

## DYNAMIC CONFIGURATION FILES

### BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] This invention relates to data processing systems, methods, and computer program products, and more particularly to configuration files for data processing systems, methods, and computer program products.

[0003] 2. Description of the Related Art

[0004] A given data processing system may be configured in many ways. Some of the configuration options are set by varying the hardware (e.g., number and capacity of disk drives). Other configuration parameters are set within configuration files stored by the computer.

[0005] Software programs typically use configuration files of one kind or another to define customizable properties. One very common configuration file format, used by “INI” files on the Windows operating system, divides configuration properties into sections, as follows:

```
property1=value1
```

```
[section2]
```

```
property1=value1
```

```
property2=value2
```

[0006] Configuration files are typically used because they can be edited using a simple text editor, thereby enabling end-users to easily make changes to them. Therefore, their use has become pervasive in application development. Some examples of properties that are often stored in configuration files include user-interface related information, such as colors, font types, font sizes, etc.; prerequisite software locations, versions, and related information; and debug settings.

[0007] Typical configuration files are static in nature (i.e. properties are only simple textual strings that contain no variables). This static nature of configuration files renders their use sub-optimal in some circumstances. When defining locale-specific values, multiple configuration files need to be created when a particular property needs to vary from one locale to the next. For example, situations arise when it is desired to define different font-types or font-sizes depending on the locale (e.g., a Japanese-language font versus an English-language font). Using typical configuration files in multi-language development, this could be accomplished by creating separate configuration files for each locale, or having the end-user manually update these fields to match the settings for the desired locale. Either process (separate configuration files or manual updating) are tedious and time-consuming.

[0008] Another situation where the static nature of configuration files is not adequate occurs when a property in one configuration file needs to reference a second property in the same or different configuration file. For example, many applications define properties for directories used by an application (e.g. where to put log files, where to find libraries, etc.). Often these directories are located inside the ‘install’ directory by default, but are exposed to the end-user so that these directories can be modified at a later date. Using

typical configuration files, each of these dependent properties have to be defined separately, as follows:

```
[Paths]
```

```
installDir=C:\Program Files\My Product
```

```
logDir=C:\Program Files\My Product\logs
```

```
libDir=C:\Program Files\My Product\lib
```

[0009] Unfortunately, if the install location changes (e.g., if the install directory changes from “My Product” to “Their Product”), each property containing the install location (e.g., “logs” and “lib” in the above example) must also be updated. This can be particularly tedious if these dependent properties are scattered throughout the configuration file, or even defined in different configuration files.

[0010] Accordingly, it would be desirable to have the ability to include dynamic elements in configuration files and then resolve the variables when the configuration files are run.

### SUMMARY OF THE INVENTION

[0011] The present invention adds a dynamic nature to configuration files so that various types of dynamic functions can be performed within them. Variables are utilized within the configuration files. Using the present invention, it is possible to have parameters specified in one configuration file and a formula to which the parameters will be applied in another configuration file. This allows changing of the values in the parameter file without having to modify the formula file, thereby streamlining the modification process when changes need to be made. In addition, it provides the capability for cross-referencing between configuration properties by allowing for variables in properties that refer to other properties.

### BRIEF DESCRIPTION OF THE DRAWINGS

[0012] **FIG. 1** illustrates the steps performed in accordance with the present invention.

### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0013] **FIG. 1** illustrates the steps performed in accordance with the present invention. As an initial matter, configuration files must either exist or be created that comprise configuration properties having one or more variables of the format “\$x{y}”, where the “x” portion of the variable is a primary variable comprising one or more letters, and the “y” portion comprises a string that itself can contain one or more variables of the same format (i.e., recursively nested variables) (e.g., “\$x{\$x{y}}”, “\$x{\$x{\$x{y}}”, “\$x{\$x{\$x{\$x{y}}}}”, etc). Creation of these variables is a relatively simple matter involving merely the use of a text editor to place, for example, \$C,\$L,\$MB variables, etc.

[0014] Configuration files are accessed through “keys” of the format “config.section.name” that identify the configuration file name (without the \*.cfg extension), section name in the file, and property name in the section, respectively. Either all configuration files can be located in the same directory, or else a “path” can be defined that specifies all directories containing all configuration files.