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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 February 2003 (27.02.2003)

PCT

(10) International Publication Number
WO 03/017066 A2

(51) International Patent Classification^7: G06F

(21) International Application Number: PCT/US02/26709

(22) International Filing Date: 20 August 2002 (20.08.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/934,314 21 August 2001 (21.08.2001) US

(71) Applicant *(for all designated States except US)*: **APOGEE NETWORKS** [US/US]; 1455 Broad Street, 4th Floor, Bloomfield, NJ 07003 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **YATVISKIY, Oleg** [US/US]; 13 Danielle Way, Morganville, NJ 07751 (US). **HO, Ivan** [US/US]; 173 Oxford Terrace #2, River Edge, NJ 07661 (US).

(74) Agents: **DUNNAM, Michael., P.** et al.; Woodcock Washburn LLP, One Liberty Place-46th Floor, Philadelphia, Pennsylvanai 19103 (US).

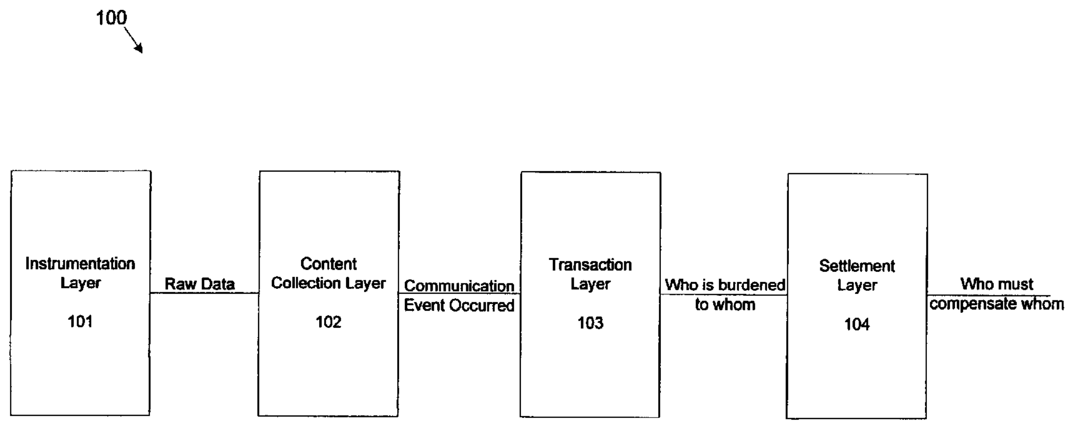
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (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*

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.

(54) Title: CONTENT DEFINITION LANGUAGE



(57) **Abstract:** The invention contemplates a method, system, and computer-readable medium having computer-executable instructions for generically transforming data and integrating multiple independent data processing components using an efficient and flexible technique.The invention method creates a set of instructions, which may be computer-executable instructions. The method then receives data, and derives data from the received data using the instructions. The instructions are independent of the data (i.e., they are not data-dependent instructions). The inventive method then modifies the instructions in real-time. The derivation of new data may include mapping the received data to the derived data. Such mapping may be accomplished using a generic lookup functionally.

WO 03/017066 A2

CONTENT DEFINITION LANGUAGE

Technical Field

The invention relates generally to the processing of data, and more particularly to efficiently and generically aggregating data available on a communication network.

Background of the Invention

Recently, the collection and processing of data transmitted over communication networks, like the Internet, has moved to the forefront of business objectives. In fact, with the advent of the Internet, new revenue generating business models have been created to charge for the consumption of content received from a data network (*i.e.*, content-based billing). For example, content distributors, application service providers (ASPs), Internet service providers (ISPs), and wireless Internet providers have realized new opportunities based on the value of the content that they deliver. As a result of this content-billing initiative, it has become increasingly important to intelligently collect and analyze content according to the business needs of the customer.

Unlike other data collection environments, communication networks like the Internet impose additional burdens on the collection and analysis process. For example, the Internet by its very nature is a network of unlimited data sources and correspondingly unlimited data types. As a result, the data collection and analysis process must be capable of understanding and processing the various types of data. Furthermore, while the Internet communicates a vast quantity of data, some of the data required by legacy systems, like billing systems for example, may not be readily provided. As a result, in order to properly account for content usage, certain characteristics or "attributes" of the transaction may have to be derived from those attributes that are provided.

Currently, deriving new attributes from available attributes is accomplished using various application specific (*i.e.,* "non-generic") methods. One method well known in the art, for example, uses database lookup techniques to map certain characteristics to other characteristics. However, these "hard-coded" lookup techniques are limited to specific purposes only, for example, mapping certain attributes only to certain other attributes. As a result, in the context of content collection and analysis, these "hard-coded" techniques are too inflexible to efficiently satisfy the ever-changing face of a communication network like the Internet. Also, these "hard-coded" techniques are too specific to operate across the many and varied content collection and billing systems found in the business community. For example, once a database lookup scenario is established, it must be rewritten to accommodate the new data sources and corresponding new data items often introduced to the Internet. Also, a particular database lookup scenario may work well for one particular content billing system, yet be ill-equipped to operate with a different billing system, or even the same billing system used by a different customer with different needs. Yet, because the required reprogramming necessary to derive new attributes is significantly time-consuming and labor-intensive, organizations often continue using outdated collection systems and techniques, or even forego altogether instituting content collection and analysis.

Therefore, there exists a need to provide a technique for allowing customers to create revenue models by determining costs from network traffic, using scalable and flexible content analysis solutions. In particular, there exists a need to flexibly transform new data characteristics from existing data characteristics. In addition, there is a need to integrate various independent data processing components in an efficient and flexible manner, through, for example an exchange and modification of generic data characteristics.

Summary of the Invention

The invention contemplates a method, system, and computer-readable medium having computer-executable instructions for generically transforming data. The inventive method creates a set of instructions, which may be computer-executable instructions. The method then receives data, and derives data from the received data using the instructions. The instructions are independent of the data (*i.e.*, they are not "data-dependent" instructions). The inventive method then modifies and optimizes the instructions in as the data is received. The derivation of the new data may include mapping the received data to the derived data. Such mapping may be accomplished using a generic lookup functionality. The generic lookup functionality may include comparing the received data to a constant value, comparing the received data to a range of values, and/or comparing the received data to a matching pattern that may include wildcard designators. Processing instructions may be written using an extensible language technique, like extensible markup language (XML). Deriving the new data may include initiating a processing function on the data, performing an arithmetic operation on the data, conditionally processing the data, and/or assigning a value to the data.

Brief Description of the Drawings

Other features of the invention are further apparent from the following detailed description of the embodiments of the invention taken in conjunction with the accompanying drawings, of which:

Figure 1 is a block diagram of a system for analyzing content transmitted over a communication network, according to the invention;

Figure 2 is a block diagram further describing the components of the system described in Figure 1;

Figures 3A and 3B provide a flow diagram further detailing the operation the system described in Figure 1;

Figure 4 is a flow diagram of a method for transforming data, according to the invention;

Figure 5 is a flow diagram describing a method for invoking the functionality of a content data language, according to the invention.

Detailed Description of the Invention

System Overview

Figure 1 is a block diagram of a system 100 for analyzing content transmitted over a communication network. Although the following description will be discussed in the context of collecting, processing and billing for data transmitted over the Internet, it should be appreciated that the invention is no so limited. In fact, the invention may be applied to any type of network, including a private local area network, for example. Also, the invention may be used for purposes other than billing for the usage of content. For example, the invention may be used to analyze the type of data transmitted over a particular network, or to determine the routing patterns of data on a network. Furthermore, the invention may be used to facilitate the intelligent collection and aggregation of data relevant to a particular industry. In addition, the invention may be used to track specific Internet protocol (IP) network resources and to detect fraud, for example.

In addition, it should be appreciated that the term "content" may be defined as data that is transmitted over the network. In the context of the Internet, content may include .mp3 files, hypertext markup language (html) pages, videoconferencing data, and streaming audio, for example. The terms "producer" and "customer" will be used throughout the description as well. Producer may refer to the primary creator or provider of the content, while customer is the primary recipient of the content. Both the producer and customer may be a human or a computer-based system.

As shown in Figure 1, an instrumentation layer 101 provides raw data to a content collection layer 102. Instrumentation layer 101 may consist of various data sources, for example, network routers. The network routers may provide information regarding the various types of routed data, including for example, data format, originating Internet protocol (IP) address, and destination IP address. One example of such information is Cisco System's NetFlow™.

