Department of Veterans Affairs

**Open Source Electronic Health Record Services**

MTools Installation and Usage Guide



**Version 1.1**

August 2013

Table of Contents

[1. Installation 4](#_Toc365913828)

[1.1. Prerequisites 4](#_Toc365913829)

[1.2. Download Latest Files 4](#_Toc365913830)

[1.3. Install KIDS Packages and Update VistALink 4](#_Toc365913831)

[1.4. Run VistALink Job 4](#_Toc365913832)

[1.5. Install the Plug-in 4](#_Toc365913833)

[1.6. Configure the Plug-in 6](#_Toc365913834)

[2. Basic Configuration 6](#_Toc365913835)

[2.1. Server Configuration 6](#_Toc365913836)

[2.2. Eclipse Projects 7](#_Toc365913837)

[2.2.1. Creating a New Project 7](#_Toc365913838)

[2.2.2. Opening an Existing Project 8](#_Toc365913839)

[2.2.1. Project VistA Properties 9](#_Toc365913840)

[2.3. VistA Perspective 9](#_Toc365913841)

[2.4. Connecting To and Disconnecting From Servers 10](#_Toc365913842)

[3. Using MEditor 11](#_Toc365913843)

[3.1. MEditor Preferences 11](#_Toc365913844)

[3.2. Loading Routines from Server 12](#_Toc365913845)

[3.2.1. Load M Routine Toolbar Button 12](#_Toc365913846)

[3.2.1. Routine Load Project Properties 15](#_Toc365913847)

[3.2.2. Backup Files 17](#_Toc365913848)

[3.2.3. MEditor Context Menu 17](#_Toc365913849)

[3.2.4. Project Explorer M Server Context Menu 18](#_Toc365913850)

[3.2.4.1. Load Routines 18](#_Toc365913851)

[3.2.4.1. Load Routine into Folder 18](#_Toc365913852)

[3.2.4.1. Load Namespace into Folder 19](#_Toc365913853)

[3.3. Saving Routines 19](#_Toc365913854)

[3.3.1. MEditor Save 19](#_Toc365913855)

[3.3.1. MEditor M Server Context Menu 20](#_Toc365913856)

[3.3.1.1. Save Routine 20](#_Toc365913857)

[3.3.1.1. Save Routine to Selected Server 20](#_Toc365913858)

[3.3.2. Project Explorer M Server Context Menu 20](#_Toc365913859)

[3.3.2.1. Save Routines 20](#_Toc365913860)

[3.3.2.1. Save Namespace 20](#_Toc365913861)

[4. Debugging 21](#_Toc365913862)

[4.1. Adding Breakpoints 21](#_Toc365913863)

[4.1.1. Line Breakpoints 21](#_Toc365913864)

[4.1.2. Tag Breakpoints 22](#_Toc365913865)

[4.1.3. Watchpoints 22](#_Toc365913866)

[4.2. Starting a Debug Session 23](#_Toc365913867)

[4.3. Variables 24](#_Toc365913868)

[4.4. Interactive Console 25](#_Toc365913869)

[5. New VistA File Wizard 26](#_Toc365913870)

[6. MTools Context Menu Tools 28](#_Toc365913871)

[6.1.1. Report Assumed Variables 30](#_Toc365913872)

[6.1.2. Report Errors 32](#_Toc365913873)

[6.1.3. Report Occurrences 32](#_Toc365913874)

[6.1.4. Report Quit Types 33](#_Toc365913875)

[7. M Refactor Context Menu Items 33](#_Toc365913876)

# 

# Installation

## Prerequisites

MTools integrated development environment (IDE) supports both Windows and Linux.

|  |  |  |
| --- | --- | --- |
| **Application** | **Version** | **Notes** |
| Veterans Health Information Systems and Technology Architecture (VistA) | Any | This is currently tested with VistA-Freedom of Information Act (FOIA), available [here](http://www.osehra.org/page/osehra-code-repository).  VistA installation instructions are available on the Open Source Electronic Health Record Agent (OSEHRA) site [here](http://www.osehra.org/). |
| Eclipse | Indigo |  |
| Java Runtime | 7 |  |

## Download Latest Files

Download the latest version of MTools [here](https://github.com/kthlnkeating/M-Tools-Project/archive/master.zip) and unpack the file. It contains the plug-in jar files needed and the Kernel Installation and Distribution System (KIDS) packages.

## Install KIDS Packages and Update VistALink

From the unpacked file, the KIDS packages are located under */MiscDependencies/KIDS/*. Install all three packages: MDebugger, MEditor, and Utilities. KIDS installation instructions can be found [here](http://www.va.gov/vdl/documents/Infrastructure/Kernel/krn8_0dg.pdf).

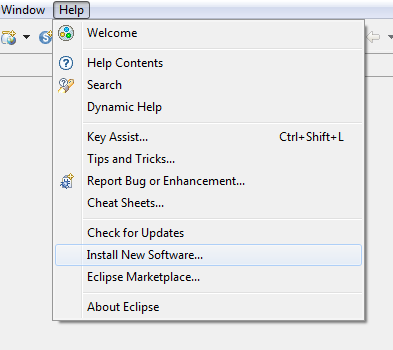
If a user is working in Linux, VistALink routines need to be updated with those found [here](http://www.seaislandsystems.com/Downloads/VistALink/xobv-gtm.ro); these files can be loaded using *D ^%RI* utility from a MUMPS Terminal.

## Run VistALink Job

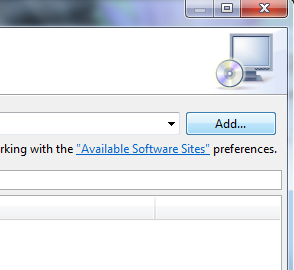
Enter the MUMPS command *JOB LISTENER^XOBVTCPL(8001)* to start VistALink. This action is required for the plug-in to connect with and talk to the server. Here 8001 refers to the port number and you can specify other ports as well.

## Install the Plug-in

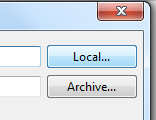
1. Open Eclipse
2. Click ‘Help’ 🡪 ‘Install New Software’



1. Then Clock ‘Add’ in the top right



1. Select the button named ‘Local’; this is towards the top right



1. Select the directory where the zip file was unpacked. Then choose the ‘MToolsUpdateSiteProject’ directory.
2. After clicking ‘OK’, the newly added update site will automatically be selected
3. Click ‘Next’ and follow the prompts to install the plug-in

## Configure the Plug-in

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’

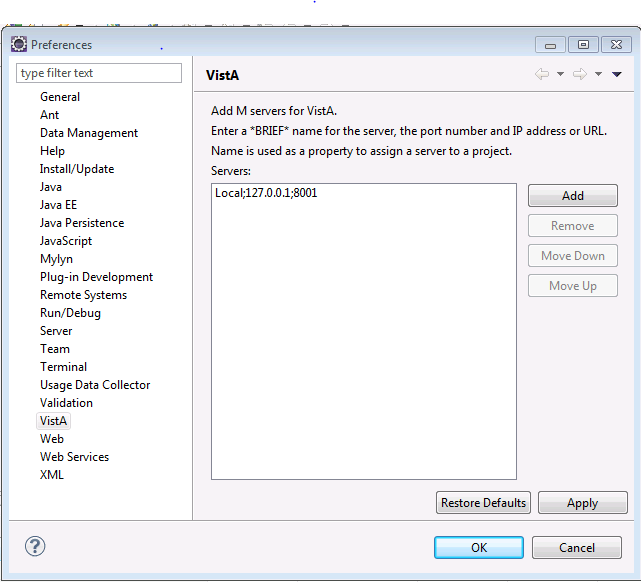
# Basic Configuration

## Server Configuration

M Server that will be used with MTools IDE is specified in Preferences.

1. From Eclipse’s main menu (the top most bar) Click Window -> Preferences .
2. Select VistA from the list on the left.
3. Add a new server using add “Add” button. You need to enter a name, an IP address or hostname and a port for the server. A typical value would be local;127.0.0.1;8001 for local development.
4. Click OK to save the server.

You can specify multiple servers and unlike the previous version of MTools there is no concept of a primary server. How servers are used will be described in later sections.



## Eclipse Projects

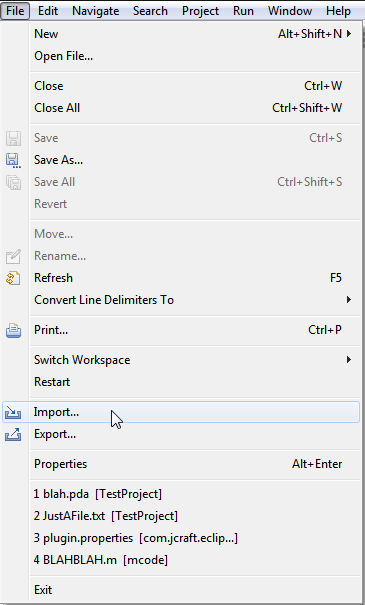
All files in Eclipse must belong to a project since Eclipse can be described as project centric. To begin using MTools IDE there are two options: (1) a new project can be created and files imported or (2) an existing project can be opened containing existing files and the capability to import new files. All files in Eclipse can be removed and/or deleted. Typically files are located in the directory hierarchy under project’s root directory. Files can also be imported via a link as opposed to being in the same directory where the project’s root is but linked files are not tested in this version of MTools and may not function properly. 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

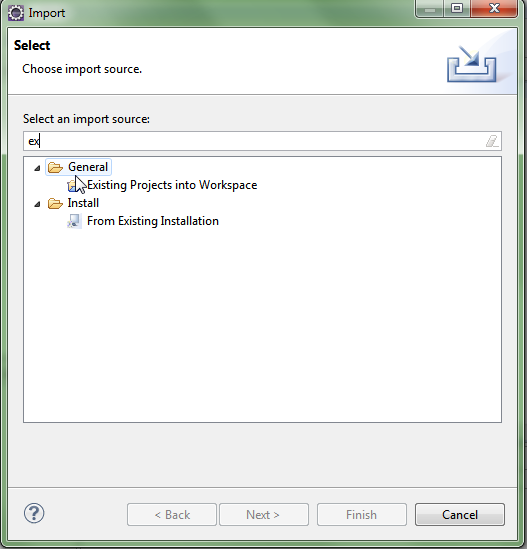
A standard Eclipse project can be created via ‘New’🡪’New Project…’.This opens the new project wizard. The standard Eclipse project can be chosen from ‘General’🡪’Project’.

