

APPARATUS AND METHOD FOR MANAGING DIGITAL ASSETS

The invention relates to an apparatus and a method for managing digital assets. The invention also relates to an apparatus and method for querying a remote apparatus for
5 an update to a toolkit having a plurality of files comprising rich content.

Resource toolkits are collections of files, often stored on a Compact Disk (CD) which are given away by business (or other) entities to provide information concerning a topic. For instance, a business might issue a resource toolkit concerning a new
10 product it is issuing or has issued. Creating such resource toolkits is a labour-intensive activity, found, generally, to be not easily repeatable. An information CD once produced cannot be customized for individual customers/prospects or specific needs or vertical-targeted marketing. Traditionally companies will engage a service consultancy to create the resource toolkits which also make the creation and issuance
15 of these to be a costly practice. These resource toolkits once created cannot be easily updated and the content cannot be automatically updated to reflect any changes made to it.

Document management tools used in the creation of such resource toolkits broadly
20 fall into three categories:

Firstly, desktop or client/server based document management tools are heavily proprietary, and expensive to install and maintain. These tools are highly suited for pure document management and organisation. However, producing resource toolkits
25 for various customized needs are usually beyond its capabilities. In addition, client-server interactions are rendered inadequate in a distributed workforce environment spread across vast geographies.

Secondly, Web-based document management tools have similar functionality as
30 desktop based tools, but over a web browser interface. Again the ability to compile the content for a particular customized ad-hoc need is extremely limited, if available at all.

Thirdly, Web-based Content Management Systems (CMS) and Publishing tools handle publishing and content management reasonably well, but the ability to draw on various media and give the user the flexibility and freedom to create customised versions of resource kits is extremely limited, if available at all. CMSs also tend to be content-specific and not asset-specific.

The invention is defined in the independent claims. Some optional features of the invention are defined in the dependent claims.

An apparatus and method for managing digital assets as defined in the claims is able to combine multiple asset types (documents, multimedia content) and make available for publication only selected content. Therefore, a system is readily available in which customised packaging of any type of file or digital asset which allows a user to combine documents with multimedia content e.g. video case-studies to put together a comprehensive resource kit. Traditional document management, content management systems and publishers are unable to provide such functionality and flexibility. These current technologies have management or publishing capabilities but not both, nor do they offer management and publishing capabilities across different types of files or assets. Another significant advantage provided by the claimed apparatus for managing digital assets is that such apparatus is asset specific, rather than content-specific as in the existing technologies discussed above.

An apparatus and method for querying a remote apparatus for an update to a toolkit having a plurality of files comprising rich content allows published content – e.g. in the form of a resource toolkit CD discussed above – to be updated after its issuance. Such an innovation can provide potentially huge advantages in terms of cost savings to toolkit suppliers. Toolkits can be created “on-the-fly” with the author selecting only those files needed for each resource kit thereby to create easily the package making it easy to use and accessible for creating up-to-date content for business use. Updates can be notified and distributed to end users on an ad hoc basis.

The invention will now be described, by way of example only and with reference to the accompanying figures in which:

Figure 1 is a block diagram illustrating an architecture for a first apparatus for managing digital assets;

5 Figure 2 is a block diagram illustrating an architecture for a second apparatus for managing digital assets;

Figure 3 is a flow diagram illustrating a process flow for the uploading of a digital asset to the apparatus of any one of Figures 1 and 2;

10 Figure 4 is a sequence diagram illustrating a process for the uploading of a digital asset to the apparatus of any one of Figures 1 and 2;

Figure 5 is a sample screen view of a repository generated for display by the apparatus of any one of Figure 2 for viewing on a user display;

15 Figure 6 is a series of sample screen views generated for display by the apparatus of any one of Figure 2 for viewing on a user display during a file upload process;

Figure 7 is a block diagram illustrating an architecture for a third apparatus for managing digital assets;

20 Figure 8 is a series of sample screen views generated for display by the apparatus of any one of Figures 1, 2 and 7 for viewing on a user display during a publication process;

Figure 9 is a flow diagram illustrating a process flow for the publication module of the apparatus of Figure 7;

25 Figure 10 is a sample screen view generated for display by the apparatus of any one of Figures 1, 2 and 7 for viewing on a user display during a organising by tag operation;

Figure 11 is a sample screen view generated for display by the apparatus of any one of Figures 1, 2 and 7 for viewing on a user display during a organising by file type operation;

30 Figure 12 is a sample screen view generated for display by the apparatus of any one of Figures 1, 2 and 7 for viewing on a user display during an access control operation;

Figure 13 is a block diagram illustrating an architecture for an apparatus for querying a remote apparatus for an update to a toolkit having a plurality of files comprising rich content;

5 Figure 14 is a flow diagram illustrating a process flow for the querying of a remote apparatus for an update to a toolkit having a plurality of files comprising rich content; and

Figure 15 is a block diagram illustrating a process of an update to a toolkit having a plurality of files comprising rich content.

