(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 April 2007 (05.04.2007)

PCT

(10) International Publication Number
WO 2007/038666 A2

(51) International Patent Classification:
G06Q 30/00 (2006.01)

(21) International Application Number:
PCT/US2006/037807

(22) International Filing Date:
27 September 2006 (27.09.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/237,270 27 September 2005 (27.09.2005) US

(71) Applicant *(for all designated States except US)*: **SIEBEL SYSTEMS, INC.** [US/US]; 2207 Bridgepointe Parkway, San Mateo, CA 94404 (US).

(72) Inventor: **DRUMMOND, Daren**; 1010 Beethoven Common, No. 100, Fremont, CA 94035 (US).

(74) Agent: **CAMPBELL, Samuel, G.**; CAMPBELL STEPHENSON ASCOLESE, LLP, 4807 Spicewood Springs Road, Building 4, Suite 201, Austin, TX 78759 (US).

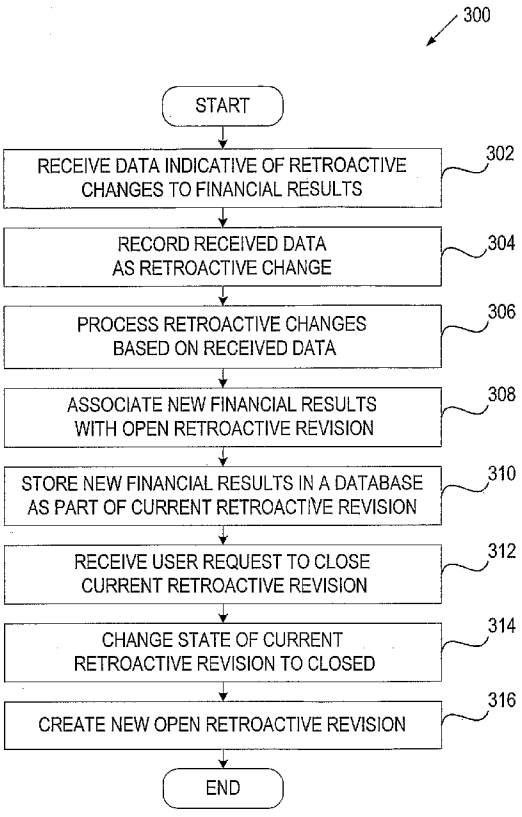
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: RETROACTIVE TRACKING AND REPROCESSING OF COMPENSATION CALCULATIONS



(57) Abstract: In one embodiment, a method for tracking retroactive changes to financial data includes receiving data indicative of one or more retroactive changes to at least one input (302), processing the retroactive changes based on the received data (306), generating new financial results, and grouping the retroactive changes and the new financial results into retroactive revisions (310).

WO 2007/038666 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

RETROACTIVE TRACKING AND REPROCESSING OF COMPENSATION CALCULATIONS

Daren Drummond

FIELD OF THE INVENTION

This invention relates generally to financial data processing, and more particularly to retroactive tracking and reprocessing of compensation calculations.

COPYRIGHT NOTICE/PERMISSION

BACKGROUND OF THE INVENTION

Sales and service organizations measure and reward high performance of their employees using various incentive compensation plans. Incentive compensation management (ICM) tools make it possible for companies to centrally manage pay-for-performance plans for their employees, contractors, brokers, suppliers, and others that need to be motivated to drive the performance of the organization.

An ICM tool may be used by various organizations. For example, an insurance company may implement an ICM tool to manage performance-based compensation of their insurance agents. An insurance agent typically receives monthly commissions for insurance policies sold during a relevant month. In addition, the insurance agent may receive a quarterly bonus based on insurance policies sold during the last quarter. However, an insured may cancel his or her insurance policy after the commissions and/or bonus have been already paid to the insurance agent. One existing ICM tool addresses such retroactive changes to incentive-based payments by writing compensation results to a log file for each payment period. Subsequently, if a retroactive change occurs, the ICM tool calculates new compensation results and writes them to a different log file for a relevant time period. A compensation administrator then compares two log files for the same time period to identify the differences. However, this approach is time-consuming and may not provide accurate results due to human errors.

SUMMARY OF THE INVENTION

The present invention relates to various aspects for tracking and reprocessing retroactive changes to financial results.

According to one aspect of the present invention, an exemplary method includes receiving data indicative of one or more retroactive changes to at least one input, processing the retroactive changes based on the received data to generate new financial results associated with the at least one input, and grouping the retroactive changes and the new financial results into a retroactive revision.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only.

Figure 1 is a block diagram of one embodiment of a financial data management system.

Figure 2 is a block diagram of one embodiment of a financial data management tool.

Figure 3A is a flow diagram of one embodiment of a process for tracking retroactive changes to financial data.

Figure 3B is a flow diagram of one embodiment of a process for tracking, reviewing, and processing retroactive changes to financial data.

Figure 4 is a flow diagram of one embodiment of a process for receiving and processing retroactive changes to financial data.

Figure 5 is a flow diagram of one embodiment of a process for executing a retroactive service batch.

Figure 6 is a flow diagram of one embodiment of a process for processing retroactive changes in a full recalculation mode.

Figure 7 is a flow diagram of one embodiment of a process for processing retroactive changes in an optimized recalculation mode

Figures 8-16 illustrate exemplary user interfaces provided by an incentive compensation management (ICM) tool.

Figure 17 is a block diagram of an exemplary computer system that may be used to perform one or more of the operations described herein.

DETAILED DESCRIPTION OF THE INVENTION

In the following description, numerous details are set forth. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In some instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable medium includes read only memory ("ROM"); random access memory ("RAM"); magnetic disk storage media; optical storage media; flash memory devices; electrical, optical, acoustical or other form of propagated signals (e.g., carrier waves, infrared signals, digital signals, etc.); etc.

Figure 1 is a block diagram of one embodiment of a financial data management system 100 of an organization. The system 100 includes a group of servers 102a through 102d, a network 106 and client devices 108. The client devices 108 may be personal computers (PCs), telephones, mobile phones, palm-sized computing devices, personal digital assistants (PDAs), fax machines, consumer electronic devices, etc. The client devices 108 are coupled to the server 102 via the network 106, which may be a public network (e.g., Internet) or a private network (e.g., Ethernet or a local area Network (LAN)). Users of the client devices 108 may be employees of the organization (e.g., participants of a compensation plan, a compensation administrator, etc.), suppliers of the organization, customers of the organization, etc.

The servers 102a through 102d host a financial data management tool 104 that is responsible for managing financial data of the organization. Collectively, the group of servers 102a through 102d may be referred to as a "cluster" of servers. In one embodiment, the server 102a operates as a controller node. The controller node 102a distributes work to the rest of the servers in the cluster. The financial data management tool 104 may be hosted on the cluster server 102a or on any of the servers 102b through 102d. The financial data may include, for example, employee compensation data, account payable data, account receivable data, etc. The tool 104 makes calculations for a current time period based on financial data. These calculations may pertain to employees' compensation (e.g., salary, commissions, bonuses, etc.), payments to suppliers, payments received from customers, etc. Sometimes input data used for calculations (e.g., financial transactions, quotas, user-defined calculation logic, etc.) may change after payments have already been made to intended recipients or received from a sender. In one embodiment, the tool 104 is responsible for identifying and reprocessing such retroactive changes, generating new financial results according to the retroactive changes, and allowing users of clients 108 to view the new financial results without destroying the previous results. In one embodiment, the previous results are maintained for historical accuracy and auditing purposes.

In one embodiment, the tool 104 groups the new financial results into a single entity referred to herein as a retroactive revision. The tool 104 maintains multiple retroactive revisions, with each retroactive revision corresponding to a specific set of retroactive input received by the tool 104. An exemplary retroactive input may be, for example, a modified commission rate applicable to commissions that have already been paid to employees or a modified bonus plan applicable to bonuses that have already been paid to employees. In one embodiment, a retroactive revision entity has a set of characteristics that will be discussed in greater detail below.

The use of retroactive revisions allows users to track, process and view changes resulting from a specific set of retroactive inputs separately from other revisions of the organization's financial data.

Figure 2 is a block diagram of one embodiment of a financial data management tool 200 such as an incentive compensation management (ICM) tool. The tool 202 includes a database 208, a retroactive data identifier 202, a retroactive revision manager 204, and a user interface generator 206.

The database 208 stores financial data of the organization. The financial data may include, for example, employee compensation data, account payable data, account receivable data, etc. In one embodiment, financial data includes transaction data and entity data. Transaction data pertains to occurrences of events. A transaction may include a transaction header and one or more transaction lines. A transaction header may include fields describing the transaction (e.g., transaction date, sales representatives, etc.). A transaction line may contain zero or more transaction events that describe the lifecycle of the transaction line. Transaction events are used as input for calculating financial results.

Entity data defines rules and requirements that apply to some or all events. Some entity data is versioned. A versioned entity is an entity for which the tool 200 maintains a historical catalogue. Every time a user edits a versioned entity, the tool 200 maintains the data for the pre-edit entity in a version record to preserve an audit trail for the entity.

The retroactive data identifier 202 is responsible for identifying input data indicative of retroactive changes to financial data stored in the database 208. Changes are considered retroactive if they are set to occur in a "closed" calendar period as may be specified, for example, by a transaction event date for transaction events or a working period begin date for versioned entities. A closed calendar period is referred to a period for which payments have already been made to intended recipients or received from a sender. Input data indicative of retroactive changes may be entered by a user via a user interface or imported from a document or a file.

In one embodiment, the retroactive data identifier 202 determines which transactions or versioned entities are subject to retroactive changes and records the received input data as separate records in the same database table to preserve an audit trial of the changes. Alternatively, the retroactive data identifier 202 stores the received input data in a separate database table designated to store retroactive revision data.

The retroactive revision manager 204 is responsible for processing the retroactive changes based on the received input data and grouping new financial results into a retroactive revision. In one embodiment, the retroactive revision manager 204 processes the retroactive changes by identifying a set of services that was used to operate for open calendar periods on transactions or versioned entities pertaining to the retroactive changes, and running a corresponding set of retroactive services using the received input data. In one embodiment, a set of services is one or more batch processes that read input, apply application logic defined in processing rules and write the results as output to the database 208. Each service uses the output of the previous service as its input. For example, in ICM, a set of services may be provided to process sales transaction event records and generate participant earning, payment and balance records. A retroactive service performs the same function as a regular service, except that it operates on data assigned to closed payment periods and affects data within the scope of a given retroactive revision. Embodiments of processing retroactive changes will be discussed in more detail below.

The retroactive revision manager 204 is responsible for maintaining a set of retroactive revisions. A retroactive revision (also referred to herein as a retro revision) groups retroactive edits resulted from a specific

input into a single entity. In one embodiment, a retro revision may include a set of retroactive transactions and adjustments as well as generated entities that are the output of the reprocessed retroactive input. For example, in February 2005 (P2), a compensation administrator may want to restate January 2005 (P1) plan earnings, by changing some transaction event pricing values and by altering some commission rates defined in the compensation plan. In this case, a retro revision called "P1_as_of_P2" will collectively group changes to transactions, commission rates, earnings, and balances.

A retro revision may have an open state or a closed state. While a retro revision is "open", each run of retroactive services deletes the data generated by the previous run of retroactive services for the same retro revision to allow the user to re-execute multiple retroactive runs without persisting permanent data to the database. As a result, an administrator can test and adjust retroactive data and plan changes without permanently committing data to the database and cluttering the audit trail with irrelevant processing history. When a retro revision is closed, no further processing may occur on the data that is part of this retro revision.

A retro revision may be associated with an operating unit (e.g., a division or department within an organization) and have a unique identifier. In one embodiment, the retroactive revision manager 204 automatically creates a first retro revision upon detecting first input or import of retroactive data. In one embodiment, the retroactive revision manager 204 ensures that there is only one open retro revision per operation unit at a time. When a system administrator is satisfied with the processing results performed in the retro revision, he or she may close this retro revision. When a retro revision is closed, the retroactive revision manager 204 may automatically create a new open retro revision so that retroactive changes are always bound to a retro revision. In one embodiment, the retroactive revision manager 204 ensures that no retroactive changes or processing occur outside the scope of a retro revision.

The user interface (UI) generator 206 is responsible for allowing users (e.g., a compensation administrator, plan participants, etc.) to view results of retroactive processing and compare them to previous calculations. UIs provided by the UI generator 206 may allow a user to see a list of retro revisions and various details for individual retro revisions. Exemplary UI will be discussed in more detail below in conjunction with Figures 8-16.

Figure 3 is a flow diagram of one embodiment of a process 300 for tracking retroactive changes to financial data. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. Processing logic may reside in a financial data management tool 104 of Figure 1.

Referring to Figure 3, process 300 begins with processing logic receiving data indicative of retroactive changes to financial results (processing block 302). As discussed above, data is indicative of retroactive changes if requested changes are set to occur in a closed calendar period as may be specified, for example, by a transaction event date for transaction events or a working period begin date for versioned entities. Input data indicative of retroactive changes may be entered by a user via a user interface or imported from a document or a file using an import service.

At processing block 304, processing logic records the data as a retroactive change. In one embodiment, processing logic identifies a currently open retroactive revision and links the retroactive change to the currently open retroactive revision.

At processing block 306, processing logic processes the retroactive changes based on the received data. In one embodiment, processing logic initiates processing of the retroactive changes by creating a retroactive processing service batch.

In one embodiment, processing logic creates a retroactive processing service batch by finding the earliest period in which retroactive changes are applicable and defining a sequence of retroactive services to recalculate the financial results (e.g., participant earnings and balances) for each period up to, but not including, the first open calendar period. The result of running retroactive services is that the calculation results for previous revisions are preserved, and new calculation results are created for the latest retro revision. As will be discussed in more detail below, retroactive processing may occur in a full recalculation mode, in which all generated entities from the earliest detected retroactive edit period are cancelled and recalculated regardless of value. Alternatively, retroactive processing may occur in an optimized recalculation mode that only recalculates those results that are affected by transaction event changes.

Next, processing logic associates the new financial results with the currently open retro revision (processing block 308) and stores them in the database as part of this retro revision (processing block 310).

At processing block 312, processing logic receives a user request to close the current retro revision. In response, processing logic changes the open state of the current retro revision to the closed state (processing block 314) and creates a new open retro revision (processing block 316).

Figure 3B is a flow diagram of one embodiment of a process 320 for tracking, processing, and viewing retroactive changes to financial data. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. Processing logic may reside in a financial data management tool 104 of Figure 1.

Referring to Figure 3B, process 320 begins with processing logic receiving retroactive changes (processing block 322). These changes may include, for example, transactions, quotas, formulas, other user-defined data, etc. The changes may be made individually through the user-interface, or in batch through a batch import mechanism. Next, processing logic processes the received retroactive changes. One embodiment for receiving and processing retroactive changes is described in greater detail in conjunction with Figure 4.

Next, the system administrator may request to review the retroactive changes, which are then presented to the system administrator through the user interface (processing block 324). Exemplary user interfaces presenting retroactive changes to the system administrator will be discussed in more detail below in conjunction with Figures 8-13.

