



OSEHRA

Open Source Electronic Health Record Agent

Instructions for Establishing and Testing the OSEHRA Code Base

Purpose

This document describes the steps required to obtain the OSEHRA open source VistA codebase from the OSEHRA code repository, establish a working test environment, execute the tests, and view the results on the OSEHRA Software Quality Dashboard.

The document comprises six sections corresponding to:

- [Obtaining and installing required auxiliary programs](#)
- [Obtaining the testing code \(fixtures and drivers\)](#)
- Configuring the environment
 - [Installing Intersystems Caché](#)
 - [Importing the OSEHRA Code Base into Caché](#)
 - [Installing GT.M](#)
 - [Importing the OSEHRA Code Base into GT.M](#)
- [Setting up the testing environment](#)
- [Running the tests and uploading testing results to the OSEHRA Software Quality Dashboard](#), and
- [Reviewing the results](#).

Obtaining and Installing Required Auxiliary Programs

Testing the code requires several auxiliary programs to be available in the testing environment. The first component is a test driver, CMake, that configures the testing system, manages the generation of automatic tests from a template, executes the tests, and reports test results to the OSEHRA Software Quality Dashboard. In addition, some of the tests require scripting of user responses. For GT.M installations, the Linux utility *Expect* provides the scripting interface, while for Caché on Windows, this support is provided by the native scripting ability of the Caché Terminal. Instructions on how to obtain and install these programs follow.

CMake testing utility

The testing capabilities are contained within the CMake family of programs. Installers for Windows, Linux and Mac can be downloaded from: <http://cmake.org/cmake/resources/software.html>. Go to the website, select the appropriate binary release and follow the instructions on installing CMake.

Scripting interfaces to OSEHRA codebase

The testing currently supports two types of MUMPS environments: Caché on Windows and GT.M on Linux. Both programs use an interface to script the interaction with XINDEX.

For **GT.M** on Linux, we support *Expect* as the interface to the OSEHRA code base. Expect is available from SourceForge at <http://sourceforge.net/projects/expect/> or through “*sudo apt-get install expect.*”

For **Caché** on Windows, we do not require an external program to interface to the OSEHRA code base. The Caché Terminal has its own scripting ability which will be utilized for this purpose. This will be installed with the Caché system.

Git

Git can be downloaded from <http://git-scm.com>. It is a “free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.” It allows developers to create their own instances of a repository and submit changes to it and, if permission is given, submit those changes to the original repository. Standard installation scripts and documentation are available directly from the Git website.

Python

Python is a scripting language used to manage the packing and unpacking of the OSEHRA code base. It is free and Open Source. Python can be downloaded from <http://www.python.org/download/> using the instructions on the web site.

[Return to the Beginning](#)

Obtaining the Testing Code

The OSEHRA testing suite is maintained separately from the OSEHRA VistA source and must be downloaded independently. It is published in a repository hosted at

<http://code.osehra.org/OSEHRA-Automated-Testing.git>

Use Git to make a local clone of this repository:

```
$ git clone http://code.osehra.org/OSEHRA-Automated-Testing.git
$ cd OSEHRA-Automated-Testing
```

We will refer to the location of the folder containing the testing code as the *Testing Source Directory*.

[Return to the Beginning](#)

Configuring the Testing Environment

VistA requires an underlying MUMPS environment. OSEHRA currently supports two options: Caché for Windows systems and GT.M for Linux systems.

Installing InterSystems Caché

The following instructions were adapted from Nancy Anthracite's document entitled [Installing VistA With Single User Version Caché 5.2](#), which was created to guide a user through installing InterSystems Caché onto a Windows operating system. The instructions were using an older version of Caché; this uses the most recent trial version and shows the most recent Management Portal interface.

Trial versions of the Caché installer can be downloaded from <http://download.intersystems.com/download/register.csp>. This installation guide uses Caché 2011.1. If you already have a Caché installation and are looking to install VistA as an additional database, you do not have to re-install Caché. Please use your existing installation and pick up the instructions at the point where the folder is created within the mgr folder.

Download and Install Caché

Download the Caché installer from the above link and double click on the downloaded .exe file. The first window that requires interaction is the Licensing Agreement shown in Figure 1. Agree to the license in order to continue in the installation process.

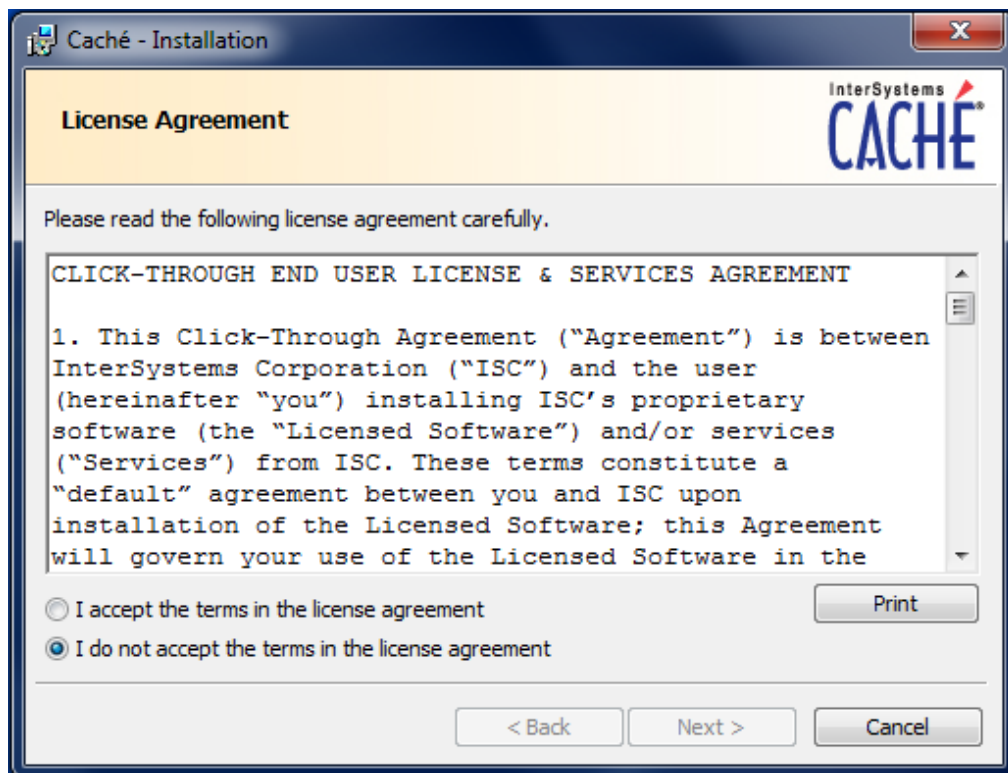


Figure 1 - License agreement within the Caché installation wizard.

The next window, Figure 2, asks to set the directory in which Caché will be installed. Most users will be able to accept the default path. If more than one instance is found on the machine, the next instances will be denoted with a number appended to the end of the instance name.

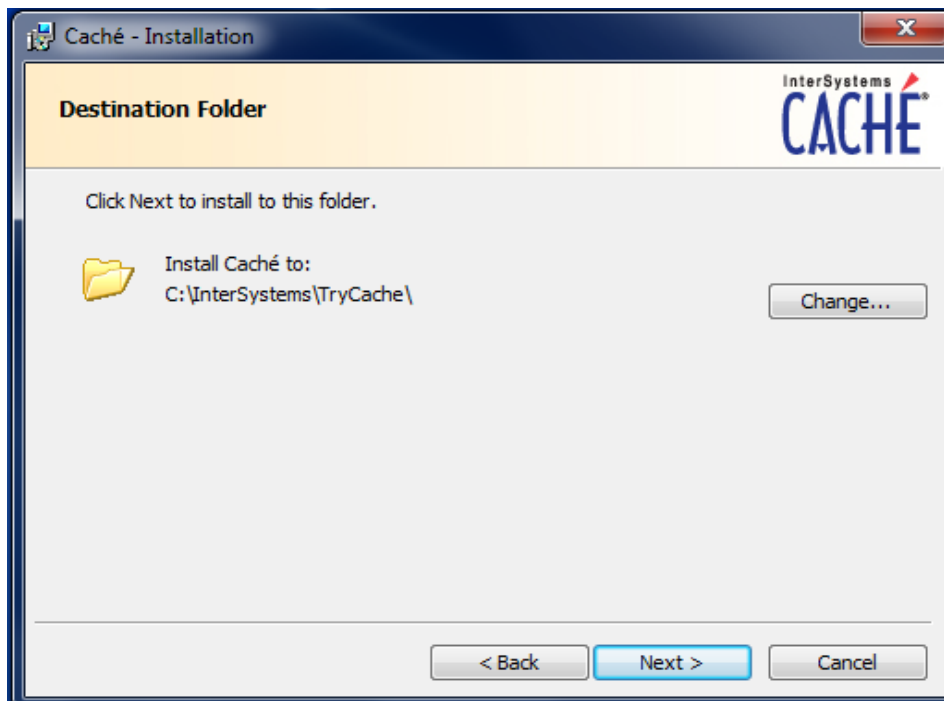


Figure 2 - Setting the installation path for Caché

Once the install directory is set, the installer will display a summary of the instance that will be installed in the process, Figure 3 . There is the option to enter a license if you have one, but this is not a required step for the current testing configuration. Figure 4 shows the final screen of the installation process.

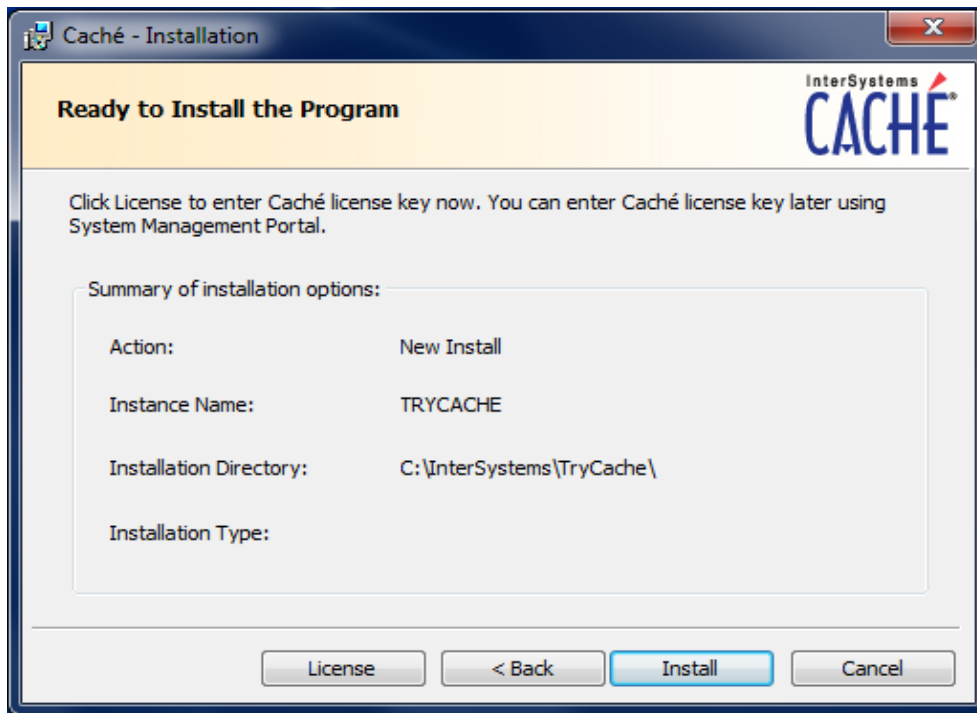


Figure 3 - Summary of Caché installation options

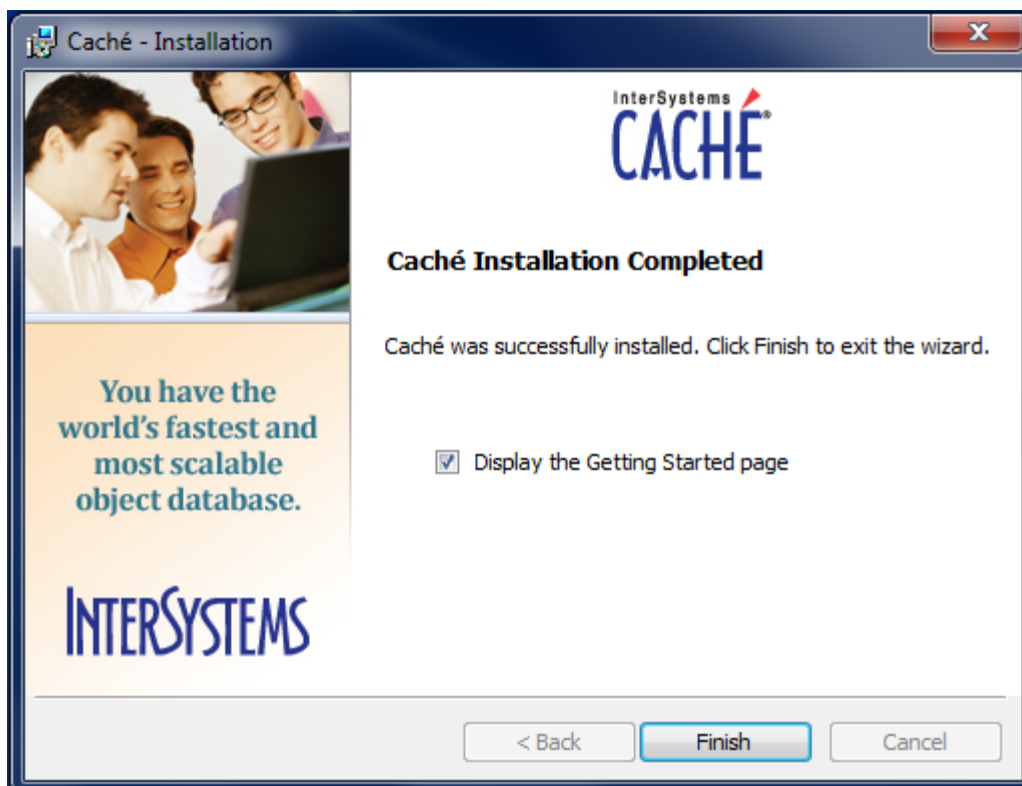


Figure 4 - Final window of Caché Installation

The first sign of a correctly installed and running instance of Caché is the Caché Cube in the taskbar shown in Figure 5.



Figure 5 - Screenshot of Caché Cube in taskbar.

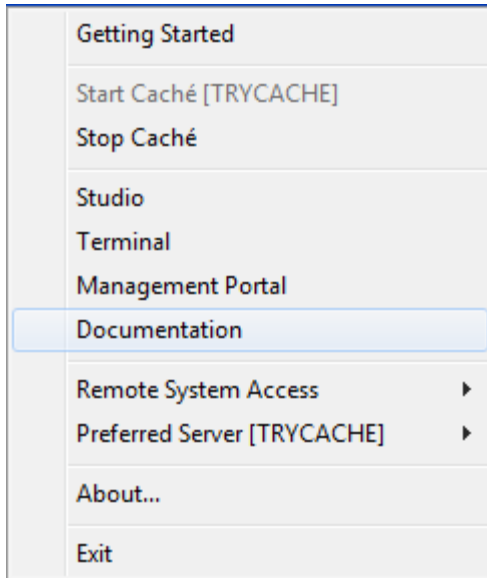


Figure 6 - The main menu that shows when the Caché Cube is clicked.

Clicking on the Cube will open a menu, Figure 6, that displays the options to interact with the Caché database. The next test to ensure that your Caché instance is working is to open the documentation. This will bring up the documentation home page in your default web browser (Figure 7).

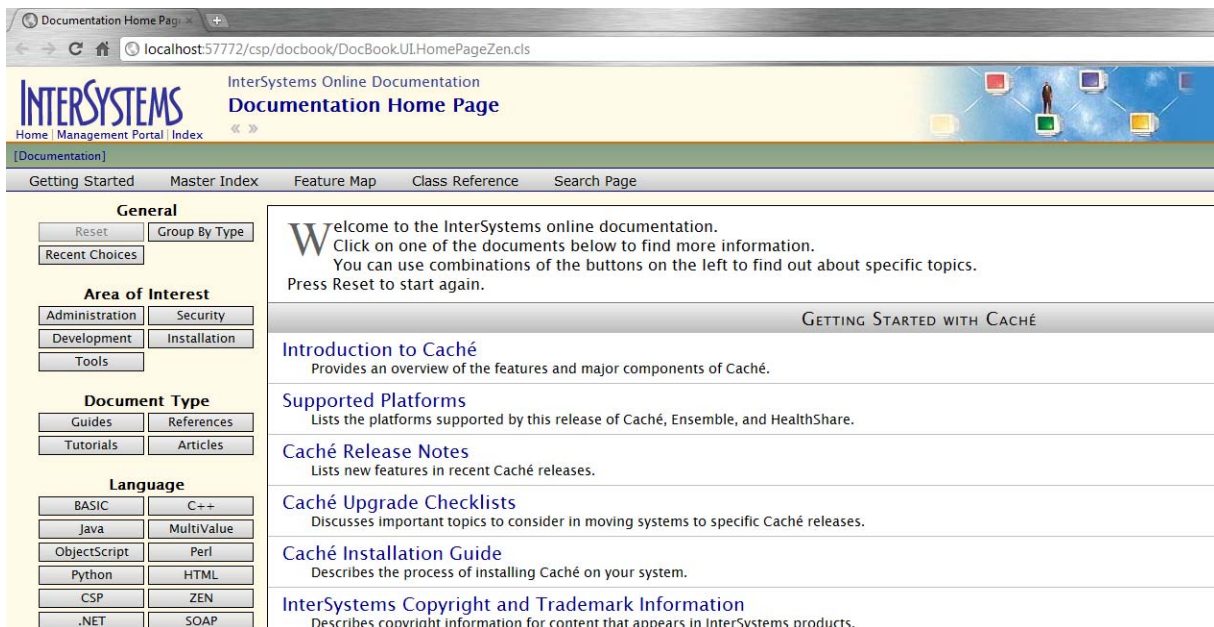


Figure 7 - Documentation font page in the web browser Google Chrome.



Figure 8 - The search window of the Caché documentation.

Configuring Caché

Once Caché is installed, it is time to create the proper folders and environment to run the VistA instance within Caché. The first step is to go to the *mgr* folder of Caché and create a new folder as in Figure 9. This folder will hold the database file cache.dat that will contain the VistA routines and globals.

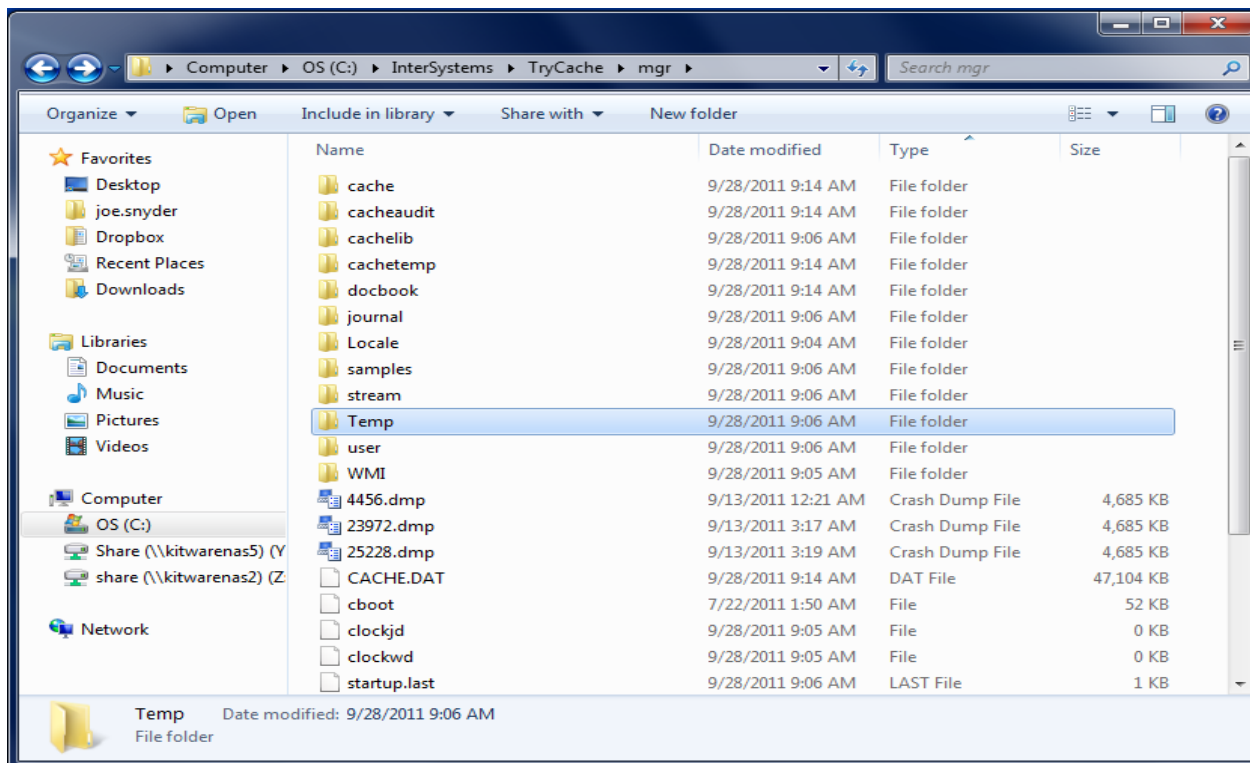


Figure 9 - mgr folder prior to creation of VistA folder

In the example shown in Figure 10, the folder has been given the name "VistA". *While the choice of name has no bearing on the installation, the testing code requires that the Namespace name chosen in Figure 21 matches the folder name created in this step.*

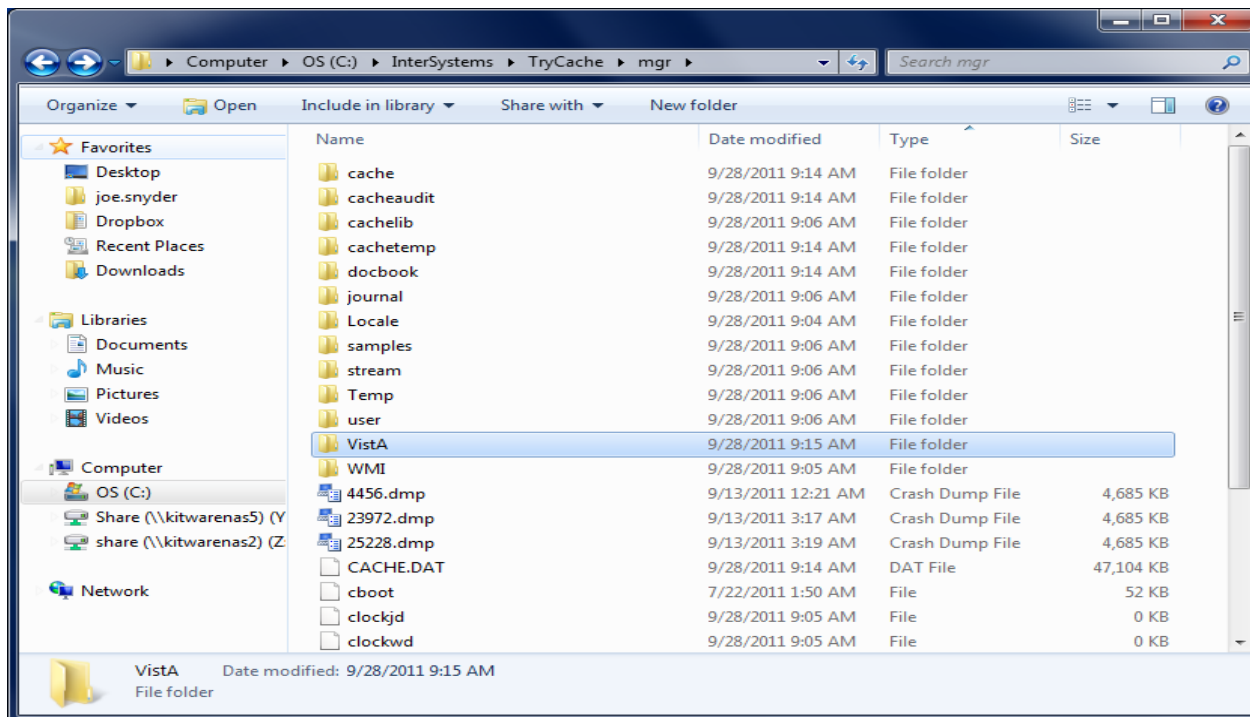


Figure 10 - mgr folder post-creation of VistA folder

At this point we are ready to stand up the VistA instance. Right click on the Caché cube and select *Management Portal* of Caché (Figure 11). This link will open a Management Portal web page as shown in Figure 12. Click on *System Administration* to show administrative options.

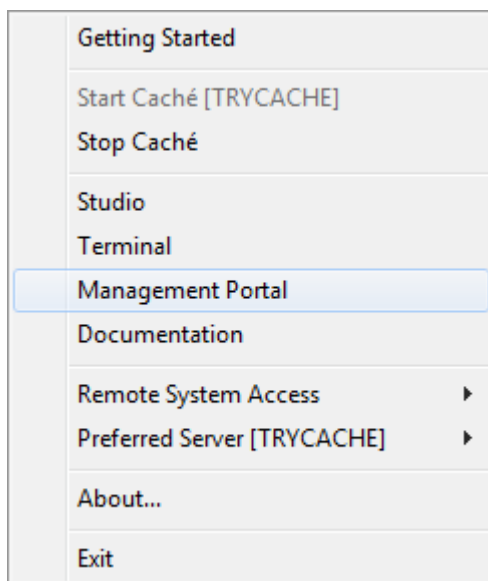


Figure 11 - Management Portal link in Caché

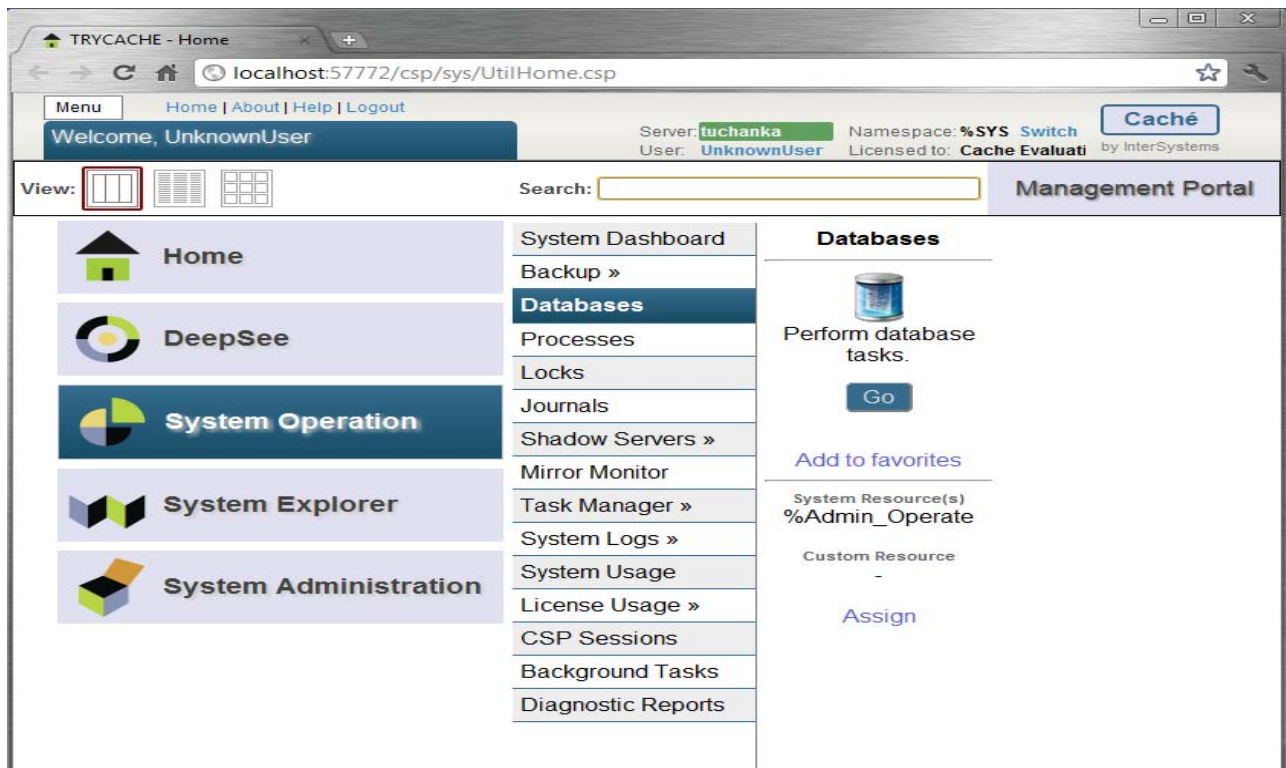


Figure 12 - Main page of the Management Portal

System Administration shows those options that can be used to change the Caché system. Our goal is to use the *Configuration* function to create and initialize an empty database that can then be filled with the Vista routines and globals. Starting from Figure 13, click on *Configuration*, *System Configuration*, and *Local Databases* to arrive at Figure 14.

