US 20080114790A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2008/0114790 A1

(43) Pub. Date: May 15, 2008

Hall et al.

(54) TECHNIQUES FOR DEFINING, USING AND MANIPULATING RIGHTS MANAGEMENT DATA STRUCTURES

(75) Inventors: **Edwin J. Hall**, San Jose, CA (US); **Victor H. Shear**, Alamo, CA (US); **Luke S. Tomasello**, San Jose, CA (US); **David M. Van Wie**, Eugene, OR (US); **Robert P. Weber**, Menlo Park, CA (US); **Kim Worsencroft**, Los Gatos, CA (US); **Xuejun Xu**, Fremont, CA (US)

Correspondence Address:
**FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER LLP**
**901 NEW YORK AVENUE, NW**
**WASHINGTON, DC 20001-4413**

(73) Assignee: **Intertrust Technolgies Corp.**

(21) Appl. No.: **11/926,972**

(22) Filed: **Oct. 29, 2007**

**Related U.S. Application Data**

(63) Continuation of application No. 11/256,518, filed on Oct. 20, 2005, Continuation of application No. 09/819,063, filed on Sep. 28, 2000, now Pat. No. 7,062,500, which is a continuation of application No. 09/300,778, filed on Apr. 27, 1999, now Pat. No. 6,138,119, which is a continuation of application No. 08/805,804, filed on Feb. 25, 1997, now Pat. No. 5,920,861.

**Publication Classification**

(51) **Int. Cl.**
***G06F 17/00*** (2006.01)

(52) **U.S. Cl.** ..... **707/100**

(57) **ABSTRACT**

A descriptive data structure provides an abstract representation of a rights management data structure such as a secure container. The abstract representation may describe, for example, the layout of the rights management data structure. It can also provide metadata describing or defining other characteristics of rights management data structure use and/or processing. For example, the descriptive data structure can provide integrity constraints that provide a way to state rules about associated information. The abstract representation can be used to create rights management data structures that are interoperable and compatible with one another. This arrangement preserves flexibility and ease of use without compromising security.

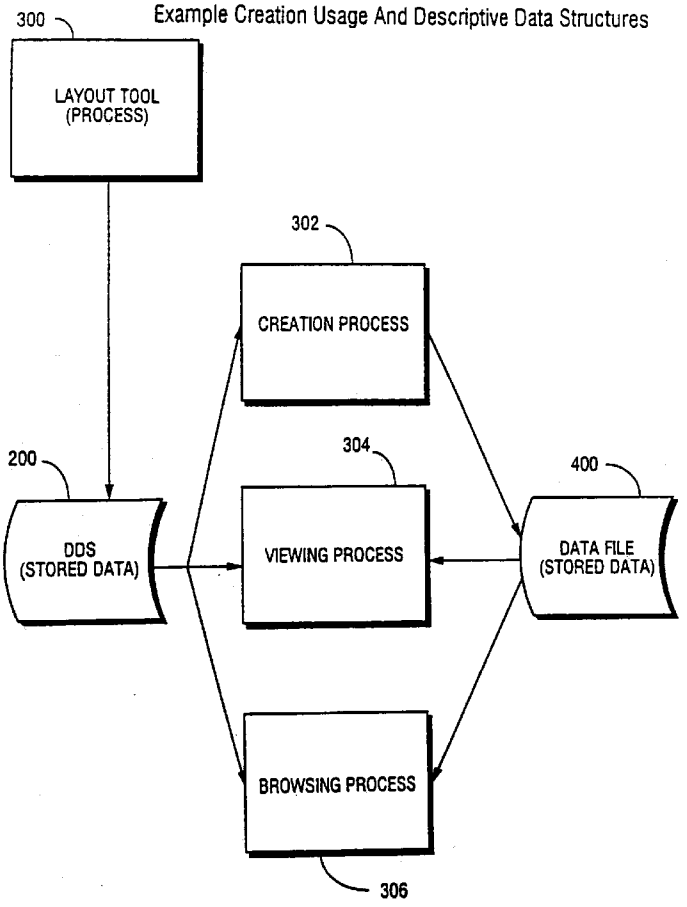
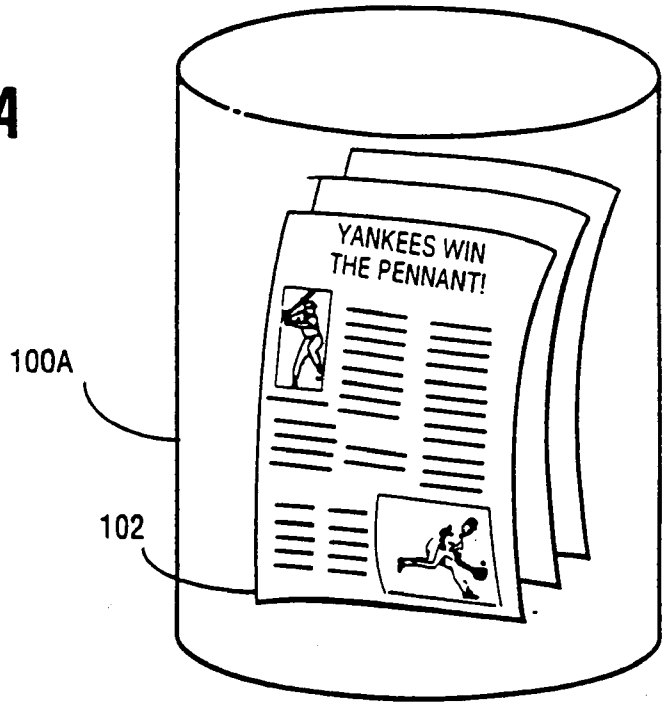


Fig. 1A



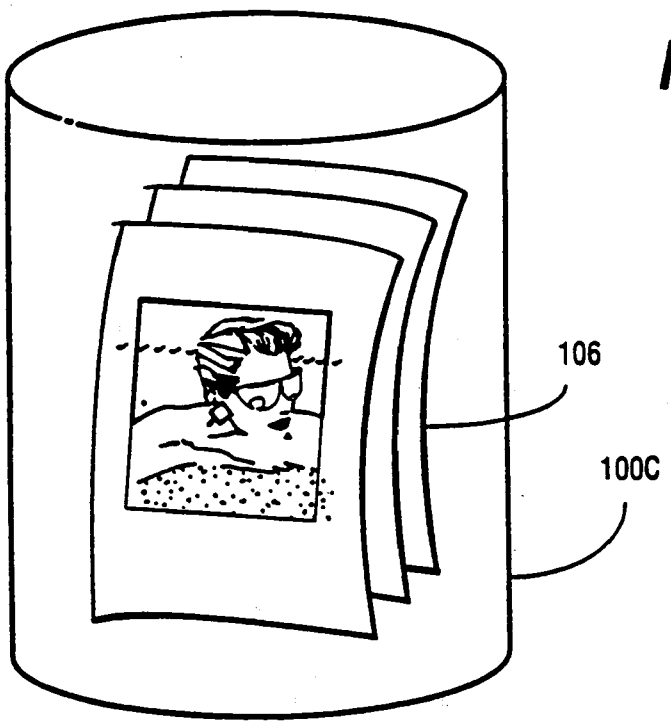
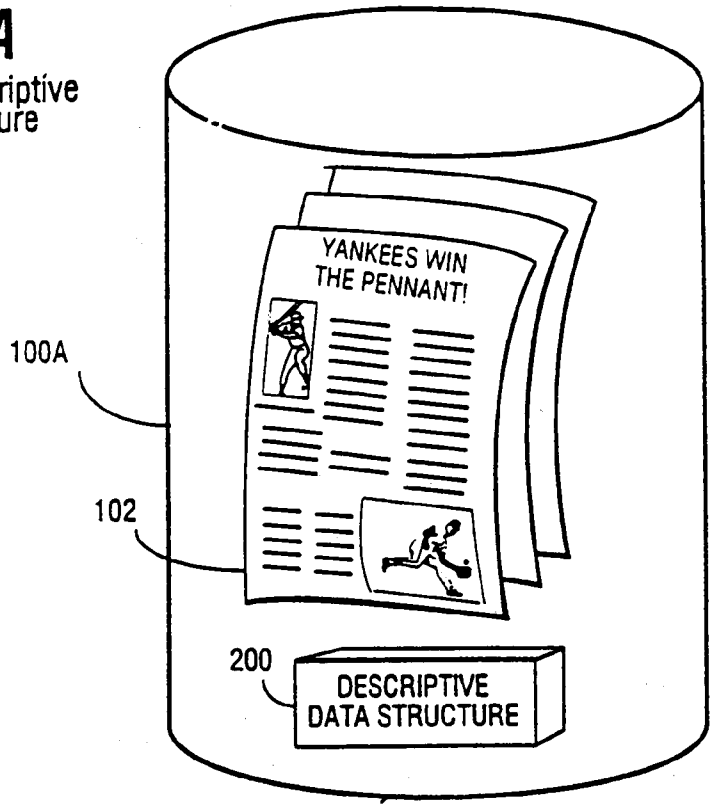




