

A SERVER AND ITS PROCESSING METHOD THEREOF

Technical Field

5 The present invention generally relates to a system with a server connected to several device controllers through a network where the server has the ability to manage, control and interact with the device controllers and the device controllers are designed to require minimum on-site configuration or maintenance, have minimum risk of data theft and are resilient to file system corruption.

Background

10 The proliferation of device controller and display screen or monitor has fuelled the deployment of such a set up in public places. One can easily find such display screen at lift lobby, train station, shopping mall and other public areas. They serve various purposes including advertising, communicating information and providing
15 entertainment to the people in the vicinity while they are waiting or passing by.

Such display screens require a device controller to process and display the information whether they are video, picture or text message. To perform such function, the device controller will require a processor, data storage and I/O ports.
20 The device controller can also have ability to connect to a server in the network or Internet.

While a server can interact with many device controllers including deploying software update and new data file to the device controllers, most of the device controllers
25 require on-site manual intervention for software upgrade installation and system process re-configuration.

Most of the device controllers are also susceptible to file system or operating system corruption, for example when they are not shut down properly, especially in the event
30 of a sudden power outage. Operating system known to have this problem include the operating systems licensed under the trademark UNIX® and the trademark WINDOWS®.

Another problem of the device controller operating unattended in a public place is that it can be stolen, resulting not only in the loss of the physical device but more importantly the loss of the data residing in the device controller.

- 5 Therefore, there is a need for a server that is able to manage device controllers such that the device controllers require very little on-site maintenance. There is also a need for a device controller that requires minimum on-site configuration or maintenance, has minimum risk of data theft and is resilient to file system corruption.

10 **Summary**

A first embodiment of the invention is a server connectable to a device controller where the server comprises a client management module, a content management module and an application management module. The server is capable of handling a plurality of requests from the device controller. The plurality of requests comprises:

15 (i) a connection request; (ii) a system instruction request for a system instruction; (iii) a software update request for a software update; (iv) a content file request for a content file; (v) a layout file request for a layout file and (vi) an executable file request for an executable file. The content file comprises at least one media file with a start date, a start time, an expiry date and an expiry time, permitting the device controller

20 to commence processing the at least one media file on the start date at the start time and stop processing the at least one media file on the expiry date at the expiry time. The layout file comprises at least one layout pattern, permitting the device controller to display the at least one media file according to the at least one layout pattern. The server is configured to:

25 (i) upon receiving the connection request and an identification from the device controller, the client management module verifies the identification and allows the connection request if the identification is valid; (ii) upon receiving the system instruction request, the server makes available the system instruction to the device controller; (iii) upon receiving the software update request, the application management module makes available the software update to the

30 device controller; (iv) upon receiving the content file request, the content management module makes available the content file to the device controller; (v) upon receiving the layout file request, the content management module makes available the layout file to the device controller; and (vi) upon receiving the

executable file request, the application management module makes available the executable file to the device controller.

5 In alternatives of the first embodiment: (i) the identification from the device controller is at least one of a device signature, a device name, a mac address and an IP address; (ii) the system instruction from the server can be a shell script; (iii) the software update can be an update to an operating system; (iv) the executable file can be an application software; or (v) the server can encrypt at least one of the system instruction, the software update, the content file, the layout file and the
10 executable file.

A second embodiment of the invention is a method of managing a device controller with a server where the server comprises a client management module, a content management module and an application management module. The server is capable
15 of handling a plurality of requests from the device controller. The plurality of requests comprises: (i) a connection request; (ii) a system instruction request for a system instruction; (iii) a software update request for a software update; (iv) a content file request for a content file; (v) a layout file request for a layout file and (vi) an executable file request for an executable file. The content file comprises at least one
20 media file with a start date, a start time, an expiry date and an expiry time, permitting the device controller to commence processing the at least one media file on the start date at the start time and stop processing the at least one media file on the expiry date at the expiry time. The layout file comprises at least one layout pattern, permitting the device controller to display the at least one media file according to the
25 at least one layout pattern. The method comprises the steps of: (i) receiving the connection request and an identification from the device controller, the client management module verifies the identification and allows the connection request if the identification is valid; (ii) receiving the system instruction request, the server makes available the system instruction to the device controller; (iii) receiving the
30 software update request, the application management module makes available the software update to the device controller; (iv) receiving the content file request, the content management module makes available the content file to the device controller; (v) receiving the layout file request, the content management module makes available the layout file to the device controller; and (vi) receiving the

executable file request, the application management module makes available the executable file to the device controller.

