

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

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
21 March 2002 (21.03.2002)

PCT

(10) International Publication Number  
**WO 02/23875 A1**

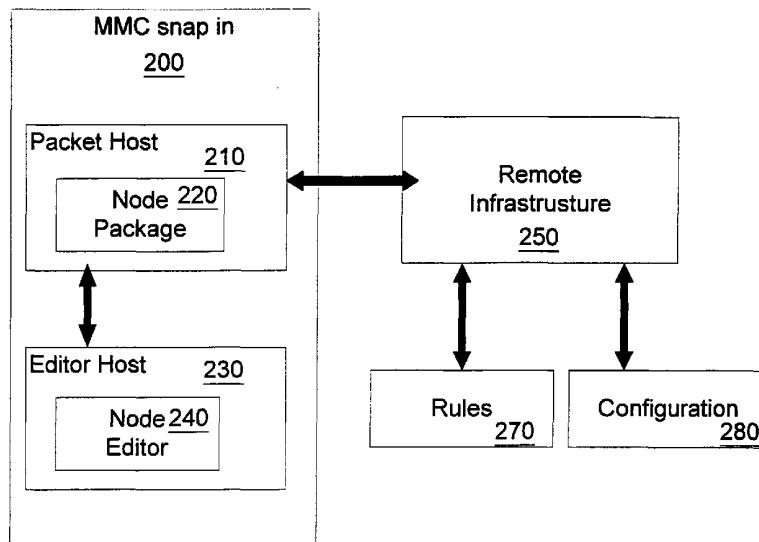
- (51) International Patent Classification<sup>7</sup>: **H04M 3/00**, G06F 17/30, 11/00, 15/16, 13/00
- (21) International Application Number: PCT/US01/28955
- (22) International Filing Date: 14 September 2001 (14.09.2001)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data: 60/232,731 15 September 2000 (15.09.2000) US
- (71) Applicant: **WONDERWARE CORPORATION** [US/US]; 100 Technology Drive, Irvine, CA 92618 (US).
- (72) Inventors: **HESSMER, Rainer**; 7 Via Floria Rancho Santa, Margarita, CA 92688 (US). **TODOROV, Ivan, A.**; 150 Las Flores, Aliso Viejo, CA 92656 (US). **HADRICH, Michael**; Thaddaus-Robl-Strasse 7, 80935-München (DE). **ROSS, Louis, D.**; 3518 NW Satinwood Street, Corvallis, OR 97330 (US).
- (74) Agents: **JOY, Mark et al.**; Leydig, Voit & Mayer, Ltd., Suite 4900, Two Prudential Plaza, 180 North Stetson, Chicago, IL 60601-6780 (EP).
- (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, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.
- (84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— with international search report

[Continued on next page]

- (54) Title: A METHOD AND SYSTEM FOR REMOTE CONFIGURATION OF PROCESS DATA ACCESS SERVERS



(57) Abstract: A remote configuration utility architecture is disclosed for a manufacturing/process control system data access server. Server agents, located on remote nodes (250), notify a configuration console of existing data access servers on the system. The configuration utility includes a control console from which a user selects one of the identified data access servers. Thereafter, the configuration console, via remote interfaces, obtains configuration parameters (280) and associated rules (270) associated with the selected data access server. The configuration utility thereafter displays the retrieved configuration information within a user interface faceplate defined for a type of configuration node selected for display/editing from the selected DAS. Using various faceplates for selected node types, the user creates, clears, examines and/or manipulates hierarchically arranged nodes for a configuration associated with the selected data access server.

**WO 02/23875 A1**



*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.*

## A METHOD AND SYSTEM FOR REMOTE CONFIGURATION OF PROCESS DATA ACCESS SERVERS

### CROSS REFERENCE TO RELATED APPLICATION

5 This application claims priority of Todorov et al. U.S. provisional application Serial No. 60/232,731 filed on September 15, 2000, entitled "Remote Multiple Client Protocol Support," the contents of which are expressly incorporated herein by reference in their entirety including the contents and teachings of any references contained therein.

### 10 FIELD OF THE INVENTION

The present invention generally relates to the field of computerized process control networks. More particularly, the present invention relates to configuration utilities that access server components within manufacturing/process control networks to tailor the operation of the server components. An example of such a server component is a data access server system that 15 supports access by supervisory level client applications to process control information.

### BACKGROUND OF THE INVENTION

Significant advances in industrial process control technology have vastly improved all aspects of factory and plant operation. Before the introduction of today's modern industrial 20 process control systems, industrial processes were operated/controlled by humans and rudimentary mechanical controls. As a consequence, the complexity and degree of control over a process was limited by the speed with which one or more people could ascertain a present status of various process state variables, compare the current status to a desired operating level, calculate a corrective action (if needed), and implement a change to a control point to affect a 25 change to a state variable.

Improvements to process control technology have enabled vastly larger and more complex industrial processes to be controlled via programmed control processors. Control processors execute control programs that read process status variables, execute control algorithms based upon the status variable data and desired set point information to render output 30 values for the control points in industrial processes. Such control processors and programs support a substantially self-running industrial process (once set points are established).

Notwithstanding the ability of industrial processes to operate under the control of programmed process controllers at previously established set points without intervention, supervisory control and monitoring of control processors and their associated processes is desirable. Such oversight is provided by both humans and higher-level control programs at an application/human interface layer of a multilevel process control network. Such oversight is generally desired to verify proper execution of the controlled process under the lower-level process controllers and to configure the set points of the controlled process.

Data access servers facilitate placing process control data within reach of a variety of higher-level monitoring/control client applications. During the course of operation, process controllers generate status and control information concerning associated processes. The controllers' process status and control information is stored within process control databases and/or distributed to a number of locations within the process control network. Other process information is generated/stored within field devices (e.g., intelligent transmitters) having digital data communication capabilities. The process information is retrieved from the databases and field devices by data servers for further processing/use by the process control system. For example, the data access servers provide the retrieved information to a variety of client applications providing high-level control and monitoring (both human and computerized) services.

In systems containing data access servers, high-level control and monitoring applications rely upon the proper configuration of the data access servers to provide information upon which such applications rely for decision-making. Such information includes real-time process variable values, alarms, etc. Manufacturing/process control systems are modified due to changes in the process control devices and the processes themselves. Many data access servers operate in complex process control computing environments in a time-critical manner. In very large systems, with hundreds, or even thousands, of data access servers spread across hundreds or thousands of computers in many buildings, the cost (in time and resources) of configuring the operation of running data access servers and the devices from which they receive their data grows exponentially with the number of such servers. The delay in completing tuning operations also increases. Such delays are costly to manufacturers. Therefore, manufacturers generally seek to minimize the delays encountered when tuning the operation of a data access

server and its associated process control devices. Thus, it is important in such instances to provide a means for quickly configuring multiple data access servers and to minimize the time that the process stands idle.

Moreover, it is important to quickly remedy configuration faults in an

- 5 industrial/manufacturing process managed by a process control system. A data access server that is malfunctioning or unable to function because of an improper configuration can result in significant downtime for a manufacturing process. In many applications, if the information or control pathway provided by a data access server is unavailable, whole production lines can be brought to a standstill.

## SUMMARY OF THE INVENTION

The present invention comprises a new architecture and method for configuring data access servers from a remote location, thereby enabling a manufacturing/process control system administrator to configure multiple data access servers without having to physically go to each node on a network executing the data access servers. The present invention establishes a centralized utility which allows configuring a set of networked data access servers (DAS) from one location.

In accordance with the present invention, a distributed configuration architecture facilitates remote configuration of process control data access servers. A system embodying the present invention includes a control console from which a user operates a configuration editor. The configuration editor includes a user interface infrastructure. The configuration system also includes a configuration database that stores information describing specific configuration parameters for identified data access servers. The configuration system further includes a rules database for storing a set of configuration rules associated with data access servers that are configurable via the configuration system. The set of configuration rules guide the construction and/or editing of a configuration definition for a data access server. In a particular embodiment of the invention, the rules are specified for each node type.

In an embodiment of the present invention, remote connection to the configuration rules and parameters for a particular data access server is facilitated by a server agent that is executed upon remote nodes capable of running data access servers. The server agents include executable procedures for notifying the configuration editor of the existence of data access servers upon their respective nodes. An aspect of a particular embodiment of the present invention is the capability of the configuration system to support custom editor faceplates for each of the various types of configurable component (node) types supported within the configuration system.

## BRIEF DESCRIPTION OF THE DRAWINGS

The appended claims set forth the features of the present invention with particularity.

The invention, together with its objects and advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

5 FIGURE 1 is a schematic drawing depicting an exemplary process control environment for the present invention wherein a process data access server includes a remote configuration utility that retrieves configuration information within other data access servers that, in turn, retrieve/receive process control information and provide such information to a variety of client applications residing at a monitoring/supervisory layer of a process control network;

10 FIG. 2 depicts a multi-tiered software arrangement for carrying out remote configuration;

FIG. 3 is a schematic drawing identifying a set of components making up the client and server components of a remote configuration facility in accordance with an embodiment of the present invention;

15 FIG. 4 depicts a set of interfaces supported by a data access server Package Host in accordance with an exemplary embodiment of the present invention;

FIG. 5 depicts a set of interfaces supported by an Editor Host in accordance with an exemplary embodiment of the present invention;

FIG. 6 depicts a set of interfaces supporting access to a remotely stored configuration database;

20 FIG. 7 is a sequence diagram summarizing a set of steps performed to open a node editor to commence editing a configuration for a selected data access server on a remote node;

FIG. 8 is a sequence diagram summarizing a set of steps for adding a node to a configuration hierarchy; and

25 FIG. 9 is a sequence diagram summarizing a set of steps for saving changes made via the configuration facility in a remote configuration storage.

## DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

The remote configuration utilities and complimentary interfaces incorporated into data access server systems incorporating the present invention enable users to configure a data access server from a single remote computer. An exemplary configuration utility architecture is disclosed for a data access server remote configuration carried out via an MMC (MICROSOFT Management Console) Snap-in. Through the remote configuration snap-in a user creates, clears, examines and/or manipulates a selected data access server configuration for any data access server within its local node or on any network-connected node.

Turning initially to FIG. 1, an exemplary portion of a process control network 10 is illustratively depicted. As previously mentioned, the present invention is useful in any network including a data access server. However, the value of performing remote configuration on data access servers increases with increases in the number of data access servers and the distance between monitoring stations and data access servers within a manufacturing/process control network. The process control network 10 can be viewed as a set of devices connected to one or more network links associated with particular levels of the process control network 10. In the exemplary embodiment, the depicted portion of the process control network 10 includes a fieldbus level 12, a local control level 14, and a supervisory control level 16. Though the exemplary embodiment is depicted as having three levels, those skilled in the art will readily appreciate the applicability of the present invention to a number of process control network architectures having more, less, or the same number of network levels. The illustratively depicted network 10 embodies a multi-level bus topology. However, the present invention can be incorporated into process control networks embodying alternative network topologies (e.g., star networks, hybrid bus/star networks, etc.) including both single-level and hierarchical configurations.

In the exemplary portion of a process control network depicted in FIG. 1, a set of intelligent field devices 20 reside at the fieldbus level 12. The field devices include intelligent process variable transmitters that sense pressure, temperature, fluid flow, etc. in a controlled industrial process. The field devices also include actuators such as those enabling opening and closing fluid flow valves for tanks, burners, etc.

Control processors 30 at the local control level 14, perform local control functions with regard to the set of intelligent field devices 20. The control processors 30 receive process state information provided by the intelligent field devices 20. State information includes, for example pressure, temperature, mass flow, volumetric flow, etc. The control processors apply the 5 received status information to a set desired points for the process, and then transmit control signals to actuators in order to obtain or maintain the desired set points. The control processors are programmed/configured to store the status and control information associated with their control function.

The supervisory control level 16 includes higher level control applications programs that 10 facilitate and/or implement enterprise/plant level decision making and supervisory (e.g., set point) control value designation functions. An alarm server 40 receives process status data from a number of lower level sources, including both the control processors 30 and the field devices 20. The alarm server 40 compares the received status data against a set of alarm/event conditions and issues appropriate notifications to either monitors or control processes (e.g., control 15 processors 30) in response to a detected alarm/event condition. The control processors 30 issue appropriate signals to controlled field devices/actuators to address the event/alarm condition. A historian 42, also operating at the supervisory control level 16, archives data received from any of the aforementioned levels of the process control system. Such data is available for auditing and verification by a variety of application programs. A human machine interface (HMI) 44 is 20 yet another node connected to the supervisory control level 16. The human-machine interface 44 provides a set of graphic/text user interface functions enabling a human to view the operation/status of the controlled process associated with the process control system with which the depicted items of FIG. 1 are associated.

In an exemplary embodiment of the present invention, a set of data access server nodes 25 50a, 50b, 50c are interposed between the supervisory control level 16's processing nodes and the lower levels of the process control system (e.g., the local control level 14 and fieldbus level 12). The data access server node 50b, executing one or more logical DA servers, receives and/or extracts data from the field devices 20 (via channel 52) and/or the control processors 30 (via channel 54) and provides corresponding (possibly reformatted) data to processing nodes at the 30 supervisory control level 16 of the process control network 10 – including one or more of the

nodes executing one of the other data access servers. The data access server nodes 50a-c perform the task of providing data to a variety of client applications that obtain data in accordance with particular data exchange protocols and are otherwise unable to access process control data provided at the local control level 14 and fieldbus level 12. A method and system for supporting multiple client data exchange protocols is described in **Todorov et al. U.S. patent application (number not yet assigned) filed on September 14, 2001, and entitled "An Industrial Process Control Data Access Server Supporting Multiple Client Data Exchange Protocols,"** which is incorporated herein by reference in its entirety including any references therein.

A configuration utility executed upon the data access server (DAS) 50c (indicated in FIG. 1 by window 56), any of the other data access servers, or any supervisory-level 16 node, provides a means for remotely configuring process control system data access servers. In an embodiment of the invention, the data access server configuration utility is an application that executes upon a same node as a data access server. For example, as depicted in FIG. 1, the configuration utility executes upon a data access server node 50c. However, there is no requirement for the configuration utility to reside upon any particular type of node (since it executes independently of DAS data acquisition/transmission processes). In the illustrative embodiment of the present invention, the configuration utility (tool), through DAS agents instantiated on at least nodes 50a and 50b (to query the registry on each of those nodes) discovers the data access servers running on DAS nodes 50a, 50b, and 50c and presents a list to a user.

The DAS agents are a tool for discovering data access servers on a network. The user is then able to select one or more of the data access servers of interest to perform configuration. Once connections are established between the configuration utility running on DAS node 50c and the data access servers, other components within the data access servers take over and handle information requests from the configuration utility. Those components are discussed herein below with reference to FIG. 2. It is also noted that in an embodiment of the present invention, the configuration utility program is incorporated into a base library included within each of the data access server nodes 50a, 50b, and 50c, and thus there is no need on the part of an administrator to add any special programs or executable code to support the operation of the configuration utility within the network.

In the illustrative embodiment set forth in **FIG. 1**, a processing engine that carries out the core functionality of each DAS node 50a-c hosts the configuration utility and exposes a default set of configuration information via a set of well-defined interfaces described herein below with reference to **FIGS. 4-6**. As mentioned herein above, data access server node functionality can be 5 enhanced, through customization of a server-specific part of the DA servers, by extending the scope of configurable features exposed by the data access servers to the configuration utility.

Furthermore, it is reiterated that the present invention is not limited to any particular process control system network topology or technology. For example, the disclosed exemplary process control network comprises a hierarchically arranged digital system. However, in an 10 alternative network embodiment the present invention is incorporated within a monitoring node connected to a single-level process control network wherein the field devices, control processor(s), and supervisory control applications constitute nodes on a single bus. In yet other cases, the DAS receives data concerning conventional, analog field devices that utilize the 4-20 milliamp standard for process communications.

15 A number of features contribute to enhancing the configuration of data access servers from a remote location. First, the configuration utility incorporates a generic multi-tier architecture. The distinguishable tiers comprise an Interface, a Rule Interpretation, and a Configuration Persistence tier. Second, the configuration utility supports hot (on-line) 20 configuration at the above-mentioned levels. Third, an arbitrary number of hierarchy levels can be defined. Fourth, the configuration utility supports performing remote configuration. Each of these features is discussed in more detail below.

With reference to **FIG. 2**, the generic multi-tier software architecture of the configuration utility comprises a data services layer 100, a business services layer 110, and a presentation 25 services layer 120. The tiers are implemented via a ‘rules’ file, carried out at the data services layer 100, that is associated with (specified for) each data access server. The data access server-specific rules files specify a set of configuration hierarchy levels and information about properties that are defined for configuration nodes at each hierarchy level. The information about properties includes: minimum/maximum number of child nodes allowed, names and types

of child nodes allowed, default property values, minimum/maximum property values, and property names and types.

The "rules" file is implemented, by way of example, by a set of XML definitions.

However, as those skilled in the art will readily appreciate, other languages/formats can be

5 utilized to specify rules. The encapsulation of the rule file access within the bottom tier allows future modifications to this scheme without affecting any other layers of the configuration hierarchy. Also contained within the rules file are the GUIDs of the ActiveX faceplates that are activated for each hierarchy level.

The hierarchical organization supports a generic architecture allowing changes at one

10 level to not affect other, unchanged, levels. The main configuration MMC snap-in at the Presentation Services tier 120 uses the Business Services tier 110 as a "rules interpreter/enforcer" that prevents a user from building an invalid hierarchy structure in a tree-view of the configuration. The Business Services tier 110 also ensures that data entered by a user is valid in view of rules specified by a particular data access server's rules file.

15 With regard to a second aspect of the disclosed remote configuration architecture, hot configuration refers to allowing a user to modify portions of a server's configuration and have those changes incorporated into the running and active server. The actual pushing of updated configuration parameters relies on an interface, specified by a server developer, that facilitates notifying a server of its configuration changes. A DAS Agent tracks currently active servers on a  
20 DAS node, and a user is allowed to make configuration modifications that are allowed for a current server state.

With regard to a third aspect of the disclosed remote configuration architecture, at any level within the hierarchy of a configuration tree (not to be confused with the program hierarchy depicted in FIG. 2), the 'Rules' file defines configuration nodes (leaves and/or branches) that can  
25 be added to the tree under the current node. Thus, configuration tree architecture is defined by a data access server developer, and there is no pre-defined hierarchy structure that would otherwise restrict such development.