### Opening an Existing Project

To open an existing project, click on ‘New’🡪’Import…’.



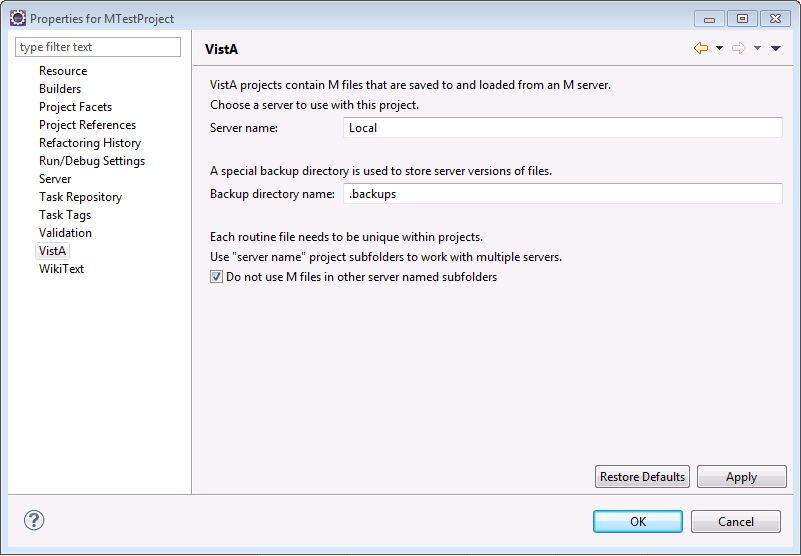
From the Import Wizard choose ‘Existing Projects into Workspace’ under the ‘General’ folder.



This can be filtered easily by typing ‘ex’. Using the file dialog navigate to the directory which contains an existing project (a directory containing a .project file). 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.

### Project VistA Properties

To be able to load and save files from an M server a project must be assigned a server. This assignment is done in VistA properties of a project. Project properties can be accessed from Eclipse’s main menu (the top most bar, click Project -> Properties) or from project context menu.



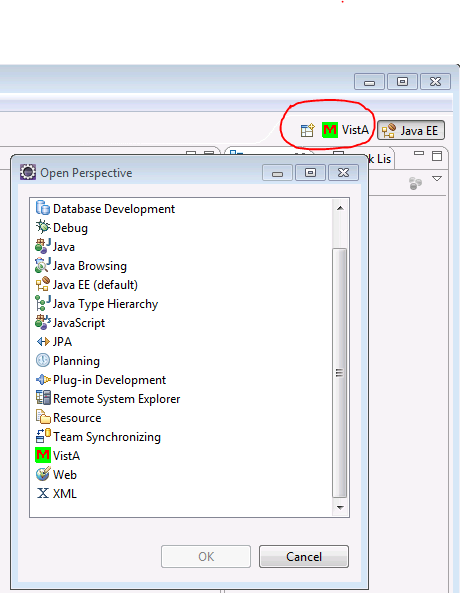
VistA projects are assigned to a server by entering the name in the properties. This server is the primary one used for loading and saving server routines. Loading and saving routines and the other two settings in this property are described later in this document.

## VistA Perspective

MTools adds a new VistA perspective to Eclipse. When this perspective is selected on the main menu, toolbar and context menus are updated to have VistA related menu items. You can select VistA perspective from Window->Open Perspective->Other on the main menu or Open Perspective button on upper right corner. For editor functionality the rest of this document assumes that you are always in VistA perspective.

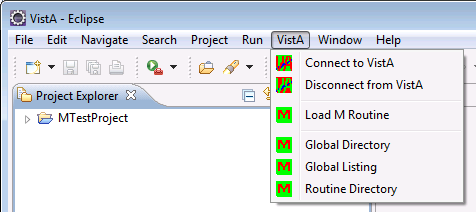
VistA Perspective adds a VistA menu to the main menu and a Load M Routine button to the toolbar. All the other additions are in context menus. Most menu items have the M icon.

By default VistA perspective uses Project Explorer and Outline View for context functionality and Console view for output.



## Connecting To and Disconnecting From Servers

Under VistA main menu you can find Connect To VistA and Disconnect from VistA menu items for servers specified in the VistA preferences. Note that most of the functionality described in this document prompts user to connect to a server if the connection has not been established so for most workflows you do not need to use these menu items directly. One notable exception is MDebugger where Connect To VistA menu item must be used.

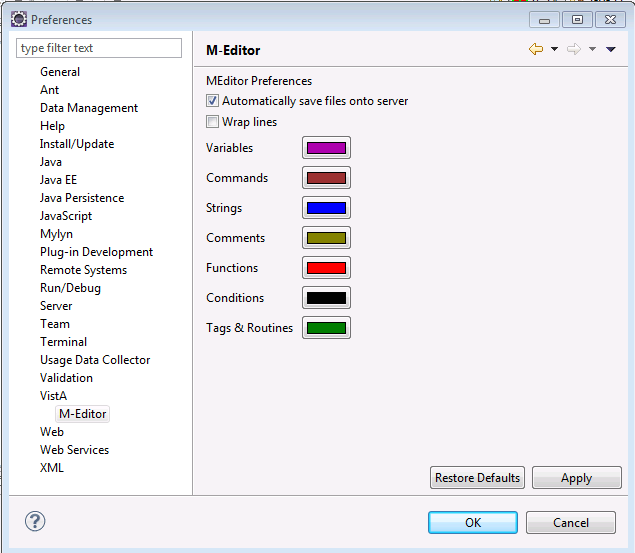


# Using MEditor

MEditor is a custom editor for MUMPS language and part of MTools IDE. To begin using MEditor 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 lower left pane) will display tags in the current routine selected in the editor.

## MEditor Preferences

The colors that are used in MEditor can be configured in VistA preferences under Window->Preferences->VistA->MEditor.

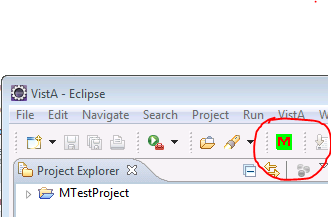


In addition to colors you can also select if lines in the editor is wrapped or not from the VistA preferences.

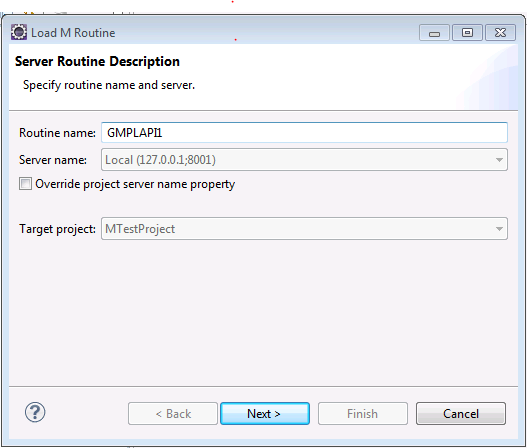
## Loading Routines from Server

### Load M Routine Toolbar Button

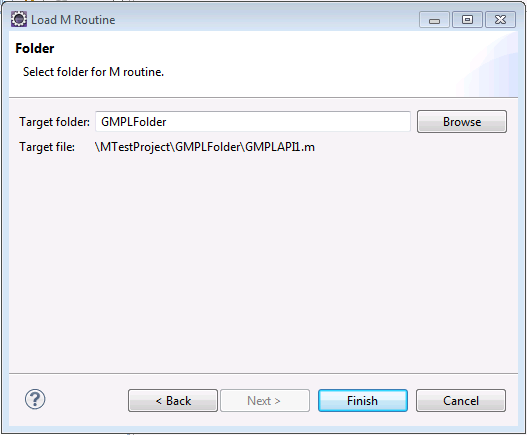
The most generic way routines can be loaded from an M server is the green ‘M’ button in the Eclipse toolbar. This button is not specific to any project and a wizard is launched when it is selected. ‘VistA’🡪‘Load M Routine’ menu item on the top VistA menu is identical in functionality to this button.



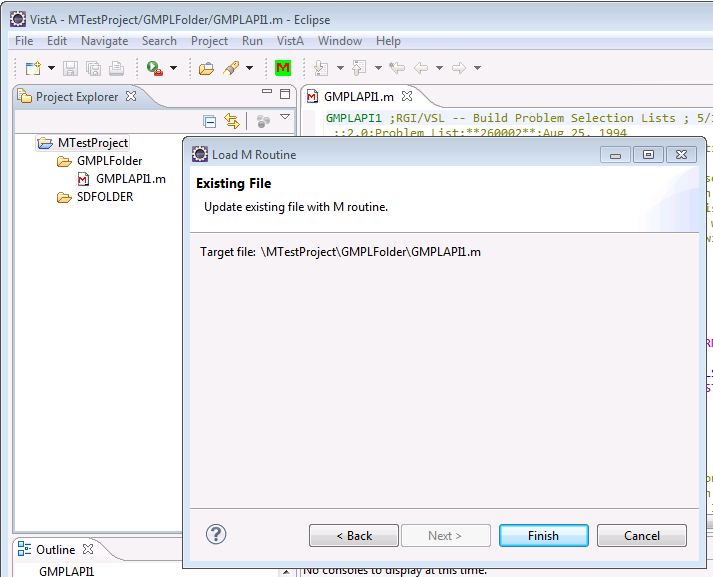
In the first page of the wizard you specify the routine name, the server name where the routine will be loaded from and project name where the routine M file will be saved to on client. When possible both the server name and the project name are defaulted. By default when a server is selected only the projects that are configured to use the selected server are shown in drop down selection list. You can override the filtering by checking “Override project server name property”. In this case all projects are shown. This way you can load routines into the project from secondary servers that are not assigned to the project.