10 Turning first to Figure 1, an apparatus 100 for managing digital assets comprises a processor 102 operably connected with controller engine 104. Apparatus 100 also comprises a first convertor 108 and a second convertor 110. Also shown in Figure 1 is a memory 103 operably connected with microprocessor 102 and controller engine 104. Apparatus 100 is in communication with a display device 112 which is
15 configured to generate for display a repository 114 containing a representation 116 of a digital asset, as will be discussed in further detail below.

Apparatus 100 is configured to receive a digital asset 106 say via an input-output module (not shown). In one implementation, the digital asset is received into memory
20 103 which may comprise of a storage-type memory for storing digital assets and/or a volatile memory such as RAM. In one implementation, the digital asset is retrieved at apparatus 100 by controller engine 104 before being stored in memory 104. The controller engine 104 is configured, under control of the processor 102, to control conversion of a digital asset 106 to a rich content format by first convertor 108. The
25 converter engine used for conversion depends on the format of the digital asset 106. First convertor engine 108 is configured to convert digital assets having a video component (V) to rich content format (RCV). Controller engine 104 is also configured to control conversion of a digital asset 106 to rich content format by second convertor engine 110. Second converter engine 110 is configured to convert a
30 digital asset not having a video component (NV) to rich content format (RCNV).

In various implementations of operation of apparatus 100, the controller engine 104 simply instructs the transmission of the digital asset to convertors 108,110 and the conversion of the digital asset 106 thereat to rich content format. Alternatively, the digital asset 106 may be received at controller engine 104 (say, from memory 103) before forwarding on to the either of the convertors 108,110 for conversion. After the assets (V, NV) have been converted, then controller engine receives the converted assets (RCV, RCNV) for further operation.

Controller engine 104 is also configured to generate, for display on display 112, repository 114 containing a representation 116 of a converted digital asset, having been converted to rich content format (either RCV or RCNV) by the first convertor engine 108 or the second convertor engine 110. In this respect, the convertor engine 104 may comprise – or operate with – a display generator (not shown) to generate the display of the repository 114 for display on display 112. The display generator may be located at the user computer (not shown) associated with display 112 or at the apparatus 100. When located at display 112, controller engine 104 transmits data for rendering of the repository 114 on display 116 under control of the display generator.

Concerning convertors 108,110, a convertor is provided for conversion of digital assets having a video component (convertor 108) and digital assets not having a video component (second convertor 110). The file type can easily be determined from a file extension (e.g. .doc, .xls, .mov extensions) of the file name of digital asset 106 and a direct one-to-one mapping determines which of convertor engines 108,110 is to be used for a particular digital asset.

If the file extension of a particular digital asset allows a determination that the digital asset is in a rich content format – e.g. a .swf or .flv file in instances where the rich content format is a Flash™ format – or a rich content-friendly format, such as .png or .jpg file format, the digital asset passes (not shown), under control of the controller engine 104, without conversion from apparatus 100 for display on display 112.

In the example of Figure 1, converters 108,110 are shown as an integral part of apparatus 100. In alternative arrangements (not illustrated), one or both of converters 108,110 may be separate entities in communication with apparatus 100/controller engine 104. In such an arrangement, controller engine 104 transmits (or controls transmission of) a digital asset for conversion to the converters. Additionally, the controller engine 104 receives and forwards converted digital assets from either or both of converters 108,110 for forwarding on for display at display 112.

A user of display device 112 has an option of viewing or manipulating the converted digital asset viewable in the repository 114 containing the representation 116 of the converted digital asset, as will be discussed further below.

Looking next at Figure 2, a second apparatus for managing digital assets is illustrated. Essentially, the apparatus 200 corresponds with apparatus 100 of Figure 1. Apparatus 200 comprises a processor 202, optionally, a memory 203, controller engine 204 and first and second converters 208, 210. Components 202, 203, 204, 208 and 210 are configured to implement functionality described for the corresponding components of apparatus 100 of Figure 1. Again, converters 208, 210 may comprise an integral part of apparatus 200 or be separate entities in communication with apparatus 200.

Apparatus 200 is in communication with storage memory 211 which may be, for example, a cloud storage memory.

The apparatus 200 is in communication with first and second user devices 212, 252. First user device 212 comprises a display 213 for displaying repository 214 containing a representation of a converted digital asset 216 and an upload manager 218 described in greater detail below. Device 212 further comprises memory 220 for storing digital asset(s) 222. Similarly, second user device 252 comprises display 253, repository 254 containing a representation 256 of a converted digital asset 256 and upload manager 258. A user may store digital assets 206, 222 in memories 220, 260 for use locally on respective user equipment 212, 252 and/or for uploading to apparatus 200.