Fig. 1B

**Fig. 2A**
Example Descriptive Data Structure



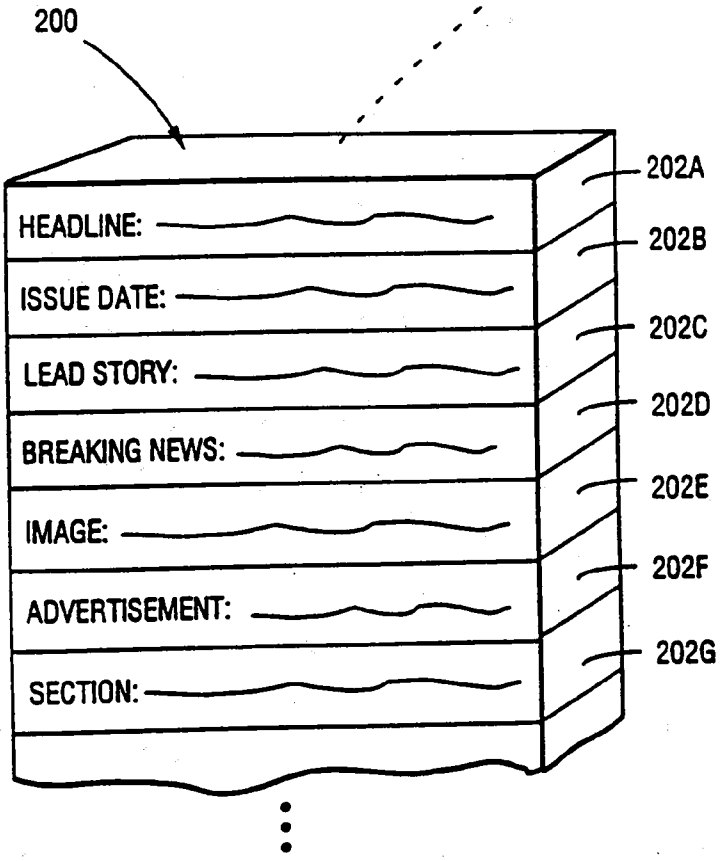



**Fig. 2B** Example Descriptive Data Structure

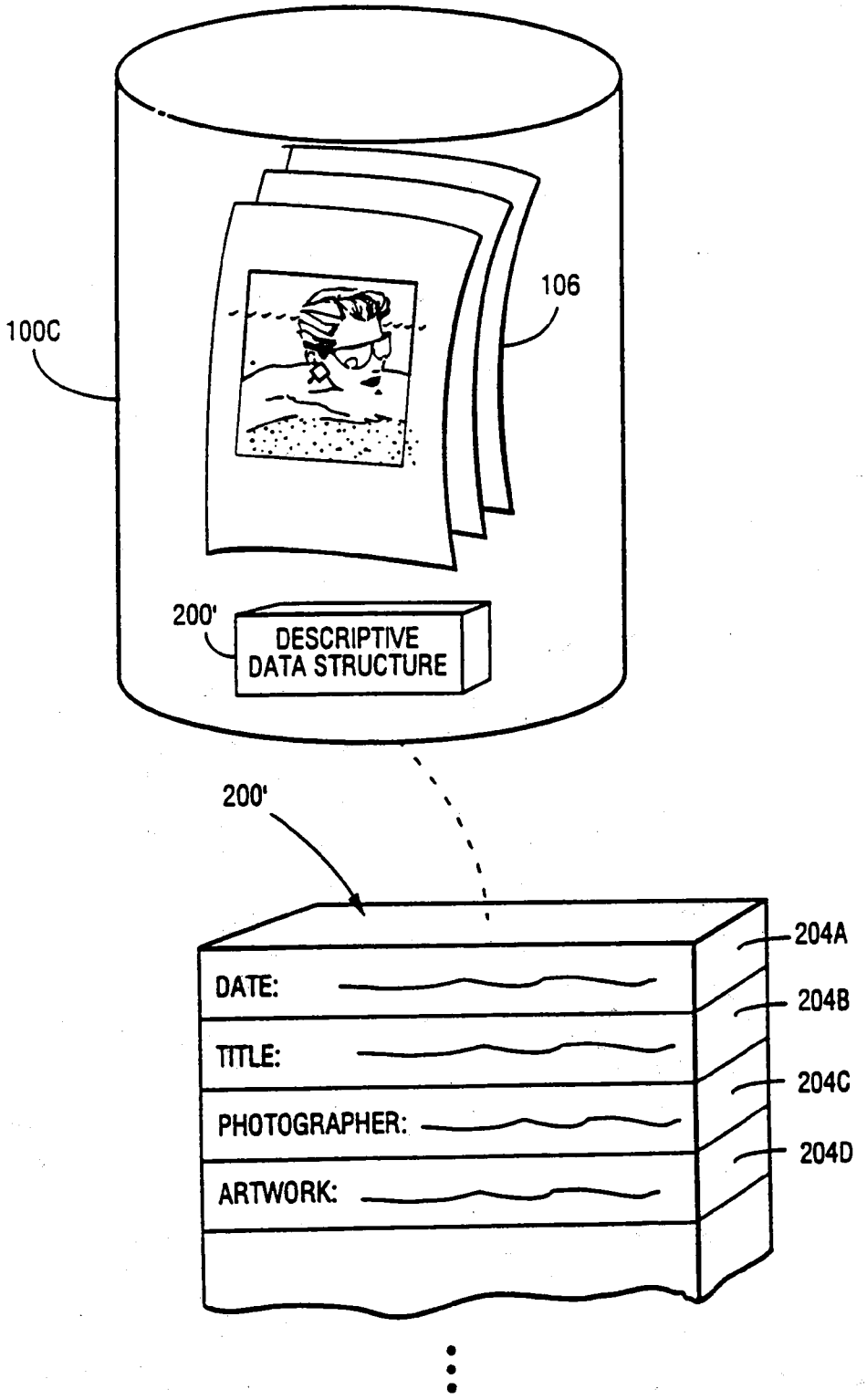
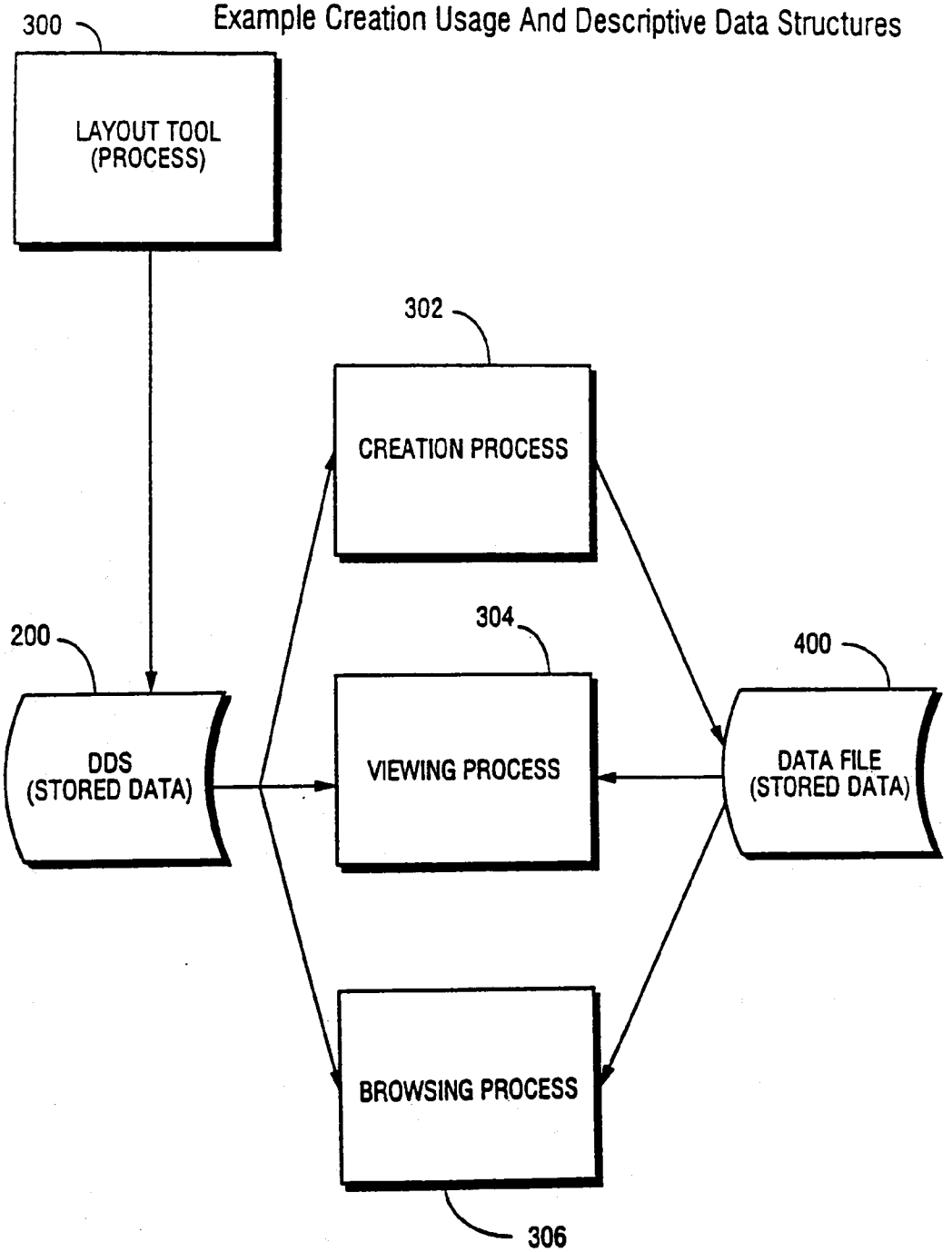


*Fig. 3*

Example Creation Usage And Descriptive Data Structures



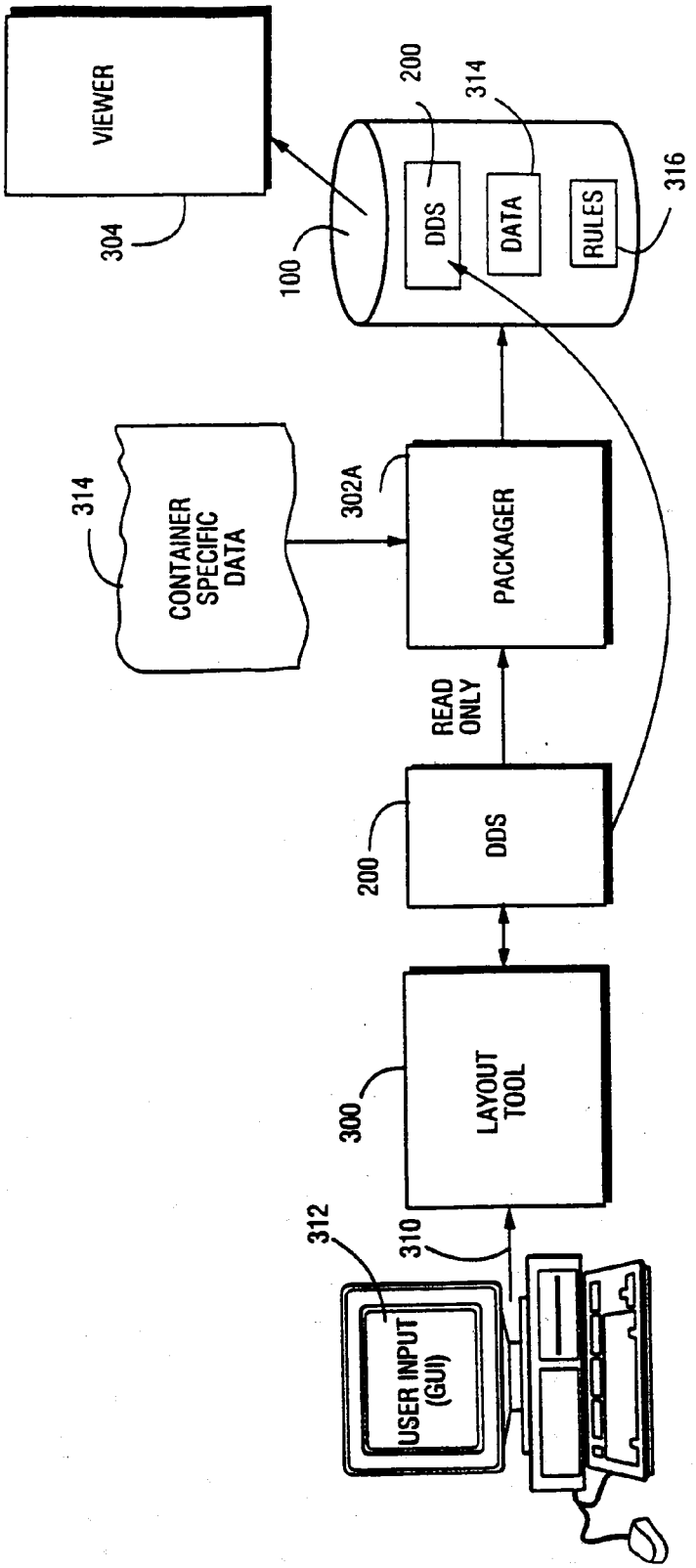
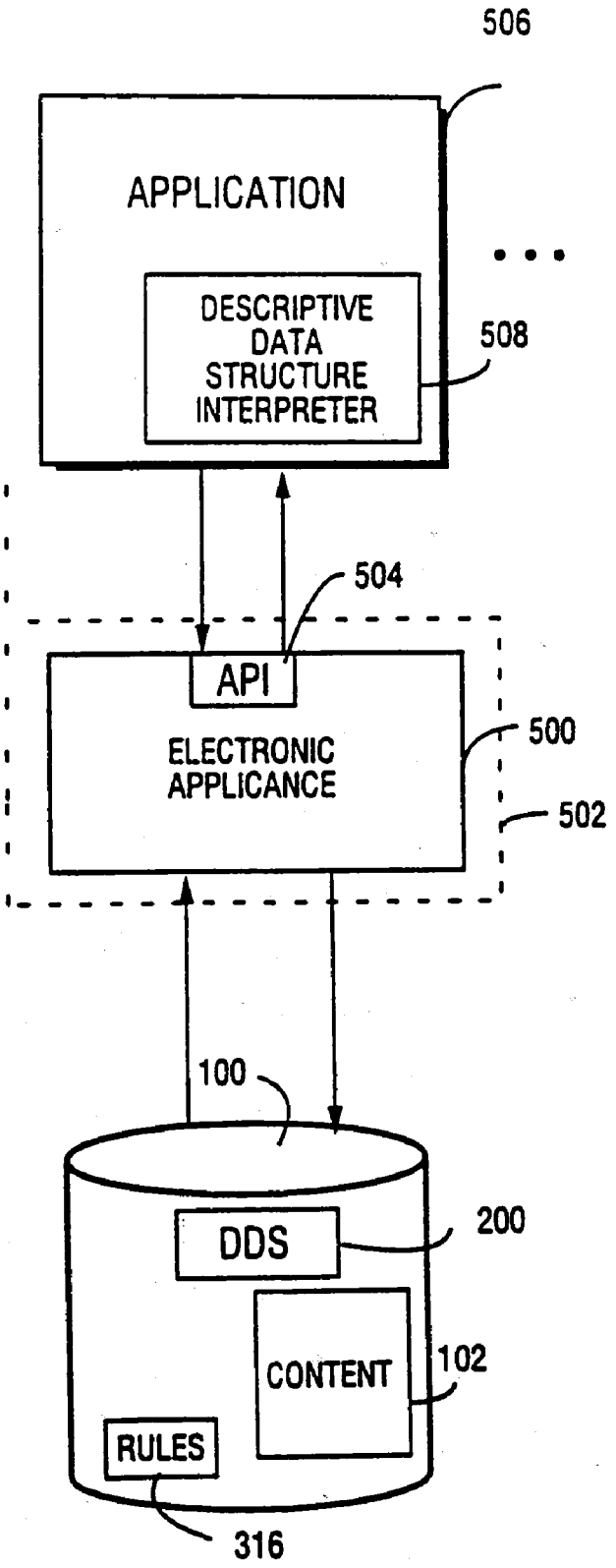