Unlike the previous version of MTools IDE, this version supports arbitrary hierarchical directory structures in a project. Thus by default user can choose the target folder in the next page in the wizard. An existing folder under the selected project’s root directory must be selected. You can also type in the folder you want but the folder name must be relative. Currently only existing folders are supported. An empty folder is allowed in which case routine is loaded to the root project directory. Full target file path is always displayed in the second page of the Load M Routine wizard. When you hit Finish the routine is loaded from the server to the file shown and the file is opened in MEditor.

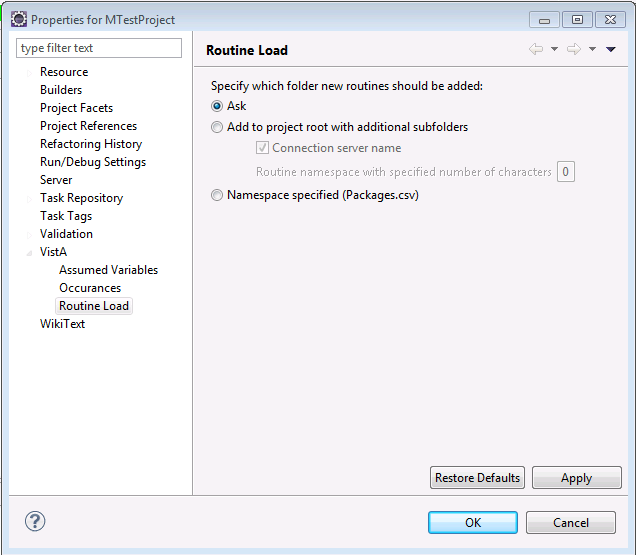


Once the routine is loaded to a particular folder all the subsequent loads from server updates the existing file and user will not be shown any folder selection widget. Various MTools functionality assumes each M file uniquely exists in the project. Load M Routine wizard enforces this requirement. Currently uniqueness is not enforced on manual file creation. If the selected routine already exists the second page of the wizard shows the path of the existing file.



### Routine Load Project Properties

As discussed in the previous section to support arbitrary hierarchical directories users are asked to enter a folder whenever a new routine is loaded from the server. You can however configure MTools to load new routines to particular folders in project properties Project->Properties->VistA->Routine Load.



By default “Ask” is the choice for a folder when a new routine is loaded from the server. You can choose “Add to project root with additional subfolders” to store all routines in a flat directory structure as in the previous version of MTools. The flat directory can be the project root directory or you can add additional subfolders to the project root. Connection server name by default is added as a subfolder under project root. Additionally you can chose to add an additional subfolder based on the namespace of the routines. The number of characters to use for the namespace folders can be configured. 0 means no namespace folder and maximum is 3 characters if namespaces are to be used as folders.

Note that MTools assumes each routine uniquely exists in a project. If your workflow requires that you need to work with multiple versions of a routine from multiple servers in the same project you have to choose “Add to project root with additional subfolders” here with “Connection server name” checked. For this case all the server named folders other than the primary server can be ignored for functionality such as Load M File. “Do not use M file in other server named subfolders” configuration is located in project VistA properties page and is not checked by default. This configuration is the only support MTools provides to work with multiple servers in the same project.

The final folder specification for new routine location is “Namespace specified”. This setting supports OSEHRA VistA-FOIA directory structure where routine directories are organized according to the packages. With this setting it is assumed that there is a file named Packages.csv in the root directory of the project similar to OSEHRA VistA-FOIA. Packages.csv specifies location of each routine based on its namespace. Note that if Packages.csv does not exist, this setting causes the routines to be always loaded to the project root directory.

### Backup Files

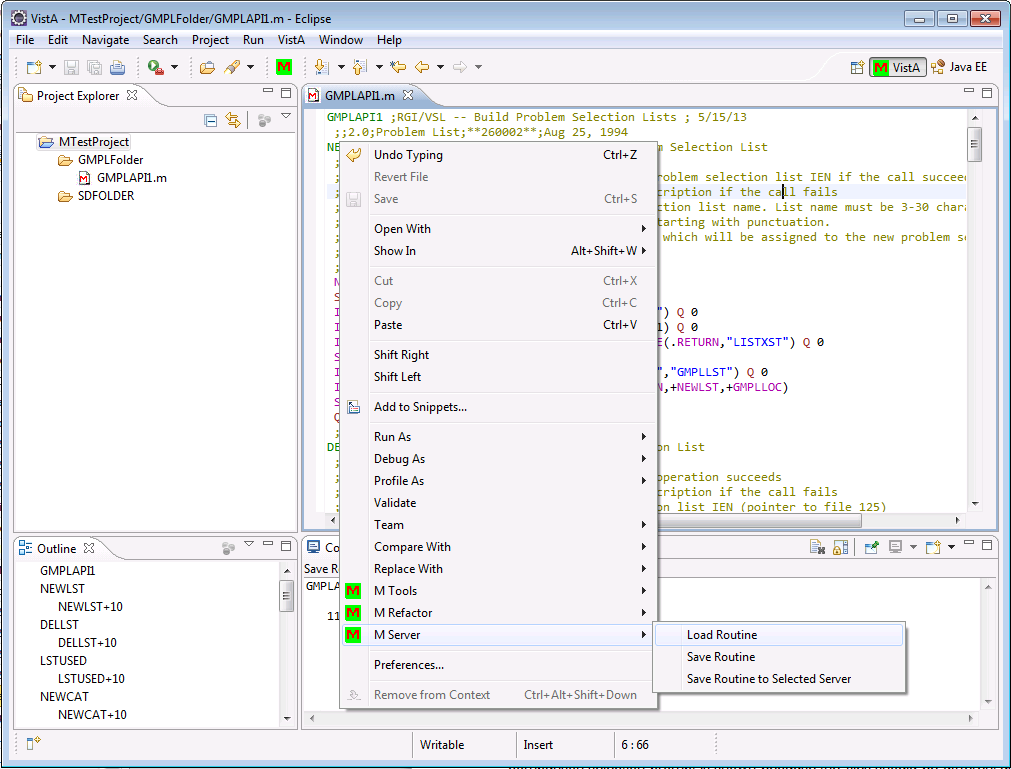
Whenever a file is successfully loaded, a backup file is either created or updated in the backup folder of the project. For each project backup folder is specified in Project VistA properties (Project->Properties->VistA) and defaults to “.backup”. MTools always use Server Name that the file is loaded from as an additional subfolder under the backup folder. You can turn off backup feature by specifying an empty backup folder in project VistA properties.

The backup file contains the latest server version locally. Backup file’s relative path with respect to backup folder and server named subfolder is always identical to its original relative path with respect to project root. Backup files have always the same name as the original file but with the extension “.mbk”. Users are free to browse the contents of the backup files and MEditor is configured to open “.mbk” files as well. However it is recommended that these files are not changed manually.

Backup files are primarily used for comparison purposes when a file is loaded from the server and backup/comparison purposes when a file is saved to the server. If MTools detects the server file is changed outside Eclipse the user is alerted to examine the changes. You may choose to include the backup folder in your version control system so that all versions of the backup files are available. Otherwise Eclipse local history can be used to compare versions of backup files. You can configure how long to keep the backups in the local history. Please see Eclipse help.

### MEditor Context Menu

MEditor M Server context menu item contains a “Load Routine” submenu item which can be used directly to update the file in the editor from the server. In this case no wizard is used since the file is already selected. Otherwise the functionality is identical to Load M Routine toolbar button.



### Project Explorer M Server Context Menu

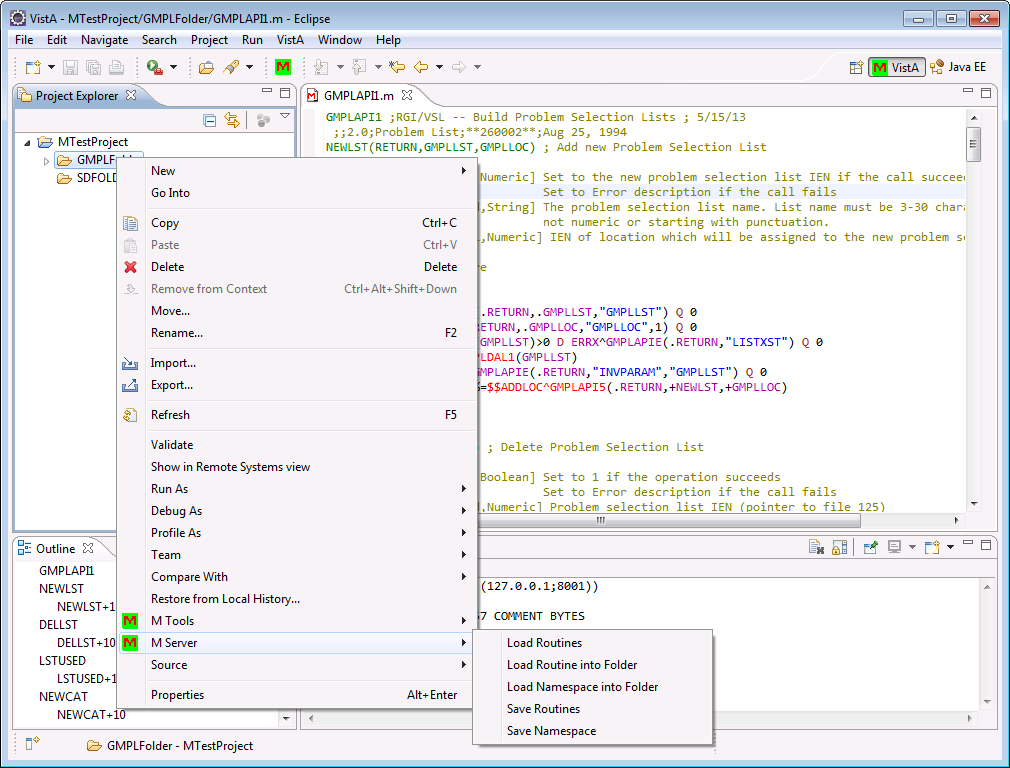
Project Explorer M Server context menu item can also be used load a single selected routine from the server and the functionality is identical to MEditor Load Routine context menu item. In addition from Project Explorer context menu multiple routines can be loaded at once.