Content collection layer 102 collects information about the delivery of the content, as well as the substance of the content itself. Content collection layer 102 also may sort, filter, aggregate, and store the content according to the particular needs of the end user. In effect, content collection layer is responsible for extracting meaningful information about IP traffic, and so it is provided with an understanding of the data sources in instrumentation layer 101. Content collection layer 102 also may transform the data from the plurality of sources in instrumentation layer 101 into standard formats for use in a transaction layer 103.

Content collection layer 102 is in communication with transaction layer 103. Generally, content collection layer 102 reports to transaction layer 103 that a relevant communication event has occurred and should be considered by the remainder of system 100. A communication event may be defined as any transfer of data between systems. Transaction layer 103 captures the predetermined agreements among all of the parties involved in system 100 regarding the value of the transferred content, as well as the value added by each of the individual parties in transferring such content. Therefore, transaction layer 103 is charged with understanding the nature of the parties, as well as the understanding the actions that one or more parties perform and the influence of such action on the respective parties.

Transaction layer 103 is in communication with a settlement layer 104. Settlement layer 104 captures the operations that are necessary to understand the significance of the transaction defined by transaction layer 103. For example, settlement layer 104 may rate a particular transaction by assigning a monetary value to the transaction. Settlement layer 104 also may divide the burdens and benefits of the monetary value among the relevant parties. In this way, settlement layer 104 ensures that certain parties receive a particular portion of the payment made by the other parties. Settlement layer 104 also may be responsible for delivering this burden and benefit information to the appropriate parties with the appropriate identifiers (*e.g.*, account numbers).

Figure 2 is a block diagram further describing the components of system 100. As shown in Figure 2, instrumentation layer 101 includes data sources 201-203 that each provide raw data 204-206, respectively, to collection layer 102. As discussed, data sources 201-203 may include various internetworking devices like routers, bridges, and network switches. Data sources 201-203 provide raw data to an input data adapter 207. Accordingly, input data adapter 207 understands the operation of and the data provided by data sources 201-203. Although one input data adapter is shown in Figure 2, it should be appreciated that more than one input data adapter may be used in system 100. For example, each data source may have a dedicated input data adapter.

Input data adapter 207 understands the incoming data and extracts the data into the appropriate fields. This field-delimited data, called flow objects 208, are sets of attributes. The attributes may be any characteristics that are provided by, or can be derived from, raw data 204-206 provided by data sources 201-203, respectively. For example, flow objects 208 may include a set of attributes describing the source and destination, including source IP address, destination IP address, source interface, and destination interface. Because input data adapter 207 is charged with understanding raw data 204-206 from data sources 201-203, as well as the required flow objects 208 of system 100, it is capable of transforming the raw data to the flow objects, where the flow objects may all be of a standard format.

CDL 209 may be invoked through any number of techniques. For example, because input data adapter 207 provides flow objects 208 to CDL 209, CDL 209 may be invoked by input data adapter 207 for each new flow object 208. Alternatively, or in addition, CDL 209 may be invoked automatically when a request by filter 212 or by aggregator 213 when an attribute has to be derived using CDL 209.

Content data language 209 may transform the attributes in flow objects 208 into other attributes that are desired by a particular customer. For example, content data language 209 may derive a network identifier attribute that is not readily available from a data source, from a source address attribute and a destination address attribute that are provided by flow object 208 attributes from input data adapter 207. This derivation may be based on a customer's desire to determine which network conveyed the transaction between the source and the destination. Therefore, following this example, content data language 209 will know to extract the source address attribute and the destination address attribute from flow objects 208.

Content data language 209 may perform other functions as well. For example, content data language 209 may perform a generic lookup function 219 that is built into content data language 209. Generally, generic lookup 219 describes a technique for mapping any number of attributes to any number of other derived attributes. For example, generic lookup 219 may be used to map a unique resource locator (URL) attribute to a particular content-type attribute. Content data language 209 will be discussed in greater detail.

Content data language 209 also is in communication with a correlator 211. Generally, correlator 211 connects the many daily network content events from various network devices, like routers for example. Often, the connected data may come from distinctly different data sources at distinctly unrelated times. Correlator 211 allows this data to be intelligently connected to each other, regardless of how different the sources or of how disparate the time received. For example, a Netflow™ enabled router and a Radius™ enabled network access switch may each provide certain data that is relevant to one particular transaction. However, because portions of the data come from different devices, the data may arrive at system 100 at different times, and in different formats. Also, because each provides data that is necessary to complete one transaction, the data from each cannot be considered separately. Correlator 211 allows this data to be intelligently grouped regardless of the format of the data or of the time the data pieces are received.

Furthermore, correlator 209 may rearrange the order of the received flow objects 208 to suit the needs of the remainder of system 100. By performing such correlation without having to first store all of the data on a disk (*e.g.*, a database), significantly faster processing is achieved. Correlator 209 may perform this correlation in real-time, for example.

Although system 100 has been described using content data language 209 and correlator 211, it should be appreciated that flow objects 208 may proceed directly to a filter 212, if no correlation is required and if no attribute derivation is required, for example. Filter 212 analyzes flow objects 208 to ensure that the provided attributes are desired by system 100. If flow objects 208 are not needed (*i.e.*, a mismatch), filter 212 may prevent flow objects 208 from passing to an aggregator 213. Also, although filter 212 has been shown as a separate device in system 100, it should be appreciated that the functionality of filter 212 may be incorporated into aggregator 213.

Filer 212 passes the matching flow objects to aggregator 213. Aggregator 213 may provide additional filtering and classification of the multitude of daily network content transactions, based on user criteria. Aggregator 213 may perform such filtering and classification in real-time. Aggregator 213 will be discussed in greater detail with reference to Figures 4-7. Aggregator 213 provides the filtered and classified information to an output data adapter 214. Output data adapter 214 may convert the data from aggregator 213 into one or more content detail records for storage in CDR database 215. Therefore, CDR database 215 stores a complete description of a content event.

CDR database 215 passes the CDR onto a transaction component 216. Transaction component 216 captures the predetermined agreements among all of the parties involved in system 100 regarding the value of the transferred content, as well as the value added by each of the individual parties in transferring such content. Therefore, transaction component 216 understands the nature of the parties and the actions that one or more parties perform and the influence of such action on the respective parties.

Transaction component 216 provides the transaction information to a rating component 217. Rating component 217 provides a weight or significance (*e.g.*, a price) to the transaction, so as to provide a tangible value to the transaction. Rating component 217 may make this determination based on various metrics including the type of the content, the quantity of content consumed, the place and time delivered, or the quality of the content, for example. Therefore, rating component 217 allows system 100 to provide some contextual value, indicting the significance or relevance that certain content or information has to the individual customer.

Rating component 217 provides the rated transaction to a presentment component 218. Presentment component 218 may provide the capability for a customer 220 to view their real-time billing information, for example, over the network. Presentment component 218 also may attach relevant identifiers to the bill (*e.g.*, account numbers, etc.).

Figures 3A and 3B provide a flow diagram further detailing the operation 300 of system 100. As shown in Figure 3A, in step 301, raw data 204-206 is received from data sources 201-203. In step 302, input data adapter 207 converts raw data 204-206 to flow objects 208, where flow objects 208 are sets of attributes, determined from raw data 204-206. In step 303, it is determined whether there is a need to derive new attributes from the existing attributes found in flow objects 208. If there is such a need, in step 304, CDL 209 is used to derive new attributes from existing attributes. Also, as discussed above, attributes may be correlated by correlator 211.

In step 305, flow objects 208 are filtered by filter 212. In step 306, the matching flow objects (*i.e.,* those passed by filter 212) are further filtered and classified by aggregator 213 (as discussed more fully with reference to Figures 4-7). In step 307, output data adapter 214 converts the data aggregated by aggregator 213 to a format compatible with transaction layer 103.

As shown in Figure 3B, in step 308, output adapter 214 may format the aggregated data into one or more content data records for storage in CDR database 215. In step 309, transaction component 216 captures the predetermined agreements among all of the parties and the value added by each of the individual parties. In step 310, the CDR is rated based on predetermined metrics (*e.g.,* time of transmission, quality of content).

In step 311, a bill is presented to the customer.