5 In alternatives of the second embodiment: (i) the identification from the device controller is at least one of a device signature, a device name, a mac address and an IP address; (ii) the system instruction from the server can be a shell script; (iii) the software update can be an update to an operating system; (iv) the executable file can be an application software; or (v) the method can further comprise a step of
10 encrypting at least one of the system instruction, the software update, the content file, the layout file and the executable file.

A third embodiment of the invention is a device controller connectable to a server. The device controller comprises an operating system, a data storage unit and a processing unit. The operating system includes a boot loader. The data storage unit
15 includes a non-volatile memory and a volatile memory. The device controller is capable of processing a system instruction, an operating system update, a content file, a layout file and an executable file downloadable from the server. The non-volatile memory is switchable between a read-only mode and a read-write mode. The non-volatile memory is capable of storing the operating system. The volatile
20 memory is capable of storing the system instruction, the operating system update, the content file, the layout file and the executable file. The processing unit is configured to perform the steps of: (i) setting the non-volatile memory to the read-only mode and loading the operation system with the boot loader; (ii) determining that the system instruction is available on the server; (iii) downloading the system
25 instruction from the server to the volatile memory and verifying validity of the system instruction; (iv) determining that the operating system update is available on the server; (vi) downloading and verifying validity of the operating system update; (v) upon verifying validity of the operating system update, switching the non-volatile memory from the read-only mode to the read-write mode and installing the operating
30 system update; (vii) switching the non-volatile memory from the read-write mode to the read-only mode after the installation of the operating system update is completed; (viii) running the operating system with the non-volatile memory in the read-only mode; (ix) determining that the content file is available on the server; (x) downloading the content file from the server to the volatile memory and verifying

validity of the content file; (xi) determining that the layout file is available on the server; (xii) downloading the layout file from the server to the volatile memory and verifying validity of the layout file; (xiii) determining that the executable file is available on the server; and (xiv) downloading the executable file from the server to the volatile memory and verifying validity of the executable file.

In alternatives of the third embodiment: (i) the system instruction can be a shell script; (ii) the executable file can be an application software; (iii) the data storage unit can be detachable from the device controller; (iv) the data storage unit can be an SD card; (v) the content file comprises at least one media file with a start date, a start time, an expiry date and an expiry time, permitting the device controller to commence processing the at least one media file on the start date at the start time and stop processing the at least one media file on the expiry date at the expiry time; or (vi) at least one of the system instruction, the operating system update, the content file, the layout file and the executable file can be encrypted. In addition: (i), the layout file can comprise at least one layout pattern, permitting the device controller to display the at least one media file according to the at least one layout pattern; or (ii) the device controller can further comprise an interface for displaying the at least one media file.

A fourth embodiment of the invention is a method of operating a device controller connectable to a server where the device controller comprises an operating system, a data storage unit and a processing unit. The operating system includes a boot loader. The data storage unit includes a non-volatile memory and a volatile memory. The device controller is capable of processing a system instruction, an operating system update, a content file, a layout file and an executable file downloadable from the server. The non-volatile memory is switchable between a read-only mode and a read-write mode. The non-volatile memory is capable of storing the operating system. The volatile memory is capable of storing the system instruction, the operating system update, the content file, the layout file and the executable file. The method comprises the steps of: (i) setting the non-volatile memory to the read-only mode and loading the operation system with the boot loader; (ii) determining that the system instruction is available on the server; (iii) downloading the system instruction from the server to the volatile memory and verifying validity of the system instruction;

(iv) determining that the operating system update is available on the server; (vi) downloading and verifying validity of the operating system update; (v) upon verifying validity of the operating system update, switching the non-volatile memory from the read-only mode to the read-write mode and installing the operating system update; (vii) switching the non-volatile memory from the read-write mode to the read-only mode after the installation of the operating system update is completed; (viii) running the operating system with the non-volatile memory in the read-only mode; (ix) determining that the content file is available on the server; (x) downloading the content file from the server to the volatile memory and verifying validity of the content file; (xi) determining that the layout file is available on the server; (xii) downloading the layout file from the server to the volatile memory and verifying validity of the layout file; (xiii) determining that the executable file is available on the server; and (xiv) downloading the executable file from the server to the volatile memory and verifying validity of the executable file.