### Load Routines

M Server->Load Routines context menu loads the selected routine(s) from the server. This menu is shown only one or more M routines and/or folders are selected in the explorer. When a folder is selected all the files under the folder is recursively includes in the list to be loaded. Except how the routine is selected the functionality is identical to Load M Routine toolbar button; this essentially repeats the load process for all the selected files.

### Load Routine into Folder

If a single folder is selected on the Project Explorer additional load options are available in the M Server context menu item. The first of this loads a single routine into the folder. Functionally this is equivalent to the case in Load M Routine wizard when the user can select a folder. Once this menu item is selected user is presented with an Input Dialog where she can enter the routine name to be loaded. Note that if the routine already exists in the project, the existing file is updated even if it is not in the selected folder.



### Load Namespace into Folder

Load Namespace into Folder enables multiple routines to be loaded to the same folder. When this menu item is selected, the user is presented with a Input Dialog where she can specify a namespace. All the routines in the specified namespace are loaded into the project; new routines are saved in the selected folder and existing routines are updated.

## Saving Routines

### MEditor Save

In MEditor routines are saved, just like all Eclipse files, to the disk. Additionally, when the save button is clicked, MEditor will also attempts to save the routine to the server. You can choose to disable this behavior by unchecking “Automatically save files onto server settings” in VistA preferences (‘Windows’ 🡪’Preferences’🡪’VistA’ 🡪’MEditor’). If automatic save is disabled, users can use context menu Save Routine to save the routine which is discussed shortly.

This version of MEditor always saves the files on client when user chooses so even if the file cannot be saved to the server (for example when a connection cannot be established). In addition tabs, control characters and empty lines are automatically removed upon saving a file. Control characters and empty lines are invalid in VistA routines and currently MEditor does not support tabs as line-start indicator.

When a routine is to be saved to the server, it will first fetch a copy of that routine from the server. MEditor will then compare what is on the server to the latest backup file for that routine. The latest backup file is a copy of what was last on the server. If this is different, a prompt will be shown to alert user to examine the differences as the difference means that the server file is changed outside MTools. In this case the file will not be immediately saved and user will have to save it again after examining the differences. Unlike the previous version of MTools no third party compare tool is used in this version. Users can use Local History or version control system compare facilities. If the file to be saved is identical to the server no saving is done and an informational dialog will be shown.

Once the routines are saved XINDEX is also ran on the server side and the results are shown in the Console View.

### MEditor M Server Context Menu

MEditor M Server context menu item contains ‘Save Routine” and “Save Routine to Selected Server” that can be used to save files to server. Both can be seen in the screenshots shown for Load Routine.

### Save Routine

This saves the routine in MEditor to the server and is functionally equivalent to saving it from MEditor.

### Save Routine to Selected Server

This is similar to MEditor but you can select the server to save the routine overriding the primary server specified in project properties. This functionality is provided to replace the secondary server saving in the previous version of MTools. This is the only way MTools provides to override the server specified in project properties.

### Project Explorer M Server Context Menu

Poject Explorer M Server context menu item can also be used save a single selected routine to the server and the functionality is identical to MEditor Save Routine context menu item. In addition from Project Explorer context menu multiple routines can be save at once.

### Save Routines

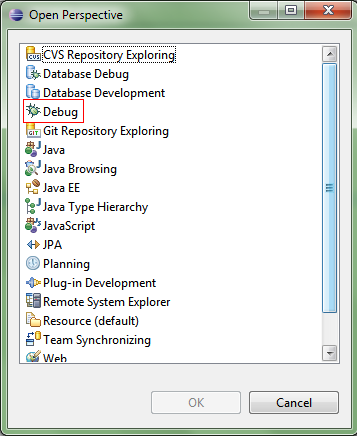
This saves all the selected files to the server. If there are folder selected all the files that are hierarchically under the folder are saved. The functionality is identical to the selected files one by one.

### Save Namespace

If there are folders selected in the Project Explorer then Save Namespace is shown in the context menu. If this item is selected then user can specify a namespace and all the files that are hierarchically under the selected folders that has the specified namespace will be saved and others will be ignored.

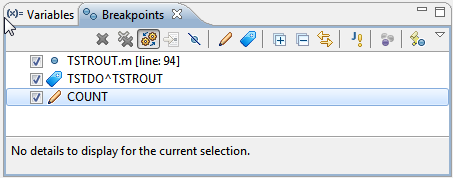
# Debugging

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



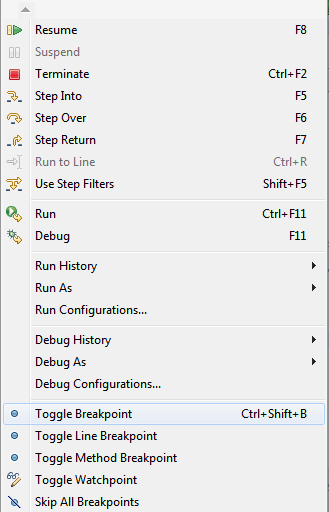
## Adding Breakpoints

The MDebug Plug-in has three types of breakpoints: line breakpoints, tag breakpoints, and variable watchpoints. All breakpoints can be seen from the breakpoints view under the debug perspective.

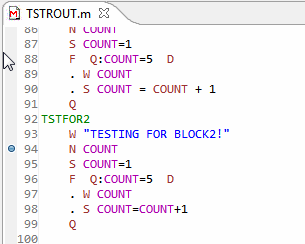


### Line Breakpoints

Open the M Code in an editor that the user wants to debug. Either press ‘Ctrl + Shift + B’ or go to the Run Menu at the top of the Eclipse application and select ‘Toggle Breakpoint’ or ‘Toggle Line Breakpoint’.



In addition, users can also double click 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

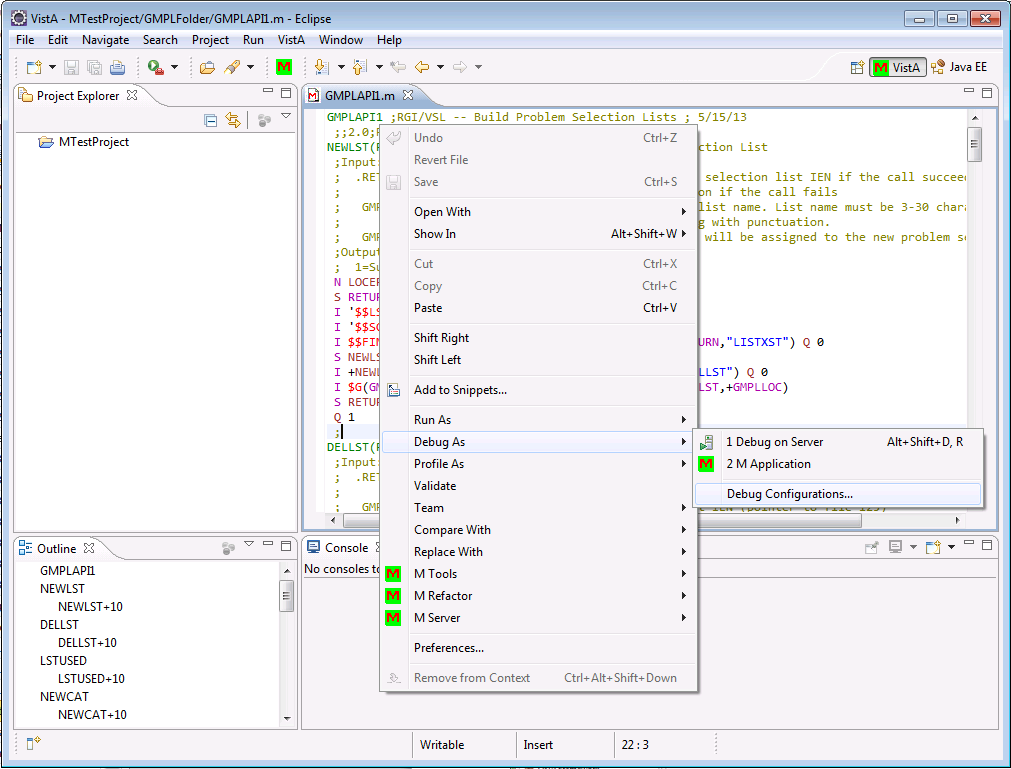
Arbitrary tags can also be entered (e.g.: *TAG^ROUTINE*). To add an arbitrary tag, click on the blue tag icon from the breakpoints view.

