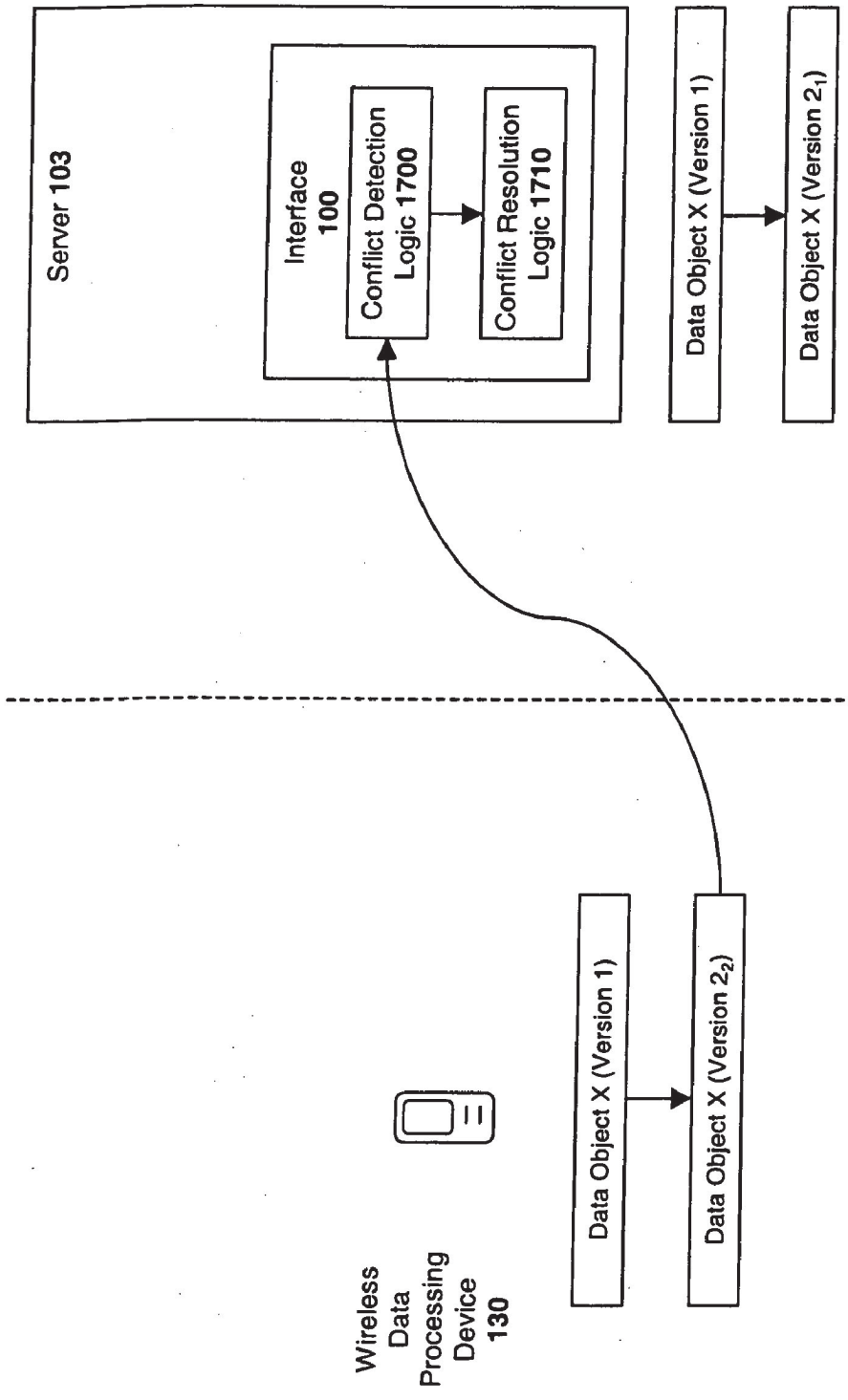

Server 103
Interface 100
Conflict Detection Logic 1700
Conflict Resolution Logic 1710
Data Object X (Version 1)
Data Object X (Version 2_1)
Wireless Data Processing Device 130
Data Object X (Version 1)
Data Object X (Version 2_2)

Fig. 17

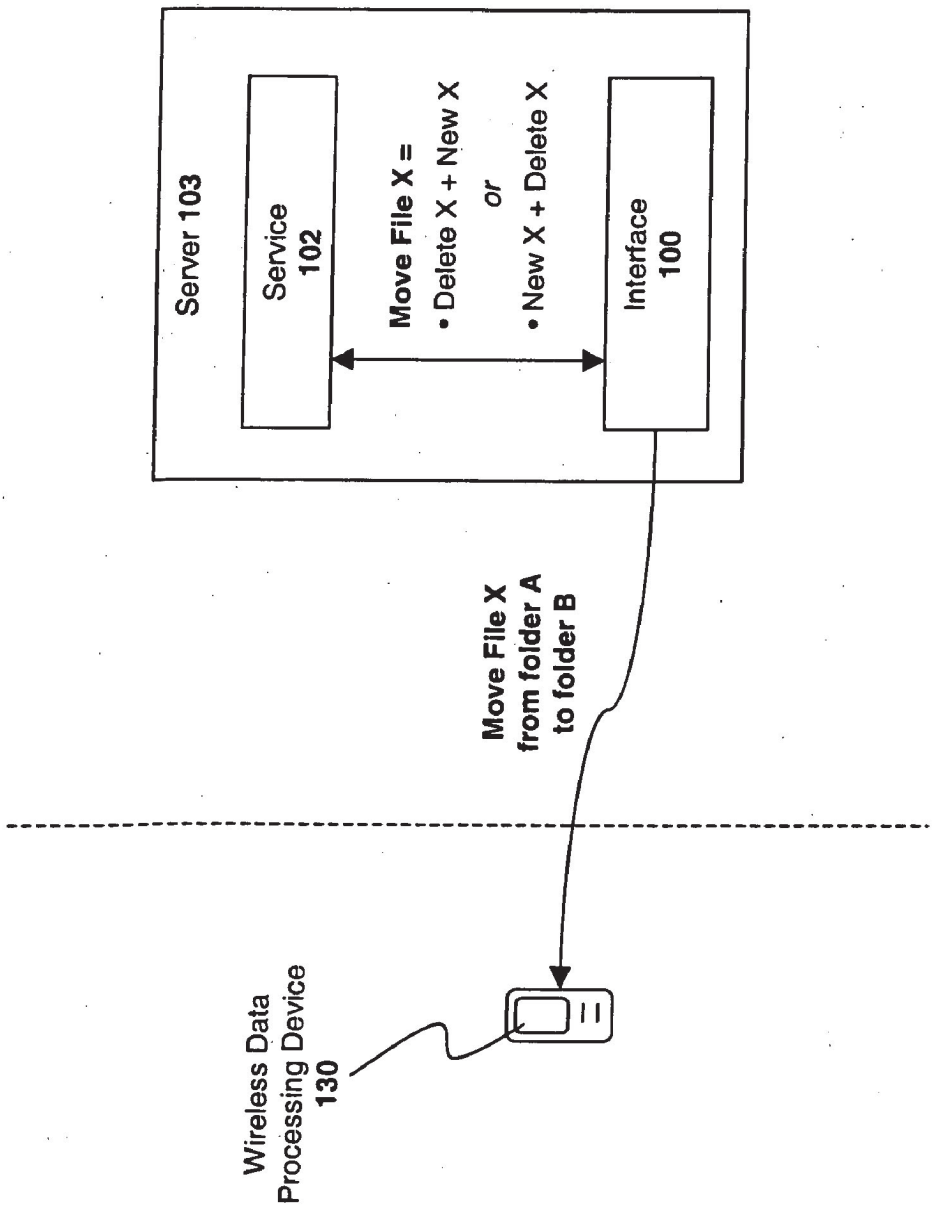

Server 103
Service 102
Move File X =
• Delete X + New X
or
• New X + Delete X
Interface 100
Move File X from folder A to folder B
Wireless Data Processing Device 130

