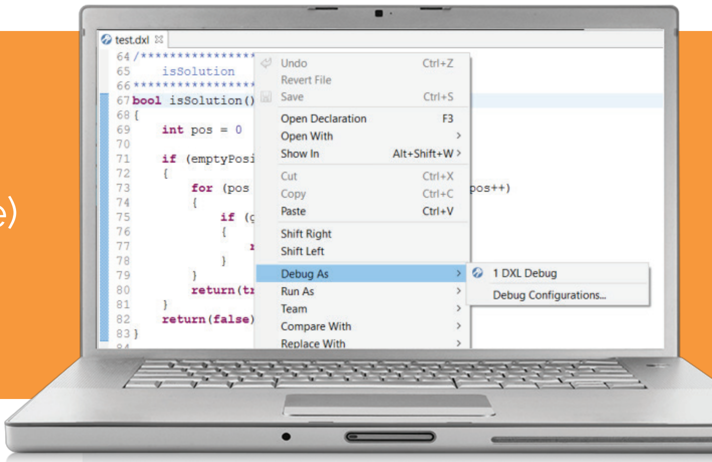
DXL Editor

Created by sodius®

Edit, run, and debug your IBM Rational DOORS DXL (DOORS eXtension Language) scripts from your favorite Eclipse IDE.

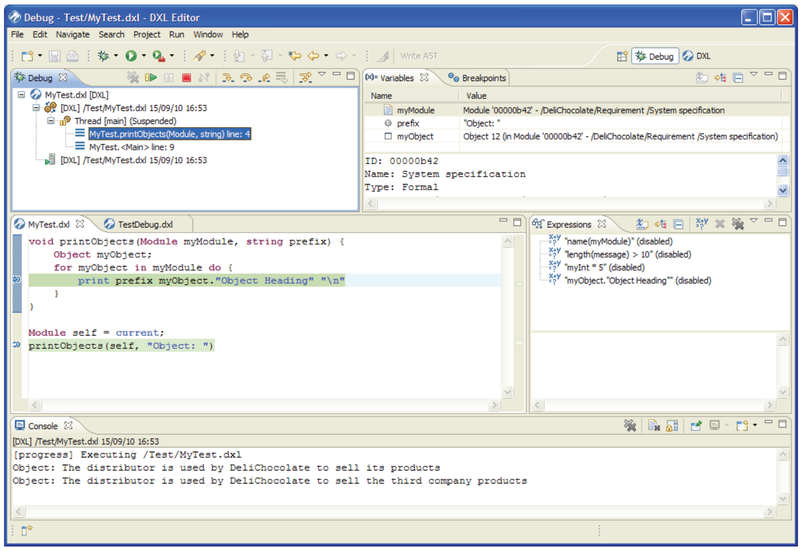


```
/*****************
    isSolution
******************
bool isSolution()
{
    int pos = 0

    if (emptyPosi
    {
        for (pos                         pos++)
        {
            if (g
            {

            }
        }
        return(tr
    }
    return(false)
}
```

IBM Rational DOORS® is a well established platform for engineers, allowing them to manage their project requirements in a flexible environment. DOORS data is manipulated with a dedicated language, DXL (the DOORS eXtension Language).

To overcome the limitations of the DOORS native editor, Sodius has developed the DXL Editor, offering unmatched features to facilitate developers' lives.

Going far beyond syntax highlighting, the DXL Editor is a real development environment for DXL built on the market-leading Eclipse platform, bringing its richness and power to provide a first-class environment for editing, executing and debugging your DXL scripts, on par with other well known languages such as Java and C++.