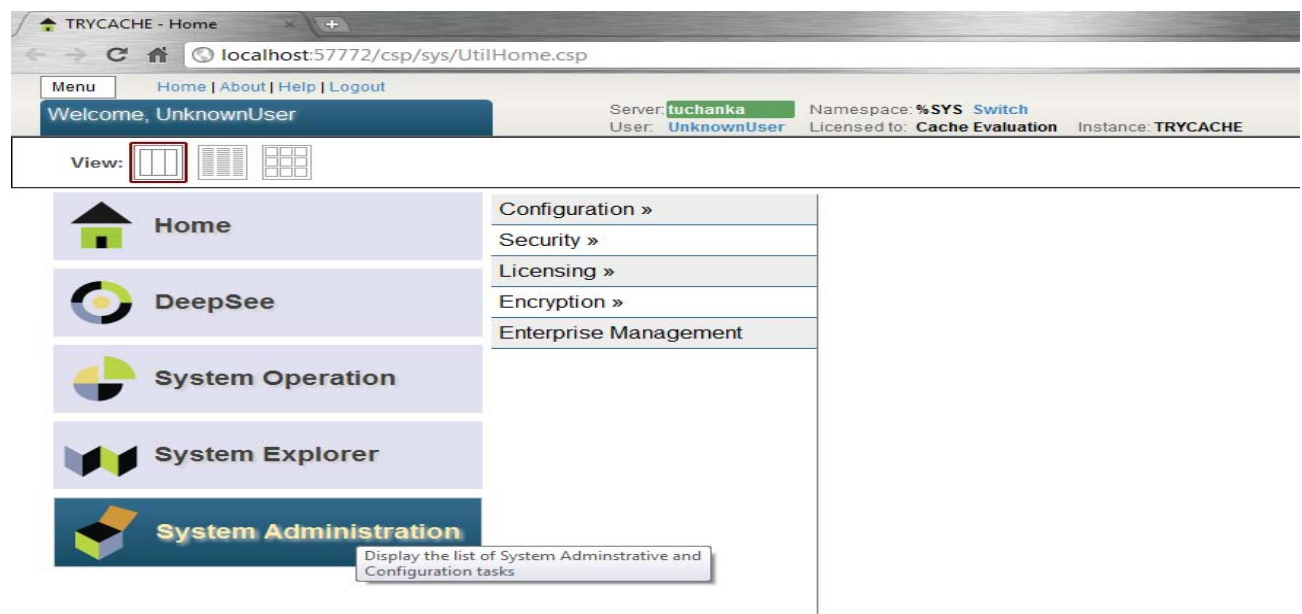


Figure 13 - System Administration page of Management Portal

Create the database by clicking on the *Local Databases* tab and then selecting *Go*.

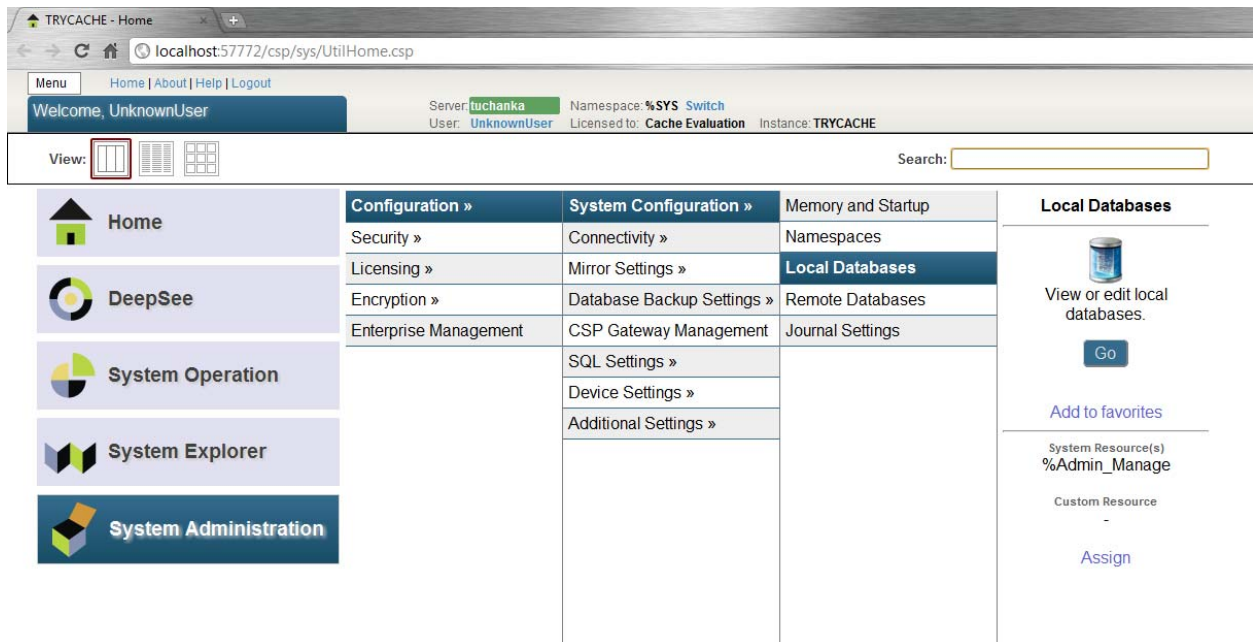


Figure 14 - System Configuration menu.

This resulting page contains the list of all of the local databases Figure 15. All of the selections shown were created automatically during the installation of Caché. Create a new database by clicking on the “Create New Database” button. This will bring up a wizard (Figure 16).

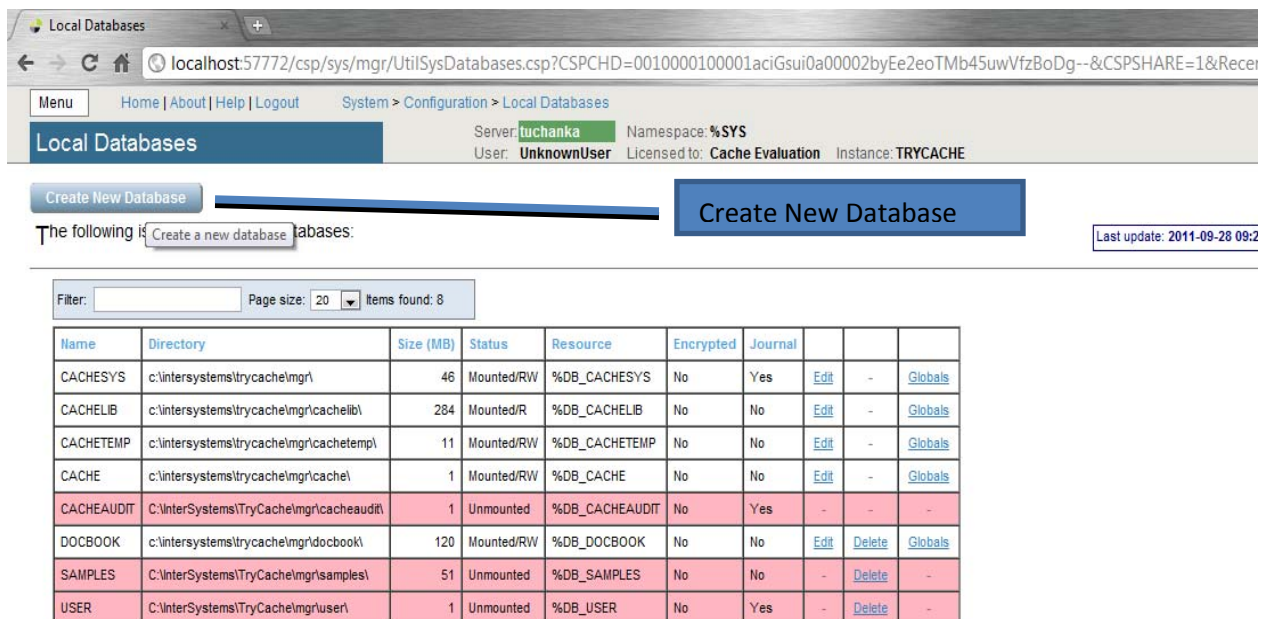


Figure 15 - Local Databases page with pointer to Create New Database button.

Set the directory entry to the folder that you created (Figure 10) and set the database name. We recommend using the same name as the folder, but this is not necessary. When satisfied, select *next* to proceed to Figure 17.

Database Wizard - Google Chrome

localhost:57772/csp/sys/mgr/UtilSysDatabaseWizard.csp?IsRemote=0&\$NAMESPACE=&

Database Wizard

This wizard will help you create a new database.

Enter the name of your database:

Database directory:

(Base directory is empty. Please enter one or 'Browse' to select one.)

< Back Next > Finish Cancel

Figure 16 - First page of the Database Wizard.

It is not necessary to change any of the default settings to enable testing and we recommend simply hitting *Finish* to proceed to Figure 18. However, if there are known required settings for the current site, these settings can be modified. Verify that the newly created database appears in the database listing.

Database Wizard - Google Chrome

localhost:57772/csp/sys/mgr/UtilSysDatabaseWizard.csp?IsRemote=0&\$NAMESPACE=&

Database Wizard

Enter details about the database.

How big do you want the initial database to be?

Initial Size (MB):

Block size is the size of the blocks that the databases uses.

Block size for this database will be: 8KB

Do you wish to journal globals in this database?

Journal globals:

You may create an Encrypted Database if Encryption is activated.

Encrypt Database? ☐ (Encryption is not activated)

< Back Next > Finish Cancel

Figure 17 - Details of the Database Wizard

Local Databases

Server: **tuchanka** Namespace: **%SYS**
User: **UnknownUser** Licensed to: **Cache Evaluation** Instance: **TRYCACHE**

Create New Database

The following is a list of the local databases: Last update: 2011-09-28 09:31:58.131

Filter: Page size: 20 Items found: 9

Name	Directory	Size (MB)	Status	Resource	Encrypted	Journal			
CACHESYS	c:\intersystems\trycache\mgr\	46	Mounted/RW	%DB_CACHESYS	No	Yes	Edit	-	Globals
CACHELIB	c:\intersystems\trycache\mgr\cache\lib\	284	Mounted/R	%DB_CACHELIB	No	No	Edit	-	Globals
CACHETEMP	c:\intersystems\trycache\mgr\cache\temp\	11	Mounted/RW	%DB_CACHETEMP	No	No	Edit	-	Globals
CACHE	c:\intersystems\trycache\mgr\cache\	1	Mounted/RW	%DB_CACHE	No	No	Edit	-	Globals
CACHEAUDIT	C:\InterSystems\TryCache\mgr\cache\audit\	1	Unmounted	%DB_CACHEAUDIT	No	Yes	-	-	-
DOCBOOK	c:\intersystems\trycache\mgr\docbook\	120	Mounted/RW	%DB_DOCBOOK	No	No	Edit	Delete	Globals
SAMPLES	C:\InterSystems\TryCache\mgr\samples\	51	Unmounted	%DB_SAMPLES	No	No	-	Delete	-
USER	C:\InterSystems\TryCache\mgr\user\	1	Unmounted	%DB_USER	No	Yes	-	Delete	-
VISTA	c:\intersystems\trycache\mgr\vista\	1	Mounted/RW	%DB_%DEFAULT	No	Yes	Edit	Delete	Globals

Figure 18 - Database listing with the inclusion of the recently created Vista database.

We now will configure the namespace for the newly created database. Navigate back to the *System Configuration* menu (Figure 19), click on the *Namespaces* option, and then click on the “Create New Namespace” button (Figure 20) to open a wizard (Figure 21).

TRYCACHE - Home

localhost:57772/csp/sys/UtilHome.csp

Welcome, UnknownUser

Server: **tuchanka** Namespace: **%SYS** Switch
User: **UnknownUser** Licensed to: **Cache Evaluation** Instance: **TRYCACHE**

View: Search:

Home
 DeepSee
 System Operation
 System Explorer
 System Administration

Configuration »

Security »

Licensing »

Encryption »

Enterprise Management

System Configuration »

Connectivity »

Mirror Settings »

Database Backup Settings »

CSP Gateway Management

SQL Settings »

Device Settings »

Additional Settings »

Memory and Startup

Namespaces

Local Databases

Remote Databases

Journal Settings

Namespaces

View or edit namespaces.

Go

Add to favorites

System Resource(s)
%Admin_Manage

Custom Resource
-

Assign

Figure 19 - Choosing Namespaces from System Configuration Menu

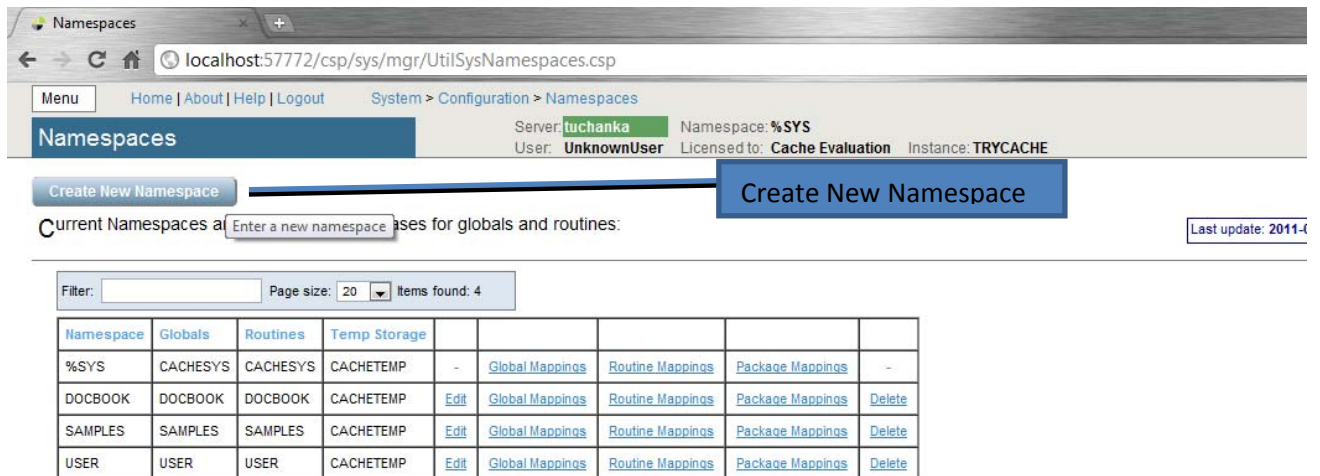
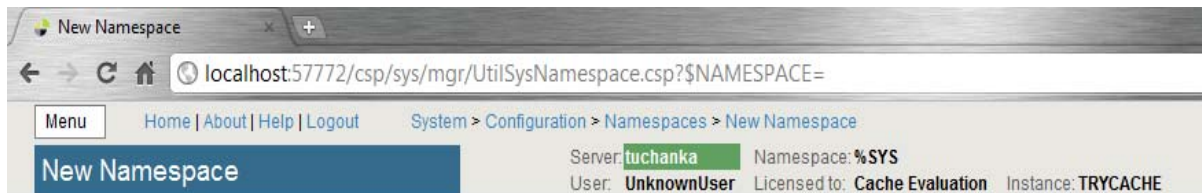


Figure 20 - Namespace listing and button to create a new namespace.

In the wizard, enter the name of the namespace and then select the database created in Figure 16. *Be certain to name the Namespace the same as the folder created in* Figure 10. Click on “Save” to finish the Namespace creation and to return to the namespace listing.



Use the form below to create a new namespace:

Name of the namespace:

The default database for this namespace is a ☒ Local Database ☐ Remote Database

Select an existing database:

☒ Create a default Web application for this namespace

Copy namespace mappings from

Figure 21 - Choosing the name of the namespace and the database it maps to.

Verify that the new namespace is now in the list of current namespaces (Figure 22).

The next steps will be configuring the global and routine mappings, both of which are accessed from this page. We will focus on the global mapping first.

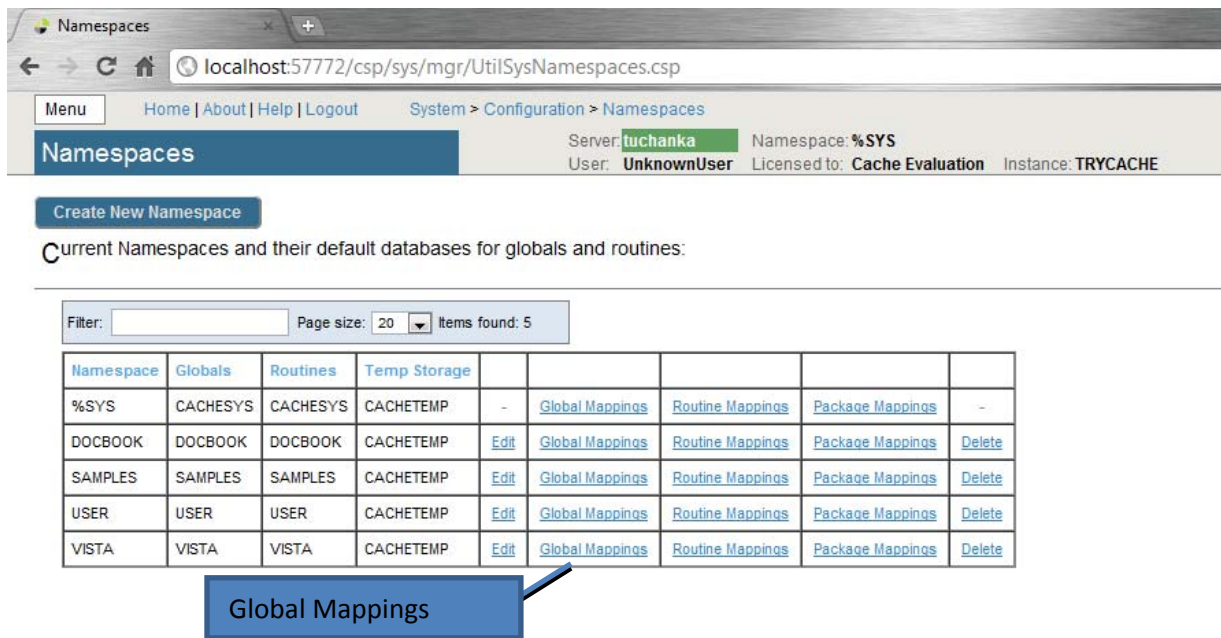


Figure 22 - Namespace listing with the new namespace in it. The boxes highlight the links for mapping globals and routines.

To create the new mapping, click on *New Global Mapping* (Figure 23). This opens the wizard in Figure 24



The global mappings for namespace VISTA are displayed below:

Global	Subscript	Database
No results		

Figure 23 - Setting the Global Mapping.

For Vista, there is only one global mapping that needs to be made. First set the Global Database location to the Vista database name, and for the Global Name enter "%Z*". This will map all globals that start with "%Z" to be specific to the Vista namespace. Click *OK* and the wizard will exit and display the new mapping in the window (Figure 25). Be sure to click on *Save Changes* before navigating back to the *Namespaces* page.

Global Mapping - Google Chrome

localhost:57772/csp/sys/mgr/UtilConfigGblMapping.csp?\$ZEN_POPUP=1&\$ID1=VISTA&IsNew=1&\$ID2=&zi

Global Mapping

Map a new global in namespace VISTA:

Global database location:

Global name:

Global subscripts to be mapped:

Subscript reference must begin with an open parenthesis. [Click here to see examples.](#)

Advanced

Figure 24 - Adding the %Z* mapping to the globals.

Global Mappings

localhost:57772/csp/sys/mgr/%25CSP.UI.Portal.Mappings.cls?MapType=Gbl&\$ID1=VISTA&\$ID2=1&\$NAMESPACE=

Menu Home | About | Help | Logout System > Configuration > Namespaces > Global Mappings

Global Mappings

Server: **tuchanka** Namespace: **%SYS**

User: **UnknownUser** Licensed to: **Cache Evaluation** Instance: **TRYCACHE**

Item(s) changed. Click 'Save Changes' to save into the Configuration file or 'Cancel Changes' to discard.

The global mappings for namespace VISTA are displayed below:

Global	Subscript	Database
%Z*		VISTA

Figure 25 - Page displaying the newly mapped globals.

The final step before Caché is ready for the import is to map the routines. From within the *Namespaces* menu in the *Management Portal*, click on the *Routine Mappings* link (Figure 26) which will bring you to the page that lists the current routine mappings for the VistA namespace.

Namespaces

Menu Home | About | Help | Logout System > Configuration > Namespaces

Server: **tuchanka** Namespace: **%SYS**
User: **UnknownUser** Licensed to: **Cache Evaluation** Instance: **TRYCACHE**

Create New Namespace

Current Namespaces and their default databases for globals and routines:

Filter: Page size: 20 Items found: 5

Namespace	Globals	Routines	Temp Storage					
%SYS	CACHESYS	CACHESYS	CACHETEMP	-	Global Mappings	Routine Mappings	Package Mappings	-
DOCBOOK	DOCBOOK	DOCBOOK	CACHETEMP	Edit	Global Mappings	Routine Mappings	Package Mappings	Delete
SAMPLES	SAMPLES	SAMPLES	CACHETEMP	Edit	Global Mappings	Routine Mappings	Package Mappings	Delete
USER	USER	USER	CACHETEMP	Edit	Global Mappings	Routine Mappings	Package Mappings	Delete
VISTA	VISTA	VISTA	CACHETEMP	Edit	Global Mappings	Routine Mappings	Package Mappings	Delete

Routine Mappings

Figure 26 - Selecting the namespace mapping link.

Much like the globals, there are no current mappings. Click on the New Routine Mapping button (Figure 27) to bring up the mapping wizard again (Figure 28).

Routine Mappings

Menu Home | About | Help | Logout System > Configuration > Namespaces > Routine Mappings

Server: **tuchanka** Namespace: **%SYS**
User: **UnknownUser** Licensed to: **Cache Evaluation** Instance: **TRYCACHE**

New Routine Mapping [Save Changes](#) [Cancel](#)

New Routine Mapping

The routine mappings for namespace VISTA are displayed below:

Routine	Type	Database
No results		

Figure 27 - Adding new Routine Mappings.

Again select the database location that corresponds to the Vista database, enter “%DT*” into the Routine name, and click Apply. This adds the first namespace mapping to the Vista database.

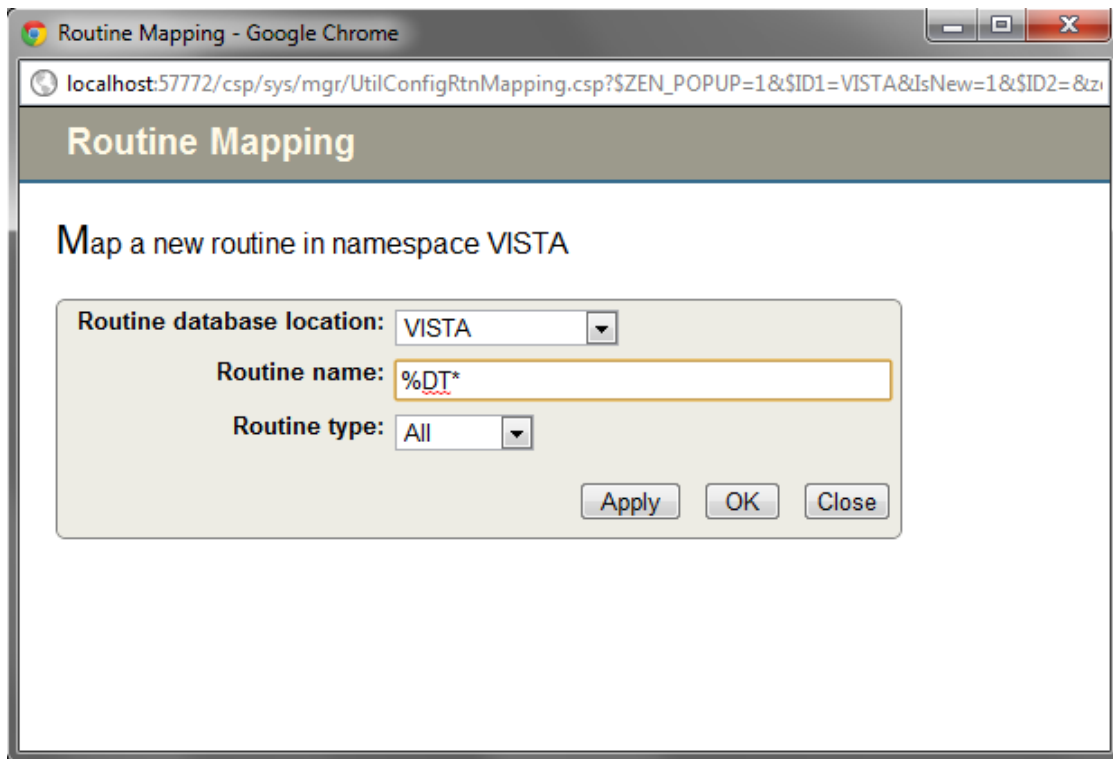


Figure 28 - Entering the first routine mapping.

There are six other mappings that need to be entered in the same manner -

%RCR	%XU*	%ZIS*	%ZO*	%ZT*	%ZV*
------	------	-------	------	------	------

After the final mapping is set, click OK to be sent back to the Routine Mapping page. You should now see the seven mappings from Figure 29 listed on the page. Click on the *Save Changes* button.

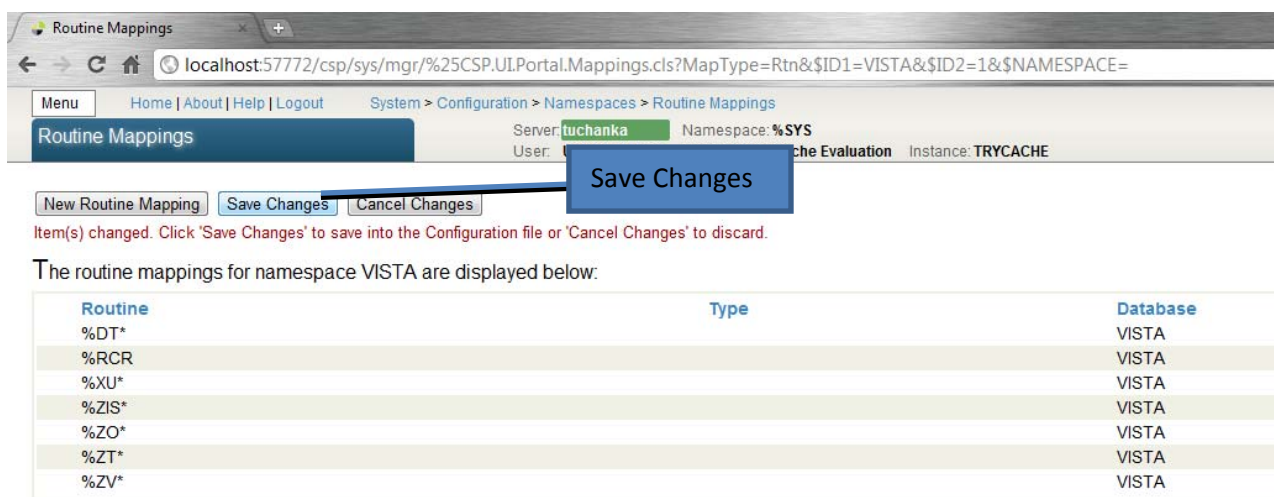


Figure 29 - Final listing of Routine Mappings and the Save Changes button.

The final step of preparing the Caché installation for testing is to set the instance to allow TELNET service. This is done though the System Administration > Security > Services menu (Figure 30).

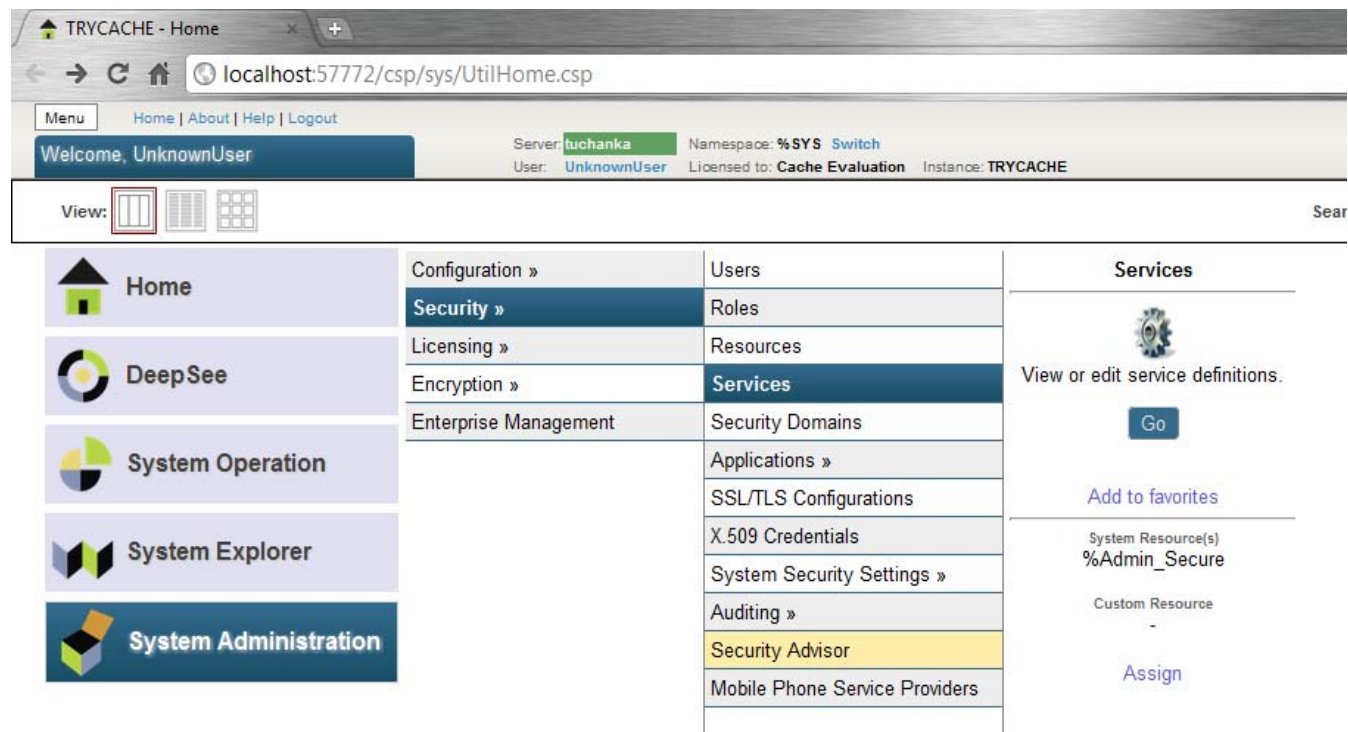


Figure 30 - Menu path to the Services option.