Content Data Language Overview

Figure 4 is a flow diagram of a method 400 of transforming data. In step 401, a set of instructions is created. The instructions may be computer-readable instructions written using XML, for example. In step 402, raw data 204-206 is received from a communication network. Raw data 204-206 may be received by input data adapter 207, or may be received directly to CDL 209. In step 403, new data is derived from the received data using the instructions created in step 401. The instructions created in step 401 are not based on the received data. The derivation of the new data may include mapping the received data. Such mapping may include a generic lookup functionality 219. Generic lookup 219 may compare the received data to a constant value, a range of values, and/or a matching patter that includes the use of wildcards. For example, the generic lookup functionality may include locating a match for the received data in a set of constant values, locating a match for the received data in a set of ranges of values, and/or locating a match for the received data in a set of matching patterns that may include wildcard designators. The derivation of new data in step 403 also may include initiating a processing function on the data, performing an arithmetic operation on the data, assigning a value to the data, and/or conditionally processing the data. In step 404, the instructions created in step 401 are modified and optimized in real-time.

Figure 5 is a flow diagram describing a method 500 for invoking the functionality of CDL 209. In step 501, raw data 204-206 is received by content collection layer 102. In step 502, raw data 204-206 may be converted to flow objects 208 by input data adapter

207. In step 503, it is determined whether flow objects 208 provide the data required by the billing system, for example, of customer 220. If flow objects 208 provide the data necessary to operate the customer's system, the data is sent to filter 212 and aggregator 213, in step 504. In step 506, it is determined whether aggregator 213 requires data that is not available from flow objects 208. If flow objects 208 provide filter 212 and/or aggregator 213 with the necessary data, the data is provided to the customer's billing system in step 507. If, on the other hand, filter 212 and/or aggregator 213 requires data that is not available in flow objects 208, in step 505 flow objects 208 are provided to CDL 209. Also, if flow objects 208 do not provide the data required by the customer's system (as determined in step 503), in step 505 flow objects are provided to CDL 209. In step 508, the data required by filter 212 and/or aggregator 213 and the customer's system (but not provided by flow objects 208) is derived by CDL 209. Derivation of the new data by CDL 209 may be accomplished using a number of techniques including mapping using generic lookup, process functions, arithmetic operations, conditional processing, and/or assignment.

As discussed, CDL 209 may serve to provide configuration-driven transformations of the attributes provided by flow objects 208. Also, CDL 209 may serve to derive new attributes from the existing attributes provided by flow objects 208. The transformations performed by CDL 209 represent the need for certain attributes that may be desired by a particular customer. This is especially pertinent in the context of content-based billing on a diverse network, like the Internet. Specifically, transformation of provided attributes to other derived attributes may be necessary to accommodate the attributes that are required by legacy billing systems.

In just one example, flow objects 208 may provide CDL 209 with attributes defining a data source, a source address attribute, and a destination address. However, a particular legacy customer billing system, for example, may require identifying the particular network that conveyed the transaction between the source and the destination. In other words, the legacy system requires an attribute that is not directly provided by flow objects 208. Therefore, CDL 209 acts to derive this information from the available flow objects 208. Following the above example, therefore, CDL 209 will know to extract the source address attribute and the destination address attribute from flow objects 208, in order to identify the subject network attribute required by the legacy system.

CDL 209 may perform other functions as well, like generic lookups, data calls to both built-in and customer defined functions, arithmetic operations, conditional processing, and attribute value assignment. For example, CDL 209 may perform a generic lookup function 219 that is built into CDL 209. Generally, generic lookup 219 describes a technique for mapping any number of given attributes to any number of other derived attributes. For example, generic lookup 219 may be used to map a uniform resource locator (URL) (*e.g.*, CNN.com) attribute to a particular content-type attribute (*e.g.*, news).

The functionality of CDL 209 is based on a set of rules that may be expressed through XML, for example (discussed with reference Table 2). Generally, each rule has certain elements that may include attributes, attribute values, and child elements. A child element is an element that it is dependent on the outcome of another element, called the parent element. For example, if a parent element is satisfied, a child element may be processed. Also, input data adapter 207 may be associated directly with one or more rules, and each rule may have a defined "scope" that is used by input data adapter 207 to decide when to apply a particular rule. For example, the scope of the rule may dictate that the rule is executed before the transaction data is processed, before the data passes through filter 212 and aggregator 213, and/or when filter 212 and aggregator 213 request a value of an attribute that is derived using the particular rule.

CDL Functionality

CDL 209 is able perform a number of functions, including function calls, mathematical operators, lookup invocations, conditional processing and assignment. It should be appreciated, however, that the operation of CDL 209 is not limited to the described functionality. Instead, the described functionality is meant to provide examples of the ability of CDL 209 to derive new attributes from given attributes.

A function call is an invocation of an C++ or Java function (external to system 100) that modifies existing attributes. The function call also may add new attributes given a set of input parameters that can be represented by other attributes, temporary variables, and/or constants, for example. Such C^{++} and Java functions typically reside in customer-provided pre-configured extension modules (*i.e.*, shared libraries and Java's

"Java Archive" or "JAR" file format). For example, a function called "URL_to_DOMAIN" may be called by CDL 209 to convert a URL attribute (provided by input data adapter 207) to a domain name. CDL 209 may include an application program interface (API) capable of supporting various operating systems, including Windows NT™, Solaris™, and Linux™ for example, as well as several programming languages, such as Java and C^{+}+, for example. Sample formatting for a function call will be described with reference to Table 3.

CDL 209 also is capable of performing mathematical operations, like addition, subtraction, division and multiplication. The mathematical operations may be performed on attributes, temporary variables, and constants, for example. For example, given a value for an attribute called "Input Bytes" and a value for an attribute "Output Bytes," one may derive an attribute "Bytes" that will be assigned a value of the sum of the values of attributes "Input Bytes" and "Output Bytes." The following is an example of how such an arithmetic operation would appear in CDL 209:

```
<operation name="+" result="BYTES" left_param="INPUT_BYTES"
right_param="OUTPUT_BYTES" />
```

Sample formatting for such operations will be described with reference to Table 3.

CDL 209 also is capable of invoking lookups, generically. Generic lookup functionality allows the mapping of multiple input attributes to multiple output attributes. The mapping of the input attributes may be based on an exact value, a numeric range, or a matching pattern. Non-overlapping numeric ranges may be provided for any numeric input attribute, while match patterns may be provided for any string input attribute. Lookups that use match patterns can use pre-configured caches for faster lookup. CDL 209 also supports multiple matches that may result from a lookup. Generic lookups will be discussed in greater detail.

CDL 209 also is capable of performing conditional processing, including for example, simple conditional operators, like equal to, greater than, and/or less than. Some of the conditional operators may be used to compare strings, while others may be used to compare numbers. CDL 209 also is capable of performing assignment, which is the direct mapping of a particular valve to an attribute and/or a temporary variable.

Also, CDL 209 may include a multi-language API in order to enable the extensibility of the functionality of CDL 209, for example, by incorporating computer code from shared libraries. Such extensibility may be user-defined functions written in C^{++} or Java™, for example. The API interface also may allow components within system 100 to find a particular rule object by its name, invoke a rule, discover the attributes that a particular rule uses, and/or discover the attributes that a particular rule modifies or derives, for example. In addition, the API allows access to the values of the constants, temporary variables, and/or attributes for the rules in CDL 209. Also, the API allows access to modify the values of a particular rule's temporary variables and attributes. Also, it should be appreciated that the additional functionality that the API permits may be created in different languages and executed on remote hosts from the resident processing.

CDL Configuration

The rules in CDL 209 are configured to operate on the attributes provided by flow objects 208 that are created by input data adapter 207. In order for the rules of CDL to operate on a particular attribute, the attribute will be defined within system 100. Table 1 provides just one example of an electronic file that defines the attributes. It should be appreciated that Table 1 is just an example of such a file and is not meant to be an exclusive representation.