Fig. 18

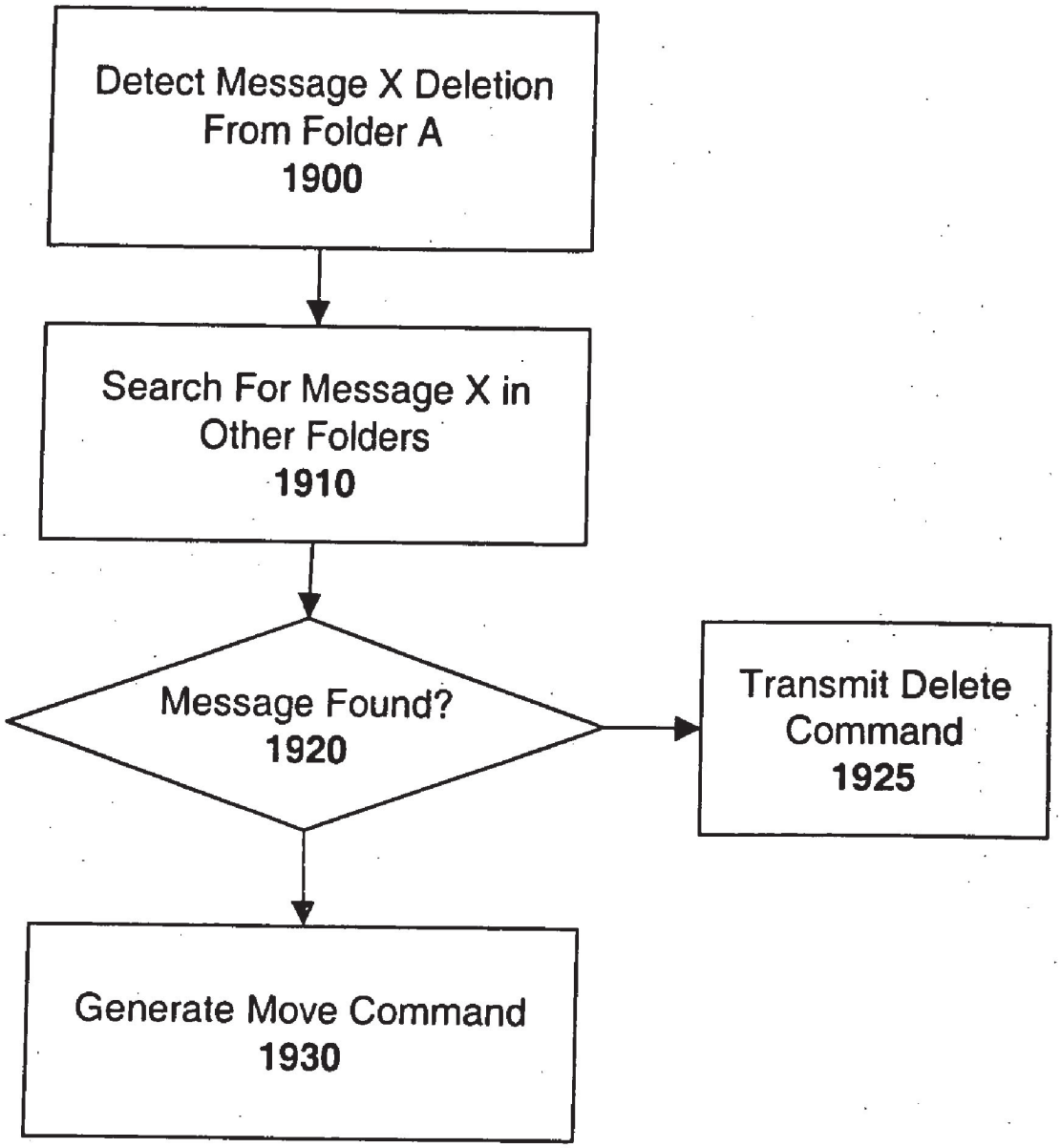

Detect Message X Deletion From Folder A
1900
Search For Message X in Other Folders
1910
Message Found?
1920
Transmit Delete Command
1925
Generate Move Command
1930

Fig. 19a

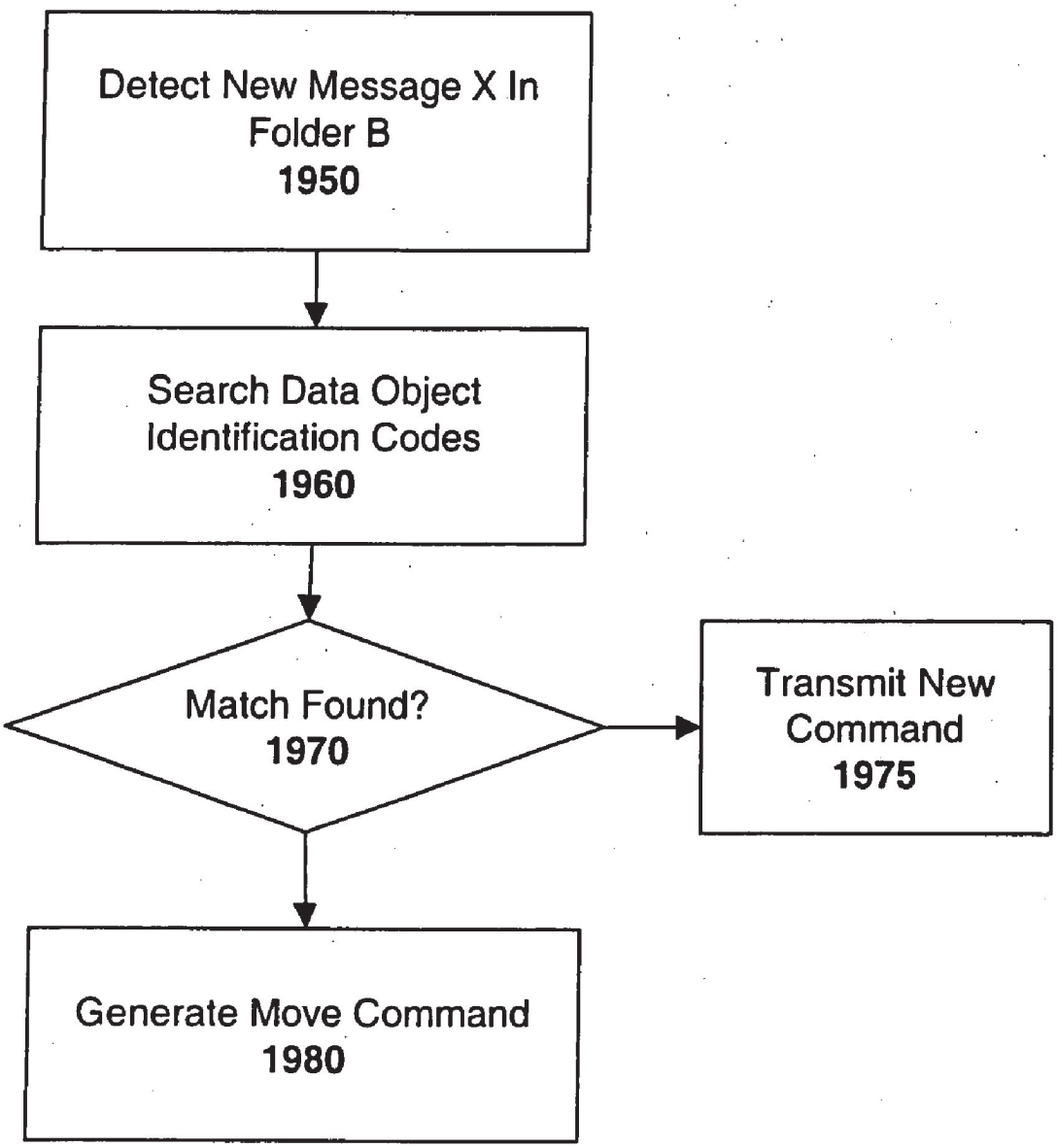

Detect New Message X In Folder B
1950
Search Data Object Identification Codes
1960
Match Found?
1970
Transmit New Command
1975
Generate Move Command
1980

Fig. 19b

SYSTEM AND METHOD FOR FULL WIRELESS SYNCHRONIZATION OF A DATA PROCESSING APPARATUS WITH A MESSAGING SERVICE

PRIORITY

[0001] This application is a continuation-in-part of co-pending U.S. Application entitled APPARATUS AND METHOD FOR CONSERVING BANDWIDTH BY BATCH PROCESSING DATA TRANSACTIONS, Ser. No. 09/924,283, filed Aug. 7, 2001.

BACKGROUND

[0002] 1. Field of the Invention

[0003] This invention relates generally to the field of network data services. More particularly, the invention relates to an apparatus and method for synchronizing a wireless data processing device with a wireless messaging service.

[0004] 2. Description of the Related Art

[0005] A variety of wireless data processing devices have been introduced over the past several years. These include wireless personal digital assistants ("PDAs") such as the Palm® VIx handheld, cellular phones equipped with data processing capabilities (e.g., those which include wireless application protocol ("WAP") support), and, more recently, wireless messaging devices such as the Blackberry™ wireless pager developed by Research In Motion ("RIM").™

[0006] Personal digital assistants such as the Palm devices typically provide only limited wireless messaging capabilities (e.g., instant messaging and basic Internet access). For example, these devices typically require a user to manually establish a connection to the Internet via an Internet Service Provider ("ISP") or to a corporate server to check e-mail messages.

[0007] Although corporate messaging systems such as the RIM Blackberry provide more comprehensive messaging capabilities, there are significant limitations to these systems. Specifically, these systems employ e-mail "redirection" or "forwarding" techniques in which messages are redirected to the wireless device only if certain conditions are met. These conditions, referred to as "redirection events," may include, for example, an indication that the user is not working at his corporate desktop (e.g., removal of the wireless device from the desktop cradle, a screen saver firing on the desktop, . . . etc) or a manual redirection command initiated by the user (e.g., via the wireless device or the corporate desktop). One such message redirection system is described in U.S. Pat. No. 6,219,694 ("System and Method for Pushing Information From a Host System to a Mobile Data Communication Device Having a Shared Electronic Address").

[0008] As a result, these systems are (as a practical matter) incapable of providing complete synchronization between the wireless device and the corporate e-mail database. For example, because messages are only redirected to the wireless device under certain conditions (e.g., following a redirection event), at any given point in time, the wireless device may contain an incomplete set of e-mail data. Moreover, even when messages are actively being forwarded to. the wireless device, the e-mail data stored at the wireless device and the e-mail database are not truly synchronized. For example, certain types of transactions performed on the wireless device, such as an indication that a message has been viewed by the user, message deletions, movement of messages from one folder to another, . . . etc., are not updated at the e-mail service wirelessly.

[0009] Moreover, only basic e-mail functions such as sending and receiving messages may be controlled at the wireless device. More advanced e-mail management functions must be set at the user's desktop (e.g., configuring e-mail filters, outgoing e-mail signatures, security settings such as passwords, . . . etc).

[0010] In addition, prior messaging systems require a corporate desktop to which the device must be attached when the user is in the office. The problem with this is not merely that a corporate desktop is required, but also that the corporate desktop must be configured with software and a proprietary "cradle" that allows it to communicate directly to the wireless device.

[0011] Accordingly, what is needed is a system and method for providing complete synchronization and management between a wireless device and a messaging service (e.g., a corporate e-mail account). What is also needed is a wireless apparatus for receiving and sending e-mail messages which does not require a corporate desktop or any software to be installed and executed on the corporate desktop.