With regard to a forth aspect of the disclosed remote configuration architecture, the MMC Snap-in that implements the configuration utility is executable on any network-accessible node  
30 in relation to a particular data access server (may be remote or local). Inter-node

communications are carried out via DCOM. A DAS agent, installed on all server nodes, handles server locating/interfacing functions. Through DAS agents, the client (configuration utility) queries installed servers and/or currently running servers.

5        Turning to FIG. 3, a configuration software component arrangement is schematically depicted for implementing remote configuration information extraction/manipulation via a configuration utility installed on the data access server node 50c. An MMC snap-in module 200, also referred to herein as the DAS control client (DASCC 200), is written according to the requirements of Microsoft MMC utility and constitutes the main executable module invoked by a  
10 user from the MMC of WINDOWS 98 (and later WINDOWS versions) Resource Toolkit. The MMC snap-in represents one of many potential ways to bundle the components of the configuration utility. A Package Host 210 is a module (DLL) that encapsulates the communication within the persistence area for remote configuration. The persistence area is a combination of a physical storage device and a logical organization of configuration/rules stored  
15 upon the physical storage device. For example, a persistence area can be a file with XML text stored upon a storage node. Alternatively, the persistence area is partially defined as a relational database on a storage node. The package host 210 fulfills the following tasks: (1) extracting available data from the persistence area and making it available for node editors, (2) validating and persisting the values provided by node editors into the persistence area, and (3) provide  
20 services to the DASCC relating to creating and deleting new/existing configuration hierarchy nodes. A node package 220 is a server-specific (i.e., user customized from toolkit template) module providing custom validation logic for the business services 110.

An editor host 230 module (DLL) is a generic host for all custom faceplates (Node Editors) that a user developer may create in support of custom configuration interfaces for  
25 editing particular node types. The editor host 230 provides a common GUI look and feel for all configuration hierarchies – such as for instance “Apply” the changes, “Restore” parameter values to the last persisted values, etc. A node editor 240, like the node package, is a custom faceplate specific to each node type supported in a DAS configuration utility. The node editor 240 is typically provided by a data access server developer and encapsulates the details of a particular  
30 node type configured on the editor host 230. In an embodiment of the invention, a library of

node editors 240 (one per node type supported by the configuration system) is stored on the DAS control client 200's computer node. However, this is not a requirement, and indeed in alternative embodiments of the invention the node editor instances (derived from a generic Node Editor interface definition) are stored on a remote node containing, for example, the configuration and rule information for a particular DAS.

A remote infrastructure 250, stored upon each of the remote nodes containing an accessible DAS, comprises a set of modules facilitating remote configuration. The remote infrastructure 250 comprises, by way of example, two modules (and can be extended in case of future needs). The first, DASConfigAccess.dll, facilitates communicating with the storage/persistence area for purposes of accessing/storing the current configuration and rules for a selected data access server. This interface abstracts the configuration editor from the various manners in which the configuration parameters and rules are stored on a system. This allows the configuration editor to operate with a variety of storage area formats (e.g., files, databases, etc.) without modification. access methods associated with particular storage format and access methods. The second, DASAgen.exe, facilitates querying the nodes for installed/active DAS instances, and reports back to the user's remote configuration utility.

Rules 270 are a persistence area, typically specified by a user during remote configuration and then stored at the remote DAS node, containing a description of all hierarchy parameters and their relations/associations. In an embodiment of the present invention, XML listings are stored within a rules file, but alternative implementations are also contemplated such as a database. In an analogy to the C++ programming language, the Rule file corresponds to a class, while a configuration 280 corresponds to a set of instances of the "rule" defined class. The configuration 280 is a persistence area (exists after remote configuration by a user ends) component of remote configuration identifying/describing all configured hierarchy nodes and their associated parameters. Again, while XML is used in an exemplary embodiment, other embodiments of the invention specify the configuration in a different format/language (e.g., a database).

**FIGs. 4, 5 and 6** comprise classes of objects associated with the remote configuration facility and the classes' associated interfaces. Turning **FIG. 4**, a set of interfaces are depicted that are implemented in a CNodeEditor class 290 from which a Node Editor 240 is instantiated. A CNodeEditor class object is created for each type of configuration node based upon the interfaces

specified herein below. An IDASEditor interface 295 provides a mechanism for the Editor Host 230 to communicate with the Node Editor 240. An IDASEditorEvents 310 interface (described herein below with reference to an object class associated with the Editor Host 230) provides a mechanism for the Package Host 210 to notify a Node Editor 240 about context changes.

5    ***Declaration***

```

[object,
uuid(320C0A36-AB7C-11d4-93E4-00B0D0201D61),
pointer_default(unique)
10 ]
/*
IDASEditor is an interface implemented by each Node Editor. It allows the
Editor host to initialize the Node Editor and notify it about extenal events.
*/
15 interface IDASEditor : IUnknown
{
    HRESULT Initialize (
        [in] IDASPackageSite *pPackage,
        [in] IDASEditorSite *pHostSite,
20        [in] BSTR bstrFullNodeName,
        [in] BSTR bstrNodeName,
        [in] BSTR bstrDelimeter,
        [in] VARIANT_BOOL bIsReadOnly,
        [in, unique] VARIANT* pVarAux);
25    HRESULT Apply ();
    HRESULT Restore();
    HRESULT Close();
}
30 {vb}

```

**Operations**

**Apply**

/\*Notifies the Node Editor to persist (store) current data\*/

35    ***Declaration***

HRESULT Apply ()

***Return Value***

<b>S_OK</b>	The status change was noted.
<b>E_FAIL</b>	Failure

**Close**

/\*Notifies the Node Editor that editor will be closed\*/

40    ***Declaration***

`HRESULT Close()`

*Return Value*

- |               |                         |
|---------------|-------------------------|
| <b>S_OK</b>   | CleanUp was successful. |
| <b>E_FAIL</b> | Failure                 |

**Initialize**

/\*Initialize the Node Editor with parameters allowing it to communicate with DAS infrastructure\*/

5    **Declaration**

```
HRESULT Initialize (
    [in] IDASPackageSite *pPackage,
    [in] IDASEditorSite *pHostSite,
    [in] BSTR bstrFullHostName,
    [in] BSTR bstrNodeName,
    [in] BSTR bstrDelimiter,
    [in] VARIANT_BOOL bIsReadOnly,
    [in, unique] VARIANT* pVarAux)
```

*Return Value*

- |               |                              |
|---------------|------------------------------|
| <b>S_OK</b>   | The status change was noted. |
| <b>E_FAIL</b> | Failure                      |

15   **Restore**

*Declaration*

```
HRESULT Restore()
    /*Instruct the Node Editor to restore the last persisted values*/
```

*Return Value*

- |               |                              |
|---------------|------------------------------|
| <b>S_OK</b>   | The status change was noted. |
| <b>E_FAIL</b> | Failure                      |

20

With continued reference to **FIG. 4**, a set of interfaces are depicted that are implemented in a CDASEditorHost class 300 from which an Editor Host 230 is instantiated. One Editor Host object is instantiated for each DAS instance. An IDASEditorEvents 310 interface provides a mechanism for the Package Host 210 to notify an Editor Host 230 about context changes. The 25 context changes include, for example: a change in the node name and a change in the parameters exposed by a particular Node Editor 240. IDASEditorEvents 310's operations are used to notify the Editor Host 230 and Node Editor 240 about events requiring action by these entities. In the case of the Editor Host 230, such calls are used, for example, by the Package Host 210.

**Declaration**

30   [

```
    object,
    uuid(320C0A33-AB7C-11d4-93E4-00B0D0201D61),
```

```

        pointer_default(unique)
    }
interface IDASEditorEvents : IUnknown
{
5    HRESULT OnNameChange
    (
        [in, string] BSTR bstrFullOldNodeName,
        [in, string] BSTR bstrOldNodeName,
        [in, string] BSTR bstrNewNodeName
10
    );
    HRESULT OnAttributeChange
    (
        [in, string] BSTR bstrFullNodeName,
15        [in, string] BSTR bstrAttrName,
        [in] DWORD PropID,
        [in] VARIANT varnewValue
    );
20}

```

**Description**

{vb}

**OnAttributeChange**

/\*Notifies the Node Editor about parameter changes\*/

25 **Declaration**

```

HRESULT OnAttributeChange
(
    [in, string] BSTR bstrFullNodeName,
    [in, string] BSTR bstrAttrName,
30    [in] DWORD PropID,
    [in] VARIANT varnewValue
)

```

**Parameters**

<i>BstrFullNodeName</i>	Specifies the full node name
<i>BstrAttrName</i>	Specifies the attribute name
<i>PropID</i>	Specifies the property ID
<i>VarnewValue</i>	New attribute Value

35 **Return Value**

S_OK	Success
------	---------

**OnNameChange**

/\*Notifies the Node Editor about node name changes\*/

**Declaration**

```

HRESULT OnNameChange
{
    [in, string] BSTR bstrFullOldNodeName,
    [in, string] BSTR bstrOldNodeName,
5     [in, string] BSTR bstrNewNodeName

}

Parameters
    bstrFullOldNodeName      Specifies the full old node name
    BstrOldNodeName          Specifies the node name
    BstrNewNodeName          Specifies the node name

```

*Return Value*

**S\_OK** Success

10

An IDASEditorHost interface 320 is implemented by EditorHost components and provides common functionality between the individual Node Editors. For instance it hosts the custom Node Editors and handles events such as when the user seeks to persist the new configuration data or restore the last persisted configuration data.

15 **Declaration**

```

[
    object,
    uuid(320C0A34-AB7C-11d4-93E4-00B0D0201D61),
    pointer_default(unique)
20 ]
interface IDASEditorHost : IUnknown
{
    HRESULT Initialize (
        [in] IDASPackageSite *PackageSite,
25        [in] BSTR strEditorGuid,
        [in] BSTR strFullPath,
        [in] BSTR strNodeName,
        [in] BSTR strNodeType,
        [in] BSTR bstrDelimiter);

30    HRESULT Close(
        [in] BOOL bForce);

        HRESULT IsDirty(
35        [out] BOOL* dirty);
}

```

**Operations****Close**

40 /\*Notification that hosted Configuration Node Editor will be closed\*/

*Declaration*

```
HRESULT Close(
    [in] BOOL bForce)
```

*Parameters*

*BForce* Flag specifying whether to Close and discard changes without prompting user to save changes or not:  
 TRUE - discard changes (if any) and do not prompt user  
 FALSE - prompt user if changes since last 'Apply'

5    *Return Value*

**S\_OK**    Success.

*Initialize*

/\*Initialize that editor host with parameters allowing it to communicate with the storage area\*/

*Declaration*

```
HRESULT Initialize (
10    [in] IDASPackageSite *PackageSite,
    [in] BSTR strEditorGuid,
    [in] BSTR strFullPath,
    [in] BSTR strNodeName,
    [in] BSTR strNodeType,
15    [in] BSTR bstrDelimiter)
```

*Parameters*

<i>PackageSite</i>	Interface to the Storage area
<i>strEditorGuid</i>	GUID for the custom editor
<i>strFullPath</i>	This is fully qualified name of the parent.
<i>strNodeName</i>	Specifies the Node Name
<i>strNodeType</i>	Specifies is Node Type
<i>bstrDelimiter</i>	Specifies is Node Type

*Return Value*

**S\_OK**    Success.

*IsDirty*

/\*Checks whether some of the parameters in the Node Editor were changed (state 'dirty') since the last "Apply"\*/

*Declaration*

```
HRESULT IsDirty(
    [out] BOOL* dirty)
```

*Parameters*

*Dirty*    Flag indicating whether the editor is "dirty" (has changed attributes)  
 TRUE - the attributes has been change  
 FALSE- no change from the last save operation

25    *Return Value*

**S\_OK**     Success.

An IDASEditorSite interface 330 provides a mechanism for the Node Editor 240 to communicate to the Editor host 230.

*Declaration*

```
5   [
    object,
    uuid(F4136354-C2A0-47db-89F1-463ADEDFFDFF),
    pointer_default(unique)
]
10  interface IDASEditorSite : IUnknown
{
    HRESULT OnStatusChange();
}
```

*Description*

15 {vb}

Operations

**OnStatusChange**

*Declaration*

HRESULT OnStatusChange()

20 *Return Value*

**S\_OK**     The status change was noted.

Referring to **FIG. 5**, a set of interfaces are depicted that are implemented in a CDASPackage class 400. An IDASPackage interface 410 provides the mechanism for a Node Editor 240 to access the storage area. The IDASPackage interface 410 also provides services to the DAS custom console (or DASCC). For instance when a user adds a new node to a configuration. The DASCC presents an indication of “available” nodes in view of rules specified for a particular stored node. For example, a user configures in the rule file a node of type “TCP.” The TCP node can have up to 3 child nodes of type PLC and up to 5 PyramidIntegrator nodes. After the user adds 3 PLC nodes, the user will not be allowed to add more nodes of this type because the quota was exhausted. This test for configuration rule compliance is facilitated by the information provided by the PackageHost 210 to the DASCC 200. The Package Host 210 “compiles” the information from the rules and configuration and provide the results of the process to the DASCC 200 or Node Editors 240.

***Declaration***

```

15  [
16      object,
17      dual,
18      uuid(320C0A37-AB7C-11d4-93E4-00B0D0201D61),
19      pointer_default(unique)
20 ]
21 /*
22  IDASPackage provides information to the DASCC about the current DAS
23  configuration and notifies the Package Host about user commands (like
24  add/delete a node for example )
25 */
26 interface IDASPackage : IDispatch
27 {
28     HRESULT Initialize(
29         [in, string] BSTR bstrServerGUID,
30         [in] IUnknown* pCfgRules
31     );
32
33     HRESULT GetChildren (
34         [in, string] BSTR bstrFullnodeName,
35         [in, string] BSTR bstrNodeDelimiter,
36         [out] SAFEARRAY(BSTR) *bstrArrayChildrenNames,
37         [out] SAFEARRAY(BSTR) *bstrArrayChildrenTypes,
38         [out] SAFEARRAY(BSTR) *bstrArrayChildrenDelimiters
39     );
40
41     HRESULT GetPossibleChildTypes (
42         [in, string] BSTR bstrFullnodeName,
43     );
44 }

```

```
[out] SAFEARRAY(BSTR) *bstrChildTypes
);
HRESULT GetNodeProperties (
[in, string] BSTR bstrNodeType,
5 [out] BSTR *strEditorGUID,
[out] BSTR *strDelimiter
);

HRESULT RenameNode(
10 [in, string] BSTR bstrFullOldnodeName,
[in, string] BSTR bstrOldnodeName,
[in, string] BSTR bstrNewnodeName
);
HRESULT AddChild (
15 [in, string] BSTR bstrFullnodeName,
[in, string] BSTR bstrType,
[in, string] BSTR bstrDelimiter
);

20 HRESULT DeleteChild (
[in, string] BSTR bstrFullnodeName
);

HRESULT SelectionChange (
25 [in, string] BSTR bstrFullnodeName,

[in, string] BSTR bstrNodeType
);

30 HRESULT GetCfgSetFileName (
[out] BSTR * pFileName
);

35 HRESULT SetCfgSetFileName (
[in] BSTR bstrFileName
);

40 HRESULT EnumCfgSets (
[out] SAFEARRAY(BSTR) * pConfigSets
);

HRESULT SwitchToCfgSet (
[in] BSTR bstrConfigSet
);
45 HRESULT SaveCfgSetAs (
[in] BSTR bstrConfigSet
);

50 HRESULT ClearConfig (
);
HRESULT DeleteCfgSet (
[in] BSTR bstrConfigSet
);
```

}

### Operations

#### AddChild

5 Declaration

```
HRESULT AddChild (
    [in, string] BSTR bstrFullNodeName,
    [in, string] BSTR bstrType,
    [in, string] BSTR bstrDelimiter
10    )
```

#### Parameters

<i>bstrFullNodeName</i>	Specifies fully qualified Node name
<i>BstrType</i>	Specifies node type
<i>bstrDelimiter</i>	Specifies the new Node Delimiter

#### Return Value

<b>S_OK</b>	Success
<b>E_FAIL</b>	Error

#### ClearConfig

Declaration

```
15 HRESULT ClearConfig (
    )
```

#### Return Value

<b>S_OK</b>	Success
<b>E_FAIL</b>	Error

#### DeleteCfgSet

Declaration

```
20 HRESULT DeleteCfgSet (
    [in]          BSTR      bstrConfigSet
    )
```

#### Parameters

<i>bstrConfigSet</i>	Specify configuration set name
----------------------	--------------------------------

#### Return Value

<b>S_OK</b>	Success
<b>E_FAIL</b>	Error

25 **DeleteChild**

Declaration

```
HRESULT DeleteChild (
    [in, string] BSTR bstrFullNodeName
    )
```

*Parameters*

*bstrFullnodeName*      Specifies fully qualified node name

*Return Value*

**S\_OK**      Success

**E\_FAIL**      Error

**EnumCfgSets***Declaration*

```
5   HRESULT EnumCfgSets (
      [out] SAFEARRAY(BSTR) *      pConfigSets
      )
```

*Parameters*

*pConfigSets*      returns list wit configuration set (only names, no path)

**GetCfgSetFileName***Declaration*

```
10   HRESULT GetCfgSetFileName (
      [out]                    BSTR *      pFileName
      )
```

*Parameters*

*pFileName*      full name

**GetChildren***Declaration*

```
15   HRESULT GetChildren (
      [in, string] BSTR bstrFullnodeName,
      [in, string] BSTR bstrNodeDelimiter,
      20    [out] SAFEARRAY(BSTR) *bstrArrayChildrenNames,
      [out] SAFEARRAY(BSTR) *bstrArrayChildrenTypes,
      [out] SAFEARRAY(BSTR) *bstrArrayChildrenDelimiters
      )
```

*Parameters*

*bstrFullnodeName*      Specifies fully qualified node name

*bstrNodeDelimiter*      Specifies fully qualified node name

*bstrArrayChildrenNames*      Array with the names of the all children

*bstrArrayChildrenTypes*      Array with the types for each children

*bstrArrayChildrenDelimiters*      Array with the delimiters for each children

*Return Value*

**S\_OK**      Success

**E\_FAIL**      Error

**GetNodeProperties**

*Declaration*

```
5   HRESULT GetNodeProperties (
        [in, string] BSTR bstrNodeType,
        [out] BSTR *strEditorGUID,
        [out] BSTR *strDelimiter
    )
```

*Parameters*

<i>bstrNodeType</i>	Specifies node type
<i>StrEditorGUID</i>	Returns the Editor GUID for the specified Node type
<i>strDelimiter</i>	Returns Node delimiter