Click on **Go** to be brought to the menu which lists all services that are supported by Caché. Near the bottom of the list you will see the “%Service_Telnet” listing (Figure 31). Click on the link to bring up the “Edit Service” page (Figure 32).

Services are the primary means by which users and computers connect to Caché. The following services are currently available:

Name	Enabled	Public	Authentication Methods	Allowed Connections	Description	Two-Factor Enabled
%Service_Bindings	Yes	N/A	Password, Unauthenticated	Unrestricted	Controls SQL or Objects	No
%Service_CSP	Yes	Yes	Password, Unauthenticated	Unrestricted	Controls CSP Gateway access	No
%Service_CacheDirect	Yes	Yes	Unauthenticated	Unrestricted	Controls Cache Direct	No
%Service_CallIn	Yes	Yes	Unauthenticated	Unrestricted	Controls the Call-In Interface	No
%Service_CommPort	No	Yes	Unauthenticated	Unrestricted	Controls COMM ports attached to a Windows system	No
%Service_Console	Yes	Yes	Unauthenticated	Unrestricted	Controls CTERM (TRM:pid) and the Windows Console	No
%Service_DataCheck	No	N/A		Unrestricted	Controls this system as a DataCheck source	No
%Service_ECP	No	N/A		Unrestricted	Controls Enterprise Cache Protocol (ECP)	No
%Service_Login	Yes	No	Password	Unrestricted	Controls SYSTEM.Security.Login	No
%Service_MSMActivate	No	N/A	Unauthenticated	Unrestricted	Controls MSM Activate Protocol	No
%Service_Mirror	No	N/A			Controls Mirroring	No
%Service_Monitor	No	N/A			Controls SNMP and remote Monitor commands	No
%Service_Shadow	No	N/A		Unrestricted	Controls if this system can be the source of a shadow	No
%Service_Telnet	No	Yes	Unauthenticated	Unrestricted	Controls Telnet sessions on a Windows server	No
%Service_WebLink	No	N/A	Unauthenticated	Unrestricted	Controls Weblink	No

Figure 31 - The list of Services available to Caché

Edit definition for service %Service_Telnet:

Service Name: %Service_Telnet
Description: Controls Telnet sessions on a Windows server
Service Enabled: <input type="checkbox"/>
Allowed Authentication Methods:
<input checked="" type="checkbox"/> Unauthenticated
<input type="checkbox"/> Password
Save Cancel

Figure 32 - Edit Service page for %Service_Telnet.

To enable the Telnet session, simply check the box next to “Service Enabled” and then click “Save” (Figure 33).

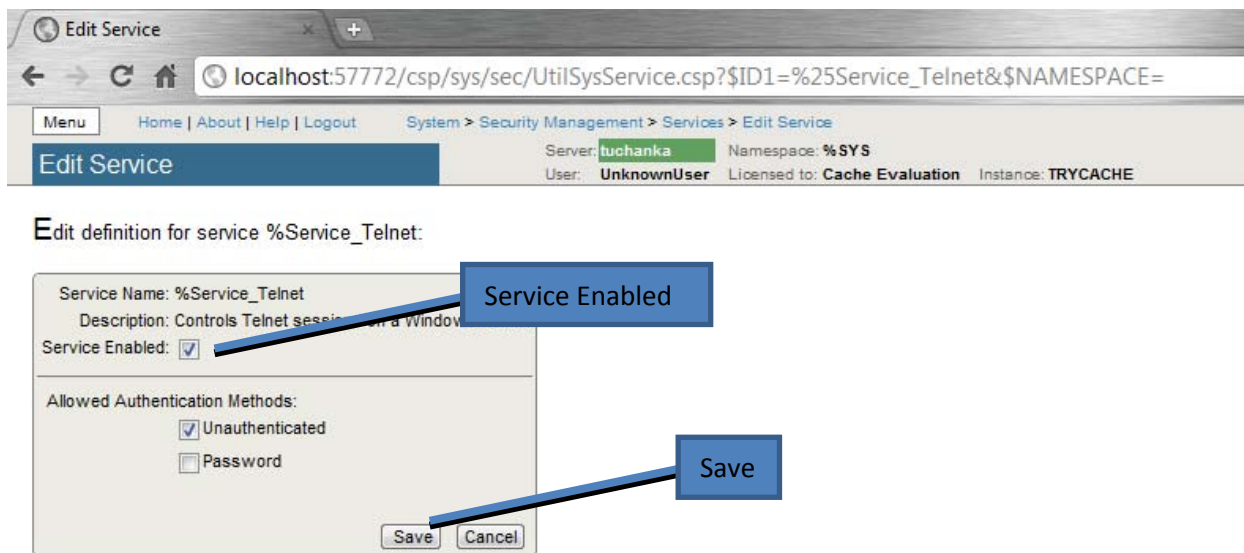


Figure 33 - Enabling the Telenet service.

After saving, the Services menu will now show that the Telnet service is enabled (Figure 34).

Figure 34 shows the 'Services' web interface. The browser address bar displays 'localhost:57772/csp/sys/sec/UtilSysServices.csp'. The page title is 'Services'. The breadcrumb navigation is 'System > Security Management > Services'. The server information shows 'Server: tuchanka', 'Namespace: %SYS', 'User: UnknownUser', 'Licensed to: Cache Evaluation', and 'Instance: TRYCACHE'. Below the header, there is a filter and pagination section: 'Filter: []', 'Page size: 20', and 'Items found: 15'. The main content is a table with the following columns: Name, Enabled, Public, Authentication Methods, Allowed Connections, Description, and Two-Factor Enabled. The table contains 15 rows of service information.

Name	Enabled	Public	Authentication Methods	Allowed Connections	Description	Two-Factor Enabled
%Service Bindings	Yes	N/A	Password, Unauthenticated	Unrestricted	Controls SQL or Objects	No
%Service CSP	Yes	Yes	Password, Unauthenticated	Unrestricted	Controls CSP Gateway access	No
%Service CacheDirect	Yes	Yes	Unauthenticated	Unrestricted	Controls Cache Direct	No
%Service CallIn	Yes	Yes	Unauthenticated	Unrestricted	Controls the Call-In Interface	No
%Service ComPort	No	Yes	Unauthenticated	Unrestricted	Controls COMM ports attached to a Windows system	No
%Service Console	Yes	Yes	Unauthenticated	Unrestricted	Controls CTERM (TRM:pid) and the Windows Console	No
%Service DataCheck	No	N/A		Unrestricted	Controls this system as a DataCheck source	No
%Service ECP	No	N/A		Unrestricted	Controls Enterprise Cache Protocol (ECP)	No
%Service Login	Yes	No	Password	Unrestricted	Controls SYSTEM.Security.Login	No
%Service MSMActivate	No	N/A	Unauthenticated	Unrestricted	Controls MSM Activate Protocol	No
%Service Mirror	No	N/A		Unrestricted	Controls Mirroring	No
%Service Monitor	No	N/A		Unrestricted	Controls SNMP and remote Monitor commands	No
%Service Shadow	No	N/A		Unrestricted	Controls if this system can be the source of a shadow	No
%Service Telnet	Yes	Yes	Unauthenticated	Unrestricted	Controls Telnet sessions on a Windows server	No
%Service Weblink	No	N/A	Unauthenticated	Unrestricted	Controls Weblink	No

Figure 34 - Services menu with Telnet enabled

[Return to the Beginning](#)

Importing the OSEHRA Code Base into Caché

Getting the code from Git and Packing into Caché Format

Starting from an empty Caché, we need to retrieve the routines and globals that will populate the Vista environment from the OSEHRA Code Repository at code.osehra.org. Begin by bringing up a Git Bash terminal from the installed Git system (Figure 35).

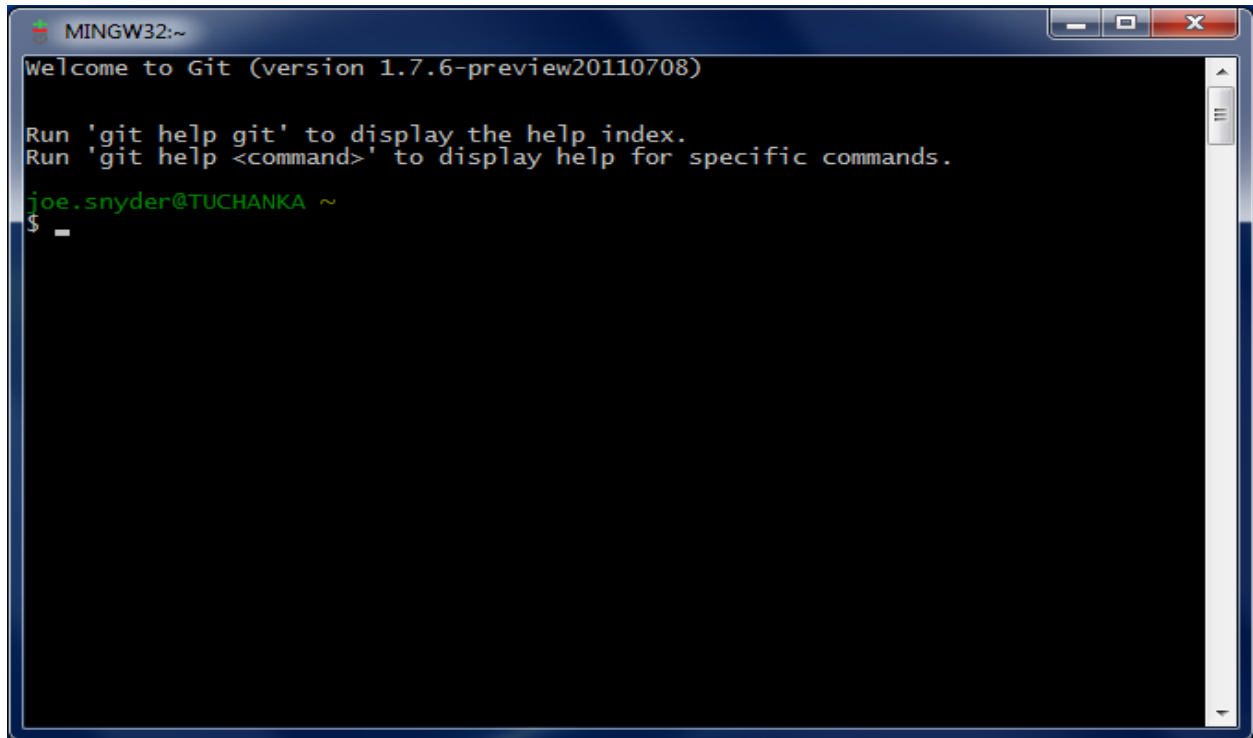
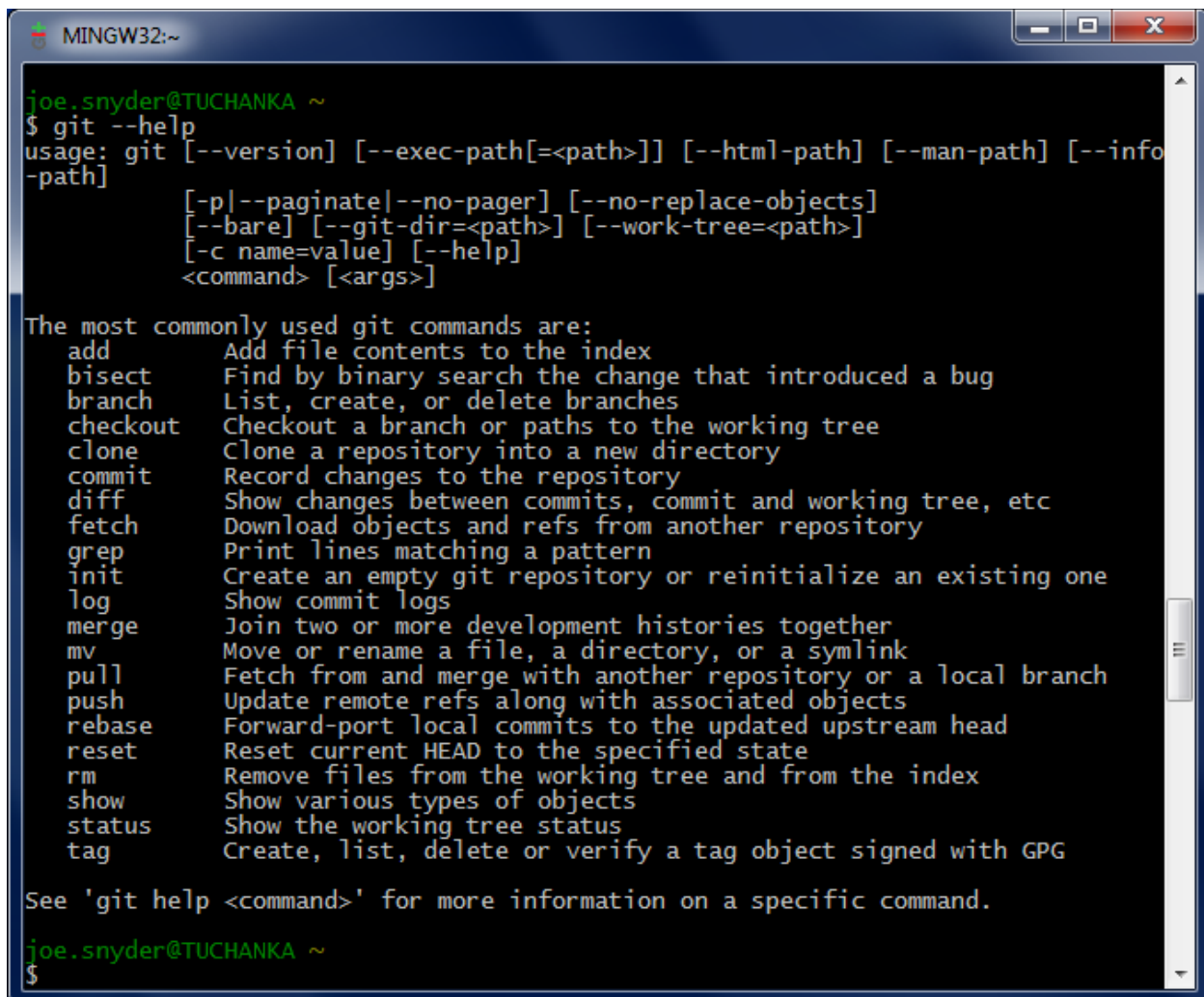


Figure 35 - Startup window for the Git Bash shell

The git interface is a command-prompt interface with much of the functionality that should be familiar to Linux users; commands like "ls" and "cd" perform as they would in Cygwin shell or a Linux terminal. Git specific actions are always preceded by the "git" command on the input line. The command "git --help" displays a list of the most common commands (Figure 36).



```
MINGW32:~
joe.snyder@TUCHANKA ~
$ git --help
usage: git [--version] [--exec-path[=<path>]] [--html-path] [--man-path] [--info-
-path]
        [-p|--paginate|--no-pager] [--no-replace-objects]
        [--bare] [--git-dir=<path>] [--work-tree=<path>]
        [-c name=value] [--help]
        <command> [<args>]

The most commonly used git commands are:
add          Add file contents to the index
bisect       Find by binary search the change that introduced a bug
branch       List, create, or delete branches
checkout     Checkout a branch or paths to the working tree
clone        Clone a repository into a new directory
commit       Record changes to the repository
diff         Show changes between commits, commit and working tree, etc
fetch        Download objects and refs from another repository
grep         Print lines matching a pattern
init         Create an empty git repository or reinitialize an existing one
log          Show commit logs
merge        Join two or more development histories together
mv           Move or rename a file, a directory, or a symlink
pull         Fetch from and merge with another repository or a local branch
push         Update remote refs along with associated objects
rebase       Forward-port local commits to the updated upstream head
reset        Reset current HEAD to the specified state
rm           Remove files from the working tree and from the index
show         Show various types of objects
status       Show the working tree status
tag          Create, list, delete or verify a tag object signed with GPG

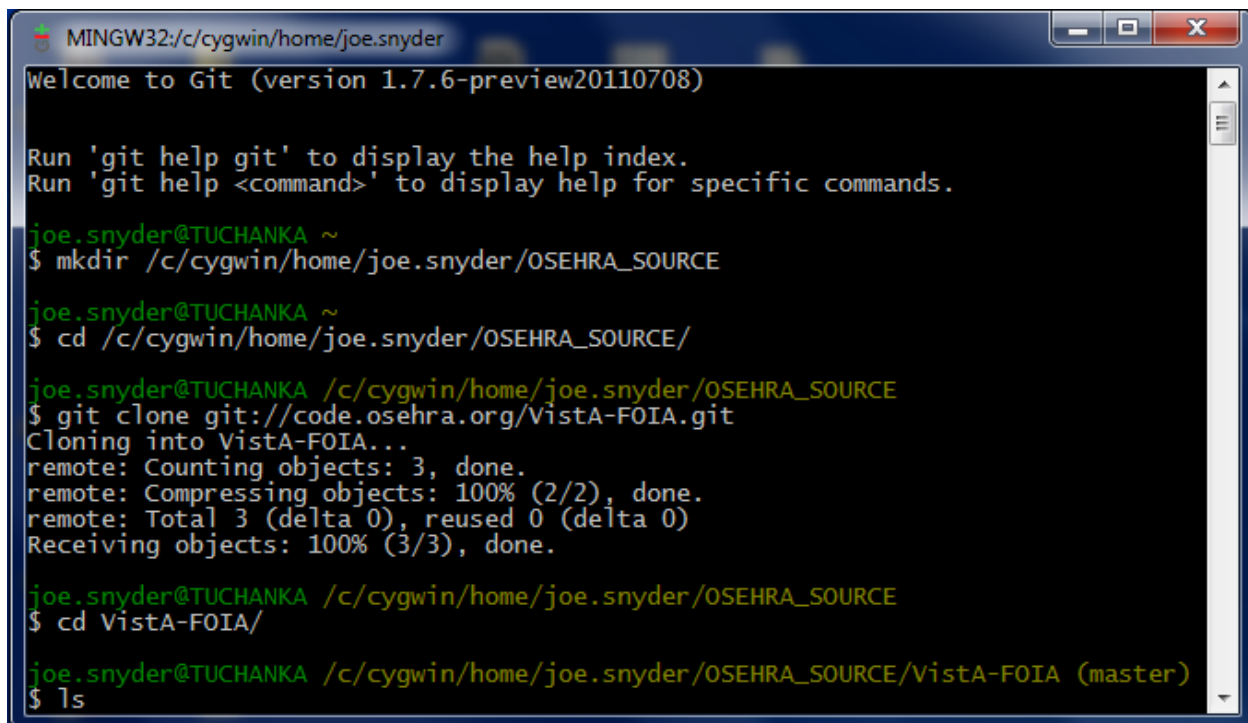
See 'git help <command>' for more information on a specific command.
joe.snyder@TUCHANKA ~
$
```

Figure 36 - Output of the “git --help” command.

To obtain a copy of the repository, create a directory (Folder) to hold the repository code and “cd” into that directory. Enter the commands

```
$ git clone git://code.osehra.org/VistA-FOIA.git
$ cd VistA-FOIA
```

to make a local clone of the remote repository (Figure 37).

A screenshot of a terminal window titled 'MINGW32:/c/cygwin/home/joe.snyder'. The terminal shows the output of 'git clone' and subsequent directory changes. The user is 'joe.snyder@TUCHANKA'. The commands and output are as follows:

```
Welcome to Git (version 1.7.6-preview20110708)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

joe.snyder@TUCHANKA ~
$ mkdir /c/cygwin/home/joe.snyder/OSEHRA_SOURCE

joe.snyder@TUCHANKA ~
$ cd /c/cygwin/home/joe.snyder/OSEHRA_SOURCE/

joe.snyder@TUCHANKA /c/cygwin/home/joe.snyder/OSEHRA_SOURCE
$ git clone git://code.osehra.org/VistA-FOIA.git
Cloning into VistA-FOIA...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.

joe.snyder@TUCHANKA /c/cygwin/home/joe.snyder/OSEHRA_SOURCE
$ cd VistA-FOIA/

joe.snyder@TUCHANKA /c/cygwin/home/joe.snyder/OSEHRA_SOURCE/VistA-FOIA (master)
$ ls
```

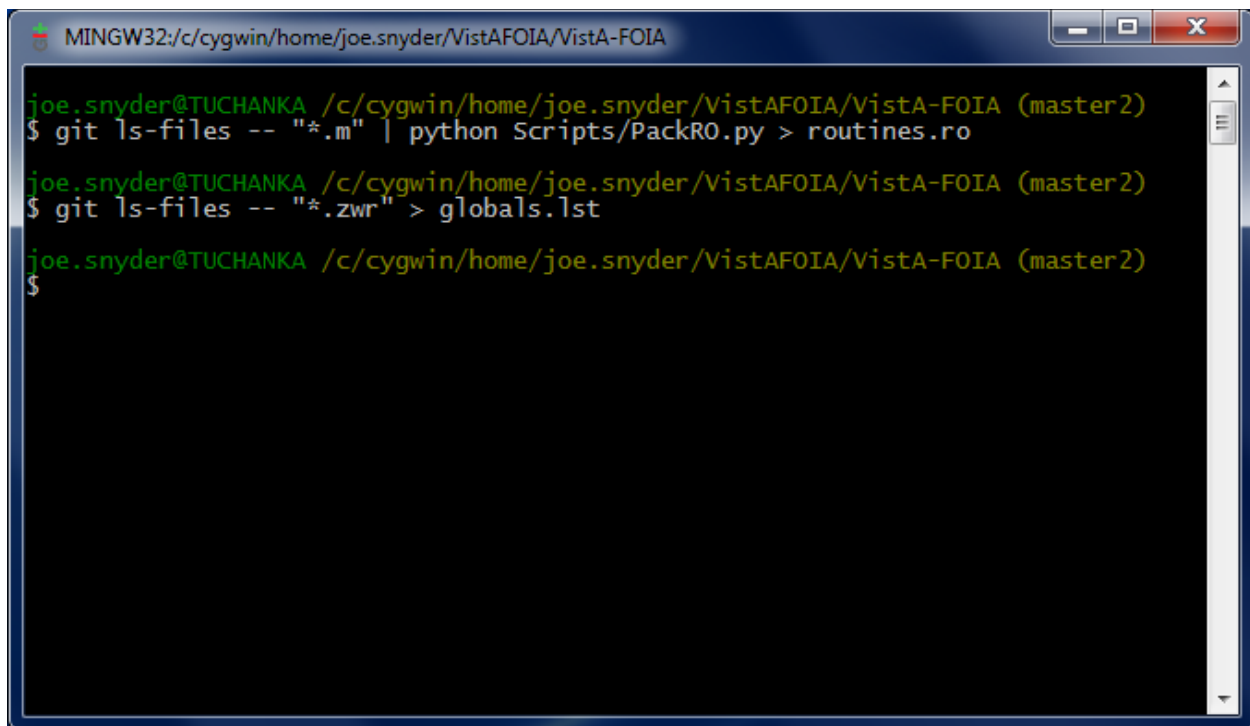
Figure 37 - Cloning the repository into an OSEHRA_SOURCE folder.

Within the Git repository, you can see the folders that contain all that is necessary to prepare the routines and globals to be imported into the Caché instance. The Packages folder contains all of the VistA FOIA software divided by package name. Inside each package directory lies a Routines directory and a Globals directory. The latter contains globals divided up by the FileMan files they contain. The Scripts folder contains OSEHRA python scripts and helper routines. The python scripts are used to pack and unpack the [routines.ro](#) file, while the routines are used to import and export the globals from the MUMPS environment.

Change directory (cd) into the the VistA-FOIA and execute the following two commands to prepare the data for import (Figure 38).

```
$ git ls-files -- "*.m" | python Scripts/PackRO.py > routines.ro
$ git ls-files -- "*.zwr" > globals.lst
```

The first command lists all the files that have the extension '.m' and passes those names to the python script [PackRO.py](#) to pack them into file [routine.ro](#) in routine transfer format. The second command lists all files that have the extension '.zwr' and writes those into [globals.lst](#). This list will be read by the OSEHRA ZGI routine during the import step.



```
joe.snyder@TUCHANKA /c/cygwin/home/joe.snyder/VistAFOIA/VistA-FOIA (master2)
$ git ls-files -- "*.m" | python Scripts/PackRO.py > routines.ro

joe.snyder@TUCHANKA /c/cygwin/home/joe.snyder/VistAFOIA/VistA-FOIA (master2)
$ git ls-files -- "*.zwr" > globals.lst

joe.snyder@TUCHANKA /c/cygwin/home/joe.snyder/VistAFOIA/VistA-FOIA (master2)
$
```

Figure 38 - Packing the routines and globals into the forms needed for import

Import routines and globals into Caché

The next set of steps is to import the routines and globals into the Caché instance and then configure the instance to the point where testing can be performed. All of these steps are performed from the Caché terminal. Right click on the Caché icon and select terminal from the pop-up as shown (Figure 39) to bring up a Caché terminal (Figure 40).

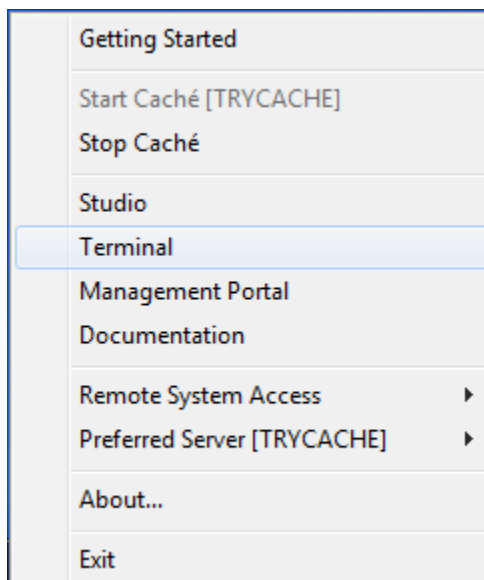


Figure 39 - Caché menu with the Terminal option highlighted.

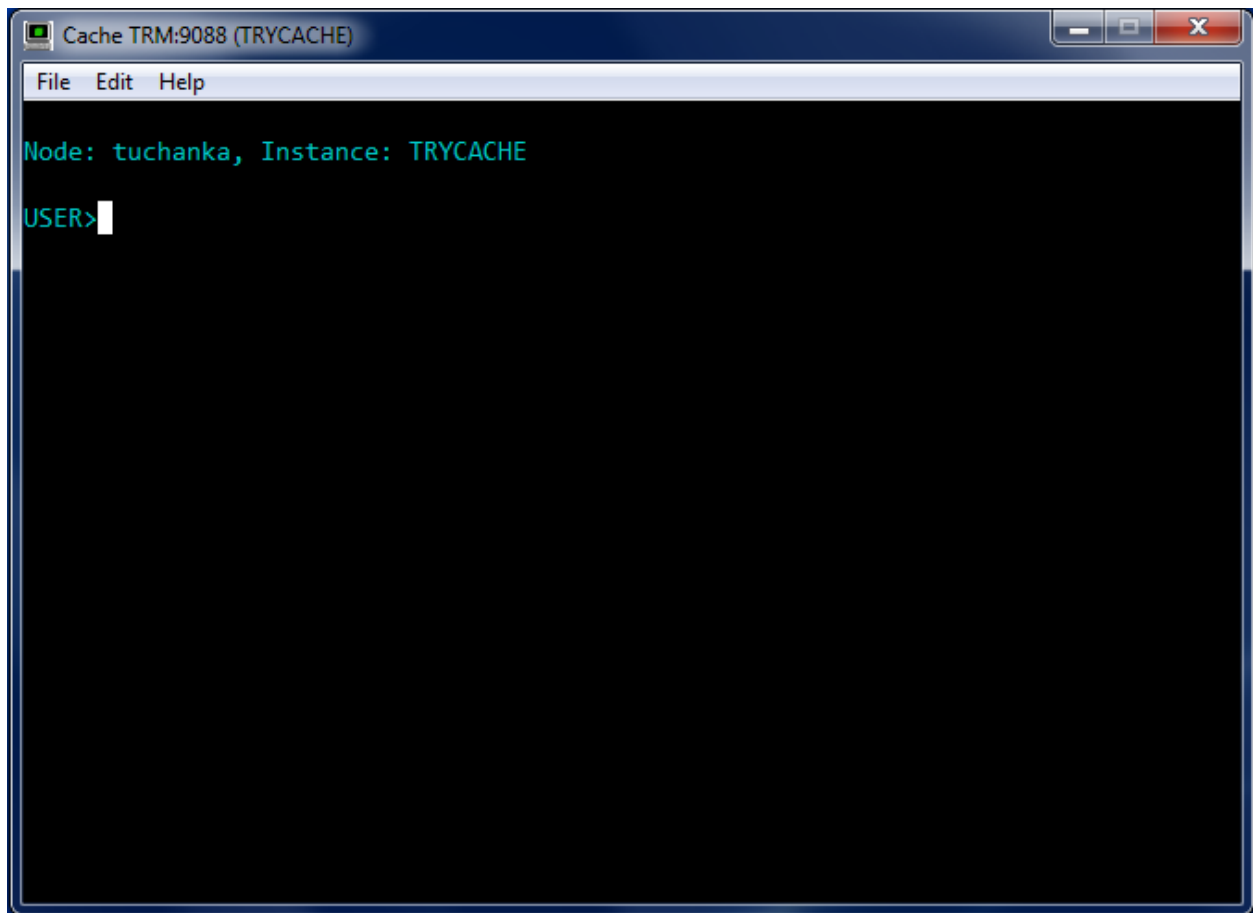


Figure 40 - Startup screen of the Caché Terminal.