If the system administrator and/or other users decide to introduce further retroactive changes, process 320 returns to processing block 322. If the system administrator decides that appropriate retroactive changes exist in the system, s/he may identify the required retroactive processes that should be performed to generate the new, retroactive financial results by sending a request to generate a "Retroactive Service Batch". Processing

logic receives the administrator's request to generate a retroactive service batch for the identified retroactive processes (processing block 326) and executes the retroactive service batch (processing block 328). One embodiment of a process for executing a retroactive service batch will be discussed in greater detail below in conjunction with Figure 5.

As discussed above, in one embodiment, the service batch may be executed in response to the request of the system administrator. Alternatively, the execution of the service batch may be triggered by a system timer.

After the batch execution is complete, the system administrator may be allowed to review the recalculated financial results (block 330). Exemplary user interfaces presenting the recalculated financial results will be discussed in more detail below in conjunction with Figures 14-16.

If the system administrator is unsatisfied with the calculation results, and decides to make further retroactive data changes and reprocess, process 320 returns to processing block 322. If the system administrator is satisfied with the recalculated financial results, then s/he may request to close the retro revision. An exemplary user interface used to close the retroactive revision is discussed in more detail below in conjunction with Figure 9.

In response to the administrator request, processing logic closes the retro revision (processing block 332) and then automatically creates a new, open retro revision (block 334). Then, if the user decides to introduce additional retroactive changes, process 320 returns to processing block 322.

Figure 4 is a flow diagram of one embodiment of a process 400 for receiving and processing retroactive changes to financial data. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. Processing logic may reside in a financial data management tool 104 of Figure 1.

Referring to Figure 4, process 400 begins with processing logic receiving data indicative of retroactive changes to financial results (processing block 402). At processing block 404, processing logic determines whether the changes are set to occur in a closed calendar period as may be specified, for example, by a transaction event date for transaction events or a working period begin date for versioned entities. Input data indicative of retroactive changes may be entered by a user via a user interface or imported from a document or a file using an import service. If it is determined that the change is not set to occur in a closed period, then the action is not considered to be a "retroactive change" (processing block 406), and process 400 ends.

If the change is set to occur in a closed period, then, processing logic further determines whether any retro revisions have been previously recorded (processing block 408). If so, processing logic identifies the currently open retro revision and records the data as a retroactive change by linking it to the retrieved retroactive revision (processing block 412). If no prior retro revisions exist, then processing logic creates the first retro revision (processing block 410) and proceeds to processing block 412.

In one embodiment, processing logic cancels previously generated cancellation results for each applicable closed period. In particular, individual transaction events may be cancelled by setting a "cancelled" bit field to true and adding an identifier of a current retro revision to a retro revision field in relevant database

records. Versioned entities do not have a "cancelled" bit field because their built-in record versioning capability provides similar functionality. Existing versioned entities may be modified during a retroactive edit operation by adding an identifier of a current retro revision to a retro revision field in relevant database records.

At processing block 412, logic associates the created, updated, or cancelled records with the current retro revision and stores these records in the database. In one embodiment, processing logic inserts each new transaction event record with its cancelled flag set to false and retro revision field (e.g., foreign key field) set to the current retro revision ID. Each new versioned entity record is inserted with its retro revision field (e.g., foreign key field) set to the current retro revision ID. In one embodiment, logic stores retroactive transaction adjustments in a transaction adjustment history table and retroactive adjustments to versioned entities in a retroactive revision entity table. The transaction adjustment history table stores record locator information that associates retroactively changed transaction events with the then-current retro revision. The retroactive revision entity table stores record locator information that associates retroactively changed versioned entities with the then-current retro revision. Recording retroactive changes in this manner allows the administrator to quickly view retroactive changes, and allows the system to rapidly determine retroactive processing requirements during the execution of retroactive services.

An exemplary transaction adjustment history table called "SALESTRANS_ADJ" is defined below. This table records all adjustment actions (both retroactive and non-retroactive). In an alternative embodiment, the user may set the "transaction.adjustment.fullHistory" configuration property to false to minimize the consumption of disk space. With the "transaction.adjustment.fullHistory" property set to "false", adjustments to transactions or lines in open periods result in a simple record update, not a cancel/create adjustment. The "SALESTRANS_ADJ" database table may have the following ANSI/ISO SQL '99 definition:

```
CREATE TABLE SALESTRANS_ADJ (
    salestrans_adj_uuid  varchar(36) NOT NULL,
    salestrans_number    nvarchar(50) NOT NULL,
    salestrans_uuid      varchar(36) NOT NULL,
    adjustee_salestrans_uuid varchar(36) NULL,
    salestrans_line_uuid varchar(36) NULL,
    adjustee_salestrans_line_uuid varchar(36) NULL,
    salestrans_event_uuid varchar(36) NULL,
    retro_revision_uuid  varchar(36) NULL,
    adjustment_action    varchar(50) NOT NULL,
    adjustment_comment   nvarchar(512) NULL,
    adjustment_date      datetime NOT NULL,
    transaction_date     datetime NOT NULL,
    unit_uuid            varchar(36) NOT NULL,
    adjustment_user_uuid varchar(36) NOT NULL,
    import_service_run_uuid varchar(36) NULL
)
```

The line_uuid (line universally unique identifier (UUID)) and event_uuid are set where appropriate as determined by the adjustment action. For example, adding a new transaction line will set the line_uuid, and adding a new transaction event will set the event_uuid. Otherwise these fields are left to NULL values. For header adjustments, the salestrans_uuid field may be set to UUID value of the adjustor (new) transaction record, and the adjustee_salestrans_uuid field may be set to the UUID value of the adjustee (old) transaction record.

For line adjustments, the line_uuid field may point to the adjustor (new) line record, and the adjustee_line_uuid field may point to the adjustee (old) line record. Transaction lines may be adjusted independently of the header. An adjustment to the transaction header causes adjustments to all of its lines (so they will be reprocessed), however this is considered a single action in the SALESTRANS_ADJ table. The salestrans_uuid and salestrans_number fields are always set. The import_service_run_uuid field is set if the adjustment action occurred by a transaction import service run. This allows grouping of adjustment actions by import service run.

In one embodiment, all retroactive activities to versioned entities are recorded in a single table. An exemplary retroactive versioned entity table called "RETRO_REVISION_MEMBER" has the following ANSI/ISO SQL '99 definition:

```
CREATE TABLE [dbo].[RETRO_REVISION_MEMBER] (
        [retro_revision_member_uuid] [varchar] (36) NOT NULL ,
        [object_type_name] [varchar] (255) NOT NULL ,
        [object_instance_uuid] [varchar] (36) NULL ,
        [object_version_uuid] [varchar] (36) NULL ,
        [retro_revision_uuid] [varchar] (36) NOT NULL ,
        [retro_period_uuid] [varchar] (36) NOT NULL ,
        [revision_action] [varchar] (36) NOT NULL ,
        [entity_code] [varchar] (30) NULL ,
        [description] [nvarchar] (255) NULL ,
        [activation_date] [datetime] NOT NULL
)
```

In one embodiment, data is inserted into this table by code that resides in an abstract base class called the BaseServicerBean. At the application level, all persistence capable service classes extend this abstract base class to host "retroactive detection" logic. The "retroactive detection" logic executes immediately before all database operations (e.g., Insert, Update, or Delete operations), and determines whether the operation occurs in a "closed" period. If the operation occurs in a closed period, then the user is performing a "retroactive" operation, which is recorded in the RETRO_REVISION_MEMBER table. This links the retroactive operation to the then-current retro revision. For each edit of a non-transaction entity, a create versioned entity record and a deactivate versioned entity record are inserted into the RETRO_REVISION_MEMBER table.

In one embodiment, UI edit actions or import adjustment actions cause the deactivation and creation of a new version in the same transaction. As a result, more than one record may be inserted into this table for a single edit operation performed through the UI or Import interface.

Figure 5 is a flow diagram of one embodiment of a process 500 for executing a retroactive service batch. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. Processing logic may reside in a financial data management tool 104 of Figure 1.

Referring to Figure 5, process 500 begins with receiving a request to generate a retroactive service batch (processing block 502). The request may be initiated by a user (e.g., manually through a batch import mechanism) or in response to a timer event (e.g., scheduled). In one embodiment, the request specifies whether retroactive processing should occur in a full recalculation mode or an optimized recalculation mode.

In one embodiment, processing logic creates a retroactive processing service batch by finding the earliest period in which retroactive changes are applicable (processing block 504). Starting from the earliest identified period, processing logic defines a sequence of retroactive services to recalculate the financial results (e.g., participant earnings and balances) for each period up to, but not including, the first open calendar period (processing block 506). After the retroactive service batch is defined, processing logic saves its definition (processing block 508). The service batch definition is saved so that it may be executed at a later time. In one embodiment, the service batch definition may be executed repeatedly.

The result of executing a retroactive service batch is that retroactive services recalculate financial results for the latest revision while preserving the calculated results linked to previous revisions. As will be discussed in more detail below, retroactive processing may occur in a full recalculation mode, in which all generated entities from the earliest detected retroactive edit period are cancelled and recalculated regardless of value. Alternatively, retroactive processing may occur in an optimized recalculation mode that only recalculates those results that are affected by transaction event changes.

Figure 6 is a flow diagram of one embodiment of a process 600 for processing retroactive changes in a full recalculation mode. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. Processing logic may reside in a financial data management tool 104 of Figure 1.

Referring to Figure 6, process 600 begins with processing logic receiving either a user request or a timer event (e.g., scheduler) initiating execution of a retroactive service batch (processing block 602). At processing block 604, processing logic identifies services defined in the retroactive service batch (processing block 604), and for each of these services, repeats processing blocks 606 through 614.

At processing block 606, processing logic deletes the results of the previous retroactive run from the database if this retroactive service was previously run for the current retro revision (processing block 606), and proceeds to processing block 608. If this retroactive service was not previously run for the current retro revision, processing logic proceeds directly to processing block 608.

At processing block 608, processing logic cancels previously generated results for the applicable closed period. Individual output records are cancelled by setting a "cancelled" bit field to true and adding an identifier of a "cancelled in retro revision" field which is set to the current retro revision in relevant database records. This step invalidates the results of the last calculation run, yet preserves the prior results for audit trail purposes.

At processing block 610, processing logic identifies input records to be processed. The input records are identified based on selection logic specific to the retroactive service encountered at block 604. Because this retroactive service is operating in "full recalculation" mode, the service identifies all active (non-cancelled) input records for the processing period as eligible for processing.

At processing block 612, processing logic executes user-defined, service-specific processing logic for each item identified at block 610. The processing of each input item may generate zero, one, or more service-specific output records that are associated with the current retro revision and persisted to the database (processing block 614). If the currently executing service is the last service defined in the batch, then processing

of the retro service batch ends (processing block 618). If not, process 600 returns to processing block 606 to start the next service.

Figure 7 is a flow diagram of one embodiment of a process 700 for processing retroactive changes in an optimized recalculation mode. The process may be performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as run on a general purpose computer system or a dedicated machine), or a combination of both. Processing logic may reside in a financial data management tool 104 of Figure 1.

Referring to Figure 7, process 700 begins with processing logic starting the execution of a retroactive service batch either in response to a user request or a timer event (e.g., scheduler) (processing block 702). Next, processing logic identifies services defined in the retroactive service batch (processing block 704), and for each of these services repeats processing blocks 706 through 718.

A retroactive service begins processing by determining whether this retroactive service was previously executed for the current retro revision (processing block 706). If so, processing logic deletes the results of the previous retroactive run from the database (processing block 706) and proceeds to processing block 708. If not, processing logic proceeds directly to processing block 708.

Processing logic "uncancels" output records that were potentially canceled by a previous full recalculation or optimized retroactive run (processing block 708). The records are identified for "uncancellation" if their created-in-retrorevision field is set to the identifier of the currently open retro revision and their cancelled flag is set to true. In one embodiment, processing logic uncancels previously canceled records by setting a retro revision field to null and setting a cancelled flag to false for the relevant output records.

At processing block 710, processing logic cancels only those previously generated results affected by retroactive changes. For example, the Retroactive Earnings Calculation Service may identify individual earnings for cancellation by examining their input credit via foreign key reference. If the input credit has a cancelled flag field set to true and a "cancelled-in-retro-revision" value equal to the identifier of the current retroactive revision, then the earning is determined to be in need of recalculation and is therefore marked as "cancelled". In this example, the identified records are cancelled by setting a "cancelled" bit field to true and adding an identifier of a current retro revision to a retro revision field in relevant database records.

At processing block 712, processing logic finds only those input records that are affected by retroactive changes, as well as any additional service-specific selection criteria. Input records will typically meet these criteria by having a "cancelled" flag set to false, and a "created-in-retroactive-revision" field equal to the value of the current retroactive revision's identifier. In one embodiment, the Retroactive Earnings Calculation Service will only identify and return those uncancelled credits as input records that were calculated as part of the current retroactive revision. In this case the identified input credits have a "canceled" flag set to false, and a "created-in-retroactive-revision" value equal to the identifier of the current retroactive revision.

At processing block 714, the system executes all user-defined, service-specific processing logic for each item identified at block 712. The processing of each input item may generate zero, one, or more service-specific output records that are associated with the current retro revision and persisted to the database

(processing block 616). If the currently executing service is the last service defined in the batch, then processing of the retro service batch ends (logic block 718). If not, the next service begins by returning program control to processing block 706.

The full recalculation mode takes into account any and all retroactive system changes that may affect calculation results (e.g., organization structure changes, advanced component changes, and transaction changes). The drawback of the full recalculation mode is that it is a resource intensive set of operations. The advantage of the optimized recalculation mode is that it is a faster and more efficient process than full recalculation mode. However, the optimized recalculation mode only takes into account only retroactive changes that result from retroactively created, updated, or deleted transaction events, but not other system changes that may affect calculation results.

In one embodiment, retroactive transaction events have an additional property referred to as a retro processing option. Based on the value of the retro processing option, the retroactive processing logic determines which period to apply the retroactive change. In one embodiment, there are two possible values for the retro processing options: PROCESS_RETRO which indicates that this event will be included in the next retro processing run (e.g., in ICM it will potentially generate new participant earning and balance values); and PROCESS_IN_OPEN_PERIOD which indicates that the event will not be included in the next retro processing run but will be processed instead in the first open period by the normal open period processing services.

In one embodiment, the retro processing option is stored as part of the transaction event. The processing option may be initially set when a retro insert, update, or delete operation is performed, and may be changed through a UI (e.g., Retroactive Transaction Event Change UI discussed below) if the retro revision is still open.

In one embodiment, for each transaction adjustment action performed by the user through the user interface or via a transaction import service, several algorithmic steps are performed to ensure that an accurate transaction event-to-credit mapping is maintained for payment and audit trail purposes. Different algorithmic steps are performed depending on the transaction adjustment action and the retro processing option. Table 1 illustrates exemplary transaction adjustment algorithmic steps performed for different adjustment types. In one embodiment, if the "transaction.adjustment.fullHistory" property is set to "true", then every retro action listed in Table 1 results in an entry added to the transaction history table. This step is not shown for each action to avoid redundancy.

