DOORS Rich Text - Behind the Markup

Michael Sutherland
Galactic Solutions Group LLC
michael.sutherland@galactic-solutions.com

Abstract:

The Telelogic product DOORS has provided its users with a comprehensive set of functionality to manage Requirements across the Enterprise. To further extend the capability of DOORS and allow users to customize the product to meet their specific needs, the makers of DOORS have provided a powerful Application Programmer Interface (API) called the DOORS Extension Language (DXL). This API gives the user access to the internal DOORS functionality, and unlocks the power of the tool beyond those functions present from the user interface.

The Telelogic product DOORS also surpasses the competition in its ability to add Rich Text Markup (including font markup, OLE Objects and pictures) to requirements documents.

Details of programmatic manipulation of Rich Text Markup in the DOORS database using DXL will be discussed, including:

(1) Detecting, adding and deleting Rich Text Markup contained in attribute text.
(2) Displaying Rich Text Markup in Layout DXL columns.
(3) Manipulating and removing embedded font markup.
(4) Differences between Rich Text Markup schemes in DOORS v5 and DOORS v6
(5) Implications of Rich Text Markup with regards to Importing from and Exporting to Microsoft Office applications.

Through the knowledge and application of DXL and Rich Text Markup, these needs can be addressed.

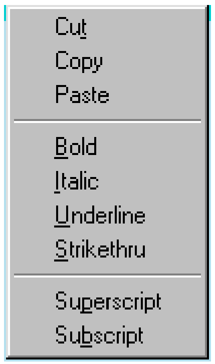
Biography:

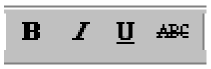
Michael Sutherland is the founder of Galactic Solutions Group, and has 13 years experience working with automotive and military suppliers and manufacturers. He has been a consultant to General Motors for 10 years, and has worked with General Motors Powertrain Division and General Motors North American Operations (NAO) Manufacturing Engineering Division. He is currently working with General Dynamics Land Systems (GDLS) Future Combat Systems (FCS), developing and deploying Systems Engineering Processes and Tools. Michael has a Masters Degree in Electrical and Computer Engineering from Oakland University in Rochester MI. He also specializes in the application of the DOORS Enterprise Requirements Suite, mentoring and teaching application, customization (DXL), and information modeling to a wide variety of clients across the nation.

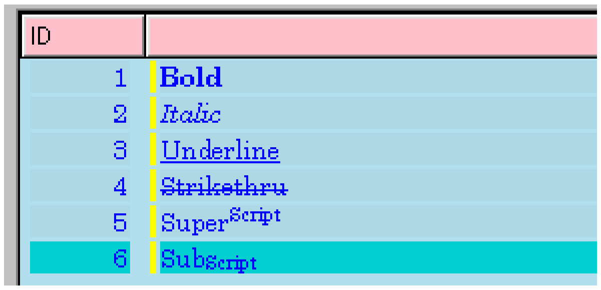
DOORS 4.0 - Introduction of Rich Text

With the release of DOORS 4.0, QSS (now Telelogic) introduced Rich Text editing into the realm of requirements management, giving users word processing style features for control over font style and effects.

When editing text in a DOORS Object, portions of the text can be highlighted, and Rich Text can be applied to the highlighted text by choosing the desired font effect from the right-click menu, the toolbar, or entering a key combination. Note: No key combinations are available for superscript and subscript.







ID	
1	**Bold**
2	*Italic*
3	Underline
4	~~Strikethru~~
5	Super^{Script}
6	Sub$_{Script}$

Ctrl+Shift+B	Makes current selection bold
Ctrl+Shift+I	Makes current selection italic
Ctrl+Shift+S	Makes current selection strikethru
Ctrl+Shift+U	Makes current selection underlined

Rich Text Markup - Overview

Markup refers to a sequence of character symbols (often called "tags") that are inserted into a data file or string to indicate how the data should be rendered or formatted when it is printed or displayed. One example of a markup language that does this is the HyperText Markup Language (HTML), a World Wide Web Consortium *(*W3C) specification for interoperable technologies.

Markup can also describe the structure of a data file, encoding a description of the document's storage layout and logical structure. One example of a markup language that does this is the Extensible Markup Language (XML), another World Wide Web Consortium *(*W3C) specification for interoperable technologies.

Rich Text Markup - DOORS 4 and above

In DOORS 4, support for simple Rich Text Markup such as font style and font effects (bold, italic, etc.) was added.

	Documented Markup Tag Strings	
Font Effect	On	Off
Bold	`"{\\b "`	`"}"`
Italic	`"{\\i "`	`"}"`
Underline	`"{\\ul "`	`"}"`
~~Strikethru~~	`"{\\strike "`	`"}"`
Super^{script}	`"{\\super "`	`"}"`
Sub$_{script}$	`"{\\sub "`	`"}"`
	`"{\\nosupersub "`	`"}"`

Note: The backslash character `'\'` is an *escape* character in strings, meaning `"\\" = '\'`, so the table above lists the proper string version of the tags as needed for programmatic use in Rich Text strings. A printed version of the string shows the desired result, which is the single backslash character `'\'`.

The syntax for using these tags within a string is as follows:

```
{tag<space>text} or {tag{text}}
```

The space character before the closing quote for "On" tags is mandatory, unless the tag is immediately followed by an opening brace character. A closing brace character turns off all markup nested between it and the previous opening brace character.

Tags can be nested, to apply more than one type of formatting, as follows:

```
{tag<space>text{tag<space>text}}
```

Example: Create the following Rich Text Formatted text:

Bold *Italic* ***BoldandItalic***

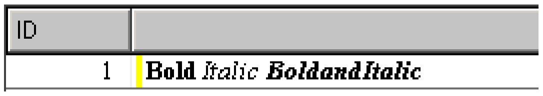
The following three examples show Rich Text Markup strings that will achieve the desired result:

```
"{\\b Bold} {\\i Italic} {\\b {\\i BoldandItalic}}"
"{\\b{Bold}} {\\i{Italic}} {\\b{\\i{BoldandItalic}}}"
"{\\b Bold} {\\i Italic} {\\b \\i BoldandItalic}"
```

DXL to set and display Rich Text strings – DOORS 4 thru 5

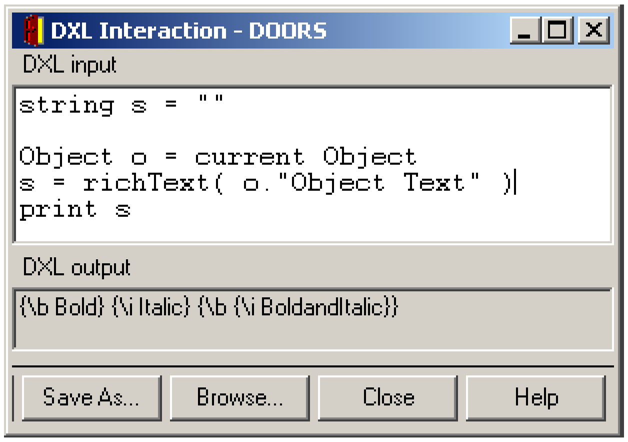
DOORS 4 thru 5 – Setting Rich Text Markup from the DOORS Graphical User Interface (GUI)

DOORS "Object Text" with Rich Text Markup



ID	
1	**Bold** *Italic* ***BoldandItalic***

DXL to show Rich Text Markup string for DOORS "Object Text"

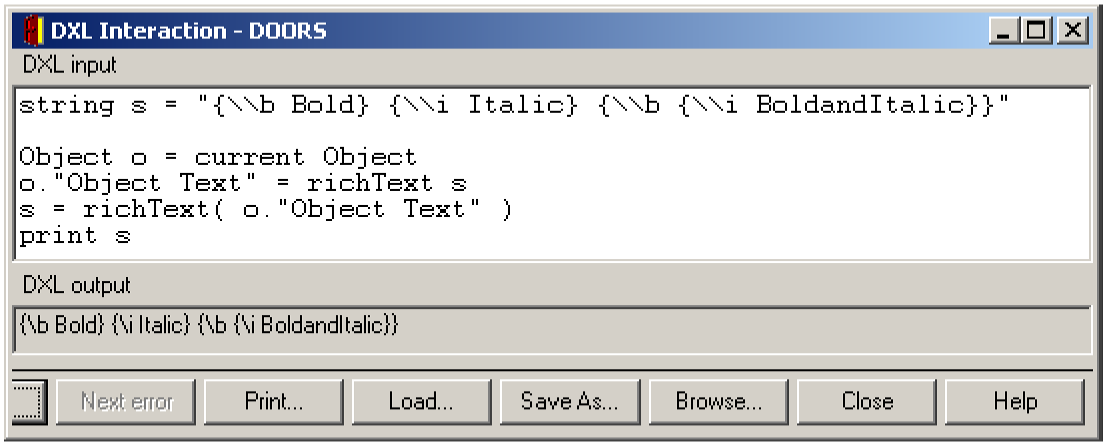


```
string s = ""

Object o = current Object
s = richText( o."Object Text" )
print s
```

```
{\b Bold} {\i Italic} {\b {\i BoldandItalic}}
```

DOORS 4 thru 5 – Setting Rich Text Markup via DOORS DXL

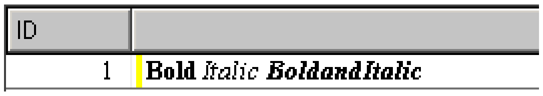
DXL to set DOORS "Object Text" with a Rich Text Markup string



```
string s = "{\\b Bold} {\\i Italic} {\\b {\\i BoldandItalic}}"

Object o = current Object
o."Object Text" = richText s
s = richText( o."Object Text" )
print s
```

```
{\b Bold} {\i Italic} {\b {\i BoldandItalic}}
```

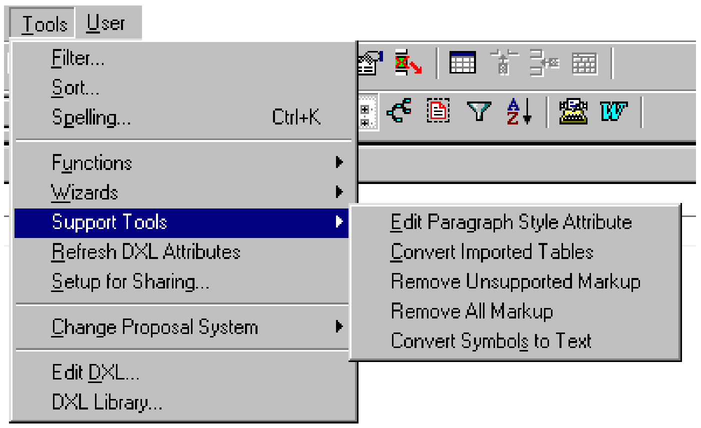
Resulting DOORS "Object Text" with Rich Text Markup



ID	
1	**Bold** *Italic* ***BoldandItalic***

DOORS 4.1 - Updates

New "Support Tools" added to the Tools menu of DOORS Formal Modules with the release of DOORS 4.1.	
<standard/doctools/normmark.dxl>	"Remove Unsupported Markup" This function removes all unsupported markup from all string and text attributes on all visible objects. Note: Uses **`removeUnlistedRichText`**()
<standard/doctools/delmark.dxl>	"Remove All Markup" (Rich Text mark-up Deletion Tool) This function removes all markup (bold, italic etc.) from all string and text attributes on all visible objects. Note: Uses **`o.attrname = o.attrname ""`** and **`deleteFontTable`**`()`
<standard/doctools/symbconv.dxl>	"Convert Symbols to Text" (Symbol to Plain Text Conversion Tool) This function converts symbols and other nonstandard character sets into the user's system default character set. Note: Uses "**`for rt in s do`**" {}



DOORS 5.0 - Minor Updates

DOORS 5 introduced no significant changes in Rich Text Markup structure and functionality.