The first step is to change from the default "USER" namespace to the namespace that you created earlier. Entering "D ^%CD" will bring up a prompt to accept the name of the new namespace that you want to enter. For demonstration purposes, we will assume that the namespace is called "VISTA" (Figure 41).

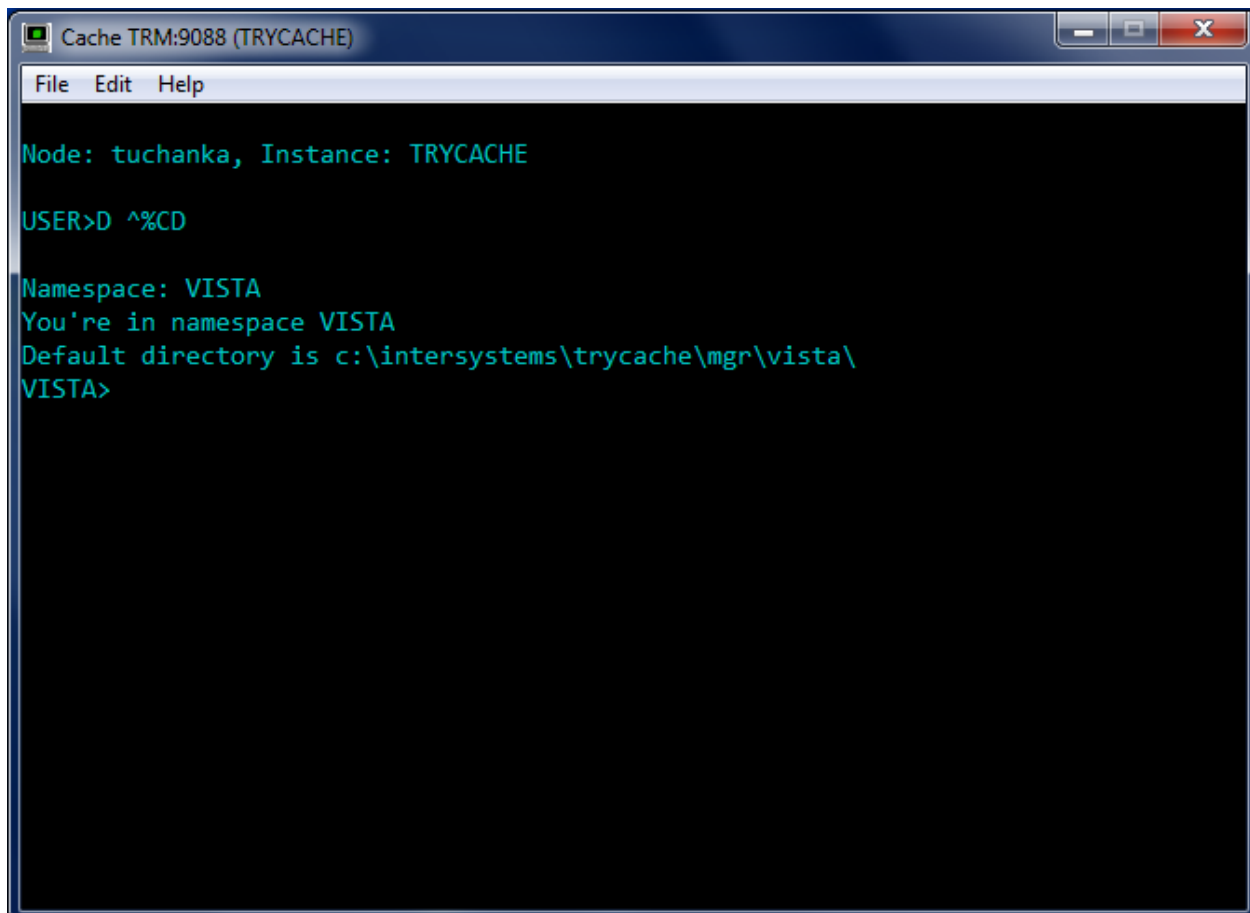


Figure 41 - Changing namespace to the created VISTA.

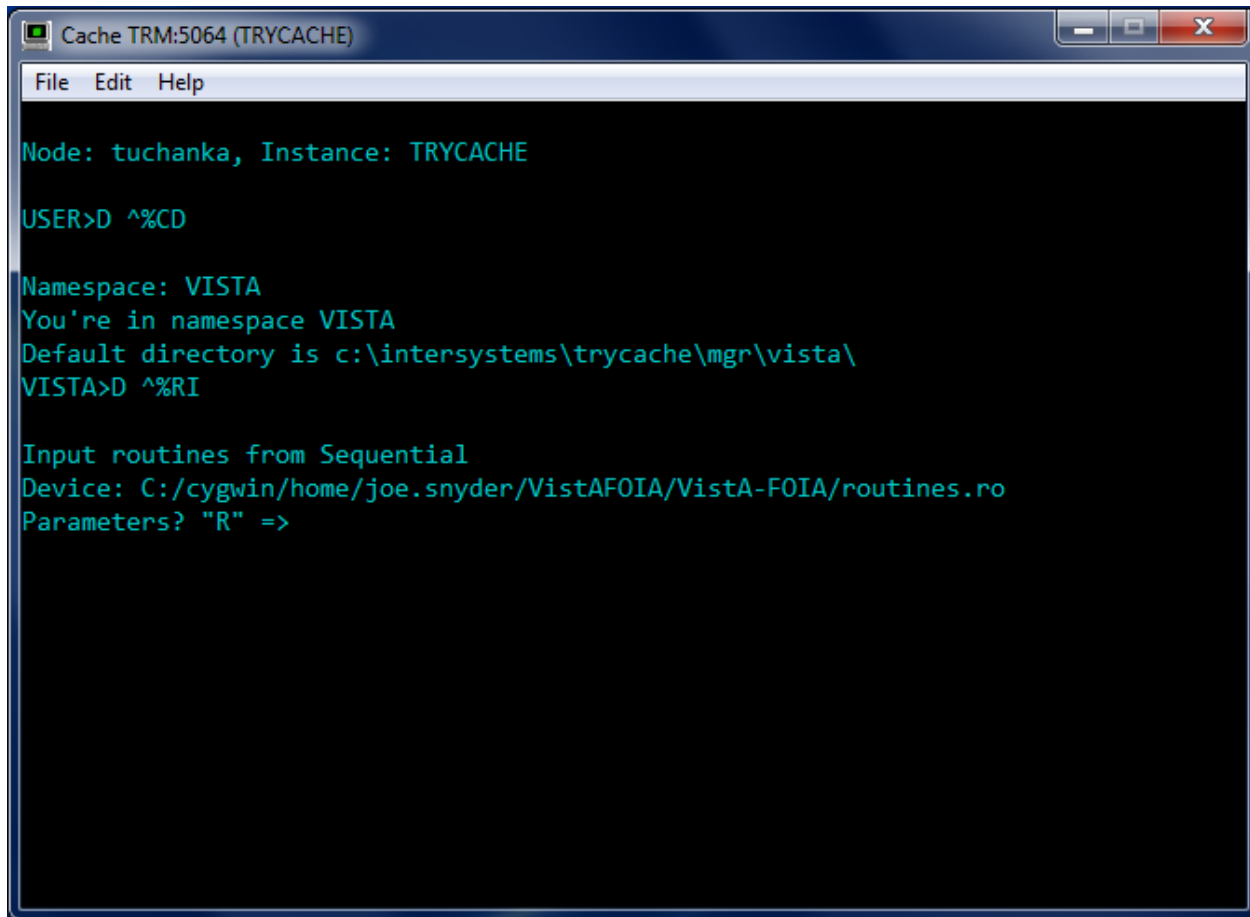
The %RI routine is a MUMPS routine that is used to import other routines from a specific file type. At the VISTA> prompt, enter the command:

```
VISTA> D ^%RI
```

This starts the routine import routine. Enter the path to the routines.ro file (Figure 38) as the device and <Enter> to accept the default, read-only mode. The terminal will then display a warning saying "This file may not be a %RO output file." This is an expected warning due to how we are creating the [routines.ro](#) file. At the prompt, enter "Yes" to continue and then enter then <Enter> to accept the 0th option (Caché) for the routines write type. To fully import the routines, enter "All Routines" at the Routine Input Option, and answer "YES" to "replace similarly named routines." All other options may be answered with <Enter> to accept the default value (Figure 42, Figure 43, and Figure 44).

The system will respond with a list of all that have been processed and then displays the VistA prompt (Figure 45). In the listing, the symbols next to the routine name have meaning:

- ^ indicates routines which will replace those now on file.
- @ indicates routines which have been [re]compiled.
- indicates routines which have not been filed.



```
Cache TRM:5064 (TRYCACHE)
File Edit Help

Node: tuchanka, Instance: TRYCACHE

USER>D ^%CD

Namespace: VISTA
You're in namespace VISTA
Default directory is c:\intersystems\trycache\mgr\vista\
VISTA>D ^%RI

Input routines from Sequential
Device: C:/cygwin/home/joe.snyder/VistAFOIA/VistA-FOIA/routines.ro
Parameters? "R" =>
```

Figure 42 - First parameters entered into the %RI routine: How to interact with the routines.ro file.

```
Parameters? "R" =>

***** W A R N I N G *****

File Header: Routines
Date Stamp:

This file may not be a %RO output file.
Override and use this File with %RI? No => Yes
%RI has detected a routine written with UNKNOWN mode.
0) Cache
1) DSM-11
2) DTM
3) Ipsum
4) Cobra
5) DSM-VMS
6) DSM-J
7) DTM-J
8) MSM
9) BASIC
10) U2/M
11) MVBASIC

Please enter a number from the above list: <0>

File written by OLD with description:
Routines

( All Select Enter List Quit )

Routine Input Option:
```

Figure 43 - Second parameters set in %RI routine: Choosing a mode to read the routines.

```
Routine Input Option: All Routines

If a selected routine has the same name as one already on file,
shall it replace the one on file? No => Yes
Recompile? Yes => Yes
Display Syntax Errors? Yes => Yes
```

Figure 44 - Third parameters set in %RI routine: Options on the import of routines.

```
WWALERTF.INT@  WWALERTP.INT@  WWALERTR.INT@  WWALERTS.INT@  WWRDUP.INT@
WWRNED.INT@    WWRNED1.INT@  WWRNEDH.INT@  WWRNOT.INT@    WWRNOT1.INT@
WWRNOT2.INT@   WWRPCD.INT@    WWRPCD1.INT@  WWRPCD2.INT@   WWRPCD3.INT@
WVCMGR.INT@    WVDIAG.INT@    WVDIAGS.INT@  WVENV.INT@     WEXPTA.INT@
WVFACE.INT@    WVFMAN.INT@    WGETAL1.INT@  WGETALL.INT@   WHS.INT@
WVLAB.INT@     WVLABAD1.INT@  WVLABADD.INT@ WLABCHK.INT@   WLABLG.INT@
WVLABLG1.INT@  WVLABLG2.INT@  WLABWP.INT@   WLABWPC.INT@   WLABWPS.INT@
WVLETDQ.INT@   WVLETPR.INT@   WVLOGO.INT@   WLRLINK.INT@   WMRGP.INT@
WVMTL.INT@     WVMTL1.INT@    WNOTIF.INT@   WNOTIF1.INT@   WPATE.INT@
WVPATP.INT@    WPRE.INT@      WPROC.INT@    WPROC1.INT@    WPROF.INT@
WVPROF1.INT@   WVPROF2.INT@   WVPROF3.INT@  WPRPCD.INT@    WPURP.INT@
WVRAD.INT@     WVRADWP.INT@   WRALIN1.INT@  WRALINK.INT@   WREFUSE.INT@
WVRPCNO.INT@   WVRPCNO1.INT@  WRPCPR.INT@   WRPCPR1.INT@   WRPCPT.INT@
WVRPPCD.INT@   WVRPPCD1.INT@  WVRPPCD2.INT@ WVRPPCD3.INT@  WRPSCR.INT@
WVRPSR1.INT@   WVRPSR2.INT@   WRPSPNP.INT@  WRPSPNP1.INT@  WRPSPNPR.INT@
WVRPST.INT@    WSELECT.INT@   WSITE.INT@    WSNOMED.INT@   WUTL1.INT@
WUTL10.INT@    WUTL1A.INT@    WUTL2.INT@    WUTL3.INT@     WUTL4.INT@
WUTL5.INT@     WUTL6.INT@     WUTL7.INT@    WUTL8.INT@     WUTL9.INT@
WYNOTP.INT@    WIIACT4.INT@   WIIADT1.INT@  WIELG.INT@     WIIGATD.INT@
WIILM.INT@     WIILM01.INT@   WIILM02.INT@  WIILM03.INT@   WIILM04.INT@
WIISERV.INT@   ZGI.INT@

26037 routines processed.
VISTA>
```

Figure 45 - End of the output of the %RI routine.

The next step is to import the Globals using the newly imported ZGI routine. Enter

```
VISTA> D LIST^ZGI("c:\path\to\VistA-FOIA\globals.lst","c:\path\to\VistA-FOIA\")
```

where “path\to” is replaced with the path to the code download (Figure 37). This routine will go through all of the globals contained in the list file and import them into the VistA instance (Figure 46). The last package to be imported is the *Wounded Injured and Ill Warriors* (Figure 47).

```

VISTA>D LIST^ZGI("C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\globals.lst", "C
:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA/")
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\PRCA.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\PRCAK.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\RC.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\RCD.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\RCPS.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\RCPSE.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\RCPSS.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\RCT.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\RCXV.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Accounts_Receivable\Glob
als\RCY.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Asists\Globals\OOPS.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Authorization_Subscripti
on\Globals\USR.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Auto_Replenishment_Ward
Stock\Globals\PSI.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Automated_Information_Co
llection_System\Globals\IBD.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Automated_Lab_Instrument
s\Globals\LAB.zwr

```

Figure 46 - Initial command and first lines of output of ZGI routine.

```

C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\VA_FileMan\Globals\DISV.
zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\VA_FileMan\Globals\DIZ.z
wr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\VA_FileMan\Globals\DMSQ.
zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\VBECs\Globals\VBEC.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\VDEF\Globals\VDEFHL7.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Visual_Impairment_Servic
e_Team\Globals\ANRV.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Voluntary_Timekeeping\Gl
obals\ABS.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Womens_Health\Globals\wV
.zwr
C:\cygwin\home\joe.snyder\VistAFOIA\Vista-FOIA\Packages\Wounded_Injured_and_Ill
Warriors\Globals\WII.zwr
VISTA>

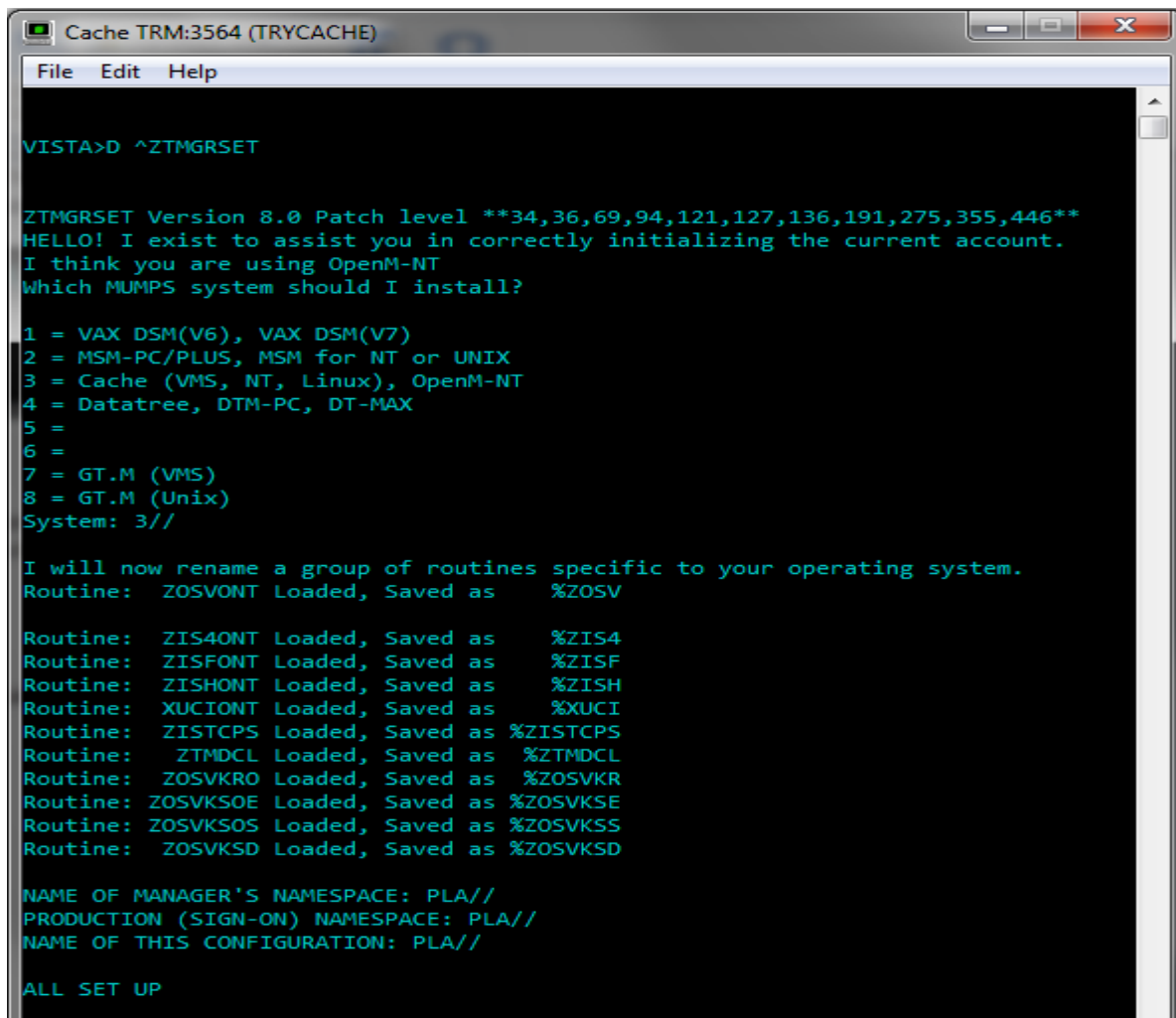
```

Figure 47 - Final lines of output from the ZGI routine.

At this point, all routines and globals are imported and the environment is ready to be configured. Enter:

```
VISTA> D ^ZTMGRSET
```

to initialize the current instance for use. Choose the default, Caché environment. Some routines are loaded, and then accept the default (PLA) for the next three prompts (Figure 48). At the fourth prompt, "Want to rename the FileMan routines," enter "Y" to override the default (Figure 49).



```
Cache TRM:3564 (TRYCACHE)
File Edit Help

VISTA>D ^ZTMGRSET

ZTMGRSET Version 8.0 Patch level **34,36,69,94,121,127,136,191,275,355,446**
HELLO! I exist to assist you in correctly initializing the current account.
I think you are using OpenM-NT
Which MUMPS system should I install?

1 = VAX DSM(V6), VAX DSM(V7)
2 = MSM-PC/PLUS, MSM for NT or UNIX
3 = Cache (VMS, NT, Linux), OpenM-NT
4 = Datatree, DTM-PC, DT-MAX
5 =
6 =
7 = GT.M (VMS)
8 = GT.M (Unix)
System: 3//

I will now rename a group of routines specific to your operating system.
Routine: ZOSVONT Loaded, Saved as %ZOSV

Routine: ZIS4ONT Loaded, Saved as %ZIS4
Routine: ZISFONT Loaded, Saved as %ZISF
Routine: ZISHONT Loaded, Saved as %ZISH
Routine: XUCIONT Loaded, Saved as %XUCI
Routine: ZISTCPS Loaded, Saved as %ZISTCPS
Routine: ZTMDCL Loaded, Saved as %ZTMDCL
Routine: ZOSVKRO Loaded, Saved as %ZOSVKR
Routine: ZOSVKSOE Loaded, Saved as %ZOSVKSE
Routine: ZOSVKSOS Loaded, Saved as %ZOSVKSS
Routine: ZOSVKSD Loaded, Saved as %ZOSVKSD

NAME OF MANAGER'S NAMESPACE: PLA//
PRODUCTION (SIGN-ON) NAMESPACE: PLA//
NAME OF THIS CONFIGURATION: PLA//

ALL SET UP
```

Figure 48 - Setting the MUMPS environment and accepting the default namespaces.


```
ALL SET UP

Now to load routines common to all systems.
Routine:  ZTLOAD Loaded, Saved as  %ZTLOAD
Routine:  ZTLOAD1 Loaded, Saved as %ZTLOAD1
Routine:  ZTLOAD2 Loaded, Saved as %ZTLOAD2
Routine:  ZTLOAD3 Loaded, Saved as %ZTLOAD3
Routine:  ZTLOAD4 Loaded, Saved as %ZTLOAD4
Routine:  ZTLOAD5 Loaded, Saved as %ZTLOAD5
Routine:  ZTLOAD6 Loaded, Saved as %ZTLOAD6
Routine:  ZTLOAD7 Loaded, Saved as %ZTLOAD7
Routine:   ZTM Loaded, Saved as    %ZTM
Routine:  ZTM0 Loaded, Saved as   %ZTM0
Routine:  ZTM1 Loaded, Saved as   %ZTM1
Routine:  ZTM2 Loaded, Saved as   %ZTM2
Routine:  ZTM3 Loaded, Saved as   %ZTM3
Routine:  ZTM4 Loaded, Saved as   %ZTM4
Routine:  ZTM5 Loaded, Saved as   %ZTM5
Routine:  ZTPTCH Loaded, Saved as %ZTPTCH
Routine:  ZTRDEL Loaded, Saved as %ZTRDEL
Routine:  ZTMOVE Loaded, Saved as %ZTMOVE
Want to rename the FileMan routines: No//Y
Routine:   DIDT Loaded, Saved as   %DT
Routine:  DIDTC Loaded, Saved as   %DTC
Routine:  DIRCR Loaded, Saved as   %RCR
Installing ^%Z editor
Setting ^%ZIS('C')

Now, I will check your % globals.....

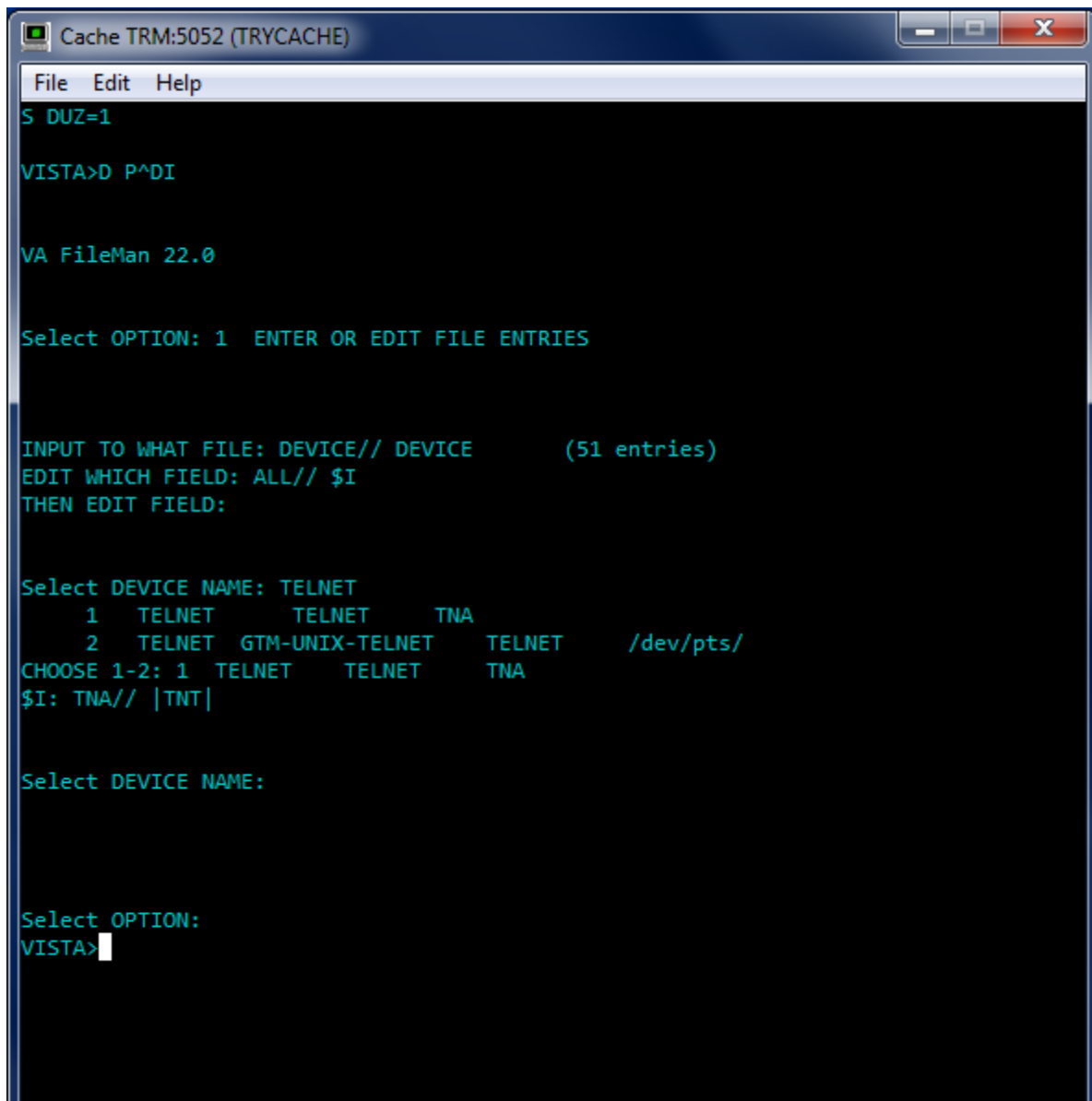
ALL DONE
VISTA>
```

Figure 49 - Loading more routines after accepting the namespace defaults and renaming FileMan routines.

The final step needed for the testing is to alter a device within the File Manager. We need to change the \$I value of the TELNET device to let the Caché terminal function as a display for the XINDEX routine. The first step is to identify yourself as a programmer and gain permissions to change the files attributes. Enter:

```
VISTA> S DUZ=1
VISTA> D P^DI
```

to first get access to the File Manager and then to start the File Manager. At the *Select OPTION* prompt, enter "1" to edit the file entries; at the *INPUT TO WHAT FILE:* prompt, enter the word "DEVICE"; and at the *EDIT WHICH FIELD:* prompt enter "\$I". Enter <Enter> to end the field queries. The system will respond with a *Select DEVICE NAME:* prompt, enter "TELNET" to bring up an option menu and then enter the option that does not reference GT.M or UNIX. Finally, the system will respond with \$I: TNA//. Enter |TNT|, and press enter until the VISTA prompt is reached (Figure 50).



```
Cache TRM:5052 (TRYCACHE)
File Edit Help
S DUZ=1

VISTA>D P^DI

VA FileMan 22.0

Select OPTION: 1 ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: DEVICE// DEVICE (51 entries)
EDIT WHICH FIELD: ALL// $I
THEN EDIT FIELD:

Select DEVICE NAME: TELNET
  1 TELNET TELNET TNA
  2 TELNET GTM-UNIX-TELNET TELNET /dev/pts/
CHOOSE 1-2: 1 TELNET TELNET TNA
$I: TNA// |TNT|

Select DEVICE NAME:

Select OPTION:
VISTA>
```

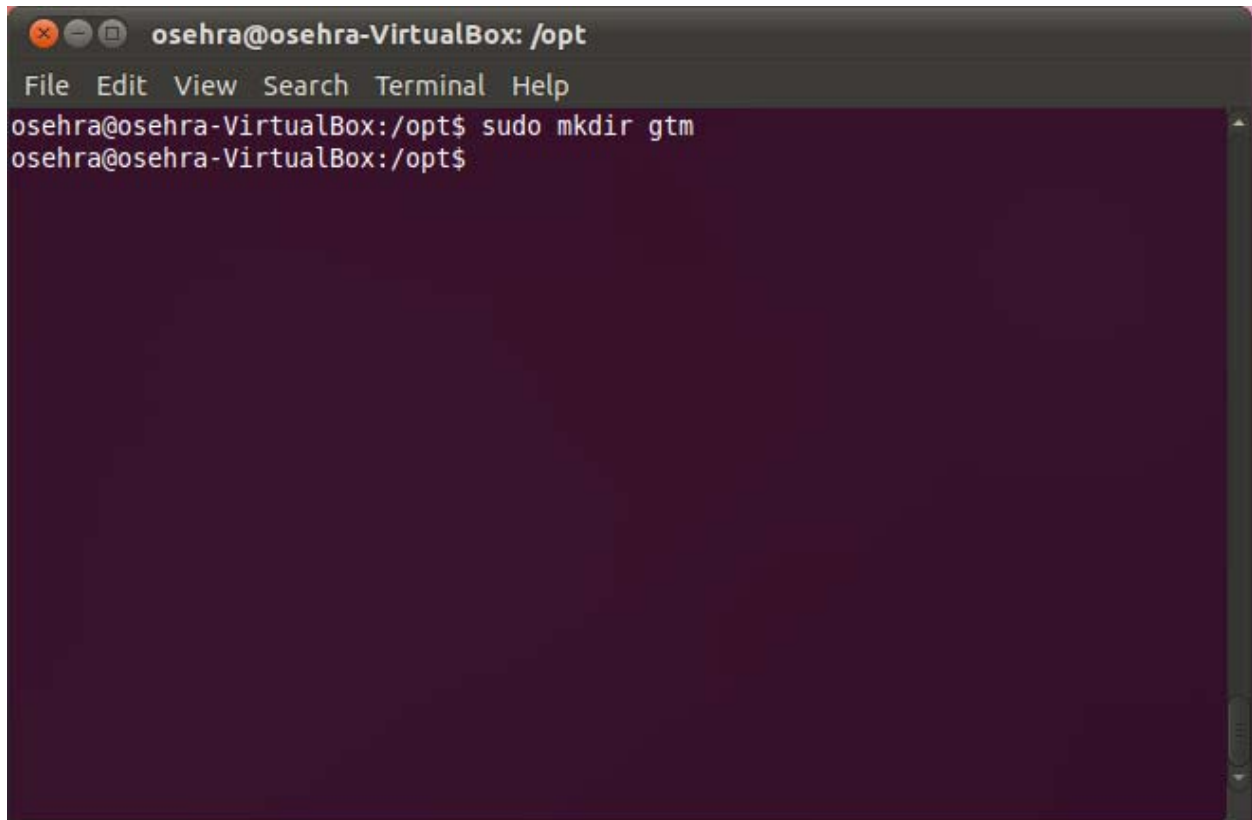
Figure 50 - Using the File Manager to change the \$I value of the TELNET device.