Table 1: Transaction Adjustment Algorithmic Steps

Adjustment Type	Retro Processing Option	Algorithmic Steps Performed
New Event	PROCESS_RETRO, PROCESS_IN_OPEN_PERIOD	• Set the processing option and created_retro_revision_uuid on the new event.
New Line	PROCESS_RETRO, PROCESS_IN_OPEN_PERIOD	• Set the processing option and created_retro_revision_uuid on events belonging to the new line.
New Transaction	PROCESS_RETRO, PROCESS_IN_OPEN_PERIOD	• Set the processing option and created_retro_revision_uuid on all events on all of the new transaction's lines.
Cancel Event	PROCESS_RETRO	• Set the cancelled flag and retroed_retro_revision_uuid on the event. • Set the cancelled flag and retroed_retro_revision_uuid on all credits generated from the newly cancelled event.
Cancel Event	PROCESS_IN_OPEN_PERIOD	• Set the cancelled flag and retroed_retro_revision_uuid on the event. • Generate reversing credits in the first open period for all credits generated from the newly cancelled event. • Set the created_retro_revision_uuid on these reversing credits.
Cancel Line	PROCESS_RETRO	• Set the cancelled flag on the line and all of its events. • Set the retroed_retro_revision_uuid and the processing option on all of the line's cancelled events. • Set the cancelled flag and retroed_retro_revision_uuid on all credits generated from the cancelled events
Cancel Line	PROCESS_IN_OPEN_PERIOD	• Set the cancelled flag on the line and all of its events. • Set the retroed_retro_revision_uuid and the processing option on all of the line's cancelled events • Generate reversing credits in the first open period for all credits

		generated from the newly cancelled events. • Set the created_retro_revision_uuid on these reversing credits.
Cancel Transaction	PROCESS_RETRO	• Set the cancelled flag on the transaction, all of its lines, and all of their events. • Set the retroed_retro_revision_uuid and processing option on the cancelled events. • Set the cancelled flag and retroed_retro_revision_uuid on all credits generated from the cancelled events
Cancel Transaction	PROCESS_IN_OPEN_PERIOD	• Set the cancelled flag on the transaction, all of its lines, and all of their events. • Set the retroed_retro_revision_uuid and processing option on the transaction's cancelled events. • Generate reversing credits in the first open period for all credits that were generated from the cancelled events. • Set the created_retro_revision_uuid on these reversing credits.
Adjust Line / Return Line	PROCESS_RETRO	• Set the cancelled flag on the adjustee line and all of its events. • Set the retroed_retro_revision_uuid on all of the adjustee line's events. Do NOT set the processing option • Set the created_retro_revision_uuid and processing option on all of the adjustor line's events. • Set the cancelled flag and retroed_retro_revision_uuid on all credits previously generated from the newly cancelled adjustee events.
Adjust Line / Return Line	PROCESS_IN_OPEN_PERIOD	• Set the cancelled flag on the adjustee line and all of its events. • Set the retroed_retro_revision_uuid on

		all of the adjustee line's events. Do not set the processing option. • Set the created_retro_revision_uuid and processing option on all of the adjustor line's events. • Generate reversing credits in the first open period for all credits that were generated from the newly cancelled events on the adjustee line. • Set the created_retro_revision_uuid on these reversing credits.
Adjust Transaction	PROCESS_RETRO	• Set the cancelled flag on the adjustee transaction, all of its lines and all of its events. • Set the retroed_retro_revision_uuid on all of the adjustee transaction's events. Do NOT set the processing option. • Set the created_retro_revision_uuid and processing option on all of the adjustor transaction's events. • Set the cancelled flag and retroed_retro_revision_uuid on all credits generated from the newly cancelled adjustee events.
Adjust Transaction	PROCESS_IN_OPEN_PERIOD	• Set the cancelled flag on the adjustee transaction, all of its lines and all of its events. • Set the retroed_retro_revision_uuid on all of the adjustee line's events. Do NOT set the processing option. • Set the created_retro_revision_uuid and processing option on all of the adjustor transaction's events. • Set the processing option on all of the adjustor transaction's lines' events. • Generate reversing credits in the first open period for all credits that were generated from the newly cancelled adjustee events. • Set the created_retro_revision_uuid on these reversing credits.

Figures 8-16 illustrate exemplary UIs provided by an incentive compensation management (ICM) tool such as a tool 200 of **Figure 2**.

Referring to **Figure 8**, Retro Revision Index UI 800 provides a time-based history of all retro revisions. The latest revision is displayed at the top of the list. As the compensation administrator (Comp Admin) performs retroactive edit operations and runs retroactive services, the retro activity is recorded and displayed as a brief summary on this UI. Code 802 is the unique code of the retro revision that is generated by concatenating the Operating Unit Code and a sequential integer. The Comp Admin may edit this code and use the convention, "Px_as_of_Py", where "Px" is the target period being revised, and "Py" is the first open calendar period. Revision Number 804 is the sequence number of the retro revision. Status 806 displays one of the two retro revision states, "open" or "closed". Period 808 is the period in which the retro revision was closed. This indicates that the revision's changes were valid "as of" the given period.

Events field 810 provides a count of the number of transaction events that have been retroactively changed during this retro revision. The number is hyperlinked to the output of the Retroactive Transaction Event Changes UI described below. Entities field 812 provides a count of the number of non-transaction entities that have been retroactively changed during this retro revision. The number is hyper linked to the Retro Entity Changes UI described below. Last launched field 814 provides the localized date and time of the last retroactive service run. If retro processing has never commenced for this retro revision, then the phrase "Not Available" is displayed. This value is hyperlinked to the Retro Service Run history UI. Description 816 provides a text description describing the nature of the retroactive edits.

The Comp Admin may edit an open retro revision by clicking on the open revision's edit icon 818. This transfers the user to the Retro Revision Edit UI 900 illustrated in Figure 9. The Comp Admin may close the retro revision and automatically create a new retro revision by selecting the "closed" option from the retro revision status field 902 and clicking the "Save" button 904. The ICM tool sets the revision status to closed, records the close date and time, sets the first open calendar period as the retro revision's period, and records the User ID of the user that closed the revision. After closing a retro revision, the ICM tool automatically creates a new "open" retro revision.

Referring to Figure 10, a Retro Entity Changes UI 1000 is provided to display the contents of the RETRO_REVISION_MEMBER table for a given retro revision. Each of the columns in the search result table has a sort link so that the results may be ordered by that column's values.

Referring to Figure 11, a Retroactive Transaction Event Changes UI 1100 is illustrated, which displays all records from the SALESTRANS_ADJ table for a given retro revision. The UI 1100 includes a transaction number field 1102 that is hyperlinked to the original transaction or line record as it exists after the retroactive change. The user has the option to set the "Batch Processing Option" for all of the events at once by clicking the "Batch Processing Options" edit icon 1104. This brings a Retroactive Transaction Event Changes view 1200 illustrated in Figure 12. Clicking the "Save" button 1202 in UI 1200 will update the processing option of every transaction event in this retro revision, not just the ones on the screen or in the current search results.

To change the processing options for an individual member in the list, the user clicks on a specific transaction number to display a more detailed Transaction Event Processing Option UI 1300 illustrated in Figure 13. UI 1300 displays all of the retroactive events for a single transaction. The Comp Admin has the option to set the processing option for all of the retro events for this transaction by selecting a desired processing option 1302. This update will apply to only the retroactive events for the selected transaction. If the Comp Admin wishes to edit each retro event individually, he or she may set the individual Process Option value and click the "save" disk icon 1304. Clicking the save button 1304 or cancel button 1306 will return the user to the Retroactive Transaction Event Changes UI 1200.

Referring to Figure 14, a Participant Ledger UI 1400 is illustrated which displays the latest calculation results for a participant's beginning balance 1402, summarized earnings 1404, payments 1406, and ending balance 1408 for the entire calendar year and a given payment group. The UI 1400 displays the Participant Ledger View for a plan participant who received retroactively calculated summarized earnings in Period 10, 2003 in the "Bonus" payment group 1412.

The presence of a clock icon 1410 for a period entry indicates that the period has undergone retroactive processing. Clicking on the clock icon 1410 transfers the user to a Summarized Earning History UI 1500 illustrated in Figure 15.

The Summarized Earning History UI 1500 provides a filter mechanism to display participant earnings across payment groups and calendar periods in both functional and user currencies. The UI 1500 displays all revision values, including the original value, calculated for a given participant, period, and payment group. For each retroactive revision 1512, the recalculated fields include beginning balance 1502, summarized earning 1504, ending balance 1510, and retroactive balance adjustment 1506. The Payment field 1508 never changes as it shows the amount that a participant was paid in a past period.

As shown in the UI 1500, a participant's earnings for Period 10, 2003 were retroactively reduced from $10,973.80 to $2,263.00 in the retro revision referred to as "1, UFinanceOU_1". This retroactive calculation resulted in a balance adjustment of $8,710.80. The plan participant now owes the company $8,715.80 for erroneous payments he received as part of the "Bonus" payment group in Period 10 of 2003.

Referring to Figure 16, a Retroactive Summary UI 1600 is illustrated, which displays the retroactive processing activity for each type of generated entity that may be retroactively calculated. The user has the option of searching for retroactive activity by calendar year and retro revision . After selecting the desired calendar year 1602 and retro revision 1604, and clicking the "Search" button 1606, the system displays the count of Credits, Cumulated Credits, Cumulated Goals, Earnings, and Summarized Earnings that were cancelled and created in each period. Each value in the "Cancelled" or "Created" columns is hyperlinked to a search result screen that will retrieve the records that were retroactively cancelled or created. The UI 1600 provides the user with a fast way to survey the results of retro processing for a single participant.

Figure 17 is a block diagram of an exemplary computer system 1700 (e.g., a server hosting the business process definition controller 100 of Figure 1) that may be used to perform one or more of the operations described herein. In alternative embodiments, the machine may comprise a network router, a network switch, a

network bridge, Personal Digital Assistant (PDA), a cellular telephone, a web appliance or any machine capable of executing a sequence of instructions that specify actions to be taken by that machine.

The computer system 1700 includes a processor 1702, a main memory 1704 and a static memory 1706, which communicate with each other via a bus 1708. The computer system 1700 may further include a video display unit 1710 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)). The computer system 1700 also includes an alpha-numeric input device 1712 (e.g., a keyboard), a cursor control device 1714 (e.g., a mouse), a disk drive unit 1716, a signal generation device 1720 (e.g., a speaker) and a network interface device 1722.

The disk drive unit 1716 includes a computer-readable medium 1724 on which is stored a set of instructions (i.e., software) 1726 embodying any one, or all, of the methodologies described above. The software 1726 is also shown to reside, completely or at least partially, within the main memory 1704 and/or within the processor 1702. The software 1726 may further be transmitted or received via the network interface device 1722. For the purposes of this specification, the term "computer-readable medium" shall be taken to include any medium that is capable of storing or encoding a sequence of instructions for execution by the computer and that cause the computer to perform any one of the methodologies of the present invention. The term "computer-readable medium" shall accordingly be taken to include, but not be limited to, solid-state memories, optical and magnetic disks, and carrier wave signals.

Whereas many alterations and modifications of the present invention will no doubt become apparent to a person of ordinary skill in the art after having read the foregoing description, it is to be understood that any particular embodiment shown and described by way of illustration is in no way intended to be considered limiting. Therefore, references to details of various embodiments are not intended to limit the scope of the claims which in themselves recite only those features regarded as essential to the invention.

CLAIMS

What is claimed is:

1. A computerized method comprising:
receiving data indicative of one or more retroactive changes to at least one input;
processing the retroactive changes based on the received data to generate new financial results;
grouping the retroactive changes and the new financial results into a retroactive revision; and
reprocessing existing calculations performed between the earliest time period and a most recent closed time period based on the received data.

2. The method of claim 1 wherein the financial results comprise incentive compensation calculations.

3. The method of claim 1 wherein the retroactive changes are associated with at least one of transaction data and versioned data.

4. The method of claim 3 wherein processing the retroactive changes comprises:
finding an earliest time period applicable to the retroactive changes.

5. The method of claim 4 wherein reprocessing existing calculations comprises:
performing new calculations only for financial results impacted by the retroactive changes;
adding a cancellation indicator only to records with financial results impacted by the retroactive changes; and
adding new records with the new calculations to a database, the new records being associated with the retroactive revision.

6. The method of claim 4 wherein reprocessing existing calculations comprises:
adding a cancellation indicator to all records with the existing calculations;
performing new calculations for the records based on the received data; and
adding new records with the new calculations to a database, the new records being associated with the retroactive revision.

7. The method of claim 4 further comprising:
displaying new calculations to a user; and
modifying an open state of the retroactive revision to a closed state upon receiving a user request.

8. The method of claim 7 wherein the open state allows changes to the retroactive revision, and the closed state does not allow changes to the retroactive revision.

9. The method of claim 1 further comprising:
automatically creating a new retroactive revision when the current retroactive revision enters a closed state.

10. The method of claim 1 further comprising:
receiving a user request to re-process retroactive changes for the retroactive revision;
deleting existing financial results of the retroactive revision; and
re-processing retroactive changes for the retroactive revision.

11. The method of claim 1 wherein the one or more retroactive changes comprise any one of a retroactive entity change and a retroactive transaction change.

12. A machine-readable medium having executable instructions to cause a machine to perform a method comprising:
receiving data indicative of one or more retroactive changes to at least one input;
processing the retroactive changes based on the received data to generate new financial results;
grouping the retroactive changes and the new financial results into a retroactive revision; and
reprocessing existing calculations performed between the earliest time period and a most recent closed time period based on the received data..

13. The machine-readable medium of claim 12 wherein the financial results comprise incentive compensation calculations.

14. The machine-readable medium of claim 12 wherein processing the retroactive changes comprises:
finding an earliest time period applicable to the retroactive changes; and
reprocessing existing calculations performed between the earliest time period and a most recent closed time period based on the received data.

15. A system comprising:
a retroactive data identifier to receive data indicative of one or more retroactive changes to at least one input;

a retroactive revision manager to process the retroactive changes based on the received data to generate new financial results, and to group the retroactive changes and the new financial results into a retroactive revision; and

a reprocessor configured to existing calculations performed between the earliest time period and a most recent closed time period based on the received data..

16. The system of claim 15 wherein the financial results comprise incentive compensation calculations.

17. The system of claim 15 wherein the retroactive revision manager is to process the retroactive changes by finding an earliest time period applicable to the retroactive changes, and reprocessing existing calculations performed between the earliest time period and a most recent closed time period based on the received data.

18. The system of claim 17 wherein the retroactive revision manager is to reprocess existing calculations by performing new calculations only for financial results impacted by the retroactive changes, adding a cancellation indicator only to records with the financial results impacted by the retroactive changes, and adding new records with the new calculations to a database, the new records being associated with the retroactive revision.

19. The system of claim 17 wherein the retroactive revision manager is to reprocess existing calculations by adding a cancellation indicator to all records with the existing calculations, performing new calculations for the records based on the received data, and adding new records with the new calculations to a database, the new records being associated with the retroactive revision.

20. The system of claim 17 wherein the retroactive revision manager is further to modify an open state of the retroactive revision to a closed state in response to a user request or a scheduled event.

21. A computer-implemented apparatus comprising:

means for receiving data indicative of one or more retroactive changes to at least one input;

means for processing the retroactive changes based on the received data to generate new financial results;

means for grouping the retroactive changes and the new financial results into a retroactive revision; and

means for reprocessing existing calculations performed between the earliest time period and a most recent closed time period based on the received data..