*Return Value*

<b>S_OK</b>	Success
<b>E_FAIL</b>	Error

**GetPossibleChildTypes***Declaration*

```
10  HRESULT GetPossibleChildTypes (
        [in, string] BSTR bstrFullnodeName,
        [out] SAFEARRAY(BSTR) *bstrChildTypes
    )
```

*Parameters*

<i>bstrFullnodeName</i>	Specifies fully qualified node name
<i>bstrChildTypes</i>	Returns a list with child types

**Initialize***Declaration*

```
20  HRESULT Initialize(
        [in, string] BSTR bstrServerGUID,
        [in] IUnknown* pCfgRules
    )
```

*Parameters*

<i>BstrServerGUID</i>	Specifies Server Name (ProgID of the Server)
<i>PCfgRules</i>	Interface to the IOAgent on the node where the Server resides

*Return Value*

<b>S_OK</b>	Success
<b>E_FAIL</b>	Error

**RenameNode***Declaration*

```
25  HRESULT RenameNode(
        [in, string] BSTR bstrFullOldnodeName,
```

```
[in, string] BSTR bstrOldNodeName,
[in, string] BSTR bstrNewNodeName
)
```

*Parameters*

<i>bstrFullOldNodeName</i>	Specifies the Full Old Node a Name
<i>bstrOldNodeName</i>	Specifies the Full New name for a Node
<i>bstrNewNodeName</i>	Specifies the Full New name for a Node

5    *Return Value*

<b>S_OK</b>	Success
<b>E_FAIL</b>	Error

**SaveCfgSetAs***Declaration*

```
10    HRESULT SaveCfgSetAs (
    [in]                    BSTR                bstrConfigSet
    )
```

*Parameters*

<i>bstrConfigSet</i>	Specify configuration set name
----------------------	--------------------------------

*Return Value*

<b>S_OK</b>	Success
<b>E_FAIL</b>	Error

**SelectionChange***Declaration*

```
15    HRESULT SelectionChange (
    [in, string] BSTR bstrFullNodeName,
    [in, string] BSTR bstrNodeType
    )
```

20    *Parameters*

<i>BstrFullNodeName</i>	Specifies fully qualified node name
<i>BstrNodeType</i>	Specifies Node type

*Return Value*

<b>S_OK</b>	Success
<b>E_FAIL</b>	Error

**SetCfgSetName***Declaration*

```
25    HRESULT SetCfgSetName (
    [in]                    BSTR                bstrFileName
    )
```

**SwitchToCfgSet***Declaration*

```
HRESULT SwitchToCfgSet (
    [in]             BSTR      bstrConfigSet
5 )
```

5

An IDASPackageSite interface 420 supports a connection between Node Editors 240 and the values for their parameters. This interface does not communicate directly with the storage area. In the illustrative embodiment there is one more level of indirection between the storage area and the PackageHost 210 (that implements IDASPackageSite). This level of indirection is provided via a DASConfigAccess.dll that implements IIOSrvCfgPersist and IioSrvCfgRules interfaces in the Remote Infrastructure 250.

***Declaration***

```

[object,
10  uuid(320C0A39-AB7C-11d4-93E4-00B0D0201D61),
    pointer_default(unique)
]
interface IDASPackageSite : IUnknown
{
15    HRESULT GetAttribute
    (
        [in] DWORD attributeHandle,
        [in] DWORD PropID,
        [out, retval] VARIANT *pVarValue
20
    );
    HRESULT SetAttribute
    (
25        [in] DWORD attributeHandle,
        [in] DWORD PropID,
        [in] VARIANT varValue,
        [out, retval] VARIANT_BOOL* pbSuccessFlag
30
    );
    HRESULT GetAttrHandle
    (
35        [in, string] BSTR attParent,
        [in, string] BSTR attName,
        [out, retval] DWORD* pAttributeHandle
    );
40    HRESULT CloseAttrHandle
    (
        [in] DWORD attributeHandle
    );
45    HRESULT GetPackageShapeInfo
    (
        [in, string] BSTR bstrFullNodeName,

```

```

    [in] SHORT nSelection,
    [out] SAFEARRAY(BSTR)* bstrAttrNames

    );
5
    HRESULT Commit
    (
        BSTR bstrFullnodeName,
        BSTR bstrType,
10       BSTR bstrDelimiter,
        SHORT *CommitStatus
    );
15
}
15 Description
{vb}

```

**Operations****CloseAttrHandle**20 **Declaration**

```

HRESULT CloseAttrHandle
(
    [in] DWORD attributeHandle
25
)

```

**Parameters**

*attributeHandle* Handle of the attribute, retrieved from Shape Data Collection.

**Return Value**

**S\_OK** ShapeInfo was successfully retrieved.

**Commit****Declaration**

```

30   HRESULT Commit
    (
        BSTR bstrFullnodeName,
        BSTR bstrType,
        BSTR bstrDelimiter,
35       SHORT *CommitStatus
    )

```

**Parameters**

*bstrFullnodeName* Full Node name

*bstrType* Node type

*bstrDelimiter* Node Delimiters

*CommitStatus* Indicates the status of the Commit. Following values are defined

COMMIT\_OK - Changes are saved  
 COMMIT\_INVALID\_HANDLE -  
 invalid handle specified  
 COMMIT\_FAIL - The changes are not saved due to  
 problems with the overall integrity of the node attributes  
 COMMIT\_REMOTE\_FAIL - Remote method call has failed

*Return Value*

<b>S_OK</b>	new values were successfully set
<b>E_FAIL</b>	Fail to save the change

**GetAttrHandle**

*Declaration*

```
5   HRESULT GetAttrHandle
    (
        [in, string] BSTR  attParent,
        [in, string] BSTR  attName,
        [out, retval] DWORD* pAttributeHandle
10  )
```

*Parameters*

<i>attParent</i>	The parent of the attribute.
<i>attName</i>	The parent of the attribute.
<i>pAttributeHandle</i>	The returned handle of the attribute. Can be used in Get/Set Attribute calls.

*Return Value*

<b>S_OK</b>	Successfully set the value of the property.
-------------	---

*Remarks*

15 The GetAttrHandle operation gets the attribute handle that can then be used for Get/Set Attribute calls. The GetAttrHandle operation does not resolve array or property references. If these are needed, the client should update the appropriate MxAttributeHandle fields directly before issuing a Get or Set call.

**GetAttribute**

*Declaration*

```
20  HRESULT GetAttribute
    (
        [in] DWORD attributeHandle,
        [in] DWORD PropID,
        [out, retval] VARIANT *pVarValue
25  )
```

*Parameters*

<i>attributeHandle</i>	Handle of the attribute, retrieved from Shape Data Collection.
<i>PropID</i>	Requested Property ID

*pVarValue* Receives the Value of the attribute/property.

*Return Value*

**S\_OK** Successfully retrieved the value.

*Remarks*

Used to get the attribute values of a primitives.

**GetPackageShapeInfo**

5 *Declaration*

```
HRESULT GetPackageShapeInfo
(
    [in, string] BSTR bstrFullnodeName,
    [in] SHORT nSelection,
10   [out] SAFEARRAY(BSTR)* bstrAttrNames

)
```

*Parameters*

*BstrFullnodeName* Full Node Name

*nSelection* Specifies the type of array to return For all allowed values please refer to enumerator SHAPE\_INFO\_ENUM

*BstrAttrNames* BSTR Safe array pointer. This array will be filled with requested shape information for specifique node

*Return Value*

**S\_OK** ShapeInfo was successfully retrieved.

15 **SetAttribute**

*Declaration*

```
HRESULT SetAttribute
(
    [in] DWORD attributeHandle,
    [in] DWORD PropID,
    [in] VARIANT varValue,

    [out, retval] VARIANT_BOOL* pbSuccessFlag
)
```

25 *Parameters*

*AttributeHandle* Handle of the attribute, retrieved from Shape Data Collection.

*PropID* Requested Property ID

*VarValue* Value of the property.

*pbSuccessFlag* Signifies whether set was accepted.

*Return Value*

**S\_OK** Successfully set the value of the property.

*Remarks*

Used to set the attribute values of a primitive.

Referring to **FIG. 6**, a set of interfaces are depicted that are implemented in a CIOSrvCfgpersist class 500. An IIOSrvCfgpersist interface 510 defines a set of methods for reading a current configuration for editing or saving a new configuration in the configuration 280.

##### 5 Declaration

```

[
    object,
    dual,
    uuid(52088D9A-DDE4-11D3-83F2-00A024A866AC),
10   helpstring("IIOSrvCfgPersist Interface"),
    pointer_default(unique)
]

/*
    IIOSrvCfgPersist provides a way to persist the current configuration into
15  the storage area and retrieve this information later
*/
interface IIOSrvCfgPersist : IDispatch
{
    HRESULT GetCfgFileName (
        [in]             BSTR      ServerClsIdString,
20      [out]            BSTR *    pFileName
    );
    HRESULT SetCfgFileName (
        [in]             BSTR      ServerClsIdString,
25      [in]             BSTR      FileName
    );

    HRESULT EnumCfgSets (
        [in]             BSTR      ServerClsIdString,
30      [out]  SAFEARRAY(BSTR) *  pConfigSets
    );

    HRESULT SwitchToCfgSet (
        [in]             BSTR      ServerClsIdString,
35      [in]             BSTR      pConfigSet
    );

    HRESULT SaveCfgSetAs (
        [in]             BSTR      ServerClsIdString,
40      [in]             BSTR      pConfigSet
    );

    HRESULT ClearCfgFile (
        [in]             BSTR      ServerClsIdString
45      );
}

HRESULT DeleteCfgFile (

```

```

[in]          BSTR      bstrServerClIdString,
[in]          BSTR      bstrConfigurationSet
);

5   HRESULT SetHierarchyObject(
    [in]          BSTR      ServerClIdString,
    [in]          BSTR      Name,
    [in]          BSTR      Type,
    [in]          BSTR      Delimiter,
10  [in, unique] SAFEARRAY(BSTR) * pProperties
);
    HRESULT GetHierarchyObject (
        [in]          BSTR      ServerClIdString,
        [in]          BSTR      Name,
15  [out]         BSTR * pType,
        [out]         BSTR * pDelimiter,
        [out]         SAFEARRAY(BSTR) * pProperties
    );
    HRESULT RemoveHierarchyObject(
20  [in]          BSTR      ServerClIdString,
        [in]          BSTR      Name
    );
    HRESULT RenameHierarchyObject(
        [in]          BSTR      ServerClIdString,
        [in]          BSTR      OldName,
        [in]          BSTR      NewName);
25
    HRESULT SetTopicObject(
        [in]          BSTR      ServerClIdString,
        [in]          BSTR      Name,
        [in]          BSTR      Parent,
        [in]          BSTR      Type,
        [in]          LONG     UpdateInterval,
        [in, unique] SAFEARRAY(BSTR) * pProperties
    );
30
    HRESULT GetTopicObject(
        [in]          BSTR      ServerClIdString,
        [in]          BSTR      Name,
        [in]          BSTR      Parent,
        [out]         BSTR * pType,
        [out]         LONG * pUpdateInterval,
        [out]         SAFEARRAY(BSTR) * pProperties
    );
35
    HRESULT RemoveTopicObject(
        [in]          BSTR      ServerClIdString,
        [in]          BSTR      Name,
        [in]          BSTR      Parent);
40
    HRESULT RenameTopicObject(
        [in]          BSTR      ServerClIdString,
        [in]          BSTR      OldName,
        [in]          BSTR      Parent,
        [in]          BSTR      NewName);
45
    HRESULT SetLeafObject(
        [in]          BSTR      ServerClIdString,
        [in]          BSTR      Name,

```

```

5      [in]          BSTR          Parent,
      [in]          BSTR          Type,
      [in]          BSTR          Delimiter,
      [in, unique] SAFEARRAY(BSTR) * pProperties
    );
10     HRESULT GetLeafObject(
      [in]          BSTR          ServerClsIdString,
      [in]          BSTR          Name,
      [in]          BSTR          Parent,
      [out]         BSTR  *       pType,
      [out]         BSTR  *       pDelimiter,
      [out]         SAFEARRAY(BSTR) * pProperties
    );
15     HRESULT RemoveLeafObject(
      [in]          BSTR          ServerClsIdString,
      [in]          BSTR          Name,
      [in]          BSTR          Parent);
20     HRESULT RenameLeafObject(
      [in]          BSTR          ServerClsIdString,
      [in]          BSTR          OldName,
      [in]          BSTR          Parent,
      [in]          BSTR          NewName);
25     HRESULT BrowseHierarchyObjects(
      [in]          BSTR          ServerClsIdString,
      [in]          BSTR          BranchName,
      [out]         SAFEARRAY(BSTR) * pBranches,
      [out]         SAFEARRAY(BSTR) * pLeaves,
      [out]         SAFEARRAY(BSTR) * pTopics
    );
30     HRESULT SetDataObject (
      [in]          BSTR          ServerClsIdString,
      [in]          BSTR          Name,
      [in]          BSTR          Type,
      [in, unique] SAFEARRAY(BSTR) * pProperties
    );
35     HRESULT GetDataObject (
      [in]          BSTR          ServerClsIdString,
      [in]          BSTR          Name,
      [out]         BSTR  *       pType,
      [out]         SAFEARRAY(BSTR) * pProperties
    );
40     HRESULT RemoveDataObject(
      [in]          BSTR          ServerClsIdString,
      [in]          BSTR          Name);
45     HRESULT RenameDataObject(
      [in]          BSTR          ServerClsIdString,
      [in]          BSTR          OldName,
      [in]          BSTR          NewName);
50     HRESULT BrowseDataObjects(
      [in]          BSTR          ServerClsIdString,
      [out]         SAFEARRAY(BSTR) * pDataObjects
    );
      );
      );

```

```

        [out] SAFEARRAY(BSTR) *      pProperties
    );
5   HRESULT SetSystemObject(
        [in]          BSTR           ServerClIdString,
        [in, unique] SAFEARRAY(BSTR) *  pProperties
    );
   HRESULT RemoveSystemObject(
        [in]          BSTR           ServerClIdString);
}
10

```

**Operations****BrowseDataObjects***Declaration*

```

15  HRESULT BrowseDataObjects(
        [in]          BSTR           ServerClIdString,
        [out]         SAFEARRAY(BSTR) *  pDataObjects
    )

```

**BrowseHierarchyObjects***Declaration*

```

20  HRESULT BrowseHierarchyObjects(
        [in]          BSTR           ServerClIdString,
        [in]          BSTR           BranchName,
        [out]         SAFEARRAY(BSTR) *  pBranches,
        [out]         SAFEARRAY(BSTR) *  pLeaves,
25    [out]         SAFEARRAY(BSTR) *  pTopics
    )

```

**ClearCfgFile***Declaration*

```

30  HRESULT ClearCfgFile (
        [in]          BSTR           ServerClIdString
    )

```

*Return Value*

```

S_OK   The operation succeeded.
E_FAIL The operation failed.
35  E_OUTOFMEMORY Not enough memory
E_INVALIDARG An argument to the function was invalid.

```

**DeleteCfgFile***Declaration*

```

40  HRESULT DeleteCfgFile (
        [in]          BSTR           bstrServerClIdString,
        [in]          BSTR           bstrConfigurationSet
    )

```

*Return Value*

```

S_OK   The operation succeeded.
45  E_FAIL The operation failed.

```

E\_OUTOFMEMORY Not enough memory  
 E\_INVALIDARG An argument to the function was invalid.

**EnumCfgSets***Declaration*

5    HRESULT EnumCfgSets (

[in]	BSTR	<i>ServerClIdString,</i>
[out]	SAFEARRAY(BSTR) *	<i>pConfigSets</i>
)		

10    *Parameters*

*pConfigSets*                 returns list wit configuration set (only names, no path)

**GetCfgFileName***Declaration*

HRESULT GetCfgFileName (

[in]	BSTR	<i>ServerClIdString,</i>
[out]	BSTR *	<i>pFileName</i>
)		

*Return Value*

S\_OK    The operation succeeded.  
 E\_FAIL The operation failed.  
 20    E\_OUTOFMEMORY Not enough memory  
 E\_INVALIDARG An argument to the function was invalid.

**GetDataObject***Declaration*

HRESULT GetDataObject (

[in]	BSTR	<i>ServerClIdString,</i>
[in]	BSTR	<i>Name,</i>
[out]	BSTR *	<i>pType,</i>
[out]	SAFEARRAY(BSTR) *	<i>pProperties</i>
)		

30    **GetHierarchyObject***Declaration*

HRESULT GetHierarchyObject (

[in]	BSTR	<i>ServerClIdString,</i>
[in]	BSTR	<i>Name,</i>
[out]	BSTR *	<i>pType,</i>
[out]	BSTR *	<i>pDelimiter,</i>
[out]	SAFEARRAY(BSTR) *	<i>pProperties</i>
)		

*Return Value*

40    S\_OK    The operation succeeded.  
 E\_FAIL The operation failed.  
 E\_OUTOFMEMORY Not enough memory  
 E\_INVALIDARG An argument to the function was invalid.

**GetLeafObject***Declaration*

```

HRESULT GetLeafObject(
    [in]          BSTR      ServerClIdString,
5     [in]          BSTR      Name,
    [in]          BSTR      Parent,
    [out]         BSTR *   pType,
    [out]         BSTR *   pDelimiter,
    [out]  SAFEARRAY(BSTR) *   pProperties
10    )

```

*Return Value*

```

S_OK   The operation succeeded.
E_FAIL The operation failed.
E_OUTOFMEMORY Not enough memory
15   E_INVALIDARG An argument to the function was invalid.

```

**GetSystemObject***Declaration*

```

HRESULT GetSystemObject (
    [in]          BSTR      ServerClIdString,
20    [out]  SAFEARRAY(BSTR) *   pProperties
)

```

**GetTopicObject***Declaration*

```

HRESULT GetTopicObject(
25    [in]          BSTR      ServerClIdString,
    [in]          BSTR      Name,
    [in]          BSTR      Parent,
    [out]         BSTR *   pType,
    [out]         LONG *   pUpdateInterval,
30    [out]  SAFEARRAY(BSTR) *   pProperties
)

```

*Return Value*

```

S_OK   The operation succeeded.
E_FAIL The operation failed.
35   E_OUTOFMEMORY Not enough memory
E_INVALIDARG An argument to the function was invalid.

```

**RemoveDataObject***Declaration*

```

HRESULT RemoveDataObject(
40    [in]          BSTR      ServerClIdString,
    [in]          BSTR      Name)

```

*Return Value*

```

S_OK   The operation succeeded.
E_FAIL The operation failed.
45   E_OUTOFMEMORY Not enough memory

```

E\_INVALIDARG An argument to the function was invalid.