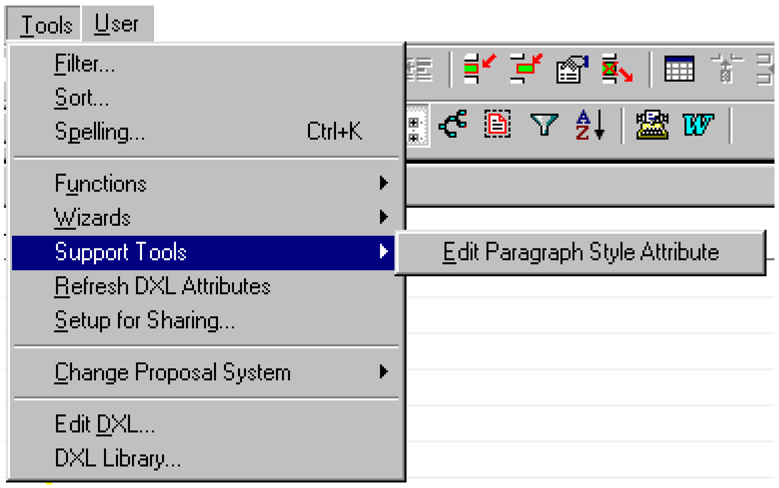
DOORS 6.0 - Major Update

With the release of DOORS 6.0, Telelogic adopted the Microsoft Rich Text Format (RTF) Specification (or some reasonable subset thereof). The Microsoft Rich Text Format is a metafile standard developed by Microsoft to encode formatted text and graphics for transfer between applications.

Note: It is a common misconception that Rich Text Format is a World Wide Web Consortium (W3C) specification for interoperable technologies. It is not.

Telelogic, believing that all of the legacy problems with unsupported markup were eliminated, removed the support tools from the Tools menu.

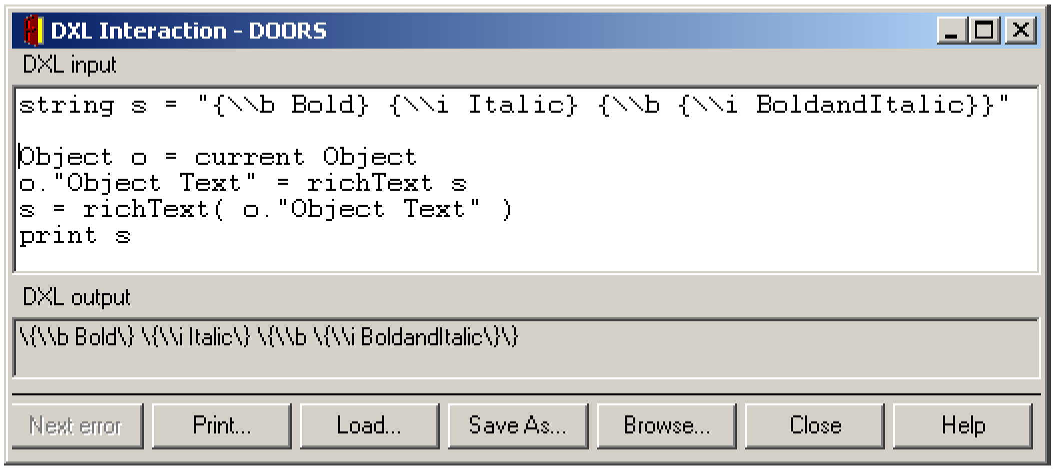
"Support Tools" removed from the Tools menu of DOORS Formal Modules with the release of DOORS 6.0.	
~~<standard/doctools/normmark.dxl>~~	"Remove Unsupported Markup"
~~<standard/doctools/delmark.dxl>~~	"Remove All Markup" (Rich Text mark-up Deletion Tool)
~~<standard/doctools/symbconv.dxl>~~	"Convert Symbols to Text" (Symbol to Plain Text Conversion Tool)



DXL to set and display Rich Text string – DOORS 6

DOORS 6 – Setting Rich Text Markup via DOORS DXL.

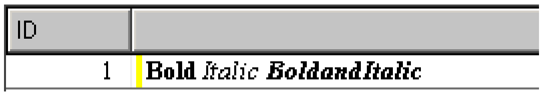
DXL to set and show resulting Rich Text Markup string for DOORS "Object Text"



```
string s = "{\\b Bold} {\\i Italic} {\\b {\\i BoldandItalic}}"

Object o = current Object
o."Object Text" = richText s
s = richText( o."Object Text" )
print s
```

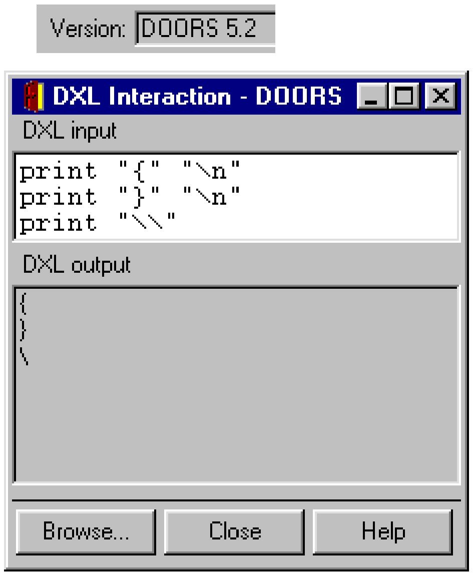
```
\{\\b Bold\} \{\\i Italic\} \{\\b \{\\i BoldandItalic\}\}
```

Resulting DOORS "Object Text" with Rich Text Markup shown.



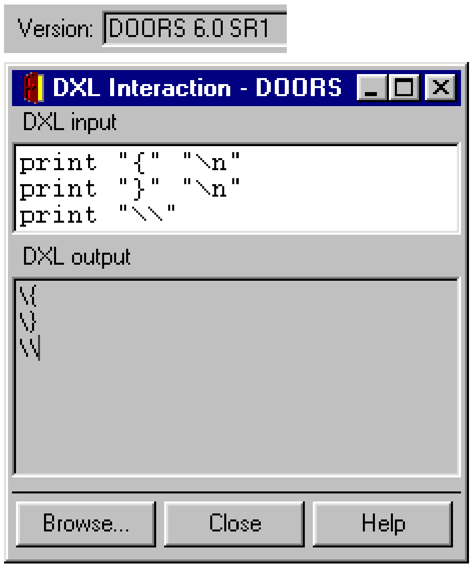
ID	
1	**Bold** *Italic* ***BoldandItalic***

Note extra "\" characters in the version of the Rich Text Markup string printed from DXL. In DOORS 6, printing the opening or closing brace characters ("{" or "}") or a backslash "\" results in an extra backslash character being displayed in the printed output as follows:



```
print "{" "\n"
print "}" "\n"
print "\\"
```

```
{
}
\
```



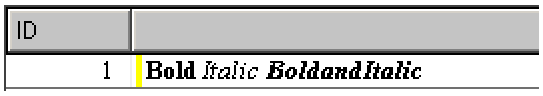
```
print "{" "\n"
print "}" "\n"
print "\\"
```

```
\{
\}
\\
```

This is a known bug in DOORS 6.x that has been fixed in DOORS 7.

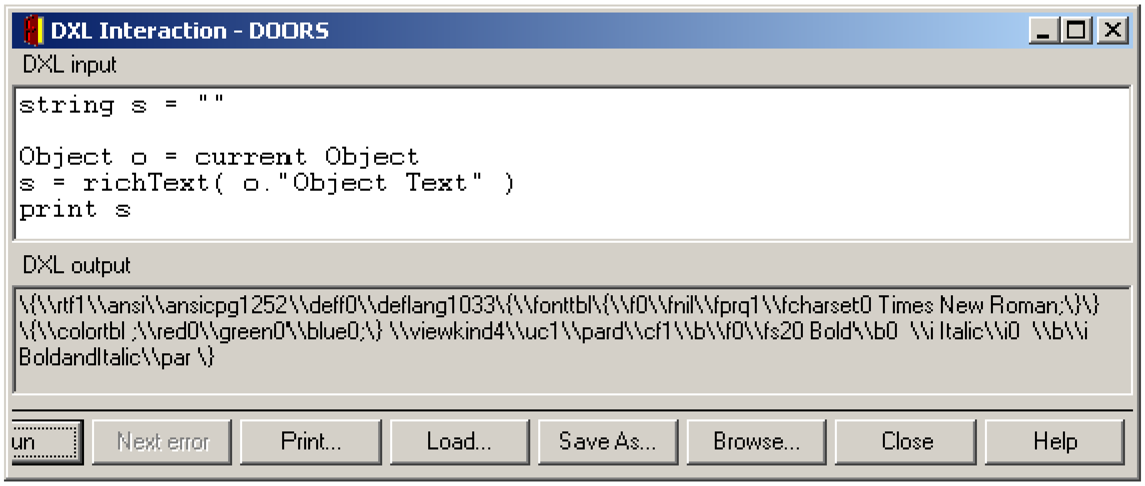
DOORS 6 – Setting Rich Text from the DOORS Graphical User Interface (GUI).

DOORS "Object Text" with Rich Text Markup



ID	
1	**Bold** *Italic* ***BoldandItalic***

DXL to show Rich Text Markup string for DOORS "Object Text"



```
string s = ""

Object o = current Object
s = richText( o."Object Text" )
print s
```

```
\{\\rtf1\\ansi\\ansicpg1252\\deff0\\deflang1033\{\\fonttbl\{\\f0\\fnil\\fprq1\\fcharset0 Times New Roman;\}\}
\{\\colortbl ;\\red0\\green0\\blue0;\} \\viewkind4\\uc1\\pard\\cf1\\b\\f0\\fs20 Bold\\b0  \\i Italic\\i0  \\b\\i
BoldandItalic\\par \}
```

Microsoft Rich Text Markup is inserted (with extra '\' characters preceding each brace and backslash character in the printed Rich Text Markup string).

Note: No documentation on Microsoft Rich Text Markup, or how it is used, is available from Telelogic.

Rich Text Markup - DOORS 6 and above

The encoded data is called a RTF *code stream*, which consists of fields called *control words* (formatting and printing instructions), *control symbols (*escape character sequences), and *groups* (enclosed in brace ({}) characters). Each field begins with a backslash (\) character.

Object Heading - Default Markup (for non-empty Object Heading)

```
\{\\rtf1\\ansi\\ansicpg1252\\deff0\\deflang1033\{\\fonttbl\{\\f0
\\fnil\\fprq1\\fcharset0 Arial;\}\} \{\\colortbl
;\\red0\\green0\\blue0;\} \\viewkind4\\uc1\\pard\\cf1\\f0\\fs28
\\par \}
```

Object Text - Default Markup (for non-empty Object Text)

```
\{\\rtf1\\ansi\\ansicpg1252\\deff0\\deflang1033\{\\fonttbl\{\\f0
\\fnil\\fprq1\\fcharset0 Times New
Roman;\}\{\\f1\\fnil\\fprq1\\fcharset0 Century Schoolbook;\}\}
\{\\colortbl ;\\red0\\green0\\blue0;\}
\\viewkind4\\uc1\\pard\\cf1\\f0\\fs20  \\f1\\par \}
```

Detailed explanation of Microsoft Rich Text Markup

The control codes **`\rtf1\ansi`** indicate that this data stream is an RTF document, that the code conforms to version 1 of the RTF specification, and that the document uses the ANSI (\ansi) rather than the PC (\pc), PS/2 (\pca), or Macintosh (\mac) character sets.