### Watchpoints

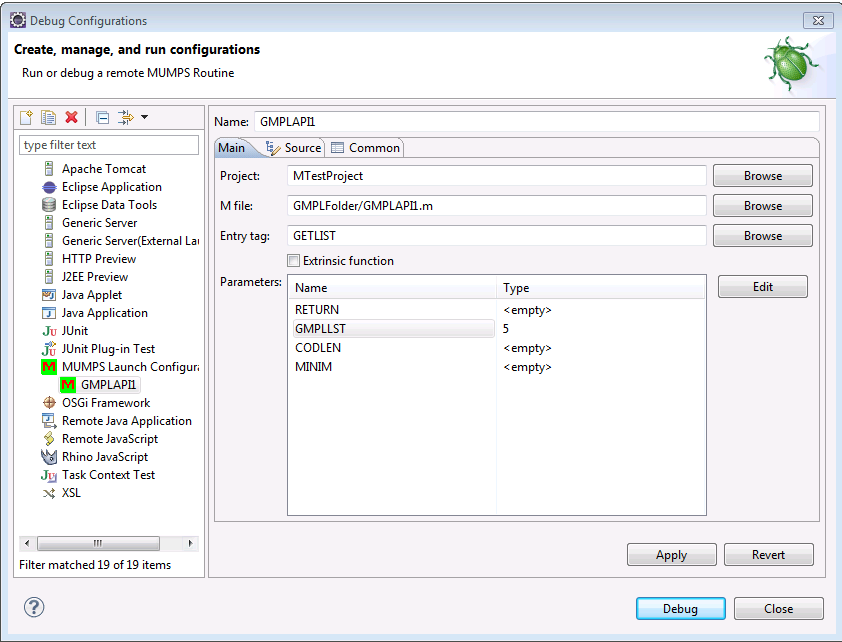
Variable watchpoints can be added. They are triggered when a variable value changes. They too are added from the breakpoints view, by clicking the pencil icon.

## Starting a Debug Session

To start the debugger, a new launch configuration must be created. Launch configurations are organized with respect to routines so it is recommended that they are create from file context menu since doing so will populate most fields in Debug configuration wizard.



Selecting Debug configurations brings Debug Configuration wizard where you can create a new Mumps launch configuration

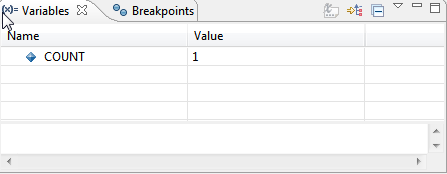


For a valid configuration Project, M File, Entry tag needs to be specified. When launched from a context menu M Tool auto populates Project and M File fields. Once an Entry tag is selected if there are parameters for the entry tag their values can be specified in the parameters table. The name of the parameters is automatically populated based on the Entry Tag. If the Entry Tag is an extrinsic function the check box for the same must be checked. Currently MTools do not automatically fills or validates this check box.

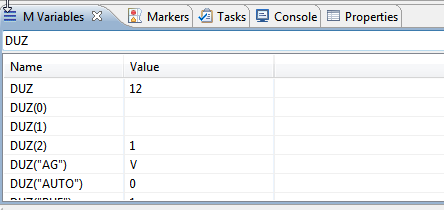
Once all the information entered then you can click on Debug to start the session. Note that once a debug launch configuration is created you can use M Application shortcut to launch the debug session for a particular file. If there are multiple configurations a selection dialog will be presented to select one.

## Variables

Variables will be displayed in the default Eclipse Debug Variables view. But only those variables created after the debug session started are shown.

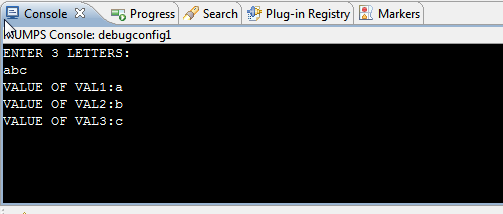


Since the default view isn’t showing all variables currently defined on the server, there is a new custom view created for that. Open it by clicking on ‘Window’ 🡪‘Show View’🡪‘Other…’ and choose ‘M Variables’. It can also be filtered by using the text box on top of the variable viewer.



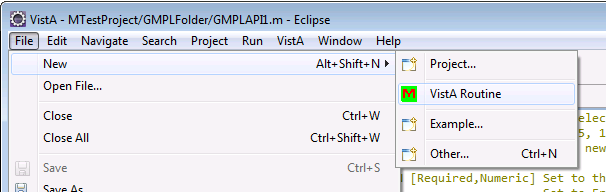
## Interactive Console

MDebug support is an interactive console which displays MUMPS WRITE commands and can capture input from READ commands. The console uses the default console view and automatically displays itself as WRITE commands output new input or when a READ command is ready to accept input.

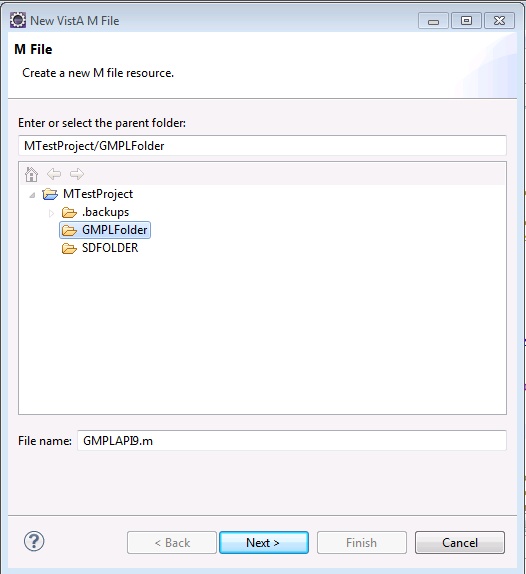


# New VistA File Wizard

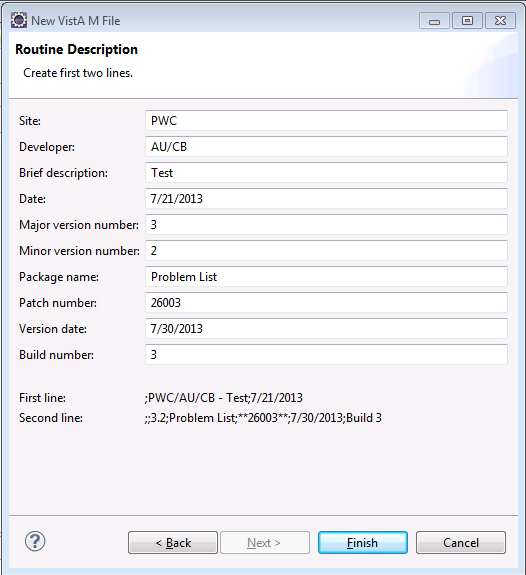
MTools provides a new New VistA File wizard from File->New menu.



In the first page of the wizard the name and folder of the new VistA file is specified.



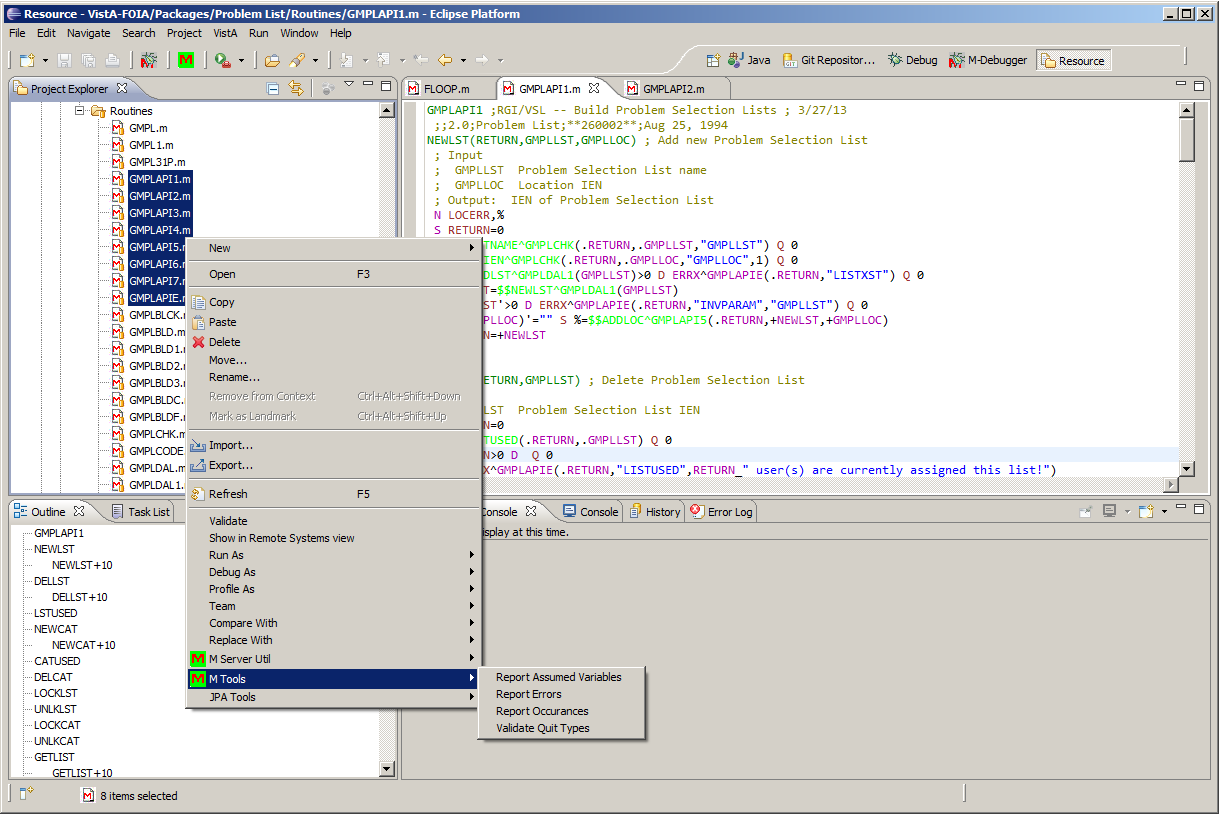
The file is restricted to be an M file. The second page of the folder helps to create the first two lines of a VistA routine which must follow certain rules.