A second user at device 252 is able to upload to apparatus 200 a digital asset 206. Any suitable means of communication may be used for the uploading of the digital asset 206, such as over the internet, or, if appropriate (i.e. within range) short-range communications protocols such as Bluetooth™. Suffice to say, for now, that digital asset 206 is received at apparatus 200 in, say, memory 203 via an input/output module (not shown). Controller engine 204 determines the file type of digital asset 206 and if digital asset 206 has a video component (V) controller engine 204 controls conversion of the digital asset having the video component with first convertor engine 208. First convertor engine 208 outputs the digital asset in the format whereby the digital asset is converted to rich content format (RCV). If controller engine 204 determines the digital asset 206 does not have a video component (NV) controller engine 204 forwards the digital asset not having a video component to second convertor engine 210 for conversion to rich content format (RCNV). Again, the determination of convertor engine to be used for conversion of a particular digital asset 206 may be made by, say, recognising the extension of the file name of digital asset 206. Apparatus 200, under control of controller engine 204, then forwards the converted digital asset in either RCV or RCNV format to memory 211 for storage therein. Further, apparatus 200 transmits to memory 211 the digital asset in original format (either V format or NV format) for storage therein. Thus, the apparatus 200 receives, under control of the controller engine 204, a digital asset 206 in an original format (V, NV) from a user of device 252 and stores the converted digital asset (RCV or RCNV) in memory 211 in rich content format. Additionally, the apparatus 200, under control of the controller engine 204, stores the converted digital asset in memory 211 in original format V/NV.

As with the apparatus 100 of Figure 1, apparatus 200 is configured to generate, for display, on a display 213 of user device 212, a repository 214 containing a representation 216 of a converted digital asset. As can be seen from Figure 2, convertor digital asset has been converted to rich content format (RCV or RCNV) by the first convertor 208 or the second convertor engine 210. User manipulation and use of the representation 216 of the digital asset will be discussed in further detail below, but controller engine 204 is set up to receive a digital asset in either converted

form or original format from memory 211 for transmission to user device 212 also discussed in further detail below, particularly with reference to Figure 5.

As with the apparatus 100 of Figure 1, if the file extension of a particular digital asset allows a determination that the digital asset is in a rich content format – e.g. a .swf or .flv file in instances where the rich content format is a Flash™ format – or a rich content-friendly format, such as .png or .jpg file format, the digital asset passes (not shown), under control of the controller engine 204, without conversion from apparatus 100 for display on display 112 and/or for storage in memory 211.

Figure 3 illustrates a process flow for an upload of a file to the apparatus of Figure 2 by user 2 of device 252. Of course, it will be appreciated that user 1 of device 212 in Figure 2 will also be able to upload digital asset 222 stored in memory 220 of device 212. The process 300 starts at 302. User 2 selects a digital asset 206 from memory 260 of device 252 for upload at step 304. At step 308, device 252 sends the digital asset to upload manager 258 for uploading to apparatus 200. The upload manager 258 queues the digital asset (or assets) for upload at step 308, described in greater detail with respect to Figure 6. Device 252 then uploads digital asset 206 to apparatus 200 at step 310. As illustrated in Figure 2, apparatus 200 sends the original format digital asset to memory 211 at step 312 for storage thereat. Before doing so, apparatus 200 checks with memory 211 to determine whether the digital asset is already stored in memory 211. If it is present, then the asset is still sent for conversion (discussed below) but saved as the latest version of an existing file. If the asset is not present in memory 211, then the asset is saved in memory 211 as a new file.

Additionally, apparatus 200 checks at step 314 for the file format of digital asset 206. As noted above, this is used to determine which of convertor 208 or convertor 210 the digital asset is forwarded to for conversion to rich content format. At step 316, the digital asset is converted to rich content format with the appropriate one of convertors 208, 210. At step 318, the converted digital asset (RCV, RCNV) is sent to memory 211 for storage thereat.

At step 320, the memory confirms to apparatus 200 that the transfer to and saving at the memory 211 of the converted digital asset is complete. At step 322, controller engine 204 updates the repository 214 for viewing at user device 212 to reflect the converted digital asset has been updated in memory. The process ends at 326, but the user has an option, in parallel, of publishing the converted digital asset at step 324 which will be discussed in greater detail below with reference to Figure 8.

