MDev MUMPS IDE

Usage and Installation

# Installation

# Prerequisites

MDev supports both Windows and Linux.

|  |  |  |
| --- | --- | --- |
| **Application** | **Version** | **Notes** |
| VistA / Fileman | Any | This is currently tested with VistA-FOI, available here: <http://www.osehra.org/page/osehra-code-repository> |
| Eclipse | Indigo |  |

# Download the latest files

Download the latest version here: <https://github.com/JimDeanSpivey/M-Tools-Project/archive/master.zip> and unpack this file. It contains the plugin jar files needed, along with the KIDS packages.

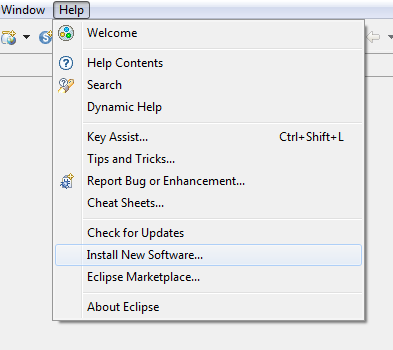
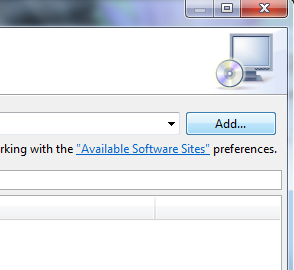
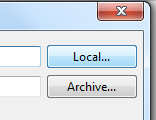
# Install the KIDS packages

From the unpacked file, the KIDS packages are located under /MiscDependencies/KIDS/. Install all three packages, M-Debugger, M-Editor and Utilities.

# Run the VistALink job

Enter the MUMPS command JOB LISTENER^XOBVTCPL(8001) to start VistALink. This is required for the plugin to connect with and talk to the server.

# Install the plugin

1. Open Eclipse.
2. Click Help 🡪 Install New Software  
   
3. Then click Add in the top right.  
   
4. Select the button named Local. This in towards the top right.  
   
5. Select the directory where the zip file was unpacked. Then choose the “MToolsUpdateSiteProject” directory.
6. After clicking OK, thenewly added update site will automatically be selected.
7. Click Next and follow the prompts to install the plugin.

# Configure the plugin

1. From Eclipse’s main menu (the top most bar) Click Window -> Preferences -> VistA -> Connection.
2. Remove the dummy “Primary” connection value.
3. Add a new connection in the format of [Name];[IP Address or hostname];[port number];[blank or an eclipse project name]. A typical value would be local;127.0.0.1;8001;
4. Click OK.

# Usage

# Eclipse Projects

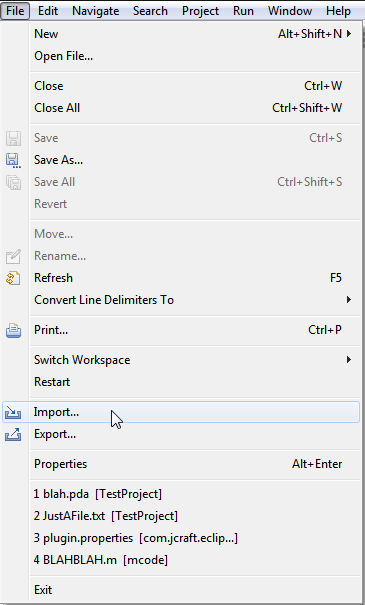
All files in eclipse must belong to a project, being that Eclipse can be described as project centric. To begin using MDev, a new project must either be created, and files imported into it. Or an existing project can be opened, which already contains files and of course new files can be imported. Files can also of course be removed and/or deleted from eclipse. Additionally a file can be imported via a link as opposed to being in the same directory where the project’s root is. The project root is a single directory which contains the .project file. A workspace however is a parent to a project, and contains 0 or many projects. Projects may or may not exist in the workspace directory root.

# Creating a new project

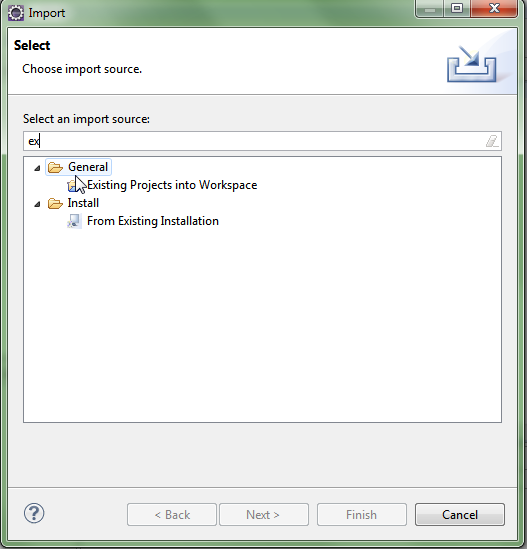
TODO: There will be steps to create a new project, wait until the MDev project type is completed.

# Opening an existing project

To open an existing project, click on ‘New -> Import…‘ .



From the Import Wizard choose ‘Existing projects into Workspace’.



This can be filtered to easily by typing ‘ex’. Using the file dialog navigate to the directory which contains an existing project (a directory with a .project file in it). This will typically be from a version controlled system such as Git, which will typically contain a parent root directory that has the .project file in it, and other files with binaries and source files in children folders.