Debug - Test/MyTest.dxl - DXL Editor
File Edit Navigate Search Project Run Window Help
Write AST
Debug
DXL
Debug
Variables
Breakpoints
MyTest.dxl
TestDebug.dxl
Expressions
Console

```
MyTest.dxl [DXL]
  [DXL] /Test/MyTest.dxl 15/09/10 16:53
    Thread [main] (Suspended)
      MyTest.printObjects(Module, string) line: 4
      MyTest.<Main> line: 9
  [DXL] /Test/MyTest.dxl 15/09/10 16:53
```

Name	Value
myModule	Module '00000b42' - /DeliChocolate/Requirement /System specification
prefix	"Object: "
myObject	Object 12 (in Module '00000b42' - /DeliChocolate/Requirement /System specification)

```
ID: 00000b42
Name: System specification
Type: Formal
```

```
void printObjects(Module myModule, string prefix) {
    Object myObject;
    for myObject in myModule do {
        print prefix myObject."Object Heading" "\n"
    }
}

Module self = current;
printObjects(self, "Object: ")
```

```
"name(myModule)" (disabled)
"length(message) > 10" (disabled)
"myInt * 5" (disabled)
"myObject."Object Heading"" (disabled)
```

```
[DXL] /Test/MyTest.dxl 15/09/10 16:53
[progress] Executing /Test/MyTest.dxl
Object: The distributor is used by DeliChocolate to sell its products
Object: The distributor is used by DeliChocolate to sell the third company products
```

Improve efficiency & reduce development time with DXL Editor.

Features include:

EDITING
- ✓ Syntax highlight
- ✓ Mark occurrences
- ✓ Content assist
- ✓ Text hover

BROWSING
- ✓ Outline
- ✓ Project explorer

COMPILATION
- ✓ Compilation
- ✓ Problems view

EXECUTION
- ✓ Run
- ✓ Console

DEBUG
- ✓ Breakpoints insertion
- ✓ Step-by-step debug
- ✓ Variable inspection
- ✓ Expression evaluation

REFERENCE MANUAL
- ✓ Added to Eclipse help view
- ✓ Copy/paste code

DXL Editor

Created by sodius®

Feature-rich platform for faster, better results

Editing Features

SYNTAX HIGHLIGHT

Keywords, strings, comments and function calls are highlighted.

MARK OCCURRENCES

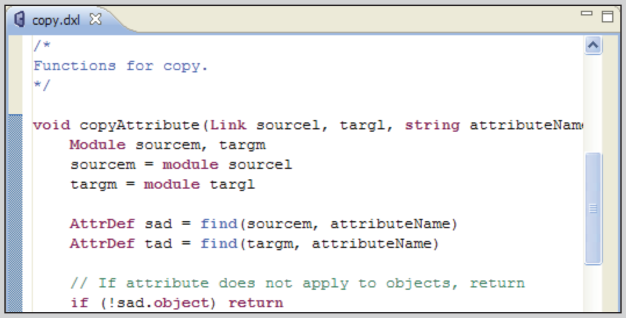
Mark Occurrences dynamically highlights the occurrences of the word currently selected in the editor.

CONTENT ASSIST

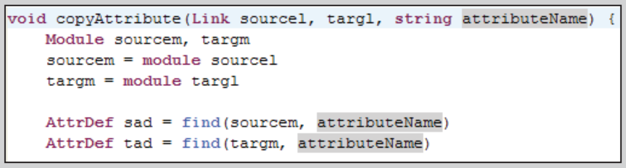
Content Assist provides a list of suggested completions for partially entered strings. Templates are shown together with the Content Assist proposals. There are existing templates, such as ‘for’, ‘if’ and more, but you can also define new templates.