[Return to the Beginning](#)

Installing GT.M

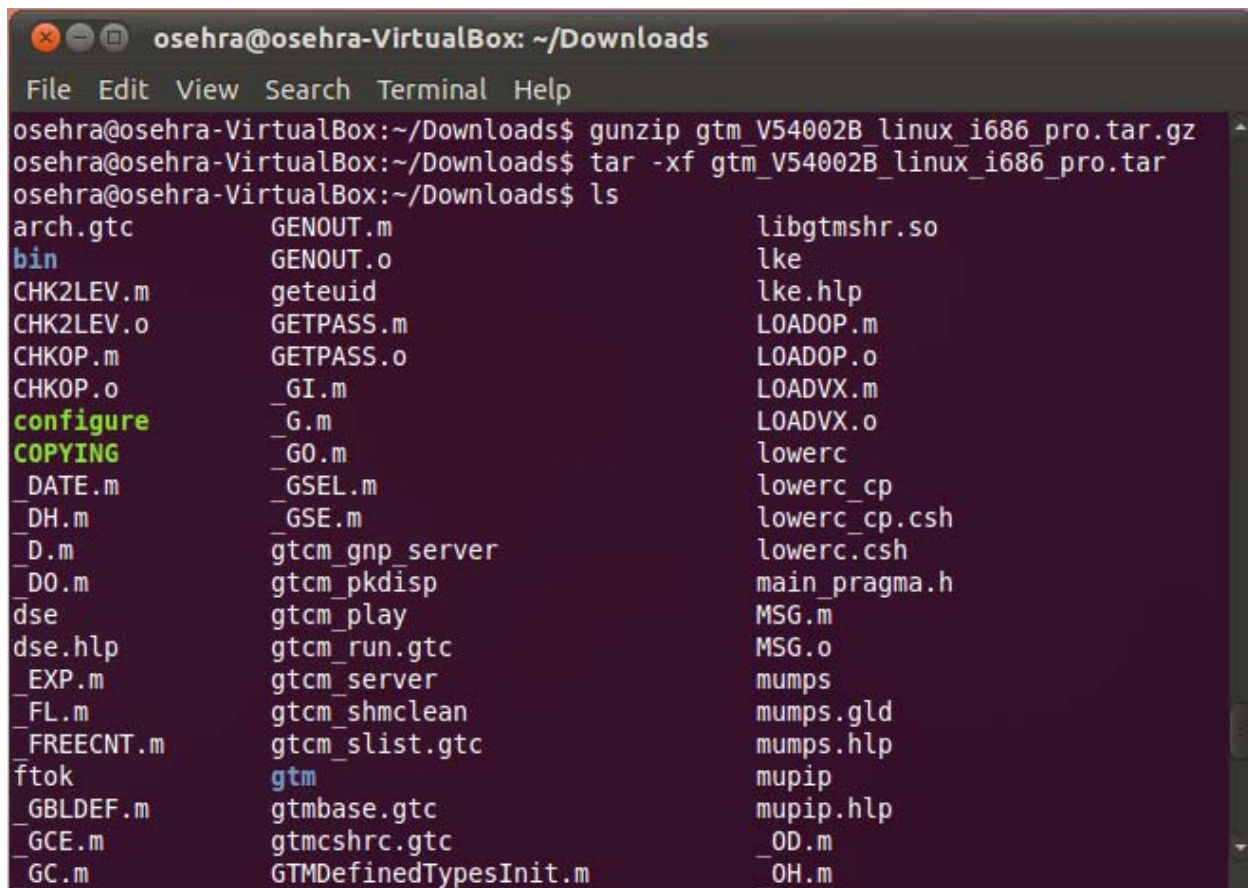
The GT.M system is available for download from SourceForge at <http://sourceforge.net/projects/fis-gtm/>. This will download a compressed package which will need to be extracted before proceeding. The extraction can be in a location of your choice, as during the installation process you will be able to set where the installed files will be placed.

To have a common place where the GT.M files will be placed, we first create a folder within the /opt/ directory which will hold the files. The command used to create the folder is shown in Figure 51 and a snapshot of the files that are extracted is shown in Figure 52.

A terminal window titled "osehra@osehra-VirtualBox: /opt" with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command "sudo mkdir gtm" being entered and executed, with the prompt changing from "osehra@osehra-VirtualBox:/opt\$" to "osehra@osehra-VirtualBox:/opt\$".

```
osehra@osehra-VirtualBox: /opt
File Edit View Search Terminal Help
osehra@osehra-VirtualBox:/opt$ sudo mkdir gtm
osehra@osehra-VirtualBox:/opt$
```

Figure 51 - Making the GT.M folder within the /opt/ directory.

A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
osehra@osehra-VirtualBox:~/Downloads$ gunzip gtm_V54002B_linux_i686_pro.tar.gz
osehra@osehra-VirtualBox:~/Downloads$ tar -xf gtm_V54002B_linux_i686_pro.tar
osehra@osehra-VirtualBox:~/Downloads$ ls
```

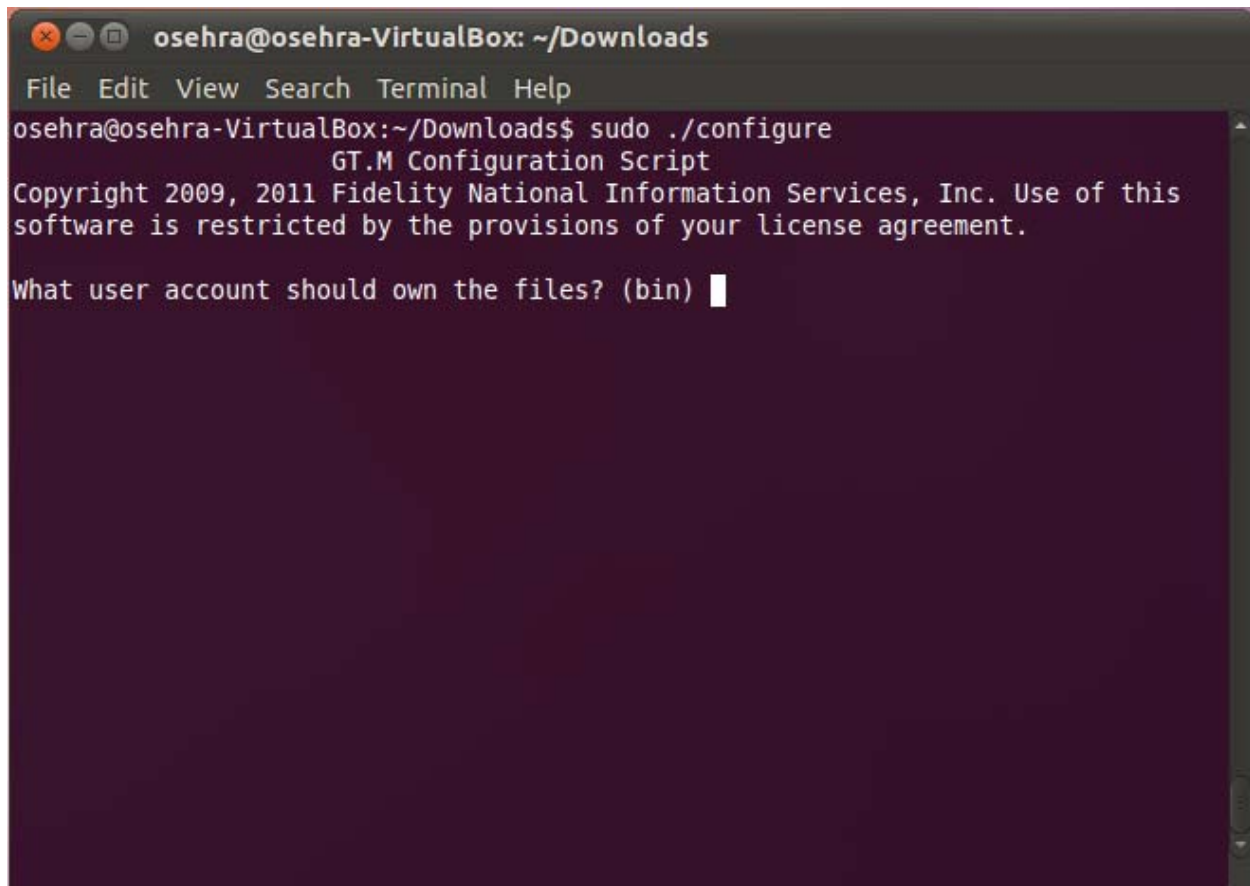
arch.gtc	GENOUT.m	libgtmshr.so
bin	GENOUT.o	lke
CHK2LEV.m	geteuid	lke.hlp
CHK2LEV.o	GETPASS.m	LOADOP.m
CHKOP.m	GETPASS.o	LOADOP.o
CHKOP.o	_GI.m	LOADVX.m
configure	_G.m	LOADVX.o
COPYING	_GO.m	lowerc
_DATE.m	_GSEL.m	lowerc_cp
_DH.m	_GSE.m	lowerc_cp.csh
_D.m	gtcm_gnp_server	lowerc.csh
_DO.m	gtcm_pkdisp	main_pragma.h
dse	gtcm_play	MSG.m
dse.hlp	gtcm_run.gtc	MSG.o
_EXP.m	gtcm_server	mumps
_FL.m	gtcm_shmclean	mumps.gld
_FREECNT.m	gtcm_slist.gtc	mumps.hlp
ftok	gtm	mupip
_GBLDEF.m	gtmbase.gtc	mupip.hlp
_GCE.m	gtmcshrc.gtc	_OD.m
_GC.m	GTMDefinedTypesInit.m	_OH.m

Figure 52 – Extracting the GT.M files from the tar/zip.

To install GT.M, run the [configure.sh](#) file from what was just extracted. Since there will be changes made to the system, this step must be run as root or using sudo privileges.

```
sudo ./configure
```

This will start an interactive series of questions which will set up the GT.M installation for the system. Figure 53 shows the start of the configuration script and the first prompt for the user to enter information.

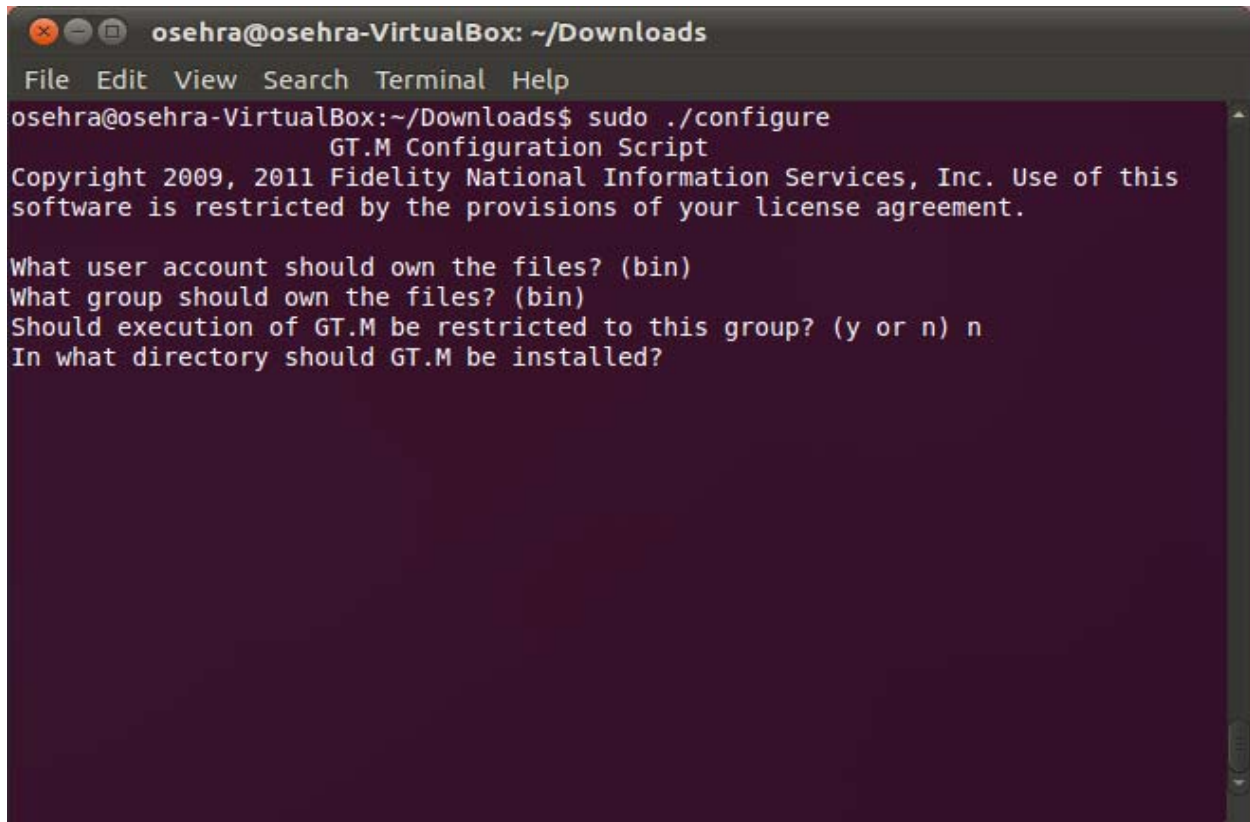
A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal content shows the command 'sudo ./configure' being executed. The output includes 'GT.M Configuration Script', a copyright notice for Fidelity National Information Services, Inc., and a prompt asking for the user account that should own the files. The prompt is '(bin)' followed by a cursor.

```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
osehra@osehra-VirtualBox:~/Downloads$ sudo ./configure
GT.M Configuration Script
Copyright 2009, 2011 Fidelity National Information Services, Inc. Use of this
software is restricted by the provisions of your license agreement.

What user account should own the files? (bin) █
```

Figure 53 – Starting the configuration of GT.M installation.

For the first two options of the configuration, the account and group that would own the GT.M files, accept the default values. At the prompt that asks “Should execution of GT.M be restricted to this group?” enter *n*, which will let all users access the GT.M environment. See Figure 54 for the entries that should be set at this point in the configuration script.

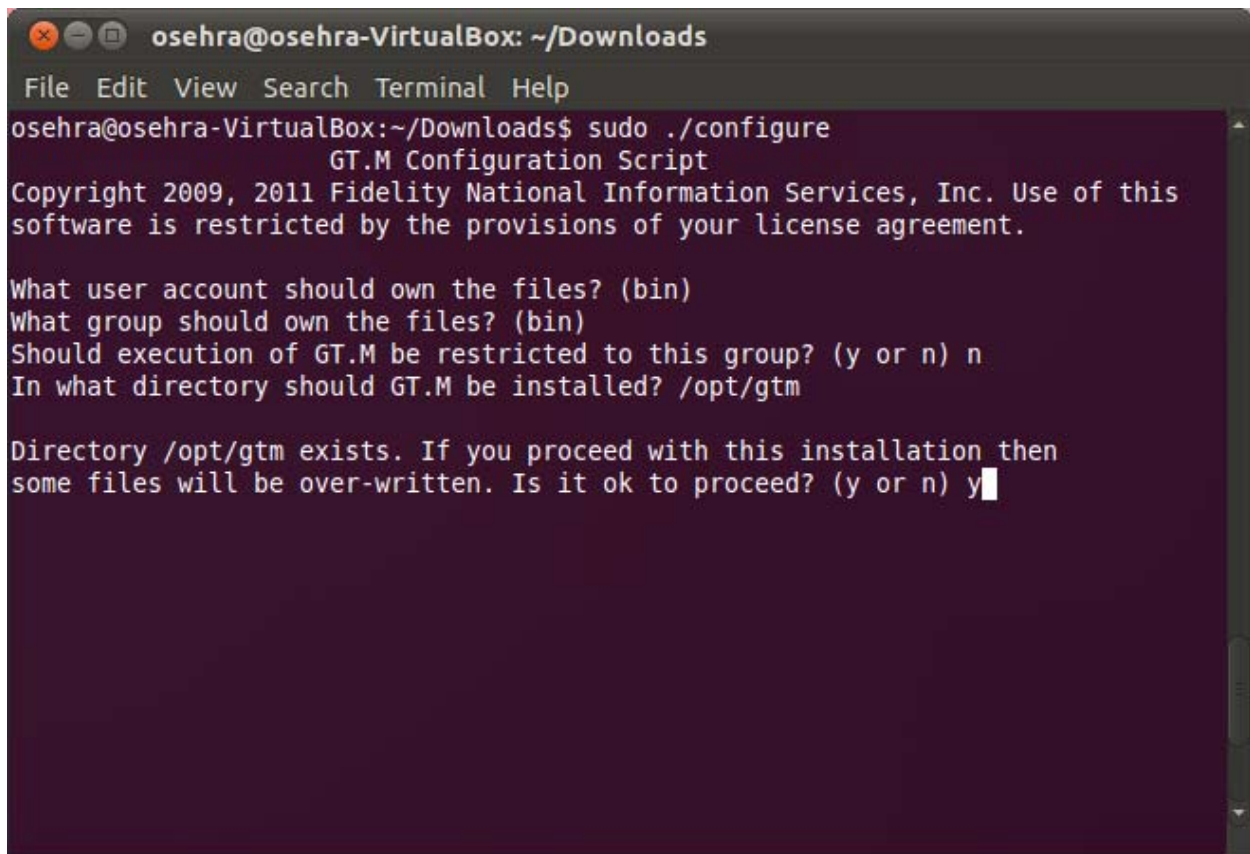
A screenshot of a terminal window titled "osehra@osehra-VirtualBox: ~/Downloads". The terminal shows the execution of the command "sudo ./configure" which runs the "GT.M Configuration Script". The script displays copyright information for Fidelity National Information Services, Inc. and asks four configuration questions: "What user account should own the files? (bin)", "What group should own the files? (bin)", "Should execution of GT.M be restricted to this group? (y or n) n", and "In what directory should GT.M be installed?". The user has entered 'n' for the third question. The terminal background is dark purple.

```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
osehra@osehra-VirtualBox:~/Downloads$ sudo ./configure
GT.M Configuration Script
Copyright 2009, 2011 Fidelity National Information Services, Inc. Use of this
software is restricted by the provisions of your license agreement.

What user account should own the files? (bin)
What group should own the files? (bin)
Should execution of GT.M be restricted to this group? (y or n) n
In what directory should GT.M be installed?
```

Figure 54 – Setting permissions of the installed GT.M files.

The next prompt asks “In what directory should GT.M be installed?”. Here enter a directory that will be used to hold all of the GT.M files, these instructions will put them into /opt/gtm . If the folder that you specify does not exist, you will get a message asking if you want to create it as part of the installation, you should answer yes. If you give it a path to an existing folder, it will warn you that some files may be overwritten during the installation process. Be sure to back up important files in the case that something is lost (Figure 55).

A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the execution of 'sudo ./configure' for the GT.M Configuration Script. It displays copyright information and asks for user, group, and directory settings. The user has entered 'n' for group restriction and 'y' for proceeding with installation to /opt/gtm.

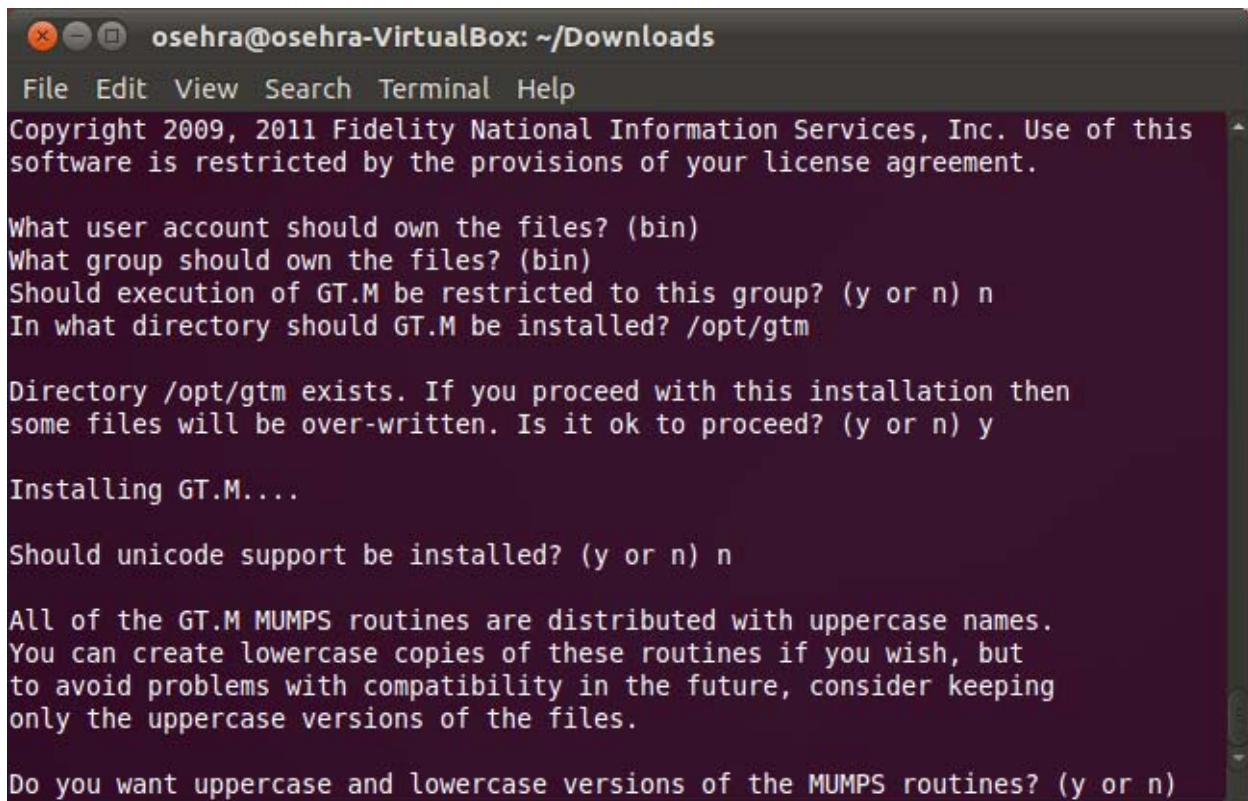
```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
osehra@osehra-VirtualBox:~/Downloads$ sudo ./configure
GT.M Configuration Script
Copyright 2009, 2011 Fidelity National Information Services, Inc. Use of this
software is restricted by the provisions of your license agreement.

What user account should own the files? (bin)
What group should own the files? (bin)
Should execution of GT.M be restricted to this group? (y or n) n
In what directory should GT.M be installed? /opt/gtm

Directory /opt/gtm exists. If you proceed with this installation then
some files will be over-written. Is it ok to proceed? (y or n) y
```

Figure 55 – Setting the installation directory

The next prompt asks the user if they want to install Unicode support (Figure 75). If you feel it will be necessary, it can be installed, but we will not do it in these instructions. Enter your answer to proceed.

A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal displays the following text:

```
Copyright 2009, 2011 Fidelity National Information Services, Inc. Use of this
software is restricted by the provisions of your license agreement.

What user account should own the files? (bin)
What group should own the files? (bin)
Should execution of GT.M be restricted to this group? (y or n) n
In what directory should GT.M be installed? /opt/gtm

Directory /opt/gtm exists. If you proceed with this installation then
some files will be over-written. Is it ok to proceed? (y or n) y

Installing GT.M....

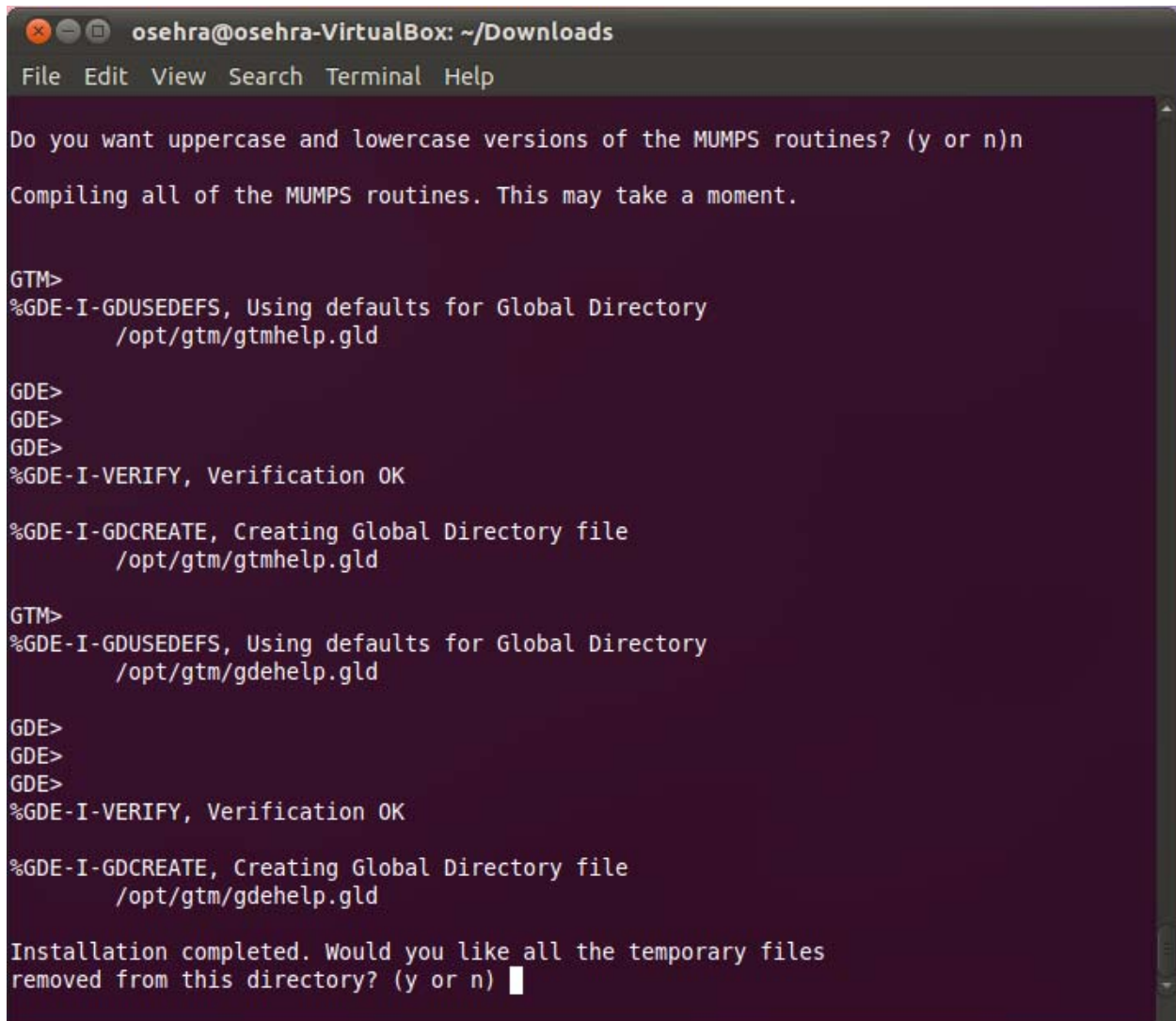
Should unicode support be installed? (y or n) n

All of the GT.M MUMPS routines are distributed with uppercase names.
You can create lowercase copies of these routines if you wish, but
to avoid problems with compatibility in the future, consider keeping
only the uppercase versions of the files.