In an alternative of the fourth embodiment: (i) the system instruction includes a system instruction checksum; (ii) the operating system update includes an operating system update checksum wherein the operating system update is re-downloaded from the server to the volatile memory upon finding the operating system update checksum invalid; (iii) the content file includes a content file checksum wherein the content file is re-downloaded from the server to the volatile memory upon finding the content file checksum invalid; (iv) the layout file includes a layout file checksum wherein the layout file is re-downloaded from the server to the volatile memory upon finding the layout file checksum invalid; and (v) the executable file includes an executable file checksum wherein the executable file is re-downloaded from the server to the volatile memory upon finding the executable file checksum invalid.

In other alternatives of the fourth embodiment: (i) the content file further comprises at least one media file with a start date, a start time, an expiry date and an expiry time; (ii) at least one of the system instruction, the operating system update, the content file, the layout file and the executable file is an encrypted file and the method further comprises the steps of detecting the encrypted file, locating an encryption signature and calling a decryption function to decrypt the encrypted file; or (iii) the executable file is an application software. In addition, the method can further

comprise the steps of: (i) processing the at least one media file with the executable file by the processing unit commencing on the start date at the start time and stop processing by the processing unit the at least one media file on the expiry date at the expiry time; and (ii) displaying the at least one media file according to the least one layout pattern.

Brief Description of the Drawings

Embodiments of the invention will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 illustrates a server connectable to multiple device controllers through a network or internet in accordance with an embodiment of the invention.

Figure 2 illustrates a series of interaction between a server and a device controller in accordance with an embodiment of the invention.

Figure 3 illustrates the components inside a device controller in accordance with an embodiment of the invention.

Detailed Description

In the present disclosure, depiction of a given element or consideration or use of a particular element number in a particular figure or a reference thereto in corresponding descriptive material can encompass the same, an equivalent, or an analogous element or element number identified in another figure or descriptive material associated therewith. While aspects of the present disclosure will be described in conjunction with the embodiments provided herein, it will be understood that they are not intended to limit the present disclosure to these embodiments. On the contrary, the present disclosure is intended to cover alternatives, modifications and equivalents to the embodiments described herein, which are included within the scope of the present disclosure as defined by the appended claims. Furthermore, in the following detailed description, specific details are set forth in order to provide a thorough understanding of the present disclosure. However, it will be recognized by an individual having ordinary skill in the art, i.e. a skilled person, that the present

disclosure may be practiced without specific details, and/or with multiple details arising from combinations of aspects of particular embodiments. In a number of instances, well-known systems, methods, procedures, and components have not been described in detail as not to unnecessarily obscure aspects of the
5 embodiments of the present disclosure.

Figure 1 illustrates a server 100 connectable to multiple device controllers 200 through a network or internet in accordance with an embodiment of the invention. The server 100 is connected to the device controller 200 through a network or
10 internet. The server 100 is connected to more than one device controller 200. The server 100 comprises a client management module 102, a content management module 104 and an application management module 106. As illustrated, a device controller 200 can be connected to a monitor, a digital media display or a touch panel input. The device controller 200 can also be, for instance, connected to a
15 point of sale (POS) system, a warehouse operation system, a data collection input/output system and/or a digital media box.

Figure 2 illustrates a series of interactions between a server 100 and a device controller 200 in accordance with an embodiment of the invention. The server 100 is
20 capable of handling a plurality of requests from the device controller 200 such as: (i) a connection request 108; (ii) a system instruction request 110 for a system instruction 111; (iii) a software update request 112 for a software update 113; (iv) a content file request 114 for a content file 115; (v) a layout file request 116 for a layout file 117; and (vi) an executable file request 118 for an executable file 119. The server
25 100 can also handle an identification 109 from the device controller 200.