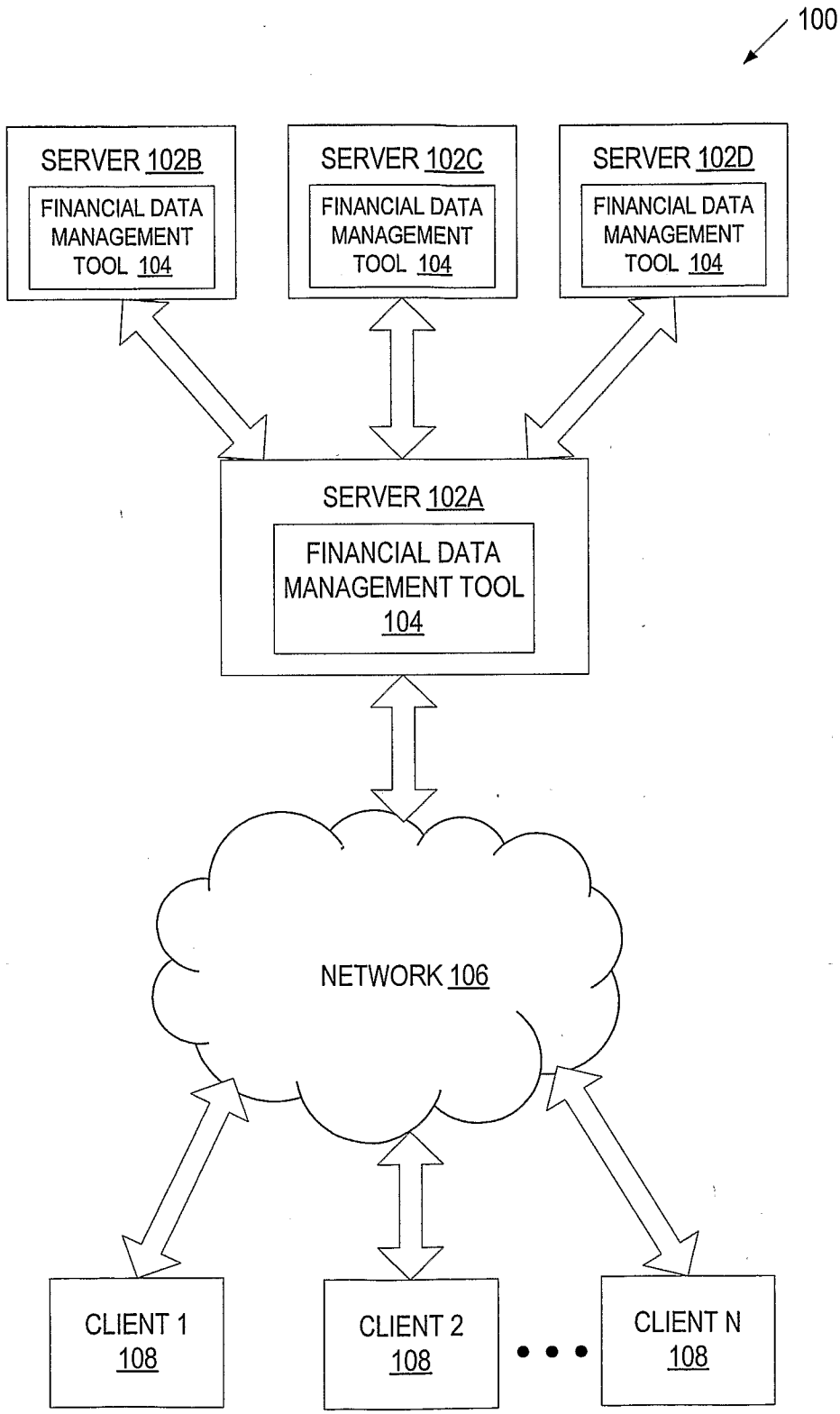

100
SERVER 102B
FINANCIAL DATA MANAGEMENT TOOL 104
SERVER 102C
FINANCIAL DATA MANAGEMENT TOOL 104
SERVER 102D
FINANCIAL DATA MANAGEMENT TOOL 104
SERVER 102A
FINANCIAL DATA MANAGEMENT TOOL 104
NETWORK 106
CLIENT 1 108
CLIENT 2 108
CLIENT N 108

FIG. 1

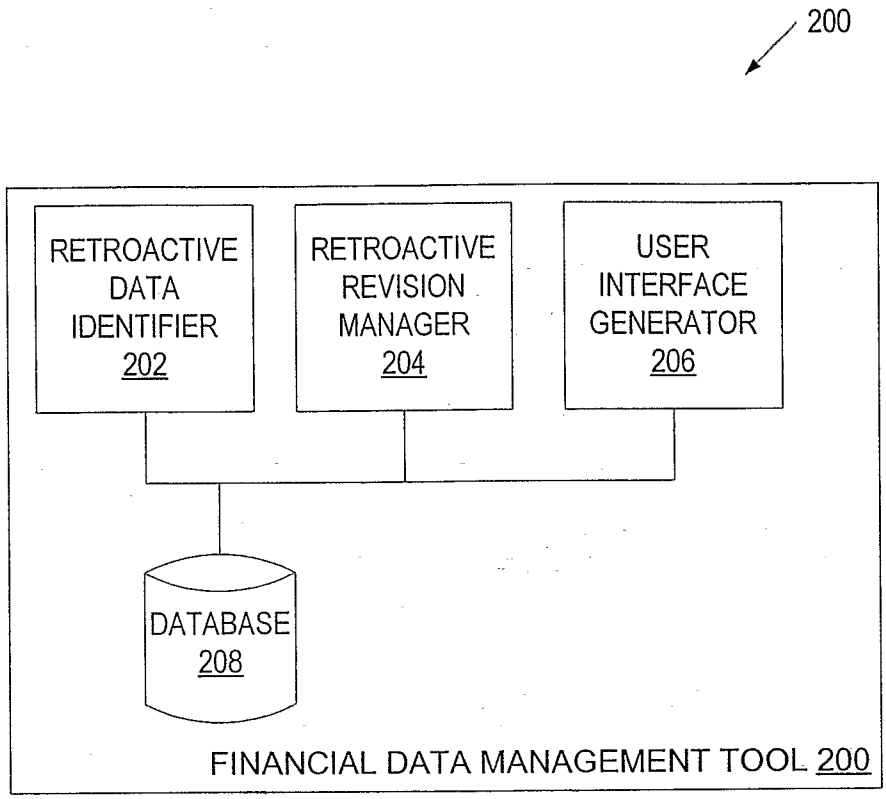

200
RETROACTIVE DATA IDENTIFIER 202
RETROACTIVE REVISION MANAGER 204
USER INTERFACE GENERATOR 206
DATABASE 208
FINANCIAL DATA MANAGEMENT TOOL 200

FIG. 2

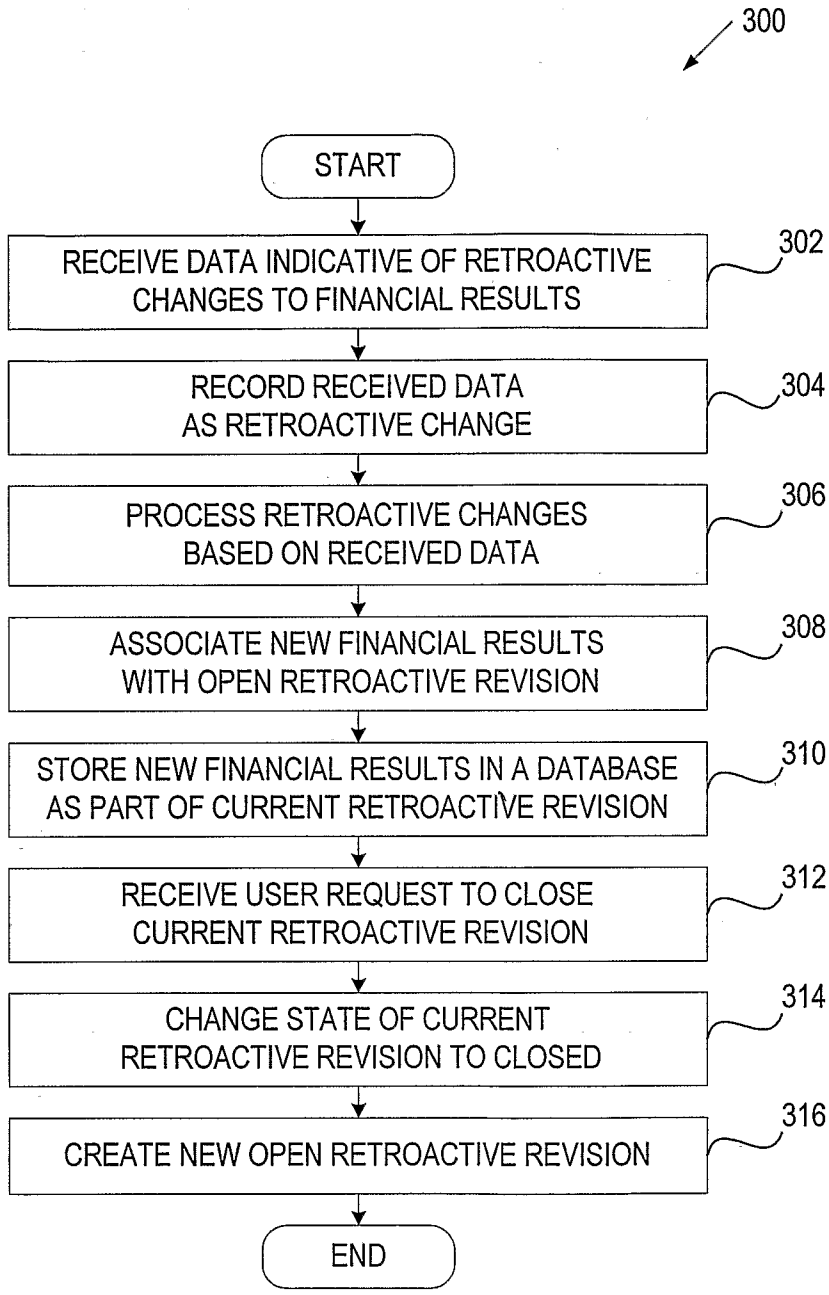

300
START
RECEIVE DATA INDICATIVE OF RETROACTIVE CHANGES TO FINANCIAL RESULTS
302
RECORD RECEIVED DATA AS RETROACTIVE CHANGE
304
PROCESS RETROACTIVE CHANGES BASED ON RECEIVED DATA
306
ASSOCIATE NEW FINANCIAL RESULTS WITH OPEN RETROACTIVE REVISION
308
STORE NEW FINANCIAL RESULTS IN A DATABASE AS PART OF CURRENT RETROACTIVE REVISION
310
RECEIVE USER REQUEST TO CLOSE CURRENT RETROACTIVE REVISION
312
CHANGE STATE OF CURRENT RETROACTIVE REVISION TO CLOSED
314
CREATE NEW OPEN RETROACTIVE REVISION
316
END

FIG. 3A

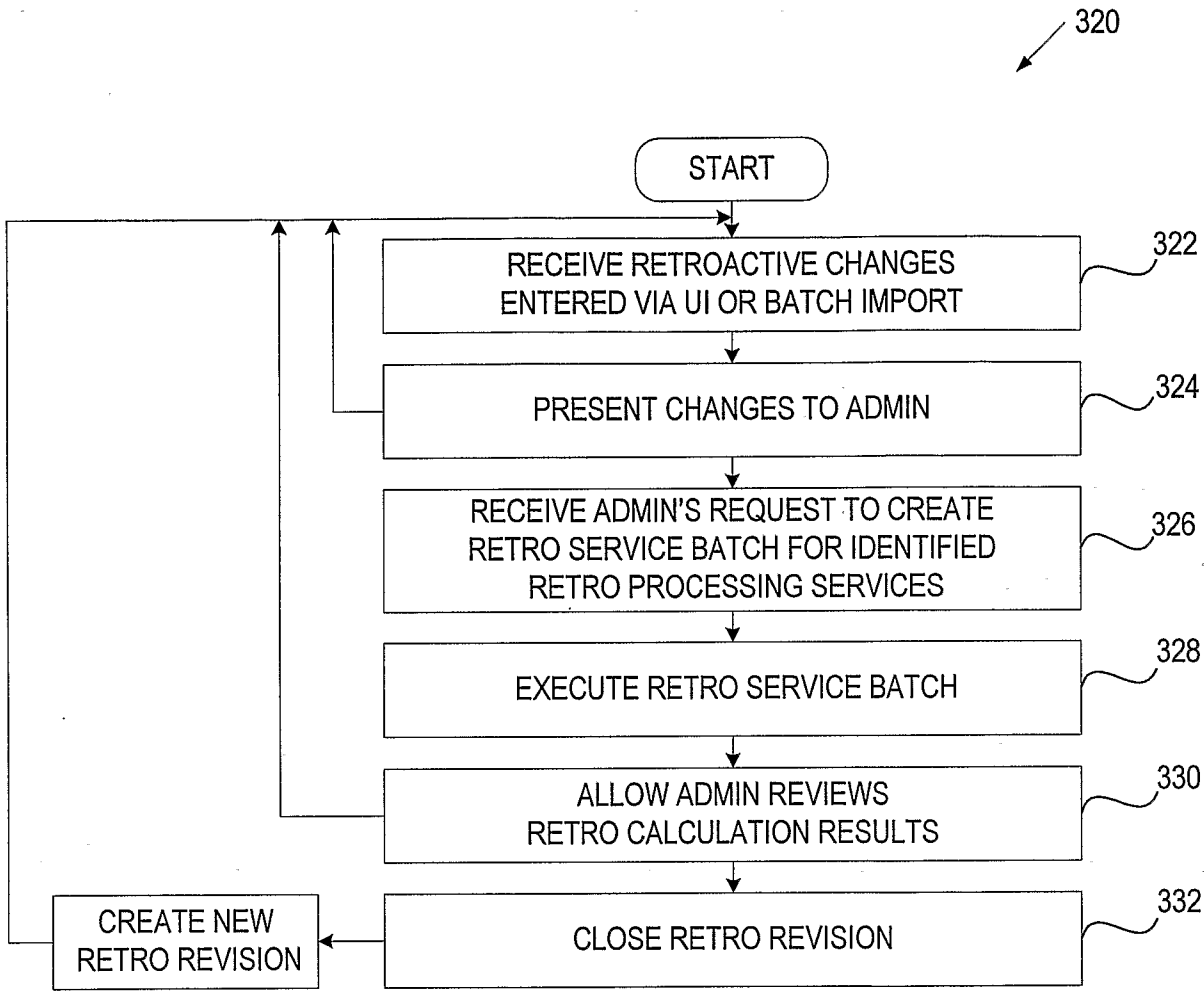

320
START
RECEIVE RETROACTIVE CHANGES ENTERED VIA UI OR BATCH IMPORT
322
PRESENT CHANGES TO ADMIN
324
RECEIVE ADMIN'S REQUEST TO CREATE RETRO SERVICE BATCH FOR IDENTIFIED RETRO PROCESSING SERVICES
326
EXECUTE RETRO SERVICE BATCH
328
ALLOW ADMIN REVIEWS RETRO CALCULATION RESULTS
330
CLOSE RETRO REVISION
332
CREATE NEW RETRO REVISION