### RemoveHierarchyObject

#### Declaration

```
HRESULT RemoveHierarchyObject(
    [in]             BSTR      ServerClIdString,
    [in]             BSTR      Name
)
```

#### Return Value

S\_OK The operation succeeded.

E\_FAIL The operation failed.

E\_OUTOFMEMORY Not enough memory

E\_INVALIDARG An argument to the function was invalid.

### RemoveLeafObject

#### Declaration

```
HRESULT RemoveLeafObject(
    [in]             BSTR      ServerClIdString,
    [in]             BSTR      Name,
    [in]             BSTR      Parent)
```

#### Return Value

S\_OK The operation succeeded.

E\_FAIL The operation failed.

E\_OUTOFMEMORY Not enough memory

E\_INVALIDARG An argument to the function was invalid.

### RemoveSystemObject

#### Declaration

```
HRESULT RemoveSystemObject(
    [in]             BSTR      ServerClIdString)
```

#### Return Value

S\_OK The operation succeeded.

E\_FAIL The operation failed.

E\_OUTOFMEMORY Not enough memory

E\_INVALIDARG An argument to the function was invalid.

### RemoveTopicObject

#### Declaration

```
HRESULT RemoveTopicObject(
    [in]             BSTR      ServerClIdString,
    [in]             BSTR      Name,
    [in]             BSTR      Parent)
```

#### Return Value

S\_OK The operation succeeded.

E\_FAIL The operation failed.

E\_OUTOFMEMORY Not enough memory

E\_INVALIDARG An argument to the function was invalid.

**RenameDataObject***Declaration*

```
HRESULT RenameDataObject(
    [in]             BSTR      ServerClIdString,
5     [in]             BSTR      OldName,
    [in]             BSTR      NewName)
```

*Return Value*

```
S_OK   The operation succeeded.  

E_FAIL The operation failed.  

10    E_OUTOFMEMORY Not enough memory  

E_INVALIDARG An argument to the function was invalid.
```

**RenameHierarchyObject***Declaration*

```
HRESULT RenameHierarchyObject(
15    [in]             BSTR      ServerClIdString,
    [in]             BSTR      OldName,
    [in]             BSTR      NewName)
```

*Return Value*

```
S_OK   The operation succeeded.  

20    E_FAIL The operation failed.  

E_OUTOFMEMORY Not enough memory  

E_INVALIDARG An argument to the function was invalid.
```

**RenameLeafObject***Declaration*

```
25    HRESULT RenameLeafObject(
        [in]             BSTR      ServerClIdString,
        [in]             BSTR      OldName,
        [in]             BSTR      Parent,
        [in]             BSTR      NewName)
```

*Return Value*

```
S_OK   The operation succeeded.  

E_FAIL The operation failed.  

E_OUTOFMEMORY Not enough memory  

E_INVALIDARG An argument to the function was invalid.
```

**RenameTopicObject***Declaration*

```
40    HRESULT RenameTopicObject(
        [in]             BSTR      ServerClIdString,
        [in]             BSTR      OldName,
        [in]             BSTR      Parent,
        [in]             BSTR      NewName)
```

*Return Value*

```
S_OK   The operation succeeded.  

E_FAIL The operation failed.
```

**E\_OUTOFMEMORY** Not enough memory  
**E\_INVALIDARG** An argument to the function was invalid.

**SaveCfgSetAs***Declaration*

```
5  HRESULT SaveCfgSetAs (
    [in]             BSTR      ServerClsIdString,
    [in]             BSTR      pConfigSet
)
```

**SetCfgFileName***Declaration*

```
10 HRESULT SetCfgFileName (
    [in]             BSTR      ServerClsIdString,
    [in]             BSTR      FileName
)
```

*Return Value*

**S\_OK** The operation succeeded.  
**E\_FAIL** The operation failed.  
**E\_OUTOFMEMORY** Not enough memory  
**E\_INVALIDARG** An argument to the function was invalid.

**SetDataObject***Declaration*

```
15 HRESULT SetDataObject (
    [in]             BSTR      ServerClsIdString,
    [in]             BSTR      Name,
    [in]             BSTR      Type,
    25   [in, unique] SAFEARRAY(BSTR) * pProperties
)
```

**SetHierarchyObject***Declaration*

```
30  HRESULT SetHierarchyObject (
    [in]             BSTR      ServerClsIdString,
    [in]             BSTR      Name,
    [in]             BSTR      Type,
    [in]             BSTR      Delimiter,
    35   [in, unique] SAFEARRAY(BSTR) * pProperties
)
```

*Return Value*

**S\_OK** The operation succeeded.  
**E\_FAIL** The operation failed.  
**E\_OUTOFMEMORY** Not enough memory  
**E\_INVALIDARG** An argument to the function was invalid.

**SetLeafObject***Declaration*

```
40  HRESULT SetLeafObject (
```

```

5      [in]          BSTR           ServerClIdString,
      [in]          BSTR           Name,
      [in]          BSTR           Parent,
      [in]          BSTR           Type,
      [in]          BSTR           Delimiter,
      [in, unique] SAFEARRAY(BSTR) * pProperties
)

```

*Return Value*

```

10     S_OK   The operation succeeded.
      E_FAIL The operation failed.
      E_OUTOFMEMORY Not enough memory
      E_INVALIDARG An argument to the function was invalid.

```

**SetSystemObject**

*Declaration*

```

15    HRESULT SetSystemObject(
        [in]          BSTR           ServerClIdString,
        [in, unique] SAFEARRAY(BSTR) * pProperties
)

```

**SetTopicObject**

*Declaration*

```

HRESULT SetTopicObject(
    [in]          BSTR           ServerClIdString,
    [in]          BSTR           Name,
    [in]          BSTR           Parent,
25    [in]          BSTR           Type,
    [in]          LONG           UpdateInterval,
    [in, unique] SAFEARRAY(BSTR) * pProperties
)

```

*Parameters*

<i>Name</i>	name of the topic
<i>Parent</i>	fully qualified name

*Return Value*

```

S_OK   The operation succeeded.
E_FAIL The operation failed.
E_OUTOFMEMORY Not enough memory
E_INVALIDARG An argument to the function was invalid.

```

**SwitchToCfgSet**

*Declaration*

```

HRESULT SwitchToCfgSet (
    [in]          BSTR           ServerClIdString,
    [in]          BSTR           pConfigSet
40
)

```

An IIOSrvCfgrules interface 520 defines a set of methods for extracting configuration rules from a storage area such as the rules 270 via the remote infrastructure 250. This interface insulates the caller from the particulars of extracting or storing data on a particular physical device in a particular location and form.

5

*Declaration*

```

[

    object,
    dual,
10     uuid(52088D9A-DED4-11D3-83F2-00A024A866AD) ,

    helpstring("IIOSrvCfgRules Interface"),
    pointer_default(unique)
]

15 /*

This interface provides a way to extract the configuration rules from the rules storage area
*/
    interface IIoSrvCfgRules : IDispatch
    {
20     HRESULT GetCfgRuleFileName(
            [in]      BSTR      ServerClsIdString,
            [out]     BSTR *     pFileName
        );
25     HRESULT SetCfgRuleFileName(
            [in]      BSTR      ServerClsIdString,
            [in]      BSTR      pFileName
        );
30     HRESULT GetHostClsIds(
            [in]      BSTR      ServerClsIdString,
            [out]     BSTR *     pPackageClsIdString,
            [out]     BSTR *     ,pEditorHostClsIdString
        );
35     HRESULT GetRulesForObjectType(
            [in]      BSTR      ServerClsIdString,
            [in]      BSTR      ObjectType,
40            [out]     BSTR *     pEditorClsIdString,
            [out]     BSTR *     pPackageClsIdString,
            [out]     BSTR *     pDelimiter
        );
45     HRESULT GetCfgChildRulesForObjectType(
            [in]      BSTR      ServerClsIdString,
            [in]          BSTR      Type,
            [out]     SAFEARRAY(BSTR) * ChildTypes,

```

```

        [out]     SAFEARRAY(LONG) *    ChildTypesMaxOccurs,
        [out]     LONG *           pOptions
    ) ;

5      HRESULT GetCfgPropertyRulesForObjectType(
        [in]      BSTR      ServerClsIdString,
        [in]      BSTR      Type,
        [out]     SAFEARRAY(BSTR) *    PropertyName,
        [out]     SAFEARRAY(LONG) *    PropertyType,
        [out]     SAFEARRAY(VARIANT) *  DefaultValue,
        [out]     SAFEARRAY(VARIANT) *  PropertyMin,
        [out]     SAFEARRAY(VARIANT) *  PropertyMax,
        [out]     SAFEARRAY(BSTR) *    PropertyEditHeader,
10     [out]     SAFEARRAY(BSTR) *    PropertyEditUnit,
        [out]     SAFEARRAY(BSTR) *    PropertyEditHelp,
        [out]     SAFEARRAY(LONG) *    PropertyAccessRights
    ) ;
15
20
}

```

**Operations****GetCfgChildRulesForObjectType***Declaration*

```

25     HRESULT GetCfgChildRulesForObjectType(
        [in]      BSTR      ServerClsIdString,
        [in]      BSTR      Type,
        [out]     SAFEARRAY(BSTR) *    ChildTypes,
        [out]     SAFEARRAY(LONG) *    ChildTypesMaxOccurs,
30     [out]     LONG *           pOptions
    )

```

*Parameters*

<i>ServerClsIdString</i>	server clsid.
<i>Type</i>	type name - if rules for the root are to be retrieved pass empty string:
<i>ChildTypes</i>	Get the array of possible child types. The configuration snap-in displays these types as choices to the user to create child objects
<i>ChildTypesMaxOccurs</i>	Number of maximum occurrences of the corresponding child object type.
<i>pOptions</i>	List of bit-options in returned DWORD currently 3 bits are defined:

*Remarks*

35 Gets the child rules for object type. Rules are user type based. This means that objects with the same user type have the same set of rules.

**GetCfgPropertyRulesForObject***Declaration*

```

HRESULT GetCfgPropertyRulesForObject(
5          [in]     BSTR      ServerClIdString,
          [in]     BSTR      Type,
          [out]    SAFEARRAY(BSTR) *      PropertyName,
          [out]    SAFEARRAY(LONG) *      PropertyType,
          [out]    SAFEARRAY(VARIANT) *   DefaultValue,
          [out]    SAFEARRAY(VARIANT) *   PropertyMin,
          [out]    SAFEARRAY(VARIANT) *   PropertyMax,
          [out]    SAFEARRAY(BSTR) *      PropertyEditHeader,
          [out]    SAFEARRAY(BSTR) *      PropertyEditUnit,
          [out]    SAFEARRAY(BSTR) *      PropertyEditHelp,
10         [out]    SAFEARRAY(LONG) *      PropertyAccessRights
15         )

```

*Parameters*

<i>ServerClIdString</i>	server clsid.
<i>Type</i>	type name - if rules for the root are to be retrieved pass empty string:
<i>PropertyName</i>	Array of unique property names. The configuration snap-in will display an edit field for each property.
<i>PropertyType</i>	Array of property types. The configuration snap-in uses this type to perform validation on user input.
<i>DefaultValue</i>	Array of property default values. The configuration snap-in uses this value to initialize user input values. Any configuration data client uses this as the default value if not present. In the case of VARIANT_EMPTY there is no default value.
<i>PropertyMin</i>	Array of minimum property values. The configuration snap-in uses this value to validate user input values. In case of type 'STRING' this represents the minimum number of (UNICODE) characters.
<i>PropertyMax</i>	Array of maximum property values. The configuration snap-in uses this value to validate user input values. In case of type 'STRING' this represents the maximum number of (UNICODE) characters.
<i>PropertyEditHeader</i>	Array of property edit headers. The configuration snap-in uses this value to precede the edit field for user input.
<i>PropertyEditUnit</i>	Array of property edit units. The configuration snap-in uses this string to be displayed after the edit field for user input.

*PropertyEditHelp*

Array of property edit help strings. The configuration snap-in uses this string to display an online help string on user request for the corresponding edit field.

*PropertyAccessRights*

Array of property edit lock flags. The configuration snap-in uses this value to determine the accessrights on this property.

*Remarks*

Gets the property rules for object type. Rules are user type based. This means that objects with the same user type have the same set of rules.

**GetCfgRuleFileName**5 *Declaration*

```
HRESULT GetCfgRuleFileName(
```

[in] BSTR	<i>ServerClsIdString</i> ,
[out] BSTR *	<i>pFileName</i>

10 )

*Parameters*

<i>ServerClsIdString</i>	DAS server class ID
<i>pFileName</i>	pointer to file name

*Remarks*

retrieves the cfg rule file name of the active configuration file of the server. The configuration snap-in usually does not need to know this file name. This is only for informational purposes

15 **GetHostClsIds***Declaration*

```
HRESULT GetHostClsIds(
```

[in] BSTR	<i>ServerClsIdString</i> ,
[out] BSTR *	<i>pPackageClsIdString</i> ,
[out] BSTR *	<i>pEditorHostClsIdString</i>

)

*Parameters*

<i>ServerClsIdString</i>	DAS server class ID
<i>PPackageClsIdString</i>	Clsid of the package
<i>PEditorHostClsIdString</i>	Clsid of the editor host

*Remarks*

25 retrieves the editor ClsId for a given object type

**GetRulesForObject***Declaration*

```
HRESULT GetRulesForObject(
```

```

5      [in]    BSTR     ServerClsIdString,
      [in]    BSTR     ObjectType,
      [out]   BSTR *   pEditorClsIdString,
      [out]   BSTR *   pPackageClsIdString,
      [out]   BSTR *   pDelimiter
)

```

*Parameters*

<i>ServerClsIdString</i>	DAS server class ID
<i>ObjectType</i>	object type (value of the type attribute of the corresponding object in the configuration file)
<i>pEditorClsIdString</i>	Clsid of the editor ActiveX responsible for editing all object properties
<i>pPackageClsIdString</i>	Clsid of the package
<i>pDelimiter</i>	Delimiter of the type (post delimiter)

*Remarks*

- 10 retrieves the editor ClsId for a given object type

**SetCfgRuleFileName***Declaration*

```

HRESULT SetCfgRuleFileName(
15          [in]    BSTR     ServerClsIdString,
          [in]    BSTR     pFileName
)

```

*Parameters*

<i>ServerClsIdString</i>	DAS server class ID
<i>pFileName</i>	file name

*Remarks*

- 20 Sets the rules file name

Turning to FIG. 7, a sequence diagram depicts a set of steps executed when a node editor opens and a particular data access server is selected by a user. During step 600, a user launches the DAS console client via an MMC. In response, the DASCC communicates with DAS Agents located on each node supporting a DAS to enumerate all DAS's on the network during step 602.

- 5 Next, during step 604 a user selects one of the enumerated DAS's for configuration, and during steps 606 and 608 the DASCC issues requests to create an instance of the Package Host 210 object and an instance of the Editor Host 230. If a node package or node editor is specified, then these objects too are instantiated.

After creating the above configuration components, at step 610 the DASCC issues a  
10 request to the Package Host 210 to request a set of node configuration parameters for a node containing the data access server selected by the user during step 604. At step 612 the Package Host 210 passes the read request to the remote infrastructure via the CIOSSrvCfgPersist interface  
510. The remote infrastructure calls the configuration storage 280 to read the existing parameters associated with the selected DAS during step 614, and calls the rules storage 2270 to read any  
15 applicable rules associated with the selected DAS during step 616. The retrieved configuration parameters and rules are then returned to the calling Package Host 210.

During step 618 the DAS control consol issues an "init" call on the previously created Editor Host 230. In response, during steps 620 and 622 the Editor Host 230 creates an instance of the Node Editor 240 custom face plate specified by the accessed node and then forwards the  
20 "init" call to the Node Editor 240. At step 624 the Node Editor 240 calls the Package Host 210 to obtain values for each parameter specified by the node (and any associated rules) and then displays the configuration parameters in accordance with the associated rules via the custom face plate during step 626. Thereafter, the user issues requests to the Node Editor 240 to change node parameters in association with a configuration session for the selected DAS.

25

Turning to FIG. 8, a sequence diagram depicts a series of exemplary steps performed to edit an existing configuration, and in particular add a new node in a configuration hierarchy specified for a selected data access server. During step 700 the user selects a parent node (type) to add to a configuration by passing an appropriate user interface request to the DASCC 200. In  
30 response, during step 702 the DASCC 200 issues a request to the Package Host 210 to get a list

of configuration parameters and rules associated with the selected node type (including any children nodes). At step 704 the Package Host 210 requests, from the CIOsRvCfGPerSist object 500, all rules for the selected node. Thereafter, the CIOsRvCfGPerSist object 500 passes the configuration rules request to the rules storage 270. Next, at step 708 the Package Host 210 issues a request to read the child node parameters for the selected node to the CIOsRvCfGPerSist object 500. In response at step 710 the request is passed to the configuration storage 280. Upon receiving both the appropriate rules and parameters, the Package Host applies the configuration rules to the retrieved parameters to create an appropriate parent node structure.

Thereafter, at step 714 the user issues an interface command to the DASCC 200 to add a new node of the previously selected type. Next, at step 716 the DASCC 200 issues a request to the Package Host 210 to add a new node to a selected/identified child node within a configuration hierarchy. At step 718 the Package Host 210 calls the CIOsRvCfGPerSist object 500 to add a new node to the storage area for currently selected configuration. Thereafter, at step 720 the CIOsRvCfGPerSist object 500 issues a request to the configuration storage 280 to store the created new node.

Turning to FIG. 9, a sequence diagram depicts the steps for saving changes made to a configuration using the previously described distributed configuration components. In response to a user issuing a save request by selecting an "apply" button on the user interface of the Editor Host 230 during step 800. In response, during step 802 the Editor Host 230 passes a call to save configuration changes to the Node Editor 240 (node type-specific faceplate). In response, during step 804 the Node Editor 240 calls the Package Host 210 with a request to save the new values of the node parameters. The Package Host 210, with reference possibly to the custom node package 220, validates the requested node change information during step 806. During step 808 the Package Host 210 calls the CIOsRvCfGPerSist object 500 requesting a save operation on a set of specified node parameters. The CIOsRvCfGPerSist object 500 carries out the save request at step 810 by storing the changed parameter values in the configuration storage 280. After the new configuration information is placed in the configuration storage 280, the CIOsRvCfGPerSist object 500 determines whether the affected DAS is currently running. In the case that the DAS is running, at step 812 the CIOsRvCfGPerSist object 500 notifies the corresponding DAS of the changes (including specifying the nodes that were changed). The notification mechanism

embodied in step 812 facilitates hot configuration of a running data access server. In response, the server reads the new values from the configuration storage 280 and updates its configuration at a point in time when such changes do not disrupt the ongoing operation of the DAS.