As illustrated in Figure 2, the device controller 200 is configured to send to the server 100 a plurality of requests such as: (i) a connection request 108; (ii) a system instruction request 110 for a system instruction 111; (iii) a software update request
30 112 for a software update 113; (iv) a content file request 114 for a content file 115; (v) a layout file request 116 for a layout file 117; and (vi) an executable file request 118 for an executable file 119. The device controller 200 can also send an identification 109 to the server 100.

Figure 3 illustrates the components inside a device controller 200 in accordance with an embodiment of the invention. The device controller 200 includes a data storage unit 206, a non-volatile memory 208, a volatile memory 210 and a processing unit 212. The non-volatile memory 208 includes an operating system 202 and a boot loader 204.

A first embodiment of the invention is a server connectable to a device controller 200 where the server comprises a client management module 102, a content management module 104 and an application management module 106. The server is capable of handling a plurality of requests from the device controller 200. The plurality of requests comprises: (i) a connection request 108; (ii) a system instruction request 110 for a system instruction 111; (iii) a software update request 112 for a software update 113; (iv) a content file request 114 for a content file 115; (v) a layout file request 116 for a layout file 117 and (vi) an executable file request 118 for an executable file 119. The content file 115 comprises at least one media file with a start date, a start time, an expiry date and an expiry time, permitting the device controller 200 to commence processing the at least one media file on the start date at the start time and stop processing the at least one media file on the expiry date at the expiry time. The layout file 117 comprises at least one layout pattern, permitting the device controller 200 to display the at least one media file according to the at least one layout pattern. The server 100 is configured to: (i) upon receiving the connection request 108 and an identification from the device controller 200, the client management module 102 verifies the identification and allows the connection request 108 if the identification is valid; (ii) upon receiving the system instruction request 110, the server 100 makes available the system instruction 111 to the device controller 200; (iii) upon receiving the software update request 112, the application management module 106 makes available the software update 113 to the device controller 200; (iv) upon receiving the content file request 114, the content management module 104 makes available the content file 115 to the device controller 200; (v) upon receiving the layout file request 116, the content management module 104 makes available the layout file 117 to the device controller 200; and (vi) upon receiving the executable file request 118, the application management module 106 makes available the executable file 119 to the device controller 200.

The server 100 can, for instance, run the operating systems 202 licensed under the trademark UNIX® or under the trademark WINDOWS®. After the server 100 verifies the identification of the device controller 200, further requests from the device controller 200 are initiated by the device controller 200. The server 100 can log and monitor every action by the device controller 200. The schedule of content displayed by the device controller 200 can be set by the server 100 through the information contained in the content file 115. The content file 115 can be stored in volatile memory 210. For content files 115 stored in non-volatile memory 208, after a power outage (or lost Internet connection) the content files 115 will be restored by download of the content files 115 from the server 100.

In alternatives of the first embodiment: (i) the identification from the device controller 200 is at least one of a device signature, a device name, a mac address and an IP address; (ii) the system instruction 111 from the server 100 can be a shell script; (iii) the software update 113 can be an update to an operating system 202; (iv) the executable file 119 can be an application software; or (v) the server 100 can encrypt at least one of the system instruction 111, the software update 113, the content file 115, the layout file 117 and the executable file 119.