*Fig. 4* Example Template Creation And Usage

***Fig. 5*** Example Secure System Architecture



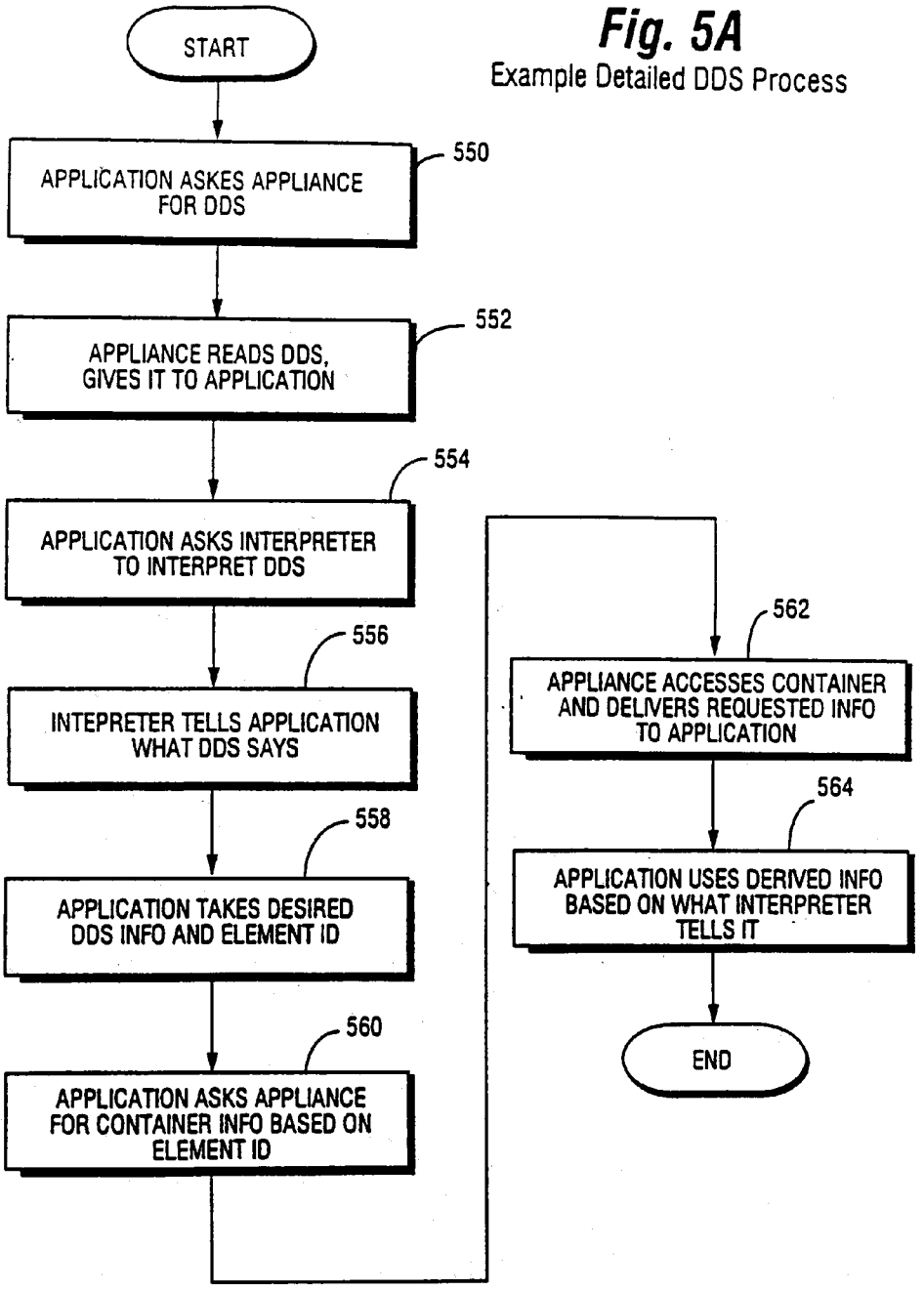



***Fig. 5A***

Example Detailed DDS Process

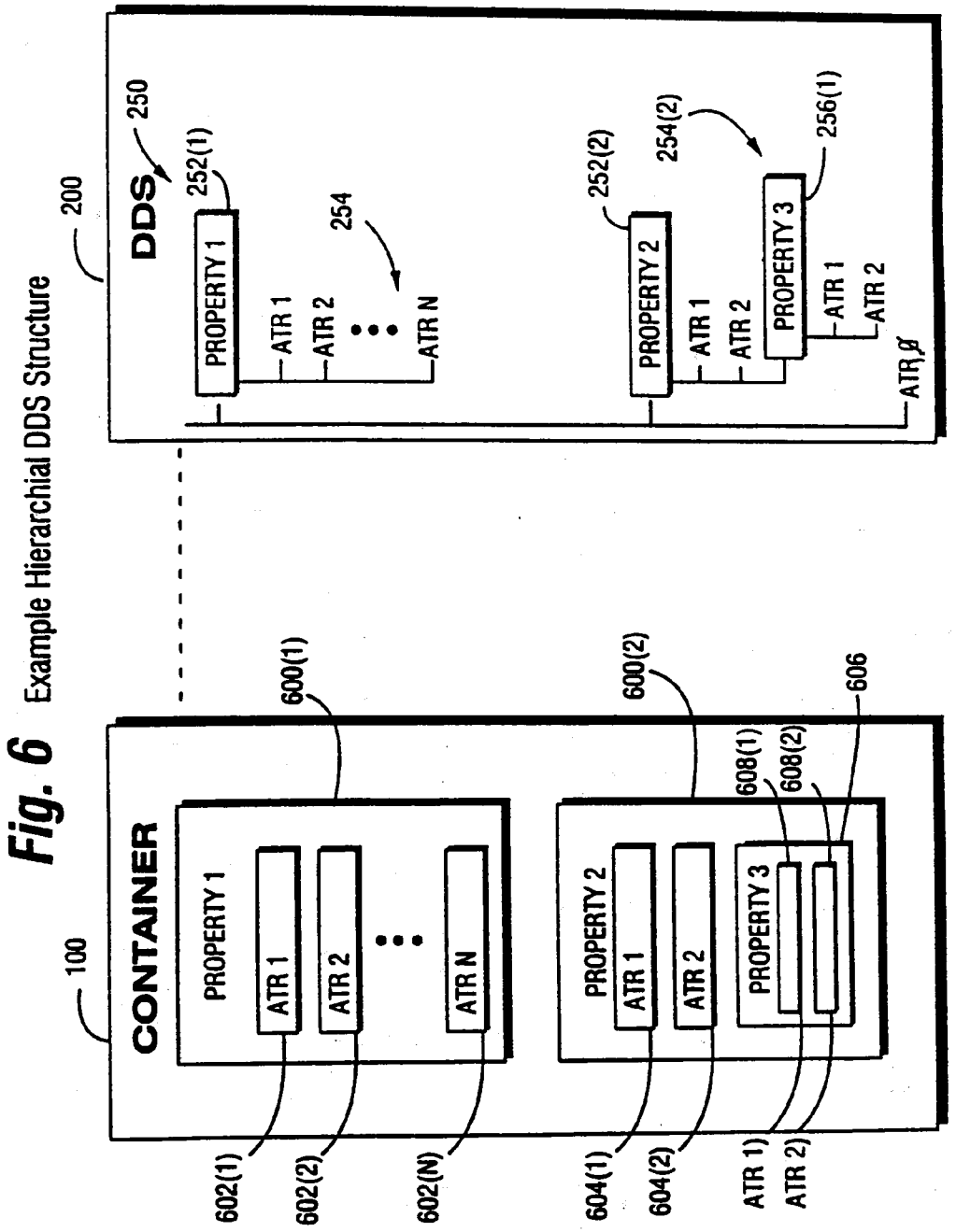

Example Hierarchial DDS Structure
100
CONTAINER
PROPERTY 1
ATR 1
ATR 2
ATR N
600(1)
602(1)
602(2)
602(N)
PROPERTY 2
ATR 1
ATR 2
PROPERTY 3
600(2)
606
608(1)
608(2)
604(1)
604(2)
ATR 1)
ATR 2)
200
DDS
250
252(1)
PROPERTY 1
ATR 1
ATR 2
ATR N
254
252(2)
PROPERTY 2
ATR 1
ATR 2
254(2)
PROPERTY 3
256(1)
ATR 1
ATR 2
ATR g

Fig. 6

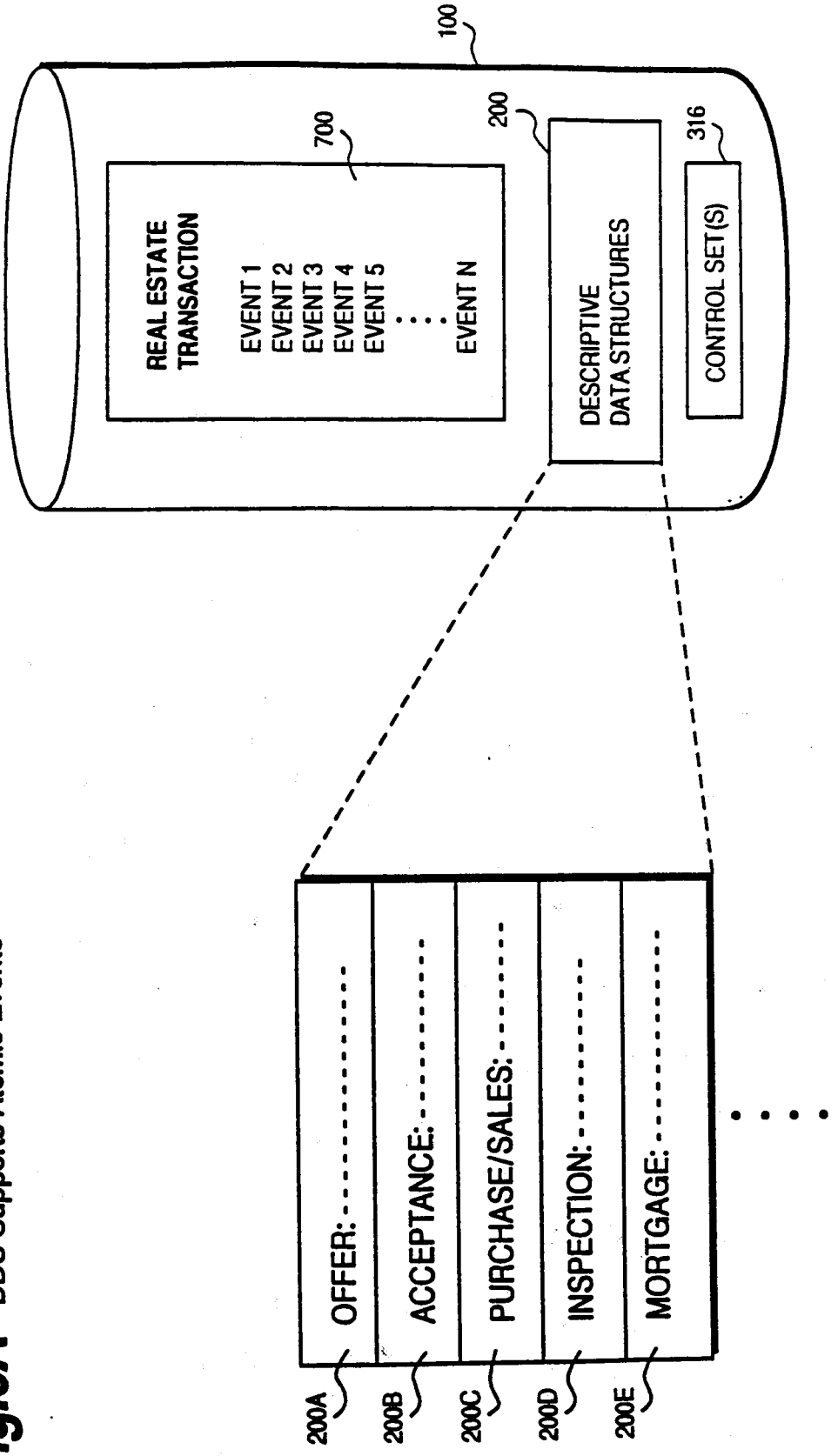
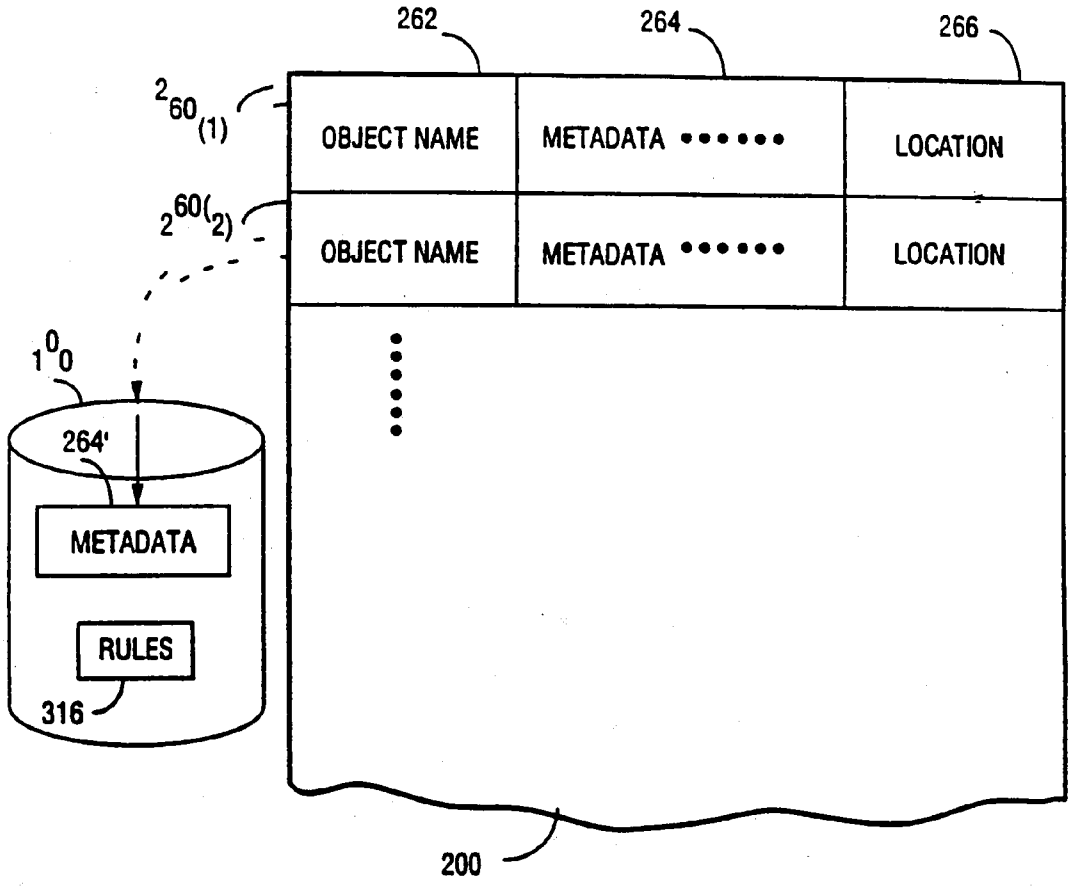