Figure 4 is a sequence diagram illustrating a process flow for the uploading of a digital asset to the apparatus of Figure 2. Upload manager 258 displayed on display 253 of user device 252 queues a file for upload at 402. The file is received at apparatus 200. When the digital asset is stored in memory 211 and this is reflected in the repository 214, repository 214 notifies the controller engine 204 that upload is complete at sequence step 406. Controller engine 204 sends the file for conversion to the appropriate file convertor 208, 210 at 408. At 410, the file convertor engine 208, 210 confirms conversion is complete. At 412 the converted digital asset is stored in memory 211 and the memory 211 confirms this is complete at step 414. Additionally, controller engine 204 sends the original format digital asset to memory 211 for storage and updates repository 214 to reflect the updating of the digital asset in memory

A similar process for uploading of a new version of an existing file commences at sequence step 420 with upload manager 258 uploading a new version of an existing file. The controller engine 204 notifies the repository 214 of the receipt and storage in memory 211 of the new version of the existing file at sequence step 422. A similar process for the conversion and storage of the converted file as already described with sequence steps 408, 410, 412 and 414 is then carried out for the new version of the existing file at sequence steps 426, 428, 430 and 432.

Figure 5 is a sample screen view of a repository 500 generated for display by the apparatus 200 of Figure 2 for viewing on user display 213 and 253. In the example of Figure 5, the view of the repository 500 is a rich internet application developed in Adobe Flex™ to present to a user the user interface rendered on the Adobe Flash™

platform. Other “rich content formats” – i.e. where the contents of the display may be changed without the user having actively to “refresh” the display – may also be used. In the example of Figure 5, repository 500 is rendered for display in a web browser on apparatus 212 but, being rendered on the Adobe Flash™ platform, it has the appearance of a desktop application. Repository 500 displays a representation 502 of the converted digital asset. In the example of Figure 5, representation 502 represents a converted digital asset which was/is a Microsoft™ Word document called “filename.doc” converted to rich content format by second convertor 210. Additional file metadata 504 such as the author’s name and modification information (e.g. when the file was created or last updated) is provided adjacent the representation 502 of the converted digital asset. A description 506 of the digital asset is given and the user has the option of clicking a hyperlink 508 to view “less info” about the digital asset. A tag field 510 is provided with a list 512 of user-created and user-editable tags for the digital asset. Tag creation is discussed in greater detail with respect to Figure 6B.

When a digital asset 206 is updated in memory 211, apparatus 200 generates, for display on the user display 213, an updated repository 500. Thus, the update of the digital asset 206 in memory 211 is reflected in the repository view 500 for the user on display 213.

Referring to panel 521 of repository 500, folders of digital assets are displayed. The apparatus can be configured to provide the panel 521a default view type, which can be customised according to user preference. One example of a customisation option which may be available to a user includes stylisation of the folder icon by selection of folder colours from a drop down menu of colour options. This allows a user to classify folders based on user preference of colour schemes. Another example which may be available to a user is that access control to the folder can be specified to a group level as well as user level for the selected folder. Alternatively or additionally, a user can decide to delete a folder and maintain the files in an uncategorised arrangement. Yet further, the user may be able to delete a folder with or without all of its contents. Other possible view types are tag view and file type view, which provide relevant filters but settings and access control are easily set in folder view.

Also illustrated are user-selectable “buttons” rendered on the user’s display in Flash™: “view” 514, “download” 516, “upload new version” 518 and “delete” 520. As suggested, if user 1 selects the “view” button 514 illustrated in repository 500 displayed on display 213 of user device 212, apparatus 200 generates, under control of controller engine 204 and/or processor 202, the converted digital asset (RCV, RCNV) in rich content format for display on display 213. The text of document “filename.doc” is then presented to user 1 in repository 500 (not shown) and the user can view and scroll through the content of the converted version of this document in the Adobe Flash™ platform of repository 500.

Turning to the remaining buttons viewed in Figure 5, if the user selects “download” button 516 the controller engine 204 of apparatus 200 retrieves from memory 211, for sending to a user, the converted digital asset in original format (in this example Microsoft™ Word format). Apparatus 200 sends this to user 1 at device 212 for saving to memory 220 and/or display/editing on display 213 with the Microsoft™ Word (or other) application.

If the user selects the “upload new version” button 518 the user can upload to the repository 500 a new version of “filename.doc” to apparatus 200 and memory 211 using the process of, for example, Figures 3 and 4.

The user can remove the converted digital asset from the repository upon selection of “delete” button 520. Optionally, the apparatus can be configured so that selection of “delete” button 520 (or another button) removes the converted digital asset and/or the digital asset in original format from memory 211.

Also illustrated in Figure 5 is a second representation 522 of a converted digital asset. Representation 522 is a representation of a Flash file “filename.swf” converted to rich content format by first convertor engine 208. Thus, the repository 500 is able to combine multiple asset types (documents, multimedia content) for a user to view all file types in a single repository. The converted digital asset represented by

representation 522 can be manipulated in a similar manner as discussed above with respect to representation 502 of Word document “filename.doc” except if the user selects to view “filename.swf”, repository 500 opens the Flash file and plays it for the user to view in the repository 500 Adobe Flash™ platform.