Do you want uppercase and lowercase versions of the MUMPS routines? (y or n)
```

Figure 56 - Installing the GT.M files in the specified folder.

Now that files are being installed, it asks if the user would like to keep both uppercase and lowercase versions of the MUMPS routines. We will answer *no* to maintain consistency between MUMPS and Vista. This will ensure that routine names for both environments are kept entirely in uppercase (Figure 57).

A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output shows the compilation of MUMPS routines. It starts with a prompt 'Do you want uppercase and lowercase versions of the MUMPS routines? (y or n)' where 'n' is entered. Then it says 'Compiling all of the MUMPS routines. This may take a moment.' The user enters 'GTM>' and the script proceeds with '%GDE-I-GDUSEDEFS, Using defaults for Global Directory /opt/gtm/gtmhelp.gld'. The user enters 'GDE>' three times, followed by '%GDE-I-VERIFY, Verification OK'. Then '%GDE-I-GDCREATE, Creating Global Directory file /opt/gtm/gtmhelp.gld'. The user enters 'GTM>' again, and the script repeats the '%GDE-I-GDUSEDEFS' step for 'gdehelp.gld'. The user enters 'GDE>' three times, followed by '%GDE-I-VERIFY, Verification OK'. Then '%GDE-I-GDCREATE, Creating Global Directory file /opt/gtm/gdehelp.gld'. Finally, it says 'Installation completed. Would you like all the temporary files removed from this directory? (y or n)' with a cursor.

```

osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Help

Do you want uppercase and lowercase versions of the MUMPS routines? (y or n)n

Compiling all of the MUMPS routines. This may take a moment.

GTM>
%GDE-I-GDUSEDEFS, Using defaults for Global Directory
/opt/gtm/gtmhelp.gld

GDE>
GDE>
GDE>
%GDE-I-VERIFY, Verification OK

%GDE-I-GDCREATE, Creating Global Directory file
/opt/gtm/gtmhelp.gld

GTM>
%GDE-I-GDUSEDEFS, Using defaults for Global Directory
/opt/gtm/gdehelp.gld

GDE>
GDE>
GDE>
%GDE-I-VERIFY, Verification OK

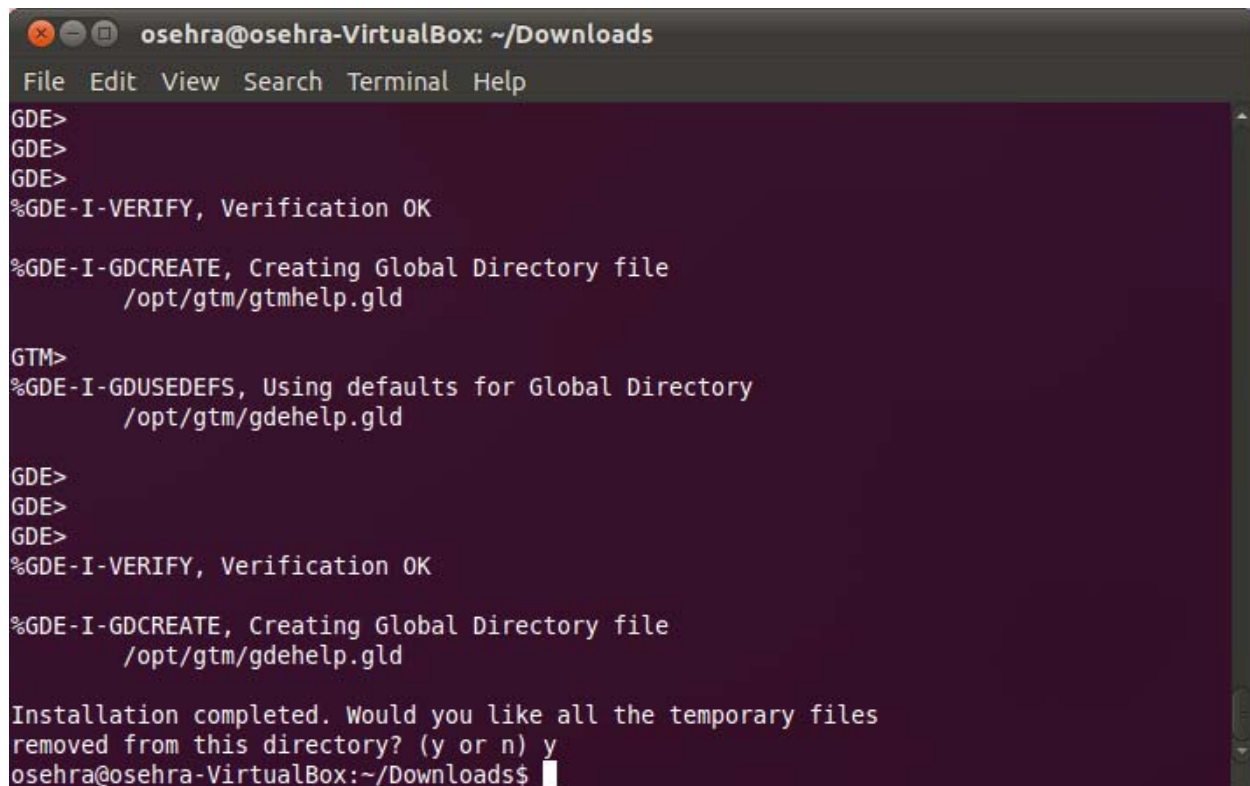
%GDE-I-GDCREATE, Creating Global Directory file
/opt/gtm/gdehelp.gld

Installation completed. Would you like all the temporary files
removed from this directory? (y or n) █

```

Figure 57 – Compiling the GT.M routines and installing them in the `/opt/gtm/` directory.

Now the installation of GT.M is complete. The last prompt asks if the user would like to remove the files in the current directory now that the installation is finished. Answer the prompt with a *y* or *n* with your preference. The script will then exit returning to the standard terminal prompt (Figure 58).

A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
GDE>
GDE>
GDE>
%GDE-I-VERIFY, Verification OK

%GDE-I-GDCREATE, Creating Global Directory file
/opt/gtm/gtmhelp.gld

GTM>
%GDE-I-GDUSEDEFS, Using defaults for Global Directory
/opt/gtm/gdehelp.gld

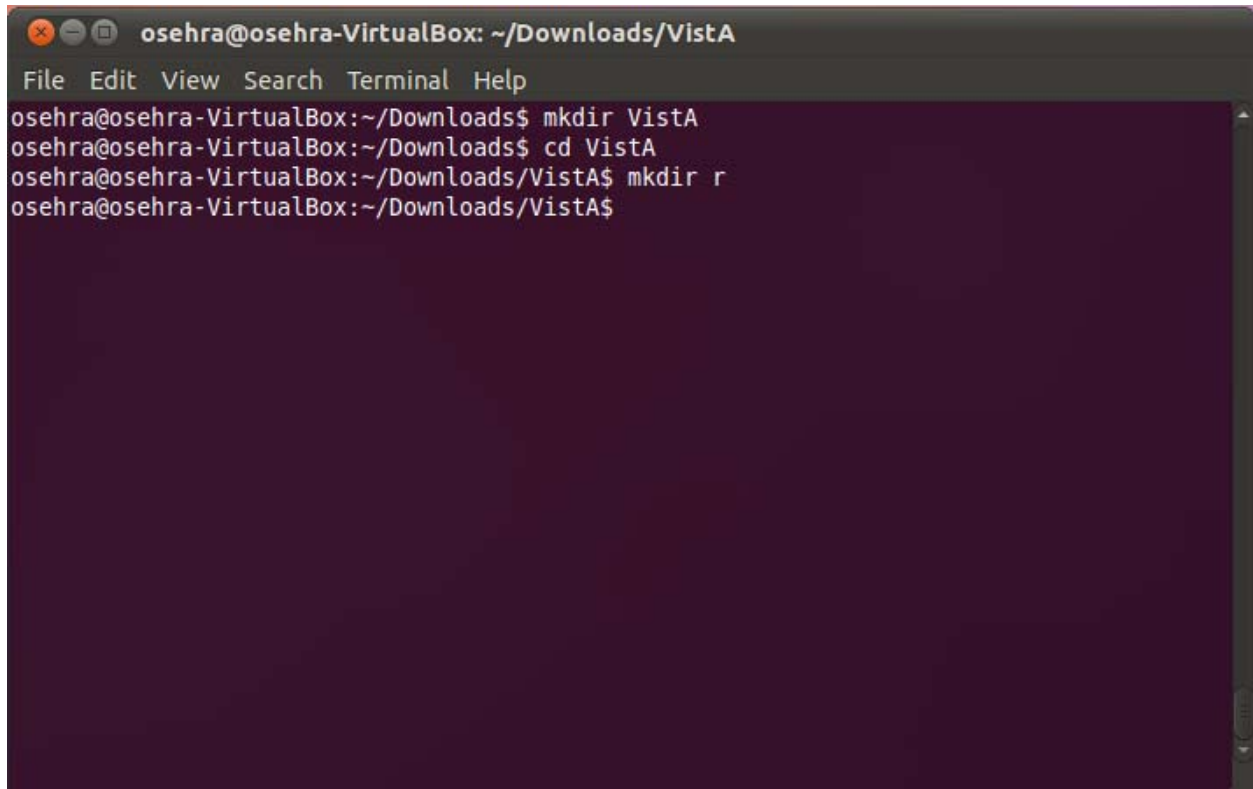
GDE>
GDE>
GDE>
%GDE-I-VERIFY, Verification OK

%GDE-I-GDCREATE, Creating Global Directory file
/opt/gtm/gdehelp.gld

Installation completed. Would you like all the temporary files
removed from this directory? (y or n) y
osehra@osehra-VirtualBox:~/Downloads$
```

Figure 58 – Finalizing the GT.M installation.

The next step is to create a directory that contains the folders and files needed to hold the VistA routines and globals that GT.M will use. We will create this folder in the /Downloads directory, but this is not the only location. This location will be used as the database directory for VistA. Make a folder called **VistA**. Inside of that VistA folder, create another folder named **r**. The **r** folder will hold the routines that GT.M will use when running VistA. These steps are shown in Figure 59.

A screenshot of a Linux terminal window. The title bar at the top reads "osehra@osehra-VirtualBox: ~/Downloads/VistA". Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal shows the following commands and their outputs:

```
osehra@osehra-VirtualBox:~/Downloads$ mkdir VistA
osehra@osehra-VirtualBox:~/Downloads$ cd VistA
osehra@osehra-VirtualBox:~/Downloads/VistA$ mkdir r
osehra@osehra-VirtualBox:~/Downloads/VistA$
```

Figure 59 – Creating the VistA folder in the Downloads directory.

The next step is to define and create the database that will be used to hold the information needed in the VistA instance. The first step is to source the gtmprofile that was created in the installation of GT.M:

```
source /opt/gtm/gtmprofile
```

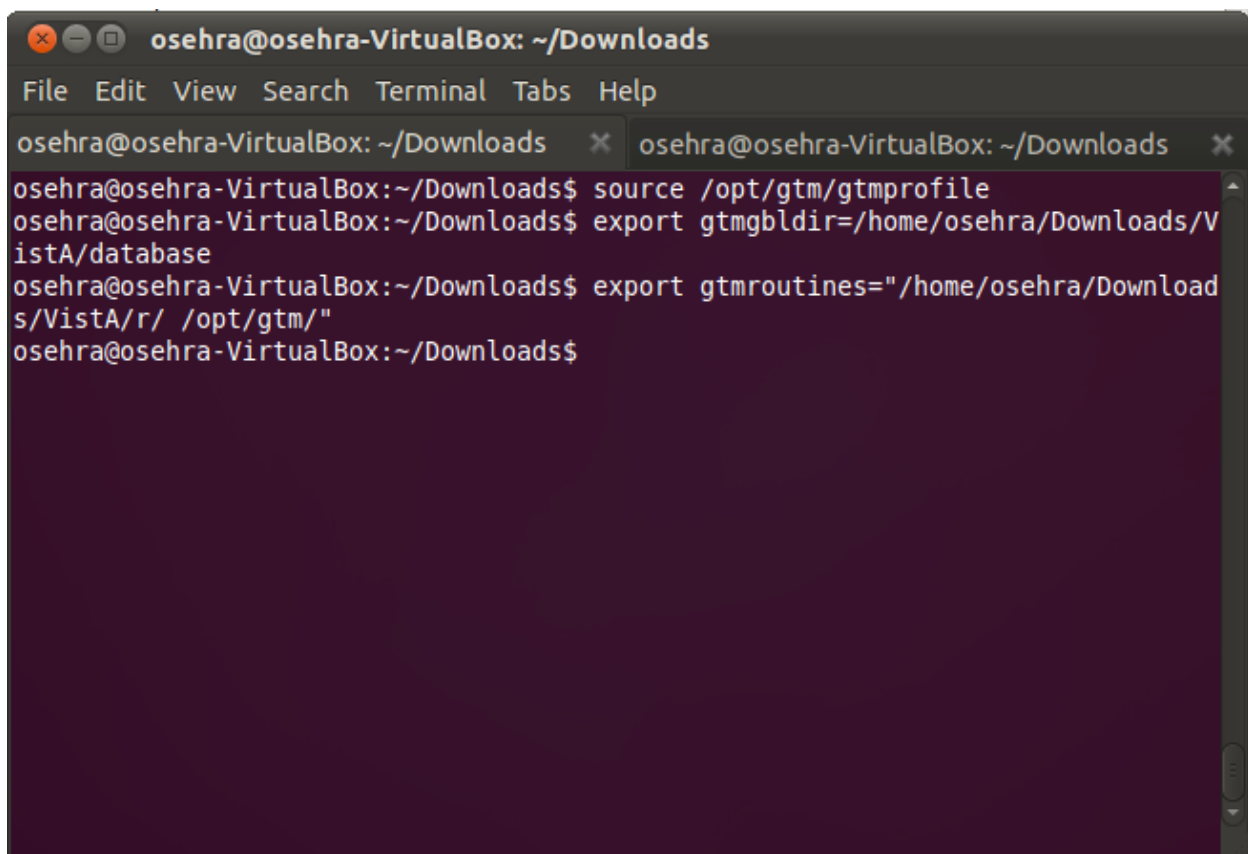
Once this is done we need to alter two environment variables that were just created to point the routines and globals to where the OSEHRA code base will reside. This will set up the environment variables needed to utilize GT.M from the command line. We will be changing the gtmgbldir entry and the gtmroutines entry. These control where the GT.M instance will look for globals and routines when it is running. These entries are set using the *export* command from the Linux terminal. The gtmgbldir should be set to the path to the **VistA** folder that was created in Figure 55 followed by database

```
export gtmgbldir=/path/to/VistA/database
```

The gtmroutines will be set to contain two paths. The first is the path to the GT.M installation, in our case in the directory /opt/gtm/, and the path to the *r* folder within the **VistA** folder.

```
export gtmroutines="/opt/gtm/ /path/to/VistA/r/ "
```

The example usage of these commands are found in Figure 60.

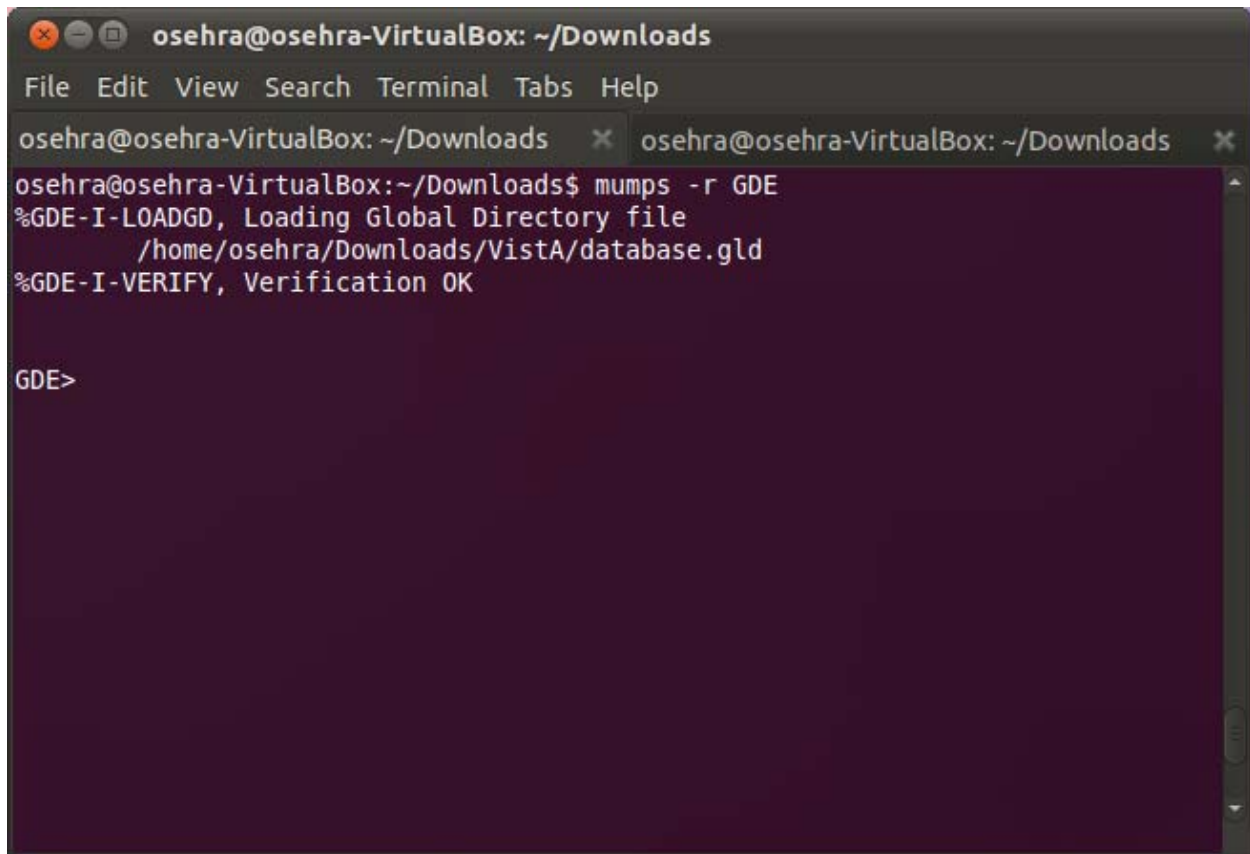
A terminal window titled "osehra@osehra-VirtualBox: ~/Downloads" with a menu bar (File, Edit, View, Search, Terminal, Tabs, Help). The terminal shows the following commands and output:

```
osehra@osehra-VirtualBox: ~/Downloads$ source /opt/gtm/gtmprofile
osehra@osehra-VirtualBox: ~/Downloads$ export gtmgbldir=/home/osehra/Downloads/VistA/database
osehra@osehra-VirtualBox: ~/Downloads$ export gtmroutines="/home/osehra/Downloads/VistA/r/ /opt/gtm/"
osehra@osehra-VirtualBox: ~/Downloads$
```

Figure 60 – Changing the environment to prepare for import.

The next step is to run the GT.M Global Directory Editor (GDE), accessed via the command:

mumps -r GDE

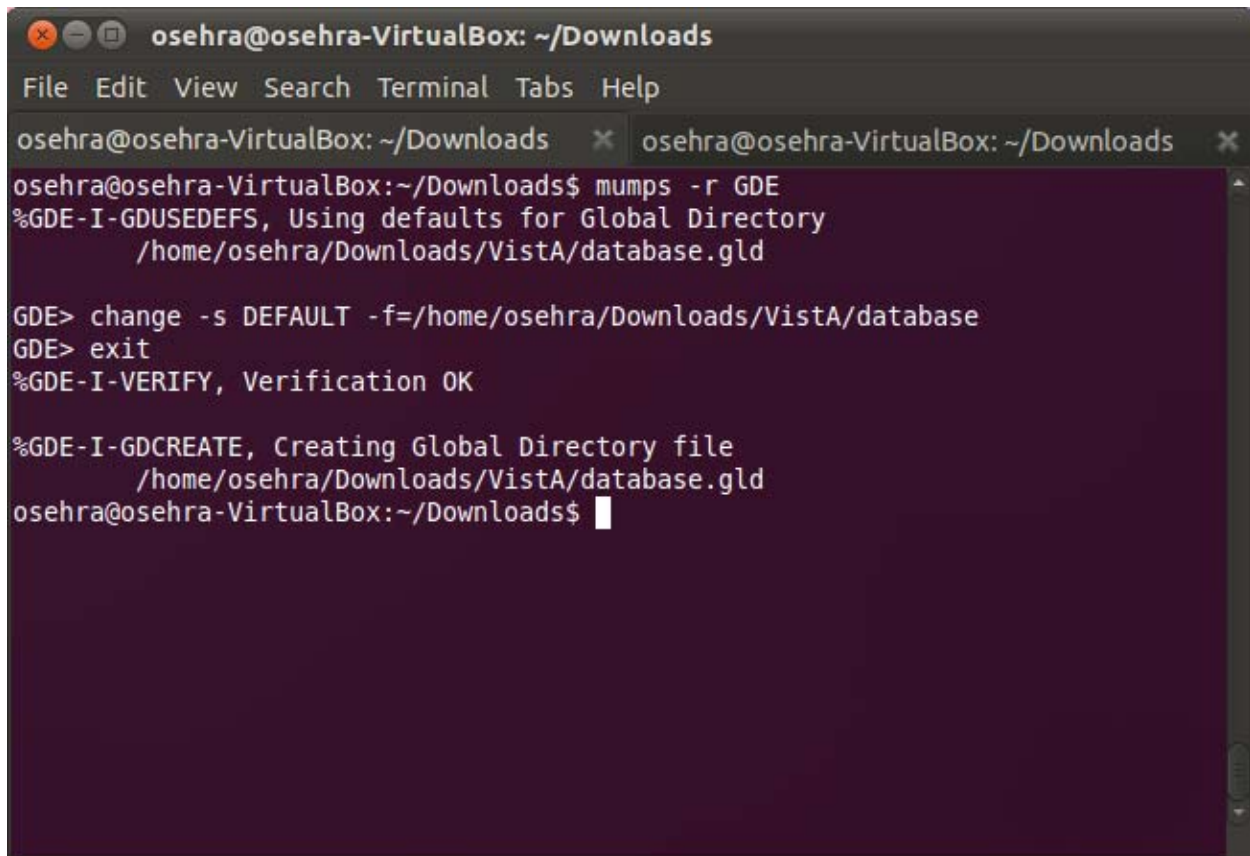
A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', 'Tabs', and 'Help'. Below the menu bar, there are two tabs, both labeled 'osehra@osehra-VirtualBox: ~/Downloads'. The terminal content shows the command 'mumps -r GDE' being executed. The output consists of two lines: '%GDE-I-LOADGD, Loading Global Directory file' followed by the path '/home/osehra/Downloads/VistA/database.gld', and '%GDE-I-VERIFY, Verification OK'. The prompt has changed from the standard shell prompt to 'GDE>'.

```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Tabs Help
osehra@osehra-VirtualBox: ~/Downloads x osehra@osehra-VirtualBox: ~/Downloads x
osehra@osehra-VirtualBox:~/Downloads$ mumps -r GDE
%GDE-I-LOADGD, Loading Global Directory file
    /home/osehra/Downloads/VistA/database.gld
%GDE-I-VERIFY, Verification OK

GDE>
```

Figure 61 – Sourcing the gtmprofile and starting the Global Directory Editor.

This command starts the GDE and will change the prompt from the standard terminal one to “GDE>”, like in Figure 61 . Within the GDE environment, the default database location needs to be changed. Enter the command *change -s DEFAULT -f=/home/\$user/Downloads/VistA/database*, replacing \$user with your user name. After that command type *exit* and the changes will be applied, like in Figure 62.



```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Tabs Help

osehra@osehra-VirtualBox: ~/Downloads x osehra@osehra-VirtualBox: ~/Downloads x
osehra@osehra-VirtualBox:~/Downloads$ mumps -r GDE
%GDE-I-GDUSEDEFS, Using defaults for Global Directory
/home/osehra/Downloads/VistA/database.gld

GDE> change -s DEFAULT -f=/home/osehra/Downloads/VistA/database
GDE> exit
%GDE-I-VERIFY, Verification OK

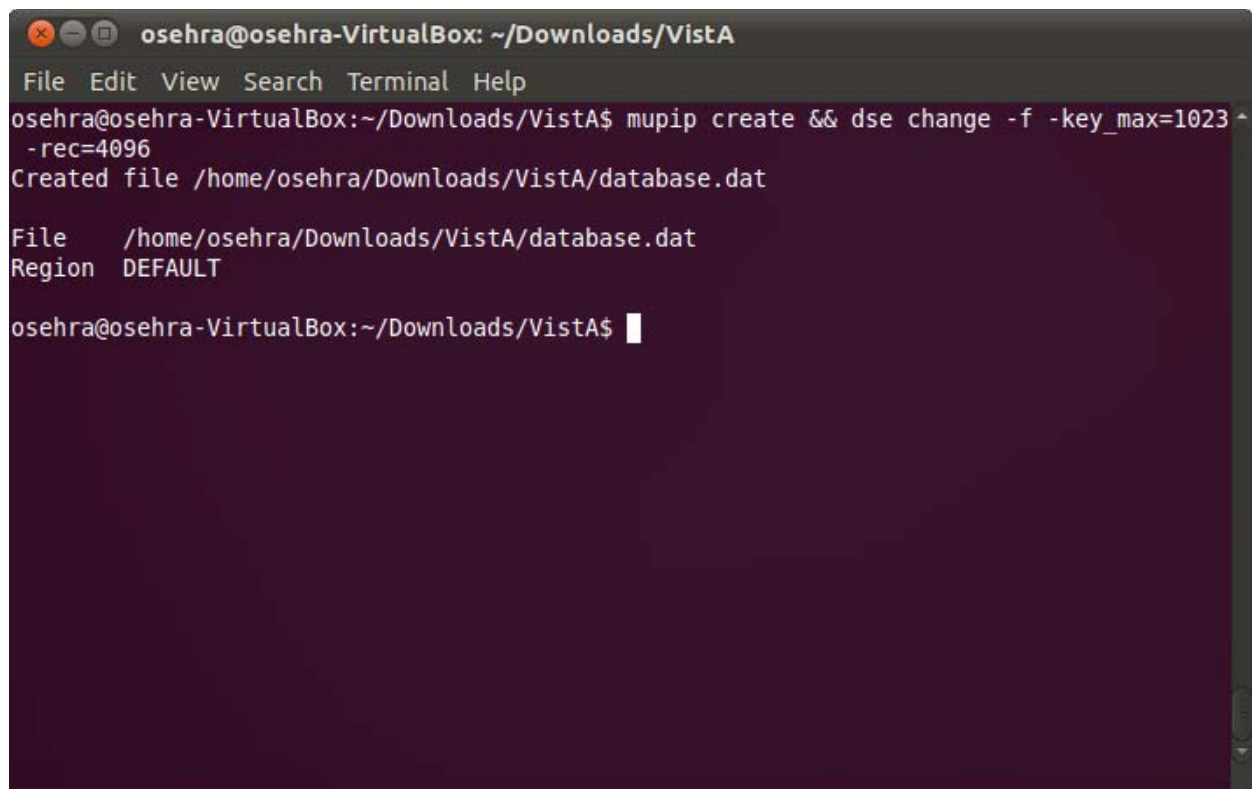
%GDE-I-GDCREATE, Creating Global Directory file
/home/osehra/Downloads/VistA/database.gld
osehra@osehra-VirtualBox:~/Downloads$
```

Figure 62 – Changing the default database location.

The next step is to create the database that is used. This is done using the mupip command. Mupip stands for MUMPS Peripheral Interchange Program. It is used to manage the database and the global directories. We will use mupip to create a database and the Database Structure Editor (DSE) to configure the database in one command. (Figure 63)

```
mupip create && dse change -f -key_max=1023 -rec=4096
```

The “mupip create” is what actually creates the database while the “dse change -f -key_max=1023 -rec=4096” changes the maximum size of a key which contains a global reference. If this is left at the default value of 255, certain globals will not be able to be imported.

A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads/VistA' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'osehra@osehra-VirtualBox:~/Downloads/VistA\$'. The command 'mupip create && dse change -f -key_max=1023 -rec=4096' is entered. The output is 'Created file /home/osehra/Downloads/VistA/database.dat'. Below this, a file listing shows '/home/osehra/Downloads/VistA/database.dat' with region 'DEFAULT'. The prompt returns to 'osehra@osehra-VirtualBox:~/Downloads/VistA\$' with a cursor.

```
osehra@osehra-VirtualBox: ~/Downloads/VistA
File Edit View Search Terminal Help
osehra@osehra-VirtualBox:~/Downloads/VistA$ mupip create && dse change -f -key_max=1023 -rec=4096
Created file /home/osehra/Downloads/VistA/database.dat

File    /home/osehra/Downloads/VistA/database.dat
Region  DEFAULT

osehra@osehra-VirtualBox:~/Downloads/VistA$
```

Figure 63 – Creating the database.dat within the VistA folder.

Now, the environment is set up to import the routines and globals from the OSEHRA code base.

[Return to the Beginning](#)

Importing the OSEHRA Code Base into GT.M

The method of importing the code base is almost the same as the importing the code base into Caché. Both require the packing of routines and globals using the accompanying python scripts. Following the instructions preceding Figure 38, create the *routines.ro* and *globals.lst* .

Importing the routines is done using the MUMPS routine named %RI. This is the MUMPS standard routine to import routines from a .RO file like what was created when the Python scripts were run. To start the GT.M instance in the terminal, simply enter the command: *gtm* (Figure 64).