Fig.6A DDS Supports Atomic Events

***Fig. 7*** Example Descriptive Data Structure Format



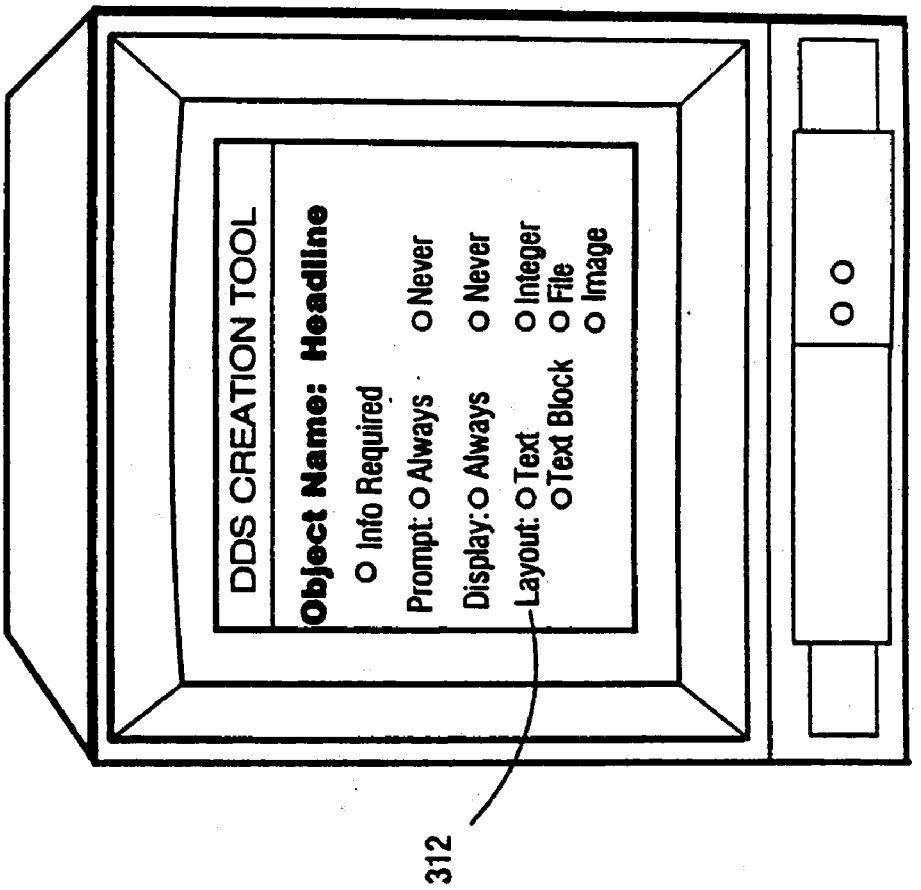



*Fig. 8* Example DDS Creation Graphics Interface

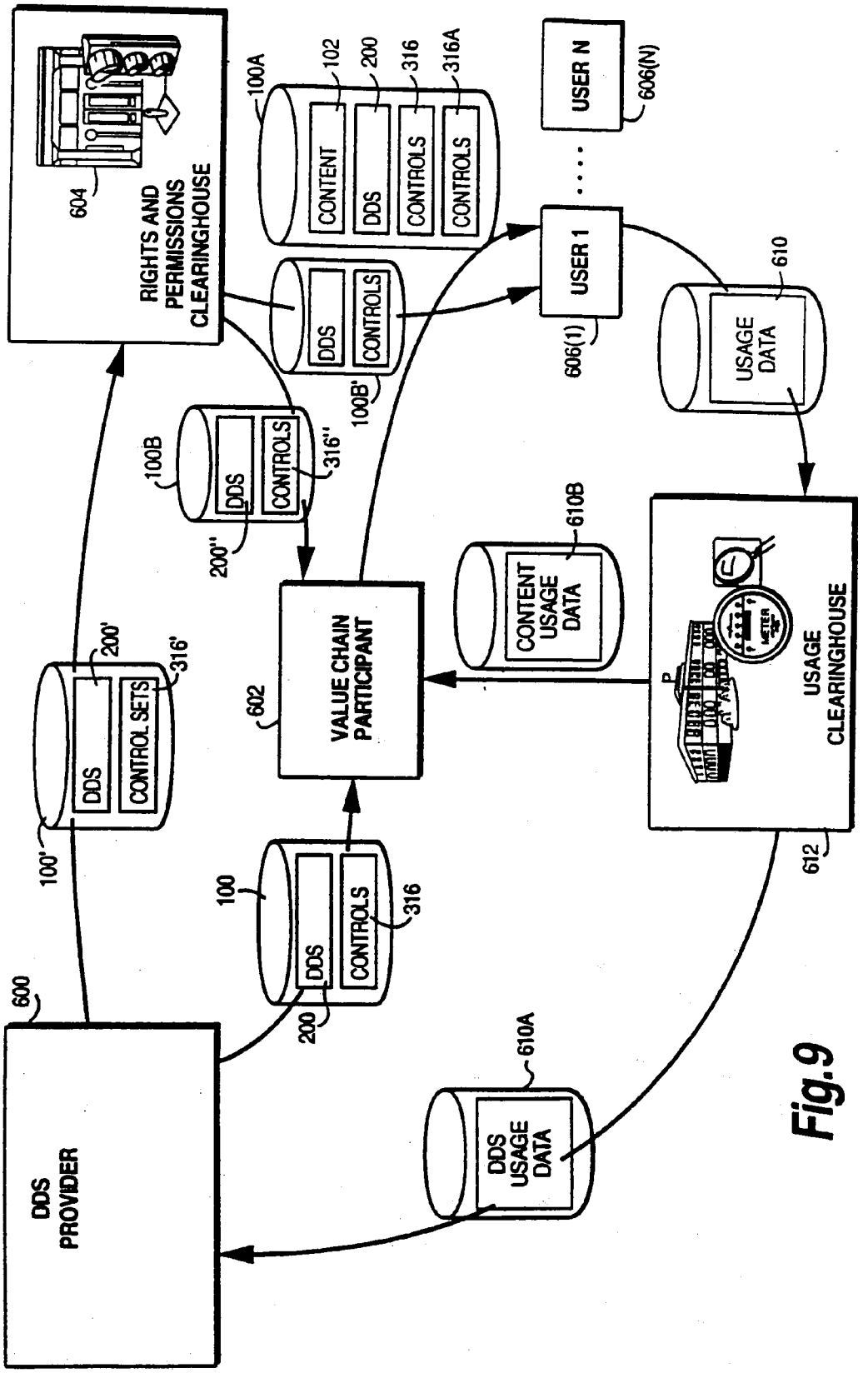

RIGHTS AND PERMISSIONS CLEARINGHOUSE
604
100A
102
200
316
316A
CONTENT
DDS
CONTROLS
CONTROLS
USER N
606(N)
USER 1
606(1)
DDS
CONTROLS
100B'
316''
100B
200''
610
USAGE DATA
610B
CONTENT USAGE DATA
VALUE CHAIN PARTICIPANT
602
USAGE CLEARINGHOUSE
612
200'
316'
DDS
CONTROL SETS
100'
100
316
200
600
DDS PROVIDER
610A
DDS USAGE DATA

Fig.9

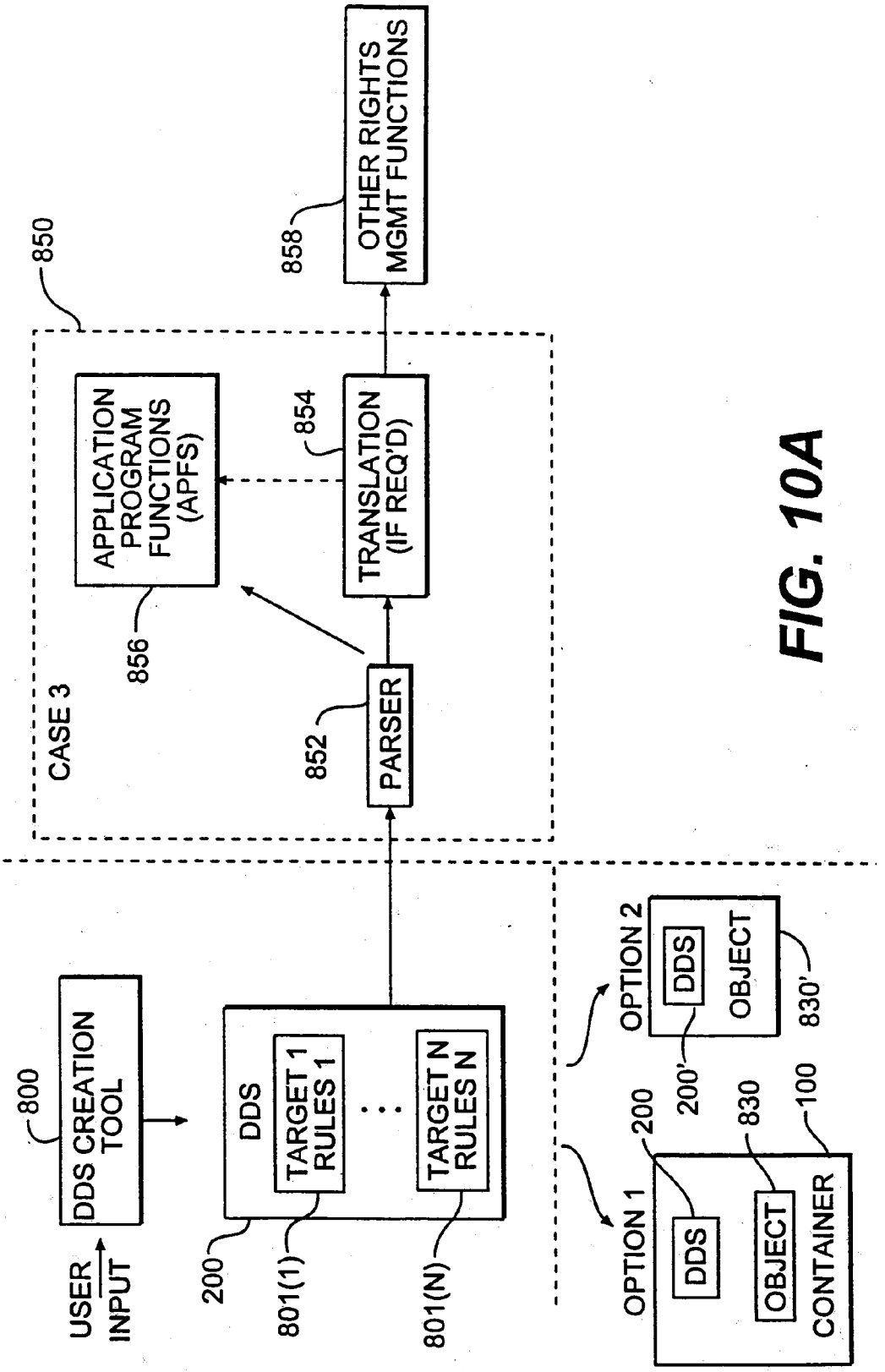

800
DDS CREATION TOOL
USER INPUT
200
DDS
TARGET 1 RULES 1
TARGET N RULES N
801(1)
801(N)
OPTION 1
200
DDS
OBJECT
CONTAINER
100
830
200'
OPTION 2
DDS
OBJECT
830'
CASE 3
850
APPLICATION PROGRAM FUNCTIONS (APFS)
856
TRANSLATION (IF REQ'D)
854
PARSER
852
OTHER RIGHTS MGMT FUNCTIONS
858

FIG. 10A

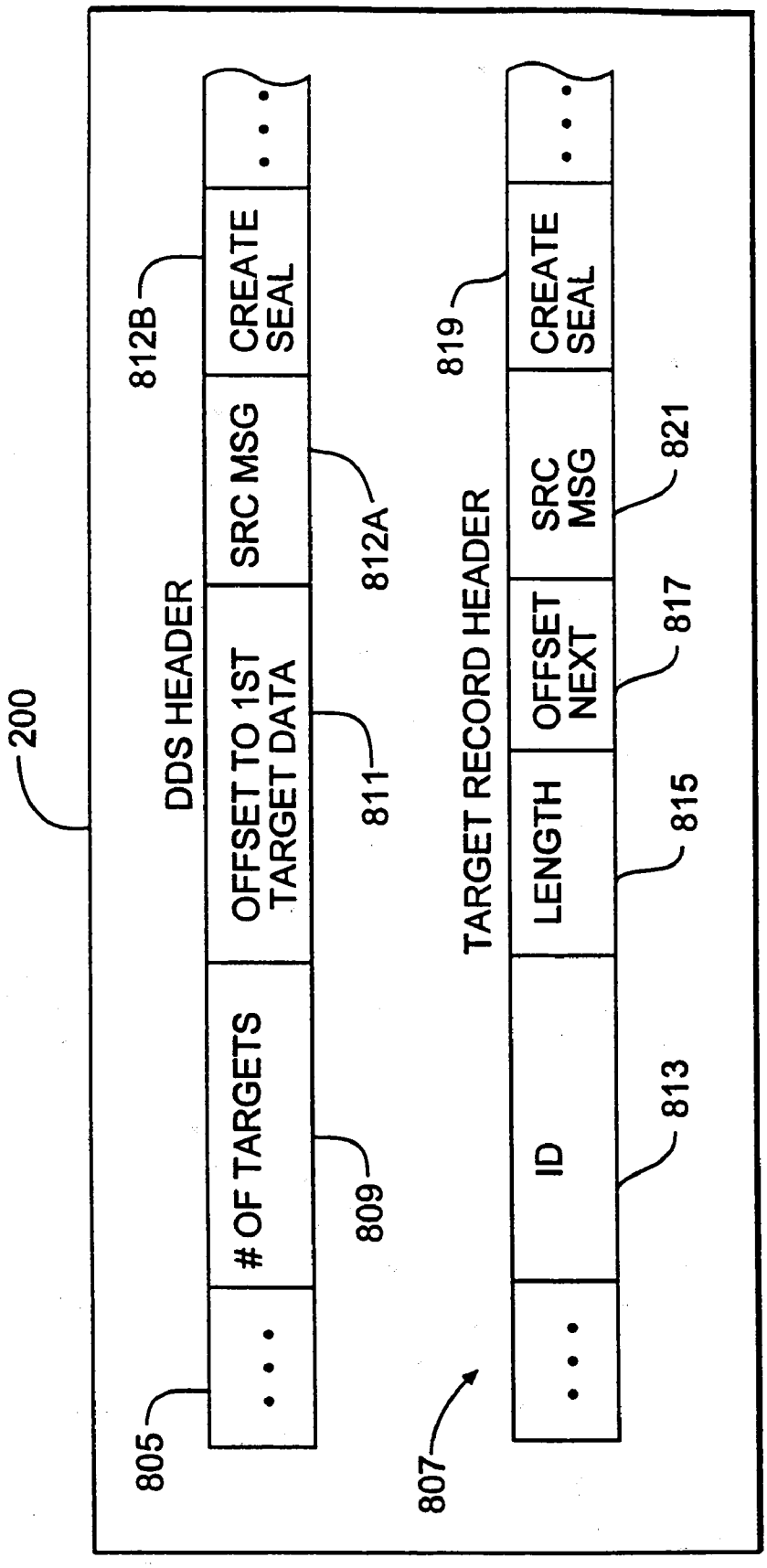

200
DDS HEADER
805
# OF TARGETS
809
OFFSET TO 1ST TARGET DATA
811
SRC MSG
812A
CREATE SEAL
812B
TARGET RECORD HEADER
807
ID
813
LENGTH
815
OFFSET NEXT
817
SRC MSG
821
CREATE SEAL
819

FIG. 10B

# TECHNIQUES FOR DEFINING, USING AND MANIPULATING RIGHTS MANAGEMENT DATA STRUCTURES

## CROSS-REFERENCE TO RELATED APPLICATIONS

**[0001]** This application is a continuation of application Ser. No. 09/819,063, filed Sep. 28, 2000, which is a continuation of application Ser. No. 09/300,778, filed Apr. 27, 1999, now U.S. Pat. No. 6,138,119, which is a continuation of application Ser. No. 08/805,804, filed Feb. 25, 1997, now U.S. Pat. No. 5,920,861, all of which are incorporated herein by reference. This application is related to commonly assigned application Ser. No. 08/388,107 of Ginter et al. entitled "SYSTEMS AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION," filed Feb. 13, 1995, now abandoned; and application Ser. No. 08/699,712 of GINTER et al. entitled "TRUSTED INFRASTRUCTURE SUPPORT SYSTEMS, METHODS AND TECHNIQUES FOR SECURE ELECTRONIC COMMERCE ELECTRONIC TRANSACTIONS AND RIGHTS MANAGEMENT" filed Aug. 12, 1996, now abandoned, both of which are incorporated herein by reference into this application.

