[*Example:* Consider a hypothetical application, ChrisOffice v1, which allows sound effects to be applied to SpreadsheetML conditional formatting. The implementers of ChrisOffice v1 decide to store the data inside application-defined extension elements because they would like other Office applications to be able to round-trip that data, and in SpreadsheetML an application-defined extension element (**extLst**) is already defined under the **conditionalFormattingElements** element. An example of this element with the additional sound effect added might look like this:

```
<conditionalFormattingElements>
   <extLst>
      <ext uri="myurl" xmlns:co11="http://chrisoffice/v1">
         <co11:soundeffect>
             <co11:sourceFile>moo.mp3</co11:sourceFile>
         </co11:soundeffect>
      </ext>
   </extLst>
</conditionalFormattingElements>
```

Understood by ChrisOffice v1 but round-tripped by other IS 29500 compliant applications

Now we add the ability to use video as well as MP3 in ChrisOffice v2. We want to allow ChrisOffice v1 to understand everything except the video, but we'll have to use a different namespace to avoid ChrisOffice discovering unknown content in understood namespaces. Because ChrisOffice v1 doesn't have its own extension elements inside its extension elements, our only option is to make the extra content ignorable, which will mean that ChrisOffice v1 discards it upon load.

Inside the file, this might look like:

```
<conditionalFormattingElements>
   <extLst>
      <ext uri="myurl" xmlns:co11="http://chrisoffice/v1">
         <co11:soundeffect mc:Ignorable="co13"
xmlns:co13="http://chrisoffice/v2">
             <co11:sourceFile>moo.mp3</co11:sourceFile>
             <co13:sourceVideo>cow.mpg</co13:sourceFile>
         </co11:soundeffect>
      </ext>
   </extLst>
</conditionalFormattingElements>
```

New element added by ChrisOffice v2. Understood by ChrisOffice v2, discarded by ChrisOffice v1 but round-tripped by other IS 29500 compliant applications

ChrisOffice v1 will discard the video segment when it reads the file, which is our desired behaviour. Because processing of MCE constructs is not permitted inside application-defined extension elements (*insert relevant reference*), applications which do not understand the original sound effect construct will not needlessly throw away the new video content.

Although it's not mandatory to use a preprocessor to implement MCE, it is useful to visualise processing by using "before" and "after" markup examples. In the case of the above document, the subtree extracted by either version of ChrisOffice is:

```
         <co11:soundeffect mc:Ignorable="co13"
xmlns:co13="http://chrisoffice/v2">
             <co11:sourceFile>moo.mp3</co11:sourceFile>
```

```
            <co13:sourceVideo>cow.mpg</co13:sourceFile>
        </co11:soundeffect>
```

If MCE was to be applied using a preprocessor for ChrisOffice v1, this subtree would result in:

```
        <co11:soundeffect mc:Ignorable="co13" 
xmlns:co13="http://chrisoffice/v2">
            <co11:sourceFile>moo.mp3</co11:sourceFile>
        </co11:soundeffect>
```

end example]