Telelogic Americas 2002
User Group Conference

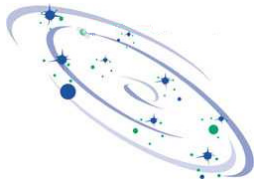
DOORS DXL -
Adventures in Microsoft OLE Automation

Michael Sutherland

Galactic Solutions Group LLC

michael@galactic-solutions.com

Telelogic

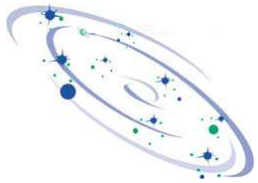


Excel Exporter

- Telelogic provides an Excel Exporter for DOORS

  `$DOORSHOME/lib/dxl/standard/export/office/excel.dxl`

- Functional, but does not export:
  - OLE Objects (graphics)
  - Rich Text
  - Outlining/Hierarchy
  - Color Columns and Object Heading row Color
  - Page Layouts (Headers, Footers, Paper Size, Paper Orientation, Margins)
  - Column Widths and Fixed Header Row
- To create an Enhanced Export to Excel for DOORS, knowledge of the DOORS API and Microsoft OLE Automation is required

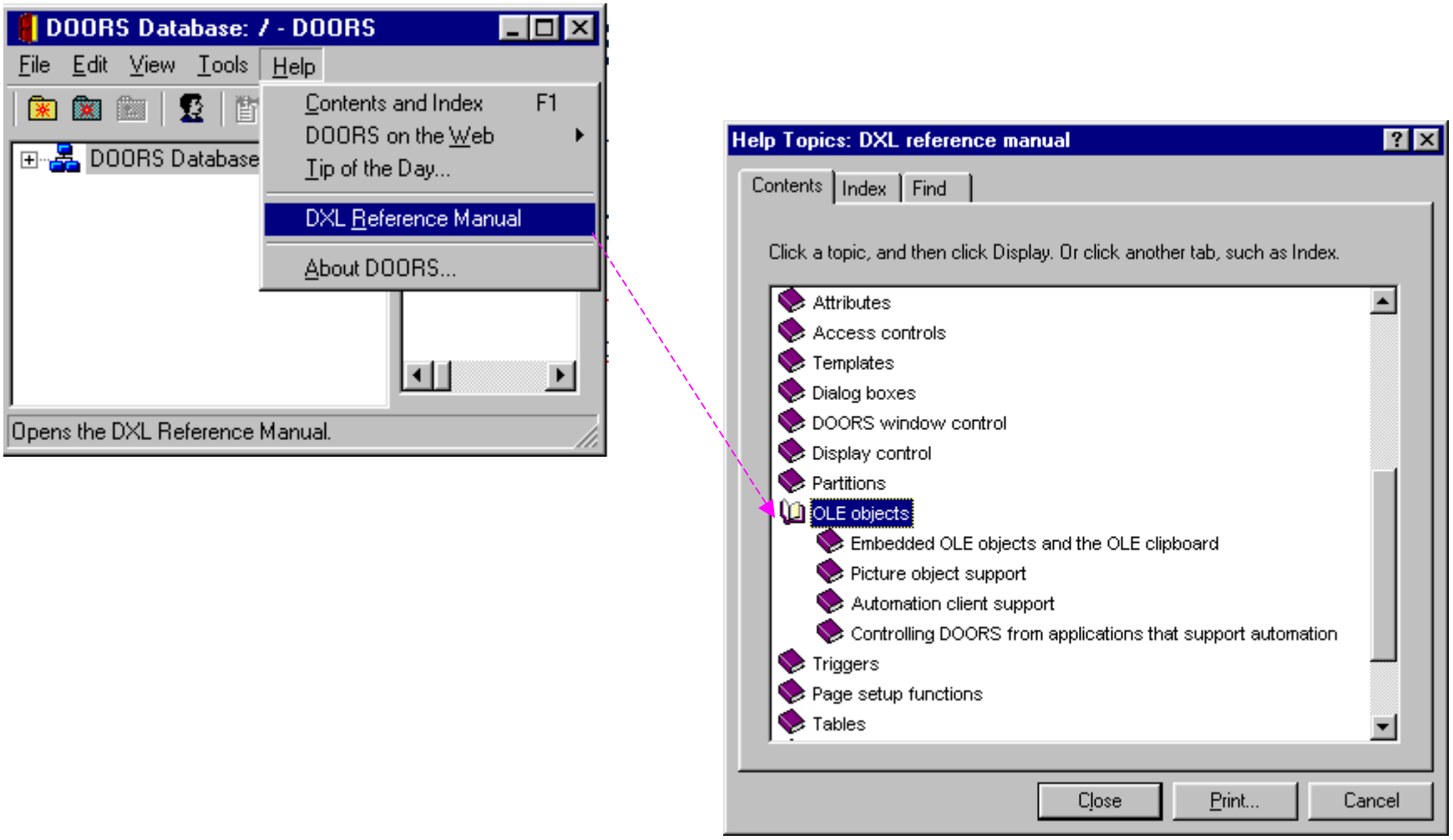


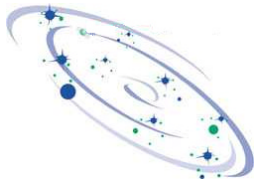
DOORS DXL (DOORS eXtension Language)

- DXL is the Application Program Interface (API) for DOORS

- DXL is a “macro” language to:
  - Automate repetitive tasks
  - Manipulate Database information
  - Create new user interactions
    - (dialogues, forms, events, etc.)