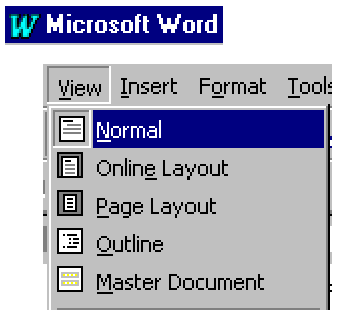
The **`\fonttbl`** group contains the descriptions of the fonts used within the document. The fields are Font Number (fn or fnnnn), Font Family (see below), and Font Name.

Control word	**Font family**
`\fnil`	Unknown or default fonts (default)
`\froman`	Roman, proportionally spaced serif fonts (TmsRmn, Palatino, etc.)
`\fswiss`	Swiss, proportionally spaced sans serif fonts (Swiss, etc.)
`\fmodern`	Fixed-pitch serif and sans serif fonts (Courier, Elite, Pica, etc.)
`\fscript`	Script fonts (Cursive, etc.)
`\fdecor`	Decorative fonts (Old English Zapf Chancery, etc.)
`\ftech`	Technical, symbol, and mathematical fonts (Symbol, etc.)

The **`\colortbl`** group is a color table used to control screen and printer colors. This file defines a basic palette of 16 colors, with each color channel containing an 8-bit index value in the range of 0 to 255.

The **`\viewkind`** control word is an integer (0-5) that represents the view mode of the document, corresponding to the views available in Microsoft Word:

RTF None	**`\viewkind0`**
RTF Normal	**`\viewkind4`**
RTF Outline Layout	**`\viewkind5`**
RTF Page Layout	**`\viewkind1`**
RTF Outline	**`\viewkind2`**
RTF Master Document	**`\viewkind3`**



The **`\uc1`** control word represents the number of bytes corresponding to a given **\u*N*** Unicode character.

The **`\pard`** control word resets to default paragraph properties.

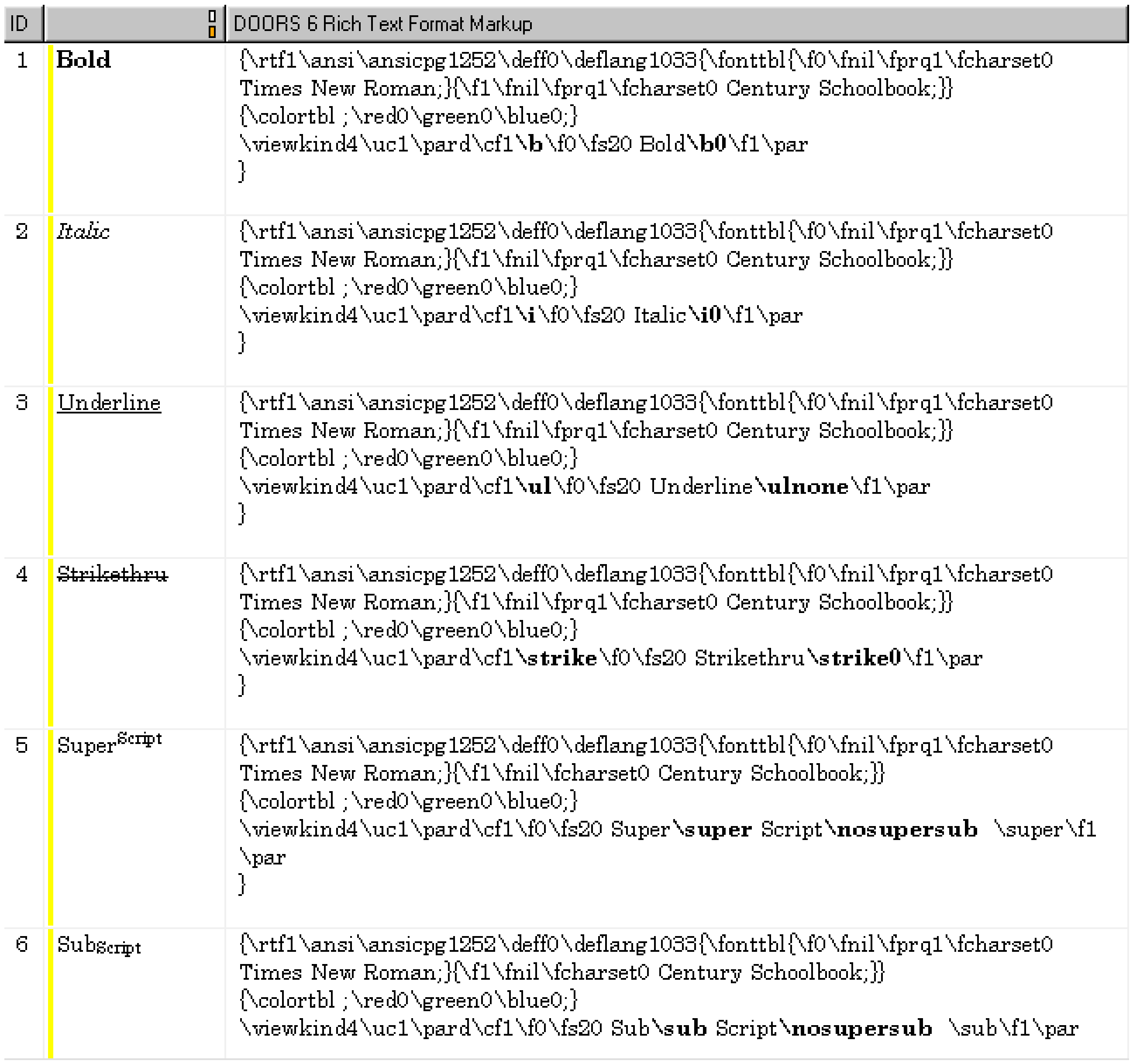
The **`\cf1`** control word specifies the foreground color.

The **`\fN`** control word Font number. *N* refers to an entry in the font table.

The **`\fsN`** control word specifies the font size in half-points.

The **`\par`** control word specifies a new paragraph.

Rich Text Markup - Font Effects in DOORS 6



ID		DOORS 6 Rich Text Format Markup

1	**Bold**	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}}
		{\colortbl ;\red0\green0\blue0;}
		\viewkind4\uc1\pard\cf1**\b**\f0\fs20 Bold**\b0**\f1\par
		}

2	*Italic*	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}}
		{\colortbl ;\red0\green0\blue0;}
		\viewkind4\uc1\pard\cf1**\i**\f0\fs20 Italic**\i0**\f1\par
		}

3	Underline	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}}
		{\colortbl ;\red0\green0\blue0;}
		\viewkind4\uc1\pard\cf1**\ul**\f0\fs20 Underline**\ulnone**\f1\par
		}

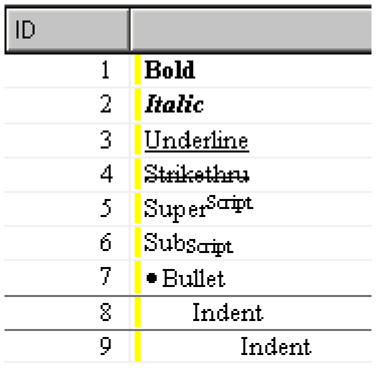
4	~~Strikethru~~	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}}
		{\colortbl ;\red0\green0\blue0;}
		\viewkind4\uc1\pard\cf1**\strike**\f0\fs20 Strikethru**\strike0**\f1\par
		}

5	Super^{Script}	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fcharset0 Century Schoolbook;}}
		{\colortbl ;\red0\green0\blue0;}
		\viewkind4\uc1\pard\cf1\f0\fs20 Super**\super** Script**\nosupersub** \super\f1
		\par
		}

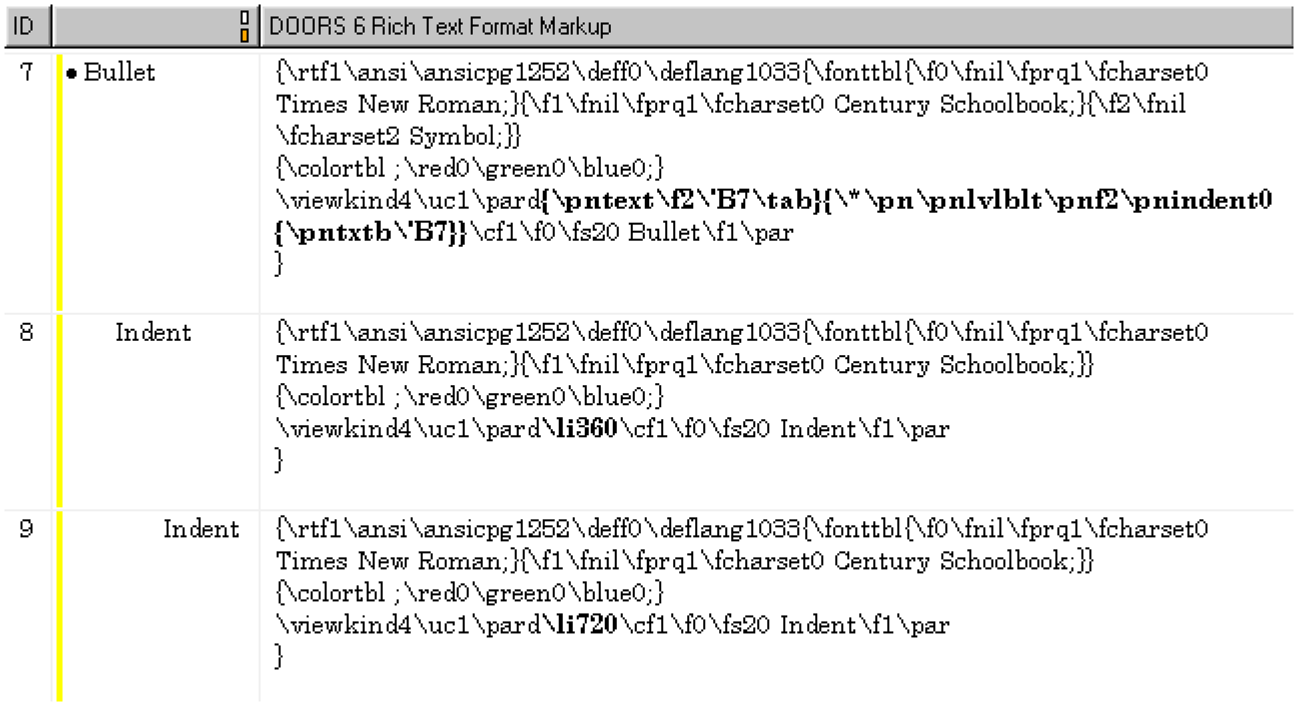
6	Sub$_{Script}$	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fcharset0 Century Schoolbook;}}
		{\colortbl ;\red0\green0\blue0;}
		\viewkind4\uc1\pard\cf1\f0\fs20 Sub**\sub** Script**\nosupersub** \sub\f1\par

Rich Text Markup - New Font Effects in DOORS 6

In DOORS 6, Rich Text Markup for the bulleting and indenting of paragraphs is supported.