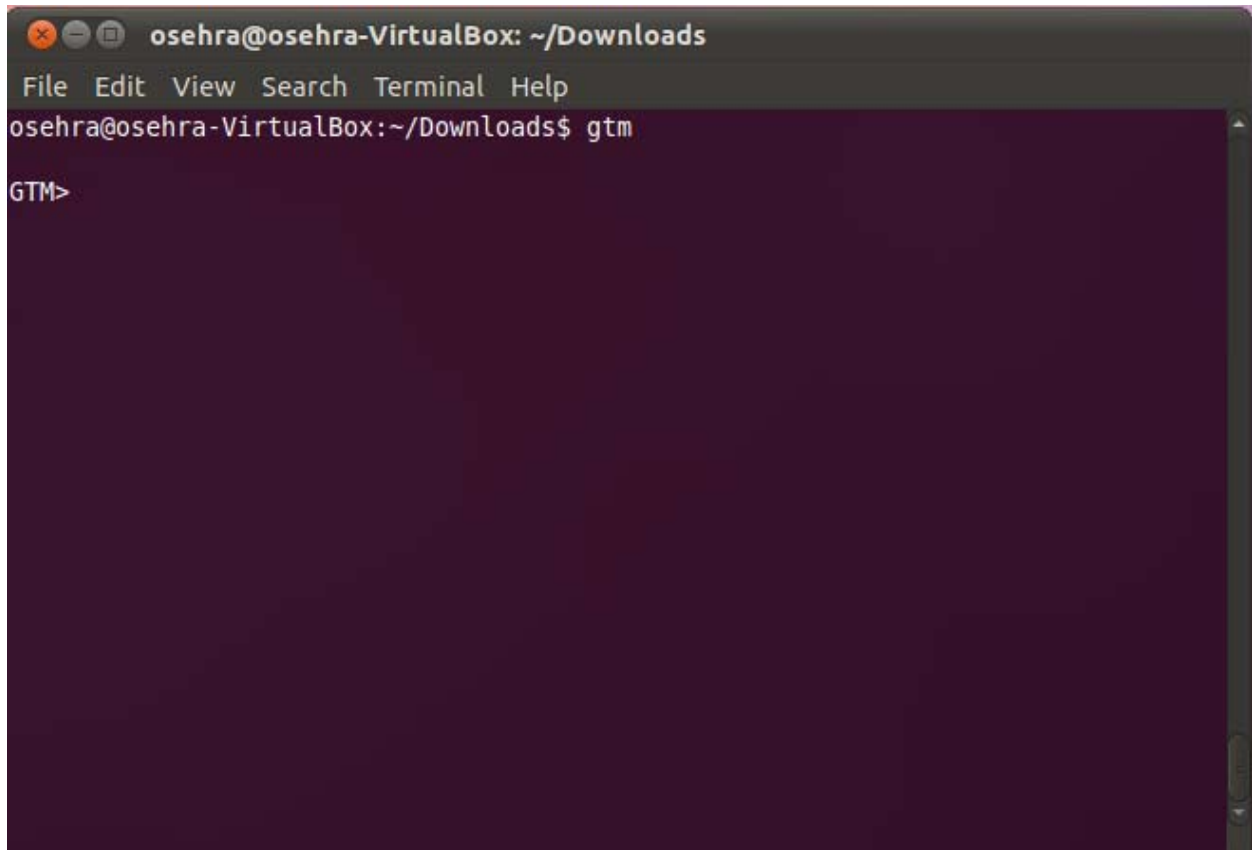
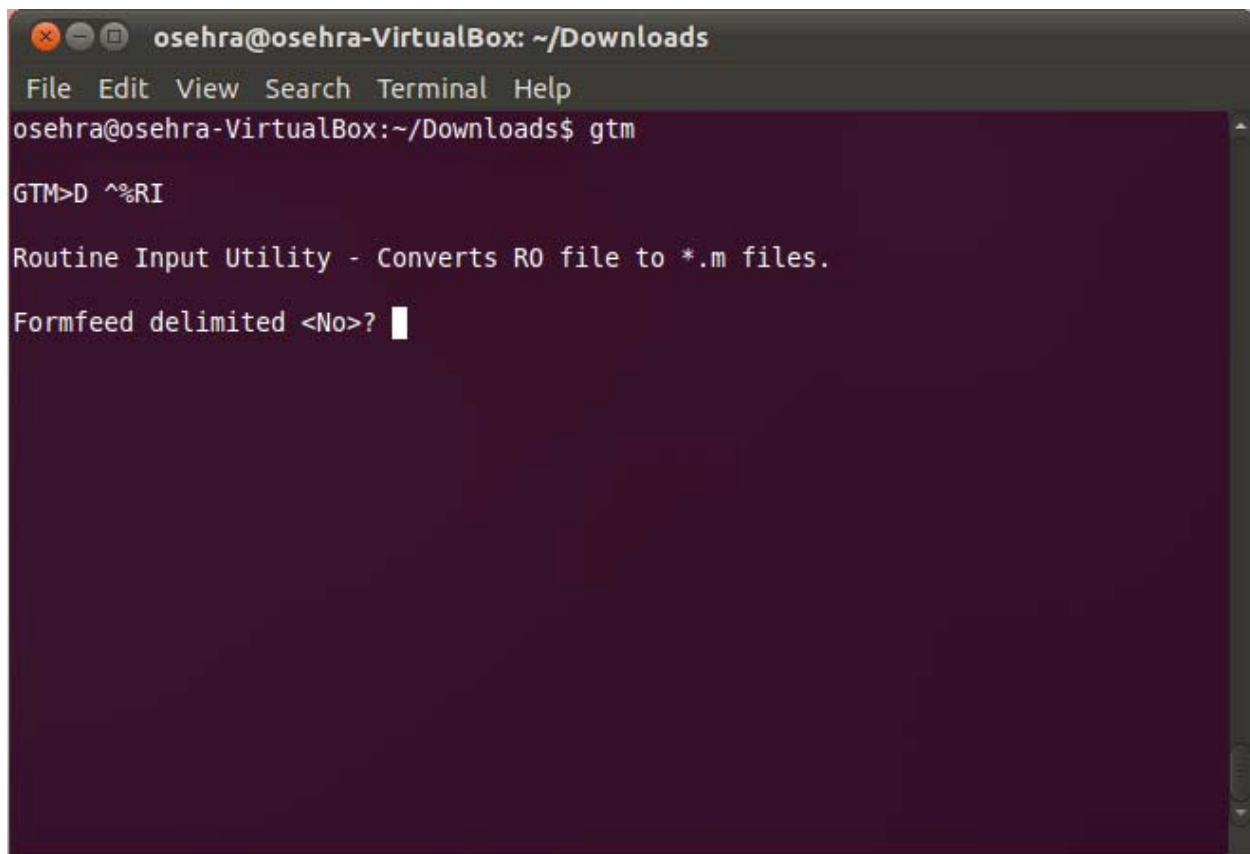
A screenshot of a terminal window titled "osehra@osehra-VirtualBox: ~/Downloads". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal shows the command "osehra@osehra-VirtualBox:~/Downloads\$ gtm" being entered, followed by a new line with the prompt "GTM>". The terminal background is dark purple.

Figure 64 – Starting the GT.M instance.

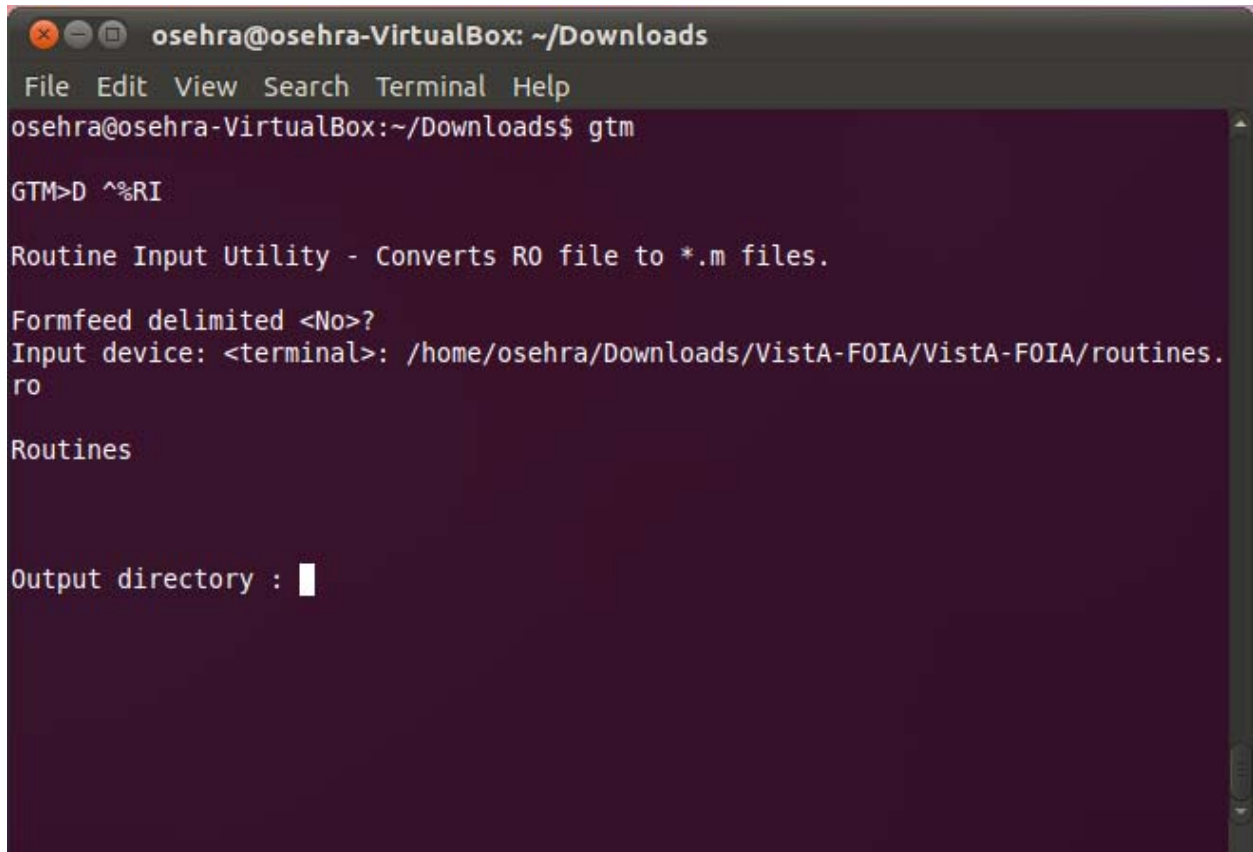
When the “GTM>” prompt is there, you are in the GT.M environment and can execute the %RI routine using the command *D ^%RI* (Figure 65).

A terminal window titled 'osehra@osehra-VirtualBox: ~/Downloads'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command prompt shows 'osehra@osehra-VirtualBox:~/Downloads\$ gtm'. Below this, the prompt changes to 'GTM>D ^%RI'. The next line of text is 'Routine Input Utility - Converts R0 file to *.m files.'. The final line is 'Formfeed delimited <No>?' followed by a cursor.

```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
osehra@osehra-VirtualBox:~/Downloads$ gtm
GTM>D ^%RI
Routine Input Utility - Converts R0 file to *.m files.
Formfeed delimited <No>? █
```

Figure 65 – Beginning of the %RI routine.

The [routines.ro](#) file that was created earlier is not formfeed delimited, so the default option for the first prompt is the correct one to choose. When the prompt asks for an “Input Device,” enter the path to the [routines.ro](#) that was created in the earlier step. Our path to the [routines.ro](#) is shown entered into Figure 66.



```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
osehra@osehra-VirtualBox:~/Downloads$ gtm

GTM>D ^%RI

Routine Input Utility - Converts R0 file to *.m files.

Formfeed delimited <No>?
Input device: <terminal>: /home/osehra/Downloads/VistA-FOIA/VistA-FOIA/routines.
ro

Routines

Output directory : █
```

Figure 66 – Entering the path to the routines.ro file.

This brings the prompt asking for the output directory. The path entered here should point to the VistA/r folder that was used to set the gtmroutines environment variable in the previous step. This is the location where the %RI routine will put the routines that it imports from [routines.ro](#).

After entering the path, the names of the routines that are imported are shown on the terminal window as they are processed. When the routine is finished, it will display the amount of lines restored and the number of routines processed, then show the GT.M prompt (Figure 67).

```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Help
ABSVSITE ABSVT ABSVTC ABSVTC1 ABSVTED ABSVTED1 ABSVTED2 ABSVTED3
ABSVTED4 ABSVTED5 ABSVTIME ABSVTP ABSVTP1 ABSVTP2 ABSVTPR ABSVTPR1
ABSVTPR2 ABSVU ABSVU1 ABSVU2 ABSVU3 ABSVVIEW ABSVYN XOBWD
XOBWENV XOBWLIB XOBWLIB1 XOBWPST XOBWPWD XOBWSSL XOBWU XOBWU1
XOBWUA XOBWUA1 XOBWUS XOBWUS1 XOBWUS2 WV14PRE WV14PST WV14PST1
WV16PST WV19PST WV22PST WV6PST WV7PST WVALERTC WVALERTF WVALERTP
WVALERTR WVALERTS WVRDUP WVRNED WVRNED1 WVRNEDH WVRNOT WVRNOT1
WVRNOT2 WVRPCD WVRPCD1 WVRPCD2 WVRPCD3 WCMGR WVDIAG WVDIAGS
WVENV WVEPTR WVFACE WVFMAN WVGETAL1 WVGETALL WVHS WVLAB
WVLABAD1 WVLABADD WVLABCHK WVLABLG WVLABLG1 WVLABLG2 WVLABWP WVLABWPC
WVLABWPS WVLETDQ WVLETPR WVLOGO WVLRLINK WVMGRP WVMSTL WVMSTL1
WVNOTIF WVNOTIF1 WVPATE WVPATP WVPRE WVPROC WVPROC1 WVPROF
WVPROF1 WVPROF2 WVPROF3 WVRPCD WVPURP WVRAD WVRADWP WVRALIN1
WVRALINK WVREFUSE WVRPCNO WVRPCNO1 WVRPCPR WVRPCPR1 WVRPCPT WVRPPCD
WVRPPCD1 WVRPPCD2 WVRPPCD3 WVRPCR WVRPCR1 WVRPCR2 WVRPSNP WVRPSNP1
WVRPSNPR WVRPST WVSELECT WVSITE WVSNOEMD WVUTL1 WVUTL10 WVUTL1A
WVUTL2 WVUTL3 WVUTL4 WVUTL5 WVUTL6 WVUTL7 WVUTL8 WVUTL9
WVYNOTP WIIACT4 WIIADT1 WIIELG WIIGATD WIILM WIILM01 WIILM02
WIILM03 WIILM04 WIISERV ZGI ZGO

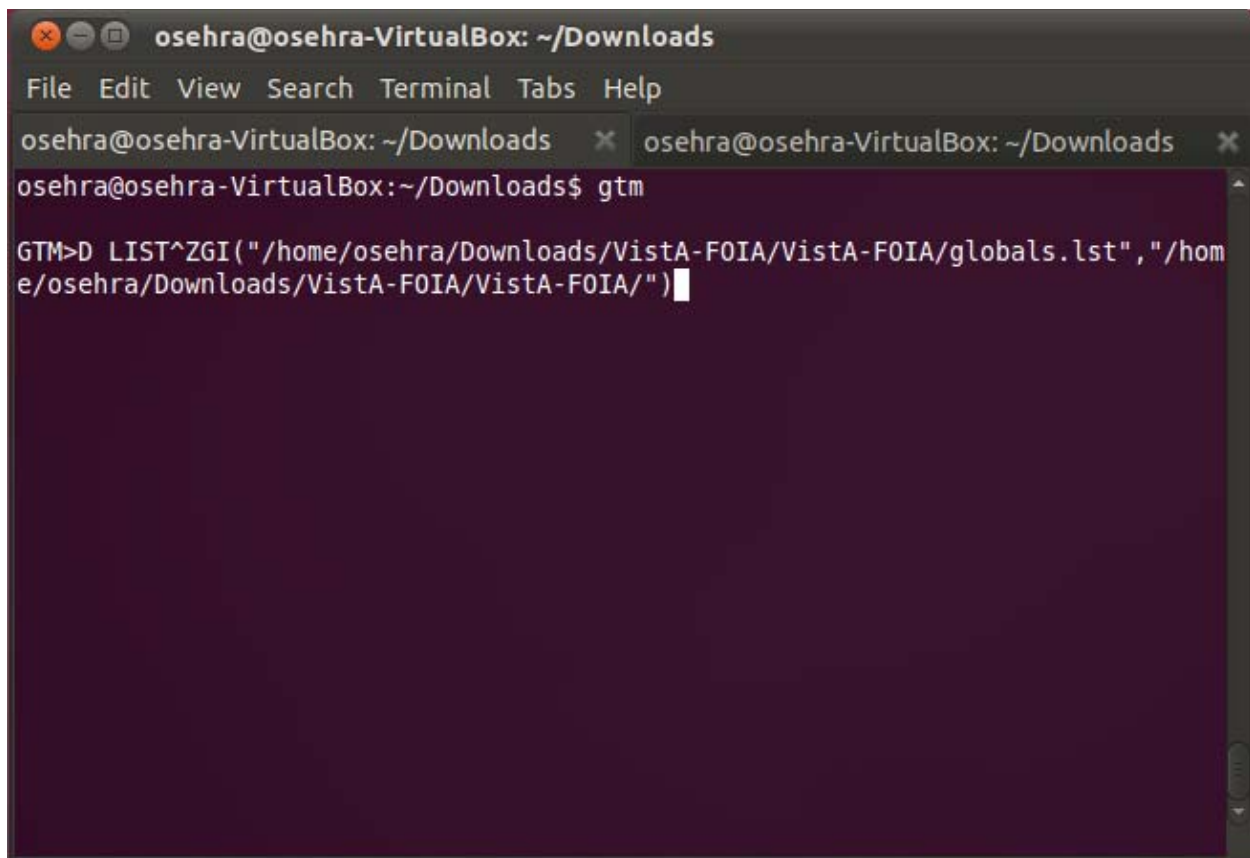
Restored 2349289 lines in 26037 routines.
GTM>
```

Figure 67 – End of the %RI routine.

Importing the globals is done with the use of a routine that was just imported. The ZGI routine was written to import the globals from the OSEHRA structure into a MUMPS environment. The command to do this is:

```
D LIST^ZGI("c:\path\to\VistA-FOIA\globals.lst","c:\path\to\VistA-FOIA\")
```

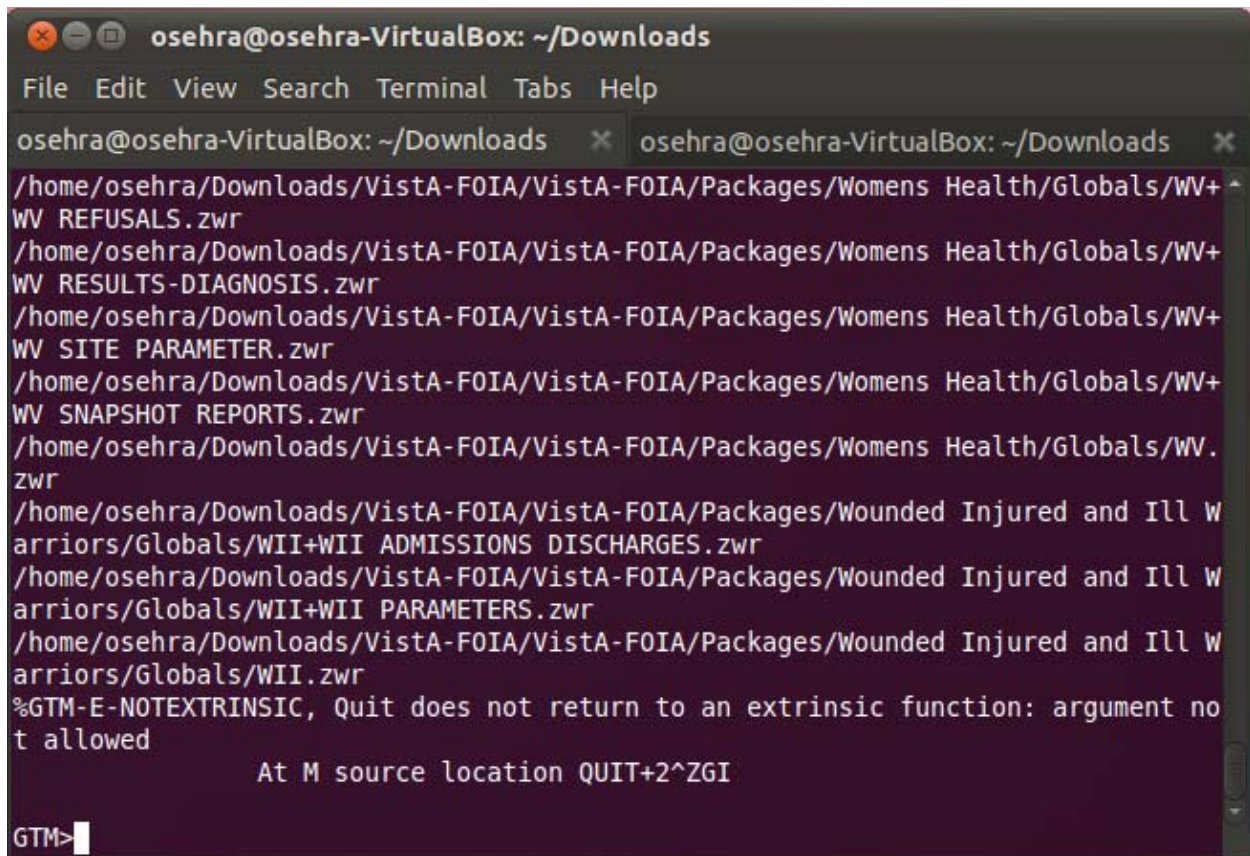
This will take the [globals.lst](#) file and use the entries in it to tell GT.M to import that .zwr file (Figure 68).



```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Tabs Help
osehra@osehra-VirtualBox: ~/Downloads x osehra@osehra-VirtualBox: ~/Downloads x
osehra@osehra-VirtualBox:~/Downloads$ gtm
GTM>D LIST^ZGI("/home/osehra/Downloads/VistA-F0IA/VistA-F0IA/globals.lst", "/home/osehra/Downloads/VistA-F0IA/VistA-F0IA/")
```

Figure 68 – Starting the ZGI import.

While the routine is running, the names of the .zwr files will be printed to the screen as they are being processed. This is going through the OSEHRA Code base and importing all of the .zwr files from each package. The final package imported is the “Wounded Injured and Ill Warriors.” There may also be a warning about not returning to an extrinsic function; this is expected and will not hamper the import of the globals. After that, the GT.M prompt is displayed and the globals are finished importing (Figure 69).



```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Tabs Help

osehra@osehra-VirtualBox: ~/Downloads x osehra@osehra-VirtualBox: ~/Downloads x
/home/osehra/Downloads/VistA-FOIA/VistA-FOIA/Packages/Womens Health/Globals/WV+
WV REFUSALS.zwr
/home/osehra/Downloads/VistA-FOIA/VistA-FOIA/Packages/Womens Health/Globals/WV+
WV RESULTS-DIAGNOSIS.zwr
/home/osehra/Downloads/VistA-FOIA/VistA-FOIA/Packages/Womens Health/Globals/WV+
WV SITE PARAMETER.zwr
/home/osehra/Downloads/VistA-FOIA/VistA-FOIA/Packages/Womens Health/Globals/WV+
WV SNAPSHOT REPORTS.zwr
/home/osehra/Downloads/VistA-FOIA/VistA-FOIA/Packages/Womens Health/Globals/WV.
zwr
/home/osehra/Downloads/VistA-FOIA/VistA-FOIA/Packages/Wounded Injured and Ill W
arriors/Globals/WII+WII ADMISSIONS DISCHARGES.zwr
/home/osehra/Downloads/VistA-FOIA/VistA-FOIA/Packages/Wounded Injured and Ill W
arriors/Globals/WII+WII PARAMETERS.zwr
/home/osehra/Downloads/VistA-FOIA/VistA-FOIA/Packages/Wounded Injured and Ill W
arriors/Globals/WII.zwr
%GTM-E-NOTEXTRINSIC, Quit does not return to an extrinsic function: argument no
t allowed
      At M source location QUIT+2^ZGI
GTM>
```

Figure 69 – End of the ZGI Import

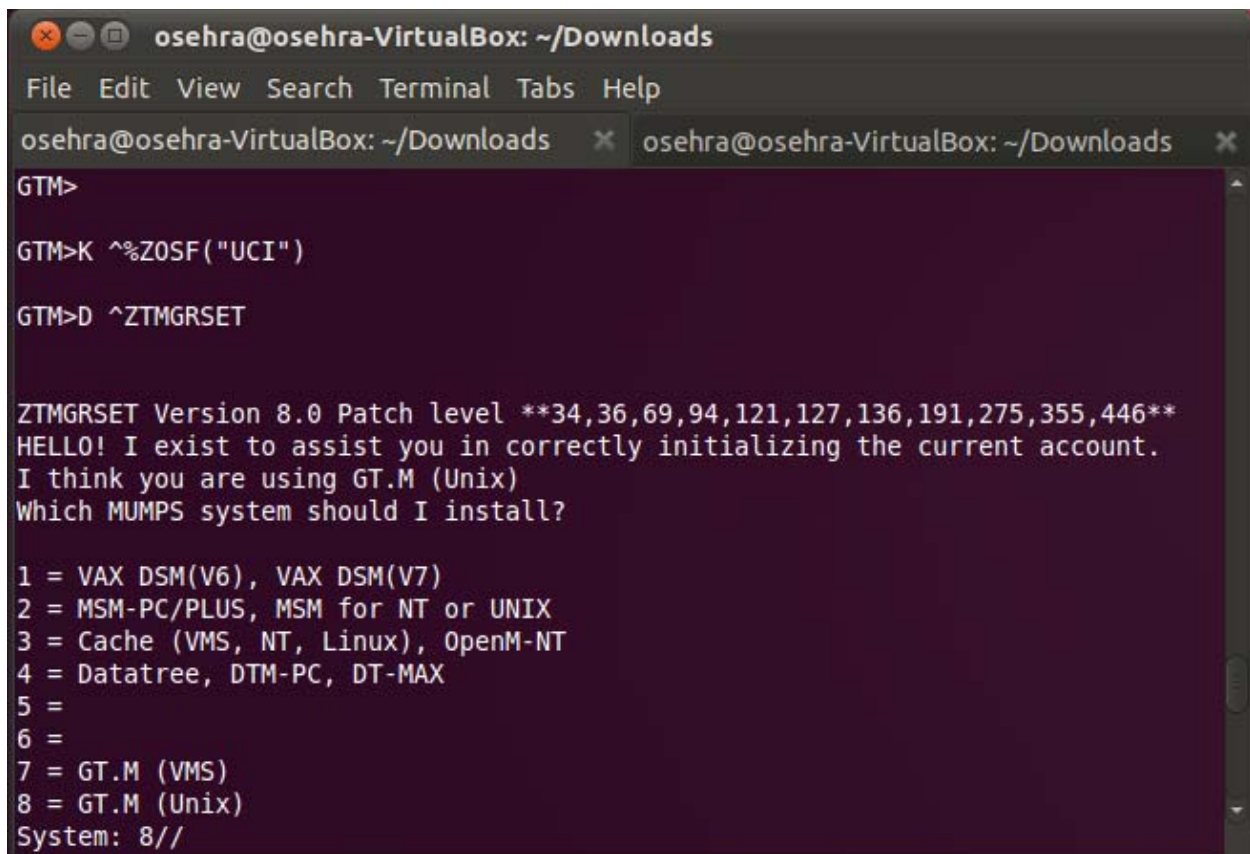
Some configuration within the VistA environment is necessary before you have a full VistA instance.