A second embodiment of the invention is a method of managing a device controller 200 with a server 100 where the server 100 comprises a client management module 102, a content management module 104 and an application management module 106. The server 100 is capable of handling a plurality of requests from the device controller 200. The plurality of requests comprises: (i) a connection request 108; (ii) a system instruction request 110 for a system instruction 111; (iii) a software update request 112 for a software update 113; (iv) a content file request 114 for a content file 115; (v) a layout file request 116 for a layout file 117 and (vi) an executable file request 118 for an executable file 119. The content file 115 comprises at least one media file with a start date, a start time, an expiry date and an expiry time, permitting the device controller 200 to commence processing the at least one media file on the start date at the start time and stop processing the at least one media file on the expiry date at the expiry time. The layout file 117 comprises at least one layout pattern, permitting the device controller 200 to display the at least one media file

according to the at least one layout pattern. The method comprises the steps of: (i) receiving the connection request 108 and an identification from the device controller 200, the client management module 102 verifies the identification and allows the connection request 108 if the identification is valid; (ii) receiving the system instruction request 110, the server 100 makes available the system instruction 111 to the device controller 200; (iii) receiving the software update request 112, the application management module 106 makes available the software update 113 to the device controller 200; (iv) receiving the content file request 114, the content management module 104 makes available the content file 115 to the device controller 200; (v) receiving the layout file request 116, the content management module 104 makes available the layout file 117 to the device controller 200; and (vi) receiving the executable file request 118, the application management module 106 makes available the executable file 119 to the device controller 200.

In alternatives of the second embodiment: (i) the identification from the device controller 200 is at least one of a device signature, a device name, a mac address and an IP address; (ii) the system instruction 111 from the server 100 can be a shell script; (iii) the software update 113 can be an update to an operating system 202; (iv) the executable file 119 can be an application software; or (v) the method can further comprise a step of encrypting at least one of the system instruction 111, the software update 113, the content file 115, the layout file 117 and the executable file 119.

A third embodiment of the invention is a device controller 200 connectable to a server 100. The device controller 200 comprises an operating system 202, a data storage unit 206 and a processing unit 212. The operating system 202 includes a boot loader 204. The data storage unit 206 includes a non-volatile memory 208 and a volatile memory 210. The device controller 200 is capable of processing a system instruction 111, an operating system update, a content file 115, a layout file 117 and an executable file 119 downloadable from the server 100. The non-volatile memory 208 is switchable between a read-only mode and a read-write mode. The non-volatile memory 208 is capable of storing the operating system 202. The volatile memory 210 is capable of storing the system instruction 111, the operating system update, the content file 115, the layout file 117 and the executable file 119. The processing unit 212 is configured to perform the steps of: (i) setting the non-volatile

memory 208 to the read-only mode and loading the operation system with the boot loader 204; (ii) determining that the system instruction 111 is available on the server 100; (iii) downloading the system instruction 111 from the server 100 to the volatile memory 210 and verifying validity of the system instruction 111; (iv) determining that the operating system update is available on the server 100; (vi) downloading and verifying validity of the operating system update; (v) upon verifying validity of the operating system update, switching the non-volatile memory 208 from the read-only mode to the read-write mode and installing the operating system update; (vii) switching the non-volatile memory 208 from the read-write mode to the read-only mode after the installation of the operating system update is completed; (viii) running the operating system 202 with the non-volatile memory 208 in the read-only mode; (ix) determining that the content file 115 is available on the server 100; (x) downloading the content file 115 from the server 100 to the volatile memory 210 and verifying validity of the content file 115; (xi) determining that the layout file 117 is available on the server 100; (xii) downloading the layout file 117 from the server 100 to the volatile memory 210 and verifying validity of the layout file 117; (xiii) determining that the executable file 119 is available on the server 100; and (xiv) downloading the executable file 119 from the server 100 to the volatile memory 210 and verifying validity of the executable file 119.

20

The device controller 200 can be, for instance, a lower end computer, a thin client or a miniature computer sold under the trademark RASPBERRY PI®.

In alternatives of the third embodiment: (i) the system instruction 111 can be a shell script; (ii) the executable file 119 can be an application software; (iii) the data storage unit 206 can be detachable from the device controller 200; (iv) the data storage unit 206 can be an SD card; (v) the content file 115 comprises at least one media file with a start date, a start time, an expiry date and an expiry time, permitting the device controller 200 to commence processing the at least one media file on the start date at the start time and stop processing the at least one media file on the expiry date at the expiry time; or (vi) at least one of the system instruction 111, the operating system update, the content file 115, the layout file 117 and the executable file 119 can be encrypted. In addition: (i), the layout file 117 can comprise at least one layout pattern, permitting the device controller 200 to display the at least one media file

according to the at least one layout pattern; or (ii) the device controller 200 can further comprise an interface for displaying the at least one media file.