FIG. 3B

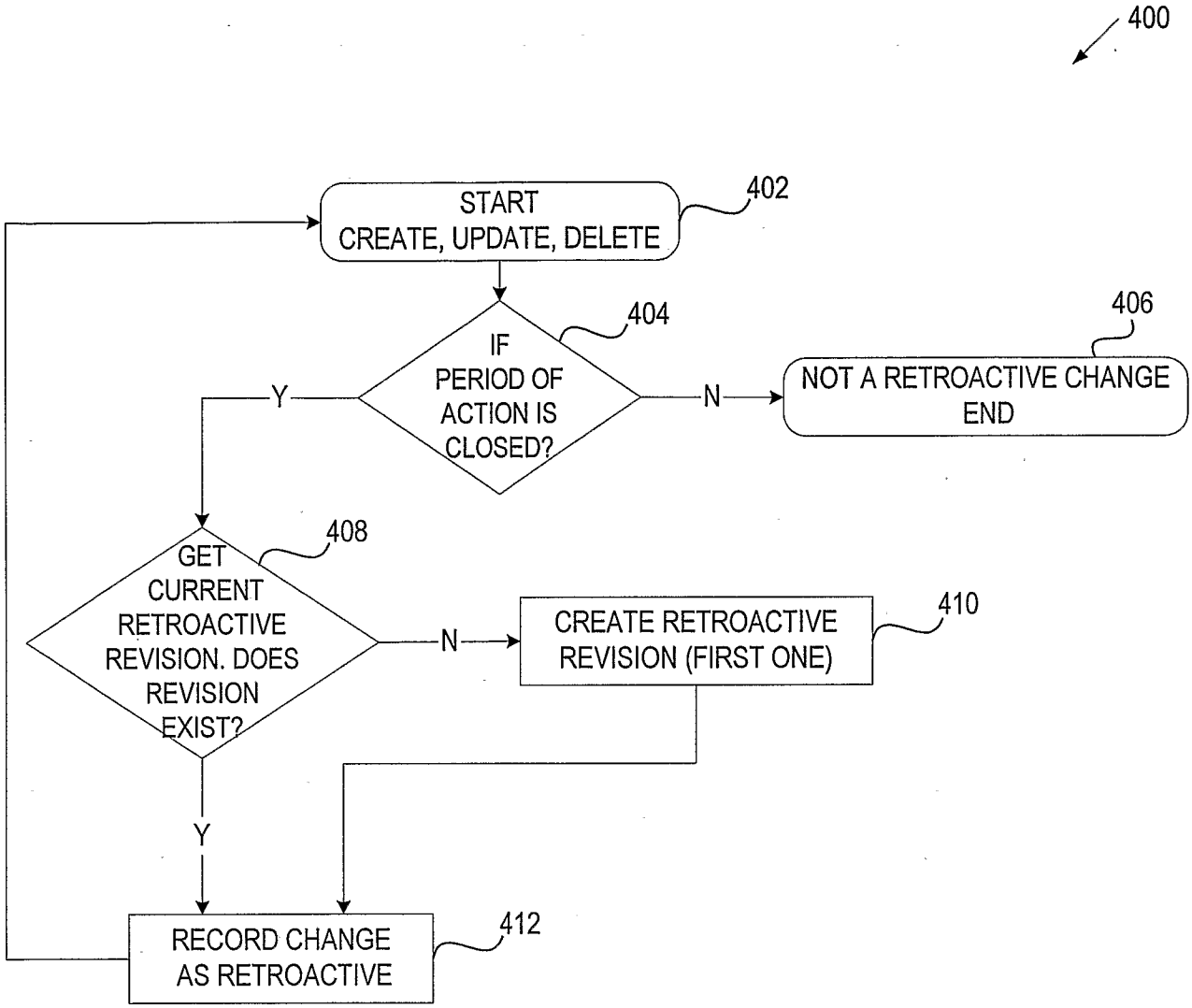

400
START
CREATE, UPDATE, DELETE
402
IF
PERIOD OF
ACTION IS
CLOSED?
404
Y
N
NOT A RETROACTIVE CHANGE
END
406
GET
CURRENT
RETROACTIVE
REVISION. DOES
REVISION
EXIST?
408
N
CREATE RETROACTIVE
REVISION (FIRST ONE)
410
Y
RECORD CHANGE
AS RETROACTIVE
412

FIG. 4

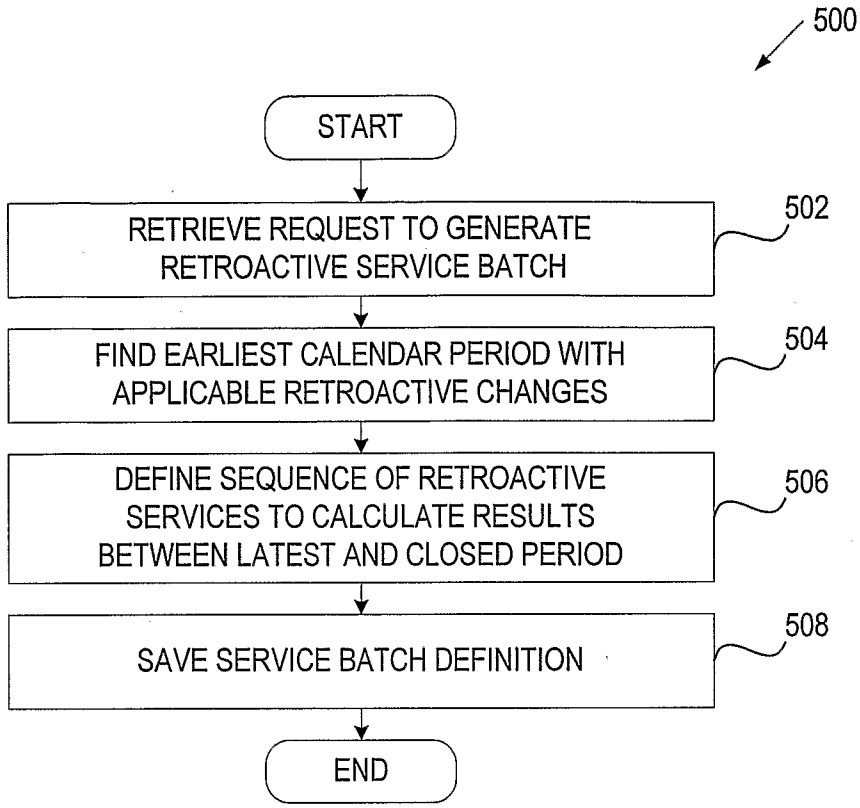

500
START
RETRIEVE REQUEST TO GENERATE RETROACTIVE SERVICE BATCH
502
FIND EARLIEST CALENDAR PERIOD WITH APPLICABLE RETROACTIVE CHANGES
504
DEFINE SEQUENCE OF RETROACTIVE SERVICES TO CALCULATE RESULTS BETWEEN LATEST AND CLOSED PERIOD
506
SAVE SERVICE BATCH DEFINITION
508
END

FIG. 5

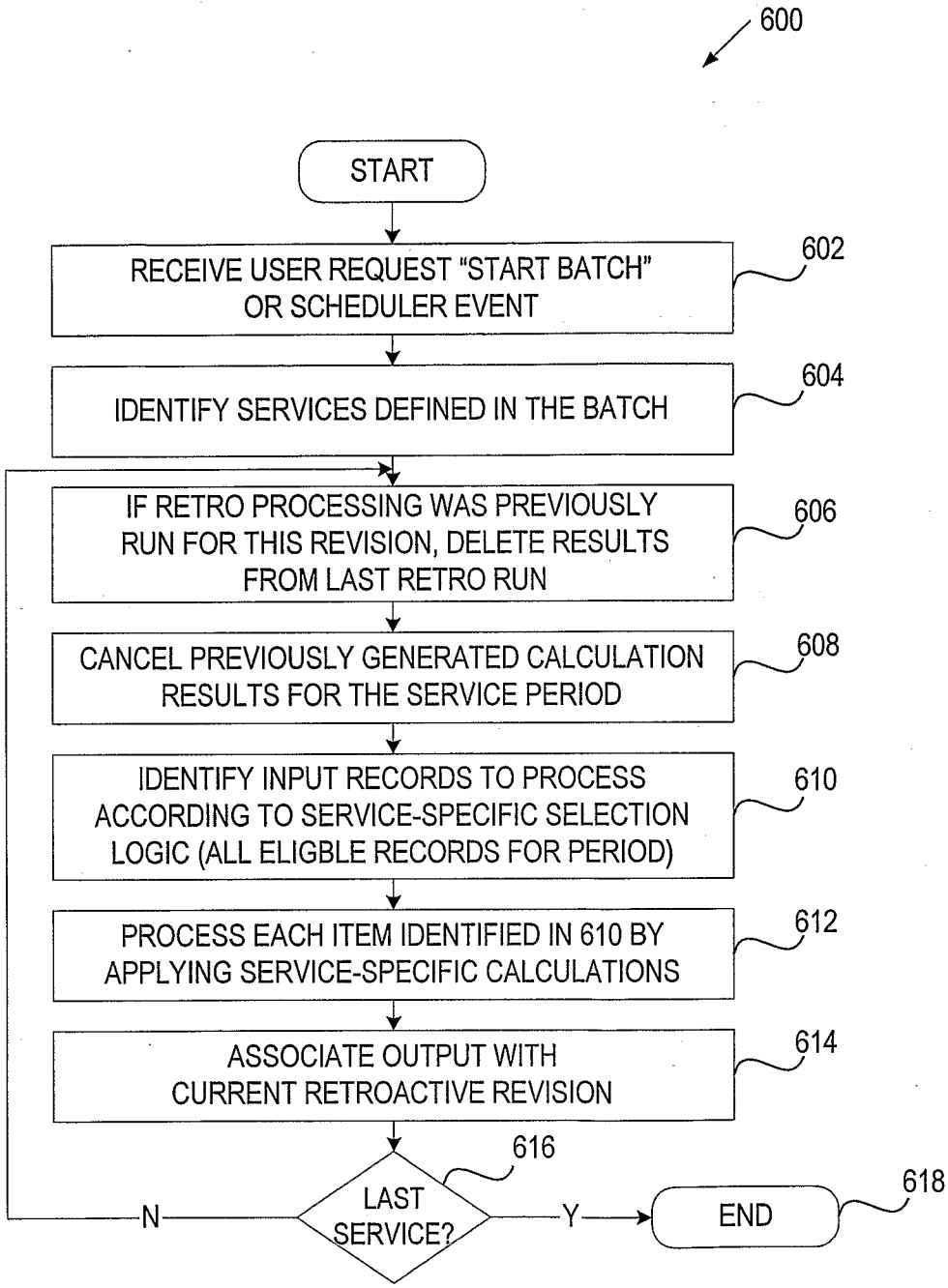

600
START
RECEIVE USER REQUEST "START BATCH" OR SCHEDULER EVENT
602
IDENTIFY SERVICES DEFINED IN THE BATCH
604
IF RETRO PROCESSING WAS PREVIOUSLY RUN FOR THIS REVISION, DELETE RESULTS FROM LAST RETRO RUN
606
CANCEL PREVIOUSLY GENERATED CALCULATION RESULTS FOR THE SERVICE PERIOD
608
IDENTIFY INPUT RECORDS TO PROCESS ACCORDING TO SERVICE-SPECIFIC SELECTION LOGIC (ALL ELIGBLE RECORDS FOR PERIOD)
610
PROCESS EACH ITEM IDENTIFIED IN 610 BY APPLYING SERVICE-SPECIFIC CALCULATIONS
612
ASSOCIATE OUTPUT WITH CURRENT RETROACTIVE REVISION
614
LAST SERVICE?
616
N
Y
END
618

FIG. 6

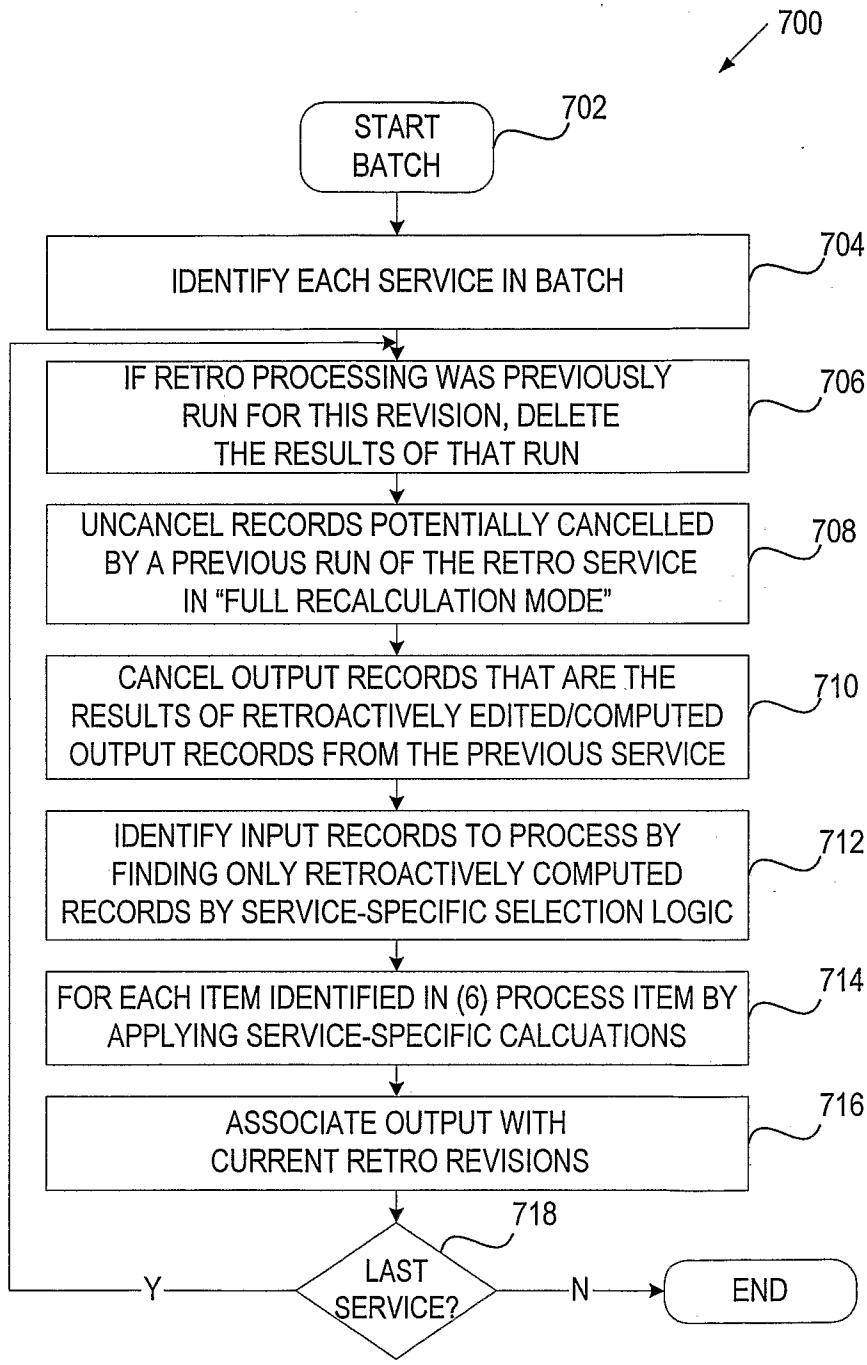

700
START BATCH
702
IDENTIFY EACH SERVICE IN BATCH
704
IF RETRO PROCESSING WAS PREVIOUSLY RUN FOR THIS REVISION, DELETE THE RESULTS OF THAT RUN
706
UNCANCEL RECORDS POTENTIALLY CANCELLED BY A PREVIOUS RUN OF THE RETRO SERVICE IN "FULL RECALCULATION MODE"
708
CANCEL OUTPUT RECORDS THAT ARE THE RESULTS OF RETROACTIVELY EDITED/COMPUTED OUTPUT RECORDS FROM THE PREVIOUS SERVICE
710
IDENTIFY INPUT RECORDS TO PROCESS BY FINDING ONLY RETROACTIVELY COMPUTED RECORDS BY SERVICE-SPECIFIC SELECTION LOGIC
712
FOR EACH ITEM IDENTIFIED IN (6) PROCESS ITEM BY APPLYING SERVICE-SPECIFIC CALCUATIONS
714
ASSOCIATE OUTPUT WITH CURRENT RETRO REVISIONS
716
LAST SERVICE?
718
Y
N
END