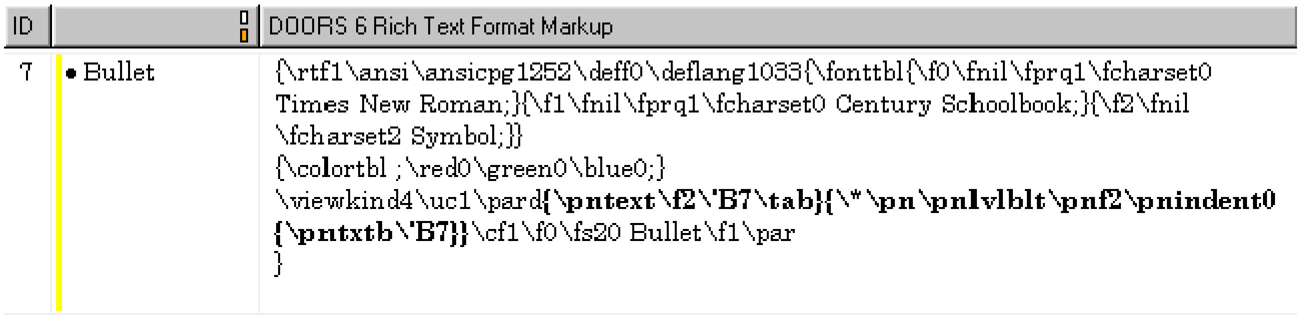


ID	
1	**Bold**
2	***Italic***
3	Underline
4	~~Strikethru~~
5	Super^{Script}
6	Sub$_{Script}$
7	• Bullet
8	Indent
9	Indent



ID		DOORS 6 Rich Text Format Markup
7	• Bullet	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}{\f2\fnil \fcharset2 Symbol;}} {\colortbl ;\red0\green0\blue0;} \viewkind4\uc1\pard**{\pntext\f2\'B7\tab}{\*\pn\pnlvlblt\pnf2\pnindent0 {\pntxtb\'B7}}**\cf1\f0\fs20 Bullet\f1\par }
8	Indent	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}} {\colortbl ;\red0\green0\blue0;} \viewkind4\uc1\pard**\li360**\cf1\f0\fs20 Indent\f1\par }
9	Indent	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}} {\colortbl ;\red0\green0\blue0;} \viewkind4\uc1\pard**\li720**\cf1\f0\fs20 Indent\f1\par }

DOORS 6 - Rich Text Markup for Bullets



ID		DOORS 6 Rich Text Format Markup
7	• Bullet	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}{\f2\fnil \fcharset2 Symbol;}} {\colortbl ;\red0\green0\blue0;} \viewkind4\uc1\pard**{\pntext\f2\'B7\tab}{\*\pn\pnlvlblt\pnf2\pnindent0 {\pntxtb\'B7}}**\cf1\f0\fs20 Bullet\f1\par }

The **\pntext** group precedes all bulleted or numbered paragraphs, and will contain all the text and formatting that would be auto-generated as plain text to provide compatibility with existing RTF readers.

The ***** control word marks a destination whose text should be ignored if not understood by the RTF reader.

The **\pn** destination control word turns on paragraph numbering.

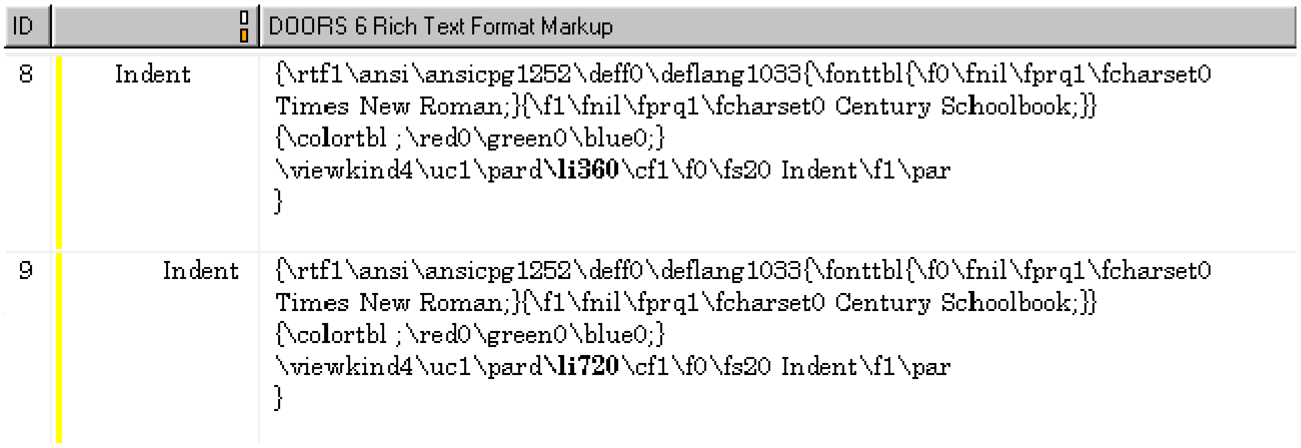
The **\pnlvlblt** control word designates a bulleted paragraph (corresponds to level 11). The actual character used for the bullet is stored in the **\pntxtb** group.

The **\pnfN** control word specifies a font number.

The **\pnindentN** control word specifies the minimum distance from margin to body text.

The **\pntxtb** destination control word contains the text that precedes the number.

DOORS 6 - Rich Text Markup for Indents



ID		DOORS 6 Rich Text Format Markup
8	Indent	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}} {\colortbl ;\red0\green0\blue0;} \viewkind4\uc1\pard**\li360**\cf1\f0\fs20 Indent\f1\par }
9	Indent	{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}{\f1\fnil\fprq1\fcharset0 Century Schoolbook;}} {\colortbl ;\red0\green0\blue0;} \viewkind4\uc1\pard**\li720**\cf1\f0\fs20 Indent\f1\par }

The **`\liN`** control word specifies a left indent level, in Twips. A Twip is twentieth of a Point (1/1440th of an inch or 1/567th of a centimeter). A Point is approximately 1/72nd of an inch. Twips are used because they are a screen-independent unit, which can be adjusted in size as screen resolution is changed, but also express an absolute value for printing (unlike pixels).

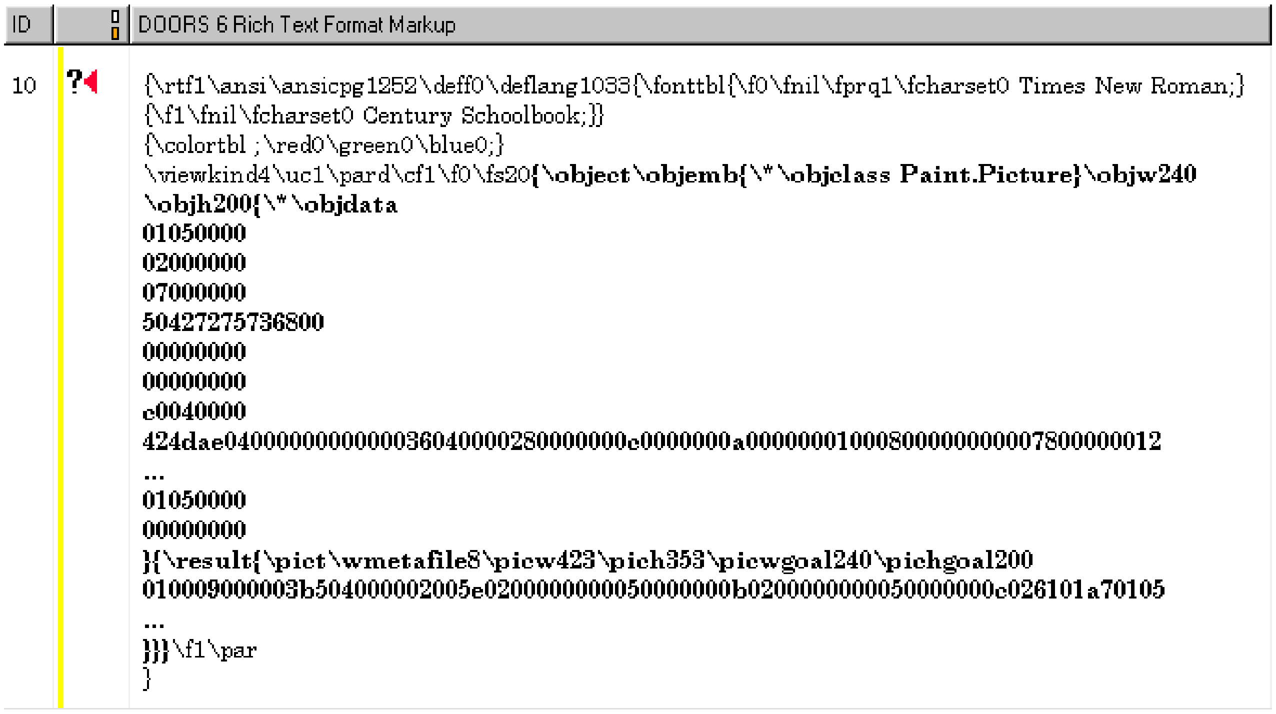
In DOORS 6 and above the base unit of indentation in DOORS 6.0 is 360 twips, so values of the left indent level will be multiples of 360.

In DOORS 7, the `applyTextFormattingToParagraph()` command was introduced to allow the application of bullets and indents to paragraphs through a single DXL command. In DOORS 6, the only way to programmatically add bullets and indents is to comprehend and apply detailed knowledge of Microsoft Rich Text Markup and string manipulation.

DOORS 6 - Rich Text Markup for OLE Objects

In DOORS 6 and above, OLE Objects are stored as Rich Text Markup data in Text Attributes (such as "Object Text".

Example: Insert -> OLE Object (from file `$DOORSHOME\lib\dxl\standard\suspect\indicate\suspect_in.bmp`).



ID		DOORS 6 Rich Text Format Markup
10	?	see below

```
{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fnil\fprq1\fcharset0 Times New Roman;}
{\f1\fnil\fcharset0 Century Schoolbook;}}
{\colortbl ;\red0\green0\blue0;}
\viewkind4\uc1\pard\cf1\f0\fs20{\object\objemb{\*\objclass Paint.Picture}\objw240
\objh200{\*\objdata
01050000
02000000
07000000
50427275736800
00000000
00000000
c0040000
424dae0400000000000036040000280000000c0000000a00000001000800000000007800000012
...
01050000
00000000
}{\result{\pict\wmetafile8\picw423\pich353\picwgoal240\pichgoal200
010009000003b504000002005e0200000000050000000b0200000000050000000c026101a70105
...
}}}\f1\par
}
```

Note: Major chunks of the image data in the above example have been replaced by "…", for brevity.

The **`\object`** control word indicates an OLE Object.

The **`\objemb`** control word indicates an object type of OLE embedded object.

The **`\objclass`** control word allows for a text argument that specifies the object class to use for this object and ignores the class specified in the object data.

The **`\objwN`** and **`\objhN`** control words specify the original object height in twips, assuming the object has a graphical representation.

The **`\objdata`** subdestination control word contains the data for the object in the appropriate format; OLE objects are in **OLESaveToStream** format.

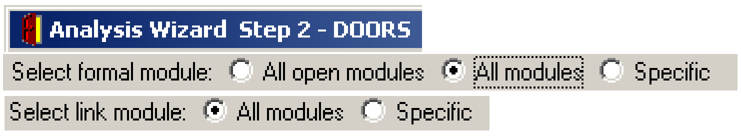
The **`\result`** optional control word contains the last update of the result of the object (a **`\pict`** (bitmap) snapshot of the OLE Object).

Application - Rich Text Markup and Layout DXL - Setup

DOORS 4.0 and above allows Rich Text Markup for Font Effects to be displayed in a Layout DXL column. This can help make traceability columns more readable.