## FIELD OF THE INVENTION

**[0002]** This invention relates to techniques for defining, creating, and manipulating rights management data structures. More specifically, this invention provides systems and processes for defining and/or describing at least some data characteristics within a secure electronic rights management container. The present invention also provides techniques for providing rights management data structure integrity, flexibility, interoperability, user and system transparency, and compatibility.

## BACKGROUND AND SUMMARY OF THE INVENTION(S)

**[0003]** People are increasingly using secure digital containers to safely and securely store and transport digital content. One secure digital container model is the "DigiBox™" container developed by InterTrust Technologies Corp. of Sunnyvale Calif. The Ginter et al. patent specification referenced above describes many characteristics of this DigiBox™ container model—a powerful, flexible, general construct that enables protected, efficient and interoperable electronic description and regulation of electronic commerce relationships of all kinds, including the secure transport, storage and rights management interface with objects and digital information within such containers.

**[0004]** Briefly, DigiBox containers are tamper-resistant digital containers that can be used to package any kind of digital information such as, for example, text, graphics, executable software, audio and/or video. The rights management environment in which DigiBox™ containers are used allows commerce participants to associate rules with the digital information (content). The rights management environment also allows rules (herein including rules and parameter data controls) to be securely associated with other rights management information, such as for example, rules, audit records created during use of the digital information, and administrative information associated with keeping the environment working properly, including ensuring rights and any agreements among parties. The DigiBox™ electronic container can be used to store, transport and provide a rights management interface to digital information, related rules and other rights management information, as well as to other objects and/or data within a distributed, rights management environment. This arrangement can be used to provide an electronically enforced chain of handling and control wherein rights management persists as a container moves from one entity to another. This capability helps support a digital rights management architecture that allows content rightsholders (including any parties who have system authorized interests related to such content, such as content republishers or even governmental authorities) to securely control and manage content, events, transactions, rules and usage consequences, including any required payment and/or usage reporting. This secure control and management continues persistently, protecting rights as content is delivered to, used by, and passed among creators, distributors, repurposers, consumers, payment disagregators, and other value chain participants.

**[0005]** For example, a creator of content can package one or more pieces of digital information with a set of rules in a DigiBox secure container—such rules may be variably located in one or more containers and/or client control nodes—and send the container to a distributor. The distributor can add to and/or modify the rules in the container within the parameters allowed by the creator. The distributor can then distribute the container by any rule allowed (or not prohibited) means—for example, by communicating it over an electronic network such as the Internet. A consumer can download the container, and use the content according to the rules within the container. The container is opened and the rules enforced on the local computer or other InterTrust-aware appliance by software InterTrust calls an InterTrust Commerce Node. The consumer can forward the container (or a copy of it) to other consumers, who can (if the rules allow) use the content according to the same, differing, or other included rules—which rules apply being determined by user available rights, such as the users specific identification, including any class membership(s) (e.g., an automobile club or employment by a certain university). In accordance with such rules, usage and/or payment information can be collected by the node and sent to one or more clearinghouses for payment settlement and to convey usage information to those with rights to receive it.

**[0006]** The node and container model described above and in the Ginter et al. patent specification (along with similar other DigiBox/VDE (Virtual Distribution Environment) models) has nearly limitless flexibility. It can be applied to many different contexts and specific implementations. For example, looking at FIGS. **1**A and **1**B, a newspaper publisher can distribute a newspaper **102** within a container **100**A. A publisher of fashion magazines **106** can distribute the fashion magazines within another container **100**C. Similarly, for example, a wholesale banking environment may use yet a further container, an electronic trading system may use a still further container, and so on.

**[0007]** The InterTrust Digibox container model allows and facilitates these and other different container uses. It facilitates detailed container customization for different uses, classes of use and/or users in order to meet different needs and business models. This customization ability is very important, particularly when used in conjunction with a general purpose, distributed rights management environment such as describe in Ginter, et al. Such an environment calls for a

practical optimization of customizability, including customizability and transparency for container models. This customization flexibility has a number of advantages, such as allowing optimization (e.g., maximum efficiency, minimum overhead) of the detailed container design for each particular application or circumstance so as to allow many different container designs for many different purposes (e.g., business models) to exist at the same time and be used by the rights control client (node) on a user electronic appliance such as a computer or entertainment device.

**[0008]** While supporting a high degree of flexibility has great advantages, it can produce difficulties for the average user. For example, think of the process of creating a painting. A master painter creates a painting from a blank canvas. Because the canvas was blank at the beginning, the painter was completely unconstrained. The painting could have been a landscape, a portrait, a seascape, or any other image—limited only by the painter's imagination. This flexibility allows a master painter to create a masterpiece such as the "Mona Lisa." However, great skill is required to create a pleasing Image starting from a blank canvas. As a result, an inexperienced painter cannot be expected to create a good painting if he or she begins with a blank canvas.

**[0009]** Consider now an amateur painter just starting out. That person does not have the skill to transform a blank canvas to a pleasing image. Instead of spending years trying to acquire that skill, the amateur can go out and buy a "paint by numbers" painting kit. Instead of using a blank canvas, the amateur painter begins with a preprinted canvas that defines the image to be painted. By following instructions ("all areas labeled "12" should be painted with dark red," "all areas labeled with "26" should be painted with light blue"), the amateur can—with relatively little skill—paint a picture that is relatively pleasing to the eye. To do this, the amateur must rigidly adhere to the preprinted instructions on the canvas. Any deviations could cause the final image to come out badly.

**[0010]** Ease of use problems in the computer field can be analogized to the "paint by numbers" situation. If it is important for untrained and/or inexperienced users to use particular software, the system designers can predefine certain constructs and design them into the system. This technique allows inexperienced users to make use of potentially very complicated designs without having to fully understand them—but this normally strictly defines, that is severely limits, the functionality and flexibility available by use of the program. As a result, creative solutions to problems are constrained in order to provide practical value. In addition, even the experienced user can find great advantage in using previously implemented designs. Because a user can program a complex program, for example, does not mean it is appropriate or efficient to create a program for a specific purpose, even if the previously implemented program is not ideal. If the creation of a new program "costs" more to create, that is takes too much time or financial resources, the experienced user will normally use a previously implemented program, if available. Therefore, the greatest total amount of value to be realized, related to customization, is to be able to customize with great ease and efficiency so that the cost of customization will not exceed the benefits.

**[0011]** Uniformity, flexibility, compatibility and interoperability are other considerations that come into play in the computer field, particularly in regards to systems supporting customization. In the painting situation, the human eye can appreciate uniqueness—and the "one of a kind" nature of a masterpiece such as the Mona Lisa is a big part of what makes a painting so valuable. In contrast, it is often desirable to make uniform at least the overall layout and format of things in the computer field. It is much more efficient for a computer to know beforehand how to treat and use objects. If the computer doesn't know beforehand how to read or handle an input object, for example, then the computer and the object are said to be "incompatible", i.e., they cannot work together. Computers are said to be "interoperable" if they can work together. Incompatibility and interoperability problems can prevent one computer from talking to another computer, and can prevent you from using computer data created by someone else.

**[0012]** For example, in the non-computer world, a Frenchman who knows only a little English as a second language, might find it far more meaningful and efficient to describe a complex problem in his native tongue, French. But if he is speaking to a second person, an Englishman, and the Englishman does not understand French, the two are not interoperable in French, and the Frenchman must resort to the far less efficient option of speaking in English to the Englishman. Of course, this is far better than if he was trying to speak to a German who understood neither English nor French. Then the two would be not be "interoperable" in regards to discussing the problem. Similarly, because rights management containers may potentially be exchanged and used for a large number of different purposes by a large number of different users, groups, and organizations, it is very important to provide compatibility and interoperability if these different parties, each participating in one or more different rights management models, are to interoperate efficiently. For example, if a rights management container is used to distribute a newsletter and is optimized for this purpose, each reader of the newsletter must have a computer system or software that "knows" how to read the container and the newsletter it contains. Since commerce, such as distributing newsletters, needs to be as efficient and cost-effective as is feasible, it is important to optimize, that is customize, rights management containers to optimally reflect the requirements of their models and not to have unnecessary features for each respective application or class of application, since unnecessary features will require unnecessary computing overhead and/or storage space.

**[0013]** Different newsletter publishers may use different container formats customized to their own particular newsletters and/or content types and/or formats. A newsletter reader interested in many different newsletters may need to be able to read a large number of different formats. It normally will not efficient (or, due to security issues, may not be appropriate) simply to analyze the different containers upon delivery and "try to figure out" or otherwise discern the particular format in use.

**[0014]** Published standards may help achieve a level of interoperability and standards for given types of applications, but it generally takes a long time for any particular standard to achieve industry-wide acceptance and standards will need to vary widely between categories of applications. Moreover, data structure and other standards are often designed to the lowest common denominator—that is, they will carry fields and requirements not needed by some, and miss others features optimal in certain cases. There will always be applications that cannot be optimized for efficiency and/or operation if forced to use a specific standard. Trade-offs between flexibility, ease of use and incompatibility and interoperability

can be further complicated when security considerations come into play. To be effective in many electronic commerce applications, electronic container designs should be tamper-resistant and secure. One must assume that any tools widely used to create and/or use containers will fall into the hands of those trying to break or crack open the containers or otherwise use digital information without authorization. Therefore, the container creation and usage tools must themselves be secure in the sense that they must protect certain details about the container design. This additional security requirement can make it even more difficult to make containers easy to use and to provide interoperability.

**[0015]** The above-referenced Ginter et al. patent specification describes, by way of non-exhaustive example, "templates" that can act as a set (or collection of sets) of control instructions and/or data for object control software. See, for example, the "Object Creation and Initial Control Structures," "Templates and Classes," and "object definition file," "information" method and "content" methods discussions in the Ginter et al. specification. The described templates are, in at least some examples, capable of creating (and/or modifying) objects in a process that interacts with user instructions and provided content to create an object Ginter et al. discloses that templates may be represented, for example, as text files defining specific structures and/or component assemblies, and that such templates—with their structures and/or component assemblies—may serve as object authoring and/or object control applications. Ginter et al. says that templates can help to focus the flexible and configurable capabilities inherent within the context of specific industries and/or businesses and/or applications by providing a framework of operation and/or structure to allow existing industries and/or applications and/or businesses to manipulate familiar concepts related to content types, distribution approaches, pricing mechanisms, user interactions with content and/or related administrative activities, budgets, and the like. This is useful in the pursuit of optimized business models and value chains providing the right balance between efficiency, transparency, productivity, etc.

**[0016]** The present invention extends this technology by providing, among other features, a machine readable descriptive data structure for use in association with a rights management related (or other) data structure such as a secure container. In one example, the machine readable descriptive data structure may comprise a shorthand abstract representation of the format of the data within a rights management related data structure. This abstract data representation can be used to describe a single rights management data structure, or it may be generic to a family of data structures all following the format and/or other characteristics the abstract representation defines. The abstract representation may be used to create rights management data structures, allow others (including "other" rights management nodes automatically) to read and understand such data structures, and to manipulate some or all of the data structures.

**[0017]** The descriptive data structure can be used as a "template" to help create, and describe to other nodes, rights management data structures including being used to help understand and manipulate such rights management data structures.

**[0018]** In one particularly advantageous arrangement, the machine readable descriptive data structure may be associated with one or a family of corresponding rights management data structures—and may thus be independent of any specific particular rights management data structure usage. For example, a copy of the descriptive data structure may be kept with such data structures. Alternatively, some or all of the descriptive data structure may be obtained from somewhere else (e.g., a clearinghouse or repository) and independently delivered on as-needed basis.

**[0019]** In accordance with one example, the machine readable descriptive data structure provides a description that reflects and/or defines corresponding structure(s) within the rights management data structure. For example, the descriptive data structure may provide a recursive, hierarchical list that reflects and/or defines a corresponding recursive, hierarchical structure within the rights management data structure. In other examples, the description(s) provided by the descriptive data structure may correspond to complex, multidimensional data structures having 2, 3 or n dimensions. The descriptive data structure may directly and/or indirectly specify where, in an associated rights management data structure, corresponding defined data types may be found. The descriptive data structure may further provide metadata that describes one or more attributes of the corresponding rights management data and/or the processes used to create and/or use it. In one example, the entire descriptive data structure might be viewed as comprising such metadata.