FIG. 7

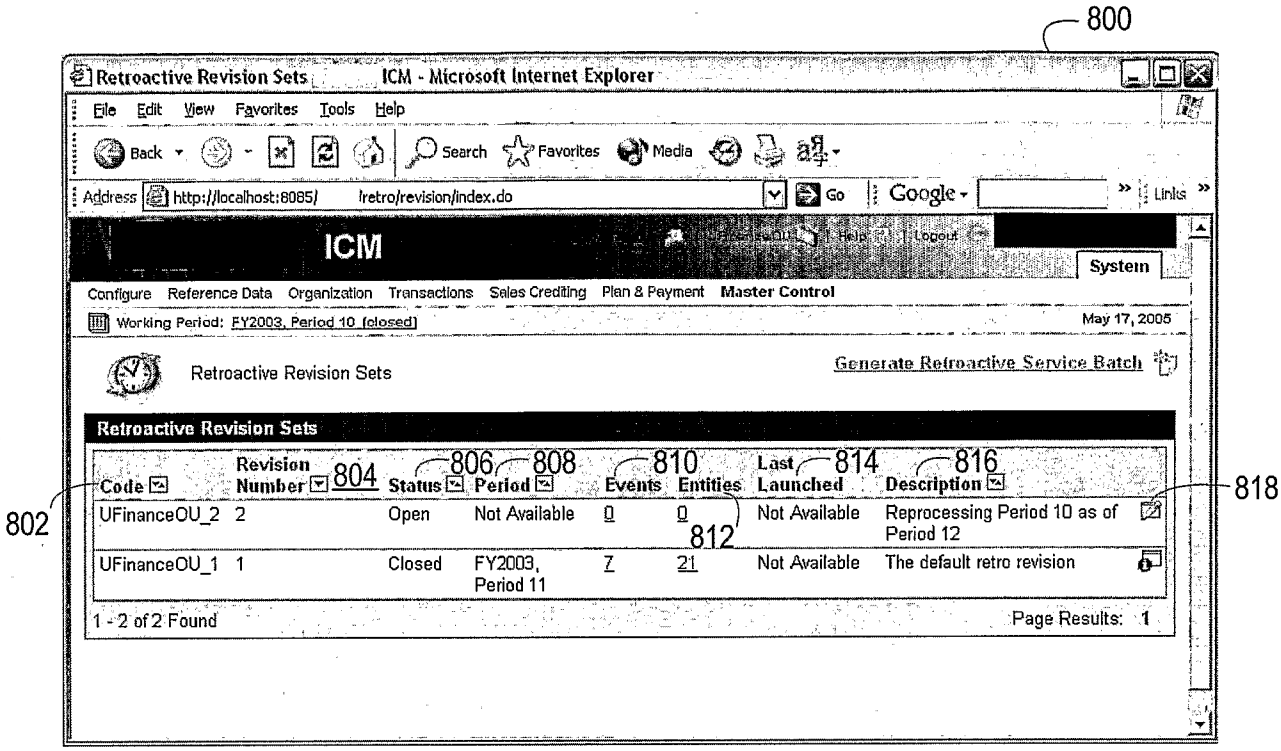

800
Retroactive Revision Sets ICM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8085/ lretro/revision/index.do Go Google Links
ICM
Logout
System
Configure Reference Data Organization Transactions Sales Crediting Plan & Payment Master Control
Working Period: FY2003, Period 10 (closed)
May 17, 2005
Retroactive Revision Sets
Generate Retroactive Service Batch
Retroactive Revision Sets
Code
Revision Number
804
Status
806
Period
808
Events
810
Entities
812
Last Launched
814
Description
816
802
818
UFinanceOU_2 2 Open Not Available 0 0 Not Available Reprocessing Period 10 as of Period 12
UFinanceOU_1 1 Closed FY2003, Period 11 7 21 Not Available The default retro revision
1 - 2 of 2 Found
Page Results: 1

FIG. 8

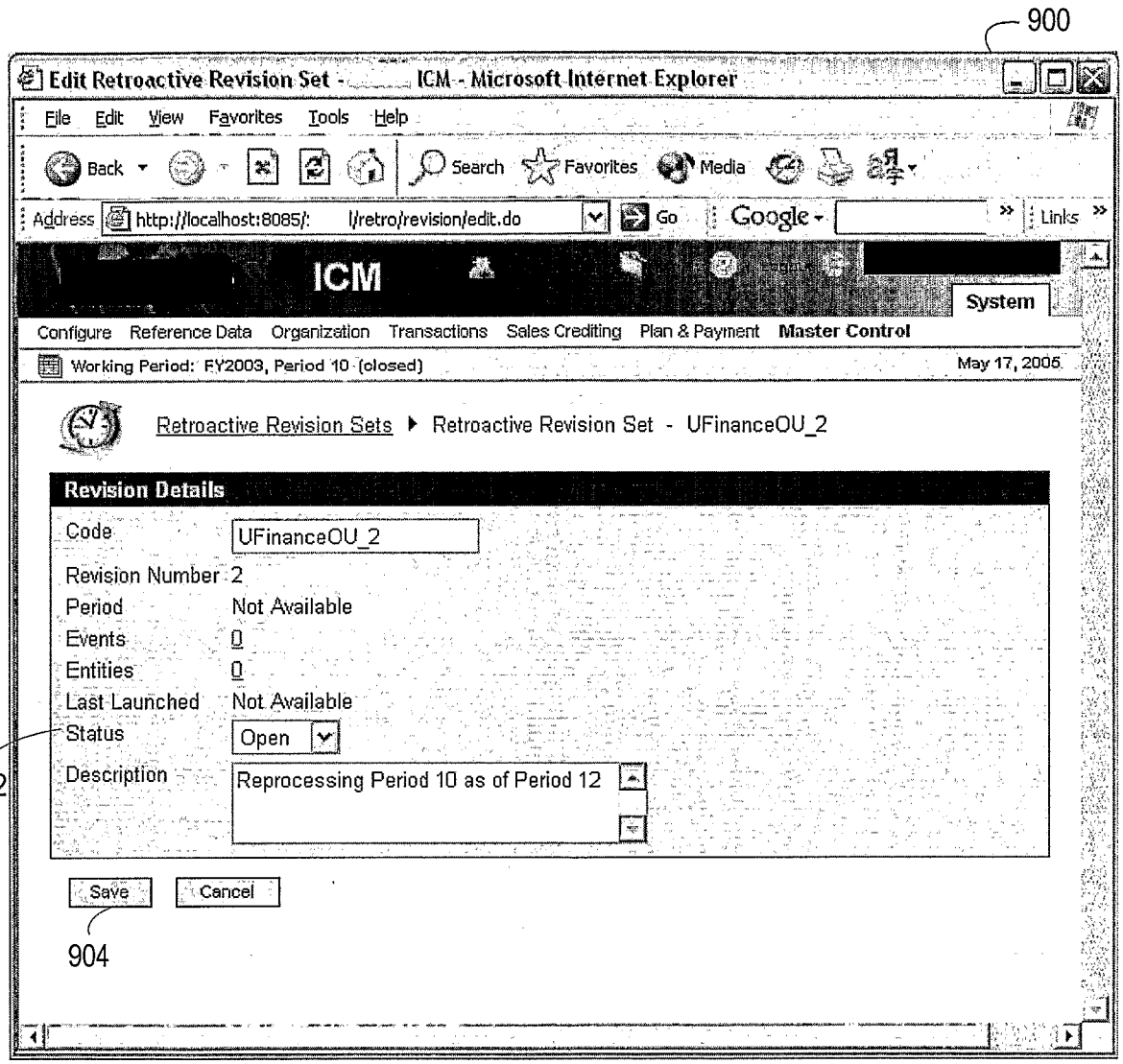

900
Edit Retroactive Revision Set - ICM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8085/ /retro/revision/edit.do Go Google Links
ICM
System
Configure Reference Data Organization Transactions Sales Crediting Plan & Payment Master Control
Working Period: FY2003, Period 10 (closed)
May 17, 2005
Retroactive Revision Sets ▸ Retroactive Revision Set - UFinanceOU_2
Revision Details
Code UFinanceOU_2
Revision Number 2
Period Not Available
Events 0
Entities 0
Last Launched Not Available
Status Open
Description Reprocessing Period 10 as of Period 12
Save
Cancel
902
904

FIG. 9

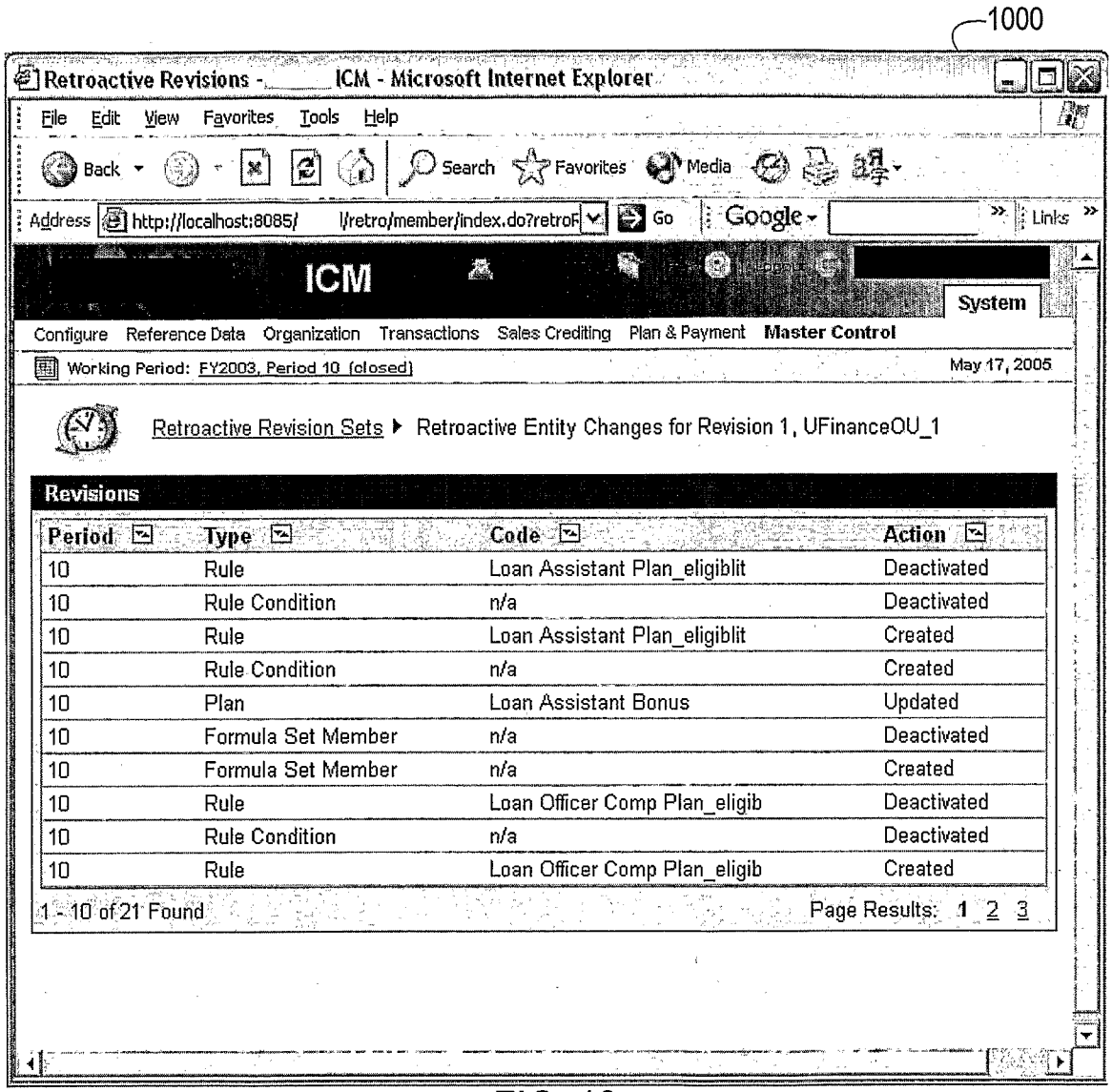

1000
Retroactive Revisions - ICM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8085/ l/retro/member/index.do?retroF Go Google Links
ICM
System
Configure Reference Data Organization Transactions Sales Crediting Plan & Payment Master Control
Working Period: FY2003, Period 10 (closed)
May 17, 2005
Retroactive Revision Sets ▸ Retroactive Entity Changes for Revision 1, UFinanceOU_1
Revisions
Period Type Code Action
10 Rule Loan Assistant Plan_eligiblit Deactivated
10 Rule Condition n/a Deactivated
10 Rule Loan Assistant Plan_eligiblit Created
10 Rule Condition n/a Created
10 Plan Loan Assistant Bonus Updated
10 Formula Set Member n/a Deactivated
10 Formula Set Member n/a Created
10 Rule Loan Officer Comp Plan_eligib Deactivated
10 Rule Condition n/a Deactivated
10 Rule Loan Officer Comp Plan_eligib Created
1 - 10 of 21 Found
Page Results: 1 2 3