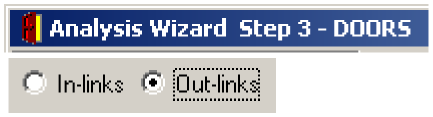
Example: Take the output from the DOORS Analysis->Wizard and add Rich Text bolding to the Attribute names.

Use the Analysis -> Wizard to generate Layout DXL:

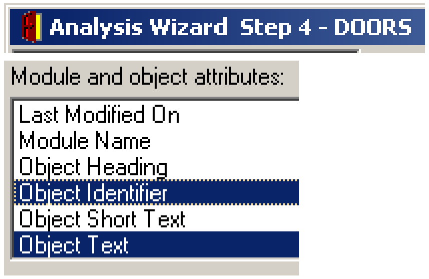

Analysis Wizard Step 2 - DOORS

Select formal module: All open modules [✓ All modules] Specific

Select link module: [✓ All modules] Specific

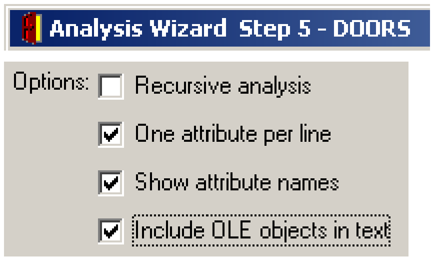

Analysis Wizard Step 3 - DOORS

In-links [✓ Out-links]

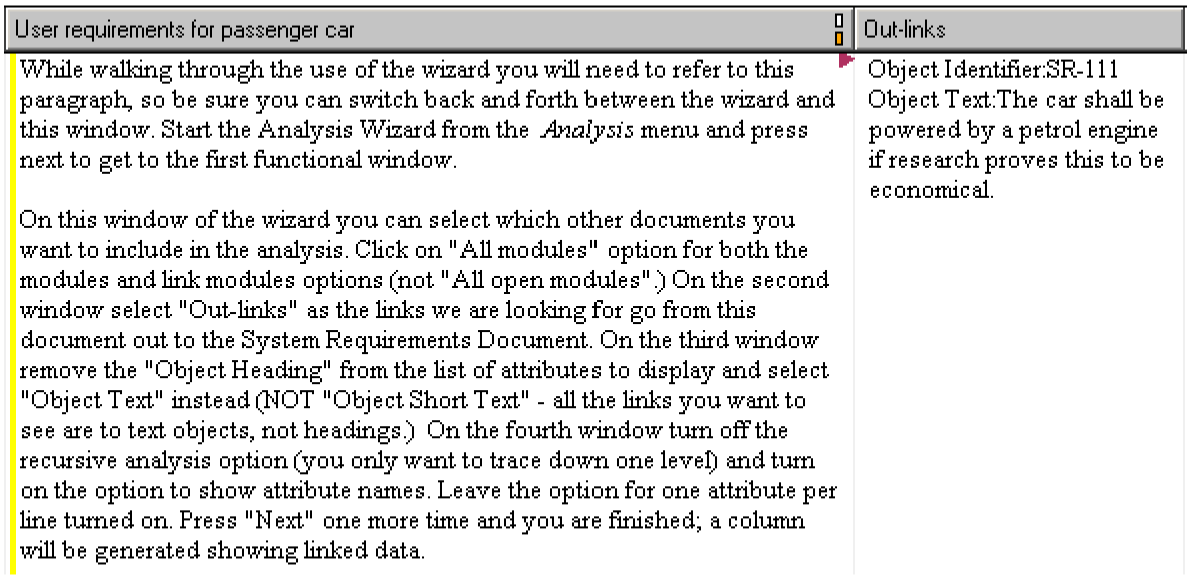

Analysis Wizard Step 4 - DOORS

Module and object attributes:

Last Modified On
Module Name
Object Heading
[✓ Object Identifier]
Object Short Text
[✓ Object Text]


Analysis Wizard Step 5 - DOORS

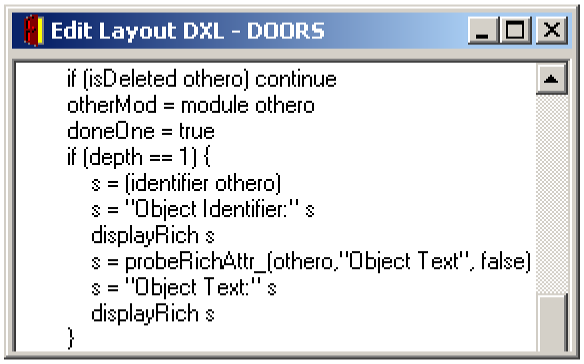
Options:
- Recursive analysis
- [✓ One attribute per line]
- [✓ Show attribute names]
- [✓ Include OLE objects in text]



User requirements for passenger car	Out-links
While walking through the use of the wizard you will need to refer to this paragraph, so be sure you can switch back and forth between the wizard and this window. Start the Analysis Wizard from the *Analysis* menu and press next to get to the first functional window. On this window of the wizard you can select which other documents you want to include in the analysis. Click on "All modules" option for both the modules and link modules options (not "All open modules".) On the second window select "Out-links" as the links we are looking for go from this document out to the System Requirements Document. On the third window remove the "Object Heading" from the list of attributes to display and select "Object Text" instead (NOT "Object Short Text" - all the links you want to see are to text objects, not headings.) On the fourth window turn off the recursive analysis option (you only want to trace down one level) and turn on the option to show attribute names. Leave the option for one attribute per line turned on. Press "Next" one more time and you are finished; a column will be generated showing linked data.	Object Identifier:SR-111 Object Text:The car shall be powered by a petrol engine if research proves this to be economical.

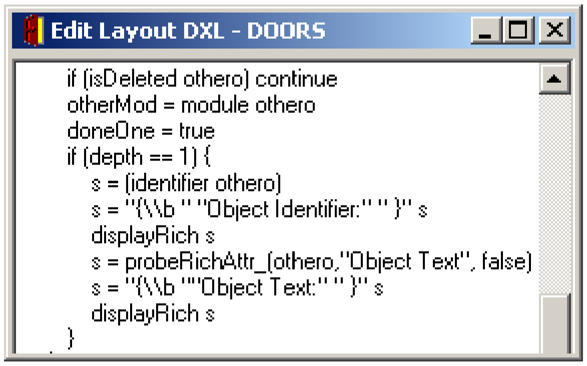
Application - Rich Text Markup and Layout DXL - Results

Layout DXL as generated by the Analysis Wizard



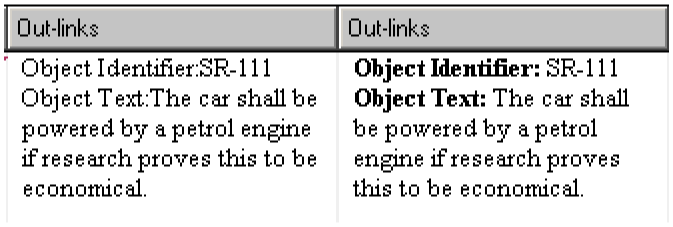
```
if (isDeleted othero) continue
otherMod = module othero
doneOne = true
if (depth == 1) {
    s = (identifier othero)
    s = "Object Identifier:" s
    displayRich s
    s = probeRichAttr_(othero,"Object Text", false)
    s = "Object Text:" s
    displayRich s
}
```

Layout DXL modified to display attribute name in bold.



```
if (isDeleted othero) continue
otherMod = module othero
doneOne = true
if (depth == 1) {
    s = (identifier othero)
    s = "{\\b " "Object Identifier:" " }" s
    displayRich s
    s = probeRichAttr_(othero,"Object Text", false)
    s = "{\\b ""Object Text:" " }" s
    displayRich s
}
```

Resulting Layout DXL Display Columns (original and modified)

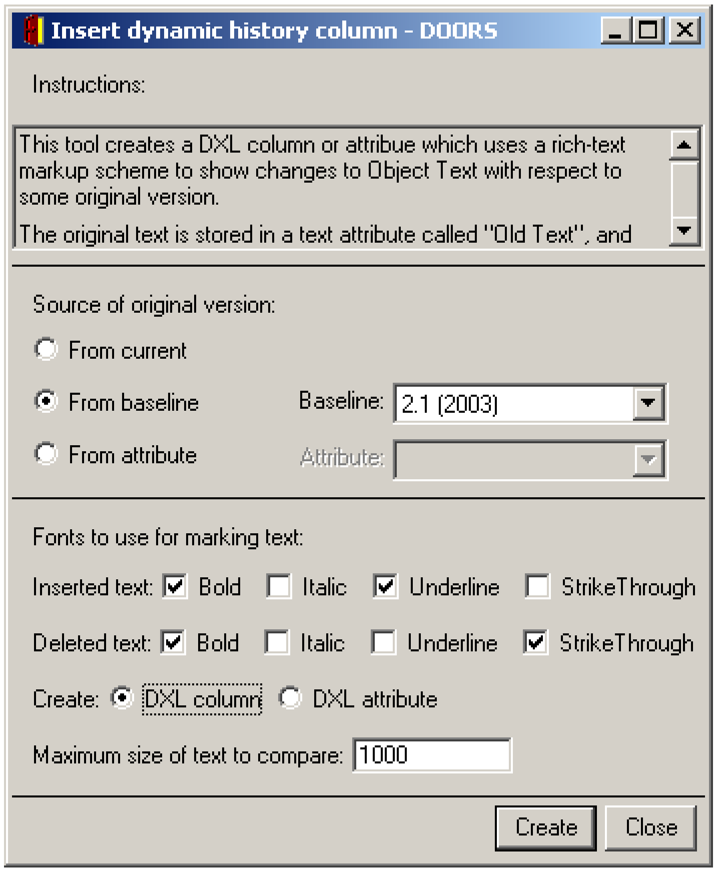


Out-links	Out-links
Object Identifier:SR-111 Object Text:The car shall be powered by a petrol engine if research proves this to be economical.	**Object Identifier:** SR-111 **Object Text:** The car shall be powered by a petrol engine if research proves this to be economical.