Name	Attribute ID	Type of Attribute
DOMAIN	1	APO_TYPE_STRING
HIT_BYTES	2	APO_TYPE_LONG_LONG
MISS_BYTES	3	APO_TYPE_LONG_LONG
TIME_STAMP	4	APO_TYPE_LONG
BYTES	5	APO_TYPE_LONG_LONG
URL	6	APO_TYPE_STRING
DOMAIN	7	APO_TYPE_STRING
CONTENT_TYPE	8	APO_TYPE_STRING
HIT_FLAG	9	APO_TYPE_STRING
URL_EXTENSION	10	APO_TYPE_STRING
CONTENT_PROTOCOL	11	APO_TYPE_STRING

Table 1

The first column in Table 1 "Name" represents a name for the attribute, by which it may be referred throughout system 100. The second column in Table 1 "Attribute ID" contains unique attribute identification that further distinguishes and catalogues a particular attribute to the components of system 100. The third column in Table 1 "Type of Attribute" contains the type of the attribute, which further categorizes a particular attribute based on its required characteristics. The following are examples of typical attribute types, along with their byte lengths:

APO_TYPE_INT_CHAR	= An initial character that is 1 byte
APO_TYPE_SHORT	= A short character that is 2 bytes
APO_TYPE_LONG	= A long character that is 4 bytes
APO_TYPE_LONG_LONG	= A longer character that is 8 bytes
APO_TYPE_IP_ADDRESS	= A representation of the IP address that is 4 bytes
APO_TYPE_DOUBLE	= A floating point number
APO_TYPE_STRING	= A text string

The rules in CDL 209 may be coded using a variety of extensible techniques, including using extensible markup language (XML) for example. Table 2 is just one example of the rules in CDL 209, using XML. It should be appreciated, however, that Table 2 is just an example of such a file and is not meant to be an exclusive representation. Also, it should be appreciated that the temporary variables in Table 2 are provided to further describe the invention and are not meant to represent exclusive variable values.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE cdl_rules SYSTEM "cdl_rules.dtd" >

<cdl_rules>

<rule name="rule1" description="sample rule">

<temp_var_declaration name="$TEMP1" type="NETFLOW_TYPE_LONG" />
```

```
06 <temp_var_declaration name="$TEMP2" type="NETFLOW_TYPE_SHORT" />

07 <function result="DOMAIN" name="url_to_domain" params="URL" />
08 <lookup name="DOMAIN_TO_CONTENT" />

09 <conditional_rule left_param="HIT_FLAG" operator="eq" right_param="'HIT'">

10 <true_rule>
11 <assignment result="HIT_BYTES" param="BYTES" />
12 </true_rule>

13 <false_rule>
14 <assignment result="MISS_BYTES" param="BYTES" />
15 </false_rule>

16 </conditional_rule>
```

Table 2

Line 04 indicates that the rule is named "rule1." The Line 07 "url_to_domain" command identifies the rule as being directed toward converting the URL attribute (provided by data adapter 207 in flow objects 208) to a domain name. For example, the URL attribute provided in flow objects 208 may be www.cnn.com\sports\baseball. "Rule 1" may act to convert the received URL to "CNN," which is the domain name associated with the received URL. The result of the rule, as indicated in line 07, will be stored as an individual attribute as "DOMAIN" in flow objects 208. In line 08, lookup name = "DOMAIN_TO_CONTENT," takes the domain determined in line 07 (*e.g.*, "CNN") and determines the type of content of (*e.g.*, news).

In line 09, a conditional function determines if the value of the received (or derived) attribute "HIT_FLAG" is set to "HIT." If the value of the "HIT_FLAG" attribute is set to "HIT", line 11 assigns the attribute "BYTES" to the attribute "HIT_BYTES." If, on the other hand, the "HIT_FLAG" attribute is not set to "HIT", line 14 assigns the attribute "BYTES" to the attribute "MISS_BYTES."

The rules of CDL 209 may be made to adhere to certain requirements. Table 3 provides one example of a file representing such a set of requirements. It should be appreciated, however, that Table 3 is just an example of such a file and is not meant to be an exclusive representation.

```
01    <!ELEMENT cdl_rules (rule*) >
02    <!ELEMENT rule
03    (temp_var_declaration*,(function|assignment|operation|lookup|conditional_rule)+
      ) >
04    <!ATTLIST rule
05          name          ID       #REQUIRED
06          description   CDATA       #REQUIRED
07          scope         CDATA       #IMPLIED
08    >
09    <!ELEMENT temp_var_declaration EMPTY>
10    <!ATTLIST temp_var_declaration
11          name CDATA #REQUIRED
12          type
13          (APO_TYPE_INT_CHAR|APO_TYPE_SHORT|APO_TYPE_LONG|AP
            O_TYPE_LONG_LONG|APO_TYPE_IP_ADDRESS|APO_TYPE_DOU
            BLE|APO_TYPE_STRING) #REQUIRED
14    >
15    <!ELEMENT conditional_rule (true_rule,false_rule?) >
```

```
<!-- operators supported ==,<=,<,eq -->
<!ATTLIST conditional_rule
        left_param    CDATA       #REQUIRED
        operator      CDATA       #REQUIRED
        right_param   CDATA       #REQUIRED
>
<!ELEMENT true_rule
((function|assignment|operation|lookup|conditional_rule)+)>
<!ELEMENT false_rule
((function|assignment|operation|lookup|conditional_rule)+)>

<!ELEMENT function EMPTY >
<!ATTLIST function
        result  CDATA             #IMPLIED
        name  CDATA               #REQUIRED
        params        CDATA              #IMPLIED
>

<!ELEMENT assignment EMPTY >
<!ATTLIST assignment
        result  CDATA             #REQUIRED
        param  CDATA              #REQUIRED
>

<!ELEMENT operation EMPTY >

<!-- operators supported +,-,/,* -->
<!ATTLIST operation
        result        CDATA       #IMPLIED
        left_param    CDATA       #REQUIRED
        operator      CDATA       #REQUIRED
        right_param   CDATA       #REQUIRED
>

<!ELEMENT lookup EMPTY >
<!ATTLIST lookup
        name  NMTOKEN             #REQUIRED
        substs  CDATA             #IMPLIED
>
```

Table 3

In Table 3, the temporary variable names begin with '$', while the constants are identified in single quotes. The built-in and user-defined functions may be stored within system 100, or they may be remote components. The conditional rule element beginning in line 15 represents a comparison operation. If the comparison operation returns true (*i.e.,* the comparison is satisfied), child element "true_rule" gets processed. If, on the other hand, the comparison operation returns false (*i.e.,* the comparison is not satisfied), the child element "false_rule" gets processed. Line 18's "left_param" attribute represents a parameter on the left side of a comparison operation and may be an attribute, a temporary variable, or a constant. Line 19's "operator" attribute is a comparison operation that is performed, where operations equal to (= =), greater than or equal to (<=), and greater than (<) are supported for numeric data types, as indicated in line 16. Operation "eq" in line 16 is used to compare strings. Line 20's "right_param" attribute represents a parameter on the right side of comparison operation, and may be an attribute, a temporary variable, or a constant.

In line 15, child element "true_rule" represents a group of functions, assignments, operations, lookups, and conditional rule elements that are processed if the comparison in the parent element "conditional_rule" returns true, while element "false_rule" represents a group of functions, assignments, operations, lookups, and conditional rule elements that are processed if the comparison in the parent element "conditional_rule" returns false.

As discussed, rules for CDL 209 may include function calls, mathematical operators, lookup invocations, conditional processing and assignments. Table 3 provides an example format for each of these functions, however, it should be appreciated that the examples provided in Table 3 are not exclusive.

A function call element begins at line 24. The "result" in line 26 defines an attribute or temporary variable to which the function's return value is assigned. The function may return at least one value, but it may modify multiple parameters that are provided to the function. The "params" in line 28 defines a list of parameters provided to the function. The parameters may be one or more attributes, temporary variables, or constants. Every parameter series can be preceded by a parameter "passing" type. The parameter passing types define different characteristics of each of the parameters. For example, a parameter with an "in" passing type may not be modified by the function, while a parameter with an "out" passing type may not be read by the function. Also, a parameter with an "in/out" passing type may be both read and modified by the function. The following is an example of parameter passing types: params="SRC_ADDRESS, out $TEMP_ADDRESS, inout BYTES, '5'". Notably, SRC_ADDRESS is an "in" type, because "in" is the default type. Also, every parameter may adhere to a certain format, for example [in|out|inout] and/or [Attribute|Temporary variable|Constant].

The assignment element beginning at line 30 is used to assign a value to an attribute or a temporary variable. The "result" command in line 32 represents an attribute or temporary variable to which the value is assigned. The "param" command in line 33 represents an attribute, temporary variable, or a constant value that is assigned to the result.

The operation element beginning at line 35 represents an arithmetic operation. The "result" command in line 38 is the result of the arithmetic operation, and may be an attribute or a temporary variable. The "left_param" in line 39 is a left parameter of an operation, and may be an attribute, temporary variable, or constant. The "operator" command in line 40 is an operation to be performed. The "right_param" in line 41 is a right parameter of the operation, and may be an attribute, temporary variable, or constant.

The lookup element beginning in line 43 provides a capability to map values of one or more derived attributes based on the known values of one or more given attributes. In particular, processing of the lookup element consists of extracting values for the configured input attributes of the lookup and mapping them to the configured output attributes of the lookup. In line 45, "name" is the name of the lookup. In line 46, "substs" provides a way to substitute a value of an attribute that is different from an attribute specified when the lookup was configured. For example, if a lookup exists that maps a "network address" to a "network," and the provided attributes are "source address" and "destination address," the "substs" attribute may be used to map the "source address" to the "network," or to map the "destination address" to the "network." The following is an example of a "subst" command: substs=PORT=SRC_PORT, ADDRESS = DST_ADDRESS". As the example illustrates, the attributes on the left side of the equal sign (*e.g.*, ADDRESS) are replaced by the attributes on the right of the equal sign (*e.g.*, DST_ADDRESS). It should be appreciated that substitutes for another attribute of the lookup should be configured with the same data type.

Generic Lookup

As discussed, CDL 209 includes rules that are used to provide configuration-driven conversions of the available flow objects 208 into other attributes that need to be passed to the consumers legacy billing system. Lookups are a part of CDL 209 that provide one technique of attribute conversion.

Generally, generic lookup 219 describes a technique for mapping a collection of values of an input set of attributes into a collection of values of an output set of attributes. Therefore, generic lookup 219 permits the mapping any number of attributes to any number of other attributes (derived or available). For example, generic lookup 219 may be used to map a collection of URLs, like "www.cnn.com\sports\baseball" and "www.cnn.com\politics," into a particular content-type attribute, like "news."

Generic lookup 219 is useful for deriving new attributes that may be needed by the customer to perform certain operations, like content billing, for example. Using the example above, a particular customer may desire to bill their customers differently, depending on the type of content. For example, news may be billed at one rate, while music may be billed at another rate. Generic lookup 219 permits the relevant content type, which is not directly available from the attributes in flow objects 208, to be determined from the URL, which is available from the attributes in flow objects 208.

Generic lookup 219 is configuration-driven in that the rules of CDL 209 are capable of performing dynamic invocation of a desired lookup. Generic lookups may be provided for any input and output attributes that are defined in system 100 (as discussed with reference to Table 1).

Modifications to Lookups may occur in real-time. Also, lookup values apply to the current time. Therefore, customers lookup values that are entered into system 100 should be applicable to the data that is submitted into system 100 by data sources 201-203. For example, if lookups were to be used to map URL to accounts, and URL data fed into system 100 is a year old, the account-to-URL information entered into system 100 must reflect the mapping a year ago, rather than at the current time.

An input attribute may be matched to multiple output attributes. For example, one URL (*e.g.*, www.cnn.com) may map to multiple content types (*e.g.*, news and video stream). Also, lookup entries may be updated by replacing existing lookup entries.

Generic lookup 219 may match attributes by any of the characteristics of the attribute including an exact value, a numeric range, and/or a matching pattern, for example. A match pattern is a string that represents a range of attribute values using, for example, a special character. For example, where the special character is "*" that matches any number of any characters (i.e, a wildcard) and escaped value of period '\.' that matches any single character. For example, "*.yah\.o.*" matches "www.yahoo.com" and www.yahio.com." Therefore, where matching patterns are used, an input attribute can match multiple patterns, and so multiple values for the output attribute may result.

Generic lookup 219 also will accommodate the mapping of multiple input attributes to multiple output attributes. Input attributes can be matched by exact value, numeric range, or matching pattern.

Table 4 provides one example of a generic lookup file. It should be appreciated, however, that Table 4 is just an example of such a file and is not meant to be an exclusive representation.