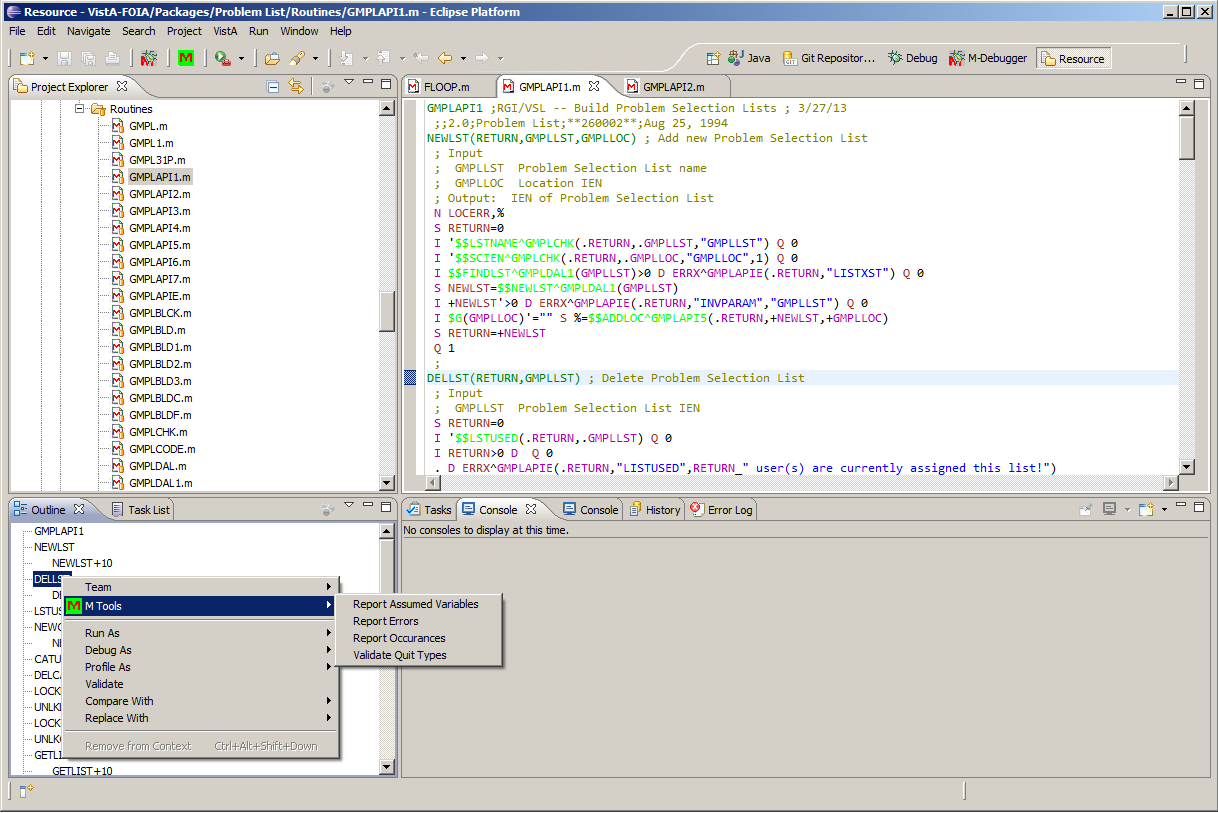


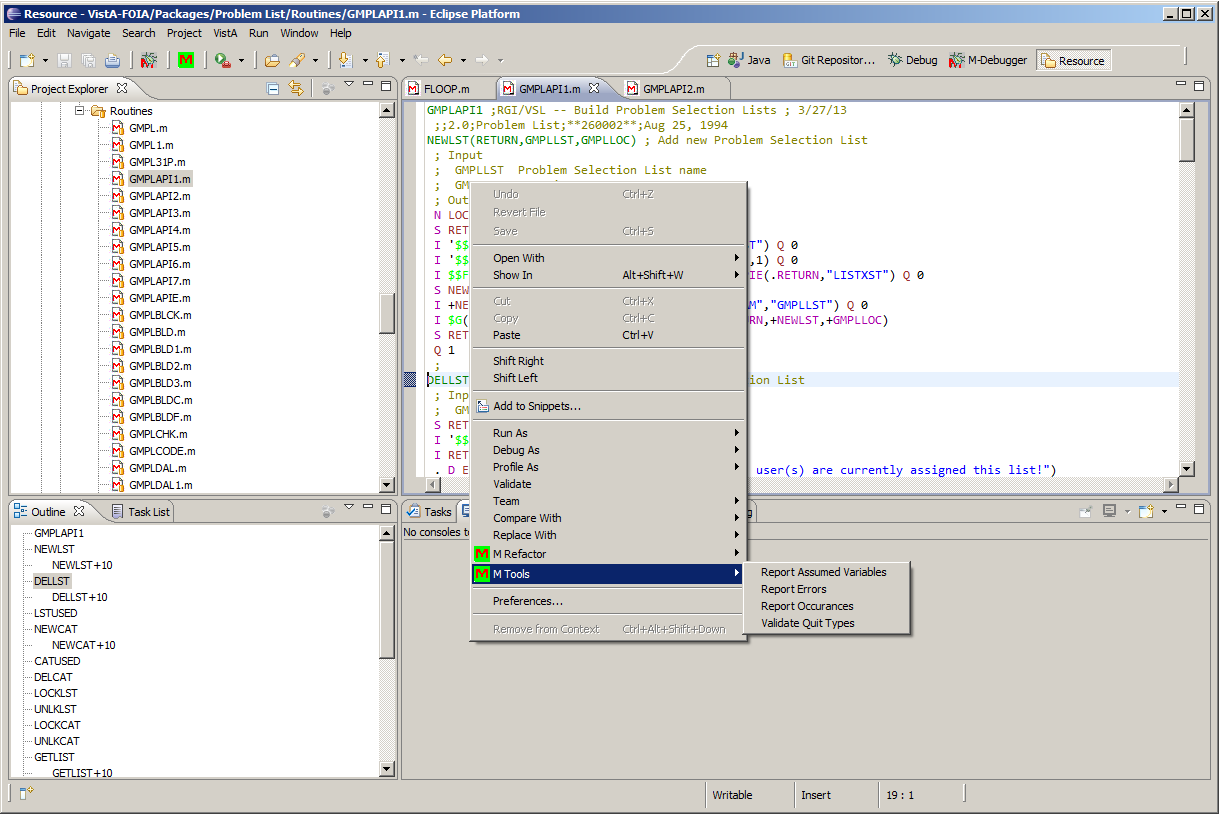
Note that new VistA files are not automatically saved to the server. When you hit finsh in the wizard the file is opened in MEditor and user can save the routine to the server as described in the previous sections. New VistA M File wizard replaces functionality in the previous version of M Tools where files were update on the server side.

# MTools Context Menu Tools

A number of M code validation and analysis tools are available in MTools IDE. These tools can be found under M Tools menu item in context menus of Project Explorer view, Outline view, and MEditor. Note that these tools all client based and do not use M server.





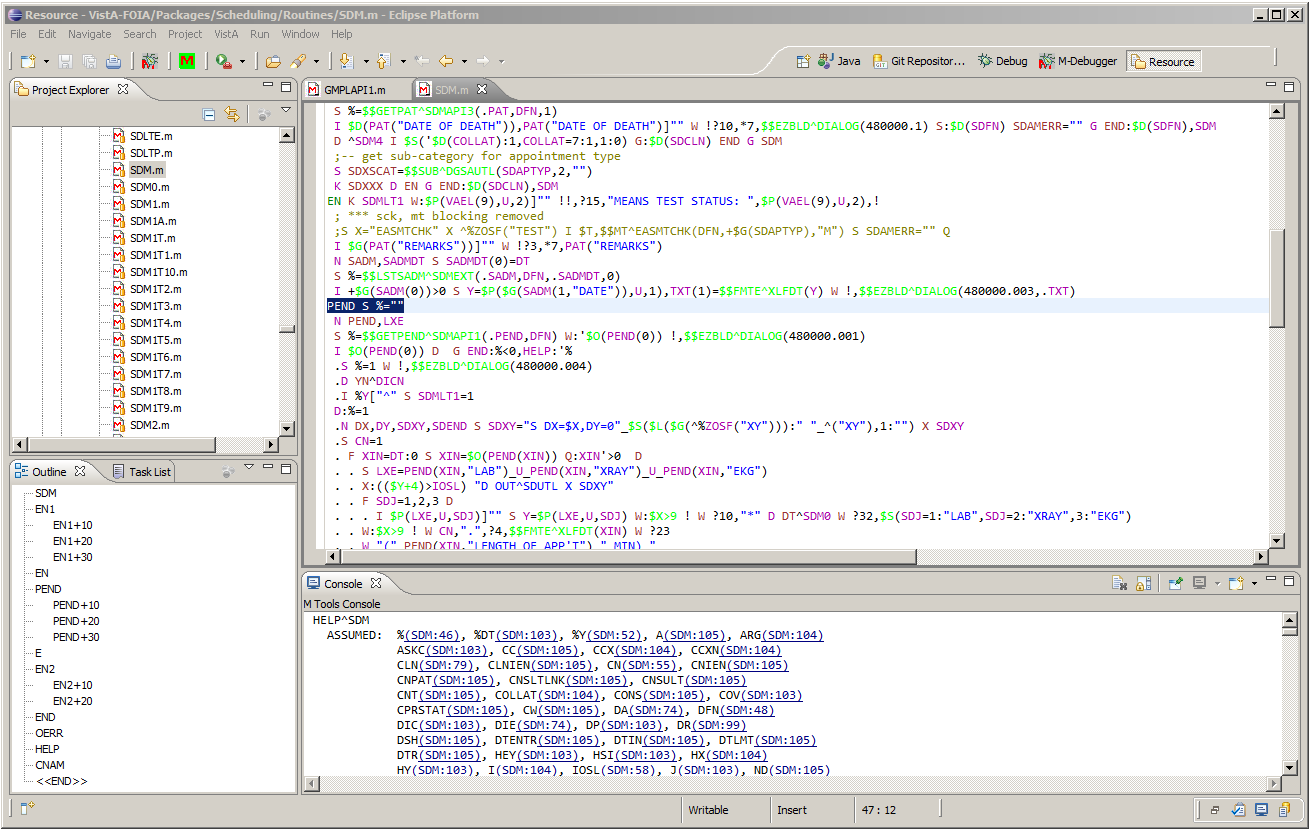


Currently there are four tools: Report Assumed Variables, Report Errors, Report Occurrences, and Validate Quit Types. Each of these tools is run for each entry point tag in routines. When selected from Outline view, they are run for a specific tag while; when they are run from MEditor, they are run for all the tags in the routine that is being edited in MEditor. All the tools can be run on multiple file selections from Project view.