**[0020]** The machine readable descriptive data structure may or may not be, in part or in whole, protected, depending on the particular application. Some machine readable descriptive data structures may be encrypted in whole or in part, while others might be maintained in "clear" form so that they are easily accessible. Some machine readable description data structures, whether encrypted or not, may be in part or wholly protected for integrity using a cryptographic hash algorithm in combination with a secrecy algorithm to form a cryptographic seal, and/or through use of other protection techniques (including hardware, e.g., secure semiconductor and/or hardware packaging protection means). The machine readable descriptive data structures may themselves be packaged within rights management data structures, and rules (e.g., permissions records) controlling their access and use may be associated with them

**[0021]** In accordance with one aspect of how to advantageously use descriptive data structures in accordance with a preferred embodiment of this invention, a machine readable descriptive data structure may be created by a provider to describe the layout of the provider's particular rights management data structure(s) such as secure containers. These descriptive data structure ("DDS") templates may be used to create containers. A choice among two or more possible DDSs may be based upon one or more classes and/or one or more classes may be based on parameter data. The DDS may be loaded and used as the layout rules for secure containers being created. The provider can keep the DDS private, or publish it so that other providers may create compatible, interoperable containers based on the same DDS.

**[0022]** Descriptive data structures can also be used by a container viewer, browser, reader, or any other end user application designed to work with containers. Truly generic viewers or other applications can be written that can process a container in any format at least in part by making use of descriptive data structures. Thus, a descriptive data structure can be used to at least temporarily convert and/or customize a generic viewer (or other application) into a specialized viewer (or other application) optimized around one or more classes of containers. Additionally, specialized readers may be pro-

vided to efficiently process descriptive data structures to locate key media elements (e.g., cover page, table of contents, advertiser's index, glossary, articles, unprotected preview, price, and/or rights information regarding viewing, printing, saving electronically, redistributing, related budgets and/or other parameter information, etc.).

**[0023]** Such specialized readers can then seamlessly, transparently, and automatically process to present the user with an easy-to-use interface (for example, an icon display for each of the key media elements) optimized for the specific application, container, and/or user. Different and/or differently presented, such elements may be displayed or otherwise employed based, for example, on the identity of the user and/or user node, including, for example, taking into account one or more class attributes which can influence such automated processing.

**[0024]** Two or more DDSs may be associated with a container and/or container contents, as well as, for example, one or more user and/or node classes. A choice among two or more possible DDSs for a given container and/or class of containers and/or container contents may therefore be based upon one or more classes and/or one or more classes based on parameter data. Overall, this ability to easily characterize, and/or reuse stored, optimized, custom container models and subsequent transparency of translation from such customized containers (e.g. specific DDSs) to general purpose rights management use is particularly useful. For example, where such customized DDSs can be used as a basis for the creation of customized, optimized display of container content and/or control information to substantially improve the ease of use, efficiency, transparency, and optimization of a distributed, generalized rights management environment. In such an environment, for example, user nodes can interact with different DDSs to automatically adjust to the requirements of the commercial or other rights models associated with such DDSs.

**[0025]** Some providers may spend considerable time designing sophisticated container descriptive data structures that describe the layout of their associated containers. With this type of investment in structure and format, the descriptive data structure will often have significant value in their reuse for the same or similar applications. Entities can use descriptive data structures in-house to ensure consistent and highly efficient creation of containers. Third party providers (i.e., a provider other than the one responsible for descriptive data structure creation) can use these descriptive data structures when they wish to create containers compatible with other entities. One example is where the publisher of a widely circulated newspaper develops a descriptive data structure for reading its newspaper. Other, smaller newspapers may want to leverage any viewers or other tools put in place for use with the widely circulated newspaper by adopting the same container format. Descriptive data structures can be copyrighted and/or otherwise protectable by both law and by the rights management system itself. For example, they may also be protected by their own containers and associated controls to ensure that descriptive data structure creators, and/or distributors and/or other users of such DDSs, receive their fair, rights system managed, return on their descriptive data structure creation and/or use related efforts.

**[0026]** In addition to the foregoing, the following is a list of features and advantages provided in accordance with aspects of this invention:

**[0027]** Integrity Constraints: The descriptive data structure allows the provider to protect the integrity of his or her content, by enabling the specification of integrity constraints. Integrity constraints provide a way to state integrity related rules about the content.

**[0028]** Application Generation: The descriptive data structure can be used to generate one or more portions of software programs that manipulate rights management structures. For example, a descriptive data structure could serve as 'instructions' that drive an automated packaging application for digital content and/or an automated reader of digital content such as display priorities and organization (e.g., order and/or layout).

**[0029]** Dynamic user interfaces for creation applications: Applications can read a descriptive data structure to generate an interface optimized for data creation, editing, and/or composition for a specific model, including models involving, for example, composing complex content from textual, audio, video, and interactive (e.g., querying) elements. The data may take the form of a container, database and/or any other digital information organization as any simple or compound and complex file format. Applications can also read a descriptive data structure to learn how to best display an interface for collection and/or creation of content.

**[0030]** Dynamic user interfaces for display applications: Applications can read a descriptive data structure to and generate an interface appropriate for data display. This data may be a container, database or any other compound complex file format. Applications can also read a descriptive data structure to learn how to best display an interface for the presentation of content. Applications can further read a descriptive data structure to learn how to manage display functions related to interacting—for content creation and/or packaging and/or user display purposes including optimizing any of such interactions—with other one or more other applications, smart agents, computing environments, identity (including any class identities) of user and/or user nodes, etc. For example, a user interface might be differently optimized for interacting with: a member of the U.S. Air Force versus a faculty member in social sciences at a university; or a member of a Kiwanis Club versus a member of a Protestant church club, a citizen of the United States versus a citizen of Saudia Arabia, including an appropriate display of expected class membership symbols and related, appropriate organization or suppression of displayed information.

**[0031]** Ability to automatically identify and locate data fields: Full text search, agents, web spiders, and the like, benefit and are able to interact with information contained within one or more areas of a DDS when areas within a data file are known to contain potentially interesting information and such information is presented in a predefined format.

**[0032]** Ability to extract needed or desired data without first-hand knowledge of data format: Full text search, agents, web spiders, and the like, benefit and are able to interact with information contained within one or more areas of a DDS when large data files of arbitrary complexity and of unknown origin can be processed without special knowledge.

**[0033]** Efficient, machine/human readable data abstract: The descriptive data structures can be optimally small, convenient, and cost-effective to process, transmit, and/or store.

**[0034]** Reusable, salable—independent of actual data: Descriptive data structures may be arbitrarily complex and therefore potentially time consuming to construct and requiring certain expertise. This gives the descriptive data structure resale value.

On-the-fly definition and redefinition of content layout: Working with a layout tool allows quick iterations (including editing and modifications) of a design (layout) which can be more convenient and cost-effective than creating such a layout, which also may be quite difficult or beyond the expertise of many users.

**[0035]** Descriptive data structure attributes allow for meta-characteristics not found in actual data: Because the same descriptive data structure is processed by both the creation and post-creation processes, meta-information can be placed into the descriptive data structure that would otherwise be unavailable in the packaged content. One example of this whether display of certain fields is "Required" or "Hidden".

**[0036]** Enables design automation via descriptive data structure "wizards": Descriptive data structures themselves enable further automation in the way of "wizards". There can, for example, be descriptive data structures that help to define other descriptive data structures. Descriptive data structures defining other descriptive data structures might represent the incomplete descriptive data structure for a book or magazine, for example. The "wizard" can comprise a series of dialog boxes displayed to the user to fill in the missing information to make it a completed descriptive data structure.

**[0037]** Applications outside of a particular rights management architecture: For example, polymorphous applications may use descriptive data structures to determine certain data visualizations attributes and/or requirements, such as what look and feel should be displayed to the user. For example, if a descriptive data structure contains a word processing document reference, the polymorphous application might create an interface appropriate for display and editing of a document. If the descriptive data structure contains references to many executable programs, the polymorphous application might ask the user where the files should be saved.

**[0038]** Enables umbrella applications to process descriptive data structures and delegate unknown file types and processes: Umbrella (or polymorphous) applications can, for example, act substantially as an operation for a particular data file. This umbrella application may extract and process those things in the data file that it cares about, while ignoring or delegating (to, for example, user and/or value chain partner (e.g., distributor) to control display of such items) those things it does not understand.

**[0039]** Runtime interpretation: It is possible to interpret a descriptive data structure at run time, providing materially increased efficiencies and timeliness.

**[0040]** Runtime adaptability: Systems can adapt to dynamic data arriving in real time through use of descriptive data structures.

**[0041]** Automatic conversion capability: Descriptive data structures be used for converting automatically from one format to another.

**[0042]** Simplified system design: The use of descriptive data structures may greatly reduce the need for a secondary "wrapper" application programming interface (API) or other arrangement to securely "contain" the container creation process. Such a "wrapper" API to control and otherwise restrict the container creation process might otherwise be needed to ensure that all created containers are compatible—thereby limiting flexibility and the ability to customize.

**[0043]** Object oriented template programming environment: The use of display related, interaction related, and rights related concept objects which may be selected through high-level user interface choices and prioritizations and specification of related parameter data, this enabling very easy creation of certain categories of templates—such as construction and display hint information.

**[0044]** The use of a template language and interpreter involving supporting programming through use of language elements and interpretation of such language by nodes described in Ginter, et al., where such language includes elements descriptive of display, rights, and program interaction elements, priorities and parameter data.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0045]** These and other features and advantages of presently preferred example embodiments in accordance with the invention may be better and more completely understood by referring to the following detailed description along with the drawings, of which:

**[0046]** FIGS. **1**A and **1**B show example content containers;

**[0047]** FIGS. **2**A and **2**B show example content containers associated with example descriptive data structures;

**[0048]** FIG. **3** shows an example descriptive data structures creation and usage process;

**[0049]** FIG. **4** shows another example creation and usage process;

**[0050]** FIG. **5** shows an example system architecture using descriptive data structures;

**[0051]** FIG. **5**A shows an example process performed by the FIG. **5** system;

**[0052]** FIG. **6** shows an hierarchical descriptive data structure organization;

**[0053]** FIG. **6**A shows an example of how descriptive data structures can be used with atomic transaction data;

**[0054]** FIG. **7** shows an example descriptive data structure format;

**[0055]** FIG. **8** shows an example descriptive data structure creation graphical interface;

**[0056]** FIG. **9** shows an example process for tracking descriptive data structure rights management related data;

**[0057]** FIG. **10**A shows an example use of descriptive data structures to provide interoperability between environments; and

**[0058]** FIG. **10**B provides more detail about how the FIG. **10**A example descriptive data structure may be organized.

## DETAILED DESCRIPTION OF PRESENTLY PREFERRED EXAMPLE EMBODIMENTS

**[0059]** FIGS. **2**A and **2**B show the example containers **100***a*, **100***c* of FIGS. **1**A, **1**B associated with machine readable descriptive data structures **200** and **200'**. Referring to FIG. **2**A, a descriptive data structure **200** is associated with content container **100***a*. This descriptive data structure **200** may be used to define the content (and certain other charac-

teristics) of container **100***a*. In the example shown, descriptive data structure **200** defines a number of sections of newspaper style content **102** such as, for example, the headline (descriptor **202***a*), the issue date (descriptor **202***b*), the lead story (descriptor **202***c*), breaking news (descriptor **202***d*), image(s) (descriptor **202***e*), advertisement (descriptor **202***f*), and section (descriptor **202***g*).

**[0060]** The descriptive data structure definitions **202** in this example do not contain or specify the particular contents of corresponding portions of the newspaper **102**, but instead define more abstractly, a generic format that a newspaper style publication could use. For example, the FIG. **2**A example descriptive data structure headline definition **202***a* does not specify a particular headline (e.g., "Yankees Win the Pennant!"), but instead defines the location (for example, the logical or other offset address) within the container data structure **100***a* (as well as certain other characteristics) in which such headline information may reside. Because descriptive data structure **200** is generic to a class or family of newspaper style content publications, it can be reused. For example, each daily issue of a newspaper might be created using and/or associated with the same descriptive data structure **200**. By abstractly defining the data format and other characteristics of newspaper style content **102**, the descriptive data structure **200** allows easy creation, usage and manipulation of newspaper style content **102**.

**[0061]** Referring to FIG. **2**B, a different descriptive data structure **200'** may be used to define another class of content publications **106** such as fashion magazines. The descriptive data structure **200'** for this content class reflects a different format (and possibly other characteristics) as compared to the descriptive data structure **200** shown in FIG. **2**A. For example, since fashion magazines typically do not include headlines or breaking news, the example descriptive data structure **200'** may not define such formatting. Instead, descriptive data structure **200'** for defining a class of fashion magazine content may define issue date (descriptor **204***a*), a magazine title (descriptor **204***b*), the name of a photographer (descriptor **204***c*) and associated artwork designation (descriptor **204***d*).

**[0062]** The FIGS. **2**A and **2**B examples show descriptive data structures **200**, **200'** being delivered within content object containers **100***a*, **100***c* along with associated content **102**, **106**. However, other forms of association may be used. For example, descriptive data structure **200** can be independently delivered in its own separate container along with associated rules controlling its access and/or use. Alternatively, descriptive data structures **200** could be stored in a library and delivered on an as needed basis in secure or insecure form depending on particular requirements.

**[0063]** In addition, although FIGS. **2**A and **2**B are printed publication content examples, the use of descriptive data structures **200** is not so limited. To the contrary, descriptive data structures **200** can be used to define the format and/or other characteristics associated with a wide variety of different types of digital information including for example:

**[0064]** images
**[0065]** sound
**[0066]** video
**[0067]** computer programs
**[0068]** methods
**[0069]** executables
**[0070]** interpretables
**[0071]** currency objects
**[0072]** currency containers for currency objects
**[0073]** rules
**[0074]** any computer input
**[0075]** any computer output
**[0076]** other descriptive data structures
**[0077]** any other information.