SUMMARY

[0012] A system is disclosed in which a data processing device is completely synchronized with a messaging service. One embodiment of the system comprises a wireless data processing device; a messaging service to maintain messages and other information on behalf of a user; and synchronization logic for maintaining synchronization of the messages and other information between the wireless device and the messaging service.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] A better understanding of the present invention can be obtained from the following detailed description in conjunction with the following drawings, in which:

[0014] FIG. **1** illustrates an exemplary network architecture used to implement elements of the present invention.

[0015] FIG. **2** illustrates one embodiment of a system for compressing data.

[0016] FIGS. **3***a-c* illustrate an exemplary sequence of related e-mail messages.

[0017] FIG. **4** illustrates one embodiment of a method for compressing data using redundant data found in previous messages.

[0018] FIG. **5** illustrates one embodiment of an apparatus for performing state-based compression.

[0019] FIG. **6** illustrates one embodiment of a state-based data compression format.

[0020] FIG. **7** illustrates a code word table employed to compress data according to one embodiment of the invention.

[0021] FIG. **8** illustrates one embodiment of a method for compressing data with code words.

[0022] FIG. **9** illustrates a text compression module coordinating data compression tasks between a plurality of other compression modules.

[0023] FIG. **10** illustrates a compressed data format according to one embodiment of the invention.

[0024] FIG. **11** illustrates one embodiment of a system for synchronizing message transactions between a wireless device and a service.

[0025] FIG. **12** illustrates an improved embodiment of a system for synchronizing message transactions between a wireless device and a service.

[0026] FIG. **13** illustrates a method for determining whether to enter a batch processing mode.

[0027] FIG. **14** illustrates an embodiment of the invention which employs multi-level batch processing.

[0028] FIG. **15** illustrates an embodiment of the invention which employs in-order control functions.

[0029] FIG. **16** illustrates an embodiment of the invention which maps synchronization identification codes to standard identification codes.

[0030] FIG. **17** illustrates an embodiment of the invention for detecting and resolving data object version conflicts.

[0031] FIG. **18** illustrates an embodiment of the invention in which a move command is transmitted to a wireless device in lieu of a delete command and a new command.

[0032] FIGS. **19***a* and *b* illustrate embodiments of a method for generating a move command.

DETAILED DESCRIPTION

[0033] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without some of these specific details. In other instances, well-known structures and devices are shown in block diagram form to avoid obscuring the underlying principles of the present invention.

An Exemplary Network Architecture

[0034] FIG. **1** illustrates one embodiment of a network architecture for implementing the compression techniques described herein. The "customer site"**120** illustrated in FIG. **1** may be any local-area or wide-area network over which a plurality of servers **103** and clients **110** communicate. For example the customer site may include all servers and clients maintained by a single corporation. The servers **103** may be configured to provide a variety of different messaging and groupware services **102** to network users (e.g., e-mail, instant messaging, calendaring, . . . etc). In one embodiment, these services are provided by Microsoft Exchange. However, the underlying principles of the invention are not limited to any particular messaging/ groupware platform.

[0035] In one embodiment of the invention, an interface **100** forwards data objects maintained by the service **102** (e.g., e-mail messages, instant messages, calendar data, . . . etc) to a plurality of wireless data processing devices (represented in FIG. **1** by wireless device **130**) via an external data network **170** and/or a wireless service provider network **171**. For example, if the service **102** includes an e-mail database, the interface **100** transmits any new e-mails which arrive in a user's mailbox on the service **102** to the user's wireless data processing device **130** (over the network(s) **170** and/or **171**). Alternatively, or in addition, the service **102** may provide the e-mail to the user's local computer (e.g., client **110**) upon request (i.e., so that the user will receive the e-mail on his/her wireless device **130** when out of the office and on his/her personal computer **110** when in the office). Conversely, e-mail messages sent from the user's wireless data processing device **130** are transmitted to the service **102** via the interface **100**.

[0036] In one embodiment, the interface **100** is a software module adapted to work with the particular service **120**. It should be noted, however, that the interface **100** may be implemented in hardware or any combination of hardware and software while still complying with the underlying principles of the invention.

[0037] In one embodiment, the external data network **170** is comprised of a plurality of servers/clients (not shown) and other networking hardware (e.g., routers, hubs, . . . etc) for transmitting data between the interface **100** and the wireless devices **130**. In one embodiment, the interface **100** encapsulates data in one or more packets containing an address identifying the wireless devices **130** (e.g., such as a 24-bit Mobitex Access Number ("MAN #")). The external data network **170** transmits the packets to a wireless service provider network **171** which, in turn, transmits the packets (or the data contained therein) over a wireless communication link to the wireless device **130**. In one embodiment, the wireless service provider network is a 2-way paging network. However, various other network types may be employed (e.g., CDMA 2000, PCS, . . . etc) while still complying with the underlying principles of the invention.

[0038] It should be noted that the network service provider network **171** and the external data network **170** (and associated interface **100**) may be owned/operated by the same organization or, alternatively, the owner/operator of the external data network **170** may lease wireless services from the wireless service provider network. The underlying principles of the invention are not limited to any particular service arrangement.

[0039] In one embodiment of the invention, the service **102** (e.g., the e-mail database) is fully synchronized with the wireless data processing device **130**. Thus, any actions performed on the wireless device **130** are automatically updated on the service **102** and any transactions occurring at the service **102** are automatically reflected on the device **130**. Synchronization updates of this type may include but are not limited to device configuration modifications, calendar updates, e-mail message updates, instant messages, to-do list updates and/or any other type of personal information management transactions or corporate data management transactions (hereinafter "message transactions"). As one example, when a user views an e-mail message using the device **130**, an indication that the user viewed the message is transmitted to the service **102** (via the interface **100**). Accordingly, if the user subsequently connects to e-mail via

a client **110**, the e-mail will appear as having already been viewed. Other actions such as message deletions, filing activities (e.g., moving a message to a particular folder), message responses, meeting confirmations/additions . . . etc, will automatically be reflected in the service **102**, thereby providing complete synchronization between the service **102**, the device **130** and/or the client **110** (if one is being used).

[**0040**] Current messaging systems do not offer complete wireless device synchronization. As such, these systems require that the user have a desktop computer with a "cradle" to which the device is attached to received certain types of synchronization updates. One reason for this is that prior systems process message transactions in a relatively inefficient manner and employ only limited compression techniques, thereby making complete synchronization impractical. As such, in order to realize complete wireless synchronization, embodiments of the invention employ one or more of the following compression and/or message processing techniques.

State-Based Compression

[**0041**] FIG. **2** illustrates certain aspects of the wireless data processing device **130** and the interface **100** in greater detail. In one embodiment, the data processing device **130** is comprised of a local data compression/decompression module **225** (hereinafter "codec module 225") and a local message cache **210**. The local codec module **225** compresses outgoing data and decompresses incoming data using the various compression techniques described herein.

[**0042**] The local message cache **210** is comprised of an input queue **211** for temporarily storing a incoming messages and an output queue **212** for storing outgoing messages. Although illustrated as separate logical units in FIG. **2**, the local message cache **210** may be comprised of only a single block of memory for storing both incoming and outgoing messages according to a cache replacement policy. In one embodiment, messages are maintained in the input queue and/or output queue using a first-in, first-out ("FIFO") replacement policy. However, various other cache replacement techniques may be employed while still complying with the underlying principles of the invention. For example, a least-recently used ("LRU") policy may be implemented where messages used least frequently by the local codec module **225** are stored in the cache for a shorter period of time than those used more frequently. As described below, messages used more frequently by the local codec module **225** may frequently include messages which form part of a common e-mail thread, whereas those used less frequently may include junk mail or "spam" (i.e., for which there is only a single, one way message transmission).

[**0043**] The interface **100** of one embodiment is comprised of a remote data compression/decompression module **220** (hereinafter "codec module **220**") and a remote message cache **200** with a remote input queue **201** and a remote output queue **202**. The codec module **220** compresses messages transmitted to the wireless data processing device **130** and decompresses messages received from the data processing device **130** according to the techniques described herein. The remote message cache **200** temporarily stores messages transmitted to/from the data processing device **130** (e.g., using various cache replacement algorithms as described above). In one embodiment, the cache replacement policy implemented on the interface **100** is the same as the policy implemented on the wireless device **130** (i.e., so that cache content is synchronized between the remote cache **200** and the local cache **210**).

[**0044**] FIGS. **3***a-c* illustrate an exemplary sequence of e-mail messages which will be used to describe various aspects of the invention. FIG. **3***a* illustrates the initial e-mail message **300** in the sequence which (like most e-mail messages) is logically separated into a header information portion **305** and a text information portion **310**. Also shown in FIG. **3***a* is an attachment **320**, indicating that a document is attached to the message and an electronic signature which may be automatically inserted in the message by the sender's (i.e., John Smith's) e-mail client.

[**0045**] FIG. **3***b* illustrates the second e-mail message **301** in the sequence transmitted by user Roger Collins in response to the initial e-mail message. As indicated by the new header information **335**, this message is transmitted directly to the initial sender, John Smith, and to a user who was CC'ed on the initial e-mail message, Tom Webster. The message is also CC'ed to everyone else in the group to whom the initial message was transmitted. This "reply to all" feature, which is found in most e-mail clients, provides a simple mechanism for allowing a sequence of e-mail messages to be viewed by a common group of individuals.

[**0046**] As illustrated in FIG. **3***b*, the text **310** of the initial e-mail message **300** is substantially reproduced in the new e-mail message. This "reply with history" feature is also common to most e-mail clients, allowing a sequence of comments from the individuals in the common group to be tracked from one e-mail message to the next. Also illustrated are a plurality of characters **316** inserted by the responder's (Roger Collins') e-mail system at the beginning of each line of the original e-mail text. This feature, which is common in some (but not all) e-mail systems, allows users to differentiate between new text and old text.

[**0047**] Accordingly, even after the initial e-mail response in a sequence of e-mails, the e-mail history (i.e., the portions of text and attachments reproduced from prior messages) represents a significant portion of the overall message, resulting in the transmission of a significant amount of redundant information being transmitted over the wireless network, in both the text portion of the e-mail and the header portion of the e-mail.