The following is a set of type definitions utilized in an exemplary implementation of the

5 Package Host 210 and the Node Editor 240.

#### Type Definitions:

### **ATTR\_PROPERTYIDS**

#### *Declaration*

```
10 enum
  {
    PROPERTY_INVALID_PROPID_ = 0,
    PROPERTY_NAME_,
    PROPERTY_TYPE_,
    PROPERTY_VALUE_,
    PROPERTY_MIN_,
    PROPERTY_MAX_,
    PROPERTY_HEADER_,
    PROPERTY_EUNITS_,
    PROPERTY_HELP_,
    PROPERTY_ACCESSRIGHTS_
  }
```

#### *Description*

Specifies the Attribute properties

#### *Elements*

<code>_PROPERTY_VALUE_</code> <code>_PROPERTY_HEADER_</code> <code>_PROPERTY_MAX_</code> <code>_PROPERTY_MIN_</code> <code>_PROPERTY_INVALID_PROPID_</code>	Zero is reserved for invalid PropID
<code>_PROPERTY_TYPE_</code> <code>_PROPERTY_HELP_</code> <code>_PROPERTY_ACCESSRIGHTS_</code> <code>_PROPERTY_EUNITS_</code> <code>_PROPERTY_NAME_</code>	

25

### **COMMIT\_ENUM**

#### *Declaration*

```
30 enum {
  COMMIT_OK =1,
  COMMIT_INVALID_HANDLE,
  COMMIT_FAIL,
  COMMIT_REMOTE_FAIL
```

}

**Description**

Specifies the options for status of Commit operation

**Elements**

- COMMIT\_INVALID\_HANDLE
- COMMIT\_REMOTE\_FAIL
- COMMIT\_OK
- COMMIT\_FAIL

5

## IOSRVPROPERTY

**Declaration**

```
typedef struct tagIOSRVPROPERTY {
    [string] BSTR szKey;
10      [string] BSTR szProperty; } IOSRVPROPERTY
```

**Description**

Describes a parameter's name and value.

## PROPERTYACCEESSRIGHT

15    **Declaration**

```
typedef enum tagPROPERTYACCEESSRIGHTS {
    PROPERTYACCEESSRIGHT_ALL = 0,
    PROPERTYACCEESSRIGHT_LOCKED =1 } PROPERTYACCEESSRIGHT
```

**Description**

20       Describes user access rights for a property

## SHAPE\_INFO\_ENUM

**Declaration**

```
enum {
25      SHAPE_INFO_ATTRIBUTES = 1,
      SHAPE_INFO_NODES,
      SHAPE_INFO_TOPICS,
      SHAPE_INFO_LEAVES
}
```

30    **Description**

Specifies the Shape information options

**Elements**

- SHAPE\_INFO\_ATTRIBU  
TES
- SHAPE\_INFO\_TOPICS
- SHAPE\_INFO\_LEAVE

**SHAPE\_INFO\_NODES**

The following is an example of the rule file 270 which makes up a first part of a remote configuration specification's persistence area. A set of tag explanations precedes the actual example.

KEY	XML Attributes	Explanation
<CONFIGURATIONRULES>	NAME PACKAGEHOSTID EDITORHOSTID	Root key for the configuration rules (DAS related data is inside this root key only).  The presence of this root key potentially allows additional (user-) data to be stored in the same file under parallel root keys.
<HIERARCHYNODE>	TYPE EDITORID PACKAGEID DELIMITER	Key for rule for an object of type "TYPE" (related to a DAS object type).  TYPE (server specific class) and NAME are required – DELIMITER is optional (defaults to ":" If not present)
<CHILDRULES>	ENABLEDEVICEGROUPS ENABLEDEVICEITEMS ENABLESYSTEMITEMS	Sub-key for CHILDRULES of an object of type "NAME".  DEVICEGROUPS, DEVICEITEMS, SYSTEMITEMS are keys for boolean flags allowing the specified objects on this branch object of type "NAME".
<"user type">	MAXOCCURENCES MINOCCURENCES	User type Key for an object under child rules.
<PROPERTYRULES>		Key for properties/ editor element
<PROPERTYTYPE>		Key for the variant property type under property rules.
<DEFAULTVALUE>		Key for the property default value under property rules..
<PROPERTYMIN>		Key for the property minimum value under property rules.
<PROPERTYMAX>		Key for the property maximum value under property rules..
<PROPERTYEDITHEADER>		Key for the property edit field header under property rules..
<PROPERTYUNIT>		Key for the property edit field unit under property rules..
<PROPERTYHELP>		Key for the property edit field help under property rules..

```

?xml version="1.0" encoding="UTF-8"?>
5 <CONFIGURATIONRULES NAME="DAS ABTCP" PACKAGEHOSTID="{7F5B8DEE-AB78-11D4-93E4-
00B0D0201D61}" EDITORHOSTID="{5C9AF1AA-AEC8-11D4-93E6-00B0D0201D61}">
<HIERARCHYNODE TYPE="$ROOT$">
6   <CHILDRULES ENABLEDEVICEGROUPS="1" ENABLEDEVICEITEMS="0"
7     ENABLESYSTEMITEMS="1">
8     <PORT_TCPIP MAXOCCURRENCES="1"/>
9     </CHILDRULES>
10    <PROPERTYRULES>
11      <UpdateInterval>
12        <PROPERTYTYPE>VT_I4</PROPERTYTYPE>
13        <DEFAULTVALUE>1000</DEFAULTVALUE>
14        <PROPERTYMIN>0</PROPERTYMIN>
15        <PROPERTYMAX>100000</PROPERTYMAX>
16        <PROPERTYEDITUNIT>ms</PROPERTYEDITUNIT>
17        <PROPERTYEDITHELP>Update interval of default topics for hierarchies without overriding
18        configuration</PROPERTYEDITHELP>
19        </UpdateInterval>
20        <SlowPollInterval>
21          <PROPERTYTYPE>VT_I4</PROPERTYTYPE>
22          <DEFAULTVALUE>10000</DEFAULTVALUE>
23          <PROPERTYMIN>0</PROPERTYMIN>
24          <PROPERTYMAX>100000</PROPERTYMAX>
25          <PROPERTYEDITUNIT>ms</PROPERTYEDITUNIT>
26          <PROPERTYEDITHELP>Update interval of topics for hierarchies in slow poll
mode</PROPERTYEDITHELP>
27        </SlowPollInterval>
28        <CaseSensitive>
29          <PROPERTYTYPE>VT_I4</PROPERTYTYPE>
30          <DEFAULTVALUE>0</DEFAULTVALUE>
31          <PROPERTYMIN>0</PROPERTYMIN>
32          <PROPERTYMAX>1</PROPERTYMAX>
33          <PROPERTYEDITUNIT>T/F</PROPERTYEDITUNIT>
34          <PROPERTYEDITHELP>TRUE means case-sensitive Device Group and Item ID
names</PROPERTYEDITHELP>
35        </CaseSensitive>
36        <DefaultPokeMode>
37          <PROPERTYTYPE>VT_I4</PROPERTYTYPE>
38          <DEFAULTVALUE>1</DEFAULTVALUE>
39          <PROPERTYMIN>0</PROPERTYMIN>
40          <PROPERTYMAX>2</PROPERTYMAX>
41          <PROPERTYEDITUNIT/>
42          <PROPERTYEDITHELP>0=Control, 1=Transition, 2=Optimized, </PROPERTYEDITHELP>
43        </DefaultPokeMode>
44        <DefaultDelimiter>
45          <PROPERTYTYPE>VT_BSTR</PROPERTYTYPE>
46          <DEFAULTVALUE>.</DEFAULTVALUE>
47          <PROPERTYMIN>1</PROPERTYMIN>
48          <PROPERTYMAX>3</PROPERTYMAX>
49          <PROPERTYEDITUNIT/>
50          <PROPERTYEDITHELP>Delimiter for hierarchies not configured</PROPERTYEDITHELP>

```

```

</DefaultDelimiter>
<SimulationMode>
  <PROPERTYTYPE>VT_I4</PROPERTYTYPE>
  <DEFAULTVALUE>0</DEFAULTVALUE>
  <PROPERTYMIN>0</PROPERTYMIN>
  <PROPERTYMAX>1</PROPERTYMAX>
  <PROPERTYEDITUNIT>T/F</PROPERTYEDITUNIT>
  <PROPERTYEDITHELP>TRUE means simulate for DA Servers that support
5   simulation</PROPERTYEDITHELP>
</SimulationMode>
<EnableSystemItems>
  <PROPERTYTYPE>VT_I4</PROPERTYTYPE>
  <DEFAULTVALUE>1</DEFAULTVALUE>
  <PROPERTYMIN>0</PROPERTYMIN>
10  <PROPERTYMAX>1</PROPERTYMAX>
  <PROPERTYEDITUNIT>T/F</PROPERTYEDITUNIT>
  <PROPERTYEDITHELP>TRUE to allow system items</PROPERTYEDITHELP>
</EnableSystemItems>
<LinkTopicCache>
  <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
  <DEFAULTVALUE>0</DEFAULTVALUE>
  <PROPERTYMIN>0</PROPERTYMIN>
  <PROPERTYMAX>1</PROPERTYMAX>
  <PROPERTYEDITUNIT>T/F</PROPERTYEDITUNIT>
15  <PROPERTYEDITHELP>TRUE merges all subscription items in a single cache</PROPERTYEDITHELP>
</LinkTopicCache>
<UniqueDeviceGroup>
  <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
  <DEFAULTVALUE>1</DEFAULTVALUE>
  <PROPERTYMIN>0</PROPERTYMIN>
  <PROPERTYMAX>1</PROPERTYMAX>
  <PROPERTYEDITUNIT>T/F</PROPERTYEDITUNIT>
  <PROPERTYEDITHELP>TRUE requires all device group names be unique</PROPERTYEDITHELP>
20  </UniqueDeviceGroup>
<ProtocolTimerTick>
  <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
  <DEFAULTVALUE>50</DEFAULTVALUE>
  <PROPERTYMIN>10</PROPERTYMIN>
  <PROPERTYMAX>100000</PROPERTYMAX>
  <PROPERTYEDITUNIT>ms</PROPERTYEDITUNIT>
  <PROPERTYEDITHELP>Protocol timer tick interval in milliseconds</PROPERTYEDITHELP>
25  </ProtocolTimerTick>
<TransactionTimeout>
  <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
  <DEFAULTVALUE>60000</DEFAULTVALUE>
  <PROPERTYMIN>10</PROPERTYMIN>
  <PROPERTYMAX>60000</PROPERTYMAX>
  <PROPERTYEDITUNIT>ms</PROPERTYEDITUNIT>
  <PROPERTYEDITHELP>Default transaction timeout for hierarchies with no
30  configuration</PROPERTYEDITHELP>
</TransactionTimeout>
<LockConfigurationFile>
  <PROPERTYTYPE>VT_BOOL</PROPERTYTYPE>
  <DEFAULTVALUE>0</DEFAULTVALUE>
35
40
45
50

```

```
<PROPERTYMIN>0</PROPERTYMIN>
<PROPERTYMAX>1</PROPERTYMAX>
<PROPERTYEDITUNIT>T/F</PROPERTYEDITUNIT>
<PROPERTYEDITHELP>TRUE for non-Magellan configurers locked out</PROPERTYEDITHELP>
5 </LockConfigurationFile>
<SubscriptionTransactionRatio>
  <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
  <DEFAULTVALUE>2</DEFAULTVALUE>
  <PROPERTYMIN>1</PROPERTYMIN>
10 <PROPERTYMAX>1000</PROPERTYMAX>
  <PROPERTYEDITUNIT/>
  <PROPERTYEDITHELP>Subscription / Transaction ratio</PROPERTYEDITHELP>
</SubscriptionTransactionRatio>
</PROPERTYRULES>
15 </HIERARCHYNODE>
<HIERARCHYNODE TYPE="PORT_TCPIP" EDITORID="{1B48839F-B68E-4CDA-A090-C8BD87932126}" PACKAGEID="{7F5B8DEE-AB78-11D4-93E4-00B0D0201D61}" DELIMITER=".">
  <CHILDRULES>
    <PLC5_TCPIP MAXOCCURRENCES="100"/>
20 <SLC500_TCPIP MAXOCCURRENCES="100"/>
    <PYRAMID_EI MAXOCCURRENCES="100"/>
  </CHILDRULES>
  <PROPERTYRULES>
    <MaxQueuedMsgs>
      <!--Maximum number of queued messages for a topic-->
      <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
      <DEFAULTVALUE>4</DEFAULTVALUE>
      <PROPERTYMIN>1</PROPERTYMIN>
      <PROPERTYMAX>20</PROPERTYMAX>
30 <PROPERTYEDITHEADER>Max QueuedMsgs</PROPERTYEDITHEADER>
    <PROPERTYEDITUNIT/>
    <PROPERTYEDITHELP>Number of Messages</PROPERTYEDITHELP>
  </MaxQueuedMsgs>
  <MaxSockets>
    <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
    <DEFAULTVALUE>200</DEFAULTVALUE>
    <PROPERTYMIN>1</PROPERTYMIN>
    <PROPERTYMAX>200</PROPERTYMAX>
    <PROPERTYEDITHEADER>Maximum number of sockets</PROPERTYEDITHEADER>
40 <PROPERTYEDITUNIT/>
    <PROPERTYEDITHELP>Maximum number of sockets</PROPERTYEDITHELP>
  </MaxSockets>
  <MaxUnsolConn>
    <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
    <DEFAULTVALUE>20</DEFAULTVALUE>
    <PROPERTYMIN>1</PROPERTYMIN>
    <PROPERTYMAX>30</PROPERTYMAX>
    <PROPERTYEDITHEADER>Maximum number of peer-to-peer
45 connections</PROPERTYEDITHEADER>
    <PROPERTYEDITUNIT/>
    <PROPERTYEDITHELP>Maximum number of peer-to-peer connections</PROPERTYEDITHELP>
  </MaxUnsolConn>
  <UnsolicitedMsgTimeout>
    <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
```

```
<DEFAULTVALUE>5000</DEFAULTVALUE>
<PROPERTYMIN>1000</PROPERTYMIN>
<PROPERTYMAX>10000</PROPERTYMAX>
<PROPERTYEDITHEADER>Timeout for peer-to-peer unsolicited
5 communication</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT>MSec</PROPERTYEDITUNIT>
<PROPERTYEDITHELP>Timeout for peer-to-peer unsolicited communication</PROPERTYEDITHELP>
</UnsolicitedMsgTimeout>
<FlipStringBytes>
10 <!--Flip String Bytes-->
<PROPERTYTYPE>VT_BOOL</PROPERTYTYPE>
<DEFAULTVALUE>0</DEFAULTVALUE>
<PROPERTYMIN>0</PROPERTYMIN>
<PROPERTYMAX>1</PROPERTYMAX>
15 <PROPERTYEDITHEADER>Flip String Bytes</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Flip String Bytes</PROPERTYEDITHELP>
</FlipStringBytes>
</PROPERTYRULES>
20 </HIERARCHYNODE>
<HIERARCHYNODE TYPE="PLC5_TCPIP" PACKAGEID="" EDITORID="{85B57144-49F8-47D2-B5B7-
E55CD82CFBC2}" DELIMITER=".">
<CHILDRULES ENABLEDEVICEGROUPS="1"/>
<PROPERTYRULES>
25 <HostName>
<PROPERTYTYPE>VT_BSTR</PROPERTYTYPE>
<DEFAULTVALUE/>
<PROPERTYMIN>0</PROPERTYMIN>
<PROPERTYMAX>255</PROPERTYMAX>
30 <PROPERTYEDITHEADER>device IP address</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Device IP Host Name/Address</PROPERTYEDITHELP>
</HostName>
<DataBlockSize>
35 <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>2000</DEFAULTVALUE>
<PROPERTYMIN>2</PROPERTYMIN>
<PROPERTYMAX>2000</PROPERTYMAX>
<PROPERTYEDITHEADER>Message data block size (bytes) for PLC5 on
40 Ethernet</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Message data block size (bytes) for PLC5 on
Ethernet</PROPERTYEDITHELP>
</DataBlockSize>
45 <ReplyTimeout>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>15</DEFAULTVALUE>
<PROPERTYMIN>1</PROPERTYMIN>
<PROPERTYMAX>300</PROPERTYMAX>
50 <PROPERTYEDITHEADER>ReplyTimeout</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT>Sec</PROPERTYEDITUNIT>
<PROPERTYEDITHELP>Reply Timeout</PROPERTYEDITHELP>
</ReplyTimeout>
<SupportsPid>
```

```

<PROPERTYTYPE>VT_BOOL</PROPERTYTYPE>
<DEFAULTVALUE>1</DEFAULTVALUE>
<PROPERTYMIN>0</PROPERTYMIN>
<PROPERTYMAX>1</PROPERTYMAX>
5 <PROPERTYEDITHEADER>Support PID and String Files</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Support PID and String Files</PROPERTYEDITHELP>
</SupportsPID>
<UnsolClientMsg>
10 <PROPERTYTYPE>VT_BOOL</PROPERTYTYPE>
<DEFAULTVALUE>1</DEFAULTVALUE>
<PROPERTYMIN>0</PROPERTYMIN>
<PROPERTYMAX>1</PROPERTYMAX>
<PROPERTYEDITHEADER>Support Unsolicited 'CLIENT' Messaging</PROPERTYEDITHEADER>
15 <PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Support Unsolicited 'CLIENT' Messaging</PROPERTYEDITHELP>
</UnsolClientMsg>
<ConnectionTimeout>
<!--Connection Timeout-->
20 <PROPERTYTYPE>VT_U4</PROPERTYTYPE>
<DEFAULTVALUE>2000</DEFAULTVALUE>
<PROPERTYMIN>1000</PROPERTYMIN>
<PROPERTYMAX>20000</PROPERTYMAX>
<PROPERTYEDITHEADER>Connection Attempt Timeout</PROPERTYEDITHEADER>
25 <PROPERTYEDITUNIT>MSec</PROPERTYEDITUNIT>
<PROPERTYEDITHELP>Connection Attempt Timeout</PROPERTYEDITHELP>
</ConnectionTimeout>
</PROPERTYRULES>
</HIERARCHYNODE>
30 <HIERARCHYNODE TYPE="SLC500_TCPIP" PACKAGEID="" EDITORID="{63B40882-4981-41F0-868B-
CBCD508C0733}" DELIMITER=".">"
<CHILD RULES ENABLEDEVICEGROUPS="1"/>
<PROPERTYRULES>
<HostName>
35 <PROPERTYTYPE>VT_BSTR</PROPERTYTYPE>
<DEFAULTVALUE>10.32.12.36</DEFAULTVALUE>
<PROPERTYMIN>0</PROPERTYMIN>
<PROPERTYMAX>255</PROPERTYMAX>
<PROPERTYEDITHEADER>device IP address</PROPERTYEDITHEADER>
40 <PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Device IP Host Name/Address</PROPERTYEDITHELP>
</HostName>
<DataBlockSize>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
45 <DEFAULTVALUE>510</DEFAULTVALUE>
<PROPERTYMIN>2</PROPERTYMIN>
<PROPERTYMAX>510</PROPERTYMAX>
<PROPERTYEDITHEADER>Data packet size for PLC5 on Ethernet</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
50 <PROPERTYEDITHELP>Data packet size for PLC5 on Ethernet</PROPERTYEDITHELP>
</DataBlockSize>
<ReplyTimeout>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>15</DEFAULTVALUE>