Application – Rich Text and History Markup

A script is included with DOORS 6.0 that uses Rich Text Markup to show changes to "Object Text" with respect to some original version (from a Baseline, etc.).

```
$DOORSHOME/lib/dxl/example/histcol.dxl
```



Insert dynamic history column - DOORS

Instructions:

This tool creates a DXL column or attribute which uses a rich-text markup scheme to show changes to Object Text with respect to some original version.
The original text is stored in a text attribute called "Old Text", and

Source of original version:

From current

[✓ From baseline] Baseline: 2.1 (2003)

From attribute Attribute:

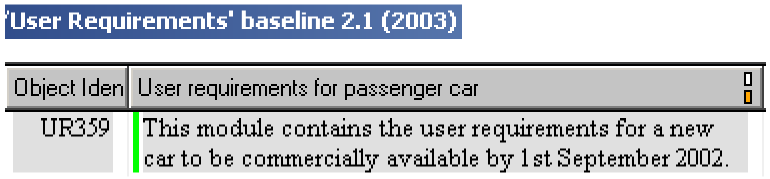
Fonts to use for marking text:

Inserted text: [✓ Bold] Italic [✓ Underline] StrikeThrough

Deleted text: [✓ Bold] Italic Underline [✓ StrikeThrough]

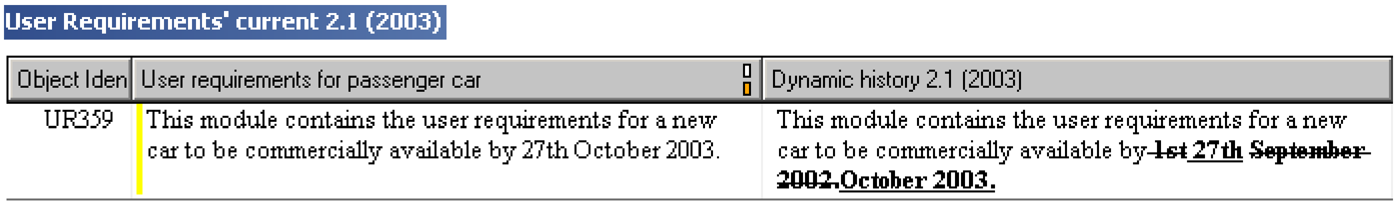
Create: [✓ DXL column] DXL attribute

Maximum size of text to compare: 1000



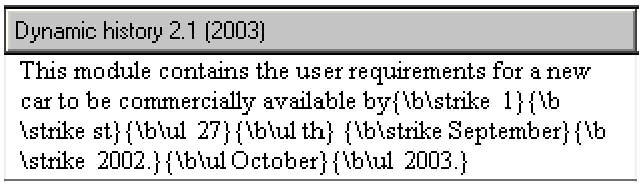
'User Requirements' baseline 2.1 (2003)

Object Iden	User requirements for passenger car
UR359	This module contains the user requirements for a new car to be commercially available by 1st September 2002.



User Requirements' current 2.1 (2003)

Object Iden	User requirements for passenger car	Dynamic history 2.1 (2003)
UR359	This module contains the user requirements for a new car to be commercially available by 27th October 2003.	This module contains the user requirements for a new car to be commercially available by **~~1st~~** **27th** **~~September 2002.~~****October 2003.**



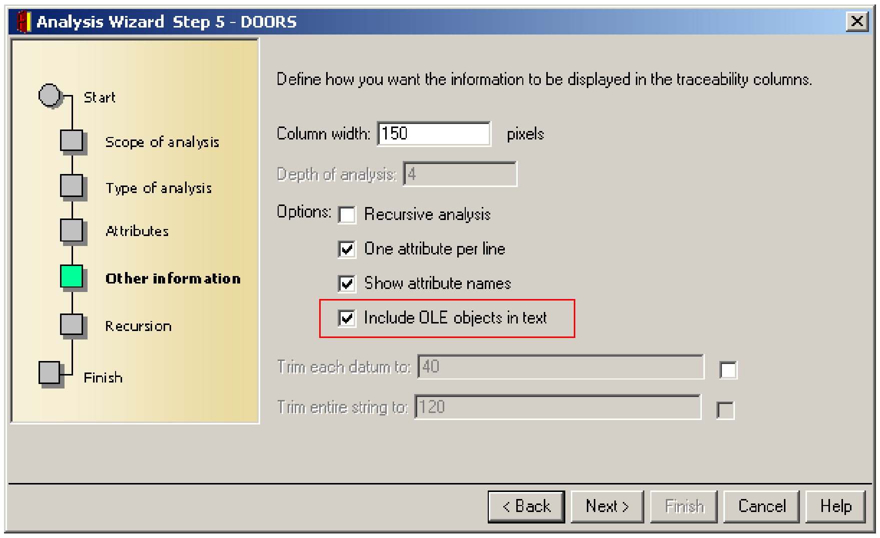
Dynamic history 2.1 (2003)
This module contains the user requirements for a new car to be commercially available by{\b\strike 1}{\b \strike st}{\b\ul 27}{\b\ul th} {\b\strike September}{\b \strike 2002.}{\b\ul October}{\b\ul 2003.}

Application - Rich Text Traceability

DOORS 6.0 and above allows the complete set of Rich Text Markup (including OLE objects and Pictures) to be displayed in a Layout DXL column. This allows complete traceability reporting to requirements residing in or supported by graphic objects.

Example:

When using the Analysis -> Wizard to create Layout DXL Column, make sure "Include OLE objects in text" is selected in Step 5.



Define how you want the information to be displayed in the traceability columns.

Column width: 150 pixels

Depth of analysis: 4

Options: [ ] Recursive analysis

[✓] One attribute per line

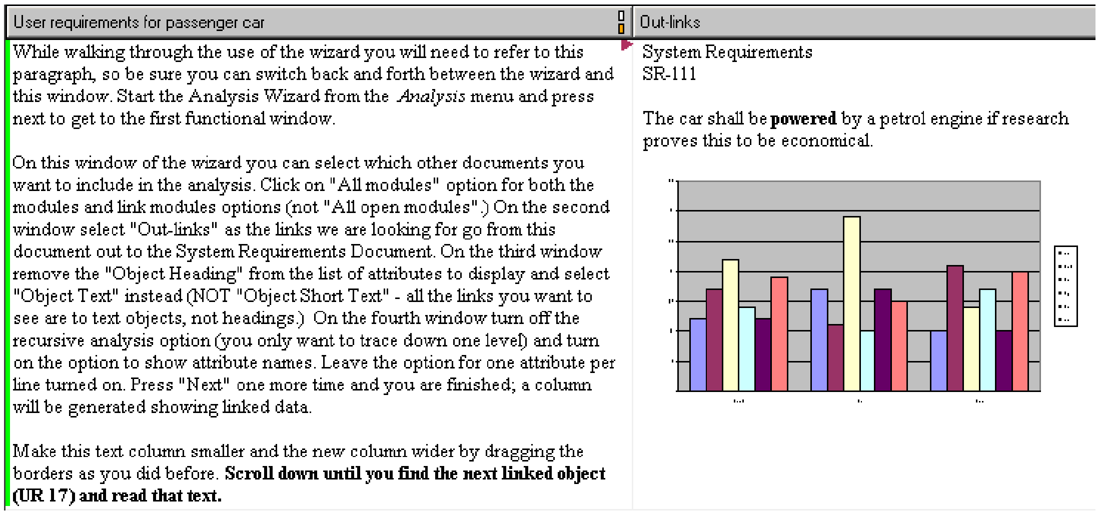
[✓] Show attribute names

[✓] Include OLE objects in text

Trim each datum to: 40 [ ]

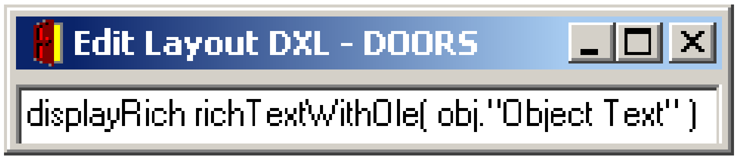
Trim entire string to: 120 [ ]

The result is a Layout DXL traceability column that displays Rich Text (including OLE Objects).



User requirements for passenger car	Out-links
While walking through the use of the wizard you will need to refer to this paragraph, so be sure you can switch back and forth between the wizard and this window. Start the Analysis Wizard from the *Analysis* menu and press next to get to the first functional window. On this window of the wizard you can select which other documents you want to include in the analysis. Click on "All modules" option for both the modules and link modules options (not "All open modules".) On the second window select "Out-links" as the links we are looking for go from this document out to the System Requirements Document. On the third window remove the "Object Heading" from the list of attributes to display and select "Object Text" instead (NOT "Object Short Text" - all the links you want to see are to text objects, not headings.) On the fourth window turn off the recursive analysis option (you only want to trace down one level) and turn on the option to show attribute names. Leave the option for one attribute per line turned on. Press "Next" one more time and you are finished; a column will be generated showing linked data. Make this text column smaller and the new column wider by dragging the borders as you did before. **Scroll down until you find the next linked object (UR 17) and read that text.**	System Requirements SR-111 The car shall be **powered** by a petrol engine if research proves this to be economical.

The following Layout DXL shows how the "displayRich" and "richTextWithOle" commands are used to achieve Rich Text Traceability.



```
displayRich richTextWithOle( obj."Object Text" )
```

DOORS Font Table - Module Level Attribute

When information from Microsoft Word or RTF documents is imported into a DOORS Module, a Module level attribute called "Font Table" may be created and populated with non-default font table entries. The DOORS DXL manual states that these entries are “used for mapping richtext font markup to character set information to override embedded font markup.”

DOORS default font table is:

```
{\f1012\fswiss\fcharset177 Arial;}
{\f1011\fswiss\fcharset162 Arial;}
{\f1010\fswiss\fcharset238 Arial;}
{\f1009\fswiss\fcharset204 Arial;}
{\f1008\fswiss\fcharset161 Arial;}
{\f1007\fswiss\fcharset0 Arial;}
{\f1006\froman\fcharset177 Times New Roman;}
{\f1005\froman\fcharset162 Times New Roman;}
{\f1004\froman\fcharset238 Times New Roman;}
{\f1003\froman\fcharset204 Times New Roman;}
{\f1002\froman\fcharset161 Times New Roman;}
{\f1001\ftech\fcharset2 Symbol;}
{\f1000\froman\fcharset0 Times New Roman;}
```

Rich Text and Importing from Microsoft Office Applications

Since DOORS 4.1.4 SR2 (and perhaps earlier), DOORS has included embedded font information when importing from Microsoft Word. The following example data was generated by importing a Microsoft Word document into DOORS 4.1.4 SR2.

Example Font Table created from data imported from Microsoft Word:

```
{\f3\froman\fcharset2\fprq2 Symbol;}
{\f2\fnil\fcharset2\fprq2 Monotype Sorts;}
{\f1\fswiss\fcharset0\fprq2 Arial;}
{\f0\froman\fcharset0\fprq2 Times New Roman;}
```

Example Imported Font Style Markup imported from Microsoft Word:

```
{\f1007 }
{\f1007 \f1 }
{\f1 }
{\f1007 \f1 \f1001 ±\f1 }
```

Example Imported Font Effects imported from Microsoft Word:

```
\b \b0 \ul \ul0 \i
```

Note: This demonstrates that DOORS, since version 4.0, has always imported and supported (without conversion) a limited amount of Microsoft Rich Text Markup. This is not documented in the DOORS manuals.

	Documented Markup Tag Strings		Alternate, Undocumented Markup Tag Strings	
Font Effect	On	Off	On	Off
Bold	`"{\\b "`	`"}"`	`"\\b "`	`"\\b0 "`
Italic	`"{\\i "`	`"}"`	`"\\i "`	`"\\i0 "`
Underline	`"{\\ul "`	`"}"`	`"\\ul "`	`"\\ul0 " or "\\ulnone"`
~~Strikethru~~	`"{\\strike "`	`"}"`	`"\\strike "`	`"\\strike0 "`
Super^{script}	`"{\\super "`	`"}"`	`"\\super "`	`"\\super0 " or "\\nosupersub "`
Sub$_{script}$	`"{\\sub "`	`"}"`	`"\\sub "`	`"\\sub0 " or "\\nosupersub "`

Rich Text Markup - Normalization in DOORS 6

The DOORS 5 Manual states: "In versions of DOORS prior to 4.1, the DOORS import tools imported rich text markup that wasn't supported by DOORS. DOORS now only imports supported rich-text markup, and discards markup that it doesn't support."

Now that DOORS 6 and above has implemented the Microsoft Rich Text Format (RTF) standard more fully, some of this markup that was previously suppressed may reemerge when DOORS 5 data is converted to DOORS 6. In particular, font markup may appear to be "non-standard" with regards to Fonts.

Note: Font markup has always been supported, and has never been suppressed.