TEXT HOVER

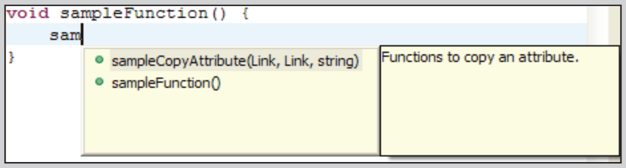
When the mouse is over a DXL function call, a tooltip displays the function signature and its associated comment, if any.



```
/*
Functions for copy.
*/

void copyAttribute(Link sourcel, targl, string attributeNam
    Module sourcem, targm
    sourcem = module sourcel
    targm = module targl

    AttrDef sad = find(sourcem, attributeName)
    AttrDef tad = find(targm, attributeName)

    // If attribute does not apply to objects, return
    if (!sad.object) return
```

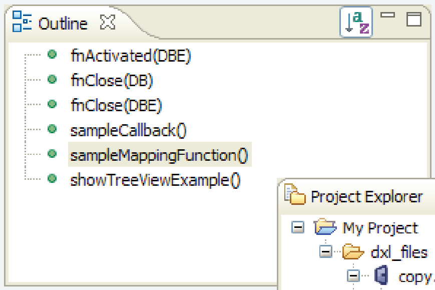


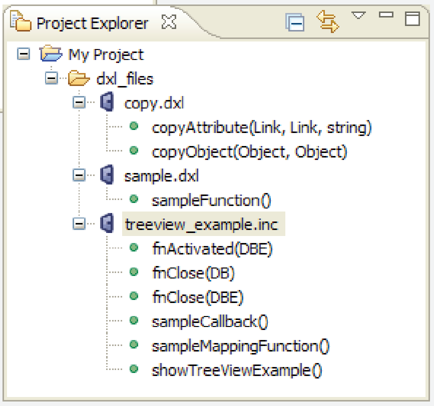
```
void copyAttribute(Link sourcel, targl, string attributeName) {
    Module sourcem, targm
    sourcem = module sourcel
    targm = module targl

    AttrDef sad = find(sourcem, attributeName)
    AttrDef tad = find(targm, attributeName)
```



```
void sampleFunction() {
    sam
}
```

sampleCopyAttribute(Link, Link, string)
sampleFunction()
Functions to copy an attribute.





Browsing Features

OUTLINE

The Outline View displays the structure of the DXL file, listing the functions declared within it, and automatically refreshing as you edit the file. Clicking a function in this view causes the editor to jump to the function’s declaration.

PROJECT EXPLORER

The Project Explorer lets you organize a consistent set of DXL files into projects. DXL Functions are displayed in the project’s view, without the need to open the corresponding DXL file in an editor, enabling you to quickly navigate into your project contents.

DXL Editor

Created by sodius®

Feature-rich platform for faster, better results

Debugging is a critical activity when developing source code.

With DXL Editor, you have full control.

WHY DXL EDITOR?

- ✓ Faster coding process
- ✓ Higher quality results
- ✓ Finalize more projects
- ✓ User-friendly interface

YOU CAN:

- ✓ Add breakpoints
- ✓ Control execution
- ✓ Inspect variables
- ✓ Inspect data

Debug Features

DEBUG VIEW

The Debug View is the primary view to manage the debugging of a program. This view displays the stack frame for the suspended DXL execution you are debugging, showing the list of function calls.

BREAKPOINTS VIEW

The Breakpoints View lists all the breakpoints you currently have set in your workspace.

You can double-click a breakpoint to display its location in the editor (if applicable). You can also enable or disable breakpoints, delete them, add new ones, and group them by working set.

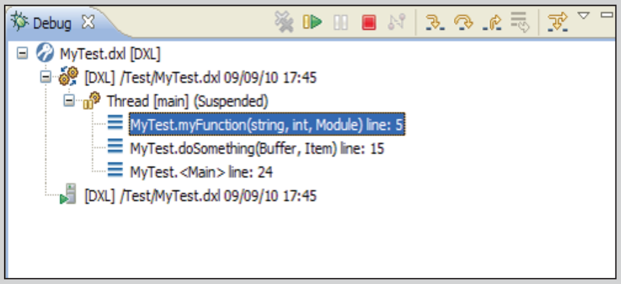
VARIABLES VIEW

The Variables View displays information about the variables associated with the selected function in the Debug View. Selecting a variable will display more detailed information in the Detail Pane.