```

```

<PROPERTYMIN>1</PROPERTYMIN>
<PROPERTYMAX>300</PROPERTYMAX>
<PROPERTYEDITHEADER>ReplyTimeout</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT>Sec</PROPERTYEDITUNIT>
5 <PROPERTYEDITHELP>Reply Timeout</PROPERTYEDITHELP>
</ReplyTimeout>
<ConnectionTimeout>
<!--Connection Timeout-->
<PROPERTYTYPE>VT_U4</PROPERTYTYPE>
10 <DEFAULTVALUE>2000</DEFAULTVALUE>
<PROPERTYMIN>1000</PROPERTYMIN>
<PROPERTYMAX>20000</PROPERTYMAX>
<PROPERTYEDITHEADER>Connection Attempt Timeout</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT>MSec</PROPERTYEDITUNIT>
15 <PROPERTYEDITHELP>Connection Attempt Timeout</PROPERTYEDITHELP>
</ConnectionTimeout>
</PROPERTYRULES>
</HIERARCHYNODE>
<HIERARCHYNODE TYPE="PYRAMID_EI" EDITORID="{4D9DE9D0-48BB-489E-9CED-
20 5284FFA3EFED}" DELIMITER=".":>
<CHILDRULES>
<PYRAMID_KA MAXOCCURRENCES="4"/>
<PYRAMID_RM MAXOCCURRENCES="1"/>
<PYRAMID_PLCS250 MAXOCCURRENCES="4"/>
25 </CHILDRULES>
<PROPERTYRULES>
<HostName>
<PROPERTYTYPE>VT_BSTR</PROPERTYTYPE>
<DEFAULTVALUE>10.32.12.36</DEFAULTVALUE>
30 <PROPERTYMIN>0</PROPERTYMIN>
<PROPERTYMAX>255</PROPERTYMAX>
<PROPERTYEDITHEADER>device IP address</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Device IP Host Name/Address</PROPERTYEDITHELP>
35 </HostName>
<ConnectionTimeout>
<!--Connection Timeout-->
<PROPERTYTYPE>VT_U4</PROPERTYTYPE>
<DEFAULTVALUE>2000</DEFAULTVALUE>
40 <PROPERTYMIN>1000</PROPERTYMIN>
<PROPERTYMAX>20000</PROPERTYMAX>
<PROPERTYEDITHEADER>Connection Attempt Timeout</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT>MSec</PROPERTYEDITUNIT>
<PROPERTYEDITHELP>Connection Attempt Timeout</PROPERTYEDITHELP>
45 </ConnectionTimeout>
</PROPERTYRULES>
</HIERARCHYNODE>
<HIERARCHYNODE TYPE="PYRAMID_PLCS250" PACKAGEID="" EDITORID="{A7EBCF9D-9ECE-
50 4DD5-B417-506ABA20855F}" DELIMITER=".":>
<CHILDRULES ENABLEDEVICEGROUPS="1"/>
<PROPERTYRULES>
<ReplyTimeout>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>15</DEFAULTVALUE>

```

```
<PROPERTYMIN>1</PROPERTYMIN>
<PROPERTYMAX>300</PROPERTYMAX>
<PROPERTYEDITHEADER>ReplyTimeout</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT>Sec</PROPERTYEDITUNIT>
5 <PROPERTYEDITHELP>Reply Timeout</PROPERTYEDITHELP>
</ReplyTimeout>
<DataBlockSize>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>2000</DEFAULTVALUE>
10 <PROPERTYMIN>2</PROPERTYMIN>
<PROPERTYMAX>2000</PROPERTYMAX>
<PROPERTYEDITHEADER>Message packet size for PLC5 on Ethernet</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Message packet size for PLC5 on Ethernet</PROPERTYEDITHELP>
15 </DataBlockSize>
</PROPERTYRULES>
</HIERARCHYNODE>
<HIERARCHYNODE TYPE="PYRAMID_RM" EDITORID="" DELIMITER=".">.
<CHILDRULES>
20 <PORT_DHP MAXOCCURRENCES="2"/>
<PORT_DH MAXOCCURRENCES="2"/>
</CHILDRULES>
</HIERARCHYNODE>
<HIERARCHYNODE TYPE="PYRAMID_KA" EDITORID="{8A04DA4D-F6C0-4C1F-A771-
25 806CDA40029F}" DELIMITER=".">.
<CHILDRULES>
<PORT_DHP MAXOCCURRENCES="2"/>
<PORT_DH MAXOCCURRENCES="2"/>
</CHILDRULES>
30 <PROPERTYRULES>
<Pushwheel>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>1</DEFAULTVALUE>
<PROPERTYMIN>1</PROPERTYMIN>
35 <PROPERTYMAX>4</PROPERTYMAX>
<PROPERTYEDITHEADER>Push Wheel</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Push Wheel</PROPERTYEDITHELP>
</Pushwheel>
40 </PROPERTYRULES>
</HIERARCHYNODE>
<HIERARCHYNODE TYPE="PORT_DHP" EDITORID="{2B22472C-2F98-4D1B-A211-7036FCBF9AA2}" PACKAGEID="" DELIMITER=".">.
<CHILDRULES>
45 <PLC5_DHP MAXOCCURRENCES="100"/>
<SLC500_DHP MAXOCCURRENCES="100"/>
<PLC3_DHP MAXOCCURRENCES="100"/>
</CHILDRULES>
<PROPERTYRULES>
50 <PortNumber>
<!--Port Number-->
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>2</DEFAULTVALUE>
<PROPERTYMIN>2</PROPERTYMIN>
```

```

<PROPERTYMAX>3</PROPERTYMAX>
<PROPERTYEDITHEADER>Channel Number</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Channel Number</PROPERTYEDITHELP>
5   </PortNumber>
</PROPERTYRULES>
</HIERARCHYNODE>
<HIERARCHYNODE TYPE="PORT_DH" PACKAGEID="" EDITORID="{67D79FDA-4DC4-43A2-9B24-
5735C3E3B428}" DELIMITER=".">.
10  <CHILDRULES>
    <PLC2_DH MAXOCCURRENCES="100"/>
</CHILDRULES>
<PROPERTYRULES>
    <PortNumber>
15    <!--Port Number-->
    <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
    <DEFAULTVALUE>2</DEFAULTVALUE>
    <PROPERTYMIN>2</PROPERTYMIN>
    <PROPERTYMAX>3</PROPERTYMAX>
20    <PROPERTYEDITHEADER>Channel Number</PROPERTYEDITHEADER>
    <PROPERTYEDITUNIT/>
    <PROPERTYEDITHELP>Channel Number</PROPERTYEDITHELP>
    </PortNumber>
</PROPERTYRULES>
25  </HIERARCHYNODE>
<HIERARCHYNODE TYPE="PLC5_DHP" EDITORID="{BF71A6EF-818F-4406-9038-60745B9A1538}"-
DELIMITER=".">.
    <CHILDRULES ENABLEDEVICEGROUPS="1"/>
    <PROPERTYRULES>
30    <NodeAddress>
        <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
        <DEFAULTVALUE>1</DEFAULTVALUE>
        <PROPERTYMIN>0</PROPERTYMIN>
        <PROPERTYMAX>63</PROPERTYMAX>
35    <PROPERTYEDITHEADER>DH+ Node Nukmber</PROPERTYEDITHEADER>
        <PROPERTYEDITUNIT>Octal</PROPERTYEDITUNIT>
        <PROPERTYEDITHELP>Datahighway Plus Node Number</PROPERTYEDITHELP>
    </NodeAddress>
    <ReplyTimeout>
40        <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
        <DEFAULTVALUE>15</DEFAULTVALUE>
        <PROPERTYMIN>1</PROPERTYMIN>
        <PROPERTYMAX>300</PROPERTYMAX>
        <PROPERTYEDITHEADER>ReplyTimeout</PROPERTYEDITHEADER>
45        <PROPERTYEDITUNIT>Sec</PROPERTYEDITUNIT>
        <PROPERTYEDITHELP>Reply Timeout</PROPERTYEDITHELP>
    </ReplyTimeout>
    <DataBlockSize>
50        <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
        <DEFAULTVALUE>240</DEFAULTVALUE>
        <PROPERTYMIN>2</PROPERTYMIN>
        <PROPERTYMAX>240</PROPERTYMAX>
        <PROPERTYEDITHEADER>Message packet size for PLC5 on Datahiway
Plus</PROPERTYEDITHEADER>

```

```
<PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Message packet size for PLC5 on Datahiway Plus</PROPERTYEDITHELP>
</DataBlockSize>
</PROPERTYRULES>
5 </HIERARCHYNODE>
<HIERARCHYNODE TYPE="SLC500_DHP" EDITORID="{78FF43FC-EA3B-4B37-AFA9-
D8D95CFB69A5}" DELIMITER=".">
<CHILDRULES ENABLEDEVICEGROUPS="1"/>
<PROPERTYRULES>
10 <NodeAddress>
    <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
    <DEFAULTVALUE>1</DEFAULTVALUE>
    <PROPERTYMIN>0</PROPERTYMIN>
    <PROPERTYMAX>63</PROPERTYMAX>
15 <PROPERTYEDITHEADER>DH+ Node Nukmber</PROPERTYEDITHEADER>
    <PROPERTYEDITUNIT>Octal</PROPERTYEDITUNIT>
    <PROPERTYEDITHELP>Datahighway Plus Node Number</PROPERTYEDITHELP>
</NodeAddress>
<ReplyTimeout>
20     <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
     <DEFAULTVALUE>15</DEFAULTVALUE>
     <PROPERTYMIN>1</PROPERTYMIN>
     <PROPERTYMAX>300</PROPERTYMAX>
     <PROPERTYEDITHEADER>ReplyTimeout</PROPERTYEDITHEADER>
25     <PROPERTYEDITUNIT>MSec</PROPERTYEDITUNIT>
     <PROPERTYEDITHELP>Reply Timeout</PROPERTYEDITHELP>
</ReplyTimeout>
<DataBlockSize>
    <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
30     <DEFAULTVALUE>204</DEFAULTVALUE>
     <PROPERTYMIN>2</PROPERTYMIN>
     <PROPERTYMAX>204</PROPERTYMAX>
     <PROPERTYEDITHEADER>Message packet size for PLC5 on Datahiway
Plus</PROPERTYEDITHEADER>
35     <PROPERTYEDITUNIT/>
     <PROPERTYEDITHELP>Message packet size for PLC5 on Datahiway Plus</PROPERTYEDITHELP>
     </DataBlockSize>
</PROPERTYRULES>
</HIERARCHYNODE>
40 <HIERARCHYNODE TYPE="PLC3_DHP" EDITORID="{F59F96D7-FA1D-4C9F-91F3-FABC09E00683}"
DELIMITER=".">
    <CHILDRULES ENABLEDEVICEGROUPS="1"/>
    <PROPERTYRULES>
        <NodeAddress>
            <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
45            <DEFAULTVALUE>1</DEFAULTVALUE>
            <PROPERTYMIN>0</PROPERTYMIN>
            <PROPERTYMAX>63</PROPERTYMAX>
            <PROPERTYEDITHEADER>DH+ Node Nukmber</PROPERTYEDITHEADER>
            <PROPERTYEDITUNIT>Octal</PROPERTYEDITUNIT>
            <PROPERTYEDITHELP>Datahighway Plus Node Number</PROPERTYEDITHELP>
50        </NodeAddress>
        <ReplyTimeout>
            <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
```

```

<DEFAULTVALUE>15</DEFAULTVALUE>
<PROPERTYMIN>1</PROPERTYMIN>
<PROPERTYMAX>300</PROPERTYMAX>
<PROPERTYEDITHEADER>ReplyTimeout</PROPERTYEDITHEADER>
5 <PROPERTYEDITUNIT>MSec</PROPERTYEDITUNIT>
<PROPERTYEDITHELP>Reply Timeout</PROPERTYEDITHELP>
</ReplyTimeout>
<DataBlockSize>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
10 <DEFAULTVALUE>240</DEFAULTVALUE>
<PROPERTYMIN>2</PROPERTYMIN>
<PROPERTYMAX>240</PROPERTYMAX>
<PROPERTYEDITHEADER>Message packet size for PLC5 on Datahiway
Plus</PROPERTYEDITHEADER>
15 <PROPERTYEDITUNIT/>
<PROPERTYEDITHELP>Message packet size for PLC5 on Datahiway Plus</PROPERTYEDITHELP>
</DataBlockSize>
<PROPERTYRULES>
</HIERARCHYNODE>
20 <HIERARCHYNODE TYPE="PLC2_DH" EDITORID="{9C53E368-C85A-44B1-8F7E-5EE6EE07DBED}">
DELIMITER=".">.
<CHILDRULES ENABLEDEVICEGROUPS="1"/>
<PROPERTYRULES>
<NodeAddress>
25 <PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>1</DEFAULTVALUE>
<PROPERTYMIN>0</PROPERTYMIN>
<PROPERTYMAX>255</PROPERTYMAX>
<PROPERTYEDITHEADER>DH Node Number</PROPERTYEDITHEADER>
30 <PROPERTYEDITUNIT>Octal</PROPERTYEDITUNIT>
<PROPERTYEDITHELP>Datahighway Node Number</PROPERTYEDITHELP>
</NodeAddress>
<ReplyTimeout>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
35 <DEFAULTVALUE>15</DEFAULTVALUE>
<PROPERTYMIN>1</PROPERTYMIN>
<PROPERTYMAX>300</PROPERTYMAX>
<PROPERTYEDITHEADER>ReplyTimeout</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT>MSec</PROPERTYEDITUNIT>
40 <PROPERTYEDITHELP>Reply Timeout</PROPERTYEDITHELP>
</ReplyTimeout>
<DataBlockSize>
<PROPERTYTYPE>VT_UI4</PROPERTYTYPE>
<DEFAULTVALUE>240</DEFAULTVALUE>
45 <PROPERTYMIN>100</PROPERTYMIN>
<PROPERTYMAX>240</PROPERTYMAX>
<PROPERTYEDITHEADER>Message packet size for PLC5 on Datahiway
Plus</PROPERTYEDITHEADER>
<PROPERTYEDITUNIT/>
50 <PROPERTYEDITHELP>Message packet size for PLC5 on Datahiway Plus</PROPERTYEDITHELP>
</DataBlockSize>
<PROPERTYRULES>
</HIERARCHYNODE>
</CONFIGURATIONRULES>

```

The following is an example of the configuration file 280 which makes up a second part of a remote configuration specification's persistence area.

KEY	XML Attributes	Explanation
<DASConfiguration>		Root key for the configuration (DAS related data is inside this root key only). The presence of this root key potentially allows additional (user-) data to be stored in the same file under parallel root keys.
<DeviceNode>	TYPE NAME DELIMITER	Key for a hierarchy level (related to a DAS object). TYPE (server specific class) and NAME are required – DELIMITER is optional (defaults to “.” If not present)
<ItemHint>	TYPE NAME	Key for a pre-configured item or item hint. TYPE (server specific class) and NAME are required
<DeviceGroup>	TYPE NAME UPDATEINTERVAL	Key for an OPC access path/ DDE/SL topic (related to device groups). NAME and UPDATEINTERVAL are required – TYPE (server specific class) is optional
<Data>	TYPE NAME	Key for DA Server global (user) settings. TYPE (server specific class) and NAME are required
<System>		Key for DA Server global system settings.

```

5   <?xml version="1.0" encoding="UTF-8"?>
<DASConfiguration NAME="Wonderware ABTCP (non-official)" DELIMITER=".">
<!-- Wonderware AB TCP/IP Server-->
<created>03.29.2000 16:40</created>
<user>Clement Lie</user>
10  <DeviceNode TYPE="PORT_TCPIP" NAME="TCP" DELIMITER=".">
    <!--TCP/IP Port for ABTCP-->
    <MaxQueuedMsgs>3</MaxQueuedMsgs>
    <MaxSockets>200</MaxSockets>
    <MaxUnsolConn>20</MaxUnsolConn>
15  <UnsolicitedMsgTimeout>5000</UnsolicitedMsgTimeout>
    <DeviceNode TYPE="PYRAMID_EI" NAME="EI" DELIMITER=".">
        <!--Ethernet Interface Module of Pyramid Integrator-->
        <HostName>P5250_1</HostName>
        <ConnectionTimeout>2000</ConnectionTimeout>
20    <DeviceNode TYPE="PYRAMID_PLCS5250" NAME="P5250_1" DELIMITER=".">
        <DataBlockSize>2000</DataBlockSize>
        <ReplyTimeout>15</ReplyTimeout>
        <DeviceGroup NAME="P5250_1_Fast" UPDATEINTERVAL="100"/>
        <DeviceGroup NAME="P5250_1" UPDATEINTERVAL="1000"/>
25    <DeviceGroup NAME="P5250_1_Slow" UPDATEINTERVAL="5000"/>
    </DeviceNode>