FIG. 10

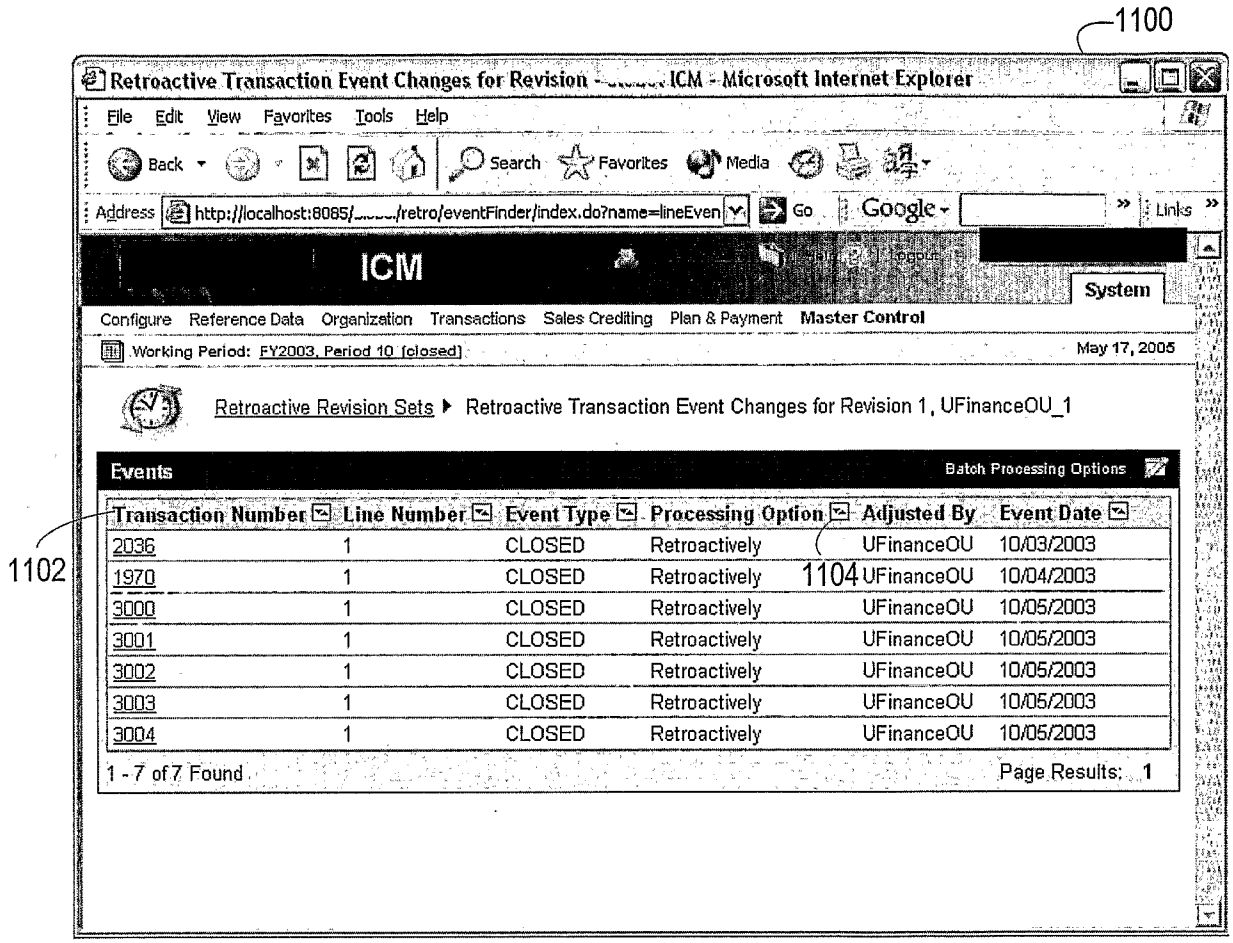

1100
Retroactive Transaction Event Changes for Revision - ICM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8085/.../retro/eventFinder/index.do?name=lineEven
Go Google Links
ICM
System
Configure Reference Data Organization Transactions Sales Crediting Plan & Payment Master Control
Working Period: FY2003, Period 10 (closed)
May 17, 2005
Retroactive Revision Sets ▸ Retroactive Transaction Event Changes for Revision 1, UFinanceOU_1
Events
Batch Processing Options
1102
1104
Transaction Number | Line Number | Event Type | Processing Option | Adjusted By | Event Date
2036 | 1 | CLOSED | Retroactively | UFinanceOU | 10/03/2003
1970 | 1 | CLOSED | Retroactively | UFinanceOU | 10/04/2003
3000 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
3001 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
3002 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
3003 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
3004 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
1 - 7 of 7 Found
Page Results: 1

FIG. 11

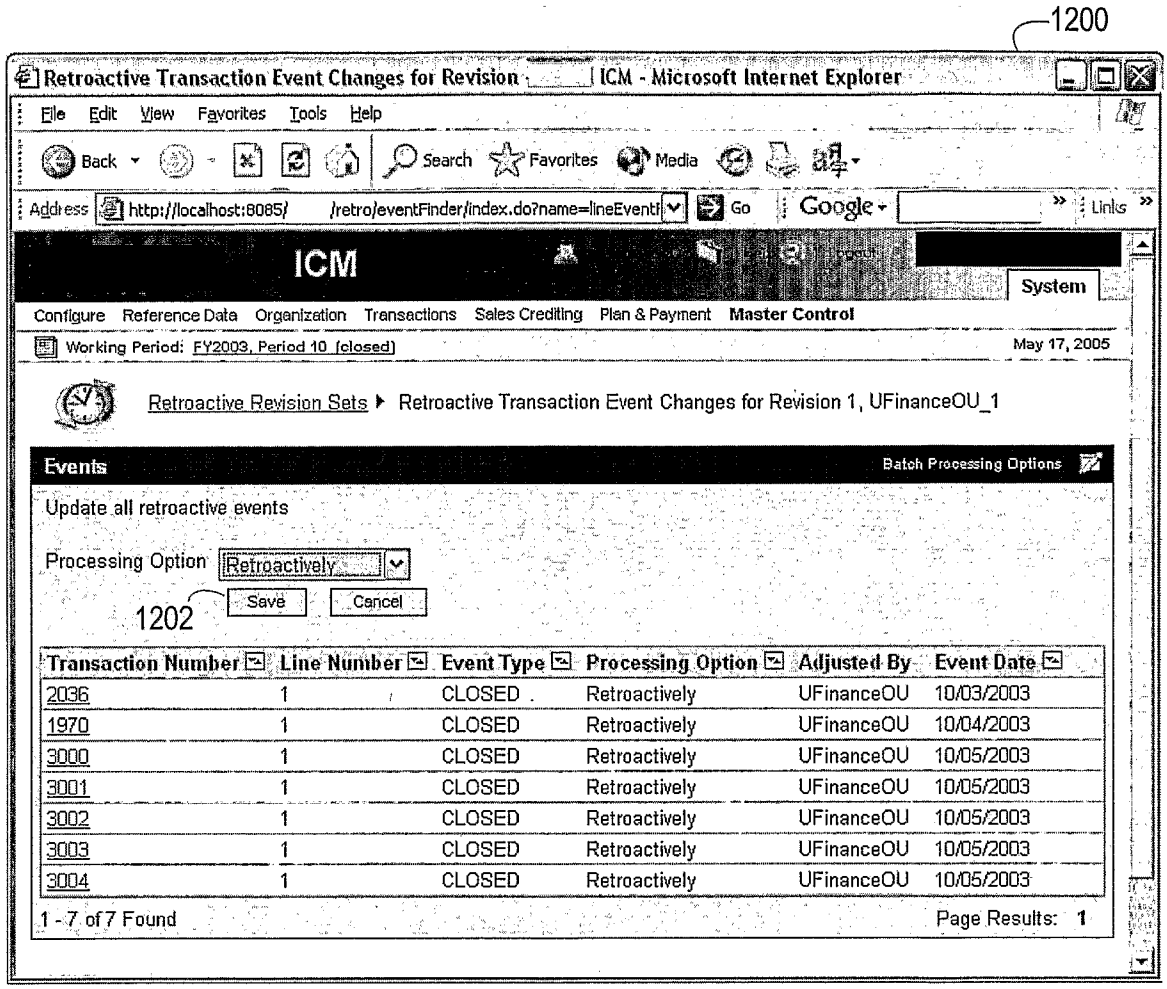

1200
Retroactive Transaction Event Changes for Revision ICM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8085/ /retro/eventFinder/index.do?name=lineEventF Go Google Links
ICM
System
Configure Reference Data Organization Transactions Sales Crediting Plan & Payment Master Control
Working Period: FY2003, Period 10 (closed)
May 17, 2005
Retroactive Revision Sets ▸ Retroactive Transaction Event Changes for Revision 1, UFinanceOU_1
Events
Batch Processing Options
Update all retroactive events
Processing Option Retroactively
1202
Save
Cancel
Transaction Number | Line Number | Event Type | Processing Option | Adjusted By | Event Date
2036 | 1 | CLOSED | Retroactively | UFinanceOU | 10/03/2003
1970 | 1 | CLOSED | Retroactively | UFinanceOU | 10/04/2003
3000 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
3001 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
3002 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
3003 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
3004 | 1 | CLOSED | Retroactively | UFinanceOU | 10/05/2003
1 - 7 of 7 Found
Page Results: 1

FIG. 12

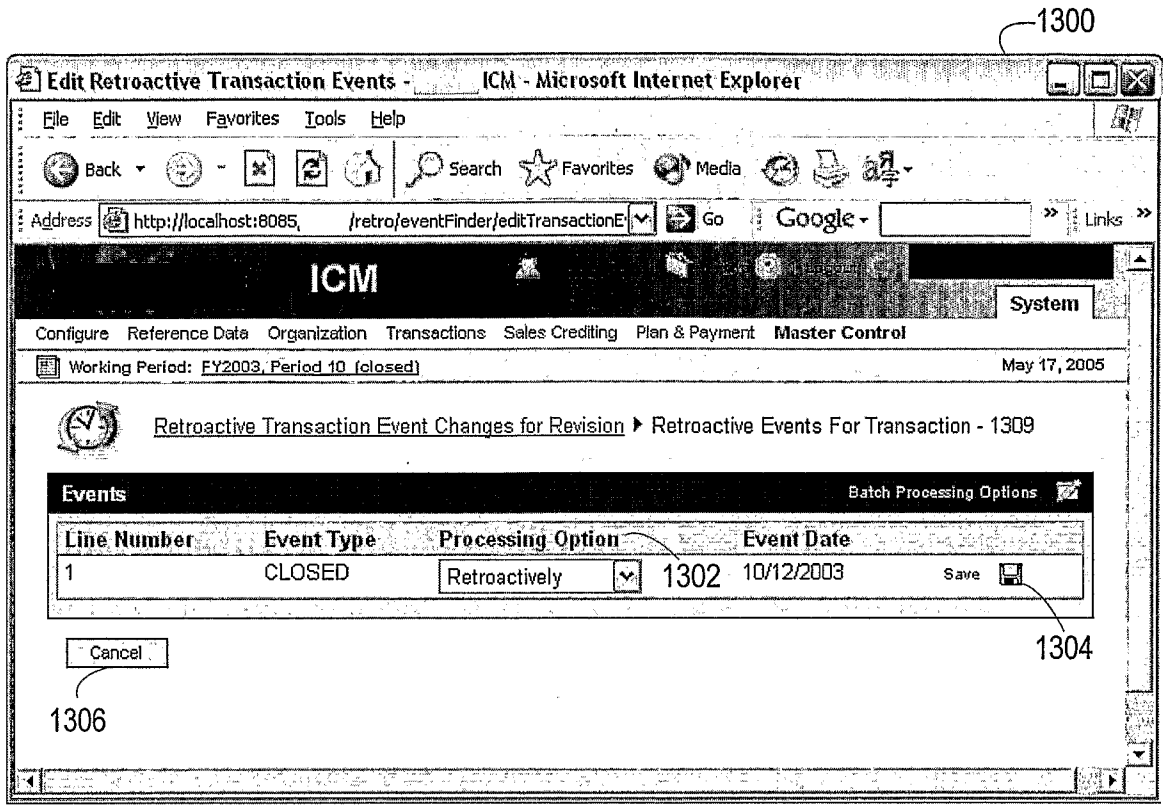

1300
Edit Retroactive Transaction Events - ICM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8085/retro/eventFinder/editTransactionE Go Google Links
ICM
System
Configure Reference Data Organization Transactions Sales Crediting Plan & Payment Master Control
Working Period: FY2003, Period 10 (closed)
May 17, 2005
Retroactive Transaction Event Changes for Revision ▸ Retroactive Events For Transaction - 1309
Events
Batch Processing Options
Line Number
Event Type
Processing Option
Event Date
1
CLOSED
Retroactively
1302
10/12/2003
Save
1304
Cancel
1306

FIG. 13

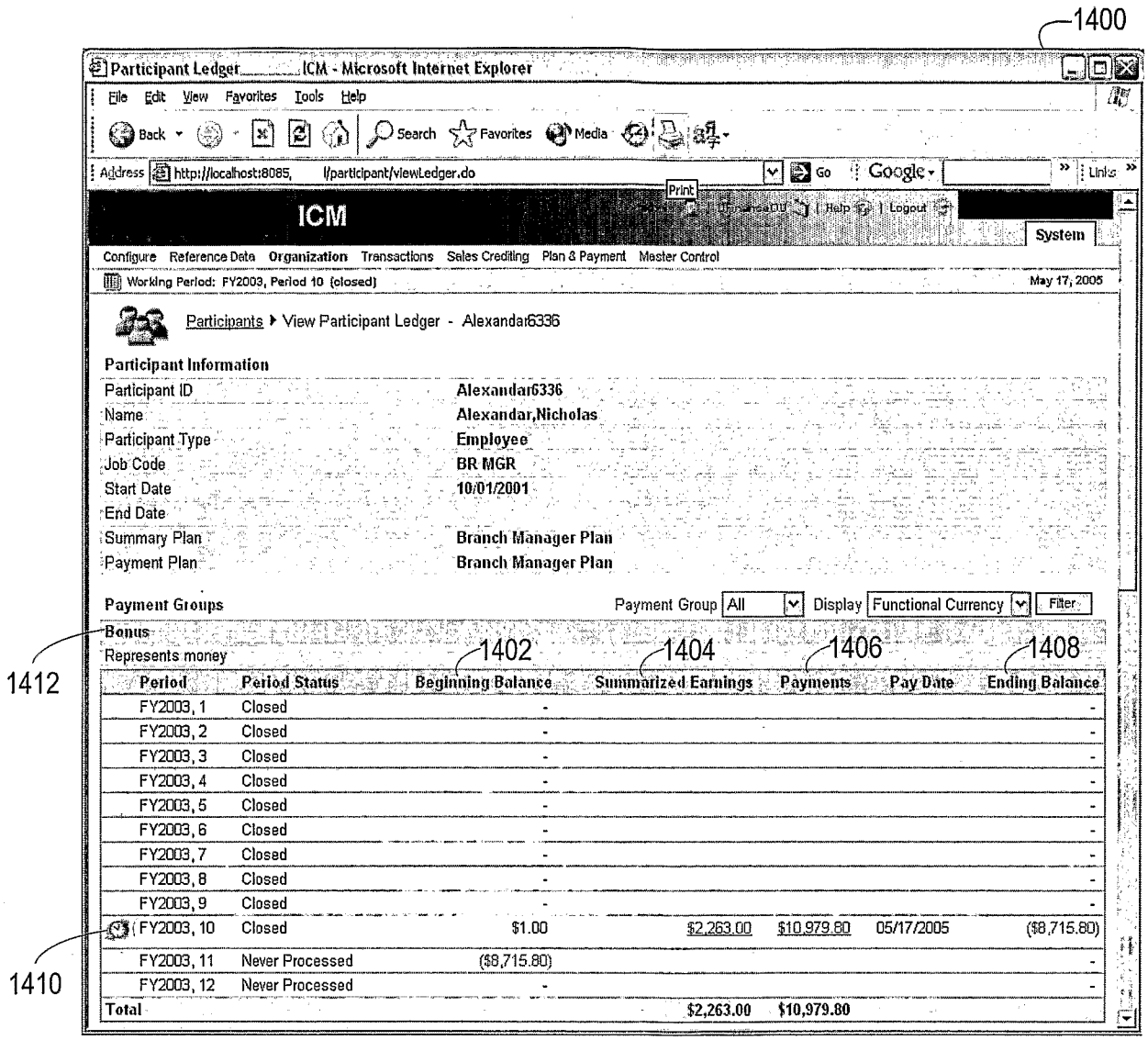

1400
Participant Ledger ICM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8085, /participant/viewLedger.do
Go Google Links
Print
ICM
Help Logout
System
Configure Reference Data Organization Transactions Sales Crediting Plan & Payment Master Control
Working Period: FY2003, Period 10 (closed)
May 17, 2005
Participants ▸ View Participant Ledger - Alexandar6336
Participant Information
Participant ID Alexandar6336
Name Alexandar,Nicholas
Participant Type Employee
Job Code BR MGR
Start Date 10/01/2001
End Date
Summary Plan Branch Manager Plan
Payment Plan Branch Manager Plan
Payment Groups
Payment Group All
Display Functional Currency
Filter
Bonus
Represents money
1412
1402
1404
1406
1408
1410
Period | Period Status | Beginning Balance | Summarized Earnings | Payments | Pay Date | Ending Balance
FY2003, 1 | Closed | - | - | - | - | -
FY2003, 2 | Closed | - | - | - | - | -
FY2003, 3 | Closed | - | - | - | - | -
FY2003, 4 | Closed | - | - | - | - | -
FY2003, 5 | Closed | - | - | - | - | -
FY2003, 6 | Closed | - | - | - | - | -
FY2003, 7 | Closed | - | - | - | - | -
FY2003, 8 | Closed | - | - | - | - | -
FY2003, 9 | Closed | - | - | - | - | -
FY2003, 10 | Closed | $1.00 | $2,263.00 | $10,979.80 | 05/17/2005 | ($8,715.80)
FY2003, 11 | Never Processed | ($8,715.80) | | | | -
FY2003, 12 | Never Processed | - | | | | -
Total | | | $2,263.00 | $10,979.80 | |