# Using the MEditor

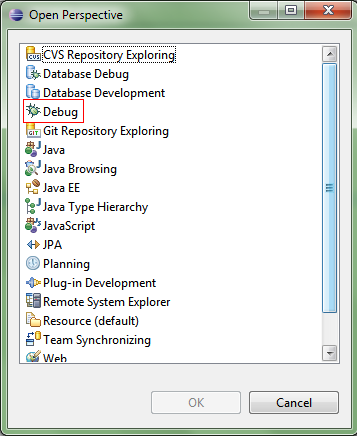
To begin using MDev’s M-Editor (a custom editor for MUMPS language), simply open a file ending with the suffix ‘.m’. A custom editor which includes syntax coloring, new contextual menus and the outline view (by default the vertical right pane) will display tags in the current routine.

# Loading Routines (Actual usage pending changes)

# Saving routines (actual usage pending changes)

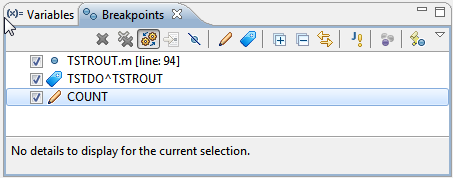
# Debugging

To use the debug features, enter the Eclipse Debug Perspective. Click the change perspective menu in the top right of the EclispeIDE. Then choose `Debug`.



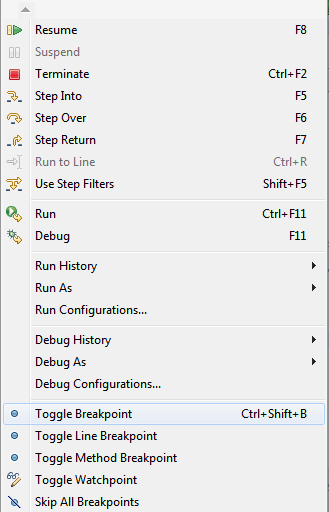
# Adding breakpoints

The MDev Debugger has 3 types of breakpoints, line breakpoints, tag breakpoints and variable watchpoints. All breakpoints can be seen from the breakpoints view, under the debug perspective.

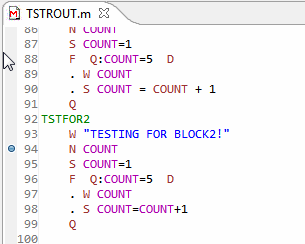


# Line breakpoints

Open the M Code in an editor that you want to debug. Either press ‘ctrl + shift + B’ or go to the Run Menu at the top of the Eclipse application and select toggle Line Breakpoint.



Third, you can also double click on the vertical bar left of the editor to add a line breakpoint. It is important to add breakpoints otherwise the debugger will run until it completes, and terminates.



# Tag breakpoints

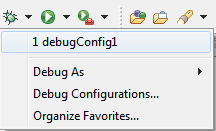
Arbitrary tags can be entered (eg: TAG^ROUTINE) too. From the breakpoints view, click on the blue tag icon.

# Watchpoints

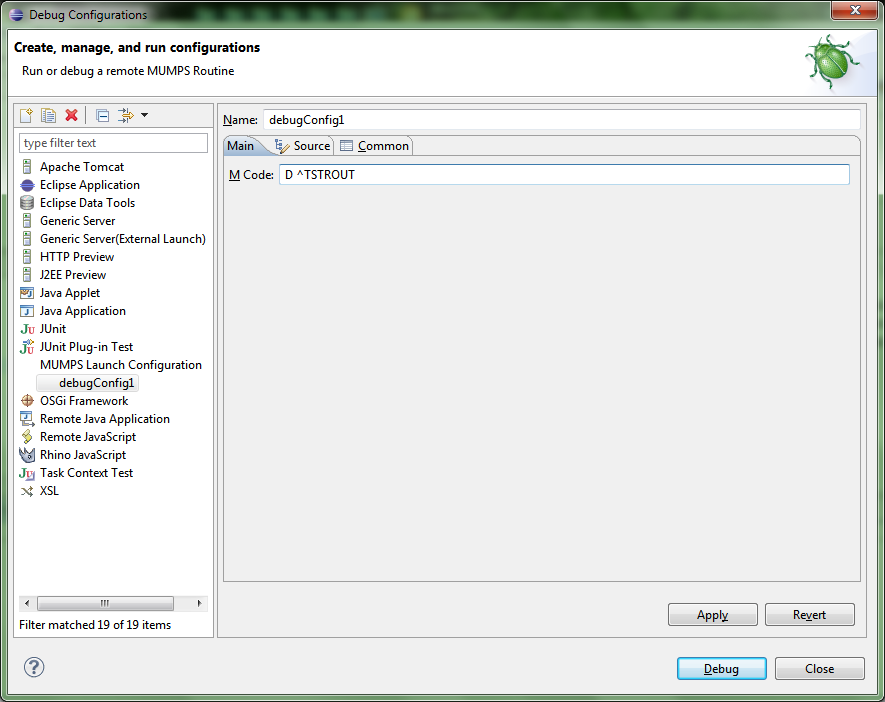
Variable watchpoints can be added. They are triggered when a variable value changes. They too are added from the breakpoints view, by clicking the pencil icon.

# Starting a debug session

To start the debugger, a new launch configuration must be created. Create a new debug configuration.

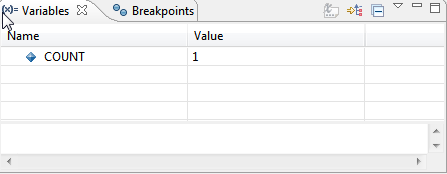


Enter the M code which you want to debug, and which ideally will encounter your breakpoints.



# Variables

Variables will be displayed in the default Eclipse Debug Variables View. But only those which were created after the debug session has started.



Since the default view isn’t showing all variables currently defined on the server, there is a new custom view created for that. Open it by clicking on Window -> Show View -> Other… and choose ‘M Variables’. It can also be filtered by using the text box atop the variable viewer.