[**0048**] FIG. **3***c* illustrates the final e-mail message **302** in the sequence in which the addressee of the second e-mail responds to the sender of the second e-mail and CC's all of the other members in the group. As illustrated, the only non-redundant information in the e-mail message **302** is a few lines of text **355**. The e-mail addresses of all of the group members are the same as in the previous two messages (although switched between different fields, the underlying addresses are the same) and the text and header information from the previous messages **300**, **301**, including the attachment **320** are reproduced, with only a few minor modifications (e.g., the additional ">"characters inserted by the e-mail system).

[**0049**] One embodiment of the invention compresses e-mail messages by taking advantage of this high level of redundancy. In particular, rather than sending the actual

content contained in new e-mail messages, portions of the new messages identified in previous e-mail messages stored in the caches **200**, **201** are replaced by pointers to the redundant portions. For example, in message **302** all of the redundant content from message **301** may be replaced by a pointer which identifies the redundant content in message **301** stored in the cache of the user's wireless device. These and other compression techniques will be described in greater detail below.

[**0050**] FIG. **4** illustrates one embodiment of a method for compressing messages using redundant content found in previous messages. This embodiment will be described with respect to FIG. **5**, which illustrates certain aspects of the message interface **100** in greater detail. At **400**, the interface **100** receives a message (or a group of messages) to be transmitted to a particular wireless data processing device **130**. At **405**, the message is analyzed to determine whether it contains redundant data found in previous messages. In one embodiment, this is accomplished via message identification logic **500** shown in FIG. **5** which scans through previous e-mail messages to locate those messages containing the redundant data.

[**0051**] Various message identification parameters **505** may be used by the message identification logic **500** to search for messages. For example, in one embodiment, the message identification logic will initially attempt to determine whether the new message is the latest in a sequence of messages. Various techniques may be employed by the message identification logic **500** to make this determination. For example, in one embodiment, the message identification logic **500** will search the subject field of the message for the stings which indicate the new message is a response to a prior message. If these strings are identified, the message identification logic **500** may then look for the most recent message in the sequence (e.g., based on the text found in the subject field). For example, referring back to the FIGS. **3***a-c,* upon receiving message **302**, the message identification logic **500** may identify the message **302** as part of a sequence based on the fact that it contains "RE: Patent Issues" in the subject field. The identification logic **500** may ignore the RE: (or FW: if the message is forwarded) and scan to the text in another message which matches the remainder of the subject field (i.e., "Patent Issues") and identify the most recent previous message containing that text in it's subject header.

[**0052**] If the message subject does not contain characters such as RE: or FW: indicating that the message is part of a sequence, then message identification logic **500** may employ a different set of identification parameters **505** for identifying previous messages. For example, in one embodiment, the message identification logic **500** will search for the most recent message in which the sender of the new message is listed in the header (e.g., as the recipient). Moreover, the message identification logic **500** may search for certain keywords or combinations of words indicating that the message contains relevant data (e.g., such as the electronic signature **315** illustrated in FIGS. **3***a-c*). In one embodiment, the message identification logic **500** may generate a prioritized subset of messages which (based on the defined parameters **505**) are the candidates most likely to contain content found in the new message.

[**0053**] If no redundant data exists in prior messages, determined at **410**, then at **420** additional compression techniques are applied to compress the message, some of which are described below. If, however, redundant data exists in prior messages then, at **415**, the redundant data is replaced with pointers/offsets identifying the redundant data on the cache **210** of the wireless device **130** (or in the cache **200** of the interface **100**, depending on the direction of message transmission). As illustrated in FIG. **5**, in one embodiment, this is. accomplished by state based compression logic **510** which generates the pointers/offsets using the messages identified by the message identification logic **500**.

[**0054**] FIG. **6** illustrates one embodiment of a state-based compression format generated by the state-based compression logic **510**. As illustrated, the format is comprised of a one or more chunks of non-redundant data **601**, **610**, **620** separated by offsets **602**, **612**, lengths **603**, **613**, and message identification data **604**, **614**, which identify blocks of data from previous messages. For example, if the compression format of FIG. **6** were used to encode message **302** shown in FIG. **3***c,* the new text **302** might be stored as non-redundant data **601**, whereas all of message **301** might be identified by a particular message ID **604**, followed by an offset **602** identifying where to begin copying content from message **301** and a length **603** indicating how much content to read from the address point identified by the offset.

[**0055**] Similarly, if message **301** from FIG. **3***b* were encoded by the state-based compression logic **510**, the new text portion **340** might be stored as non-redundant data **601**. Moreover, each of the ">" characters automatically inserted by the e-mail system **316** might be transmitted as non-redundant data, separated by lines of redundant data identified by offsets and lengths (i.e., at the end of each redundant line in message **300** identified by lengths/offsets in the new message, a new, non-redundant ">" would be inserted).

[**0056**] In one embodiment, when a user has not received messages for a long period of time, numerous related messages (e.g., such as messages **300-302**) may build up in his inbox on the e-mail service **102**. Accordingly, in one embodiment, the interface **100** will employ state-based compression techniques as described above using pointers to messages which have not yet arrived in the cache of the user's wireless device. That is, the interface **100** will determine where messages in the group (stored in the user's inbox on the service **102**) will be stored in the cache **210** of the wireless data processing device **130** once the user reconnects to the service.

[**0057**] Referring once again to FIGS. **4** and **5**, once the state-based compression logic **510** finishes compressing the message, the compressed message **515** may be transmitted to the user's wireless device **130**. Alternatively, at **420**, additional compression techniques (described below) may be applied to compress the message further. Once the message is fully compressed it is transmitted to the wireless device (at **425**) where it may be decompressed via codec module **225**.

[**0058**] The state-based compression techniques were described above in the context of an interface **100** compressing messages before transmitting the messages to a wireless device **130**. It will be appreciated, however, that the same compression techniques may be performed by the wireless device **130** before it transmits a message to the interface **100** (e.g., lengths/offsets may identify redundant data stored in the remote message cache **200**). In addition,

although described above with respect to e-mail messages, the described compression techniques may be employed to compression various other message types (e.g., newsgroup articles, instant messages, HTML documents . . . etc).

Supplemental/Alternative Compression

[0059] Various additional compression techniques may be employed, either in addition to or as an alternative to the state-based compression techniques just described.

[0060] In one embodiment of the invention, common characters and strings of characters (i.e., which are frequently transmitted between the wireless device **130** and the interface **100**) are encoded using relatively small code words whereas infrequent characters or strings of characters are encoded using relatively larger code words. In order to encode data in this manner, a statistical analysis is performed to identify common character strings. Based on the statistical analysis, a lookup table similar to the one illustrated in FIG. **7** is generated and maintained at both the wireless device **130** and the interface **100**. As illustrated, certain character strings such as the domain used for corporate e-mail "@good.com" and the first **6** digits of the corporate telephone number, e.g., "(408)720-" may be quite common. As such, replacing these common bit strings with relatively small code words may result in a significant amount of compression. Referring back to messages **300-302**, using this compression technique, the domain "@good.com" encountered numerous times in each message header could be replaced by a short, several-bit code word.

[0061] In one embodiment, a different look up table may be generated for different types of data transmitted between the interface **100** and the wireless data processing device **130**, resulting in greater precision when identifying common strings of characters. For example, a different set of code words may be used to compress e-mail messages than that used to compress the corporate address book. Accordingly, the code word table used to compress e-mail messages would likely contain relatively small code words for the most common e-mail domains whereas the corporate address book might also contain relatively small code words for the corporate address and portions of the corporate phone number.

[0062] Moreover, in one embodiment, a unique code word table may be generated for each field within a particular type of data. For example, a different code word table may be employed for the e-mail header field than that used for the remainder of the e-mail message. Similarly, a different table may be generated for the "address" field of the corporate address book than that used for the "e-mail address" field, resulting in even greater precision when generating the set of code words.

[0063] Rather than statistically generating and transmitting a code word table for each field, alternatively, or in addition, one embodiment of the invention refers to a dictionary of "known" words, like an English dictionary, and therefore does not need to transmit the dictionary with the data. For example, in one embodiment, a spell-check dictionary maintained on the wireless device **130** and/or the interface **100** may be used to compress content. Rather than sending the actual text of the e-mail message, each word in the message would be identified by its entry in the spell-check dictionary (e.g., the word "meeting" might be replaced by entry#3944).

[0064] One type of data particularly suitable to the foregoing types of compression is the corporate address book maintained on most corporate e-mail servers. In one embodiment of the invention, the corporate address book is synchronized initially through a direct link to the client **110** (see FIG. **1**). On the initial synchronization (e.g., when the wireless device is directly linked to the client **110**), statistics on common letters and "tokens" (e.g., names, area codes, e-mail domains) are generated. The statistics and tokens are then used to compress the data as described above. Thereafter, any changes to the address book are wirelessly transmitted. On subsequent updates, the compressors on both sides (wireless device **130** and interface **100**) would refer to the earlier statistics gathered, and thus compress without any new statistics or words being transmitted.

[0065] The updates may represent a small percentage of the entire address book, but may still represent a significant number of bytes, especially when multiplied by all the wireless devices in use in use at a given company. Accordingly, reducing the amount of data required to transmit the updates to the address book as described above, would result in a significant savings in transmission costs. Additionally, as the address book can be very large relative to the storage available on the client, storing the address book on the client in a compressed form will allow more entries to be stored.

[0066] In one embodiment, to conserve additional space, only certain fields of the corporate address book will be synchronized wirelessly. For example, only the Name, Address, E-mail, and Phone Number fields may be updated wirelessly. All fields of the address book may then be updated when the wireless device is once again directly linked to the client **110**.