FIG. 14

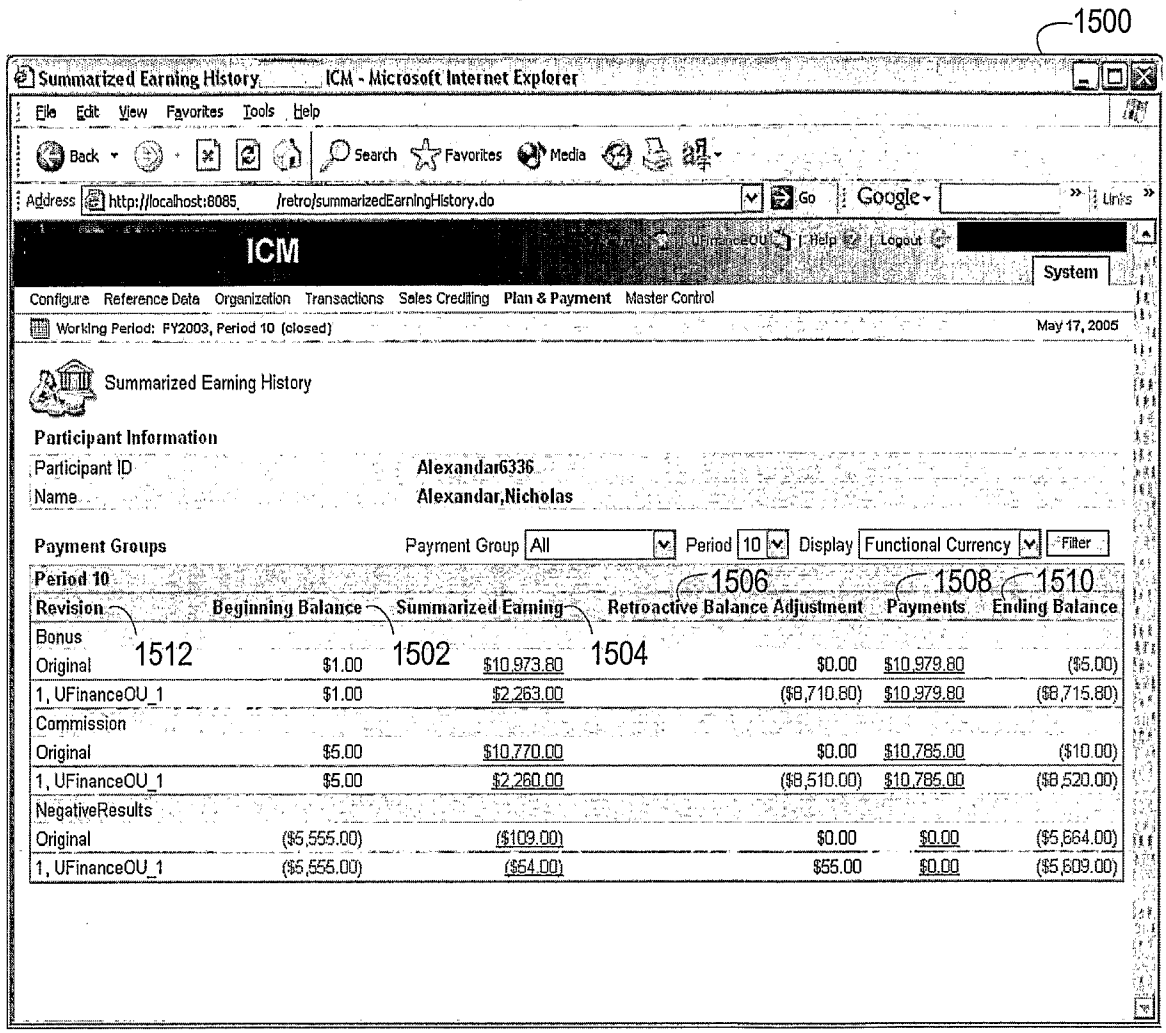
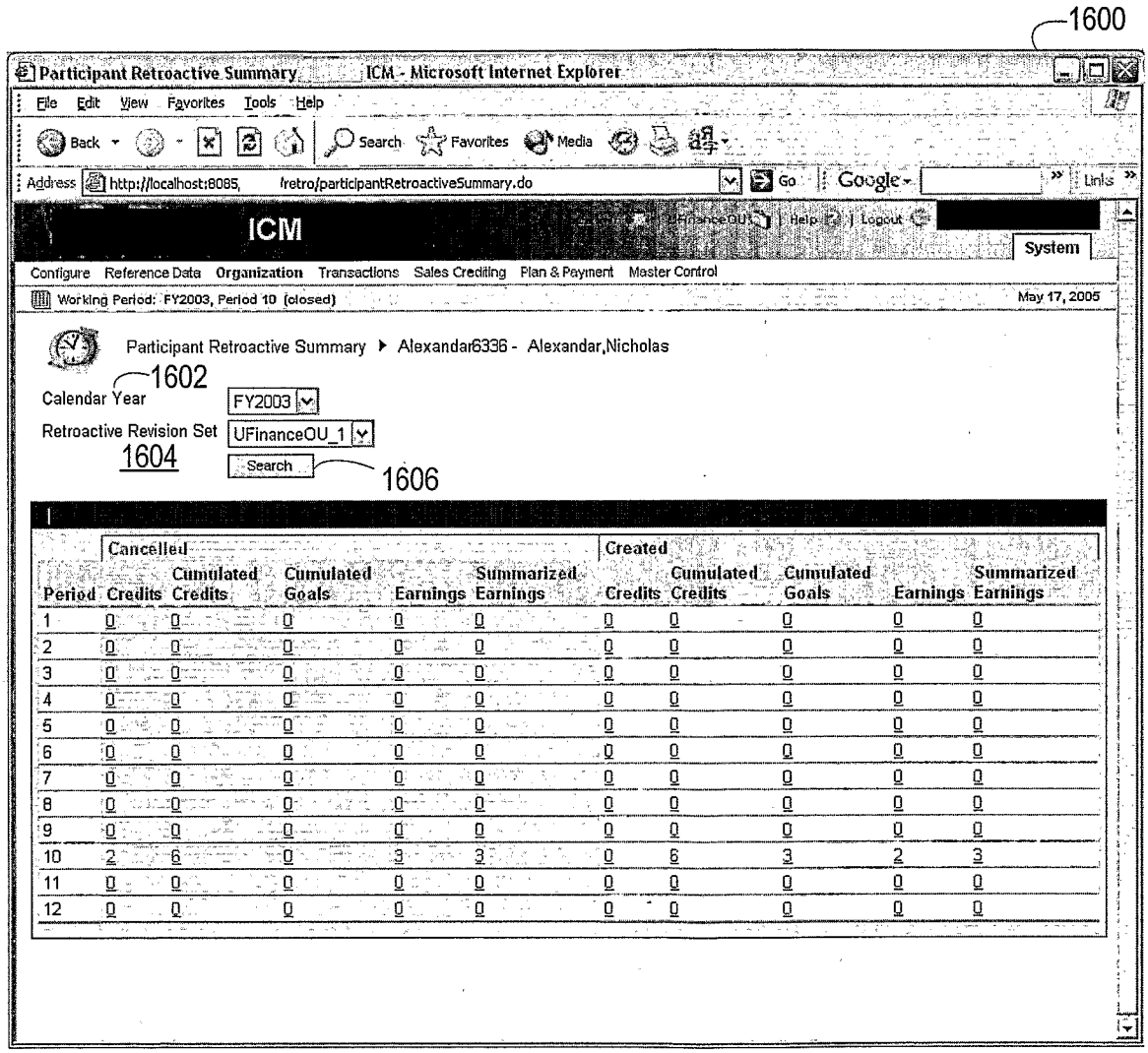

1500
Summarized Earning History ICM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8085 /retro/summarizedEarningHistory.do
Go Google Links
ICM
Help Logout
System
Configure Reference Data Organization Transactions Sales Crediting Plan & Payment Master Control
Working Period: FY2003, Period 10 (closed)
May 17, 2005
Summarized Earning History
Participant Information
Participant ID Alexandar6336
Name Alexandar,Nicholas
Payment Groups
Payment Group All
Period 10
Display Functional Currency
Filter
Period 10
1506
1508
1510
1512
1502
1504
Revision
Beginning Balance
Summarized Earning
Retroactive Balance Adjustment
Payments
Ending Balance
Bonus
Original $1.00 $10,973.80 $0.00 $10,979.80 ($5.00)
1, UFinanceOU_1 $1.00 $2,263.00 ($8,710.80) $10,979.80 ($8,715.80)
Commission
Original $5.00 $10,770.00 $0.00 $10,785.00 ($10.00)
1, UFinanceOU_1 $5.00 $2,260.00 ($8,510.00) $10,785.00 ($8,520.00)
NegativeResults
Original ($5,555.00) ($109.00) $0.00 $0.00 ($5,664.00)
1, UFinanceOU_1 ($5,555.00) ($54.00) $55.00 $0.00 ($5,609.00)

FIG. 15



	Cancelled					Created				
Period	Credits	Cumulated Credits	Cumulated Goals	Earnings	Summarized Earnings	Credits	Cumulated Credits	Cumulated Goals	Earnings	Summarized Earnings
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	2	6	0	3	3	0	6	3	2	3
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0

FIG. 16

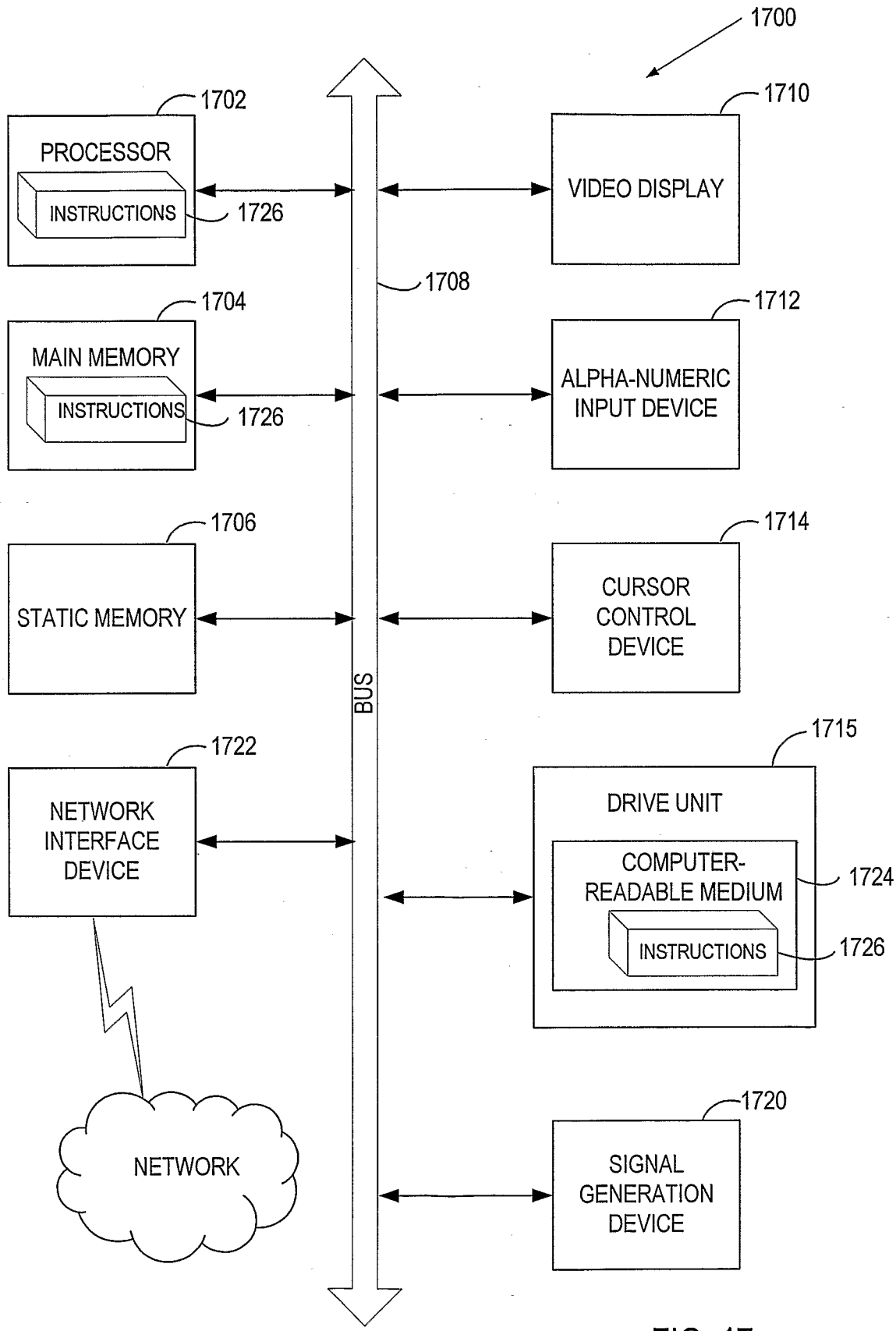

1700
1702
PROCESSOR
INSTRUCTIONS
1726
1704
MAIN MEMORY
INSTRUCTIONS
1726
1706
STATIC MEMORY
1722
NETWORK INTERFACE DEVICE
NETWORK
BUS
1708
1710
VIDEO DISPLAY
1712
ALPHA-NUMERIC INPUT DEVICE
1714
CURSOR CONTROL DEVICE
1715
DRIVE UNIT
COMPUTER-READABLE MEDIUM
1724
INSTRUCTIONS
1726
1720
SIGNAL GENERATION DEVICE

FIG. 17