The solutions recommended by Telelogic are:

(1) "Remove Unsupported Markup" (attempt to remove only unsupported markup)
(2) "Remove All Markup" (strip out all Rich Text Format, leaving raw text with no formatting)

These functions first provided in DOORS 4 and available in DOORS 5 were not released with DOORS 6. Attempting to use them results in less than desirable results. Solutions (1,2) does not preserve Symbol Font characters, and leaves unwanted Font style information. Solution (2) is not acceptable because valid markup must be preserved.

In DOORS 5, deleting the contents of the Module level "Font Table" attribute eliminated some non-standard fonts from appearing. In DOORS 6, this does not seem to have the same effect.

The requirements for the desired normalization are:

(1) All Font Markup shall be eliminated for Object Heading and Object Text Attributes, except:

  (a) Symbol Font characters shall be preserved.

(2) Valid Rich Text Markup (bold, italic, strikethru, underline, superscript, subscript) shall be preserved.

(3) Object Text with embedded OLE, bullet, or indent markup (new features in DOORS 6) shall not be modified.

Rich Text Markup - Normalization in DOORS 6 - Algorithm:

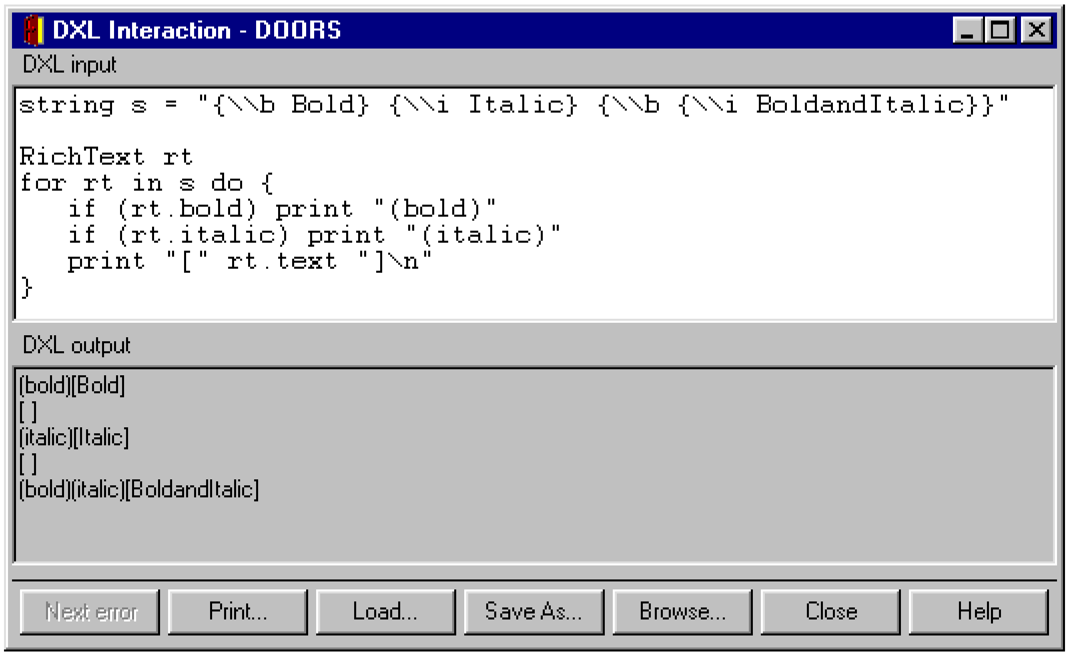
Use a "for rt in s" loop to scan the existing Rich Text string "chunk" by "chunk" for valid markup. This method has the benefit of ignoring any invalid markup. A new version of the Rich Text string is built by taking each successive chunk of formatting and reapplying the valid markup to the plain text version of the string.

```
// s is Rich Text string
string s = richText( o."Object Text" )

Buffer richBufferNew = create

RichText rt
for rt in s do {
     string text = rt.text
     string markupPrefix = ""
     string markupSuffix = ""
     if ( rt.bold ) {
               markupPrefix = "{\\b " markupPrefix
               markupSuffix = markupSuffix "}"
     }
     if ( rt.italic ) {
               markupPrefix = "{\\i " markupPrefix
               markupSuffix = markupSuffix "}"
     }
     .......
          if ( rt.charset == charsetSymbol ) {
               markupPrefix = markupPrefix "{\\f1001 "
               markupSuffix = "}" markupSuffix
     }

     // Apply markup prefix and suffix to plain text
     richBufferNew += markupPrefix text markupSuffix
}
```

The above code is from a free DXL utility called "Normalize Rich Text Markup", available from http://galactic-solutions.com.

Rich Text Markup - Normalization in DOORS 6 - Implication:

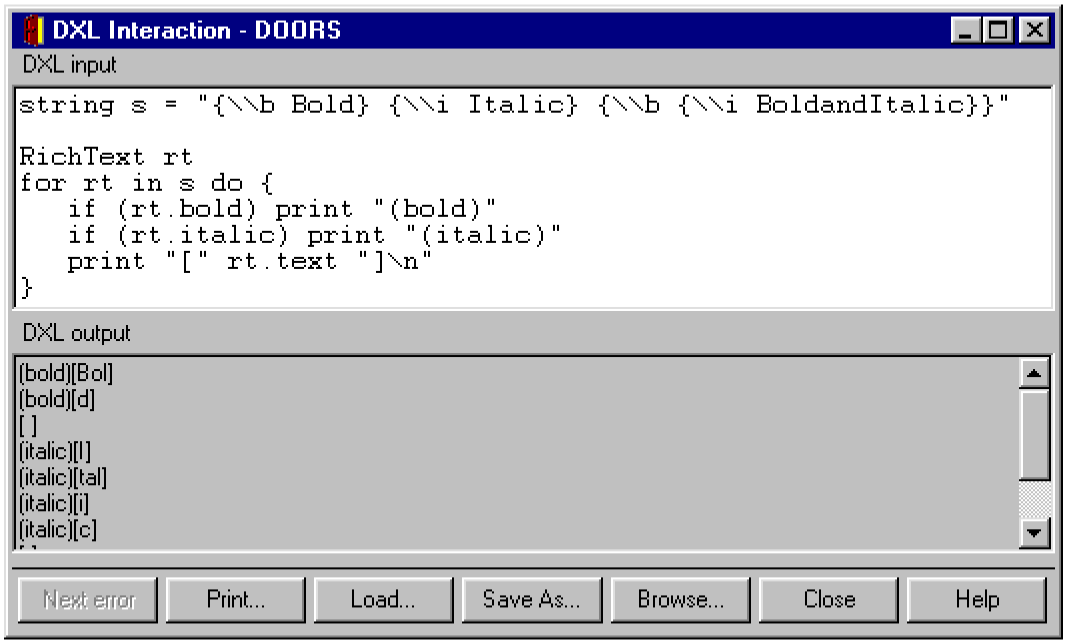
DOORS 5 - Example of "`for rt in s`" loop



```
string s = "{\\b Bold} {\\i Italic} {\\b {\\i BoldandItalic}}"

RichText rt
for rt in s do {
   if (rt.bold) print "(bold)"
   if (rt.italic) print "(italic)"
   print "[" rt.text "]\n"
}
```

```
(bold)[Bold]
[ ]
(italic)[Italic]
[ ]
(bold)(italic)[BoldandItalic]
```

DOORS 6 - Example of "`for rt in s`" loop



```
string s = "{\\b Bold} {\\i Italic} {\\b {\\i BoldandItalic}}"

RichText rt
for rt in s do {
   if (rt.bold) print "(bold)"
   if (rt.italic) print "(italic)"
   print "[" rt.text "]\n"
}
```

```
(bold)[Bol]
(bold)[d]
[ ]
(italic)[I]
(italic)[tal]
(italic)[i]
(italic)[c]
```

This is a known bug in DOORS 6.x that has been fixed in DOORS 7.

The "Normalize Rich Text Markup" algorithm will still work with this bug, but it will turn this artificial segmentation into actual segmentation, creating more Rich Text "chunks" than actually necessary.

Exporting Rich Text to Microsoft Office Applications

The advantage of adopting the Microsoft RTF standard is that formatted text and OLE objects (graphics) can be easily transferred between DOORS and Microsoft Office Applications (Word, Powerpoint, etc.).

Microsoft Word

DOORS 4.0 thru 5.2 - copying Rich Text to Microsoft Word

```
string s = richText( o."Object Text" )
setRichClip( richText s, styleName )
oleMethod( objSel, cMethodPaste )
```

The DOORS `setRichClip()` function above copies the unprocessed Rich Text Markup string directly from the Rich Text string variable to the Windows clipboard, with the desired target Style Name . Then, the Microsoft Visual Basic for Applications method "Paste" is used to paste the Rich Text Markup into the Word document.

DOORS 4.0 thru 5.2 - Copying OLE Objects to Microsoft Word

If a DOORS Object contains an OLE Object, the DOORS `oleCopy()` command will copy it to the Windows clipboard. It can then be pasted into the Word document using the "Paste" method as described above.

DOORS 6 and above - copying Rich Text to Microsoft Word

```
string s = richText( o."Object Text" )
setRichClip( richText s )
oleMethod( objRange, cMethodPaste )
olePut( objRange, cPropertyStyle, styleName )
```

DOORS 6 and above - Copying OLE Objects to Microsoft Word

OLE Objects are now embedded in the RTF stream, so no separate `oleCopy()` is necessary.

Microsoft Word - Paragraph vs. Character Style

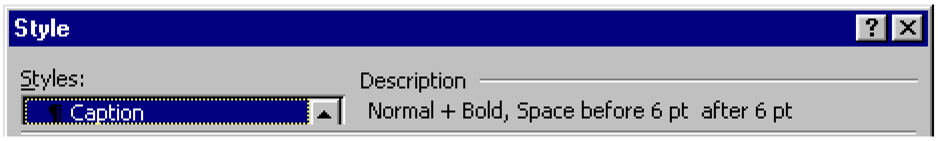
Styles are the architecture upon which Word is based. There are two types of styles in Microsoft Word; ***character*** and ***paragraph***.

- A Paragraph Style is a set of pre-defined formatting instructions that you can use repeatedly throughout the document.
- Character styles can be applied to individual words, or even single characters.

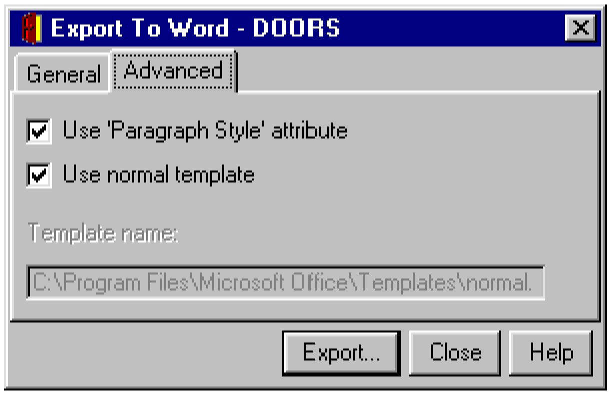
A paragraph style contains both font and paragraph formatting which makes it more comprehensive than a character style. When you apply a paragraph style the formatting affects the entire paragraph. For example, when you center text, you cannot center a single word. Instead, the entire paragraph is centered. Other types of paragraph-level formats that paragraph styles control are: line spacing (single-space, double-space, etc.), text alignment, bullets, numbers, indents, tabs and borders.

DOORS (v4 and above) rely on embedded Rich Text Markup to specify font effects (bold, italics, etc.), so when such text is pasted from DOORS and mapped to a Paragraph Style that also controls font markup, conflicts can arise.