[0067] One embodiment of a method for generating a code word table is illustrated in FIG. **8**. At **810**, occurrences of certain byte strings are calculated for use by a standard Huffman compression algorithm. At **820** certain "tokens" are generated for a particular field based on the natural boundaries for that field type. For example, as described above, e-mail addresses could be broken into ".com" and "@good.com" as described above for e-mail fields. Phone numbers might be broken into "(650)" and "(650)620-" for address book fields.

[0068] At **830** the occurrences of tokens are counted in the same way as the occurrences of the byte strings are counted, though one occurrence of, say, a four-byte token adds four to the count. At **840** a code word table of all the letters and those tokens that occur more than once (or maybe the top N tokens that occur more than once) is generated. Part of the table will include the tokens themselves. At **850**, each record is compressed using the code word table of characters and tokens and, at **860**, the code word tables and the compressed records are then sent to the wireless device **130**.

[0069] In one embodiment, the code word tables are identified with a unique number, such as a timestamp. Both the interface **100** and the wireless device **130** would store the tables. On the wireless device **130**, the records may remain compressed to conserve space, being decompressed only when opened. On subsequent syncs, the wireless device **130** may request updates to the corporate dictionary. As part of the request, the wireless device **130** may include the unique number assigned to the code word tables. If, for some reason, the wireless device **130** doesn't have the original

tables, it may send a particular type of ID to notify the interface **100** (e.g., by using a "0" for the ID). Likewise, if the host doesn't recognize the ID for some reason, it can ignore the original tables and create new ones.

[**0070**] In most cases, however, the wireless device **130** and interface **100** will agree on what the ID is, and the compression of the update will use the existing code word tables previously computed. For example, a new employee with the same e-mail domain and phone prefix as existing employees would compress nicely. Since the updates should be a small percentage of the overall address book, it will most likely be very similar to the existing data.

[**0071**] One embodiment of the invention converts alphanumeric characters (e.g., standard ASCII text) into a proprietary variable-bit character format, allocating relatively fewer bits for common characters and relatively more bits for uncommon characters. In one particular embodiment, 6 bits are allocated for most characters, and 12 bits are allocated for all other characters. This embodiment may be seamlessly integrated with the other forms of compression described above (e.g., message pointer generation, code word lookups, . . . etc) through an escape function described below.

[**0072**] Most messages will have ASCII text in them. For example, the TO: field in an e-mail, or the name in an Address Book entry are generally comprised of ASCII text. Most ASCII text use 7 bits/character. Typical exceptions are accented characters, like ñ or ö. Realistically, though, most text in a text field consists of a-z, 0-9, space, and a few symbols.

[**0073**] Compressing text using code word tables as described above is a good way to encode large amounts of text, because it gathers statistics about how frequently a given character occurs, and represents more frequent characters in fewer bits. For example, the letter 'e' occurs more often than the letter 'k', so it may be represented in, say, 3 bits. It is also particularly suitable for compressing data in specific data fields where it is known that the same character strings appear regularly (e.g., such as the e-mail domain "@good.com"). One problem with this technique, however, is that it requires transmitting and storing the statistical information with the encoded text. For small amounts of text (e.g., short e-mail messages), this becomes impractical.

[**0074**] A 6-bit character format provides for 64 characters (2^6=64). In one embodiment, the following symbols are encoded using 6-bits: a zero, handy for denoting the end of strings; 'a' through 'z;''0' through '9;' space; and the most common symbols (e.g., dot, comma, tabs, new-lines, @, parens, !, colon, semicolon, single, double quotes, . . . etc). The values above account for 48 of the 64 values, leaving 16 values remaining.

[**0075**] In one embodiment, the remaining **16** values are used for the following escape values:

[**0076**] (1) Four values for combining with the next 6-bits to allow any possible ASCII value to be encoded in two 6-bit values. It allows for any upper case letter, symbols not in the top ten, accented characters, and so on. For example, binary values of 60, 61, 62, and 63 may each identify another 6-bit value which contains the underlying character information. This provides for the coding of an additional 256 characters (4*64=256), more than enough to encode the entire US-ASCII character set.

[**0077**] (2) Shift Lock. Turns on shifting until a subsequent Shift Lock turns off shifting. For letters, this is like a caps lock. For numbers and symbols, this may have no effect. Alternatively, a second set of values may be defined when shift lock is on (e.g., a second "top ten" list of symbols).

[**0078**] In one embodiment, the remaining 11 6-bit characters are "installable escape values," allowing one or more standard or custom compressors. For example, the TO:, FROM:, CC:, and BCC: fields in an e-mail all contain a list of e-mail addresses, separated by a semicolon. As such, the following special escape values may be defined: (1) the customer's/user's e-mail address may be converted into a 6-bit value; (2) the customer's/user's domain may be converted into a 6-bit value (e.g., "@Good.Com" would become 6 bits); (3) "common" domain names and suffixes may be converted into a 6-bit value and a 6-bit argument (e.g., the "common" list may be 64 of the most common names, and might include "@aol.com", "@webtv.com", ".com", ".net",".org", ".gov", ".us", ".uk", . . . etc); and (4) names "used recently" in an e-mail may be converted into a 6-bit value and a 6-bit argument. Elsewhere in the message is the e-mail ID this is dependent on. The argument might include 2 bits identifying the field (TO:, FROM:, CC:, or BCC:), and 4 bits identifying the first 16 e-mail addresses in that field.

[**0079**] The new character format may be employed seamlessly with the other types of compression described above (e.g., code words, repeated characters; LZ compression; dictionary lookups; and/or referring to prior messages). In one embodiment, illustrated in FIG. **9**, a text compression module **900** compresses text according to the 6-bit character format described above and coordinates compression functions between various other compression modules. In the illustrated embodiment, this includes a state-based compression module **910** for compressing messages by referring to prior, cached messages (as described above) and a code word compression module **920** which compresses common character strings using code words (e.g., by encoding statistically-analyzed tokens, referring to a spell-check dictionary, . . . etc, as described above). In addition, as indicated by alternative compression module **930**, various other types of compression may be employed on the system to attain an even greater level of compression (e.g., standard LZ compression).

[**0080**] FIG. **10** illustrates an exemplary portion of e-mail message **302** (from FIG. **3***c*) encoded according to this embodiment of the invention. Starting from the upper right corner of the e-mail message **302**, the text compression module **900** begins encoding the first set of characters (i.e., starting with the addressee field "TO:"). With each character it coordinates with the other compression modules **910**, **920**, **930** to determine whether those modules can achieve greater compression. If not, then the text compression module **900** encodes the text according to the **6**-bit character format. If a higher level of compression can be achieved with one of the other compression modules **910**, **920**, **930**, however, the text compression module **900** hands off the compression task to that module and inserts an "escape" sequence of bits indicating where the compression task was accomplished by that module.

[**0081**] For example, as illustrated in FIG. **10**, the escape sequence "110010" following the first three characters

("TO:") indicates that the code word generation module **920** compresses the subsequent portion of data. In operation, once this point in the e-mail message is reached, the code word generation module **920** notifies the text compression module **900** that it can achieve a higher level of compression using code words (e.g., using a tokenized e-mail address). Accordingly, the sequence "1011001000" following the escape sequence "110010" is a code word representing the tokenized e-mail address "Collins, Roger" <rcollins@good.com>. Alternatively, two or more code words may be used to encode the e-mail address, depending on the particular set of code words employed by the system (e.g., one for the individual's name and a separate one for the domain "@good.com"). As indicated in FIG. **10**, the text compression module **900** may then pick up the encoding process following the tokenized e-mail address (i.e., the return character followed by the text "FROM:").

[**0082**] After the e-mail header information is encoded, the block of new text **355** is encoded using the **6**-bit character format. Of course, depending on the code words employed by the code word generation module **920** and/or previous e-mails on the system, portions of the block of new text **355** may also be encoded using code words and/or pointers to previous messages. Following the text block **355**, the state-based compression module **910**, after analyzing the message, notifies the text compression module **900** that it can achieve a higher level of compression by identifying content found in a previous message. As such, an escape sequence "110011" is generated indicating that compression is being handled by the state-based compression module **910** from that point onward. The state-based compression logic **910** then identifies a previous e-mail message using a message ID code (indicating message **301**), and generating an offset and a length indicating specific content within that e-mail message (e.g., employing one or more of the state-based compression techniques described above).

[**0083**] It should be noted that the specific example shown in FIG. **10** is for the purpose of illustration only. Depending on the code words employed by the system and/or the previous messages stored on the system, the actual encoding of the e-mail message **302** may turn out to be different than that illustrated. For example, as mentioned above, the block of text **355** may be encoded using code words and/or pointers to previous messages as well as the 6-bit character format.

[**0084**] Various supplemental/alternative compression techniques may also be employed (e.g., represented by alternate compression module **930**). In one embodiment, certain types of data are not transmitted wirelessly between the wireless data processing device **130** and the interface **100**. For example, in one embodiment, when a device has been unable to receive messages for a certain period of time (e.g., one week), only message headers are initially transmitted to the device **130**, thereby avoiding an unreasonably long download period (i.e., wherein all messages received over the period of unavailability are transmitted to the device). Alternatively, or in addition, in one embodiment, when the device is out of touch for an extended period of time, only relatively new messages (e.g., received over a 24-hour period) are transmitted to the device when it comes back online. Similarly, in one embodiment, only e-mail header information is transmitted to the wireless device **130** (e.g., indicating the subject and the sender) when the user is a CC addressee and/or when the e-mail is from a folder other than the user's inbox.

[**0085**] In one embodiment, only certain fields are updated on the device **130**. For example, with respect to a corporate or personal address book, only Name, E-mail Address and Phone Number fields may be synchronized on the device **130**. When the device is connected directly to the client, all of the fields may then be updated.

[**0086**] In one embodiment, certain details are stripped from e-mail messages to make them more compact before transmitting them to the device **130**. For example, only certain specified header information maybe transmitted (e.g., To, From, CC, Date, Subject, body, . . . etc). Similarly, the subject line may be truncated above a certain size (e.g., after 20 characters). Moreover, attachments and various formatting objects (e.g., embedded pictures) may not be transmitted. In one embodiment, when a user lists him/herself as a CC addressee on an outgoing message, this message will not be retransmitted back to the wireless device **130**.