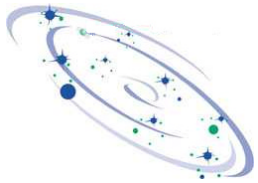
DOORS DXL Reference



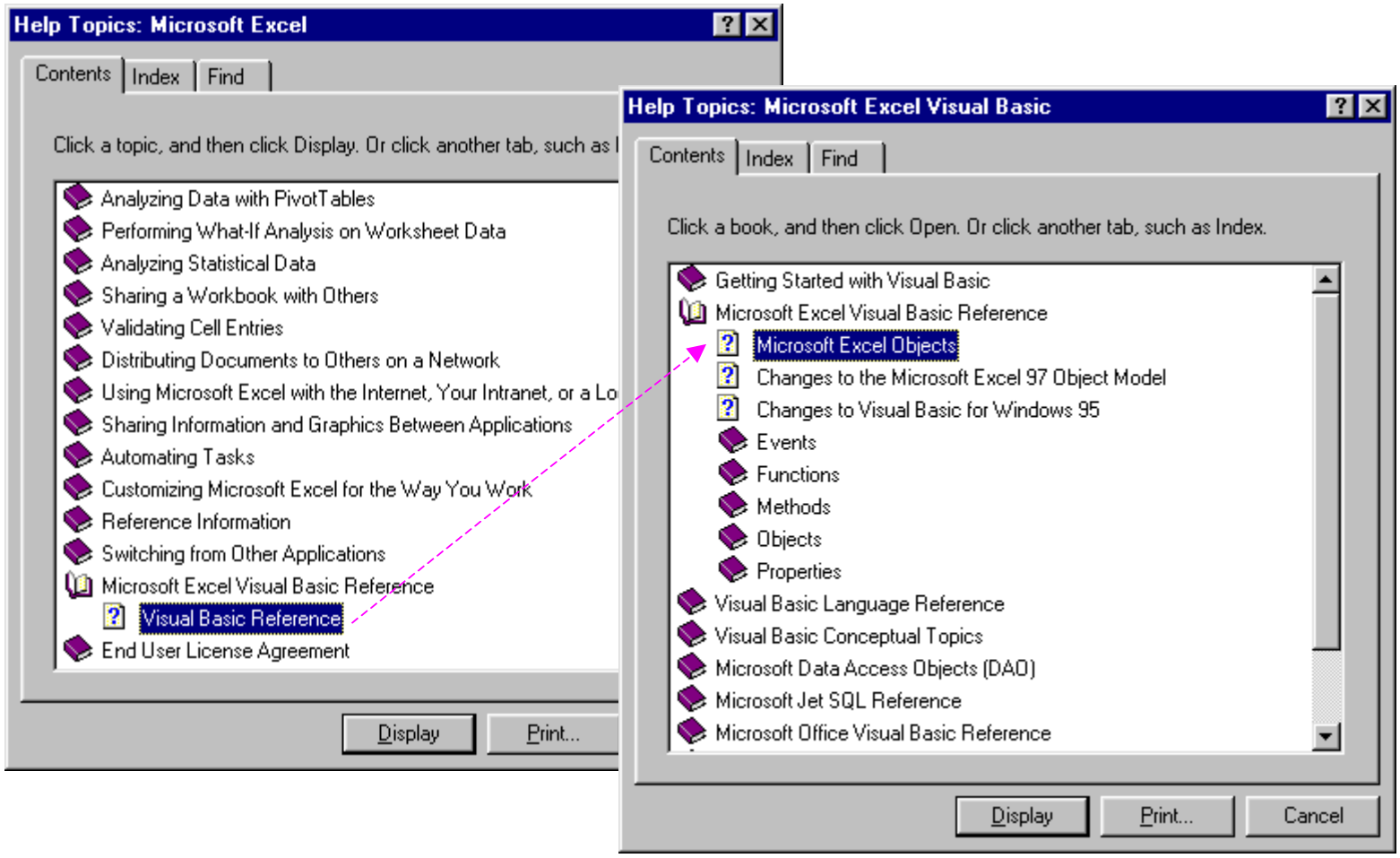


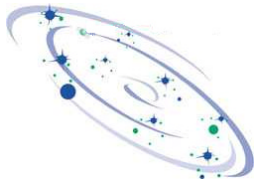
Microsoft OLE Automation

- Uses Component Object Model (COM)
- Method used in Microsoft Windows Operating System to communicate with Windows Applications
- Allows referencing of another Microsoft Windows Application's Objects, Properties and Methods
- Note: Microsoft Office 97 and above have dropped the "OLE" (Object Linking and Embedding) and called this "*Microsoft Automation*"
  - OLE still used to create Compound Documents



Microsoft Visual Basic Reference





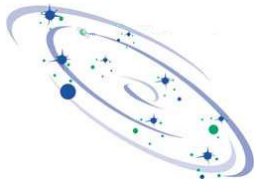
OLE Application Object References

- DOORS uses `OleAutoObj` to declare Object variables that reference Application Objects
  - Create an reference to the Server Application Object using the Application's "OLE Programmatic Identifier"

```
OleAutoObj objExcel =
    oleCreateAutoObject( "Excel.Application" )
```

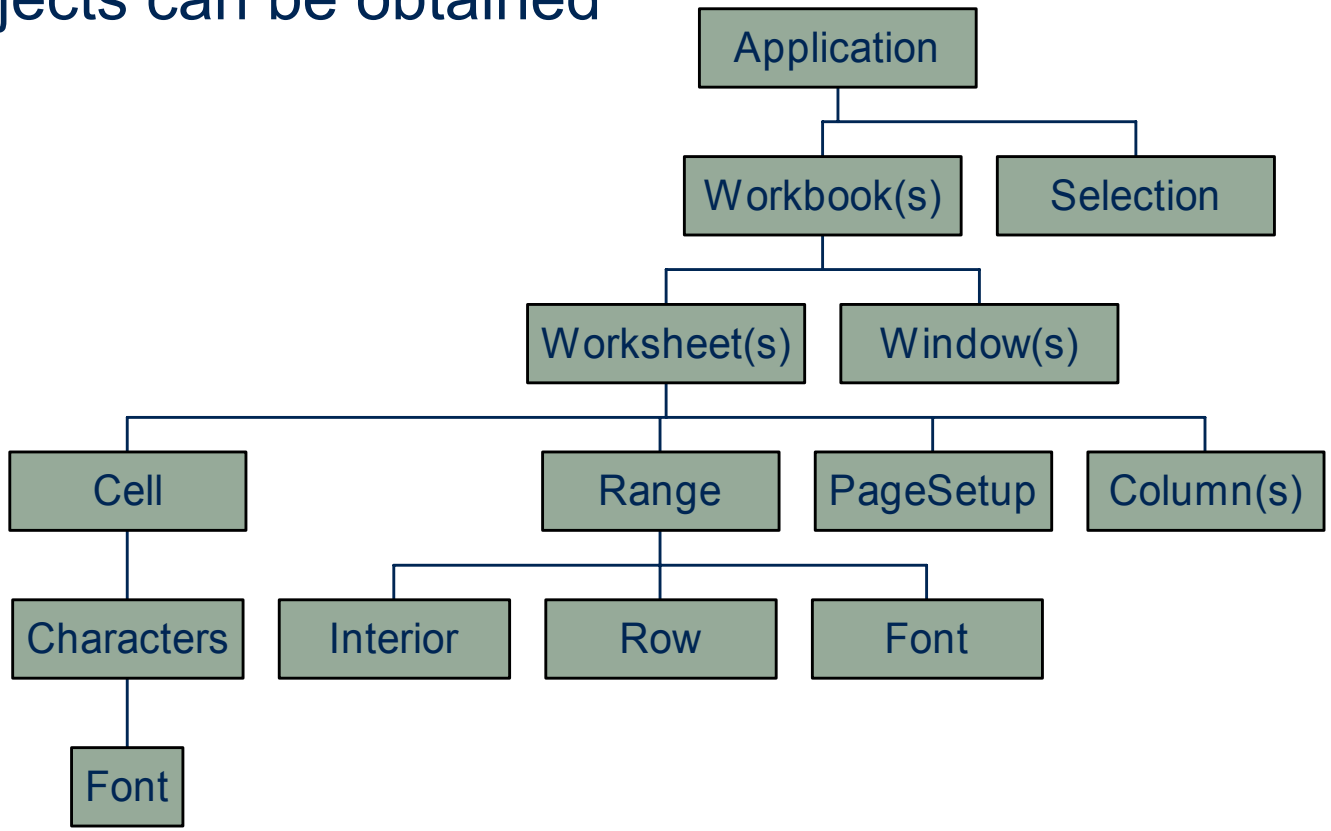
  - The Objects of the Application are now accessible, and the Properties and Methods of the Objects can be applied
  - When finished, close the Application

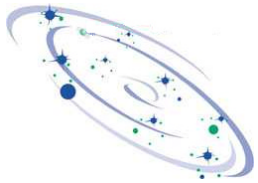
```
oleCloseAutoObject( objExcel )
```



Microsoft Excel Object Hierarchy

- Starting with the Excel Application, references to other Excel Objects can be obtained





Object Properties

- OLE Objects have Properties (Attributes)
- DOORS uses `oleGet` and `olePut` to access the properties of Objects
- Property can be of type:

  `string | int | bool | char | OleAutoObj`

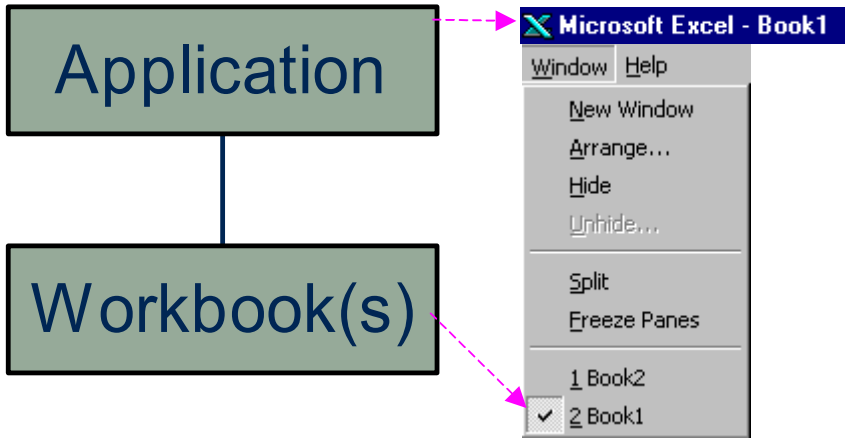
  Note: A few OLE properties are of type real, which DOORS cannot currently handle
- Getting a property uses a “return value” variable
- Example: Boolean property of the Excel Application

```
bool isVisible
oleGet( objExcel, "Visible", isVisible )
olePut( objExcel, "Visible", true )
```

Note (“**isVisible**” is the return value)

Obtaining Object Reference

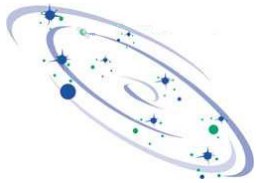
- One important Property of an Object is the Objects it contains



- Example: The Excel Application (v97) can have 0-255 Workbook(s) open. The Workbooks Collection Object can be obtained from parent Excel Application Object:

```
OleAutoObj objWorkbooks

oleGet( objExcel, "Workbooks", objWorkbooks )
```



Object Methods

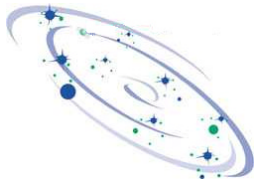
- Object have Methods, which are procedures or functions that act on the Object or transform data
- DOORS uses `oleMethod` to access Methods

- Example:

```
oleMethod( objWorkbooks, "Add" )
```

- Example: Sheet Activation

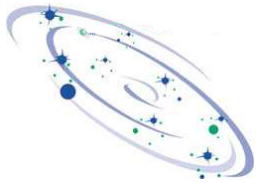
```
oleMethod( objSheet, "Activate" )
```



OleAutoArgs

- Properties and Methods sometimes requires arguments to be passed
- Declare and define `OleAutoArgs` variable
- Example: get first Sheet from Workbook Collection

```
OleAutoObj objSheet = null
OleAutoArgs objArgBlock = create
clear( objArgBlock )
put( objArgBlock, 1 )
oleGet( objWorkbook, "Sheets",
        objArgBlock, objSheet
delete objArgBlock
```



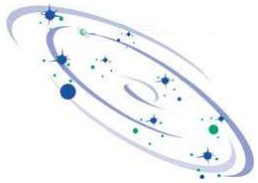
Object Methods with Arguments

- Example: Inches to Points

```
clear( objArgBlock )
put( objArgBlock, inches )
int points
oleMethod( objExcel, "InchesToPoints", objArgBlock, points )
```

- Example: Saving Changes to Workbook

```
clear( objArgBlock )
put( objArgBlock, "SaveChanges", true )
oleMethod( objWorkbook, "Close", objArgBlock )
```



DOORS Provided OLE Library

- DOORS MS Office Exporters uses

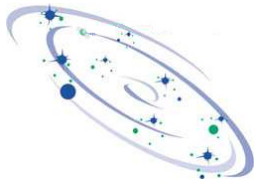
```
#include <utils/ole.inc>
```

- Contains constants for the names Properties, Methods, Parameters and values of Symbolic References

```
Objects:    const string cObjExcelApplication = "Excel.Application"
Properties: const string cPropertyRange       = "Range"
Methods:    const string cMethodSelect        = "Select"
Parameter:  const string cParamSaveChanges    = "SaveChanges”
Symbolic:   const int    xlActiveWindow       = 1
```

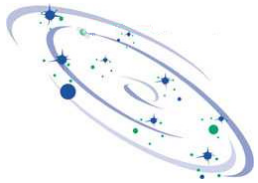
- Contains common functions to aid in communication with Microsoft OLE Applications

```
checkPlatform, closeIfNonNull, checkRes
makeVisible, connectToApp, disconnectFromApp
```



Experienced VBA Programmers

- Although DOORS allows access to all of an OLE Application's Objects, Properties, and Methods, the actual programming is done in the DOORS API (DXL) and not in Microsoft *Visual Basic for Applications* (VBA)
- Veteran VBA programmers will miss constructs which loop through collections of Objects such as:
  - **For Each...Next**: Loop through each Object in a Collection, and allows a group of statements to be executed for each Object in the Collection
    - Use "Count" Property and use "Item" Method to index into the Collection
  - **With**: Runs a series of statements on the same Object
- It is possible to execute VBA macros stored in an Application Library, although this was not necessary for the implementation of the Enhanced Excel Exporter



Running VBA Macros from DOORS

- Procedure to execute a Macro stored in an Excel workbook:

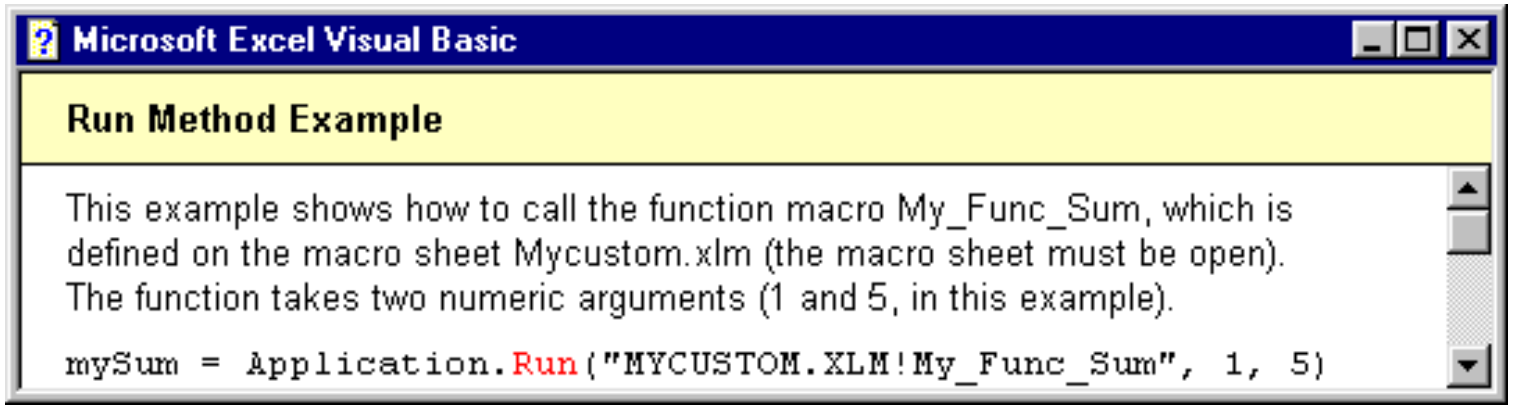
(1) Connect to Excel Application

(2) Get Workbooks Collection

(3) Add Workbook (file) containing Macro(s) to Workbooks Collection

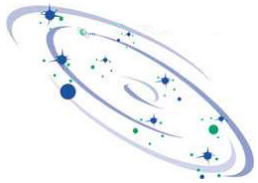
Note: This does not have to be the same Workbook file that data will be exported to

(4) Run Macro, passing arguments if necessary



- If this method is used, the Excel Workbook containing the Macro(s) must be distributed to all DOORS users and placed in the proper directory