```

```

<DeviceNode TYPE="PYRAMID_RM" NAME="RM" DELIMITER=".">
  <DeviceNode TYPE="PORT_DHP" NAME="CH2" DELIMITER=".">
    <PortNumber>2</PortNumber>
    <DeviceNode TYPE="PLCS_DHP" NAME="P511_1" DELIMITER=".">
      <NodeAddress>6</NodeAddress>
      <DataBlockSize>240</DataBlockSize>
      <ReplyTimeout>15</ReplyTimeout>
      <DeviceGroup NAME="P511_1" UPDATEINTERVAL="1000"/>
      <DeviceGroup NAME="P511_1_Fast" UPDATEINTERVAL="100"/>
      <DeviceGroup NAME="P511_1_Slow" UPDATEINTERVAL="5000"/>
    </DeviceNode>
  </DeviceNode>
<DeviceNode TYPE="PORT_DHP" NAME="CH3" DELIMITER=".">
  <PortNumber>3</PortNumber>
  <DeviceNode NAME="SLC504_2" TYPE="SLC500_DHP" DELIMITER=".">
    <NodeAddress>24</NodeAddress>
    <ReplyTimeout>15</ReplyTimeout>
    <DataBlockSize>204</DataBlockSize>
  </DeviceNode>
  <DeviceNode NAME="P520E_2" TYPE="PLC5_DHP" DELIMITER=".">
    <NodeAddress>7</NodeAddress>
    <ReplyTimeout>15</ReplyTimeout>
    <DataBlockSize>240</DataBlockSize>
  </DeviceNode>
</DeviceNode>
<DeviceNode TYPE="PYRAMID_KA" NAME="KA" DELIMITER=".">
  <Pushwheel>1</Pushwheel>
  <DeviceNode TYPE="PORT_DHP" NAME="CH2" DELIMITER=".">
    <PortNumber>2</PortNumber>
    <DeviceNode TYPE="PLC3_DHP" NAME="P310_1" DELIMITER=".">
      <NodeAddress>9</NodeAddress>
      <DataBlockSize>240</DataBlockSize>
      <ReplyTimeout>15</ReplyTimeout>
      <DeviceGroup NAME="PLC3" UPDATEINTERVAL="1000"/>
      <DeviceGroup NAME="PLC3_Fast" UPDATEINTERVAL="100"/>
      <DeviceGroup NAME="PLC3_Slow" UPDATEINTERVAL="5000"/>
    </DeviceNode>
    <DeviceNode NAME="P520C_1" TYPE="PLC5_DHP" DELIMITER=".">
      <NodeAddress>52</NodeAddress>
      <ReplyTimeout>15</ReplyTimeout>
      <DataBlockSize>240</DataBlockSize>
    </DeviceNode>
    <DeviceNode NAME="P580E_1" TYPE="PLC5_DHP" DELIMITER=".">
      <NodeAddress>5</NodeAddress>
      <ReplyTimeout>15</ReplyTimeout>
      <DataBlockSize>240</DataBlockSize>
    </DeviceNode>
    <DeviceNode NAME="P520E_2" TYPE="PLC5_DHP" DELIMITER=".">
      <NodeAddress>7</NodeAddress>
      <ReplyTimeout>15</ReplyTimeout>
      <DataBlockSize>240</DataBlockSize>
    </DeviceNode>
  </DeviceNode>
</DeviceNode>

```

```
<DeviceNode TYPE="PORT_DH" NAME="CH3" DELIMITER=".">>
  <PortNumber>2</PortNumber>
  <DeviceNode TYPE="PLC2_DH" NAME="P216_1" DELIMITER=".">>
    <NodeAddress>83</NodeAddress>
    <DataBlockSize>100</DataBlockSize>
    <ReplyTimeout>15</ReplyTimeout>
  </DeviceNode>
</DeviceNode>
</DeviceNode>
</DeviceNode>
<DeviceNode TYPE="PLC5_TCPIP" NAME="P520E_2" DELIMITER=".">>
  <!--PLC5 on TCP/IP-->
  <HostName>P520E_2</HostName>
  <ReplyTimeout>300</ReplyTimeout>
  <DataBlockSize>1000</DataBlockSize>
  <SupportsPID>1</SupportsPID>
  <UnsolClientMsg>1</UnsolClientMsg>
  <ConnectionTimeout>2000</ConnectionTimeout>
  <DeviceGroup NAME="P520E_2" UPDATEINTERVAL="1000"/>
  <DeviceGroup NAME="P520E_2_Slow" UPDATEINTERVAL="5000"/>
  <DeviceGroup NAME="P520E_2_Fast" UPDATEINTERVAL="100"/>
  <DeviceGroup NAME="P520E_2Unsol" UPDATEINTERVAL="10000000000"/>
</DeviceNode>
<DeviceNode TYPE="SLC500_TCPIP" NAME="SLC505_2" DELIMITER=".">>
  <HostName>SLC505_2</HostName>
  <ReplyTimeout>10</ReplyTimeout>
  <DataBlockSize>510</DataBlockSize>
  <ConnectionTimeout>2000</ConnectionTimeout>
  <DeviceGroup NAME="SLC505_1_Fast" UPDATEINTERVAL="100"/>
  <DeviceGroup NAME="SLC505_1" UPDATEINTERVAL="1000"/>
  <DeviceGroup NAME="SLC505_1_Slow" UPDATEINTERVAL="5000"/>
</DeviceNode>
<DeviceNode NAME="MyPLC5" TYPE="PLC5_TCPIP" DELIMITER=".">>
  <HostName>10.32.12.36</HostName>
  <DataBlockSize>2000</DataBlockSize>
  <ReplyTimeout>3</ReplyTimeout>
  <SupportsPID>1</SupportsPID>
  <UnsolClientMsg>1</UnsolClientMsg>
  <ConnectionTimeout>2000</ConnectionTimeout>
  <DeviceGroup NAME="Fast PLC5" TYPE="Topic" UPDATEINTERVAL="20"/>
  <DeviceGroup NAME="Medium PLC5" TYPE="Topic" UPDATEINTERVAL="1000"/>
  <DeviceGroup NAME="Slow PLC5" TYPE="Topic" UPDATEINTERVAL="10000"/>
  <DeviceItem TYPE="" NAME="N7:0"/>
  <DeviceItem TYPE="" NAME="N7:2"/>
  <DeviceItem TYPE="" NAME="N7:3"/>
  <DeviceItem TYPE="" NAME="N7:11"/>
  <DeviceItem TYPE="" NAME="N7:14"/>
  <DeviceItem TYPE="" NAME="N17:0"/>
  <DeviceItem TYPE="" NAME="N17:1"/>
  <DeviceItem TYPE="" NAME="N17:2"/>
  <DeviceItem TYPE="" NAME="N17:3"/>
  <DeviceItem TYPE="" NAME="N17:50"/>
  <DeviceItem TYPE="" NAME="N17:99"/>
  <DeviceItem TYPE="" NAME="N27:0"/>
```

```

<DeviceItem TYPE="" NAME="N27:1"/>
<DeviceItem TYPE="" NAME="N27:2"/>
<DeviceItem TYPE="" NAME="N27:3"/>
<DeviceItem TYPE="" NAME="N27:50"/>
5 <DeviceItem TYPE="" NAME="N27:99"/>
<DeviceItem TYPE="" NAME="B3:0/8"/>
<DeviceItem TYPE="" NAME="B13:0/180"/>
<DeviceItem TYPE="" NAME="A9:0-8"/>
<DeviceItem TYPE="" NAME="F8:2"/>
10 <DeviceItem TYPE="" NAME="F8:12"/>
<DeviceItem TYPE="" NAME="ST10:0"/>
</DeviceNode>
<DeviceNode NAME="MyPLC37" TYPE="PLC5_TCPIP" DELIMITER=".">
<HostName>10.32.12.37</HostName>
15 <DataBlockSize>2000</DataBlockSize>
<ReplyTimeout>3</ReplyTimeout>
<SupportsPID>1</SupportsPID>
<UnsolClientMsg>1</UnsolClientMsg>
<ConnectionTimeout>2000</ConnectionTimeout>
20 <DeviceGroup NAME="Fast PLC37" TYPE="Topic" UPDATEINTERVAL="20"/>
<DeviceGroup NAME="Medium PLC37" TYPE="Topic" UPDATEINTERVAL="1000"/>
<DeviceGroup NAME="Slow PLC37" TYPE="Topic" UPDATEINTERVAL="10000"/>
<ItemHint TYPE="" NAME="N7:0-4"/>
<ItemHint TYPE="" NAME="F8:0-3"/>
25 <ItemHint TYPE="" NAME="BT9:0-3"/>
<ItemHint TYPE="" NAME="N10:0-74"/>
<ItemHint TYPE="" NAME="B11:0-0"/>
<ItemHint TYPE="" NAME="PD12:0-0"/>
<ItemHint TYPE="" NAME="N13:0-3"/>
30 <ItemHint TYPE="" NAME="N14:0-4"/>
<ItemHint TYPE="" NAME="N15:0-4"/>
</DeviceNode>
</DeviceNode>
<System NAME="SYSTEM" TYPE="SYSTEM">
<UpdateInterval>1000</UpdateInterval>
35 <SlowPollInterval>10000</SlowPollInterval>
<CaseSensitive>0</CaseSensitive>
<DefaultPokeMode>0</DefaultPokeMode>
<DefaultDelimiter>".</DefaultDelimiter>
40 <SimulationMode>0</SimulationMode>
<EnableSystemItems>1</EnableSystemItems>
<LinkTopicCache>0</LinkTopicCache>
<UniqueDeviceGroup>1</UniqueDeviceGroup>
<ProtocolTimerTick>60</ProtocolTimerTick>
45 <TransactionTimeout>2000</TransactionTimeout>
<LockConfigurationFile>0</LockConfigurationFile>
<SubscriptionTransactionRatio>2</SubscriptionTransactionRatio>
</System>
</DASConfiguration>
50

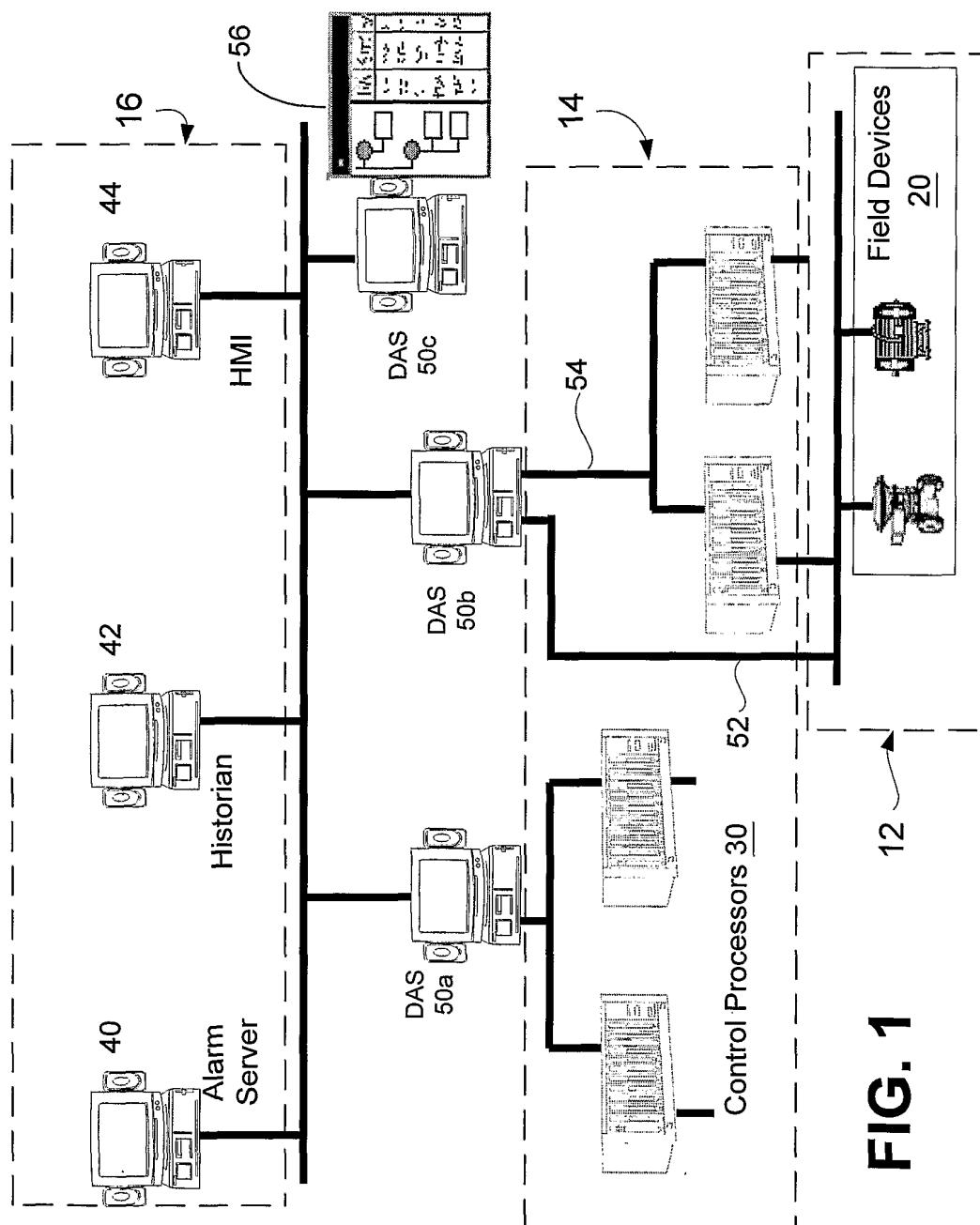
```

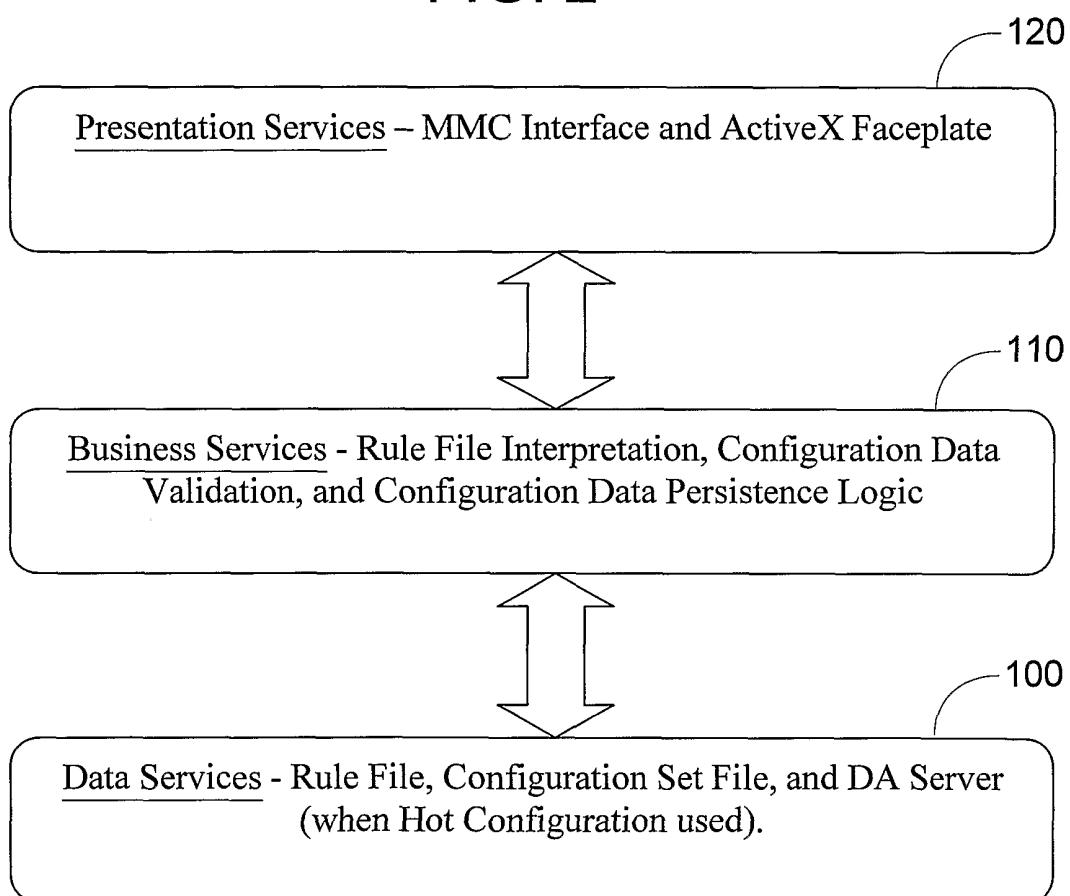
Illustrative embodiments of the present invention and certain variations thereof have been provided in the Figures and accompanying written description. The present invention is not

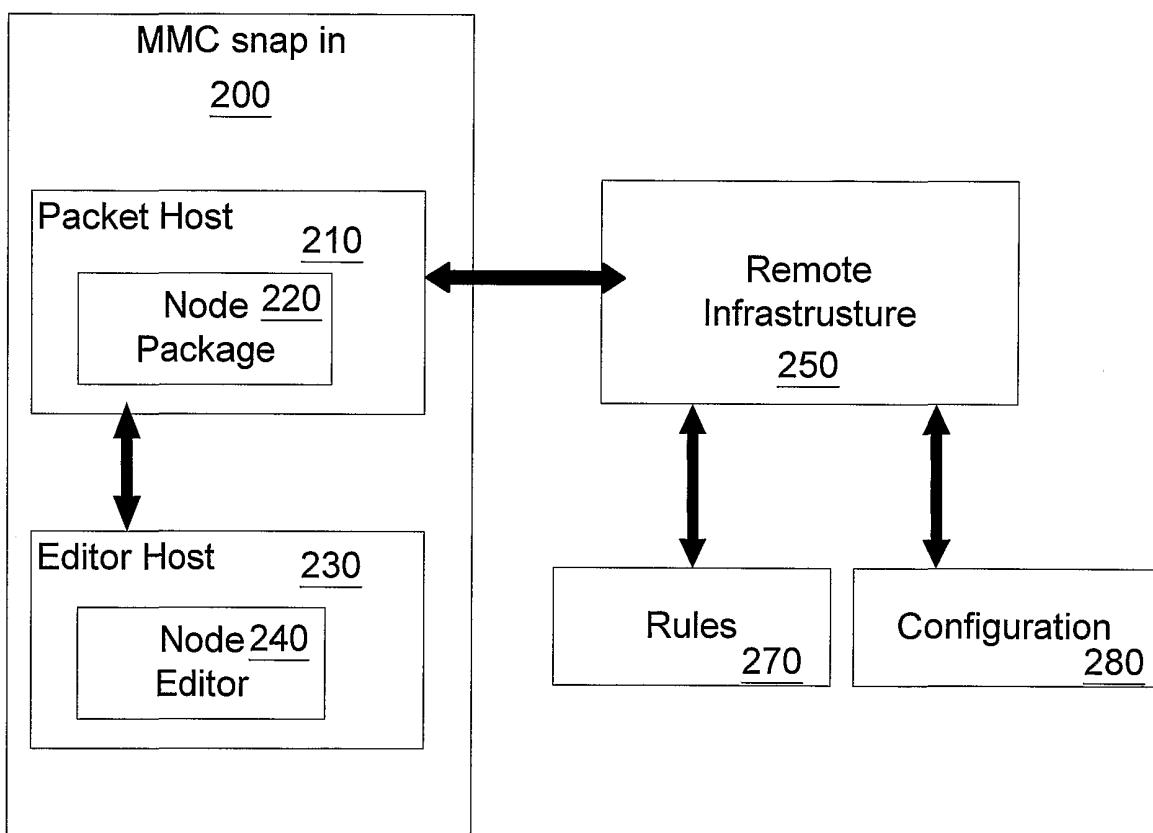
intended to be limited to these embodiments. Rather the present invention is intended to cover the disclosed embodiments as well as others falling within the scope and spirit of the invention to the fullest extent permitted in view of this disclosure and the inventions defined by the claims appended herein below.

