

Additionally, ML elements can also be used to represent implicit user setting and preferences. The implicit user setting and preferences can be used to preserve the different document option settings and preferences set by the user in editing the document. The implicit user setting and preferences are generally intended to restore the application environment to the same state in terms of user interface behavior (and other aspects) when the document is reopened. For example, the restored application environment may include the view in which the document was last edited, whether XML validation was enabled, and the like.

Some of the implicit user setting and preferences properties can be saved inside the “docPr” element (discussed with reference to FIG. 8, below) and others of the implicit user setting and preferences can be saved inside the DocumentProperties element. The values of the children of DocumentProperties can be arbitrary strings that are entirely user-defined.

FIG. 7 illustrates an exemplary document properties element having user-defined arbitrary strings, in accordance with aspects of the present invention. The figure illustrates listing 700 having a document properties element (710). Document properties element 710 is a container element, which may include arbitrary strings that can be user-defined. Document child elements 720 include elements such as “Title,” “Subject,” “Author,” and the like. The contents of the element typically include user-defined arbitrary strings such as “My Document,” “The business plan of a company I work for,” and the like.

The “docPr” element is a container element in which the children elements can be used by the application to preserve the states of the different application behaviors activated or deactivated by a user. The states of the different application behaviors are available to the user in various parts of the application user interface (e.g., such as the “Options” dialog, and the like). The states of the different application behaviors are represented by elements. Some of the children elements are arranged to accept special attributes as well as accepting other elements as children. Table 1 lists and describes various application behaviors that are represented by various elements.

TABLE 1

view	represents the view in which the document was last edited. Its “val” attribute can be used to specify the view.
zoom	specifies the zoom in which the document was last edited. Its attributes contain the actual zoom setting.
doNotEmbedSystemFonts	determines whether font descriptions are embedded in the file even if they are typically present on the system.
attachedTemplate	a pointer to the template file the document is based on.
documentProtection	represents various aspects of the document protection state.
defaultTabStop	determines the positions of the default tab stop.
characterSpacingControl	specifies different settings for the algorithm that lays out characters when the document is displayed in the application or printed.
optimizeForBrowser	used to determine for which browser the document, when saved as HTML, is supposed to be optimized.
validateAgainstSchema	determines whether the document should be validated against the attached XML schema (if any).
saveInvalidXML	determines whether the app should allow the user to save the document as XML if

TABLE 1-continued

	it does not adhere to the attached customer-defined schema.
5 ignoreMixedContent	represents the option to ignore mixed XML content for validation purposes and when saving to customer-defined schema only.
alwaysShowPlaceholderText	determines whether placeholder text is automatically generated and shown by the app for each empty customer-defined XML element.
10 doNotUnderlineInvalidXML	controls the underlines’ appearance near customer-defined schema violations.
footnotePr	complex element (with additional children) used to represent default properties of a footnote.
15 endnotePr	complex element (with additional children) used to represent default properties of an endnote.

FIG. 8 illustrates an exemplary document preservation (“docPr”) element, in accordance with aspects of the present invention. The figure illustrates listing 800 having a document preserve (810). Document preserve element 810 is a container element, which includes child elements that may include child elements that store settings for selected behaviors. Document preserve elements 820 include elements such as “view,” “zoom,” “doNotEmbedSystemFonts,” and the like. Each of the disclosed elements and attributes can be mapped to an internal word processor structure that (if present) represents a corresponding feature in the application.

FIG. 9 illustrates of a process 900 flow for representing document options, properties and backwards compatibility settings using XML, in accordance with aspects of the invention. After a start block, the process moves to block 910, at which point a document is opened for editing by a user. The selection of the file for opening may, for example, include highlighting the selected file within a file browser. The document may be, for example, a document that includes spreadsheet cells or word-processor paragraphs. The document may be stored in a proprietary format of the application process.

At block 915, the process encodes in an ML format the automatically generated properties of the electronic document. The automatically generated properties can be saved inside the “DocumentProperties” element container. The application process typically provides a dialog that allows the user to specify the properties.

Continuing at block 920, the process typically provides a dialog that allows the user to specify the custom/user-defined properties. The custom/user-defined properties can be saved as ML elements inside the “CustomDocumentProperties” container element.

At block 925, the process encodes in an ML format the backwards compatibility settings supported by an application. The ML elements used to represent these backwards compatibility settings can be saved inside of the “compat” element container

Flowing to block 930, the process encodes in an ML format the application environment properties of the opened electronic document. The “docpr” element is a container element in which the children elements can be used by the application to preserve the states of the different application behaviors activated or deactivated by a user.

At block 935, the document is saved using an ML format. Saving the “native” properties of the process in an external ML file permits other editing applications to preserve the