Example Process for Creating and Using Descriptive Data Structures

**[0078]** FIG. **3** shows an example process for creating and using descriptive data structures **200**. In this example, a layout tool **300** is used to create descriptive data structure **200**. This layout tool **300** may be, for example, a software-controlled process interacting with a human being via a graphical user interface. The resulting descriptive data structure **200** (which may be stored on a mass storage device or other memory) can then be used to facilitate any number of other processes to create or interpret stored data. For example, the descriptive data structure may be used in a creation process **302**. The creation process **302** may read the descriptive data structure and, in response, create an output file **400** with a predefined format such as, for example, a container **100** corresponding to a format described by the descriptive data structure **200**. A viewing process **304** may use the descriptive data structure **200** to locate important items in the output file **400** for display. A browsing process **306** may use the descriptive data structure **200** to locate items within the stored output file **400** such as, for example, key words or other searchable text. Descriptive data structure **200** may supply integrity constraints or rules that protect the integrity of corresponding content during use of and/or access to the content.

**[0079]** FIG. **4** shows a more detailed example descriptive data structure creation and usage process. In this example, the layout tool **300** may accept user input **310** provided via a graphical user interface **312**. The output of the layout tool **300** may be a descriptive data structure **200** in the form of, for example, a text file. A secure packaging process **302***a* may accept container specific data as an input, and it may also accept the descriptive data structure **200** as a read only input. The packager **302***a* could be based on a graphical user interface and/or it could be automated. The packager **302***a* packages the container specific data **314** into a secure container **100**. It may also package descriptive data structure **200** into the same container **100** if desired. A viewer **304** may view data **314** with the assistance of the descriptive data structure **200** and in accordance with rules **316** packaged within the container applying to the data **314** and/or the descriptive data structure **200**.

Example Architecture for Using Descriptive Data Structures

**[0080]** FIG. **5** shows an example secure system architecture suitable for use with descriptive data structure **200**. In this example, an electronic appliance **500** of the type described in the above-referenced Ginter et al. patent specification may be provided within a tamper resistant barrier **502**. Electronic appliance **500** may include an application program interface (API) **504**. One or more applications **506** may communicate with electronic appliance **500** via API **504**. In some examples, the application **506** may execute on the secure electronic appliance **500**. Each application **506** may include a descriptive data structure interpreter **508**. In use, electronic appliance **500** may access secure container **100** and—in accordance with rules **316**—access the descriptive data structure **200** and

content **102** it contains and provide it to application **506**. The interpreter **508** within application **506** may, in turn, read and use the descriptive data structure **200**. In addition, application **506** may be polymorphic in the sense that it can take on personality or behavior as defined at least in part by descriptive data structure **200**.

**[0081]** FIG. **5**A shows an example detailed process performed by the FIG. **5** example secure system architecture. In this example, application **506** asks appliance **500** to retrieve the descriptive data structure **200** from container **100** (block **550**). Electronic appliance **500** reads the descriptive data structure **200** and, subject to the conditions specified by associated rules **316**, provides the descriptive data structure **200** to the application **506** (block **552**). Application **506** then asks its interpreter **508** to interpret the descriptive data structure **200** (block **554**). The interpreter **508** tells the application **506** what the descriptive data structure **200** says (block **556**). The application **506** extracts or obtains the descriptive data structure information it needs or wants from interpreter **508** (block **558**). For example, suppose the application **506** wants to display the "headline" information within newspaper style content shown in FIG. **2**A. Application **506** may ask interpreter **508** to provide it with information that will help it to locate, read, format and/or display this "headline" information.

**[0082]** As another example, interpreter **508** may provide application **506** with an element identification (e.g., a hexadecimal value or other identifier) that corresponds to the headline information within the newspaper style content (block **558**). Application **506** may then ask electronic appliance **500** to provide it with the Headline (or other) content information **102** within container **100** by providing appropriate content information to electronic appliance **500** via API **504** (block **560**). For example, application **506** may pass the electronic appliance **500** the element ID that interpreter **508** provided to the application. Even though application **506** may have no direct knowledge of what is inside container **100** (and may only be able to access the container **100** through a secure VDE node provided by appliance **500**), interpreter **508** (by looking at descriptive data structure **200**) can tell application **506** enough information so that the application knows how to request the information it wants from the electronic appliance **500**.

**[0083]** The electronic appliance may then access information **102** within container **100**, and deliver (in accordance with the rules **316** within the container) the requested information to the application **506** (block **562**). The application **506** may then use the information electronic appliance **500** provides to it, based at least in part on what interpreter **508** has told it about the content information (block **564**). For example, the descriptive data structure **200** may provide characteristics about the way application **506** should handle the information **102**. Descriptive data structure **200** can, for example, tell application **506** to always display a certain field (e.g., the author or copyright field) and to never display other information (e.g., information that should be hidden from most users). DDS **200** can also provide complete presentation or "visualization" information so that an information provider can, for example, control the look and feel of the information when it is displayed or otherwise rendered. Descriptive data structure **200** may provide encodings of other characteristics in the form of metadata that can also be used by application **506** during a process of creating, using or manipulating container **100**. The DDS **200** can be used to generate a software program to manipulate rights management structures. For example, a DDS **200** could serve as the 'instructions' that drive an automated packaging application for digital content or an automated reader of digital content.

Example Description(s) Provided by Descriptive Data Structure

**[0084]** FIG. **6** shows one example of how a descriptive data Structure **200** may describe and define an arbitrarily complex, information structure such as, for example, an hierarchical container **100**. In this particular example, container **100** includes properties **600(1)**, **600(2)**. Property **600(1)** may include n attributes **602(1)**, **602(2)** . . . **602(*n*)**. Property **600(2)** may include any number of attributes **604(1)**, **604(2)**, and it may also include an additional property **606**. Property **606** may, in turn, include its own attributes **608(1)**, **608(2)** . . . . Associated descriptive data structure **200** may be organized as a tree structure list **250** providing a recursive structure to reflect the recursive structure of the contents of container **100**. For example, list **250** may include "branches" in the form of "property" descriptors **252(1)**, **252(2)** corresponding to properties **600(1)**, **600(2)**. Each property descriptor **252** may, in turn, include a list **254** of attributes and may include additional property descriptors **256** in the same recursive, hierarchical arrangement as is reflective of the example content container structure. DDS **200** may be used to describe arbitrarily complex, hierarchical or non-hierarchical data structure of any dimension (1 to n).

**[0085]** FIG. **6**A shows that descriptive data structure **200** can be used in conjunction with any kind of information such as, for example, events or methods defining an "atomic transaction" such as a real estate transaction. In this FIG. **6**A example, a container **100** includes one or more descriptive data structures **200** and associated control set(s) **316** relating to a sequence of "events" **700** that define a real estate transaction. The DDS **200** may, for example, include a number of different entries **200**A-**200**N pertaining to each different "event" within the transaction (e.g., "offer", "acceptance", "purchase/sales", "inspection", "mortgage", etc.). These entries **200**A-N may, for example, define where in container **100** the event can be found. The entries **200**A-**200**N may also include metadata that provides additional characteristics corresponding to the event (for example, how certain information related to the event should be displayed).

Example Descriptive Data Structure Formatting

**[0086]** FIG. **7** shows an example of how descriptive data structure **200** may be formatted. As mentioned above, descriptive data structure **200** may comprise a list such as a linked list. Each list entry **260(1)**, **260(2)**, . . . may include a number of data fields including, for example:

**[0087]** an object name field **262**,

**[0088]** one or more metadata fields **264** (which may be part of and/or referenced by the descriptive data structure); and location information **266** (which may be used to help identify the corresponding information within the container data structure **100**).

**[0089]** The object name field **262** may include a constant that may corresponds to or describes a type of information. For example, object name field **262** may act as a "handle" to the content or data; it may be an indirect reference to the content or data; and/or it may be used to look up the content or data The following are examples of object names:

**[0090]** General Purpose Object Names
**[0091]** NUMBER
**[0092]** STRING
**[0093]** DATE
**[0094]** TITLE
**[0095]** DESCRIPTION
**[0096]** AUTHOR
**[0097]** PROVIDER
**[0098]** MIME_TYPE
**[0099]** VERSION
**[0100]** URL
**[0101]** EMAIL
**[0102]** NEWGROUP
**[0103]** FILE_NAME
**[0104]** KEYWORDS
**[0105]** CREATION_DATE
**[0106]** MODIFICATION_DATE
**[0107]** LAST_ACCESS_DATE
**[0108]** NATIVE_PLATFORM
**[0109]** SIZE
**[0110]** CONTENT
**[0111]** PREVIEW
**[0112]** THUMBNAIL
**[0113]** TEXT
**[0114]** ARTWORK
**[0115]** ILLUSTRATION
**[0116]** UNKNOWN
**[0117]** TEMPLATE
**[0118]** BILLING_NAME
**[0119]** CONTAINER
**[0120]** Book-style Object Names
**[0121]** DEADLINE_DATE
**[0122]** TITLE_PAGE
**[0123]** PROLOGUE
**[0124]** INTRODUCTION
**[0125]** ABSTRACT
**[0126]** TABLE_OF_CONTENTS
**[0127]** CHAPTER
**[0128]** CHAPTER_NUMBER
**[0129]** INDEX
**[0130]** Electronic Mail-style Object Names
**[0131]** FROM
**[0132]** TO
**[0133]** CC
**[0134]** SUBJECT
**[0135]** MESSAGE_BODY
**[0136]** ENCLOSURE
**[0137]** Newspaper-style Object Names
**[0138]** ISSUE_DATE
**[0139]** ARTICLE
**[0140]** COLUMN
**[0141]** COVER_STORY
**[0142]** LEAD_STORY
**[0143]** BREAKING_NEWS
**[0144]** ADVERTISEMENT
**[0145]** SECTION
**[0146]** EDITORIAL

**[0147]** The DDS **200** may include or reference any type of data or metadata. In one example, the DDS **200** uses the object name field **262** to points or refers to metadata. This metadata can define certain characteristics associated with the object name. For example, such metadata may impose integrity or other constraints during the creation and/or usage process (e.g., "when you create an object, you must provide this information", or "when you display the object, you must display this information"). The metadata **264** may also further describe or otherwise qualify the associated object name.

**[0148]** In one preferred example, the DDS **200** uses object name **262** to refer to metadata stored elsewhere—such as in a container **100**. This referencing technique provides several advantages. For example, one situation where it may be useful to store the metadata in a secure container **100** separately from DDS **200** is in situations where it is desirable to make the DDS readily accessible to an outside application but to protect the associated metadata. For example, consider the case of handling web spider queries. A web spider may query the DDS **200** for a particular object name **262**. If the object name is found, then the web spider may request the corresponding metadata. The web spider may have ready access to the metadata, but may only be able to access the associated metadata from the container **100** under appropriate conditions as controlled by a corresponding secure electronic appliance **500** based on associated rules **316**. As another example, storing metadata separately from the DDS **200** may allow the same DDS to be used with different metadata in different contexts. Suppose for example that a DDS **200** contains an Object Name, for example KEYWORDS. When DDS **200** is associated with container **100**A then the DDS Object Name KEYWORDS refers to container **100**A's KEYWORDS metadata. Conversely, if later this same DDS **200** is associated (e.g., packaged with) a different container **100**C, then the DDS Object Name KEYWORDS refers to container **100**B's KEYWORDS data.

**[0149]** Although it is preferred to use object name **262** to refer to metadata stored elsewhere, there may be other instances where there is a need or desire to explicitly include metadata within the DDS **200**. For purposes of illustration, FIG. **7** shows an example DDS **200** that includes metadata field **264** and also refers to metadata within a container **100** using the object name **262**. Either or both techniques may be used.

**[0150]** The DDS **200** thus allows value chain participants to protect the integrity of content, by enabling the specification of integrity constraints. DDS **200** integrity constraints provide a way to state rules about the content. For example, DDS **200** can specify that an article of a newspaper cannot be viewed without its headline being viewed. The corresponding integrity constraint can indicate the rule 'if there is an article, there must also be a headline". Another example is a photograph that is part of a magazine and the credit that goes with it. The integrity constraint rule provided by DDS **200** might be 'do not present this photograph without its associated credit'.

**[0151]** DDS integrity constraints give value chain participants a tool for protecting the use of the DDS **200**, ensuring that content represented by a particular DDS contains all the essential components—that it is representative of the DDS. This gives providers a way to set up conventions and enforce standards of use. There are many possible integrity constraints. The following are a few examples:

**[0152]** Required: a is required as part of the content Optional: a is an optional component of the content

**[0153]** Required relationship: if a is present, then b must be present, or if a is present b, c and d must be present. Conversely, if b is not present, then a is not allowed to be present. Relationships in this category are 1:m where $m>0$.

**[0154]** Optional relationship: If a is present b may or may not be present. If b is present, then a is guaranteed to be present. Relationships in this category are 1:n, where n>=0.

**[0155]** Repetition: a must occur n times where n>1. This could be specified with ranges of values, etc.

**[0156]** Other rules and/or requirements.

Metadata **264**

Example Graphical Interface For Creating Descriptive Data Structures

**[0157]** FIG. **8** shows an example descriptive data structure creation graphical user interface **312**. In this example, the graphical user interface **312** may prompt the user for the object name. In addition, the graphical user interface **312** may provide options for specifying the associated metadata **264**. The options shown in FIG. **8** may, for example, include:)

**[0158]** "construction type" metadata (upon object construction, the information is required; upon object construction, the object creation tool is to always or never prompt for the information);

**[0159]** display metadata (e.g., always display the associated information (e.g., for copyright notices, author names and the like) or always or never make the information visible; and/or

**[0160]** layout "hints" and field definitions (e.g., text, text block, integer, file, image or other data type).

The above metadata descriptions are non-limiting examples. Other metadata characteristics and attributes may be used.

Example Process Using Descriptive Data Structures

**[0161]** FIG. **9** shows one example arrangement for using the infrastructure described in co-pending related U.S. patent application Ser. No. 08/699,712 (referenced above) for descriptive data structures **200**. The arrangement shown in FIG. **9** may be useful in a number of different contexts. For example, a provider **600** of descriptive data structures **200** may want to know which descriptive data structures **200** are the best liked by his customers so he or she can improve the quality of his products. Or, a provider **600** may charge customers for using descriptive data structures **200** on a per use or other basis. In still another example, some descriptive data structures **200** or classes of DDS **200** may be restricted to use only by authorized users or classes of authorized users.