Enhanced Export to Excel



Copying OLE Objects to Excel

(1) Copy OLE Object to Windows clipboard

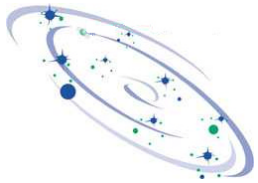
(2) Choose Target Sheet and Cell

(3) Set “Range” to single Cell

(4) Select Range

(5) Paste OLE Object

```
if ( oleCopy( o ) ) {
   put( objArgBlock, "C12" )
   OleAutoObj objRange = null
   oleGet( objSheet, "Range", objArgBlock, objRange ) )
   oleMethod( objRange, "Select" )
   put( objArgBlock, "Link", false )
   put( objArgBlock, "DisplayAsIcon", false )
   oleMethod( objSheet, "PasteSpecial", objArgBlock )
}
```

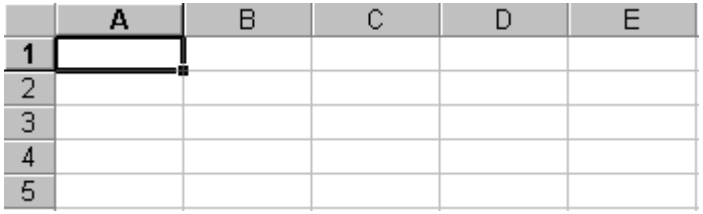


OLE Objects on Sheet

- OLE Objects reside on Worksheet, **not** in a cell
- OLE Objects are associated with a Cell (upper-left corner of OLE Object)
- OLE Objects are not strongly attached to cell, and can be easily moved around the sheet
- Sheet is formatted so that OLE Object initially fits within cell

DOORS and Excel Columns

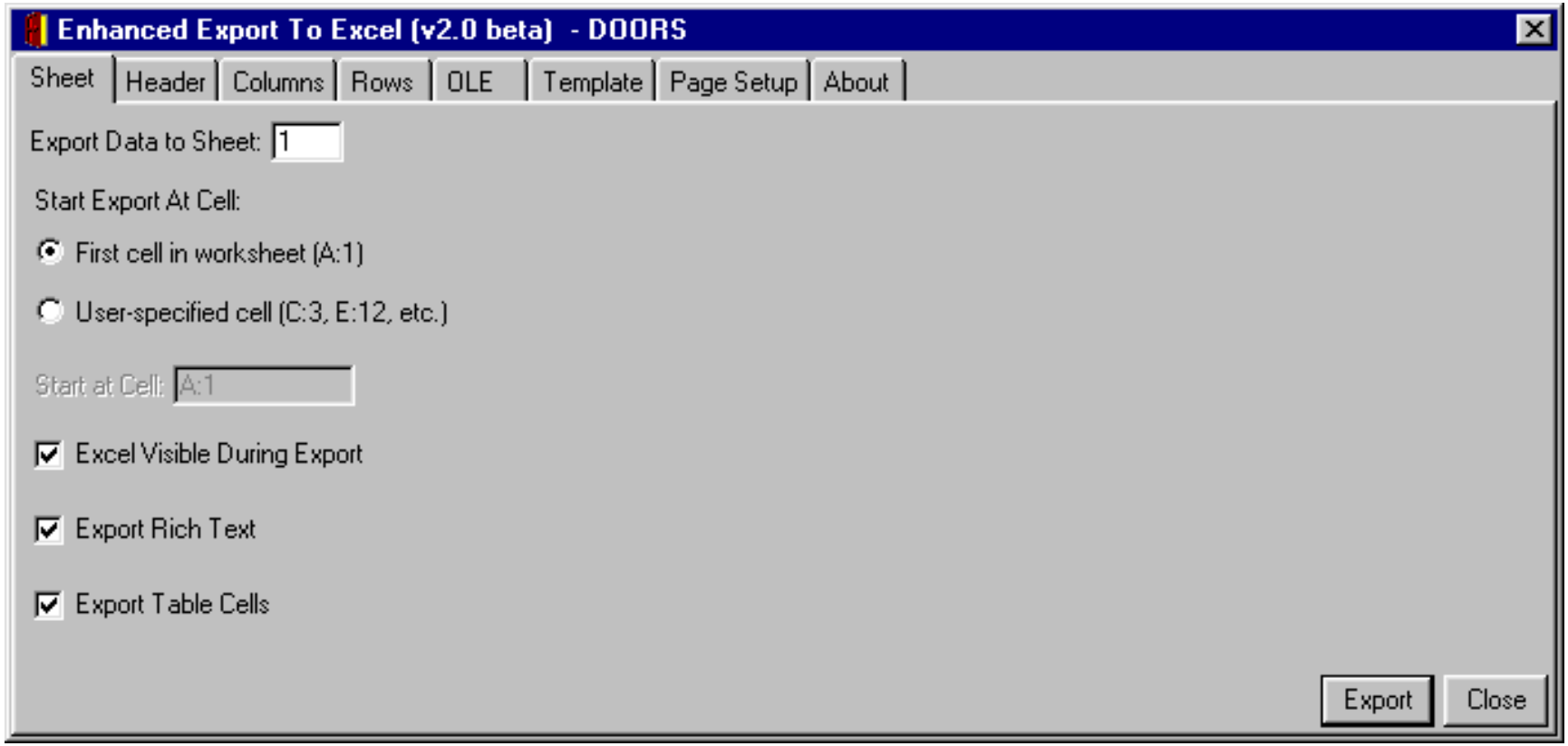
- Excel uses letters to enumerate Columns



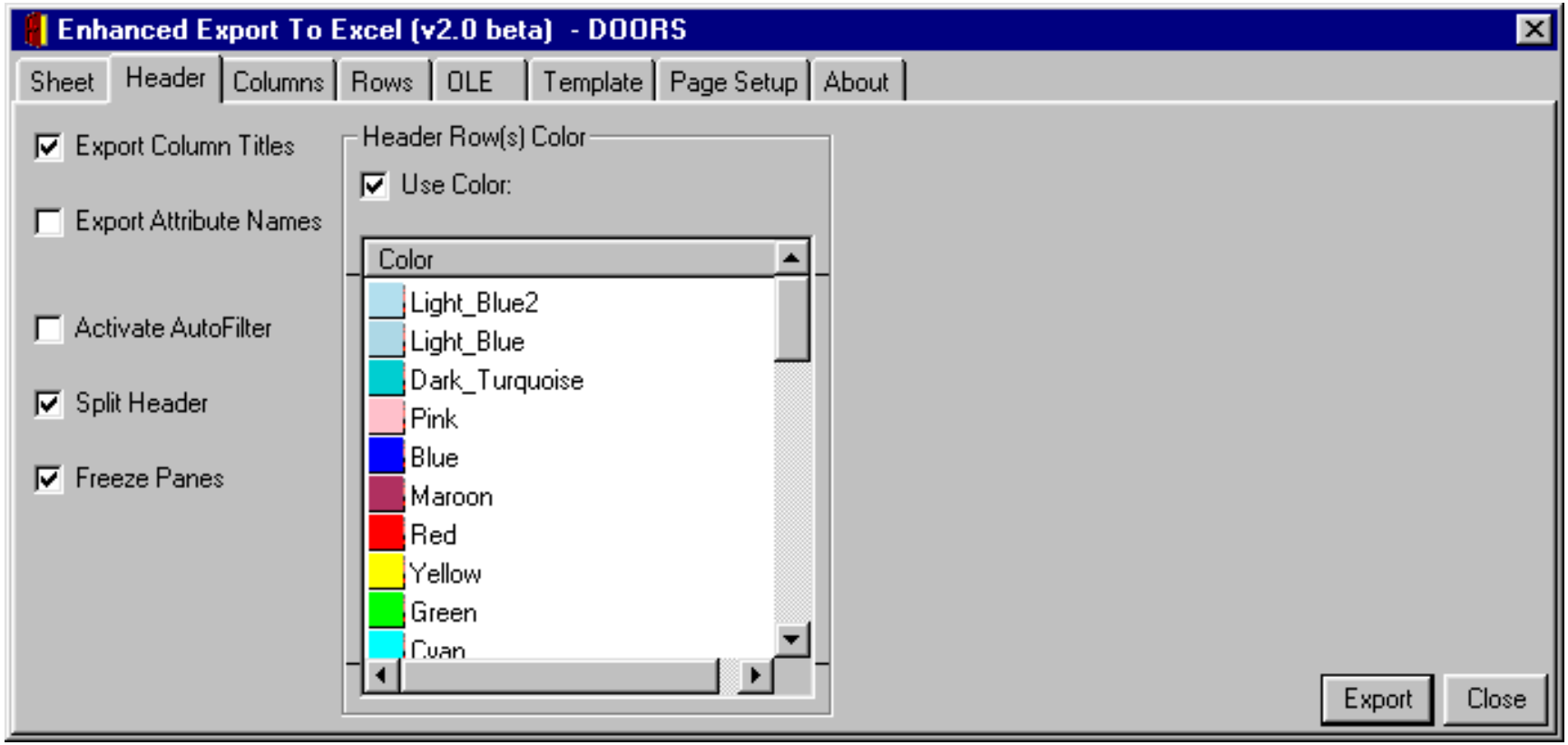
	A	B	C	D	E
1					
2					
3					
4					
5					