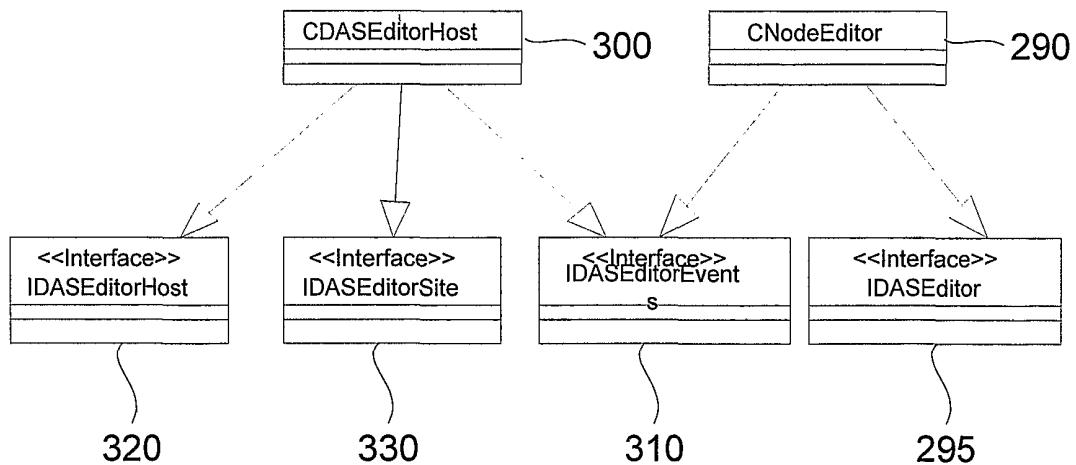
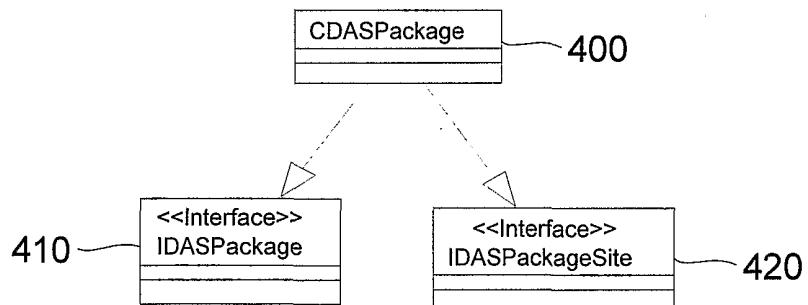
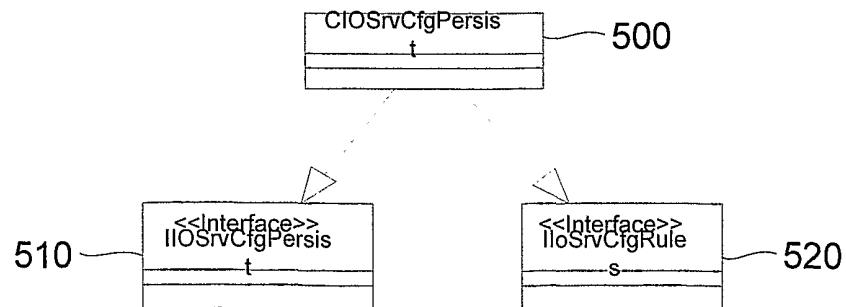
## WHAT IS CLAIMED IS:

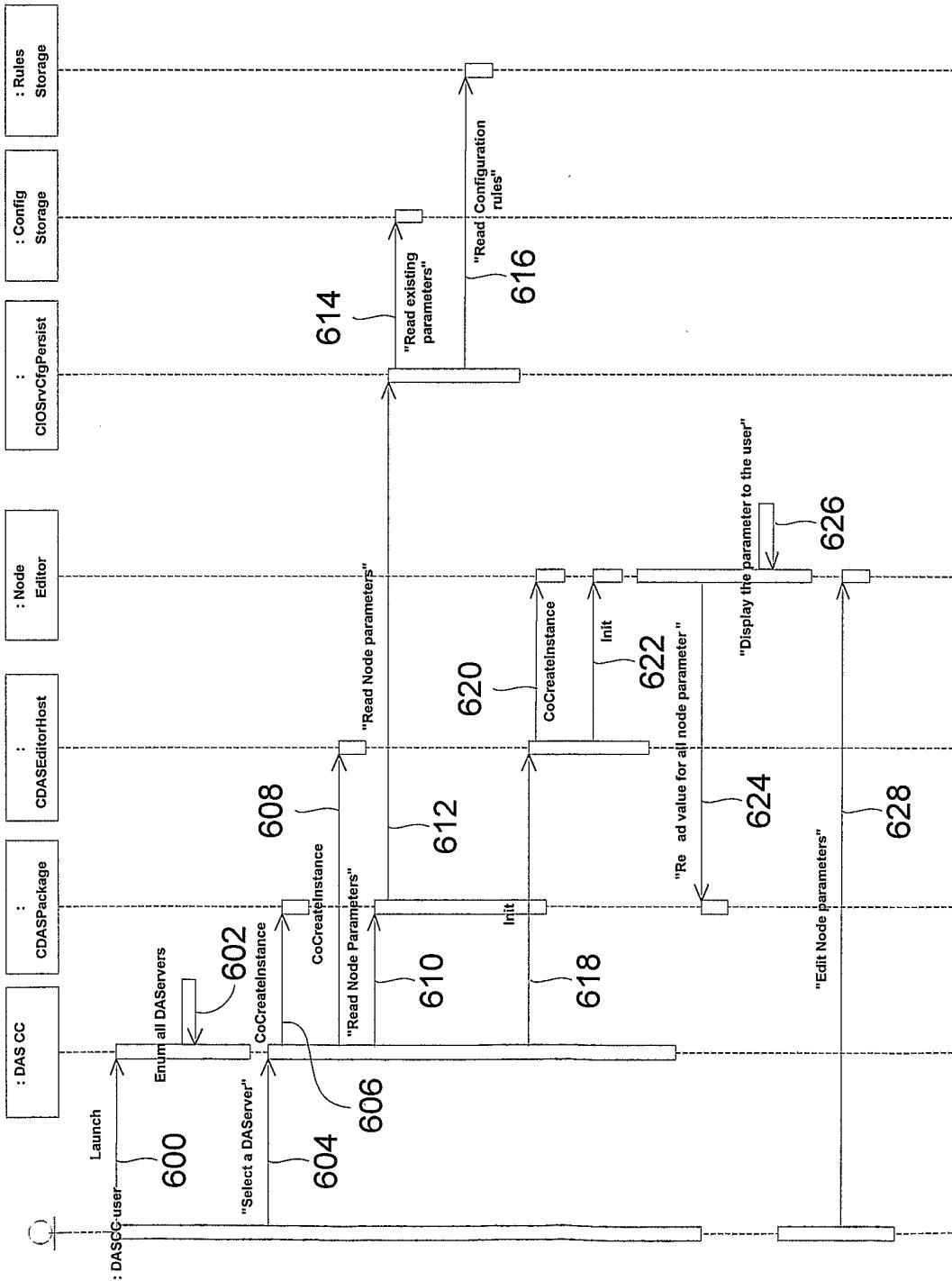
1. A distributed configuration utility facilitating remote configuration of process control data access servers, the distributed configuration utility comprising:
    - 5 a configuration editor located at a control console;
    - a configuration storage comprising descriptions of present server configurations;
    - a rules database for storing a set of configuration rules associated with each data access server configurable via the configuration utility, wherein the set of configuration rules provide a guide to the information within the configuration storage specifying a configuration for ones of the process control data access servers; and
  - 10 a server agent, located on a computing node remote from the control console and containing a configurable process control data access server, the server agent including executable procedures for notifying the configuration editor of the existence of the configurable process control data access server and thereby facilitating establishing a configuration interface between the configuration editor and the configurable process control data access server.
- 15
2. The distributed configuration utility of claim 1 wherein the set of configuration rules specify configuration hierarchy levels.
  3. The distributed configuration utility of claim 2 wherein the set of configuration rules specify properties associated with particular ones of configuration nodes of the configuration hierarchy levels.
  - 20
  4. The distributed configuration utility of claim 1 further comprising an executable notification mechanism that establishes a connection to an active data access server having changed configuration parameter values, and notifies the active data access server of the change to its configuration.
  - 25
  5. The distributed configuration utility of claim 1 further comprising a user editor host providing an interface supporting editor interface faceplates specified for particular component types of a process control data access server configuration.

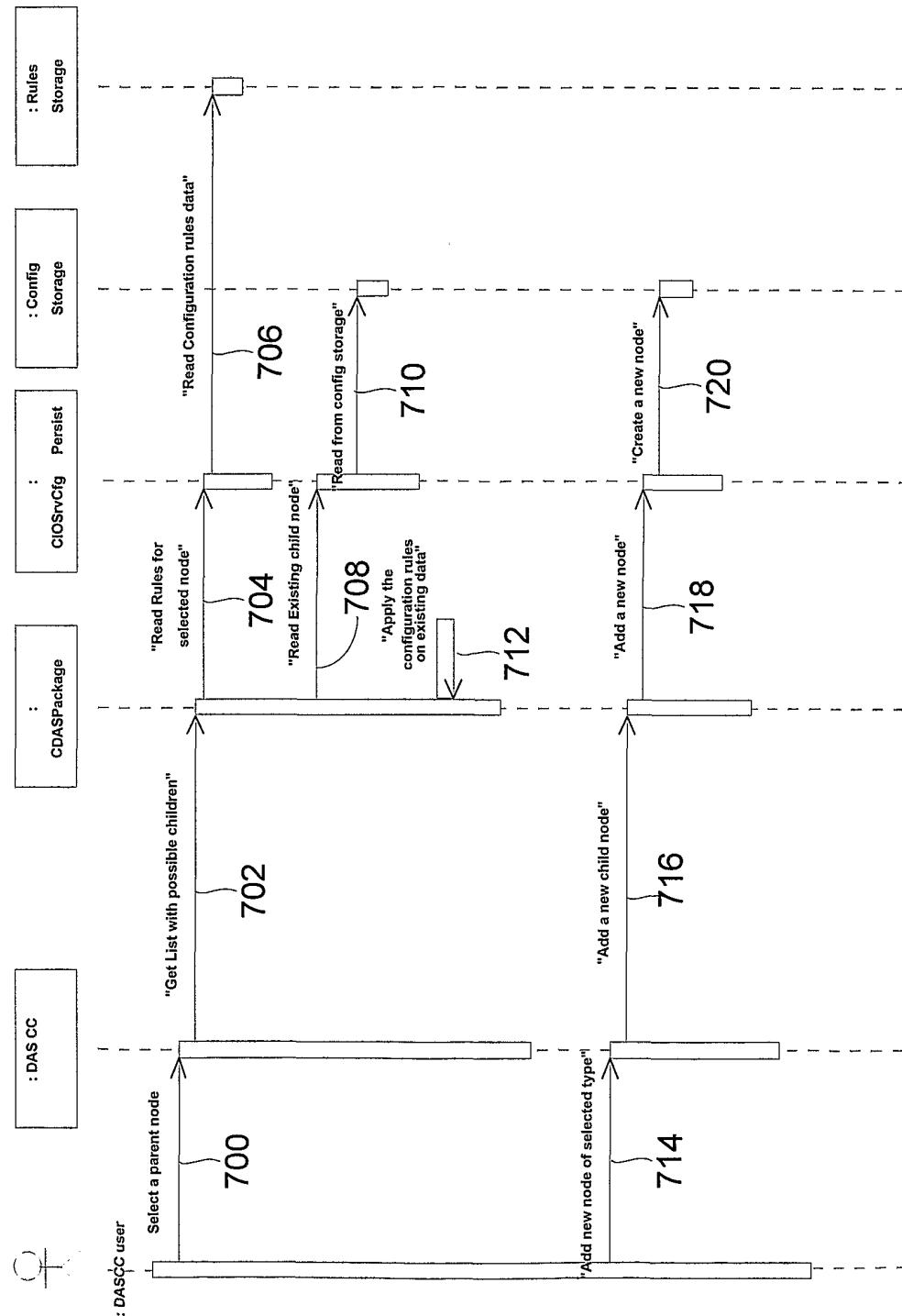
**FIG. 1**

**FIG. 2**

**FIG. 3**

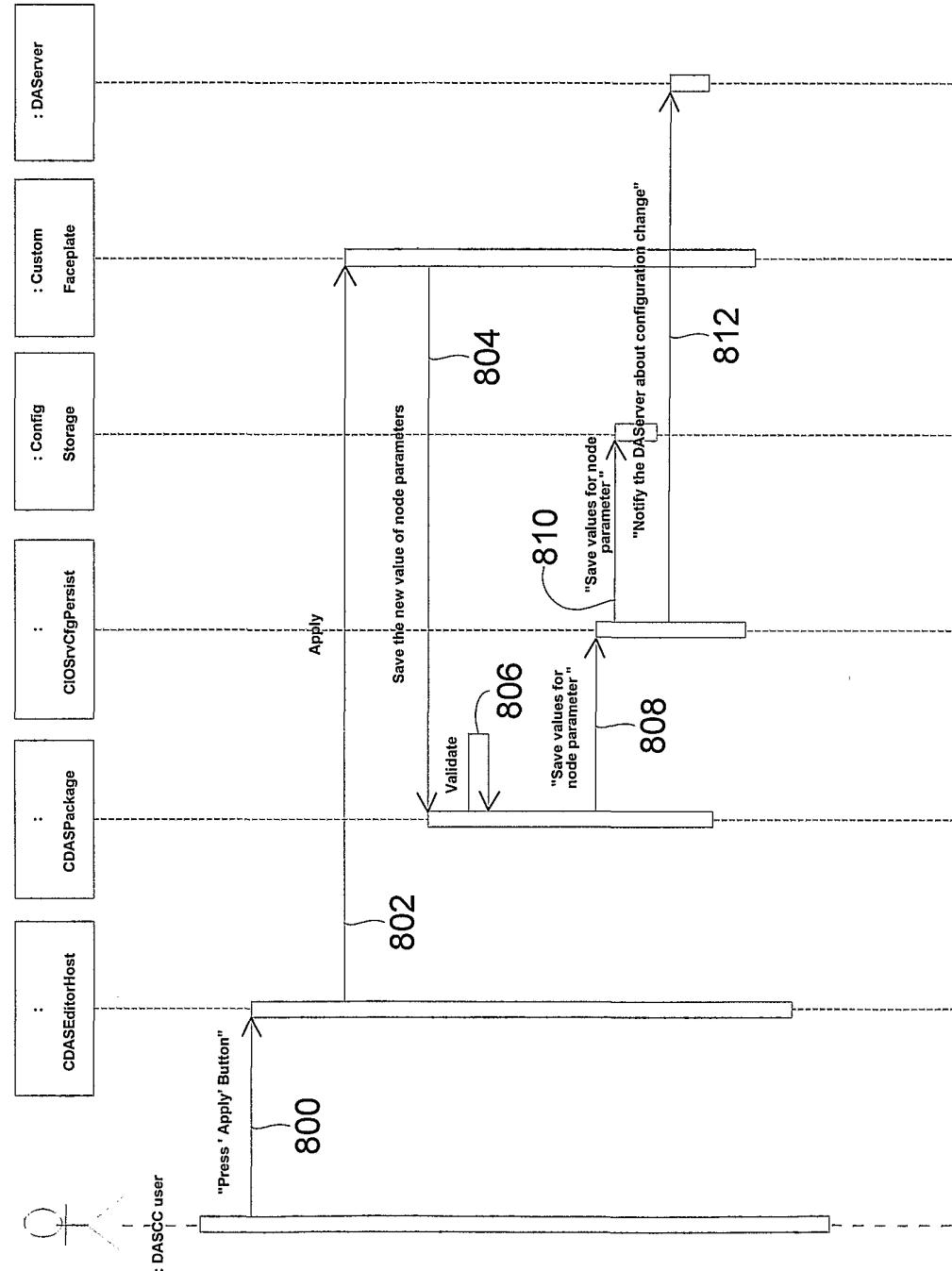
**FIG. 4****FIG. 5****FIG. 6**

**FIG. 7****Open Node Editor**

**FIG. 8****Add a new node in the configuration hierarchy**

**FIG. 9**

**Save Changes in a node editor**



## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US01/28955

## A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :H04M 3/00; G06F 17/30, 11/00, 15/16, 18/00  
US CL :709/202, 228; 707/10; 714/712; 379/265

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 709/202, 228; 707/10; 714/712; 379/265

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched  
NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,887,171 A (TADA et al.) 23 March 1999, the entire paper is relevant	1-5
Y	US 5,987,633 A (NEWMAN et al.) 16 November 1999, the entire paper is relevant	1-5
Y	US 6,049,819 A (BUCKLE et al.) 11 April 2000, the entire paper is relevant	1-5
Y,P	US 6192,364 B1 (BACLAWSKI) 20 February 2001, the entire paper is relevant	1-5
Y, P	US 6,154,778 A (KOISTINEN et al.) 28 November 2000, the entire paper is relevant	1-5

Further documents are listed in the continuation of Box C.  See patent family annex.

* Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means		
"P" document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
26 OCTOBER 2001	96 DEC 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer THUY PARDO Telephone No. (703) 305-1091

## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US01/28955

## C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6,091,811 A (CHANG et al.) 18 July 2000, the entire paper is relevant	1-5