**[0162]** FIG. **9** shows a DDS provider **600** who delivers a DDS **200** and an associated control set **316** to a value chain participant **602**. Controls **316** may provide rules and associated consequences for controlling or otherwise affecting the use or other aspects of what value chain participant **602** can do with DDS **200**. The controls **316** and DDS **200** may be packaged within a container **100**. Value chain participant **602** may get the container **100** containing DDS **200** directly from DDS provider **600**; alternatively, the provider can provide it a rights and permissions clearinghouse **604** and participant **602** and get it from the clearinghouse (or elsewhere) (see container **100**B).

**[0163]** Value chain participant **602** can use DDS **200** to author content **102**. Participant **602** can package content **102** with associated controls **316**A in a container **100**A. Participant **600** may, if he desires, include DDS **200** and associated controls **316***a*, **316***b* with content **102** in the same container—or depend on the provider **600** and/or rights and permissions clearinghouse **604** to independently deliver the DDS and its controls to end users **606** in another container **100***c* for example.

**[0164]** End users **606**(**1**), . . . , **606**(*n*) use DDS **200** (in accordance with controls **316**) in conjunction with content **102** (for example, to read, browse or otherwise access the container content). Controls **316**, **316**A may require user appliances to provide usage data **610** to a usage clearinghouse **612**. The usage clearinghouse **612** can provide usage data **610**A related to access and/or usage of DDS **200** to DDS provider **600**, and may independently provide usage data **610**B related to access and/or usage of content **102** to value chain participant **602**.

Descriptive Data Structures can be Used to Achieve

A Degree of Interoperability Between Rights Management Environments

**[0165]** Descriptive data structures **200** provided in accordance with the present invention can provide a degree of interoperability between source and target rights management environments, and/or to provide a bridge to achieve at least some degree of interoperatibility between a rights management environment and the outside world.

**[0166]** Different rights management environments may have substantially incompatible mechanisms for defining rights pertaining to an object. Descriptive data structures **200** can provide at least a partial bridge to achieve a degree of compatibility and interoperability. For example, a provider that defines an object within a source rights management environment may create a descriptive data structure for use by processes within one or more target rights management environments. For example, an object creator or other provider can specify, within a descriptive data structure **200**, certain rules, integrity constraints and/or other characteristics that can or should be applied to the object after it has been imported into a target rights management environment. The target rights management environment can choose to selectively enforce such rules, constraints and/or other characteristics depending on the degree to which it can trust the source environment. For example, objects imported from an EDI system employing X.12 security may be more trustworthy than objects presented from environments with lesser (or no) security.

**[0167]** In another example, a provider that creates an object outside of any rights management environment can create a descriptive data structure **200** for use if and when the object is imported into one or more rights management environments. The target rights management environment(s) can use such descriptive data structure(s) to help efficiently understand and handle the object. Further, a descriptive data structure created within a rights management environment can be exported to one or more applications outside of the rights management environment and used to assist the application(s) in interpreting exported content or other information.

**[0168]** FIG. **10**A shows an example of how descriptive data structures **200** may be used to provide interoperability. In the FIG. **10**A example, a DDS creation tool **800** creates a DDS **200** that includes one or more target data blocks **801**. In one example, the DDS creation tool **800** may be based on, and/or incorporate some or all of the capabilities of layout tool **300** and provide interoperability capabilities in addition to features associated with layout tool **300**. In another example, DDS creation tool **800** may not incorporate any of the capa-

bilities of layout tool **300**, and may create DDS **200** solely for interoperability purposes. DDS creation tool **800** may, for example, be an application program with a graphical user interface, a background process that only displays a user interface when being configured by a user, a portion of an operating system, a portion of a computer's firmware, a server process that may act independently or as part or all of a "gateway" between one system and another (e.g., a public network and a private network, two or more private networks, a local area network and a wide area network, etc.), or any other desirable implementation or integration.

**[0169]** Target data block **801** may provide information used to provide interoperability with a particular target environment **850**. A single DDS **200** can, in one example, provide interoperability with N different target environments **850** by including N target data blocks **801**(**1**), . . . **801**(N) each corresponding to a different target environment **850**(**1**), . . . **850**(N).

**[0170]** In this example, each target data block **801** includes rule (control) information. Different target data blocks **801** can provide different rule information for different target environments **850**. The rule information may, for example, relate to operations (events) and/or consequences of application program functions **856** within the associated target environment **850** such as specifying:

**[0171]** permitted and/or required operations;

**[0172]** nature and/or extent of operations permitted and/or required operations; and/or

**[0173]** consequences of performing permitted and/or required operations.

The target data block **801** may also include additional information if desired that gives directions to a DDS parser **852** and/or a translator **854** within a corresponding target environment **850**.

**[0174]** FIG. **10**B shows one detailed example of how target information may be organized within DDS **200**. In this example, DDS creation tool **800** creates a DDS header **805** that references one or more target record headers **807**. DDS header **805** may, for example, include a "number of targets" field **809** indicating the number of target data blocks **801** within the DDS **200**, a "offset to first target data portion" field **811** that provides the location of the first target data block **801**(**1**) within the DDS **200**, a source message field **812**A that identifies the source environment, and an optional creator seal **812**B that may be used to verify the integrity and authenticity of the DDS **200**. Source message field **812**A (which can be optional) may include a source ID that may be used to help verify the source environment of DDS **200**, and an optional source seal (that may or may not be present in the source message). Each target data block **801** within DDS **200** may begin with a target record header **807** including a "target ID" field **813**, a "length" field **815**, a "offset to next target data portion" field **817**, an optional creator seal **819**, and an optional source message **821**. The "target ID" field **813** may specify a unique identification number or value corresponding to the associated target data block **801** and/or identifying the intended target environment(s), the "length" field **815** may specify the length of the target data block **801**, and the "offset" field **817** may specify the location (relative or absolute) of the next target data block **801** within the DDS **200** (and may take on a null value for the last target data block).

**[0175]** The optional creator seals **812**B, **819** (and source seals) may be cryptographic seals that help to ensure that the DDS **200** and target records **801**, respectively, have not be altered since they were created, and also the identity of the DDS **200**'s creator and/or source. The optional source messages **812**C and **821** may be information that helps to ensure that a target environment knows which source environment created DDS **200**.

**[0176]** Referring again to FIG. **10**A, DDS creation tool **800** may, upon creating the DDS **200**, cryptographically seal it and each target data block **801** for integrity using appropriate cryptographic processes, for example by first running a cryptographic hash function (e.g., SHA, MD5, etc.) on the data and then encrypting the resulting hash value using a private key of the DDS creator associated with an asymmetric cryptosystem (e.g., RSA, El Gamal, etc.). If sealing is used, the DDS creator preferably should ensure that the public key associated with the encrypting private key is certified (e.g., encrypted with a private key of a certifying authority) and available for use by target environments to validate the seal (e.g., by including a certificate in DDS **200**, publishing the certificate on a public network, etc.)

**[0177]** If source messages **812**C, **821** are used, they should preferably represent information provided by the source environment that may help a target environment identify the source environment, and further may also help to ensure that the DDS **200** was actually created by the source environment (and therefore may, for example, be trusted to the extent that the source environment is trusted). For example, a source environment may have a protected processing environment (PPE) of the form described in the above referenced Ginter, et al. patent application. Certain of such PPEs may have cryptographic keys (e.g., a private key of a public key/private key pair) available that may be used to encrypt a cryptographic hash taken of the DDS header **805** or target block header **807**, as appropriate. In such an example, a target environment would need to acquire a corresponding cryptographic key (e.g., a public key of a public key/private key pair) using trusted techniques (e.g., delivery in a certificate signed by a trusted certifying authority) in order to evaluate such a source message. In another example, DDS creation tool **800** may have been equipped with cryptographic keys when it as manufactured, and may use these cryptographic keys instead of keys from a PPE, although generally this technique would be more susceptible to tampering by an experienced computer hacker and might therefore be somewhat less trusted by target environments.

**[0178]** In addition, or alternatively (for example, if cryptographic techniques are not appropriate or desired), the source message may contain a unique identifier that corresponds to the source environment.

**[0179]** The DDS creation tool **800** (see FIG. **10**A) may then package the resulting DDS **200** into a secure container **100** along with an associated object **830**. In another example, DDS creation tool **800** may embed DDS **200** within, or otherwise associate the DDS with, an object **830**' that provides a method for releasing the DDS to the target environment parser **852**. The DDS **200** and its associated object **830** may then be delivered to one or more target environments **850** for processing.

**[0180]** Target environment parser **852** (and/or translator **854**) may, for example, be part of an application program, part of an operating system, or part of a utility program used by, or in conjunction with, an application program and/or an operating system. The target environment parser **852** receives the DDS **200** and parses it to locate the target data block **801**(*k*)

corresponding to the target environment **850**(*k*). Parser **852** may then determine, from the corresponding target data block **801**, the rules the target data block contains. Parser **852** preferably understands enough about the structure of DDS **200** to find (e.g., using the header information shown in FIG. **10**B) the appropriate target data block **801** corresponding to it, and also to understand the rules within the target data block. The target environment parser **852** doesn't need to understand any additional rules **316** that may be packaged within container **100** or otherwise delivered with object **830**, but it may use any such additional rules if desired (e.g., when it finds no target data block **801** within DDS **200** for the particular target environment **850** (for example, if it is capable of understanding some other target data block **801** whose rules are based on a published specification and/or standard)).

**[0181]** The target environment parser **852** may obtain applicable target rules from target data block **801** and provide these rules to application program functions **856**. Application program functions **856** may define any operation pertaining to object **830** such as for example:

**[0182]** cut

**[0183]** copy print

**[0184]** paste

**[0185]** save

**[0186]** change

**[0187]** delete

**[0188]** any other operation.

**[0189]** The target rules provided by parser **852** may be used, for example, to permit, require and/or prevent certain operations; to define the extent to which certain operations can be performed (e.g., limit number of copies, define extent of cut, the rules that should be applied to cut information in subsequent use, etc.); and/or to define the consequences of performing a particular operation (e.g., charge the user for printing or otherwise using and/or accessing all or part of object **830**, maintain records of the time and/or number of such operations performed, etc.).

**[0190]** Parser **852** may also, or alternatively, provide some or all of the rules it obtains from target data block **801** to other arrangements for applying the rules such as, for example, the "other rights management functions" block **858**. Block **858** may provide any kind of rights management functions. Translator **854** may be used if needed to allow the application program functions **856** and/or the "other rights management" block **858** to understand the rules. As one example, translator **854** may be used to further elaborate, parameterize and/or secure the rule information obtained from target data block **801** so they are more or fully compatible with the "other rights management functions" block **858**.

**[0191]** A useful data structure definitional method and arrangement has been described in connection with its most practical and presently preferred example embodiments. The present invention is not to be limited to those embodiments, but on the contrary, is intended to encompass variations and equivalents as defined within the spirit and scope of the claims.

**1**. A data processing method comprising:

creating a machine readable, abstract descriptive data structure; and

using the representation to interoperate with at least one rights management data structure.

**2**. A method as in claim **1** wherein the using step includes the step of formatting at least one part of at least one rights management data structure at least in part in accordance with the descriptive data structure.

**3**. A method as in claim **1** wherein the using step includes the step of formatting display of at least one part of at least one rights management data structure at least in part in accordance with the descriptive data structure.

**4**. A method as in claim **1** wherein the using step includes the step of formatting reading of at least one part of at least one rights management data structure at least in part in accordance with the descriptive data structure.

**5**. A method as in claim **1** wherein the using step includes the step of displaying at least a part of at least one rights management data structure based at least in part on the descriptive data structure.

**6**. A method as in claim **1** wherein the creating step includes the step of providing metadata within the descriptive data structure, and the displaying step comprises displaying at least some information from the rights management data structure at least in part in accordance with the metadata.

**7**. A method as in claim **1** wherein the using step includes the step of dynamically generating a user interface based at least in part on the descriptive data structure.

**8**. A method as in claim **1** wherein the using step includes the step of automatically identifying and/or locating at least one data field at least in part based on the descriptive data structure.

**9**. A method as in claim **1** wherein the using step includes the step of automatically extracting data within the rights management data structure based at least in part on the descriptive data structure.

**10**. A method as in claim **1** wherein the creating step comprises creating a descriptive data structure that is independent of any particular rights management data structure but abstractly describes a class of rights management data structures.

**11**. A method as in claim **1** wherein the creating step includes the step of creating metadata for defining at least one characteristic of the using step.

**12**. A method as in claim **1** wherein the creating step includes the step of creating the abstract representation at least in part by using a wizard, the operation of the wizard being defined at least in part by a further descriptive data structure.

**13**. A method as in claim **1** wherein the using step includes the step of altering the behavior of a polymorphous process at least in part based on the descriptive data structure.

**14**. A method as in claim **1** wherein the using step includes the step of interpreting at least part of the descriptive data structure at run time.

**15**. A method as in claim **1** wherein the using step includes the step of dynamically adapting at least part of data processing of the rights management data structure at run time.

**16**. A method as in claim **1** wherein the using step includes using at least part of the descriptive data structure as instructions for driving and automated digital content handler.

**17**. A method as in claim **1** wherein the creating step includes the step, of creating at least one integrity constraint, and the using step includes the step of enforcing the integrity constraint.

**18**. In a rights management data processing architecture of the type including a secure electronic appliance that interacts

with an application through an interface, a method of interoperating with secure electronic containers comprising the following steps:

(a) delivering an abstract data structure representation to the application;

(b) generating container access requests with the application based at least in part on the abstract data structure representation; and

(c) accessing the container with the secure electronic appliance at least in part based on the container access requests the container generates.

**19**. A method as in claim **18** further including the steps of:

(d) providing, with the secure electronic appliance, information from the container to the application; and

(e) processing the provided information at least in part in accordance with the abstract data structure representation.

**20**. A method as in claim **19** wherein the processing step (e) includes the step of processing the provided information in accordance with metadata provided within the abstract data structure representation.

**21-28**. (canceled)

* * * * *