5

Figure 6 illustrates a series of sample screen views presented to user 2 of device 252 on display 253. The view of repository 500 is the “upload new version” file allowing user 2 to upload a new version of “filename.doc” or any other digital asset within repository 500. Repository 500 is, essentially, similar to the repository 500 of Figure 5. User 2 selects button 518 to upload a new version of “filename.doc” and window 550 opens in repository 500 as a Flash animation to allow user 2 to upload a newer version of “filename.doc”. A browse button 552 is provided for user to browse device 252 (e.g. browsing memory 260 of device 252) for a digital asset 206 to upload upon selection of button 554. Checking of checkbox 556 allows the user to retain the earlier version of “filename.doc” in the repository 500 and in memory 211. A user may select a previous version from the list of versions in a drop down list 519.

Figure 6B presents a view 560 of upload manager 218, 258 in the repository 500 generated for display on display 213 or display 253 by apparatus 200 as described above. The upload manager screen 560 is a screen which allows a user to upload one or more files to apparatus 200 for viewing in repository 500. The screen lists all files being uploaded to apparatus 200 by user. As files are added to the list, they are automatically queued for upload. For instance, and as viewed in Figure 6B, a column 562 is provided for viewing files queued for upload. Column 564 allows a user to provide a description of the file being uploaded which may be illustrated in the view of repository 500 in Figure 5 at portion 506 of the repository view.

A progress bar 566 is provided to apprise a user of the upload progress of the file to apparatus 200. Icon 568 is provided to allow a user to select should the user wish to cancel the upload. Tags for the file (viewed in Figure 5 at 510, 512) may be entered and modified in column 570 of view 560 prior to upload. The destination folder name

30

in the repository may be specified in column 572. A window 574 of destination folders which may be selected can be opened in view 560 of repository 500.

One of any number of different queue algorithms can be chosen to implement the upload manager: the algorithm can be priority queue based on parameters such as file size or date deleted or simply be uploaded on a first-come first-served basis.

Figure 7 illustrates a third apparatus 700 for managing digital assets. Apparatus 700 comprises controller engine 704, file convertor engines 708, 710 and memory 711.

The apparatus 700 is configured to display for a user a repository 714 comprising representations 706 of digital assets converted to rich content format by one of the file convertor engines 708, 710. Essentially, the functionality of the controller engine 704, file convertor engines 708, 710, repository 714 and memory 711 are as discussed with respect to Figures 1 and 2. Upload manager 718 corresponds, effectively, with upload manager 218, 258 of Figure 2 and as discussed above with respect to Figure 6. Apparatus 700 further comprises a publication module 750 and template module 752 which will be described in greater detail below. Publication module 750 also comprises a site management module (not shown) and a passcode generator module (also not shown) discussed in greater detail below with respect to Figures 8 and 9.

Version control module 754 provides version control for multiple versions of files within repository 714. Access control module 756 interacts with user table 758 to specify user access rights to the repository 714 and the digital assets represented therein.

Publication module 750 is configured, under control of a processor (not shown) to receive a first user selection of a publication template (e.g., from template module 752), to receive a second user selection of a digital asset (e.g. represented by representation 706 of digital asset in repository 714) for insertion into the publication template. Optionally, the user can customise the publication template module, which is discussed in greater detail below. Publication module 750 is also configured to insert the digital asset into the publication template in rich content format thereby to

create a publication resource as will now be discussed with reference to Figure 8. Optionally or alternatively, the publication module 750 is configured to insert the digital asset into the publication template in original format. Optionally, the publication module can allow a user to insert a digital asset in original format only if the original format is recognisable by the native platform, such as the Adobe Flash™ platform.

There are essentially four steps to publishing digital assets. The user starts to publish a digital asset from publish view 802 of repository 500 illustrated in Figure 8A.

Portion 804 of display 500 lists a number of sites 806 a user has already published and offers the user the option of customising one or more of these websites by clicking any one of hyperlinks 810. Alternatively, a user can proceed to publish a new site by clicking hyperlink 812. The first of the four stages of publication is discussed with reference to Figure 8B. Still in publish view 802 of repository 500, step 814 for

selection of files provides a list of all folders accessible to a user from repository 714. On clicking the folder name (1), the user can select either individual files from the list of files (2) or select an entire folder. The user is also presented with an option of selecting all files in the repository 500 and may click checkbox 815 for this purpose.