```
01    <GROUP>
02    GROUP_NAME DOMAIN_TO_CONTENT
03    MANDATORY_MATCH     no

04    <IN_ATTRIBUTE>
05    ATTRIBUTE_NAME DOMAIN
06    MATCH_PATTERN yes

07    <CACHE>
08    MAX_CACHE_ENTRIES 100
09    MIN_EXPECTED_FREQUENCY 2
10    </CACHE>

11    </IN_ATTRIBUTE>

12    <OUT_ATTRIBUTE>
13    ATTRIBUTE_NAME CONTENT_TYPE
14    DEFAULT_VALUE  "unrecognized content"
15    </OUT_ATTRIBUTE>
16    </GROUP>
```

Table 4

As shown in Table 4, multiple lookups are referred to as "groups." Line 01 indicates that the series of commands that follow represent a lookup or "group." As indicated in line 02, the lookup's name "DOMAIN_TO_CONTENT" indicates its functionality of mapping a domain name to a content type. The lookup's configuration may identify whether a match is always expected (*i.e.,* "mandatory"). As indicated in line 03, when the "MANDATORY_MATCH" value is set to "no," which may be the default setting, then a resulting match need not be returned by the lookup functionality.

Every group will have one or more input attributes that are "looked up" and one or more output attributes that are the "result" of the lookup. The input attribute may be identified by the type of attribute. For example, the input attribute may be a match pattern or a numeric range. If the input attribute is identified as a match pattern, for example, it will match an entry on lookup if its value matches the pattern specified in the lookup (as discussed above). If the input attribute is identified as a numeric range, for example, it will match an entry on lookup if its numeric value falls in the range specified in the lookup.

As indicated in lines 04 through 07, the input attribute name is "DOMAIN" and it has a mandatory match pattern requirement. Also, as indicated in line 12 through 15, the output attribute name is "CONTENT_TYPE," whose default value is "unrecognized content," which used to specify the value for the output attribute when the lookup does not result in any matches. If a match is mandatory, and there are no default values provided for a non-match, the lookup will report that no match was found, and thus will not assign any values to the output attributes. The lookup also will notify the corresponding data adapter of this condition, so that it may make certain adjustments.

The following example is provided for discussion:

```
<DOMAIN_TO_CONTENT>
*yahoo*  yahoo_content
*ford*  ford_content
</DOMAIN_TO_ACCT>
```

"DOMAIN_TO_CONTENT is the name of the grouping, where Domain is the input attribute and Content is the output attribute. "*yahoo*" is a match pattern that is applied to the input attribute Domain, while "yahoo_content" is the output attribute, Content. "*ford*" is a match pattern applied on the input attribute Domain, while "ford_content" is an output attribute Content. "</DOMAIN_TO_ACCT>" signals the end of the mappings for the grouping.

Caching, or storing the results of the lookup in cache memory, may be a part of the lookup functionality. If input attribute is identified as a match pattern, caching can be used for the actual value of the input attribute and the match patterns that result. Line 08 in Table 4 identifies a maximum number of entries to store in the cache. Also, a minimum expected frequency may identify certain entries that should not be removed from cache when new entries are added, because of their frequency. For example, a certain entry may occur so often that it must remain within cache by default, regardless of the number of overwriting entries.

There are a number of utility functions that may be provided with the lookup configuration. For example, a utility may be provided to convert the text file containing the attribute lookup entry into a run-time attribute lookup database file that is required by system 100. Also, there may be a utility that converts run-time attribute lookup database into a text file. In other words, the utility will export the database to a text file. In addition, there may be provided a utility that adds, modifies, and removes lookup entries at runtime. A utility may be provided that retrieves attribute mappings.

The invention is directed to a system and method for aggregating data. The invention often was described in the context of the Internet, but is not so limited to the Internet, regardless of any specific description in the drawing or examples set forth herein. For example, the invention may be applied to wireless networks, as well as non-traditional networks like Voice-over-IP-based networks and/or private networks. It will be understood that the invention is not limited to the use of any of the particular components or devices herein. Indeed, this invention can be used in any application that requires aggregating data. Further, the system disclosed in the invention can be used with the method of the invention or a variety of other applications.

While the invention has been particularly shown and described with reference to the embodiments thereof, it will be understood by those skilled in the art that the invention is not limited to the embodiments specifically disclosed herein. Those skilled in the art will appreciate that various changes and adaptations of the invention may be made in the form and details of these embodiments without departing from the true spirit and scope of the invention as defined by the following claims.

What is Claimed is:

1. A method of transforming data, comprising:

   creating a set of instructions,

   receiving data;

   deriving data from said received data using said instructions, wherein said instructions are independent of said data; and

   modifying said instructions as said data is received.

2. The method of claim 1, wherein said deriving comprises mapping said received data to said derived data.

3. The method of claim 2, wherein said mapping comprises a generic lookup functionality.

4. The method of claim 3, wherein said generic lookup functionality compares said received data to a constant value.

5. The method of claim 4, wherein said comparing includes locating a match for the received data in a set of constant values.

6. The method of claim 3, wherein said generic lookup functionality compares said received data to a range of values.

7. The method of claim 6, wherein said comparing includes locating a match for the received data in a set of ranges of values.

8. The method of claim 3, wherein said generic lookup functionality compares said received data to a matching pattern.