[**0087**] Although attachments may not be transmitted to the wireless device **130**, in one embodiment, users may still forward the attachments to others from the wireless device (the attachments will, of course, be stored on the e-mail server). Moreover, in one embodiment, attachments may be sent to a fax machine in response to a user command from the wireless device **130**. Accordingly, if a user is away from the office and needs to review a particular attachment, he can type in the number of a nearby fax machine and transmit this information to the interface **100**. The interface **100** will then open the attachment using a viewer for the attachment file type (e.g., Word, Power Point, . . . etc) and transmit the document via a fax modem using the fax number entered by the user. Thus, the user may view the attachment without ever receiving it at the device.

Batch Processing of Message Transactions

[**0088**] As illustrated in FIG. **11**, under certain conditions, maintaining complete synchronization between the device **130** and service **102** may consume a significant amount of wireless bandwidth. For example, if a user has been out of range for an extended period or time (e.g., the device is turned off) a plurality of messages may be transmitted in succession from the interface **100** to the wireless device **130** when the device is back within range. In some cases, of course, the user may not necessarily be out of range at all. Rather, the user may simply receive/ transmit a significant number of e-mail messages is succession.

[**0089**] As illustrated, once the user begins viewing messages on the device **130**, message transaction updates are continually sent to the interface **100**. For example, when the user reads message **1**, an indication that the message was read is transmitted to the interface **100**. This may be followed by an acknowledgement from the interface **100** (e.g., indicating that the communication was received). Similarly, when the user reads and then deletes message **2**, separate indications that the message was read and then deleted are transmitted to the interface **100**, respectively, followed by an acknowledgement for each transaction.

[**0090**] Because each individual data transmission between the device **130** and the interface **100** may include a signifi-

cant amount of overhead (e.g., header information such as the device address **130**, the service address **102** and various other types of header/control information), and because each message may require an acknowledgement from the interface **100**, synchronizing messages in this manner may consume a significant amount of bandwidth. Put another way, the ratio of actual data (e.g., database updates) to control data (e.g., header data) will be relatively low. Moreover, continual data transmissions of this type will tend to consume significantly more power (e.g., because the device's radio will not be idle long enough to enter its low-power mode).

[**0091**] Accordingly, in one embodiment of the invention, under certain conditions (described below), data transactions between the device **130** and the interface **100** are combined, or batch-processed to conserve bandwidth. For example, as illustrated in FIG. **12**, in this embodiment, a plurality of message transactions are performed on the data processing device before the device is synchronized with the service **102**. Subsequently, a single transmission **1201** containing all of the synchronization updates (e.g., message viewings and deletions, message responses, . . . etc) is transmitted to the interface **100**, followed by a single acknowledgement **1202** that the update was received.

[**0092**] Similarly, under certain conditions, database modifications at the service **102** may be batch-processed before being transmitted to the device **130**. For example, if the user is in the office reading through and responding to a series of e-mail messages (e.g., from the client **110**), transmitting each message transaction to the wireless device **130** independently of one another may not be efficient for the reasons set forth above. As such, in one embodiment, these transactions (or a subset thereof) are combined and concurrently transmitted to the wireless device **130**.

[**0093**] As indicated in FIG. **12**, the specific conditions under which batch-processing is initiated and (once initiated) the specific manner in which the messages are combined may be based on processing parameters **1210**, **1220** configured in the wireless device **130** and/or the interface **100**, respectively. For example, in one embodiment, batch processing will be triggered if the user has not checked messages for an extended period of time (e.g., two days). In this case, it is expected that once the user begins to check messages he/she will perform a significant number of message transactions within a relatively short period of time. It should be noted, however, that various different batch-processing triggers may be employed while still complying with the underlying principles of the invention (e.g., two or more successive message transactions within a predetermined period of time, manual triggering set by the end user, . . . etc).

[**0094**] Once batch-processing is triggered, message transactions occurring over periodic intervals (e.g., every **10** minutes) may be combined and transmitted at the end of each interval. Alternatively, or in addition, once the combined message transactions reach some predetermined threshold (e.g., based on the sheer number of transactions and/or the amount of data contained within the combined transactions), the combined messages may be transmitted together. Various other message combination parameters may be employed while still complying with the underlying principles of the invention.

[**0095**] One embodiment of a method for performing batch processing of message transactions is illustrated in FIG. **13**. At **1301**, current message transaction conditions are evaluated (e.g., the frequency with which message transactions are performed, when the last message transaction was initiated, . . . etc). At **1305** it is determined whether the current conditions match the threshold conditions required for batch processing. For example, as described above, if the user's wireless data processing device **130** has been out of range for a predetermined period of time and/or if the user has not checked his e-mail for a period of time, the batch processing mode may be invoked.

[**0096**] If the conditions are not met, then at **1310**, the system remains in standard message transaction mode. If, however, the conditions have been met, then at **1315**, the system (i.e., the wireless device **130** and/or interface **100**) processes messages according to the established batch-processing parameters. For example, at this stage the device **130** and/or interface **100** may combine message transactions which occur over a predetermined period of time (or which result in a specified number of transactions or amount of data as described above).

[**0097**] At **1325** it is determined whether the standard message processing conditions have once again been met. For example, if the user's data processing device has been in range for a predetermined period of time after entering the batch-processing mode, and the user is periodically receiving and quickly responding to messages, this may cause the system to revert back to the standard message transmission mode. Depending on the system configuration, various additional/alternative conditions may cause the system to enter its standard message processing mode.

Multi-Level Batch Processing

[**0098**] In one embodiment of the invention, two levels of batch processing are employed: one at the customer site **120** and another at a data center located on the external data network **170**. This embodiment will be described with respect to FIG. **14** which shows a data center **1410** communicatively coupled to the customer site via an outbound gateway **1413** and to the wireless network **171** via a wireless gateway **1411**.

[**0099**] Batch processing logic **1400** at the customer site provides the first level of batch processing. Specifically, in one embodiment, when a user concurrently performs a group of message transactions, the batch processing logic **1400** logically combines the message transactions before transmitting them to the data center **1410**. For example, when a user deletes a block of e-mail messages or moves a block of messages from one folder to another, the block of individual deletions/moves are transmitted as a group (i.e., as opposed to transmitting a series of individual deletes/moves and waiting for an equal number of individual acknowledgements form the data center **1410**). In addition; the block of message transactions are temporarily stored off in the remote message cache **200** (described above with respect to FIG. **2**), or in an alternate cache at the customer site.

[**0100**] At the data center **1410**, the batch-processed message transactions are initially stored off in a secondary cache, referred to herein as a "message switch"**1412**. After receiving and storing the block of message transactions, the

message switch sends a block acknowledgement to the batch processing logic **1400**, which may thereafter delete the block of message transactions from the remote message cache **200**. Alternatively, the batch processing logic **1400** may continue to store the block of message transactions for some predetermined amount of time or after some predetermined event has occurred (e.g., until it receives an indication that the message transactions have been successfully received by the wireless device **130**).

[**0101**] If the wireless device is actively connected to the wireless network, the message transactions are forwarded from the message switch **1412** to the wireless device as a group (via the wireless gateway **1411**). For example, an indication that **10** messages have been moved from the user's "inbox" to the user's "saved mail" folder may be transmitted together. The wireless device **130** may then respond with a single acknowledgement that it received all **10** message transactions. Alternatively, if one of the message transactions had not been successfully received, the wireless device **130** may request that individual message transaction as opposed to the entire group (as described in detail below in the section entitled "In-Order Delivery of Message Transactions").

[**0102**] In one embodiment, the message switch **1412** performs a second level of batch processing functions in addition to (or in lieu of) the first level of batch processing performed by the batch processing logic **1400** at the customer site. Specifically, the message switch **1412** batch-processes sequences of message transactions generated over a period of time as opposed to the bulk message transactions (e.g., "delete 10 messages") just described. For example, a user will typically read one new e-mail message after another at the customer site, and may continually add new to-do list entries and calendar entries throughout the day. In one embodiment, these individual message transactions are transmitted from the interface **100** to the message switch **1412** as they occur at the service **102**. For example, when a user reads a single new e-mail message, an indication that the message has been read is transmitted to the message switch **1412**. Similarly, when a user generates a new calendar entry, the new entry is automatically transmitted to the message switch **1412**.

[**0103**] In one embodiment, the message switch **1412** groups the various individual message transactions together before transmitting them to the wireless device **130**. If the wireless device **130** is actively connected to the wireless network, the message switch **1412** may group a certain number of message transactions together and/or may group all message transactions together occurring over a period of time before transmitting them as a group to the wireless device **130**. While the wireless device **130** is not actively communicating over the wireless network, the message switch **1412** may combine all message transactions and transmit them as a group once the wireless device comes online. In one embodiment, the message switch **1412** and/or the batch processing logic **1400** may batch-process message transactions based on the batch processing parameters **1210** and **1220** described above with respect to FIG. **12**.

In-Order Delivery

[**0104**] In order to fully synchronize a wireless device **130** with a service **102** as described herein, it is not only important that message transactions are reliably communicated to and from the wireless device but also that the message transactions are communicated in the proper order (e.g., in the same sequential order in which they occur at the service). For example, if a user creates a new folder at the service **102** and then moves several messages into the folder, the transaction creating the folder must be received by the wireless device before the move transaction.

[**0105**] While wireless networks such as Mobitex ensure reliable delivery of data, they do not necessarily ensure that the delivered data arrives in-order. Moreover, while network protocols such as the Transmission Control Protocol ("TCP") ensure in-order delivery of data, these protocols assume that both the sending node and the receiving node are always active and, therefore, are not necessarily adapted to a system in which one of the nodes (i.e., the wireless device) is inactive for extended periods.