The user proceeds to the next step by proceeding to view 816 to choose a template for publication of the digital assets. A user can view and select a publishing template from a list provided (not shown). The apparatus presents a series of rich internet format templates (e.g. Flash™ templates) for the user to make a selection from. In step 3, illustrated in Figure 8C, the user can customise the template by, for example, specifying a banner by choosing a local file or one from the web. The user can also

add a Flash animation to the published site to, for example, the intro screen for the publication. If a user decides to use an animation from an existing file in the repository 500, the application switches view to a rich media explorer 850 in repository 500 which displays all the rich media files 820 (viewable upon selection of icon 826 in the right hand side of the repository view) and can be added to the Flash animation. The user can also choose images upon selection of icon 822 of Flash files upon selection of icon 824 for customisation of the template.

Although not explicitly illustrated herein, the user can customise the template in a number of other ways such as by customising a theme colour and font properties.

5 In step 4 of the publication process, the user is presented with a final preview 828 of the publication resource (containing one or more digital assets inserted into the selected template) before proceeding to publish. The user has the option to publish a downloadable version of the publication resource by selection of icon 830 or to publish a micro site by clicking hyperlink 832. The micro site URL 832 may be automatically generated by the publication module 750 or it may be specified by the user.

10 If the user chooses to download a version of the publication, it can be saved locally at, for example, user device 212 or 252 of Figure 2 and can be used for, for example, creation of toolkits such as the toolkits described below with respect to Figures 13 to 15. If the user chooses to publish a micro site available on the internet, the micro site can be either publically or privately published. If publically published, anyone may access the micro site via the internet. If the user chooses to publish the micro site privately, publication module 750 also comprises a passcode generator module (not shown) for generation of a passcode which can be given to authorised user to allow them to access the private micro site.

25 Figure 9 is a flow diagram illustrating a process flow for the publication module of the apparatus of Figure 7. The main publish home view of repository 500 is presented to the user at point 902. At this point the user has the option of 903 modifying the publication of an existing site or 954 of creating a new site for publication following the four step publication process described above with respect to Figure 8. If the user chooses to amend the publication of an existing published site at step 903, the first option presented to the user is the option of customising existing site options at which point the user is transferred to step 954 which will be discussed in further detail below. The user may wish to remove an existing published site at which point he is presented with the option to confirm deletion of the site at step 904. If the user does not confirm deletion of the site, the process loops back to step 903. If the user does

confirm deletion of the site, the site is removed 908 before looping back to step 903. If the customisation of the existing published site is relating to customisation of the access control of the site, the user is presented with the option to make the site public at step 906, in which case the site has no authorisation required and the user is
5 returned to step 903. If the site is to be private, the passcode generator module generates a passcode at step 910 prior to returning to step 903.

If creating a new site, the user follows step 1 (of Fig 8) to select files at 956, step 2 to view/select a template for publication at step 958, step 3 for customisation of the
10 template selected at step 960 prior to step 4 having a final preview at 974. If a user chooses to customise a template at step 960 the user has a choice of choosing a colour at 962, amending a banner for the publication resource by choosing a file from the web at 964 or choosing a file from a local machine at 966. If customising the introductory animation a determination is made as to whether the input animation is in
15 the repository 500 library at step 968 whereby a user may select an animation file from the repository library at step 970 or upload an animation from a local machine at step 972. After the final preview at step 974, the user is presented with the option of making the site public at step 978. Choosing to make the site public, the user is presented with information concerning the micro site hyperlink at step 976. The user
20 chooses to make the site private by selecting "no" at step 978, after which the passcode generator sub-module of publication module 750 generates a passcode and presents this to the user. At step 976, the user is presented with the opportunity of downloading the publication resource file or following a micro site link. The publication process flow stops at 982.

25 Figure 10 illustrates another view of repository 500. In this view, the user has activated the side pane to filter files based on the tag cloud 580. The user does this by selecting field 582 "tags". In tag cloud 584, various different tags associated with the files in the repository are illustrated. Selecting a particular tag, for example the tag
30 586 "adipicing" filters for viewing only files having that tag. The same could be said for tags 588 and 590. It will be apparent that the appearance of the various tags may be varied according to various criteria. For instance, in the example of Figure 10, tag

586 appears in larger font size than the tags 588, 590. The reason for this is the tags in the tag cloud have their font size set dependent on the relative number of occurrences of the tag for filenames in the repository. By rendering the tag 586 in a larger font size when compared with tags 588, 590 the user is immediately apprised that tag 586 has a higher number of occurrences for files within the repository than either of tags 588, 590. The user may also choose to filter or organise files by file type by selection of the “file type” icon 592. This is illustrated in Figure 11 where another view of repository 500 is given.

- 10 In the view of repository 500 of Figure 11, a file type cloud 581 opens allowing a user to filter or organise dependent upon file type (file type of the digital assets in original format) by selection of any one of icons 594A, 594B ... 594F.

- 15 Figure 12 is a sample screen view generated for display by the apparatus of Figures 1, 2 and 7. In this view of repository 500, side pane 580 opens up to illustrate a list of group names 598. The user can set access control rights to members of the group with opening of the access control pop-up 596.