- Existing DOORS Excel Exporter will not export more than 26 Columns (“A” - “Z”)
  - A DOORS Module allows 32 Columns
  - Excel 97 allows 256 Columns (“A” - “IV”)
- Routines have been enhanced to allow for 32 Column Export with offset
  - Export need not start at cell “A1”, start cell is user defined

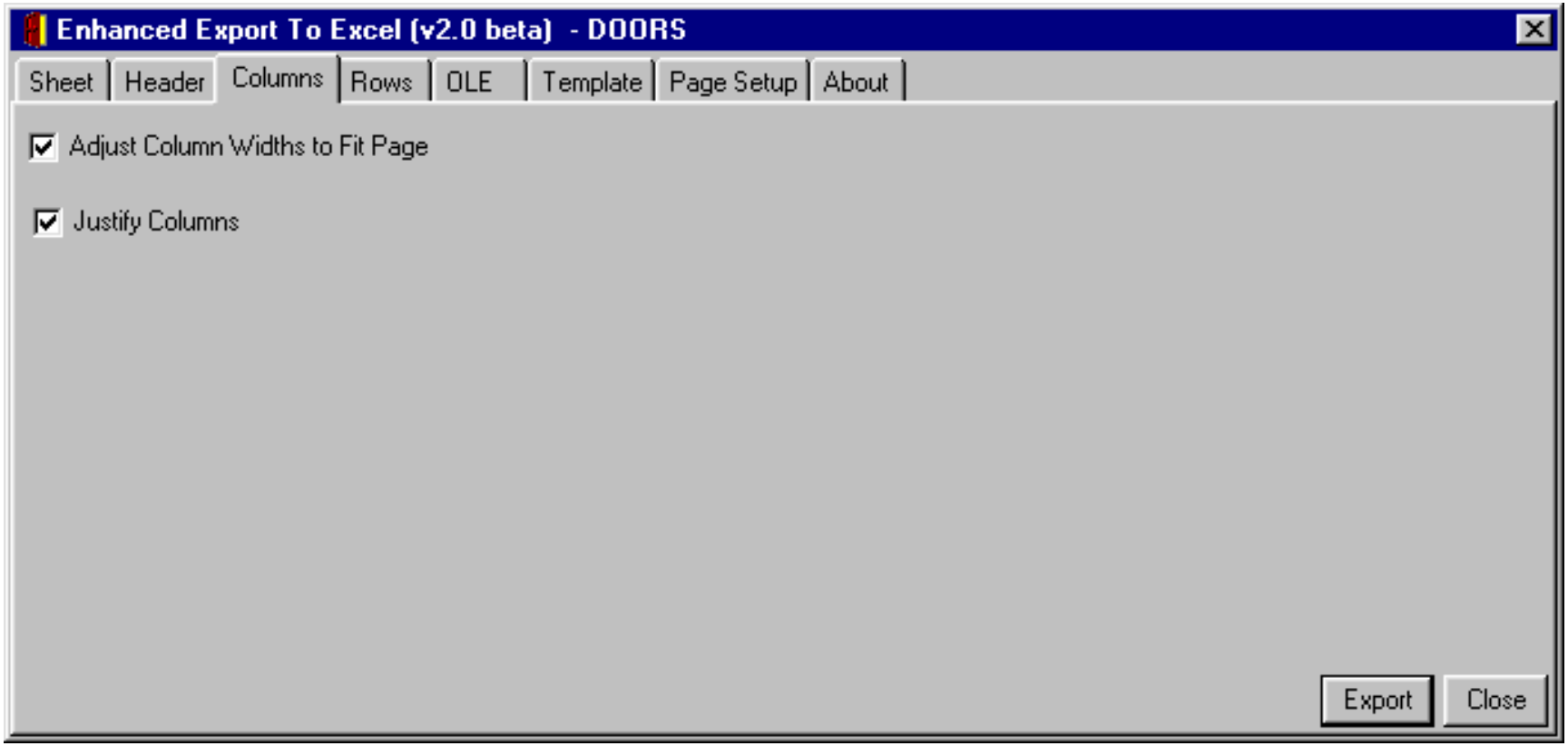
Enhanced Export to Excel - Sheet



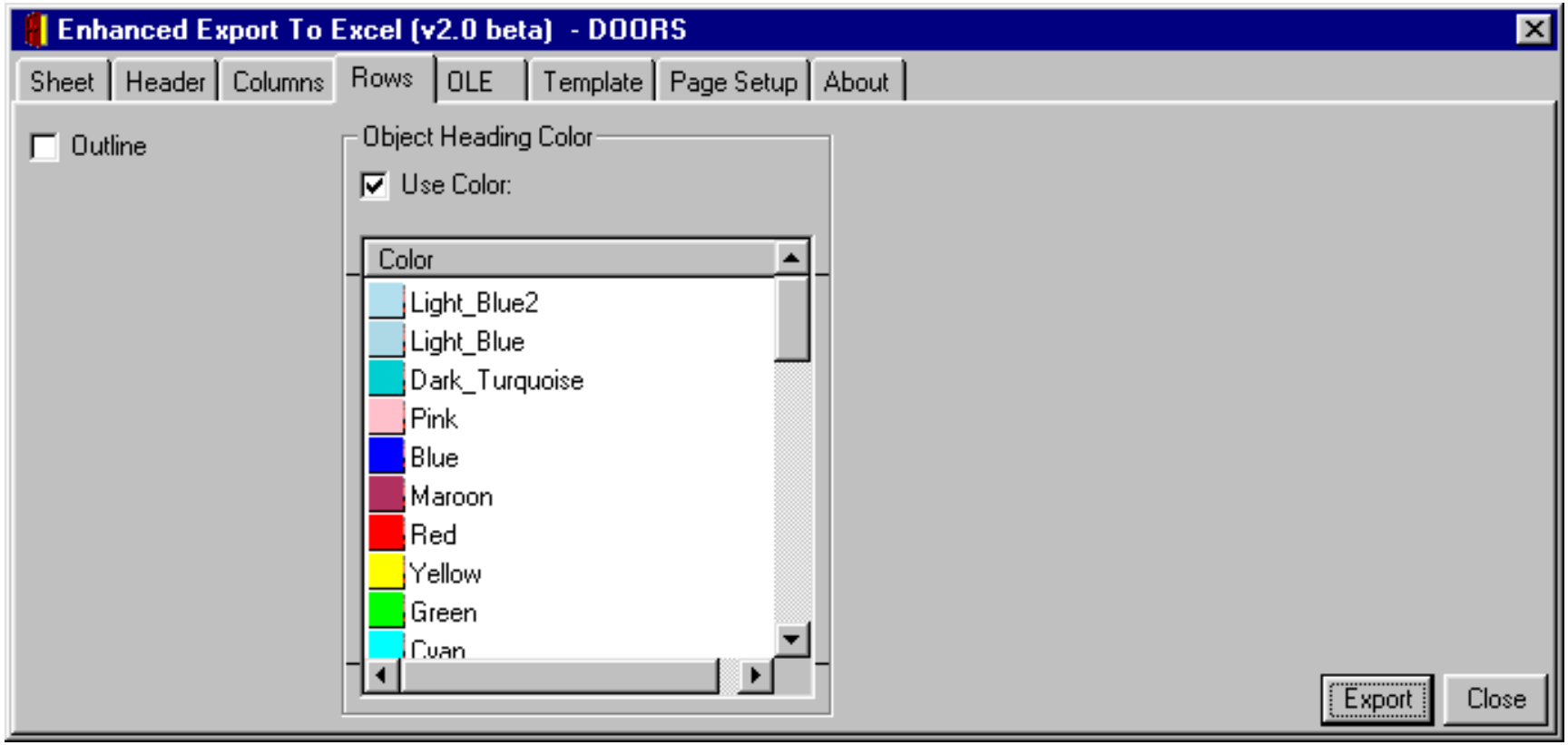
Enhanced Export to Excel - Header



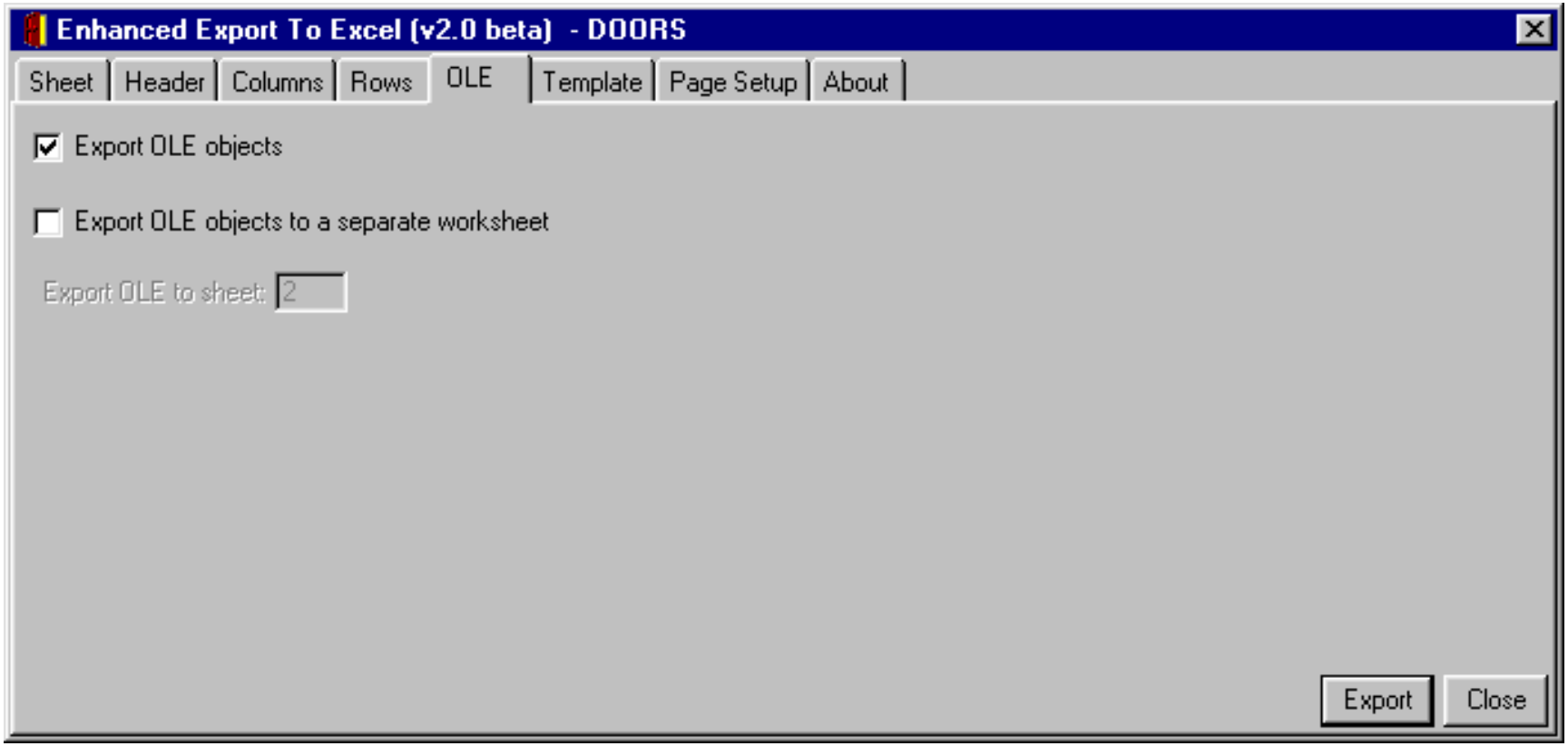
