

## EXTENSIBLE FILE FORMAT

## TECHNICAL FIELD

The present invention relates generally to computer systems and programs. More particularly, the present invention relates to a system and method that can provide a file format compatible with previous, active, and future versions of an application program.

## BACKGROUND OF THE INVENTION

Computer application programs are generally used to perform many computer processing tasks. An application program can be used to create and modify a data file, which stores information of a particular item for the application program. For example, a document can be created using a desktop publishing type application program. The document can contain many individual elements, such as headers, text, graphics, styles, fonts, etc. The publishing type application program can be used to create a data file that stores all of the information of that particular document.

Every application program that saves files on a computer system must determine what format in which to save data. There are many ways of saving data, ranging from saving out memory images of data to writing out data in an industry standard form. Additionally, computer application programs are continually being updated and changed to provide the latest features and technology available. The updates and changes are typically provided to end users by releasing an updated version of the application program. The file format used by the updated version of the application program needs to be compatible with previous and future versions. However, the updated version typically uses a different file format than previous versions, because new features in the updated version require new data to be written to the data file. Problems can arise between two different versions of the application program, because the different file format can prevent previous versions from being able to read the updated version's files.

An "object," as used in this disclosure, refers to an entity that has characteristics and that is displayed by an application program, is part of an application program, or is part of the data stored/manipulated by the program. An example of an object displayed by an application program is a text box contained in a window of an application program into which a user can input text. The characteristics of the text box can include its color, the font of the text, and the point size of the text. An example of an object that is part of an application program is an in-memory representation of an animal, where its characteristics can include its color, number of legs, and whether it is a carnivore. This in-memory representation can be implemented as a data structure with the elements of the data structure storing the characteristics. An example of such a data structure is the C++ class data structure. The characteristics of an object are referred to as properties of the object. Each property of an object typically has a value. For example, the color property may have the value red.

An example of an object **100** is illustrated in FIG. 1. Object **100** can comprise a border **102** and text **104**. Object **100** can have six properties for border **102** and text **104**. Border **102** can have properties of border style and border size. Text **104** can have properties of font, text size, justification, and text style. The default values for border **102** and text **104** can be that the border style is solid, the border size is 4 point, the text font is Times New Roman, the text size is 20 point, the text justification is left, and the text style is non-italic.

In an application program, the default values of the objects and properties can be established during initial programming. During operation of the program, the default values can be changed and stored in the data file. For example, in a draw program that can display objects such as rectangles and triangles, a user can modify the objects' properties. Returning to the example of FIG. 1, the properties of object **100** can be changed such that the solid border becomes a dashed border and the text style becomes italic. Later versions of the program can typically read the default values and the changes for each object. However, when later versions of the program are created, the file format typically differs from that of the previous version. Accordingly, the previous version cannot read files from the later version. Using the above example from the previous paragraph, a later version could include a new feature such as a shadow for object **100**. The added feature requires new data to be written to the program's files, thereby changing its format. The previous version cannot read the files of the later version containing the new data, because the file formats are different.

One conventional solution to the problem discussed above is to limit the new features of the updated version so that the file format remains compatible with previous versions. However, that solution can prevent the best product from being distributed to the user and can limit the overall function and performance of the program.

Another conventional solution to the problem discussed above is to create an adapter program that converts the old file from the previous version to a new file compatible with the updated version. The adapter program also can convert a new file to a file compatible with the previous version. However, that solution can be inconvenient for the user and can require a large amount of memory and processing time to convert the files. Furthermore, that solution can cause the user to lose important information needed by the newer version of the program when the file is saved by a previous version.

Therefore, there is a need in the art for an improved system and method that can provide a file format compatible with previous, current, and later versions of an application program. There is also a need in the art for an extensible file format that can allow new features to be added in future versions of an application program while remaining compatible with previous versions.

## SUMMARY OF THE INVENTION

The present invention relates generally to a system and method for providing an extensible file format that can be compatible with previous, current, and future versions of an application program. The present invention is directed to an extensible file format that supports the addition of new features to future versions of the application program, while maintaining compatibility (without modification) with previous and/or current versions. Accordingly, the present invention can enable a software vendor to provide multiple version compatibility with the same files without limiting the features that can be added to future versions.

In one exemplary aspect, the present invention can comprise including file version watermarks in an original data file of an application program. The file version watermarks can be an element in the original data file that can indicate various properties of the original data file. For example, a high version watermark can be provided in the original data file to indicate the highest version of the application program used to save the file. A last version watermark can be