[**0106**] As such, one embodiment of the invention illustrated in FIG. **14**, employs in-order control logic **1500**, **1510** and **1520** at the customer site, the data center and/or the wireless device, respectively, to ensure in-order delivery of message transactions. In operation, each message transaction at the customer site is assigned a sequential code which indicates the relative order in which the message transaction was generated. In one embodiment, when a series of message transactions are transmitted to the wireless device **130** (or transmitted from the wireless device **130** to the interface **100**), the wireless device **130** (or the interface **100**) will not execute a particular message transaction until it has received all previous sequential message transactions. Thus, if the wireless device **130** receives a series of message transactions coded sequentially from **1** to **3** and from **5** to **10**, it may execute message transactions **1** to **3** but will not execute message transactions **5** to **10** until it receives message transaction **4**.

[**0107**] In one embodiment, if the wireless device has not received message transaction **4** after some specified period of time (e.g., because the message transaction was lost during transmission), the wireless device **130** will send a request to the data center **1410** and/or the interface **100** to retransmit message transaction **4**. The in-order control logic **1500** or **1510** executed at the interface **100** and/or the data center **1410**, respectively, will then retransmit message transaction **4** from either the remote message cache **200** or the message switch **1412**, respectively.

[**0108**] The wireless device **130** notifies the interface **100** and/or the message switch **1412** once it successfully receives message transaction **4**, thereby allowing the message transaction to be removed from the remote message cache **200** and/or the message switch **1412** (i.e., assuming that other cache removal conditions described herein have been met). In one embodiment, the wireless device may send a block notification as opposed to an individual notification for each message transaction. For example, rather than simply sending a notification that it has received message transaction **4**, the wireless device **130** may send a single notification that it has successfully received messages **1-10** (or some alternate number of message transactions), thereby allowing all messages to be cleared from remote message cache **200** and/or the message switch **1412** with a single notification. It should be noted that the sequential transaction numbers set forth above are for the purpose of illustration only. Various

alternate sequential codes may be employed to indicate message transaction order while still complying with the underlying principles of the invention.

Identification Code Allocation

[0109] Each e-mail message, calendar entry, to-do list entry, . . . etc, is assigned a unique identification code by the service **102**. For example, if the service is Microsoft Exchange, a 128-byte identification code is generated for each new data object. Accordingly, when fully synchronizing a wireless device **130** to the service **102**, some mechanism must be provided to ensure that no duplicate identification codes are assigned for two distinct data objects. For example, if both the service **102** and the wireless device **130** are capable of independently generating data objects, they may both concurrently generate data objects with the same identification codes, resulting in a conflict.

[0110] One mechanism for solving this problem is to require the wireless device **130** to request a new identification code from the service **102** each time it generates a new data object. One potential problem with this scenario is that it may take an unreasonably long time for the wireless device **130** to acquire the identification code, depending on the speed of the wireless network. For example, several seconds may be considered an unreasonable amount of time to wait to begin entering a new e-mail message or calendar entry.

[0111] Alternatively, in one embodiment, the range of all possible data object codes are divided between the wireless device **130** and the service **103**. In other words, a certain percentage (e.g., ½) of all possible codes are allocated to the wireless device **130** and the remaining possible codes are allocated to the service **103**. In operation, when a new data object is generated at the wireless device (e.g., a new "to-do" list entry) the wireless device **130** will select a data object code only from within its pre-assigned range, thereby preventing a conflict at the service **102**. In one particular embodiment, all negative codes are assigned to the wireless device **130** and all positive codes are assigned to the service **102**. If a 32-bit (4-byte) code is used, this will result in 2,147,483,648 (2^{31}) negative codes and 2,147,483,648 (2^{31}) positive codes. It should be noted, however, that the particular manner in which codes are divided up is not pertinent to the underlying principles of the invention.

[0112] Another potential problem which exists when fully synchronizing a wireless device with a service is that the standard data object identification codes employed by many services are unnecessarily large. As mentioned above, Microsoft Exchange generates a 128-byte (1024 bit) code to identify each unique data object.

[0113] Accordingly, in one embodiment of the invention illustrated in FIG. **16**, the interface **100** includes object identification code mapping logic **1600** for mapping standard data object identification codes **1620** (e.g., such as the 128-byte codes used by Microsoft Exchange) to data object identification codes **1610** generated specifically for use in the synchronization system described herein (hereinafter "synchronization system identification codes"). As illustrated, object identification code mapping logic **1600** maintains a data object identification table **1605** in which each standard identification code **1620** is associated with a corresponding synchronization system identification code **1610**. As described above, in one embodiment, the synchronization system identification codes **1610** are **32**-bits in length, thereby significantly reducing the amount of information transmitted across the wireless network. In addition, as indicated in FIG. **16**, negative identification codes **1610** identify data objects created by the wireless device **130** and positive identification codes **1610** identify data objects created at the service **102** (e.g., from a local desktop PC).

Data Object Conflict Resolution

[0114] Because copies of data objects may be maintained at both the wireless device **130** and on the service **102**, one embodiment of the invention employs techniques to ensure that concurrent modifications to the same data object at both the wireless device **130** and the service **102** are resolved in a logical manner. For example, in one embodiment, a version number is associated with each data object. Each time the data object is modified, the version code is changed to indicate the new version.

[0115] In one embodiment, illustrated in FIG. **17**, the interface **100** and/or the wireless data processing device **130** includes conflict detection logic **1700** and **1701**, respectively, for detecting when a version conflict has occurred and conflict resolution logic **1710** for implementing one or more predefined conflict resolution techniques to resolve the version conflict. By way of example, in FIG. **17**, a copy of Data Object X, Version **1** is initially stored on both the wireless device **130** and the service **102**. Version **1** may be, for example, the initial version of a calendar entry or a to-do list entry. Both copies of Data Object X, Version **1** are concurrently modified at the service **102** and the wireless device **130** to generate Versions $\mathbf{2}_1$ and $\mathbf{2}_2$, respectively, producing a version conflict. One way in which this may occur is that a user modifies Data Object X at the wireless device **130** at the same time as the user' administrative assistant modifies Data Object X at the service **102**. The wireless device **130** subsequently attempts to update the service **102** with Version $\mathbf{2}_2$ and, likewise, the service **102** attempts to update the wireless device **130** with Version $\mathbf{2}_1$.

[0116] In one embodiment, the conflict detection logic **1700**, **1701** executed on the interface **100** and/or the wireless device **130**, respectively, detects the version conflict. In response, the conflict detection logic **1700**, **1701** triggers conflict resolution logic **1710**, **1711** which attempts to resolve the conflict by applying one or more conflict resolution techniques. Various techniques may be employed to resolve the conflict. For example, in one embodiment, the version of the data object at the service **102** (Version $\mathbf{2}_1$) is automatically retained, and the user is notified that his modification of the data object from the wireless device **130** will not be entered. The notification may be accompanied by a visual indication of the new version (Version $\mathbf{2}_2$) and/or an explanation as to why the modification will not be entered. Alternatively, in one embodiment, the user may be prompted from the data processing device to select between the two potential versions. Upon making a selection, the selected version will be stored on both the wireless device **130** and the service **102**. If another individual attempted to enter the non-selected version (e.g., the user's administrative assistant), then that individual may subsequently be notified. In one embodiment, the version which is selected is based on who entered it. For example, one embodiment of the invention may be configured to always accept the version gener-

ated by the user (i.e., and not the user's administrative assistant). Thus, if the user modifies Data Object X from either the wireless device **130** or directly at the service **102** (i.e., from a desktop connected to the service **102**), the user's modifications will be accepted over any other modifications.

[**0117**] It should be noted that the specific conflict resolution techniques described above are for the purpose of illustration only. Various additional conflict resolution techniques may be employed by the conflict resolution logic **1710**, **1711** while still complying with the underlying principles of the invention.

Full Wireless Synchronization and Zero Desktop Install

[**0118**] The advanced compression and message processing techniques described above allow the wireless device **130** to be fully synchronized with the service **102**. For example, in one embodiment of the invention, all major components of the messaging service are completely synchronized on the wireless device **130**. For example, if the service is Microsoft Exchange, these components will include e-mail, electronic calendar, contacts, tasks and notes. Accordingly, all user transactions (message filings, to-do list entries, . . . etc) are maintained up-to-date on the wireless device without the need for a cradle.

[**0119**] In one embodiment, not only are messages synchronized, but the entire state of the service **102** may be synchronized. This state information may include, for example, the creation of new folders, the deletion of old folders, filing of messages to folders, reading a message from the device, marking a message unread, e-mail deletions, arrival of new messages, copying of messages to a folder, filing of messages and/or any other transaction which has an effect on the mailbox maintained at the service.

[**0120**] In addition, in one embodiment, the wireless device **130** is provisioned wirelessly. Thus, once a user's account has been enabled on the service, all initial user data is sent wirelessly. This data may include, for example, initial contacts (e.g. address book), notes, tasks and calendar data. In one embodiment, a unique encryption key may initially be installed on the wireless device **130** to encrypt communication between the device and the service (e.g., by device installation software).

[**0121**] In one embodiment, even though the data on the wireless device **130** is completely synchronized, an aging algorithm may be employed to conserve space on the device. For example, at a given point in time, the service may be storing 40,000 data objects (e.g., e-mail messages, calendar entries, . . . etc), whereas, the wireless device (having a limited amount of memory) may only be capable of storing 20,000 data objects. Accordingly, in one embodiment, the wireless device **130** will store data objects which have not been modified or otherwise manipulated (e.g., moved from one folder to another) for the longest period of time. In one embodiment, the user may specify which types of messages should be automatically deleted (e.g., only messages in the "sent mail" folder, any messages over 1 month old, etc). Once a message has been removed from the device, however, it may always be recovered from the service.