For example, export formatted text to Microsoft Word using the "Caption" Paragraph Style built into the "normal.dot" template.

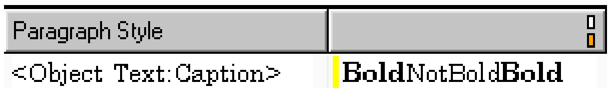


Styles: Caption
Description: Normal + Bold, Space before 6 pt after 6 pt



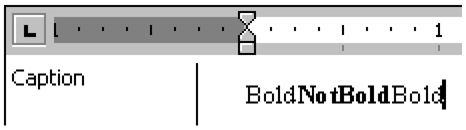
[✓] Use 'Paragraph Style' attribute
[✓] Use normal template
Template name: C:\Program Files\Microsoft Office\Templates\normal.

Formatted text in DOORS, with Paragraph Style attribute set to export "Object Text" to the "Caption" Style in Word



Paragraph Style	
<Object Text:Caption>	**Bold**NotBold**Bold**

Resulting export to Word, showing undesired reversal of intended font effects.

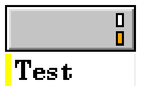


Caption Bold**NotBold**Bold

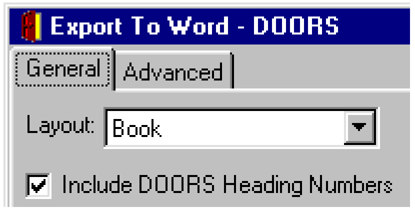
To avoid this situation, avoid exporting to Paragraph Styles in a Microsoft Word template that set font effects.

Microsoft Word - Updated DOORS 6 Export to Word

DOORS "Object Text" with Rich Text Markup (bolded text).



Export to Microsoft Word, using the built-in DOORS 6.0 SR1 "Export to Word".

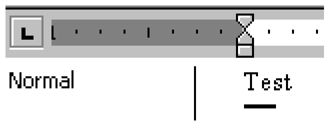

Export To Word - DOORS

General | Advanced

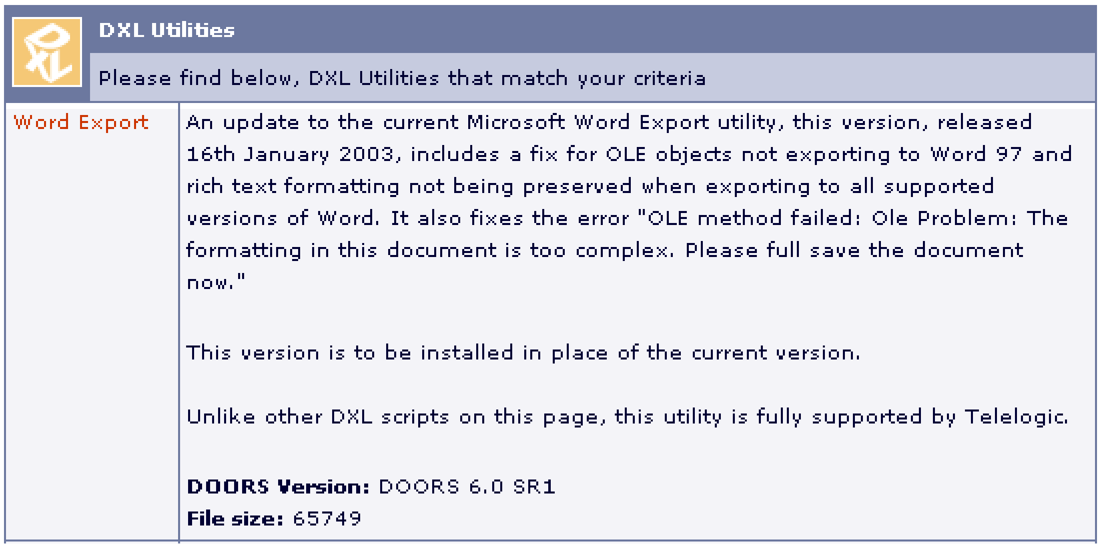
Layout: Book

☑ Include DOORS Heading Numbers

Resulting output in Microsoft Word, with incorrect Rich Text Format representation (non-bolded text) shown.



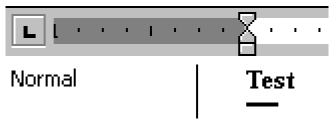
This is a known bug in the built-in DOORS 6.x "Export to Word". Telelogic has provided a fixed version, available for download at http://support.telelogic.com/en/doors/dxl/


DXL Utilities

Please find below, DXL Utilities that match your criteria

Word Export	An update to the current Microsoft Word Export utility, this version, released 16th January 2003, includes a fix for OLE objects not exporting to Word 97 and rich text formatting not being preserved when exporting to all supported versions of Word. It also fixes the error "OLE method failed: Ole Problem: The formatting in this document is too complex. Please full save the document now." This version is to be installed in place of the current version. Unlike other DXL scripts on this page, this utility is fully supported by Telelogic. **DOORS Version:** DOORS 6.0 SR1 **File size:** 65749

Resulting output in Microsoft Word, with correct Rich Text Format representation (bolded text) shown.



Export to Microsoft Excel

When Rich Text Format strings are copied from DOORS to an Excel Worksheet Cell, formatting is stripped and the resulting unformatted text is pasted directly into the cell by Excel.

DXL to paste Rich Text string from DOORS "Object Text" to Excel

```
string s = richTextNoOle( o."Object Text" )
olePut( objCell, "Value", s )
```

Then, using the "`for rt in s do`" construct described earlier, the desired Rich Text Markup can be rebuilt in Excel by scanning each segment as stored in DOORS, and duplicating the formatting using Excel Visual Basic for Applications.

```
void excelSetCellRichText( OleAutoObj objCell, string cellRichTextContents ) {

    OleAutoObj objFont = null

    int characterCount = 1
    RichText rt
    for rt in cellRichTextContents do {

        objFont = excelGetFontVBA( objCell, characterCount, length( rt.text ) )

        if ( rt.charset == charsetSymbol ) excelSetFontSymbolVBA( objFont )

        if ( !( rt.bold ) && !( rt.italic ) ) excelSetFontRegularVBA( objFont )
        if (  ( rt.bold ) && !( rt.italic ) ) excelSetFontBoldVBA( objFont )
        if ( !( rt.bold ) &&  ( rt.italic ) ) excelSetFontItalicVBA( objFont )
        if (  ( rt.bold ) &&  ( rt.italic ) ) excelSetFontBoldItalicVBA( objFont )

        if ( rt.underline ) {
            excelSetFontUnderlineSingleVBA( objFont )
        }
        else {
            excelSetFontUnderlineNoneVBA( objFont )
        }

        if ( rt.strikethru ) excelSetFontStrikeThroughVBA( objFont )
        if ( rt.superscript ) excelSetFontSuperscriptVBA( objFont )
        if ( rt.subscript ) excelSetFontSubscriptVBA( objFont )

        characterCount += length rt.text
    }
}
```

The above code is from a free DXL utility called "Enhanced Export to Excel", available from http://galactic-solutions.com and presented at the Telelogic Americas 2002 Users Group Conference.

Appendix:

DOORS 4.1 - Updates

New functions added to the "Rich text functions" section:	
string removeUnlistedRichText(string s)	Removes rich text markup which DOORS does not recognize. Recognized markup is: italic, bold, underline, strikethru, font, style.
New functions added to the "Rich text string processing" section:	
string stringOf(RTF_string__ richString)	allows access to rich text as a string.
RTF_string__ richClip()	gets the rich text contents of the system clipboard as a rich text string
void setRichClip(RTF_string__ richString) void setRichClip(RTF_string__ richString, string styleName)	Sets the system clipboard to be the rich text string parameter , or, can also include a minimal RTF stylesheet that contains a supplied style name which consequently sets the string style.
bool pasteToEditbox()	pastes the contents of the clipboard into an module object that is ready for in-place editing. Returns false if paste fails.
New functions added to the "Enhanced character support" section:	
int charsetDefault()	returns the system default character set. On UNIX platforms, this will always be charsetAnsi. On Windows systems, the user's local setting is returned.
charset()	
string fontTable(Module mod)	returns the module's font table, which is used for mapping richtext font markup to character set information.
New functions added to the "Importing rich text" section:	
importRTF	
New "Support Tools" added to the Tools menu of DOORS Formal Modules with the release of DOORS 4.1.	
<standard/doctools/normmark.dxl>	"Remove Unsupported Markup"
<standard/doctools/delmark.dxl>	"Remove All Markup" (Rich Text mark-up Deletion Tool)
<standard/doctools/symbconv.dxl>	"Convert Symbols to Text" (Symbol to Plain Text Conversion Tool)

DOORS 5.0 - Minor Updates

string cutRichText(string s, int start, int end)	Returns the string s with the displayed characters from start to end removed. For the purposes of counting characters, rich text markup is ignored, and markup is preserved.
string exportRTFString(string s)	Translates a DOORS rich text string to the RTF standard. The only difference is that tab characters are replaced by \tab and newline characters are replaced by \newline.

DOORS 6.0 - Major Update

string exportRTFString(string s)	optional arguments added; font information is always included; newlines are converted into /par tags instead of /newline tags
richtext_identifier	returns the object identifier as an RTF string
~~stripRTFheaders~~ richTextFragment	returns the content of the passed string with any font color or other RTF header information removed
richTextWithOle	returns rich text of specified attribute, including OLE objects
richTextNoOle	returns rich text of specified attribute, excluding OLE objects
richTextWithOle(column)	returns the text contained in specified column as rich text, including OLE objects
richTextNoOle(column)	returns the text contained in specified column as rich text, excluding OLE objects
RichTextParagraph type	Data type to handle looping through paragraphs with bullets and indents
"Support Tools" removed from the Tools menu of DOORS Formal Modules with the release of DOORS 6.0.	
~~<standard/doctools/normmark.dxl>~~	"Remove Unsupported Markup"
~~<standard/doctools/delmark.dxl>~~	"Remove All Markup" (Rich Text mark-up Deletion Tool)
~~<standard/doctools/symbconv.dxl>~~	"Convert Symbols to Text" (Symbol to Plain Text Conversion Tool)

DOORS 7.0 - Minor Update

void useRTFColour(DBE dbe, bool useRTF)	Use the rtf colour markup instead of the default colour for text in dialog boxes.
string applyTextFormattingToParagraph(string s, bool addBullets, int indentLevel, int paraNumber)	Applies bullet and/or indent style to the given text, overwriting any existing bulleting/indenting.
string exportRTFString (sring text [, int level [, bool isHeading]])	The appropriate font information is included when the supplied plain string text is rendered as rich text.
string richTextFragment(string richString) string richTextFragment(string richString, string fontTable)	Returns an equivalent representation of the rich text with rtf header information removed.

References:

Microsoft Corporation Rich Text Format (RTF) Specification, version 1.6
(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec.asp)

Microsoft Corporation RTF Specification to Word 97 (RTF version 1.4 or later
(http://download.microsoft.com/download/word97win/spec2/1/WIN98/EN-US/GC1282.exe)

Microsoft Corporation Rich-Text Format (RTF) Specification, version 1.0
(http://www.nist.fss.ru/hr/doc/spec/rtf1.htm) or (http://www.nwsta.com/Soft/ntd/spec/rtf1.php)

Obtaining Software:

DOORS Users are encouraged to obtain, use, share, and improve upon the utilities mentioned in this presentation.

For a free copy:

Contact: michael@galactic-solutions.com, or download from http://galactic-solutions.com.