- 20 Turning now to Figure 13, an apparatus for querying a remote apparatus for an update to a toolkit having a plurality of files comprising rich content will now be described. Apparatus 1300 comprises microprocessor 1302, display 1304 operable to display a graphic user interface 1306 and memory 1308. Memory 1308 may comprise a storage-type memory 1310 for storing a toolkit such as that published using the techniques described with reference to Figures 1 to 12. Memory 1312 comprises of a volatile-type memory such as RAM for the running of routines for controlling apparatus 1300.

- 30 Memory 1310 stores the toolkit which is loaded from an external source 1314. In one example, the external source is a CD-ROM provided by a supplier of a resource toolkit for loading into a disk drive (not shown) of apparatus 1300. The toolkit is loaded from a CD-ROM and stored in memory 1310. In another example, the toolkit is downloaded from the repository view presented to a user and as illustrated in the

screen view of 8D discussed above and saved in memory 1310. The files and content of the toolkit are displayed on the display 1304 via GUI 1306.

Memory 1312, as noted above, is provided for the storing of one or more routines
5 which, when executed on the apparatus 1300 under control of processor 1302 cause apparatus 1300 to query 1316 the remote apparatus for an update to the toolkit stored in memory 1310 and loaded from the external source 1314. If apparatus 1300 determines an update to the toolkit is available from remote apparatus 1318 – stored in memory 1312 or a remote memory (not shown) – the update is downloaded 1312
10 and stored in memory 1310 of apparatus 1300. The downloading of the update is discussed in greater detail with reference to Figure 15.

Apparatus 1300 is configured to check for the update to the toolkit by comparing a
metadata parameter of a file in the toolkit stored in memory 1310 with a related
15 metadata parameter at the remote apparatus. This metadata information is of the update to the toolkit (if available) allowing the apparatus 1300 to make a determination and decision to download 1322 the update(s).

Figure 14 is a process flow diagram illustrating a process for a toolkit update in the
20 apparatus of Figure 13. The process 1400 starts at 1402 where the user of apparatus 1300 launches the toolkit at step 1404. As noted above, the user may do this by loading the toolkit from external source 1314 which may comprise either loading the toolkit from a CD-ROM or downloading it from the repository view 500 of Figure 8D. A user is prompted at step 1406 to create a local copy of the toolkit resources. If
25 the user chooses to do so the toolkit is copied to the user's device and is stored in memory 1310 at step 1408. Apparatus 1300 is configured to operate a run-time environment for managing, under control of the processor 1302 the querying of the remote apparatus 1318 for an update to the toolkit stored in memory 1310. One exemplary run-time environment which may be used for this purpose is the Adobe
30 AIR™ cross-platform run-time environment for building rich internet applications that can be deployed as a desktop application on apparatus 1300.

At step 1410, apparatus 1300, under control of the processor and the routine stored in memory 1312 determines whether the RTE is installed in apparatus 1300. If the RTE is not installed, the user is prompted at step 1412 to download and install a suitable RTE.

5

At step 1414, apparatus 1300 determines whether any updates are available to the toolkit from the remote apparatus. If query 1316 in Figure 13 yields that updates are available, at step 1414 the user is prompted to download and save the updated content at step 1416. The newly-downloaded content is merged with the non-superseded content originally loaded from the external source as step 1418. Apparatus 1300 is also configured to determine whether the remote apparatus has an update to the framework for the toolkit. The contents of the resource kit e.g. the digital assets and the publication resource are reviewed and generated from the framework for the toolkit. In the present example, the toolkit framework is named ORKA™ (Online Resource Kit Application). If the framework application has any bugs like, for example, all images are displayed with a fixed size that cannot be modified and if the bug is also transmitted to the resource kit, then an incremental update to the framework application will also update the corresponding resource kit, hence providing the fix for the bug. If updates to the platform are available then the user is prompted at step 1422 to download and save platform updates. At step 1424, the toolkit is run comprising updates to the files downloaded from the remote apparatus. The process ends at step 1426.

10

15

20

25

In one implementation, the queries for toolkit updates and framework updates are made in a single call.