5 The shell script can, for example, be a shell script licensed under the trademark BASH™, KORNSHELL™ or C SHELL™. An example of application software is an MP4 player. The device for displaying the media file can be, for example, a computer monitor or a TV.

10 A fourth embodiment of the invention is a method of operating a device controller 200 connectable to a server 100 where the device controller 200 comprises an operating system 202, a data storage unit 206 and a processing unit 212. The operating system 202 includes a boot loader 204. The data storage unit 206 includes a non-volatile memory 208 and a volatile memory 210. The device controller 200 is capable of processing a system instruction 111, an operating system update, a
15 content file 115, a layout file 117 and an executable file 119 downloadable from the server 100. The non-volatile memory 208 is switchable between a read-only mode and a read-write mode. The non-volatile memory 208 is capable of storing the operating system 202. The volatile memory 210 is capable of storing the system instruction 111, the operating system update, the content file 115, the layout file 117
20 and the executable file 119. The method comprises the steps of: (i) setting the non-volatile memory 208 to the read-only mode and loading the operation system with the boot loader 204; (ii) determining that the system instruction 111 is available on the server 100; (iii) downloading the system instruction 111 from the server 100 to the volatile memory 210 and verifying validity of the system instruction 111; (iv)
25 determining that the operating system update is available on the server 100; (vi) downloading and verifying validity of the operating system update; (v) upon verifying validity of the operating system update, switching the non-volatile memory 208 from the read-only mode to the read-write mode and installing the operating system update; (vii) switching the non-volatile memory 208 from the read-write mode to the
30 read-only mode after the installation of the operating system update is completed; (viii) running the operating system 202 with the non-volatile memory 208 in the read-only mode; (ix) determining that the content file 115 is available on the server 100; (x) downloading the content file 115 from the server 100 to the volatile memory 210 and verifying validity of the content file 115; (xi) determining that the layout file 117 is

available on the server 100; (xii) downloading the layout file 117 from the server 100 to the volatile memory 210 and verifying validity of the layout file 117; (xiii) determining that the executable file 119 is available on the server 100; and (xiv) downloading the executable file 119 from the server 100 to the volatile memory 210 and verifying validity of the executable file 119.

In an alternative of the fourth embodiment: (i) the system instruction 111 includes a system instruction checksum; (ii) the operating system update includes an operating system update checksum wherein the operating system update is re-downloaded from the server 100 to the volatile memory 210 upon finding the operating system update checksum invalid; (iii) the content file 115 includes a content file checksum wherein the content file 115 is re-downloaded from the server 100 to the volatile memory 210 upon finding the content file checksum invalid; (iv) the layout file 117 includes a layout file checksum wherein the layout file 117 is re-downloaded from the server 100 to the volatile memory 210 upon finding the layout file checksum invalid; and (v) the executable file 119 includes an executable file checksum wherein the executable file 119 is re-downloaded from the server 100 to the volatile memory 210 upon finding the executable file checksum invalid.

In other alternatives of the fourth embodiment: (i) the content file 115 further comprises at least one media file with a start date, a start time, an expiry date and an expiry time; (ii) at least one of the system instruction 111, the operating system update, the content file 115, the layout file 117 and the executable file 119 is an encrypted file and the method further comprises the steps of detecting the encrypted file, locating an encryption signature and calling a decryption function to decrypt the encrypted file; or (iii) the executable file 119 is an application software. In addition, the method can further comprise the steps of: (i) processing the at least one media file with the executable file 119 by the processing unit 212 commencing on the start date at the start time and stop processing by the processing unit 212 the at least one media file on the expiry date at the expiry time; and (ii) displaying the at least one media file according to the least one layout pattern.

The device controller 200 can decrypt the encrypted system instruction in the form of a shell script within its shell program such that the encrypted shell script can run as if it is unencrypted in the first place.