9. The method of claim 8, wherein said comparing includes locating a match for the received data in a set of matching patterns.

10. The method of claim 8, wherein said matching pattern includes wildcard designators.

11. The method of claim 1, wherein said data comprises at least one of the following: an attribute, an attribute value, a variable value, and a constant value.

12. The method of claim 1, wherein said modifying comprises providing additional instructions.

13. The method of claim 1, wherein said instructions are computer-executable instructions written using an extensible language technique.

14. The method of claim 13, wherein said extensible language technique comprises an extensible markup language (XML).

15. The method of claim 1, wherein said deriving comprises initiating a processing function on said data.

16. The method of claim 1, wherein said deriving comprises performing an arithmetic operation on said data.

17. The method of claim 1, wherein said deriving comprises conditionally processing said data.

18. The method of claim 1, wherein said deriving comprises assigning a value to said data.

19. A computer-readable medium having computer-executable instructions for transforming data, comprising:

invoking computer-executable instructions, receiving data;

deriving data from said received data using said computer-executable instructions, wherein said computer-executable instructions are independent of said data; and

modifying said computer-executable instructions in real-time.

20. The computer-readable medium of claim 19, wherein said deriving comprises mapping said received data to said derived data.

21. The computer-readable medium of claim 20, wherein said mapping comprises a generic lookup functionality.

22. The computer-readable medium of claim 21, wherein said generic lookup functionality compares said received data to a constant value.

23. The computer-readable medium of claim 22, wherein said comparing includes locating a match for the received data in a set of constant values.

24. The computer-readable medium of claim 21, wherein said generic lookup functionality compares said received data to a range of values.

25. The computer-readable medium of claim 24, wherein said comparing includes locating a match for the received data in a set of ranges of values.

26. The computer-readable medium of claim 21, wherein said generic lookup functionality compares said received data to a matching pattern.

27. The computer-readable medium of claim 26, wherein said comparing includes locating a match for the received data in a set of matching patterns.

28. The computer-readable medium of claim 26, wherein said matching pattern includes wildcard designators.

29. The computer-readable medium of claim 19, wherein said data comprises at least one of the following: an attribute, an attribute value, a variable value, and a constant value.

30. The computer-readable medium of claim 19, wherein said modifying comprises providing additional computer-executable instructions.

31. The computer-readable medium of claim 19, wherein said computer-executable instructions use an extensible language technique.

32. The computer-readable medium of claim 31, wherein said extensible language technique comprises an extensible markup language (XML).

33. The computer-readable medium of claim 19, wherein said deriving comprises initiating a processing function on said data.

34. The computer-readable medium of claim 19, wherein said deriving comprises performing an arithmetic operation on said data.

35. The computer-readable medium of claim 19, wherein said deriving comprises conditionally processing said data.

36. The computer-readable medium of claim 19, wherein said deriving comprises assigning a value to said data.

37. A system for transforming data received from a communication network, comprising:

a content collection layer that receives data from said communication network;

a transaction layer in communication with said content collection layer, wherein said transaction layer captures predetermined agreements regarding the value of said data among users of the system; and a settlement layer in communication with said transaction layer, wherein said settlement layer rates a significance of said data.