Enhanced Export to Excel - Columns



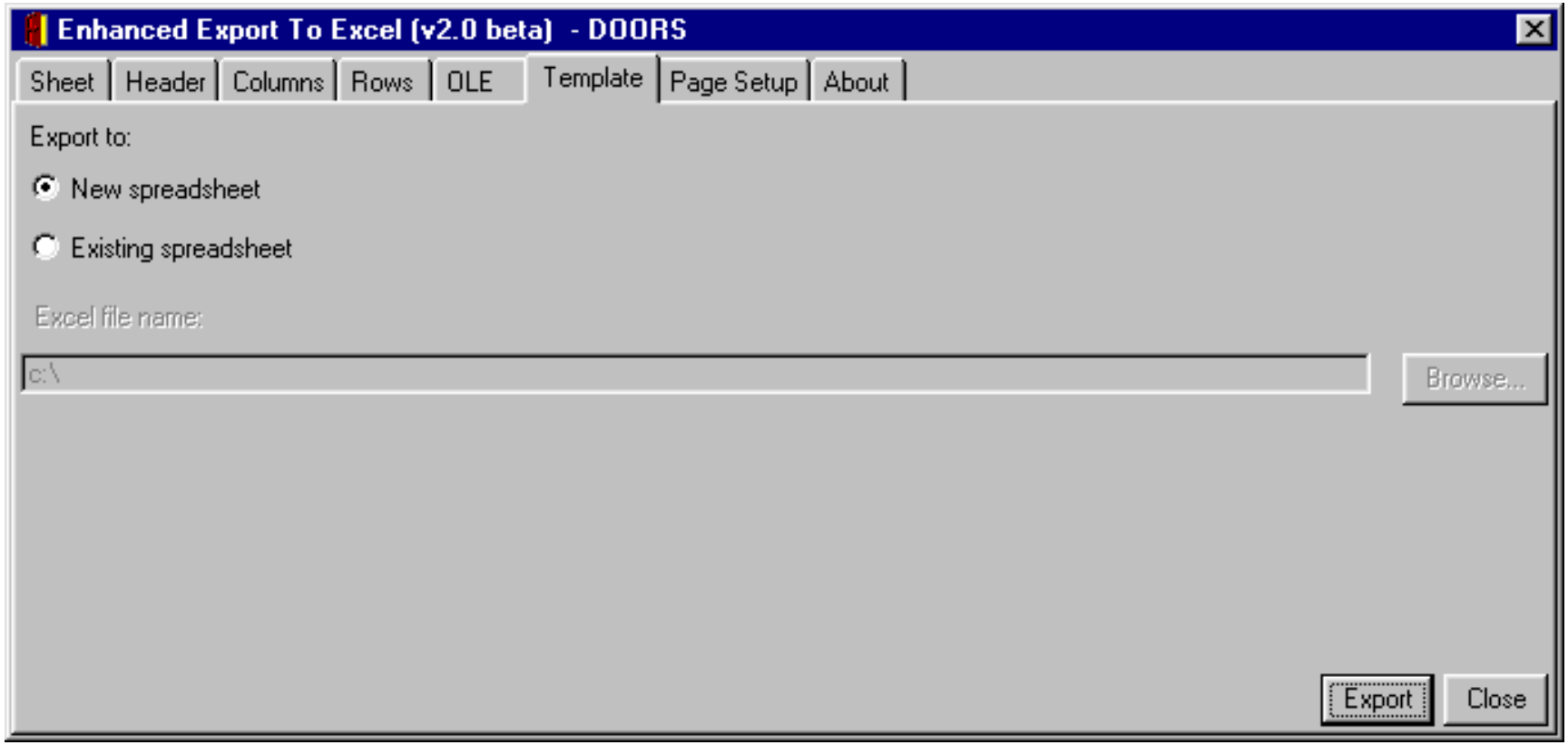
Enhanced Export to Excel - Rows



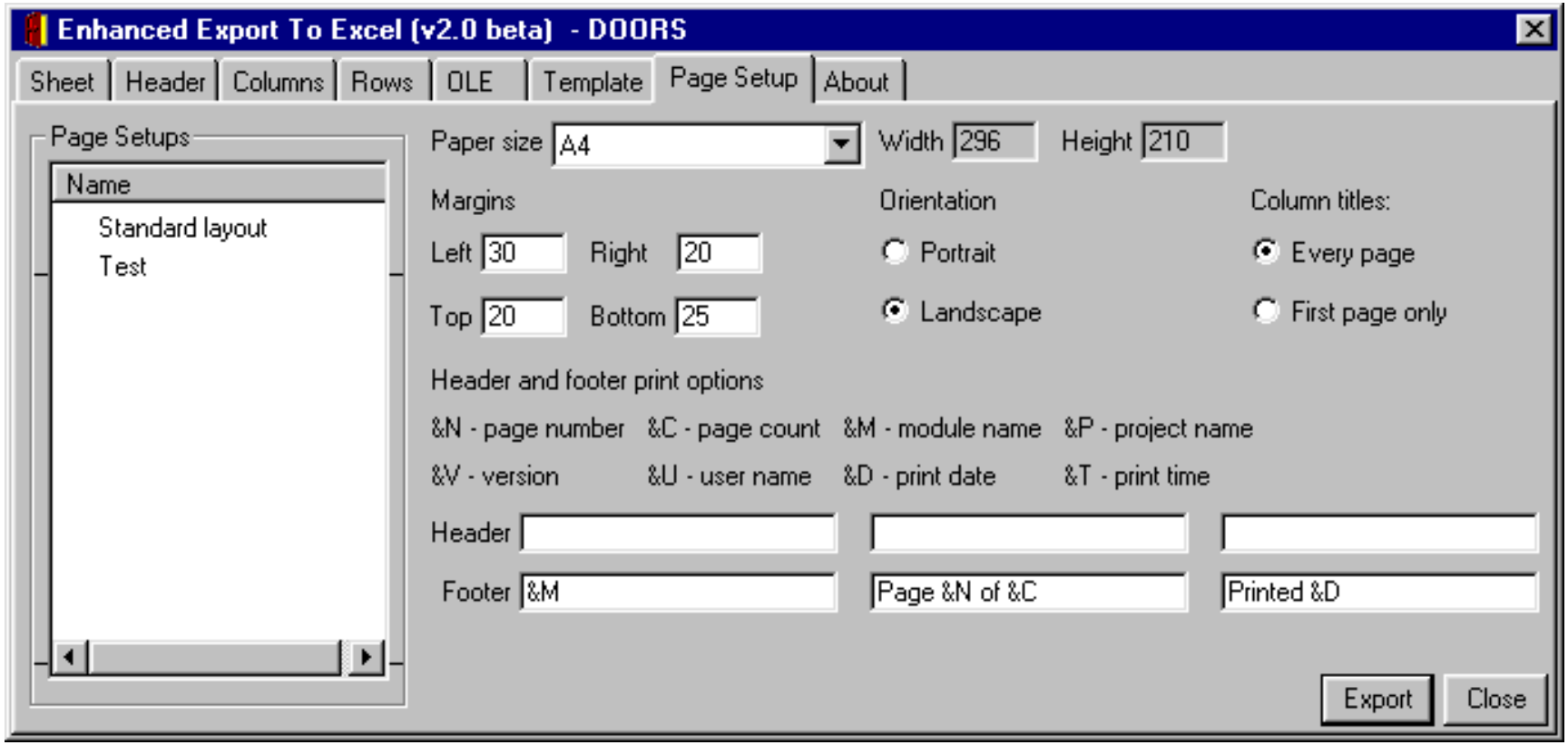
Enhanced Export to Excel - OLE



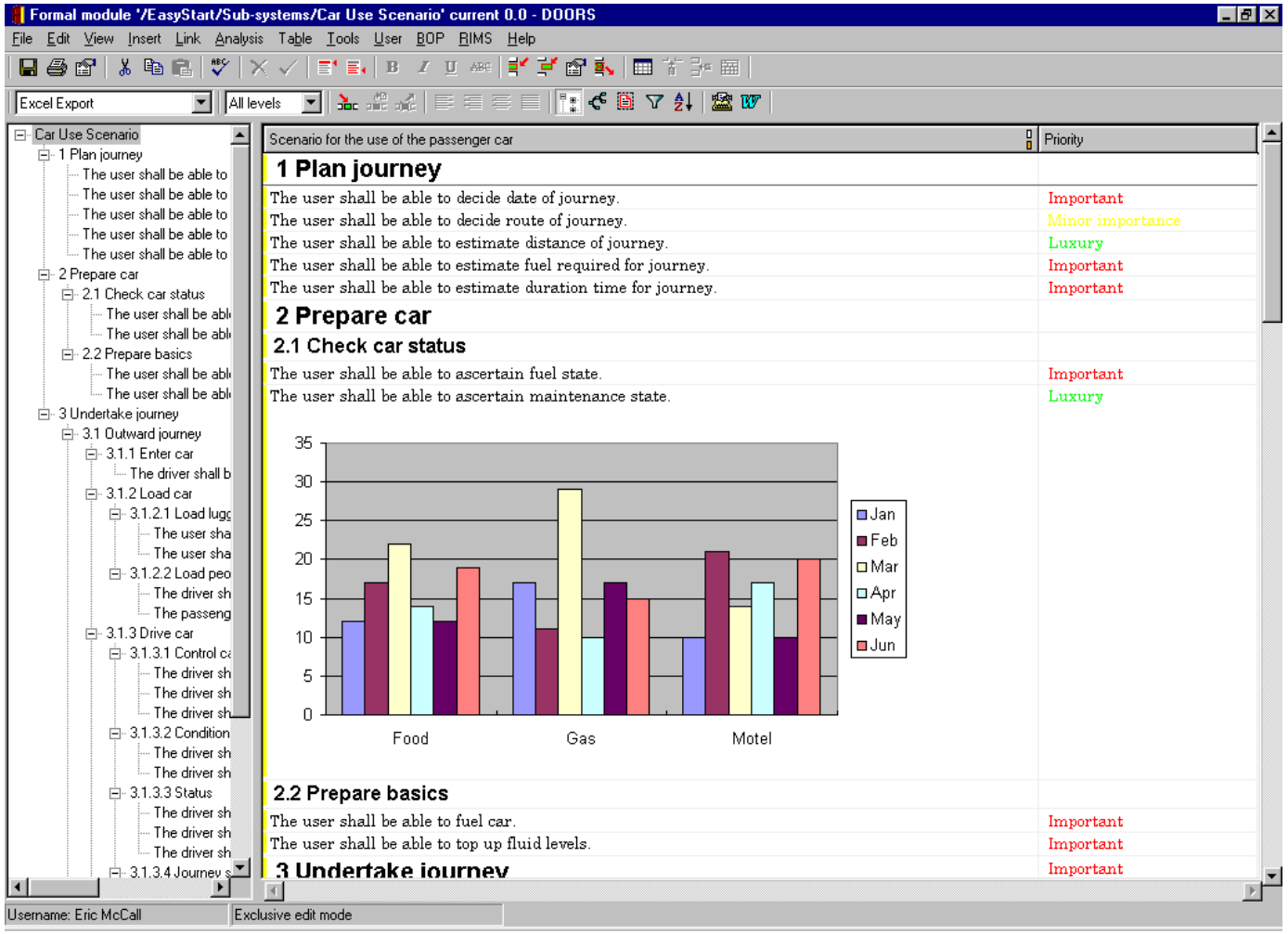
Enhanced Export to Excel - Template



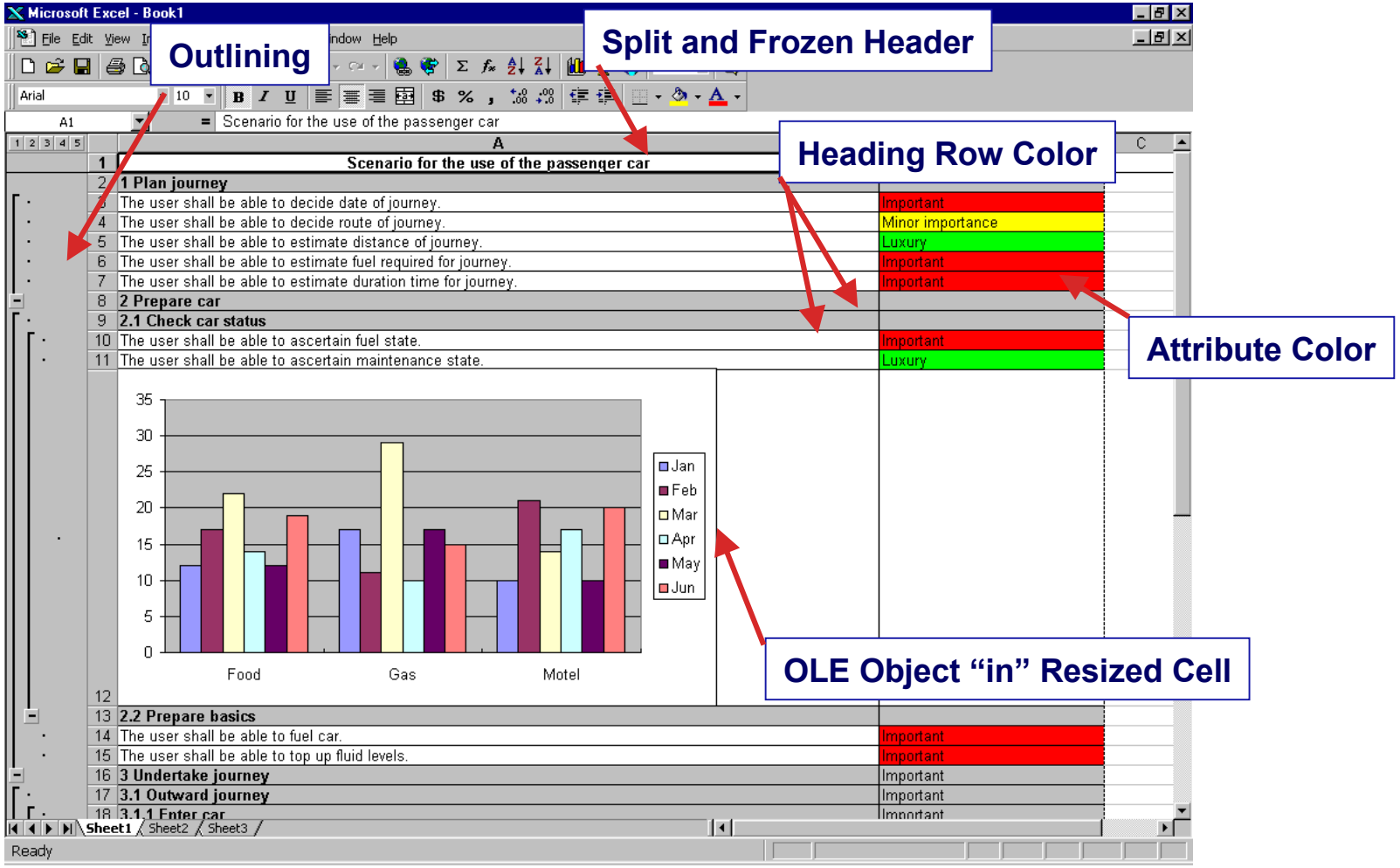
Enhanced Export to Excel - Page Setup

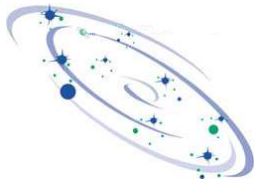


DOORS Module to Export



DOORS Module Exported to Excel





Issue - DOORS Tables

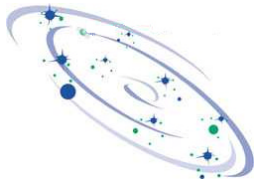
- What to do with DOORS Tables?
  - Current DOORS Excel Exporter “linearizes” them (Table Cells are exported in row order as equal children)

1 Heading One

Cell 1	Cell 2
Cell 3	Cell 4

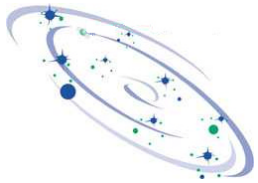
	A
1	1 Heading One
2	Cell 1
3	Cell 2
4	Cell 3
5	Cell 4

- Proposed solution:
  - Export as Word Table, embed into Excel as OLE Object



References

- Microsoft Visual Basic for Applications
  - VBA for Dummies®, 3rd Edition http://www.dummies.com/extras/vba_fd_3e/
  - Introduction to Office Automation - Sheffield Hallam University http://maths.sci.shu.ac.uk/units/ioa/
  - Microsoft Developer Network http://msdn.microsoft.com
- Microsoft Excel Automation
  - Jwalk & Associates, The Spreadsheet Page http://j-walk.com/ss
  - Pearson Software Consulting, LLC http://www.cpearson.com/excel.htm



Obtaining a copy of the Software

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

- For a free copy:

Contact: michael@galactic-solutions.com

or download from

http://galactic-solutions.com