5 In the foregoing detailed description, embodiments of the present disclosure in relation to a server 100 and a device controller 200 are described with reference to the provided figures. The description of the various embodiments herein is not intended to call out or be limited only to specific or particular representations of the present disclosure, but merely to illustrate non-limiting examples of the present
10 disclosure.

The present disclosure serves to address at least some of the mentioned problems and issues associated with the prior art. Although only some embodiments of the present disclosure are disclosed herein, it will be apparent to a person having
15 ordinary skill in the art in view of the present disclosure that a variety of changes and/or modifications can be made to the disclosed embodiments without departing from the scope of the present disclosure. The scope of the present disclosure as well as the scope of the following claims is not limited to embodiments described herein.

20

Claims

1. A server connectable to a device controller:

(a) the server comprising a client management module, a content management module and an application management module, wherein the server is capable of
5 handling a plurality of requests from the device controller, the plurality of requests comprising:

(i) a connection request;

(ii) a system instruction request for a system instruction;

(iii) a software update request for a software update;

10 (iv) a content file request for a content file, wherein the content file comprising at least one media file with a start date, a start time, an expiry date and an expiry time, permitting the device controller to commence processing the at least one media file on the start date at the start time and stop processing the at least one media file on the expiry date at the expiry time;

15 (v) a layout file request for a layout file, wherein the layout file comprising at least one layout pattern, permitting the device controller to display the at least one media file according to the at least one layout pattern; and

(vi) an executable file request for an executable file;

(b) the server configured to:

20 (i) upon receiving the connection request and an identification from the device controller, the client management module verifies the identification and allows the connection request if the identification is valid;

(ii) upon receiving the system instruction request, the server makes available the system instruction to the device controller;

25 (iii) upon receiving the software update request, the application management module makes available the software update to the device controller;

(iv) upon receiving the content file request, the content management module makes available the content file to the device controller;

30 (v) upon receiving the layout file request, the content management module makes available the layout file to the device controller; and

(vi) upon receiving the executable file request, the application management module makes available the executable file to the device controller.

2. The server of claim 1, wherein the identification is at least one of a device signature, a device name, a mac address and an IP address.

3. The server of claim 1, wherein the system instruction is a shell script.

5

4. The server of claim 1, wherein the software update is an update to an operating system.

5. The server of claim 1, wherein the executable file is an application software.

10

6. The server of claim 1, wherein the server encrypts at least one of the system instruction, the software update, the content file, the layout file and the executable file.

15 7. A method of managing a device controller with a server,

(a) the server comprising a client management module, a content management module and an application management module, wherein the server is capable of handling a plurality of requests from the device controller, the plurality of requests comprising:

20

(i) a connection request;

(ii) a system instruction request for a system instruction;

(iii) a software update request for a software update;

(iv) a content file request for a content file, wherein the content file comprising at least one media file with a start date, a start time, an expiry date and an expiry time, permitting the device controller to commence processing the at least one media file on the start date at the start time and stop processing the at least one media file on the expiry date at the expiry time;

25

(v) a layout file request for a layout file, wherein the layout file comprising at least one layout pattern, permitting the device controller to display the at least one media file according to the at least one layout pattern; and

30

(vi) an executable file request for an executable file;

(b) the method comprising the steps of:

- (i) receiving the connection request and an identification from the device controller, the client management module verifies the identification and allows the connection request if the identification is valid;
- (ii) receiving the system instruction request, the server makes available the system instruction to the device controller;
- (iii) receiving the software update request, the application management module makes available the software update to the device controller;
- (iv) receiving the content file request, the content management module makes available the content file to the device controller;
- (v) receiving the layout file request, the content management module makes available the layout file to the device controller; and
- (vi) receiving the executable file request, the application management module makes available the executable file to the device controller.
8. The method of claim 7, wherein the identification is at least one of a device signature, a device name, a mac address and an IP address.
9. The method of claim 7, wherein the system instruction is a shell script.
10. The method of claim 7, wherein the software update is an update to an operating system.
11. The method of claim 7, wherein the executable file is an application software.
12. The method of claim 7, further comprising a step of encrypting at least one of the system instruction, the software update, the content file, the layout file and the executable file.