38. The system of claim 37, wherein the content collection layer comprises a content data language component that transforms said data.

39. The system of claim 38, wherein the content collection layer comprises an input data adapter in communication with said content data language component, and wherein said input data adapter converts data from one or more data sources to sets of relevant attributes.

40. The system of claim 39, wherein said content data language component transforms said data before said data is converted by said input data adapter.

41. The system of claim 38, wherein the content collection layer comprises an aggregator component in communication with said content data language component.

42. The system of claim 41, wherein said content data language component manipulates said data before said aggregator component manipulates said data.

43. The system of claim 41, wherein said content data language component manipulates said data based on a request from said aggregator component.

44. The system of claim 38, wherein the content collection layer comprises a correlator component in communication with said content data language component, and wherein said correlator component groups said data.

45. A method for integrating multiple independent data processing components in a system, comprising:

receiving data having predetermined characteristics;

converting said predetermined characteristics to generic characteristics; and

modifying said generic characteristics as said data is received.

46. The method of claim 45, wherein said converting is conducted by a set of computer-executable instructions.

47. The method of claim 45, wherein said modifying includes exchanging said generic characteristics for other characteristics.

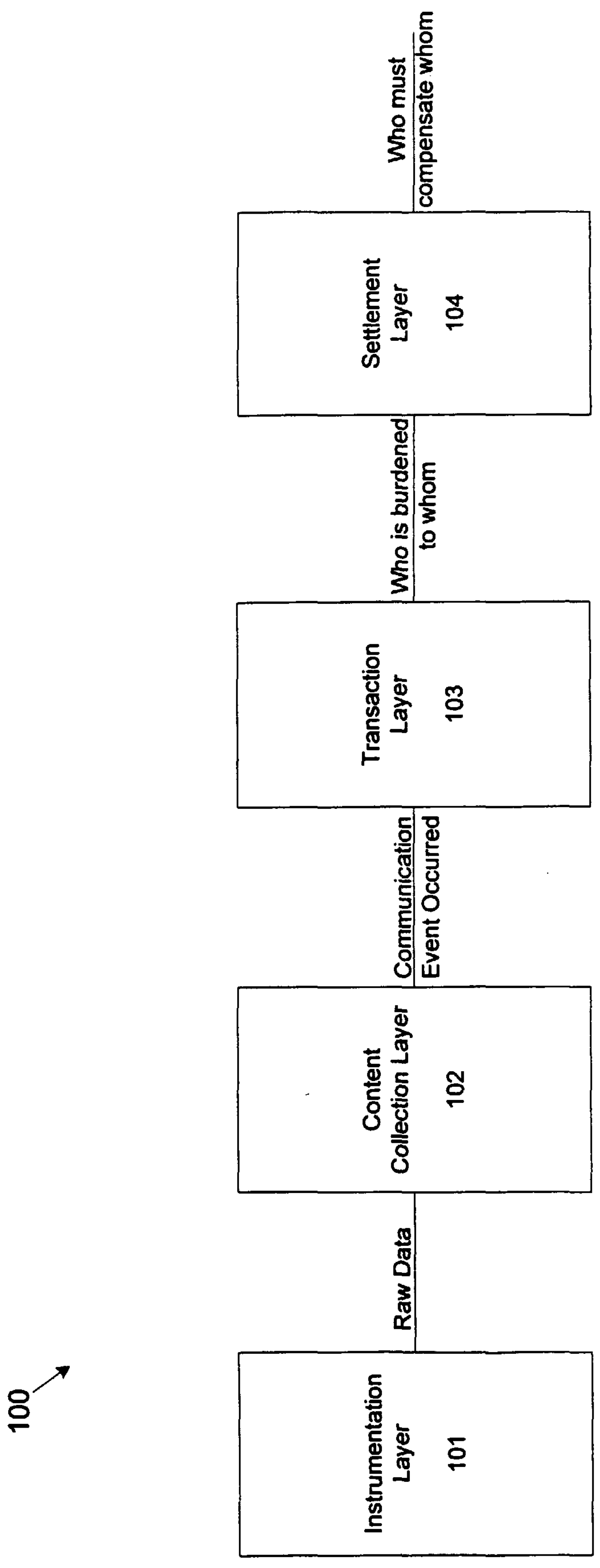

100
Instrumentation Layer 101
Raw Data
Content Collection Layer 102
Communication Event Occurred
Transaction Layer 103
Who is burdened to whom
Settlement Layer 104
Who must compensate whom

Figure 1

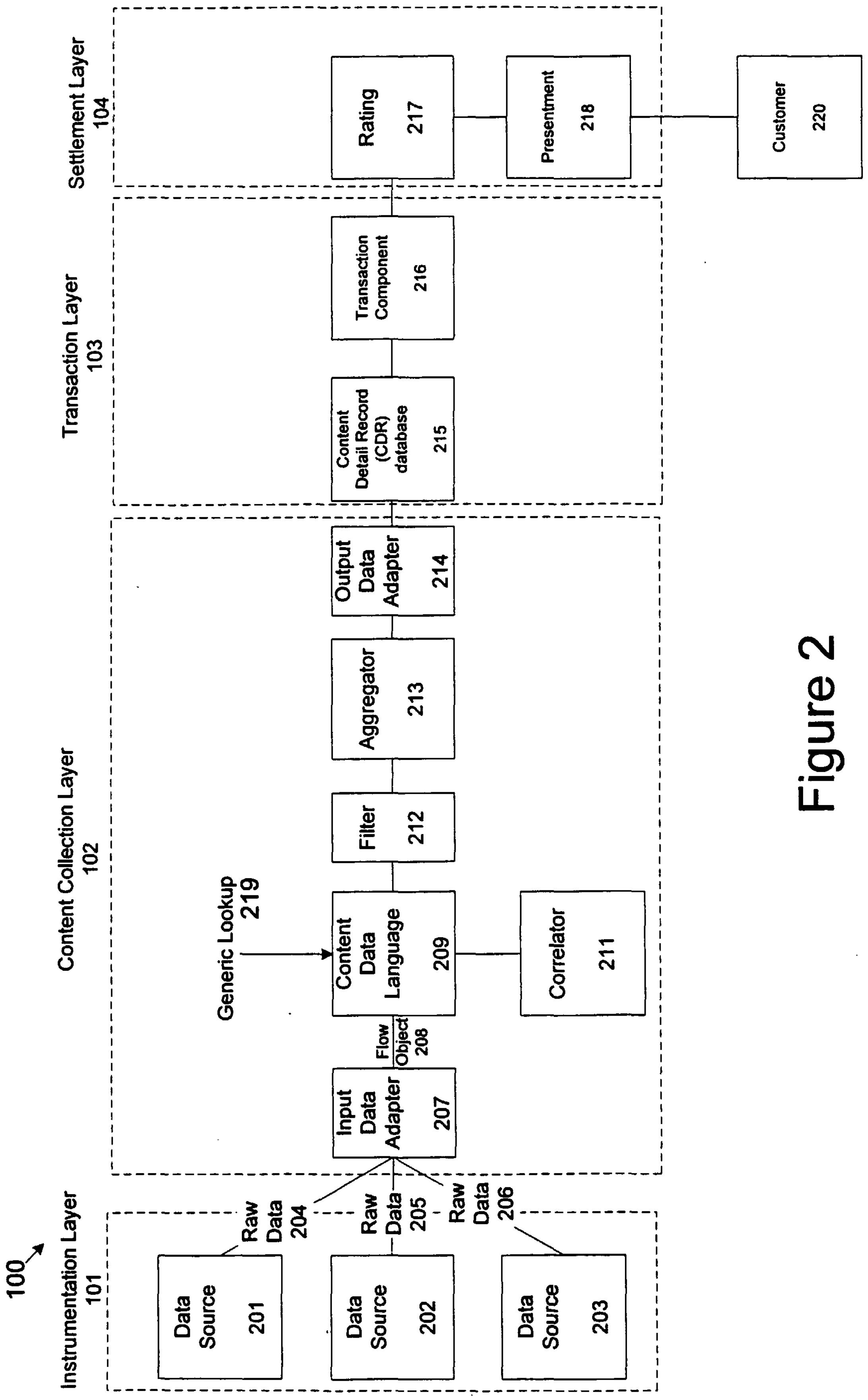

100
Instrumentation Layer 101
Data Source 201
Data Source 202
Data Source 203
Raw Data 204
Raw Data 205
Raw Data 206
Content Collection Layer 102
Input Data Adapter 207
Flow Object 208
Content Data Language 209
Generic Lookup 219
Correlator 211
Filter 212
Aggregator 213
Output Data Adapter 214
Transaction Layer 103
Content Detail Record (CDR) database 215
Transaction Component 216
Settlement Layer 104
Rating 217
Presentment 218
Customer 220

Figure 2

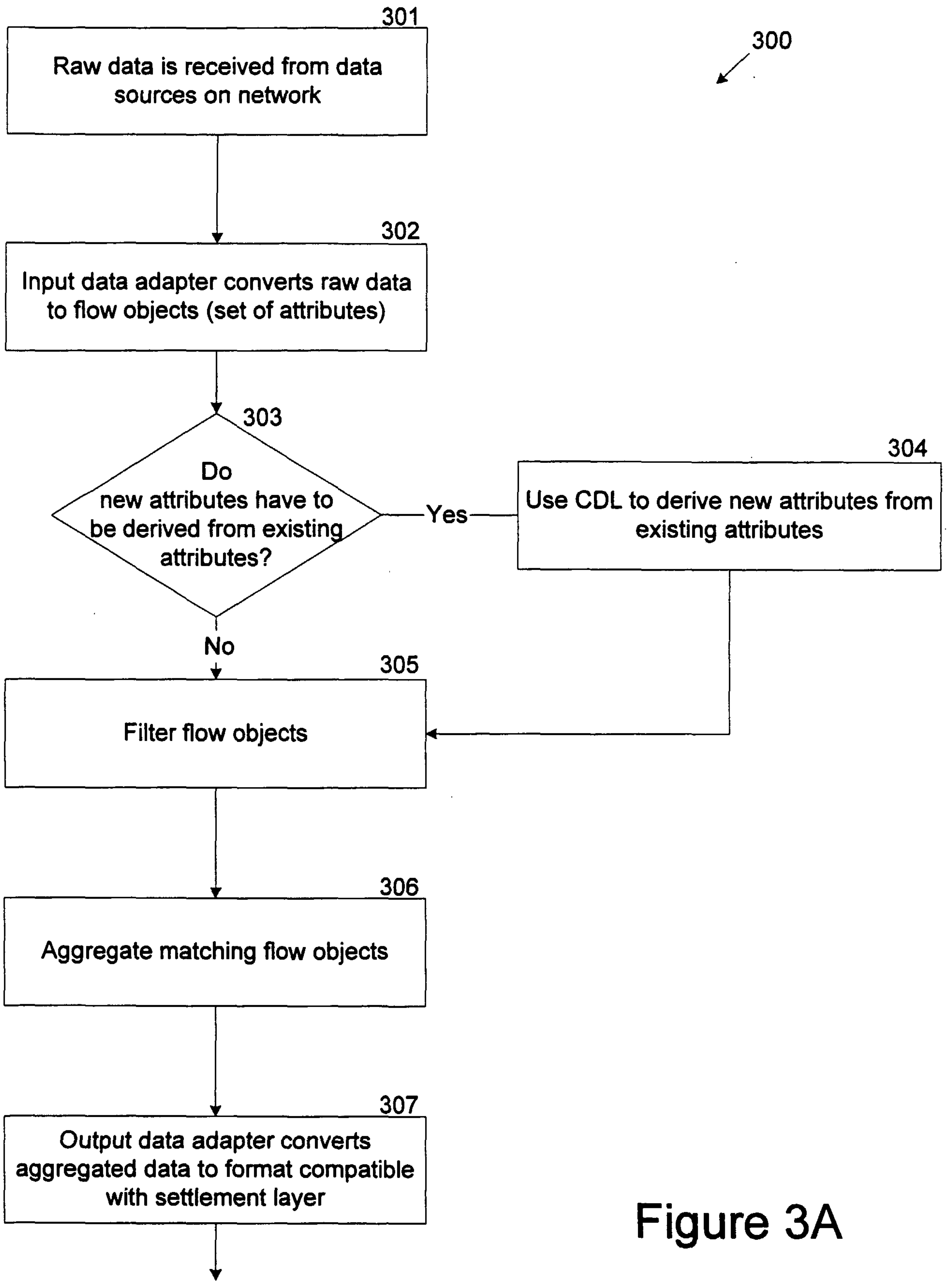

300
301
Raw data is received from data sources on network
302
Input data adapter converts raw data to flow objects (set of attributes)
303
Do new attributes have to be derived from existing attributes?
Yes
304
Use CDL to derive new attributes from existing attributes
No
305
Filter flow objects
306
Aggregate matching flow objects
307
Output data adapter converts aggregated data to format compatible with settlement layer

Figure 3A

Continued on Figure 3B

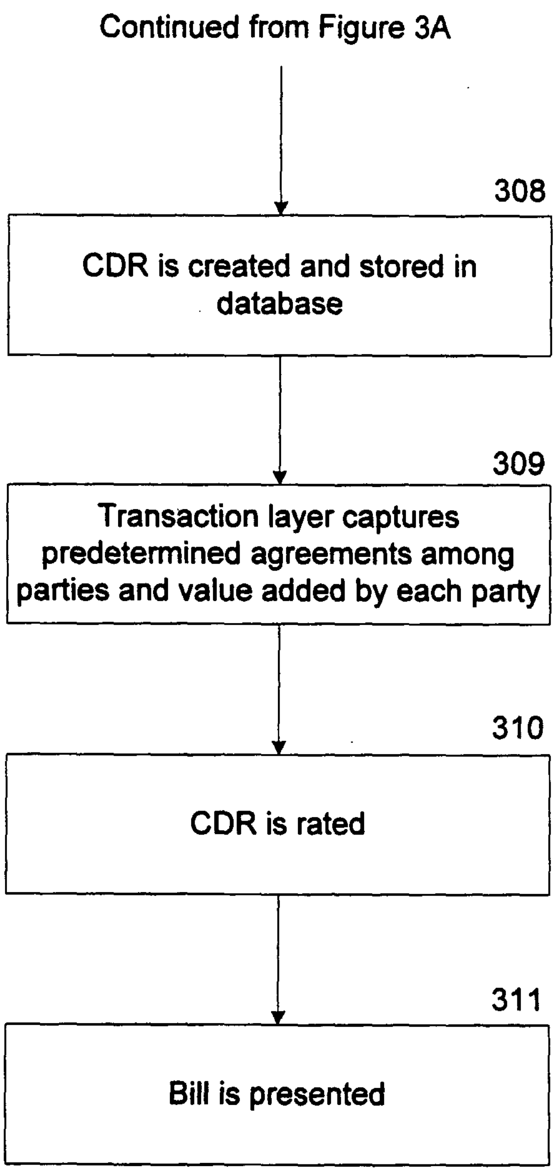

Continued from Figure 3A
308
CDR is created and stored in database
309
Transaction layer captures predetermined agreements among parties and value added by each party
310
CDR is rated
311
Bill is presented

Figure 3B

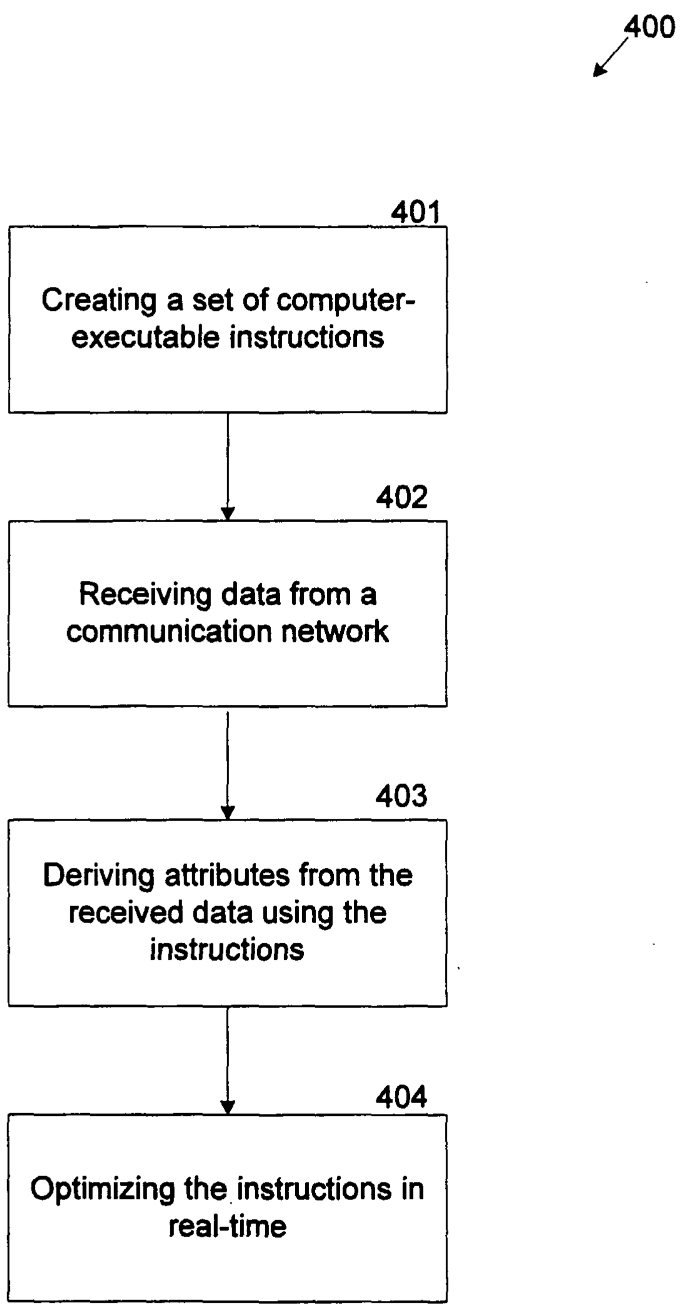

400
401
Creating a set of computer-executable instructions
402
Receiving data from a communication network
403
Deriving attributes from the received data using the instructions
404
Optimizing the instructions in real-time

Figure 4

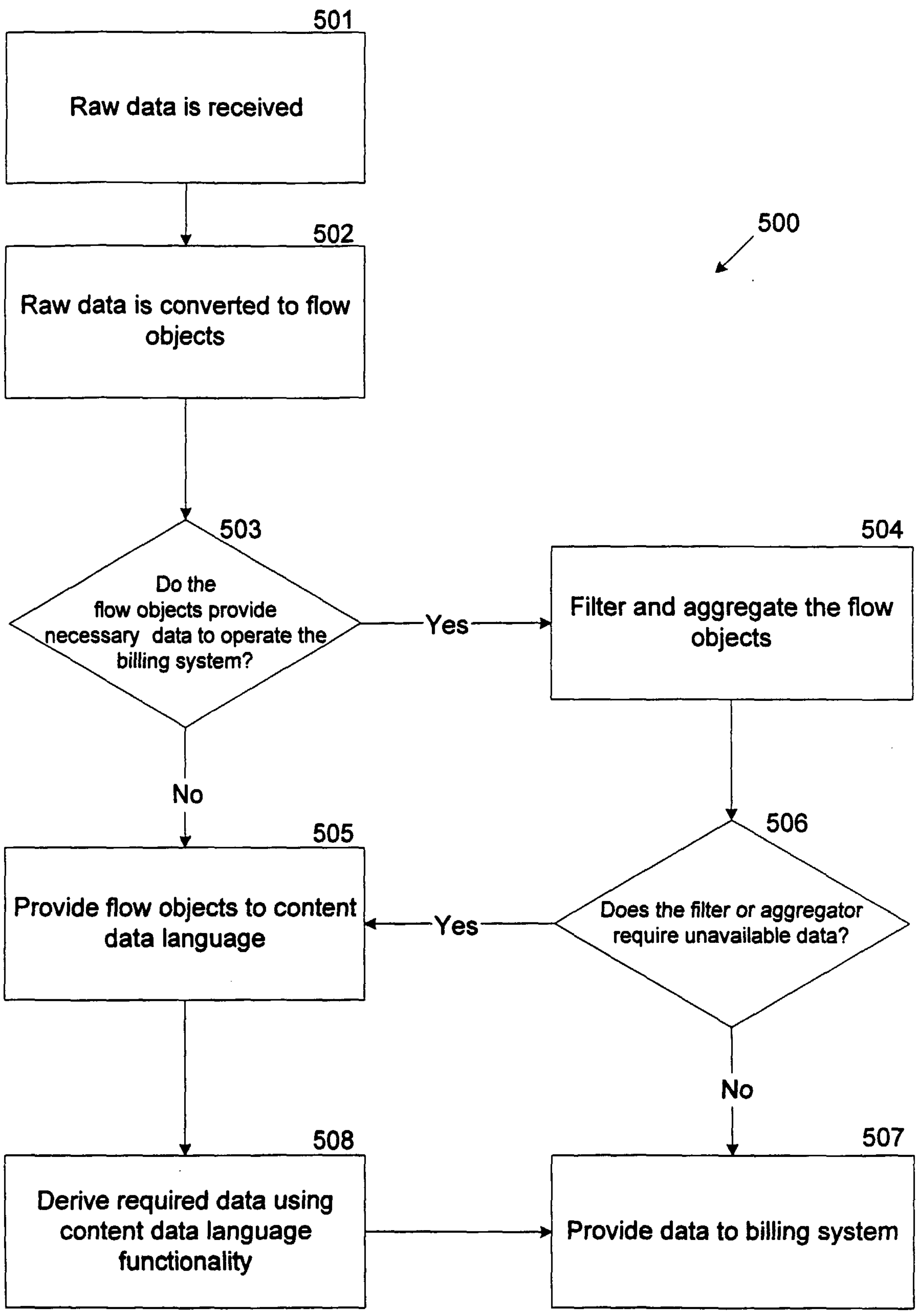

501
Raw data is received
502
Raw data is converted to flow objects
500
503
Do the flow objects provide necessary data to operate the billing system?
Yes
504
Filter and aggregate the flow objects
No
505
Provide flow objects to content data language
Yes
506
Does the filter or aggregator require unavailable data?
No
508
Derive required data using content data language functionality
507
Provide data to billing system

Figure 5