

TABLE 2-continued

Example Embodiment of a CellDef Element	
Attribute	Description
	7. DWORD: 4 byte unsigned long
	8. LONG: 4 byte signed long
	9. FLOAT: 4 byte float
	10. DOUBLE: 8 byte double
	11. PERCENT: 8 byte percent, 100% = 1.0
	12. GUID: 16 byte GUID
	13. MULTIDIM: 10 byte multi-dimensional UNUM
	14. TWIPS: 2 byte signed 1/20ths of a point
	15. CAL: 1 byte, MSO Calendar enumeration
	16. ANY: 9 byte any allowed result
	17. UNUM: double with units;
F	The cell default formula if no formula or value is written.
IX	The position of the column within a row.

In embodiments, a property element may also define a formula or function that may be used to calculate the value of the property represented by the element. An application that does not support the property represented by the element may nonetheless use the formula or function provided as an attribute of the property element to calculate a value for the property. By doing so, the self-describing file 100 provides the application with the ability to preserve the value of the property that the application may not support, thereby preserving content, or aspects of the content, that may be unknown to the application. In other embodiments, the property element may define an object, property, or other component of the file. The definition may include data that a less featured application (e.g., an application that does not natively support the object, property, etc.) can utilize to calculate values for the non-supported object, property, etc.

In further embodiments, the extension element may contain one or more child elements that define, objects, properties, functions, and/or formulas that may be used to calculate values for properties, objects, or other components stored in the self-describing file 100. A function element may be associated with a property, an object, or any other type of data stored in the self-describing file 100. In such embodiments, the function element may be used, similar to the function attribute of the property element, to provide functions and/or formulas that an application may not natively support. By doing so, the function element provides for the preservation of values present in the self-describing file 100 that may be unknown to an application access the self-describing file. In embodiments, the function child provides an application with the ability to preserve unknown file contents, even if the application does not otherwise support or handle the file contents.

In embodiments, in order to preserve unknown file contents, an application accessing (e.g., opening) the self-describing file 100 may parse and load any information provided by one or more function elements stored in the extension section that is part of or associated with the self-describing file 100. In such embodiments, an application may use the data from the extension section to calculate and store a value for an object or property, even if that version of the application does not otherwise natively support the object or property. The calculated value may be stored in the self-describing file, thereby allowing the application to preserve unknown file content. In embodiments, the calculated value may differ from a value calculated by a full featured application; however, the information provided in the extension section may allow the less featured application to provide a value that is compliant with an expected value (e.g., a value calculated by a full featured application). As such, the information

allows the less featured application to, at the very least, provide a meaningful value where without the information the less featured application may have returned an error or provided meaningless information.

As described herein, a self-describing file 100 may be used to prevent issues where object and/or property data not natively supported by an application is improperly maintained when the application modifies the content of the self-describing file 100. Returning to the transparency example in which the value of a transparency property is dependent upon the value of a shading property, an application that modifies the shading property of an object may nevertheless calculate a value for the transparency property using the functions, object, properties and/or other information described by the extension element and/or its child elements provided in the extension section 102 of the self-describing file 100, even if the application does not natively support calculation of the transparency property. Moreover, native objects and properties that have values dependent on an extended object, property, relationship, etc. (e.g., the transparency property from the examples provided) may also be calculated by the less featured application even though the extended property, object, relationship etc. may not be known to the lesser featured application.

FIG. 2 is an illustration of a flow chart representing an embodiment of a method 200 for creating a self-describing file. Flow begins at operation 202 where an application creates a self-describing file, such as the self-describing file 100 from FIG. 1. In embodiments, an application may create the self-describing file in response to a command issued by a user or by another application. In embodiments, the self-describing file may be a diagram file created by a diagramming application, a document file created by a word processing application, a spreadsheet file created by a spreadsheet application, or any other type of file. In further embodiments, information provided by an extension element may also be used in conjunction with a data stream. As such, a less featured application may use self-describing information (provided in an extension element or otherwise) to properly manipulate data in a data stream.

During the creation operation 202, objects, properties, and/or components may be included in the self-describing file. In embodiments, the objects, properties, and or components may be written to the self-describing file. At operation 204 a first portion of data is written to the self-describing file. In embodiments, the first portion of data may be a property representing a characteristic of an object. For example, if the self-describing file is created by a diagramming application, it may contain an object representing a shape. The first portion of data may be a characteristic of the shape, such as the shape's color, size, outlining, shading, etc. As another example, if the self-describing file is created by a word processing application, an object representing a string of text may be written to the self-describing file. In such an embodiment, the first portion of data may be a characteristic of the text such as font face, font size, color, indentation, etc. As such, in embodiments, a first portion of data may represent a characteristic of any object in the self-describing file, any component of the self-describing file (such as a section, a template, etc.) or a characteristic of the self-describing file itself. In further embodiments, the first portion of data may be an object, a property, a formula, a relationship, a component or any other type of data that may be written to the self-describing file.

Flow continues to operation 206 where at least a second portion of data is written to the self-describing file. Similar to the first portion of data, the second portion of data may