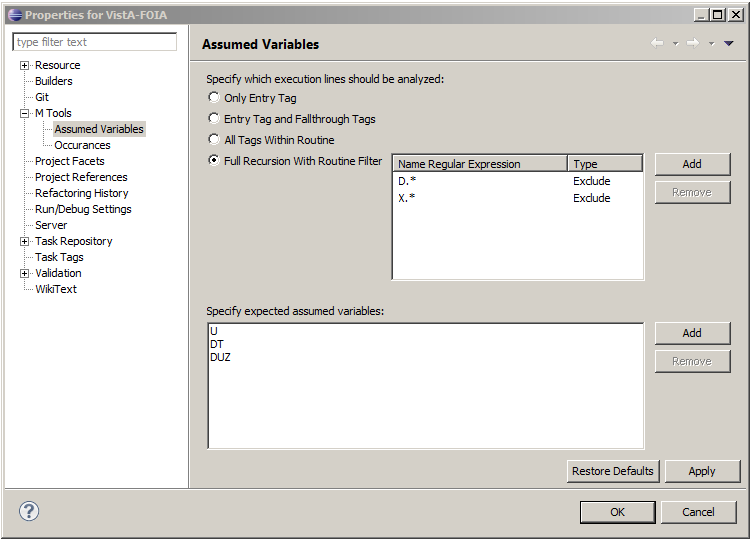
### Report Assumed Variables

The tool finds all the variables that are used in the entry tags that are not “newed”. Using no “newed” variables results in un-maintainable code and should be avoided. Running the tool writes the list of assumed variables in the Console per entry point tag. The output also includes clickable location of the assumed variable and when selected it brings the line that the assumed variable is on to focus.

Note that some of these tools do recursive analysis. However, recursive analysis is limited to the routines that are on the client Eclipse project and does not consider routines that might be on the server, but not on the client.



Similar reports are also generated by XINDEX; however, the reports here are generated by a Java-based MUMPS code analyzer that can also recursively search for all the tags that entry tag under test calls. Recursion specifics are configurable, and configuration and can be found under Project Properties’ M Tools section. (Note: Project Properties is available under the top Project menu item.)



Users can disable recursive analysis by selecting “Only Entry Tag” options. Selecting “Entry Tag and Fallthrough Tags” adds the tags that follow the entry tag to the analysis when entry tag does not end with an unconditional QUIT command. When option “All Tags within Routine” is selected, all the tags which are local to the routine and called by DO or GOTO commands and extrinsic functions are also included in the analysis. Full recursion is available with “Full Recursion with Routine Filter” option where can include all DO and GOTO commands and extrinsic functions in the analysis. Users can specify routines that would be included or excluded to limit the extent of the recursion. The routines are identified by name regular expressions.

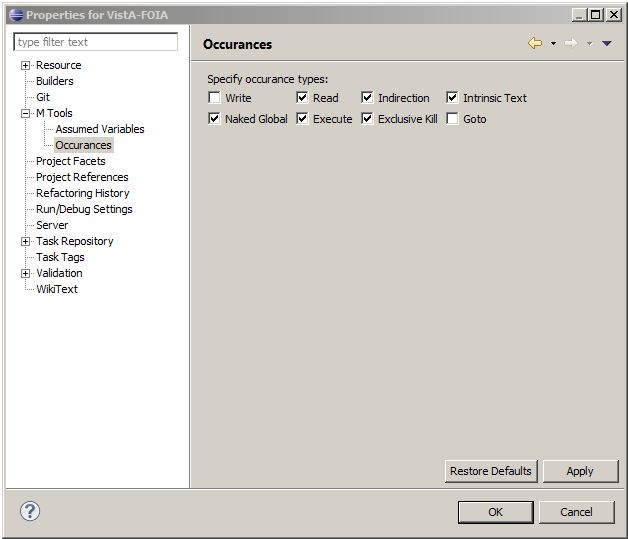
In addition to the setting that configures the recursive analysis, users can also specify the variables that should be excluded from the analysis. This is useful when there are variables that are assumed by design.

### Report Errors

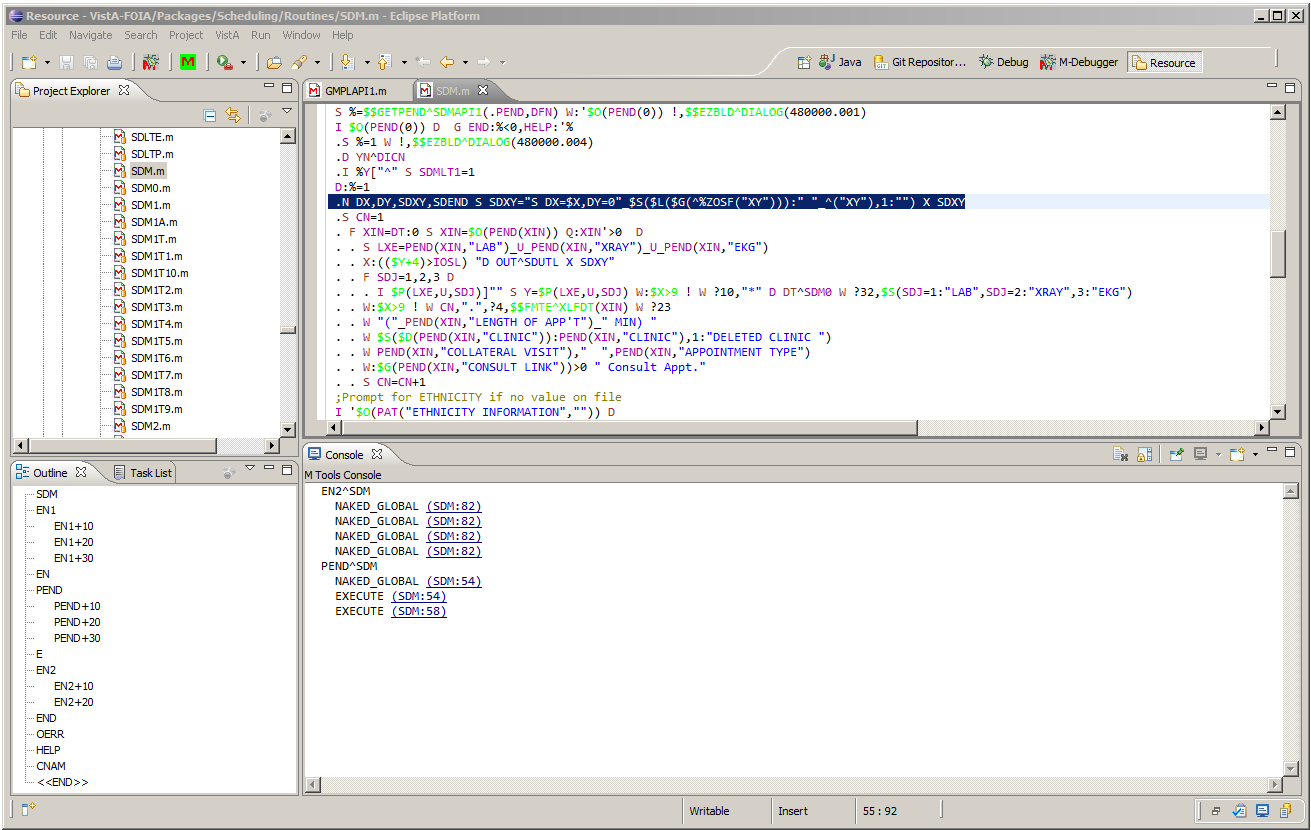
“Report Errors” writes out all the syntactical errors to the Console. The validation is independent from XINDEX validation and implemented on Java. It serves as confirmation of XINDEX results or can be used offline validation tool where users do not need to save the routine to the server to get validated.

### Report Occurrences

Report Occurrences writes a number of MUMPS language constructs used in the routines to the Console. Most of these constructs should be avoided in VistA development and this report identifies the location which can be reviewed easily. Users can configure the constructs to report on from Project Properties’ M Tools section.



The following shows an example run; all the constructs include clickable locations.