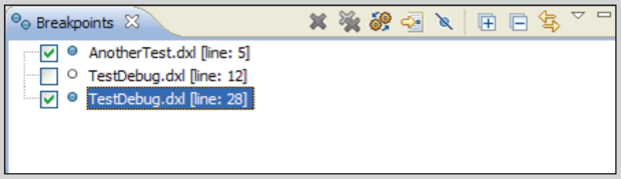
EXPRESSIONS VIEW

The Expressions View allows you to dynamically execute queries to assist your debugging. You can enter any valid code and have the results calculated at every execution step to monitor values. Entries in the Expressions View can be selected to have more detailed information displayed in the Detail Pane.

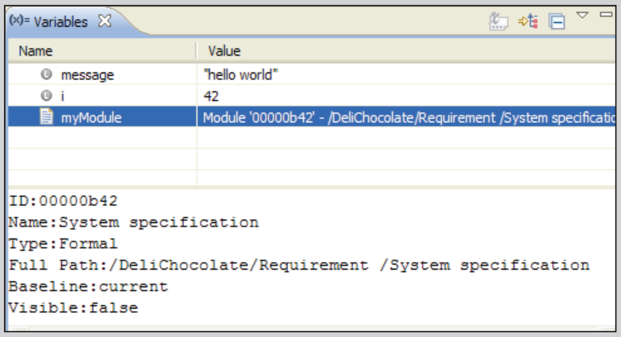
Note: Please refer to online documentation for a full description of features and known limitations.



```
MyTest.dxl [DXL]
  [DXL] /Test/MyTest.dxl 09/09/10 17:45
    Thread [main] (Suspended)
      MyTest.myFunction(string, int, Module) line: 5
      MyTest.doSomething(Buffer, Item) line: 15
      MyTest.<Main> line: 24
  [DXL] /Test/MyTest.dxl 09/09/10 17:45
```

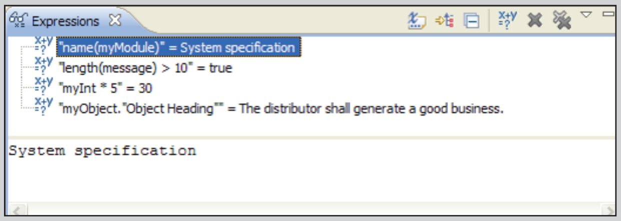


- ✓ AnotherTest.dxl [line: 5]
- ☐ TestDebug.dxl [line: 12]
- ✓ TestDebug.dxl [line: 28]



Name	Value
message	"hello world"
i	42
myModule	Module '00000b42' - /DeliChocolate/Requirement /System specificatio

```
ID:00000b42
Name:System specification
Type:Formal
Full Path:/DeliChocolate/Requirement /System specification
Baseline:current
Visible:false
```



```
"name(myModule)" = System specification
"length(message) > 10" = true
"myInt * 5" = 30
"myObject."Object Heading"" = The distributor shall generate a good business.
```

```
System specification
```

COMPATIBLE SOFTWARE: DOORS® 8.x, 9.x.
SYSTEM REQUIREMENTS: Microsoft® Windows®

DXL Editor is expected to run locally on the same machine where DOORS is running.

DXL Editor

Created by sodius®

Feature-rich platform for faster, better results

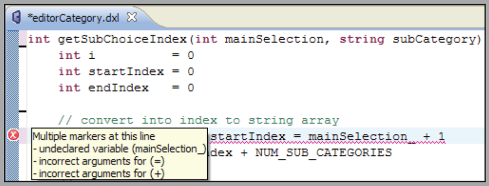
Compilation Features

COMPILATION

Each time a DXL file is modified and saved, the DXL compiler is automatically executed on that file. For each error, an annotation is added in the editor sidebar, with a tooltip showing the error message, and the corresponding line is underlined.