[**0122**] For example, a user may request certain data objects to be re-transmitted from the service **102** based on one or more specified variables (e.g., creator, client, sender, recipient, date range, . . . etc). Similarly, in one embodiment, if the user manipulates a data object which has been deleted from the wireless device **130** from the user's desktop (e.g., moves an e-mail message from one folder to another) that data object will be re-transmitted to the wireless device and stored in the destination folder.

[**0123**] One embodiment of the invention maintains synchronization events even if any part of the system is "down" (e.g., the data network **170** and/or the wireless service provider network **171**). For example, as described above, any synchronization events which occur during system downtime may be maintained in one of the batch processing caches **1412** or **200** at the data center **1410** and/or the interface **100**, respectively. Thus, the interface **100** may be down for a period of time, the data network **170** may be unavailable, the wireless device **130** may be off, out of coverage or broken, and synchronization updates will still be maintained. Once all parts of the system are again working properly, the queued synchronization updates are processed.

[**0124**] In one embodiment of the interface **100**, "move" events are detected and processed in an efficient manner. As indicated in FIG. **18** between the service **102** and the interface **100**, when a message (or other data object) is moved from one folder to another on messaging systems such as Microsoft Exchange (e.g., from "sent mail" folder to a "saved mail" folder, from the "inbox" folder to a "read mail" folder, . . . etc), a new copy of the message is made in the location of the destination folder and the original message is then deleted from the source folder. Alternatively, the message may initially be deleted from the source folder and then re-created in the destination folder. Transmitting a delete command followed or preceded by a copy of the underlying message to the wireless device **130** is an inefficient way to perform move transactions. Accordingly, as indicated in FIG. **18**, one embodiment of the interface **100** combines the "delete" and the "new" commands into a single "move" command using the data object (i.e., message) identification code, the source folder and/or the destination folder, thereby significantly reducing the amount of information transmitted across the wireless network.

[**0125**] In order to provide a move command to the wireless device **130** in this manner, the system (e.g., the interface **100**) must first identify the message which is to be moved. One embodiment of the interface identifies the message using the methods set forth in FIG. **19***a* and/or FIG. **19***b*, either alone or in combination. Referring initially to FIG. **19***a*, at **1900** the interface **100** detects that Message X has been deleted from Folder A. At **1910**, the interface **100** attempts to determine if the deletion forms part of a move command. As such, it searches other folders in the user's account to locate the same message. If it finds the same message in a particular folder, e.g., Folder B, it transmits a move command to the wireless device **130** at **1930** indicating that Message X should be moved from Folder A to Folder B. If, however, it does not locate Message X in another folder, it transmits a delete command to the wireless device indicating that Message X should be deleted from Folder A.

[**0126**] Referring now to FIG. **19***b*, in one embodiment, the interface **100** initially detects that Message X has arrived in Folder B. In response, the interface **100** searches the table of

data object identification codes **1605** (see, e.g., FIG. **16**) to locate a match for the identification code associated with the Message X. If a match is found (determined at **1970**), then the interface **100** transmits a move command to the wireless device **130** indicating that Message X should be moved from Folder A to Folder B. If, however, the interface **100** does not locate an identification code match, it transmits a delete command to the wireless device indicating that Message X should be deleted from Folder A.

[**0127**] When a wireless device **130** has been "out-of-touch" with the service **102** for an extended period of time, a significant number of transactions may have accumulated which need to be synchronized. Accordingly, in one embodiment, in the interest of both saving bandwidth and time on the device (e.g. not swamping it with unsynchronized data), only representative portions of some data may be transmitted. For example, if the wireless device **130** has been off for two weeks, only message headers may be transmitted to the device (i.e., not the message bodies). The underlying reason for this is that the user will not likely want/need to read all of the older mail on the device.

[**0128**] In one embodiment, the specific manner in which data is transmitted to the device after an extended period of time may be selected by the user. Thus, the user may select a period of time after which only headers should be sent (e.g., older than 1 week, never, etc). In any case, the user may still request the full messages bodies after the headers have been transmitted.

[**0129**] As used herein, "zero desktop install" refers to the ability of the wireless device **130** to function normally without the installation of any client software on a user's desktop computer. One embodiment of the invention does not require a desktop because, as described above, all messaging features (e.g., management of device options, configuration of the messaging service, message filters, outgoing e-mail signatures, security settings, . . . etc) may be accessed by the wireless device. This feature is not available in current messaging systems because current wireless devices support only a subset of all messaging functions. As such, current systems require desktop software and a cradle to complete the synchronization process.

[**0130**] In one embodiment, the wireless device's configuration settings are stored and continually updated on the messaging server. Accordingly, if the device settings are ever lost (e.g., because the device is initialized or lost) the settings may be automatically recovered along with the messaging data. In fact, in one embodiment, the device does not ever need to be backed up because there is no data unique to the device that isn't synchronized to the messaging server.

[**0131**] In addition, in one embodiment, software upgrades are transmitted wirelessly to the device, thereby completely removing any required link between the device and a desktop. Software upgrades may include upgrades to the device's operating system as well as application installations.

[**0132**] Embodiments of the invention may include various steps as set forth above. The steps may be embodied in machine-executable instructions. The instructions can be used to cause a general-purpose or special-purpose processor to perform certain steps. Alternatively, these steps may be performed by specific hardware components that contain hard wired logic for performing the steps, or by any combination of programmed computer components and custom hardware components.

[**0133**] Elements of the present invention may also be provided as a machine-readable medium for storing the machine-executable instructions. The machine-readable medium may include, but is not limited to, floppy diskettes, optical disks, CD-ROMs, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, magnetic or optical cards, propagation media or other type of media/machine-readable medium suitable for storing electronic instructions. For example, the present invention may be downloaded as a computer program which may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., a client) by way of data signals embodied in a carrier wave or other propagation medium via a communication link (e.g., a modem or network connection).

[**0134**] Throughout the foregoing description, for the purposes of explanation, numerous specific details were set forth in order to provide a thorough understanding of the invention. It will be apparent, however, to one skilled in the art that the invention may be practiced without some of these specific details. For example, while illustrated as an interface **100** to a service **102** executed on a server **103** (see FIG. **1**), it will be appreciated that the underlying principles of the invention may be implemented on a single client in which the client transmits data over a network. Moreover, although described in the context of a wireless data processing device, the underlying principles of the invention may be implemented to compress data in virtually any networking environment, both wired and wireless. Accordingly, the scope and spirit of the invention should be judged in terms of the claims which follow.

1-41. (canceled)

42. A system comprising:

a wireless data processing device;

a messaging service to maintain data objects on behalf of a user; and

synchronization means for maintaining synchronization of said data objects between said wireless device and said messaging service over a wireless network, said synchronization means transmitting data object transaction updates originating at said service to said wireless device as modifications occur to data objects at the service and wirelessly transmitting data object transaction updates originating at said wireless device to said service as modifications occur to data objects at the wireless device.

43. The system as in claim 42 wherein said data objects include e-mail messages.

44. The system as in claim 43 wherein said data objects include electronic calendar data.

45. The system as in claim 44 wherein said data objects includes a to-do list data.

46. The system as in claim 45 wherein said data objects include contact information.

47. The system as in claim 42 wherein said synchronization means further comprises compression means for compressing said messages and other information transmitted between said wireless device and said service.

48. The system as in claim 42 wherein said synchronization means further comprises batch-processing means for combining groups of data object transaction updates before transmitting said updates between said wireless data processing device and said messaging service.

49. The system as in claim 42 wherein said data processing device comprises control means allowing said user to modify configuration parameters of said messaging service from said data processing device.

50. The system as in claim 42 wherein said synchronization means synchronizes message transaction updates including the movement of messages between e-mail folders.

51. The system as in claim 50 wherein one of said message transaction updates comprises indications that said user has viewed a message from said wireless data processing device.

52. The system as in claim 42 wherein said synchronization means further comprises:

a first batch processing means configured at a customer site at which said service operates; and

a second batch processing means configured at a data center communicatively coupled between said wireless data processing device and said customer site.

53. The system as in claim 52 wherein said first batch processing means combines data object transactions concurrently executed by a user at said customer site before transmitting said data object transactions and wherein said second batch processing means combines data object transactions not concurrently executed by said user.

54. A system for synchronizing a wireless device with a service comprising:

an interface communicatively coupled between said service and said wireless device, said interface executed at a customer site at which said service in installed and configured to transmit data object transaction updates to said wireless device as modifications occur to data objects at the service, and to transmit data object transaction updates to said service as modifications occur to data objects at the wireless device; and

a data center communicatively coupled between said wireless device and said interface, said data center temporarily storing said data object transaction updates until said data object transaction updates are successfully transmitted to said wireless device.

55. The system as in claim 54 further comprising:

batch processing logic to group said data object transaction updates together prior to transmission to said wireless device and/or to said interface.

56. The system as in claim 55 wherein said batch processing logic further comprises:

a first level of batch processing logic implemented at said interface, said first level of batch processing logic grouping data object transaction updates concurrently executed by a user prior to transmitting said data object transaction updates; and

a second level of batch processing logic implemented at said data center, said second level of batch processing logic grouping data object transaction updates individually executed by said user prior to transmitting said data object transaction updates to said wireless device.

57. The system as in claim 54 wherein said interface combines two or more data object transactions at said service into an equivalent single data object transaction to be transmitted to said wireless device.

58. The system as in claim 57 wherein said two or more data object transactions comprise a delete command deleting a data object from a source location on said service and a new command reproducing said data object in a destination location, and wherein said single data object transaction transmitted to said wireless device is a move command.

59. A wireless data processing device comprising synchronization logic for maintaining synchronization of said data objects with a messaging service via a wireless network by transmitting data object transaction updates to said service as modifications occur to data objects at the wireless device.

60. The wireless data processing device as in claim 59 wherein further comprising compression logic for compressing said messages and other information transmitted to said service.

61. wireless data processing device as in claim 42 further comprising batch-processing logic for combining groups of data object transaction updates before transmitting said updates to said messaging service.

* * * * *