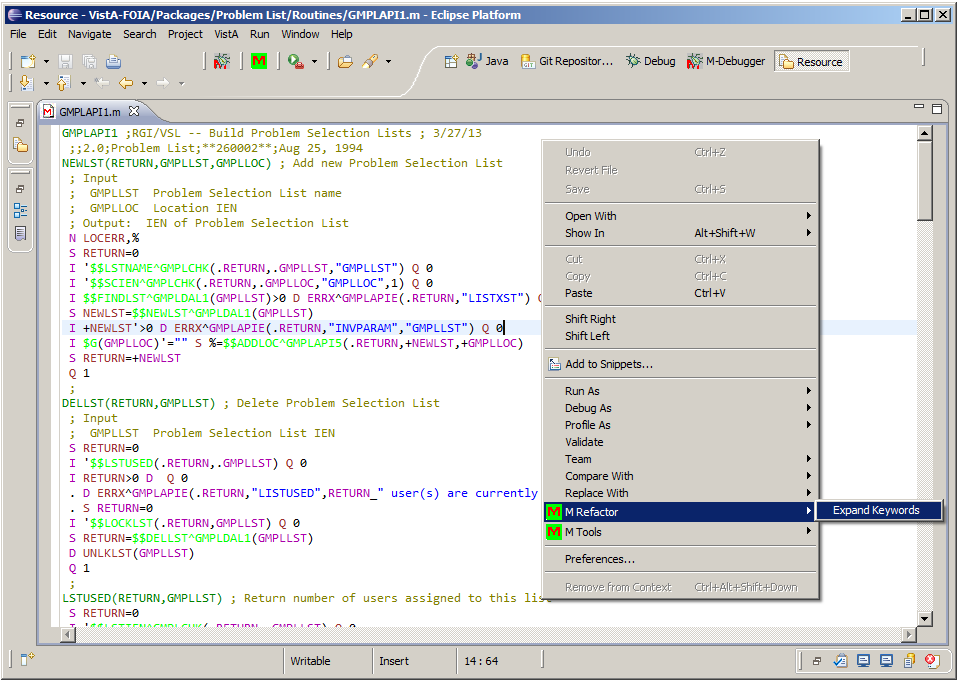
### Report Quit Types

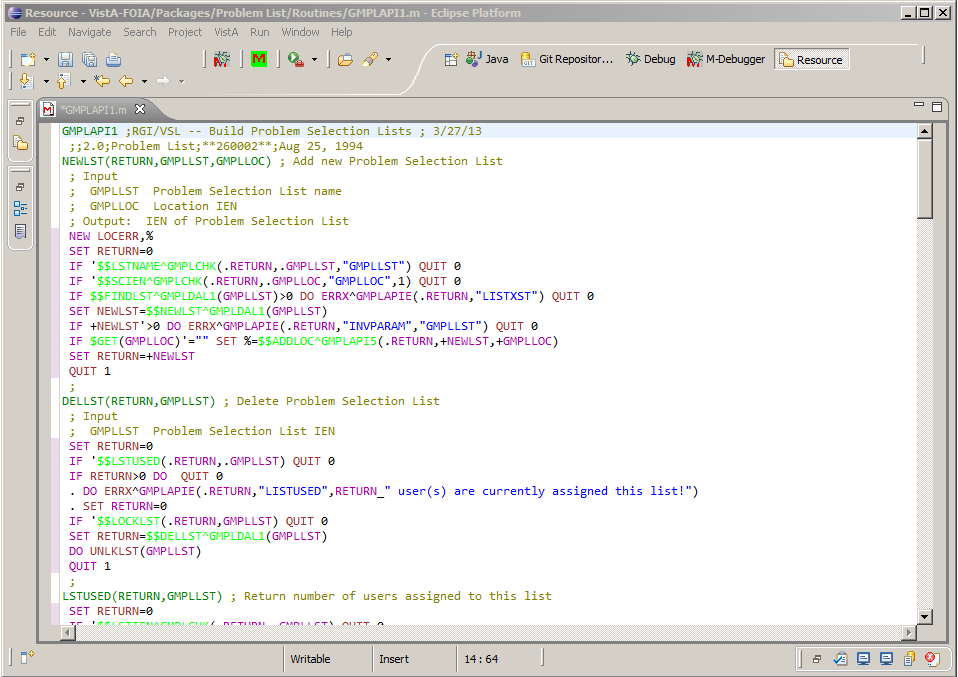
Report Quit Type performs two analyses: the first ensures that each entry tag has consistent QUIT command; all QUIT commands in an entry tag must either all return values or all not return values. The analysis is recursive with respect to GOTO commands.

The second analysis looks for DO/GOTO commands that are called for extrinsic functions and SET commands that are called for tags that not return value.

# M Refactor Context Menu Items

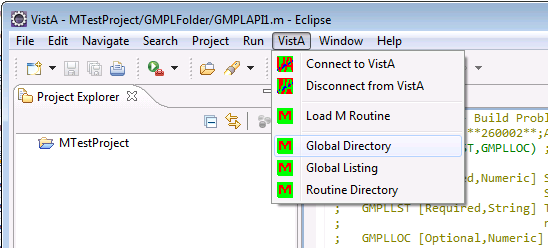
M Refactor context menu item is available only from MEditor and it has current single action “Expand Keywords”. This item changes all the command and intrinsic function mnemonics to their full form. The following screenshots illustrate before and after Expand Keywords state of a routine:





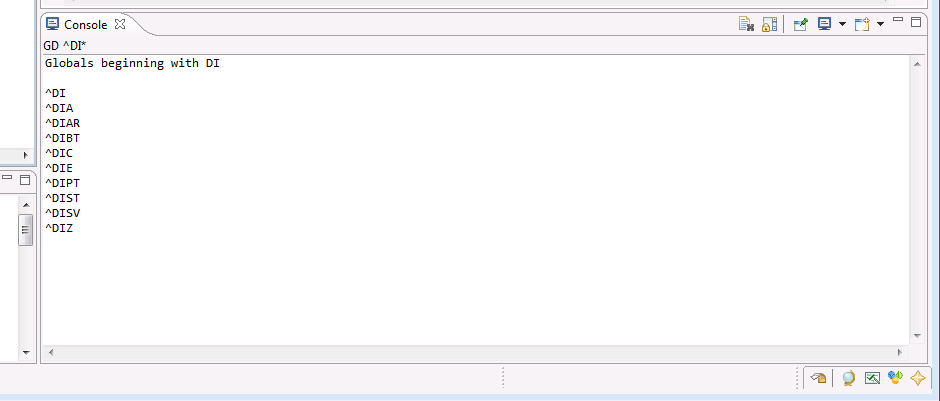
# VistA Main Menu Utilities

MTools provides three utilities that are not specific to any project but give information on routines and globals on the server. These utilities are available from VistA main menu



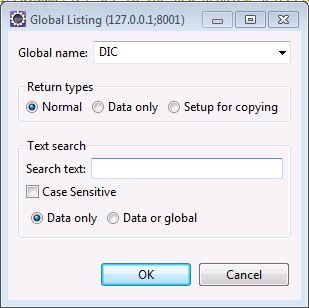
## Global Directory

This utility lists all the globals with a specified namespace. When this menu item is selected user is prompted to enter the namespace. The results are displayed in the Console view. The following shows the output when the namespace is specified to be DI.



## Global Listing

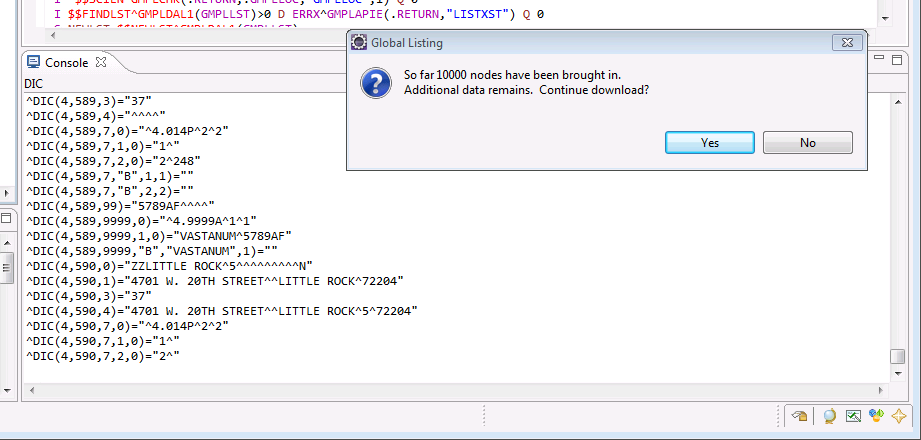
This utility lists the content of a specified global. The global information to be listed is entered in a dialog which is displayed when Global Listing is selected.



If Normal return type is selected the global, indices and data are displayed. For Data only the global and indices are not displayed. If Setup for copying is selected the output is a legal MUMPS script with SET commands that can be coped and pasted to a routine. In all cases the output is written to Console View.

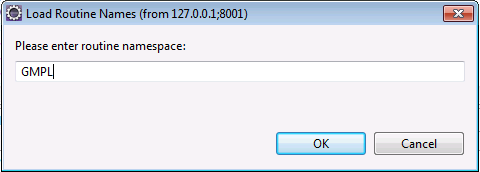
You can further filter the results by specifying a search text. For this case only those entries that contains the search text is displayed.

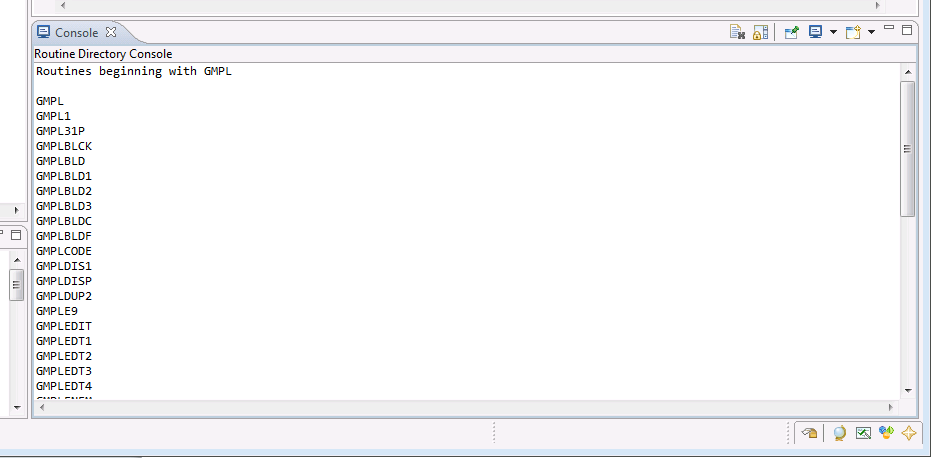
The result below shows the results for global DIC as specified in the above dialog. Note that the results are shown in 1000 line increments.



## Routine Listing

This is similar to Global Listing but show all the routines in a specified namespace. The result below shows when user specifies namespace GMPL. The namespace input is shown immediately after user selects Routine Listing





**Approval Signatures**

This section is used to document the initial approval of the draft Open Source Electronic Health Record (EHR) Services MTools Installation and Usage Guide, and any subsequent updates and modifications; updates will be submitted as needed.

All members of the governing Open Source EHR Services Management Team are required to sign. Please annotate signature blocks accordingly.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*LaWanda Wells, Contracting Officer’s Representative*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed: Date:

*Maureen Coyle, Program Manager*