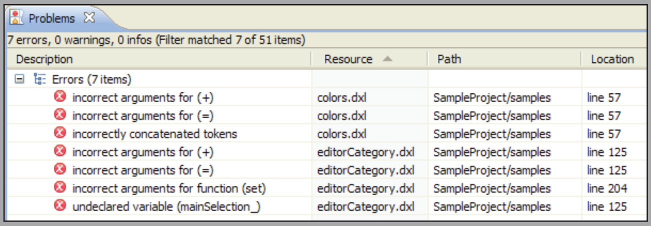
PROBLEMS VIEW

Navigation through DXL errors for a particular file, a project or the entire workspace is done using the Problems view.


*editorCategory.dxl

```
int getSubChoiceIndex(int mainSelection, string subCategory)
    int i          = 0
    int startIndex = 0
    int endIndex   = 0

    // convert into index to string array
                         startIndex = mainSelection_ + 1
                         dex + NUM_SUB_CATEGORIES
```

Multiple markers at this line
- undeclared variable (mainSelection_)
- incorrect arguments for (=)
- incorrect arguments for (+)


Problems

7 errors, 0 warnings, 0 infos (Filter matched 7 of 51 items)

Description	Resource	Path	Location
Errors (7 items)			
incorrect arguments for (+)	colors.dxl	SampleProject/samples	line 57
incorrect arguments for (=)	colors.dxl	SampleProject/samples	line 57
incorrectly concatenated tokens	colors.dxl	SampleProject/samples	line 57
incorrect arguments for (+)	editorCategory.dxl	SampleProject/samples	line 125
incorrect arguments for (=)	editorCategory.dxl	SampleProject/samples	line 125
incorrect arguments for function (set)	editorCategory.dxl	SampleProject/samples	line 204
undeclared variable (mainSelection_)	editorCategory.dxl	SampleProject/samples	line 125

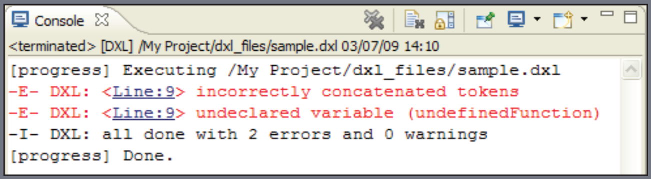
Execution Features

RUN

A DXL file can be executed directly from Eclipse.

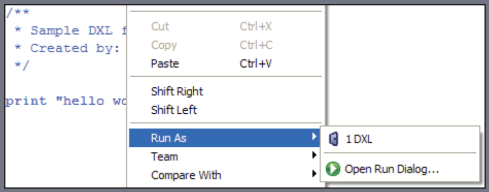
CONSOLE

A console is opened when a DXL file is executed. This console logs any DXL text output, and indicates any DXL windows that may be opened by the executed file. Execution errors are displayed in the console and hyperlinks are available on line numbers to jump to the corresponding location in the DXL editor.


Console

<terminated> [DXL] /My Project/dxl_files/sample.dxl 03/07/09 14:10

```
[progress] Executing /My Project/dxl_files/sample.dxl
-E- DXL: <Line:9> incorrectly concatenated tokens
-E- DXL: <Line:9> undeclared variable (undefinedFunction)
-I- DXL: all done with 2 errors and 0 warnings
[progress] Done.
```



```
/**
 * Sample DXL f
 * Created by:
 */

print "hello wo
```

Other DOORS-related products

MDConnect

Get DOORS database navigation in an Eclipse environment. The RSA Extension interconnects DOORS and Rational Software Architect, facilitating the setup and maintenance of links between requirements and models.

- ✓ Trace Eclipse designs to DOORS requirements
- ✓ Built-in RSA traceability & analysis tool
- ✓ Produce traceability reports from Eclipse

MDWorkbench

MDWorkbench for DOORS brings advanced functionality that helps to architect DOORS databases, to reverse engineer DOORS schemas, to generate documents and to exchange any DOORS information with other environments.

- ✓ Leverage, convert & transform data across tools
- ✓ Code complex transformation rules in Java
- ✓ Leverage models across tools