Figure 15 provides an illustration of the determination of the updated content to be downloaded and the download update indicated by arrow 1322 in Figure 13 at process steps 1416 and 1418 of Figure 14. Memory 1310 of apparatus 1300 comprises files File A 1502, File B 1504, File C 1506 and File D 1508. These are shown to be stored in the memory 1312 of remote apparatus 1318 as File A 1502', File B 1504', File C 1506' and File D 1508'. When apparatus 1300 queries remote apparatus 1318 for an

30

update, the apparatus determines that updates are available to Files A and C and a new File E is available. File A* 1502A, File C* 1506A and File E* 1510 are stored in memory 1312 of remote apparatus 1318. Apparatus 1300 is operable to determine the updates by comparing a metadata parameter of a file in the toolkit with a related
5 metadata parameter at a remote apparatus. For instance, File A 1502 stored in memory 1302 is known to have been last saved at a certain date and time, where the date and/or time comprise the metadata parameter. A comparison of the corresponding or related metadata parameters for File A 1502' and File A* 1502A determines that a newer version of File A (File A* 1502A) is available for
10 downloading. Once the user is prompted at step 1416 of Figure 14 to download the updated content, the user downloads the updated content 1322 and stores it in memory 1310 of apparatus 1300.

It will be appreciated that the invention has been described by way of example only
15 and that various modifications to the techniques described above may be made without departing from the spirit and scope of the invention. Features presented in relation to one aspect of the invention may be combined with features presented in relation to another aspect of the invention.

Claims

1. Apparatus for managing digital assets, the apparatus comprising:
a processor; and
5 a controller engine configured, under control of the processor:
to control conversion of a digital asset having a video component to rich
content format by a first converter engine;
to control conversion of a digital asset not having a video component to rich
content format by a second converter engine; and
10 to generate, for display on a display, a repository containing a representation
of a converted digital asset, the converted digital asset having been converted to rich
content format by the first converter engine or the second converter engine.
2. The apparatus of claim 1 configured, under control of the controller engine, to
15 generate, for display on the display, the converted digital asset in rich content format.
3. The apparatus of claim 1 or claim 2 configured to receive, under control of the
controller engine, a digital asset in an original format from a user and to store the
converted digital asset in memory in rich content format.
20
4. The apparatus of claim 3 configured, under control of the controller engine, to
store the digital asset in memory in original format.
5. The apparatus of claim 3 configured, under control of the controller engine, to
25 retrieve, for sending to a user, the converted digital asset in original format.
6. The apparatus of any of claims 3 to 5 configured, under control of the
controller engine, to update the repository, for display on the display, when a
converted digital asset is updated in memory.
30
7. The apparatus of any preceding claim further comprising a publication module
configured, under control of the processor:

to receive a first user selection of a publication template;
to receive a second user selection of a digital asset for insertion into the
publication template; and
to insert the digital asset into the publication template in original format or
rich content format thereby to create a publication resource.

8. The apparatus of claim 7, wherein the publication module is configured, under control of the processor, to receive the first user selection of the publication template as a rich internet format template.

9. Apparatus for querying a remote apparatus for an update to a toolkit having a plurality of files comprising rich content, the apparatus comprising:

a processor;
memory for storing the toolkit, the toolkit loaded from an external source;
memory for storing one or more routines which, when executed on the
apparatus under control of the processor, cause the apparatus to query the remote
apparatus for an update to the toolkit.

10. Apparatus according to claim 9, the apparatus being configured, under control of the processor, to query the remote apparatus for an update to a framework for the toolkit.

11. Apparatus according to claim 9 or claim 10, the apparatus being configured to operate a run time environment for managing, under control of the processor, querying the remote apparatus for an update to the toolkit.

12. Apparatus according to any of claims 9 to 11, the apparatus being configured to compare a metadata parameter of a file in the toolkit with a related metadata parameter at the remote apparatus.

13. A method for managing a digital asset in an apparatus comprising a processor, the method comprising controlling a processor to control a controller engine:

to control conversion of a digital asset to a converted digital asset in rich content format, wherein if the digital asset has a video component the controller engine controls conversion of the digital asset with a first converter engine and if the digital asset does not have a video component the controller engine controls
5 conversion of the digital asset with a second converter engine; and

to generate, for display on a display, a repository containing a representation of the converted digital asset.

14. A method for querying a remote apparatus for an update to a toolkit having a
10 plurality of files comprising rich content, the method comprising:
storing the toolkit in memory, the toolkit loaded from an external source; and
controlling a processor to execute one or more routines to cause the apparatus to query the remote apparatus for an update to the toolkit.

15. A machine readable medium having stored thereon machine readable instructions for executing a method for managing a digital asset in a machine comprising a processor, the method comprising controlling a processor of the machine to control a controller engine:

to control conversion of a digital asset to a converted digital asset in rich
20 content format, wherein if the digital asset has a video component the controller engine controls conversion of the digital asset with a first converter engine and if the digital asset does not have a video component the controller engine controls conversion of the digital asset with a second converter engine; and

to generate, for display on a display, a repository containing a representation
25 of the converted digital asset

16. A machine readable medium having stored thereon:
a toolkit having a plurality of files comprising rich content; and
machine readable instructions for controlling a processor of a machine to
30 cause the machine to query a remote apparatus for an update to the toolkit.