Figure 70 shows the two steps that need to be run to configure the VistA instance. The first step is to kill the box/volume pair that the imported globals have. This will let you set the box/volume pair to match your system during the next configuration step. This is done with the command:

```
K ^%ZOSF("UCI")
```

Then start the ZTMGRSET routine, which will configure the VistA instance by renaming some system-specific routines. This is done using the command:

```
D ^ZTMGRSET
```



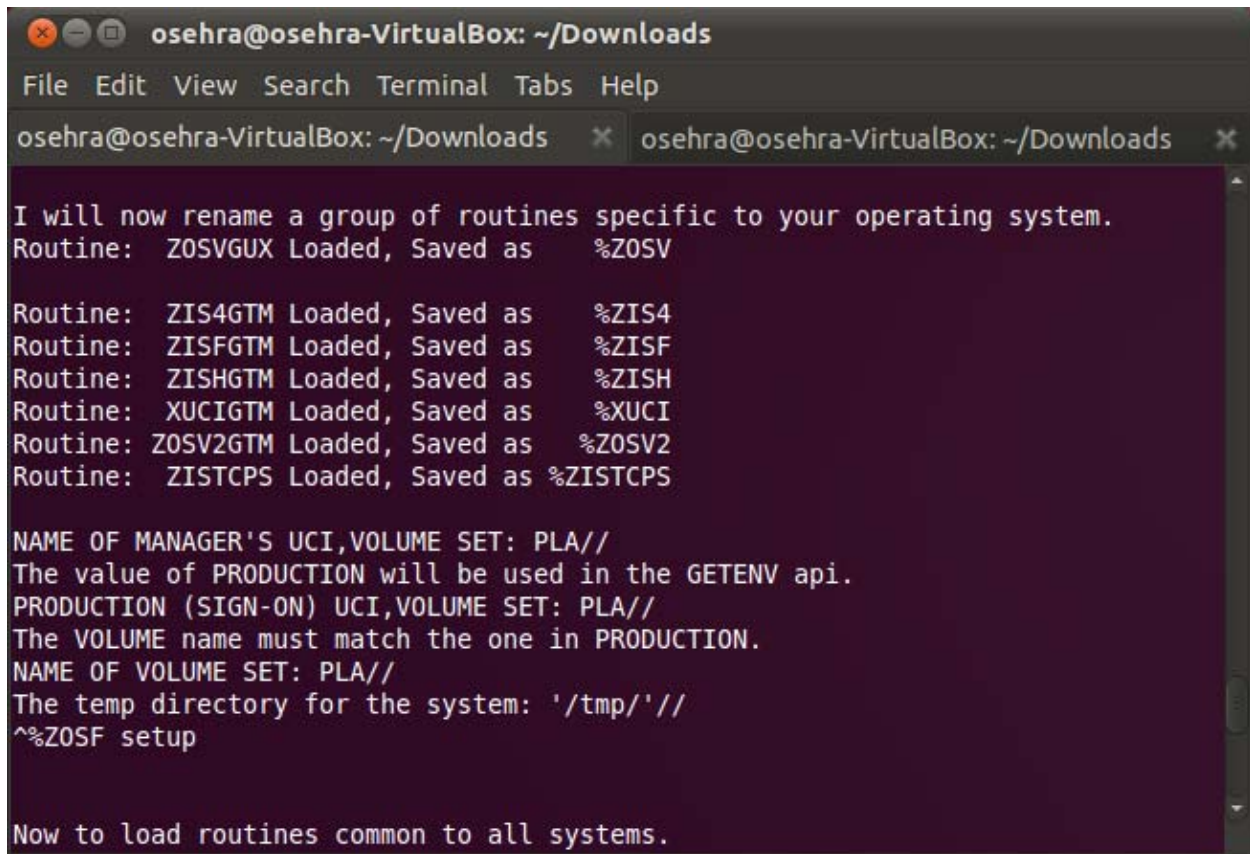
```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Tabs Help
osehra@osehra-VirtualBox: ~/Downloads x osehra@osehra-VirtualBox: ~/Downloads x
GTM>
GTM>K ^%ZOSF("UCI")
GTM>D ^ZTMGRSET

ZTMGRSET Version 8.0 Patch level **34,36,69,94,121,127,136,191,275,355,446**
HELLO! I exist to assist you in correctly initializing the current account.
I think you are using GT.M (Unix)
Which MUMPS system should I install?

1 = VAX DSM(V6), VAX DSM(V7)
2 = MSM-PC/PLUS, MSM for NT or UNIX
3 = Cache (VMS, NT, Linux), OpenM-NT
4 = Datatree, DTM-PC, DT-MAX
5 =
6 =
7 = GT.M (VMS)
8 = GT.M (Unix)
System: 8//
```

Figure 70 – Killing the UCI and Starting the ZTMGRSET routine.

This option tells Vista what kind of MUMPS environment you are using. We are using GT.M on Unix so if 8 is not the default option, enter 8 to select it (Figure 70).



```
osehra@osehra-VirtualBox: ~/Downloads
File Edit View Search Terminal Tabs Help

osehra@osehra-VirtualBox: ~/Downloads x osehra@osehra-VirtualBox: ~/Downloads x

I will now rename a group of routines specific to your operating system.
Routine: ZOSVGUX Loaded, Saved as %ZOSV

Routine: ZIS4GTM Loaded, Saved as %ZIS4
Routine: ZISFGTM Loaded, Saved as %ZISF
Routine: ZISHGTM Loaded, Saved as %ZISH
Routine: XUCIGTM Loaded, Saved as %XUCI
Routine: ZOSV2GTM Loaded, Saved as %ZOSV2
Routine: ZISTCPS Loaded, Saved as %ZISTCPS

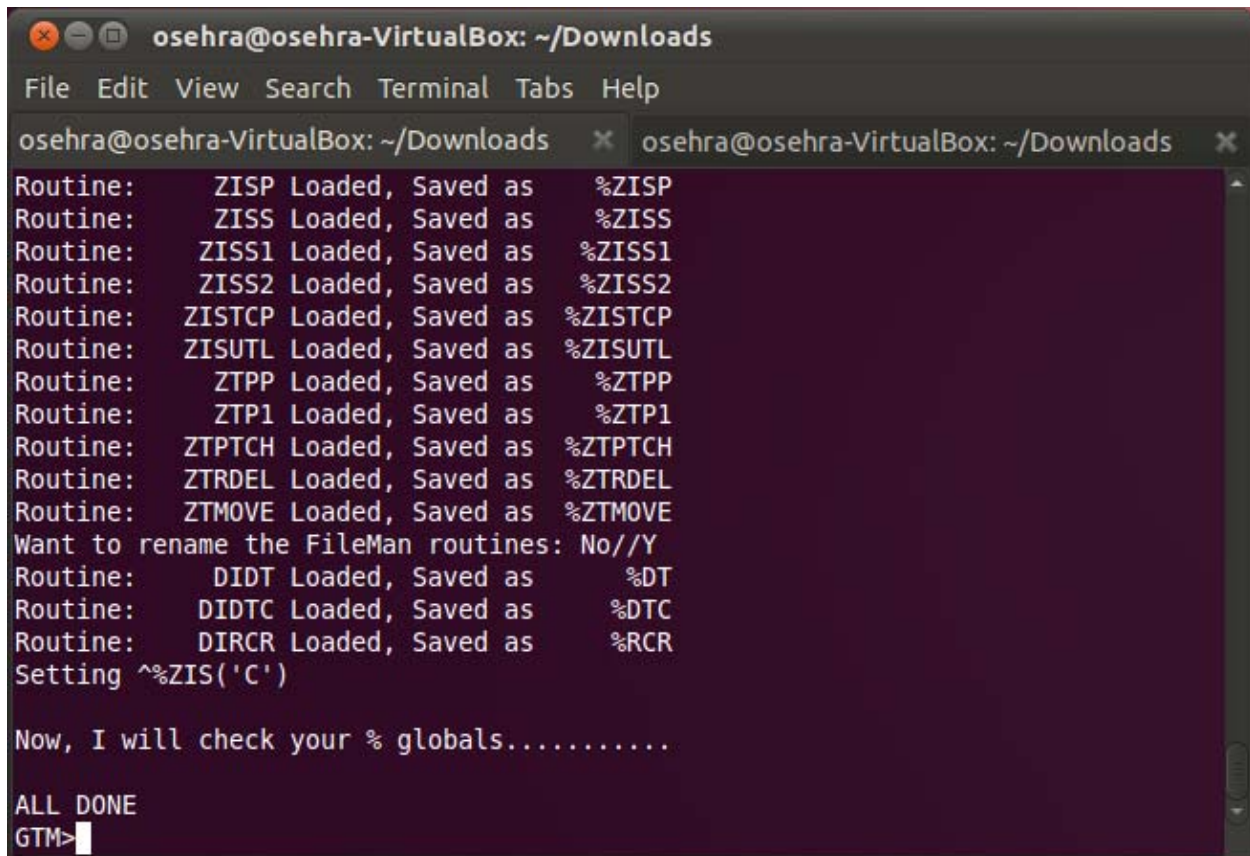
NAME OF MANAGER'S UCI,VOLUME SET: PLA//
The value of PRODUCTION will be used in the GETENV api.
PRODUCTION (SIGN-ON) UCI,VOLUME SET: PLA//
The VOLUME name must match the one in PRODUCTION.
NAME OF VOLUME SET: PLA//
The temp directory for the system: '/tmp/'//
^%ZOSF setup

Now to load routines common to all systems.
```

Figure 71 – Setting the Box/Volume pair and temp directory.

After loading a few routines, the configuration will ask you for the names of the box/volume pair of the system, the name of the manager's namespace, and the temp directory. Figure 71 shows the default answers being accepted for these prompts. They can be set if you need a specific name, but we used the defaults of PLA for all names and the /tmp/ directory for the system.

It will load and save some other routines, then ask if you "Want to rename the FileMan routines:." We answer this option with a YES. The routine then loads three more routines, checks the % globals, and exits. Now you are ready to start testing the OSEHRA Code base (Figure 72).



The image shows a terminal window titled "osehra@osehra-VirtualBox: ~/Downloads". The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", "Tabs", and "Help". There are two tabs open, both with the same title. The terminal output is as follows:

```
osehra@osehra-VirtualBox: ~/Downloads
Routine:      ZISP Loaded, Saved as      %ZISP
Routine:      ZISS Loaded, Saved as      %ZISS
Routine:      ZISS1 Loaded, Saved as      %ZISS1
Routine:      ZISS2 Loaded, Saved as      %ZISS2
Routine:      ZISTCP Loaded, Saved as      %ZISTCP
Routine:      ZISUTL Loaded, Saved as      %ZISUTL
Routine:      ZTPP Loaded, Saved as      %ZTPP
Routine:      ZTP1 Loaded, Saved as      %ZTP1
Routine:      ZTPTCH Loaded, Saved as      %ZTPTCH
Routine:      ZTRDEL Loaded, Saved as      %ZTRDEL
Routine:      ZTMOVE Loaded, Saved as      %ZTMOVE
Want to rename the FileMan routines: No//Y
Routine:      DIDT Loaded, Saved as      %DT
Routine:      DIDTC Loaded, Saved as      %DTC
Routine:      DIRCR Loaded, Saved as      %RCR
Setting ^%ZIS('C')

Now, I will check your % globals.....

ALL DONE
GTM>
```

Figure 72 – End of the ZTMGRSET configuration routine.

[Return to the Beginning](#)

Setting up the Testing Environment

To configure the environment for testing, start the cmake-gui.exe. Once it has started (Figure 73), set the two paths at the top of the window so that the source code points to the *Testing Source Directory* from step 1 above. This directory will point to a directory (folder) containing *CMakeLists.txt* file. The Binaries path can go to a folder of your choice, but note that whatever directory is provided will be the *Testing Binary Directory*, and after configuration, it will contain CMake files defining the tests to be run.

Once those are set, click the Configure button. The interface will then ask you to specify a generator (Figure 74). These are normally used to check for working C and C++ compilers. For Vista testing, the generator does nothing and is unneeded. This selection can be chosen arbitrarily to any displayed generator so we recommend selecting the default value. Click finish after the selection is made to continue the configuration process.

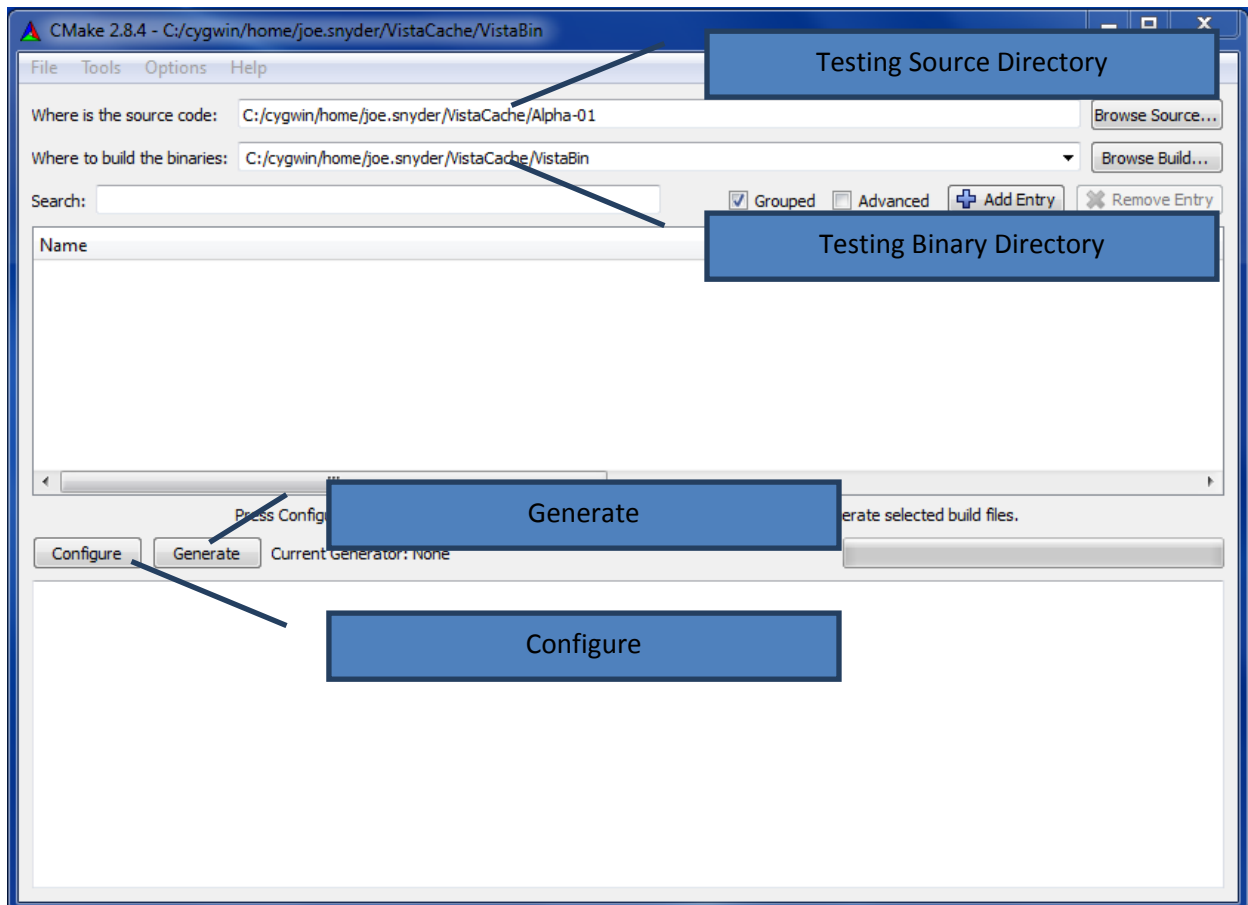


Figure 73 - Initial cmake-gui page.

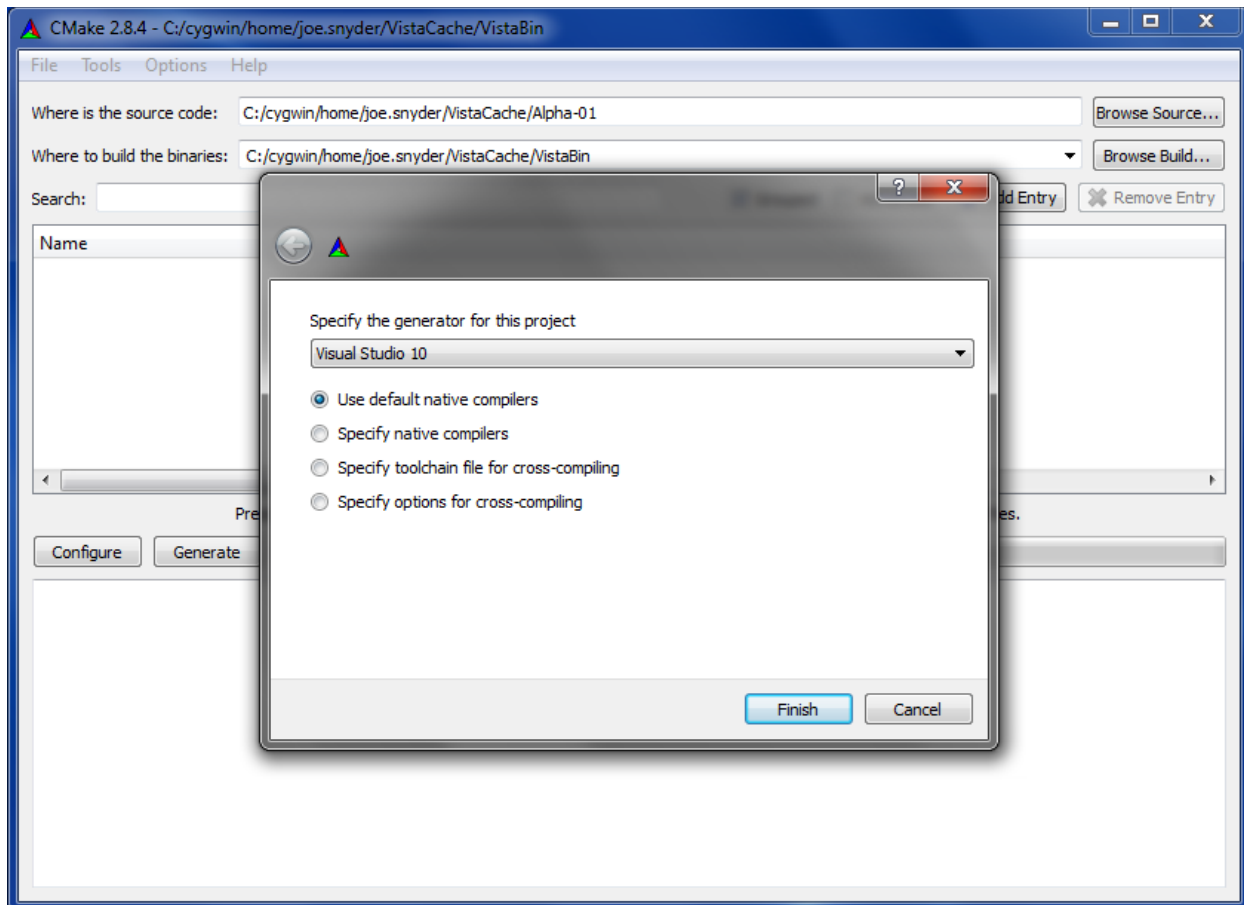


Figure 74 - Generator selection.

Following generator selection, the interface will produce a highlighted display such as in Figure 75 for Linux and Figure 76 for Windows. The entries in the window are the variables which can be set to control the testing process. Most of the values should be set correctly by the automated configuration process, but the scripting environment and the location of the Vista source code may need to be set appropriately. To aid in the configuring, most variables have a mouse-over tip which explains in greater detail what the variable should contain.

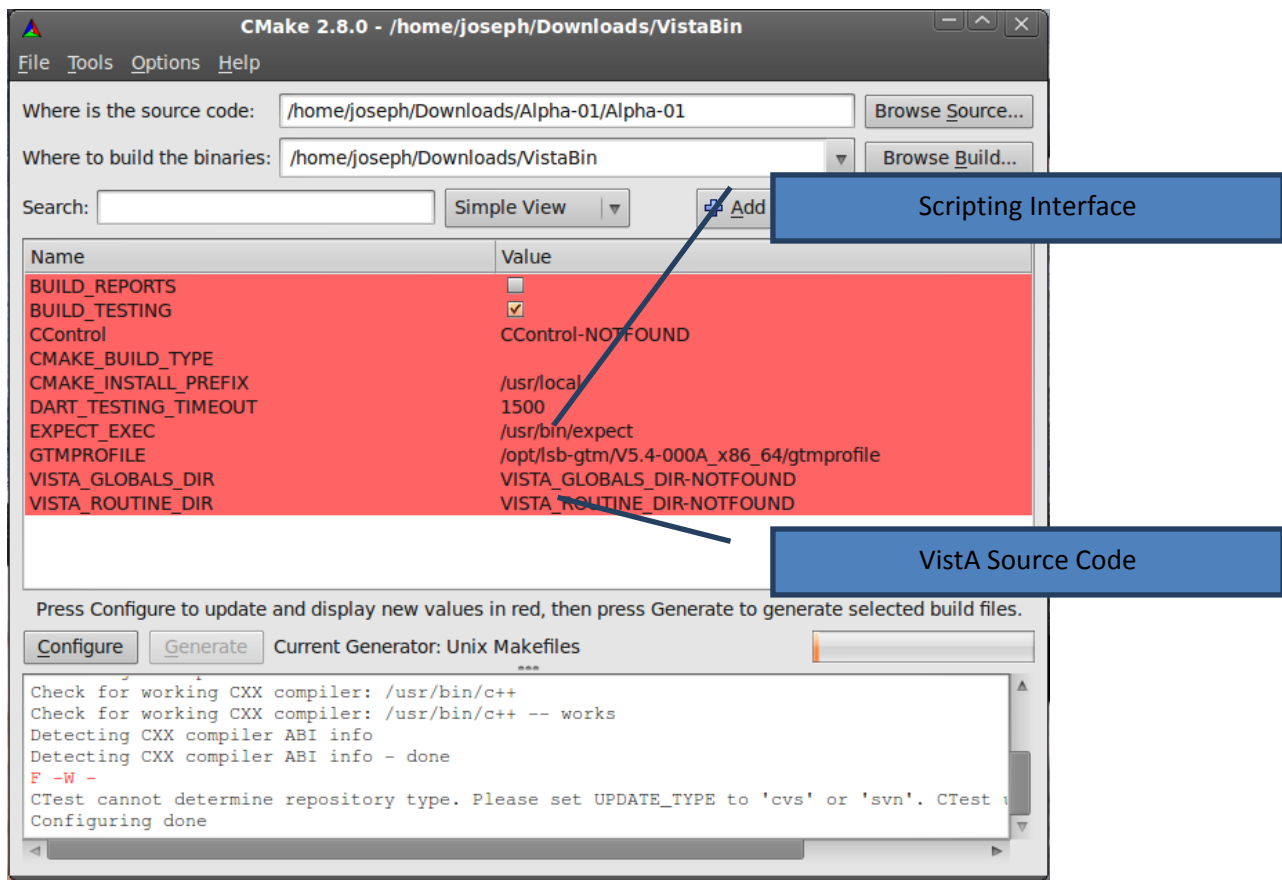


Figure 75 - Linux interface after generator selection is complete.

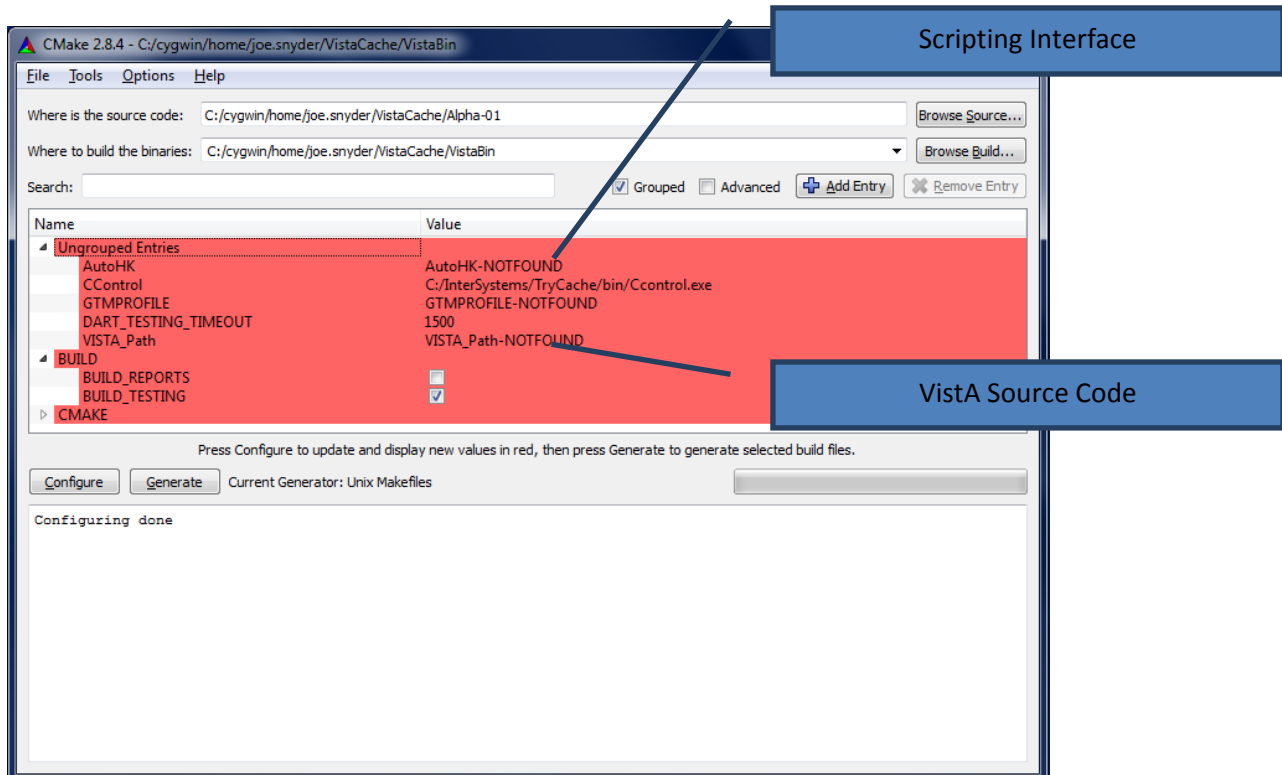


Figure 76 - Windows interface after generator selection is complete.

Looking first at the Windows example with Caché (Figure 77), three variables corresponding to AutoHK, CControl, and VISTA_Path need to be set or verified according to the values in Figure 78.

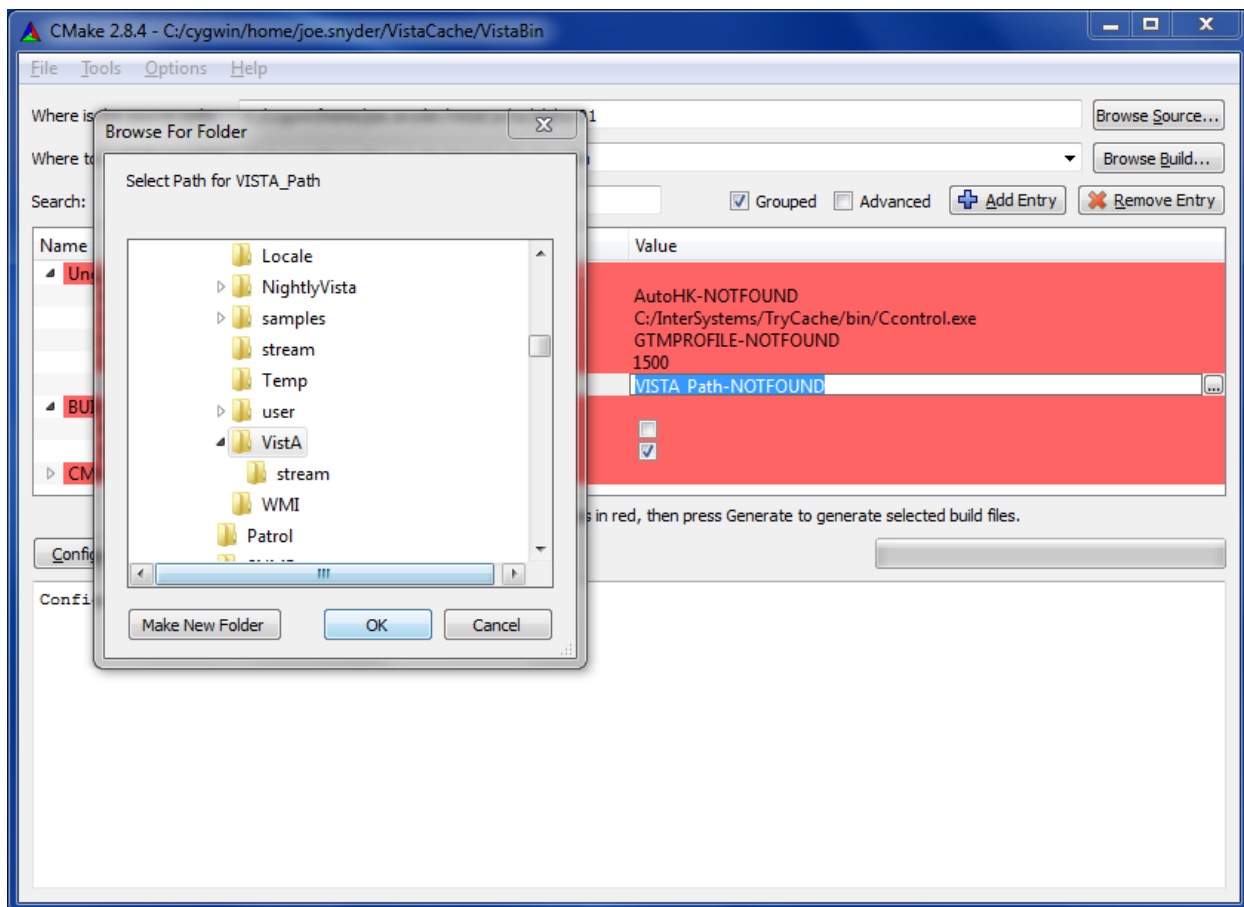


Figure 77 - Setting the VistA variables for Caché and Windows. The pop-up shows setting the VISTA_Path. The tool-top for this entry reads “Path to the VistA folder within Cache”.

Once the variables needed for the MUMPS environment are set, press “Configure” again to accept the changes into the environment and then press “Generate” to complete the process.

Variable Name:	Value for Testing in Caché	Value for Testing in GT.M
CTerm	Path to CTerm Executable	-
CControl	Path to CControl Executable	-
EXPECT_EXEC	-	Path to Expect Executable
GTMPROFILE	-	Path to gtmprofile file
VISTA_GLOBALS_DIR	-	Path to Folder containing the database.dat
VISTA_ROUTINES_DIR	-	Path to folder containing Vista Routines
VISTA_Path	Path to Vista Source folder in mgr folder of Caché	-

Figure 78 - Variable settings for Caché and GT.m

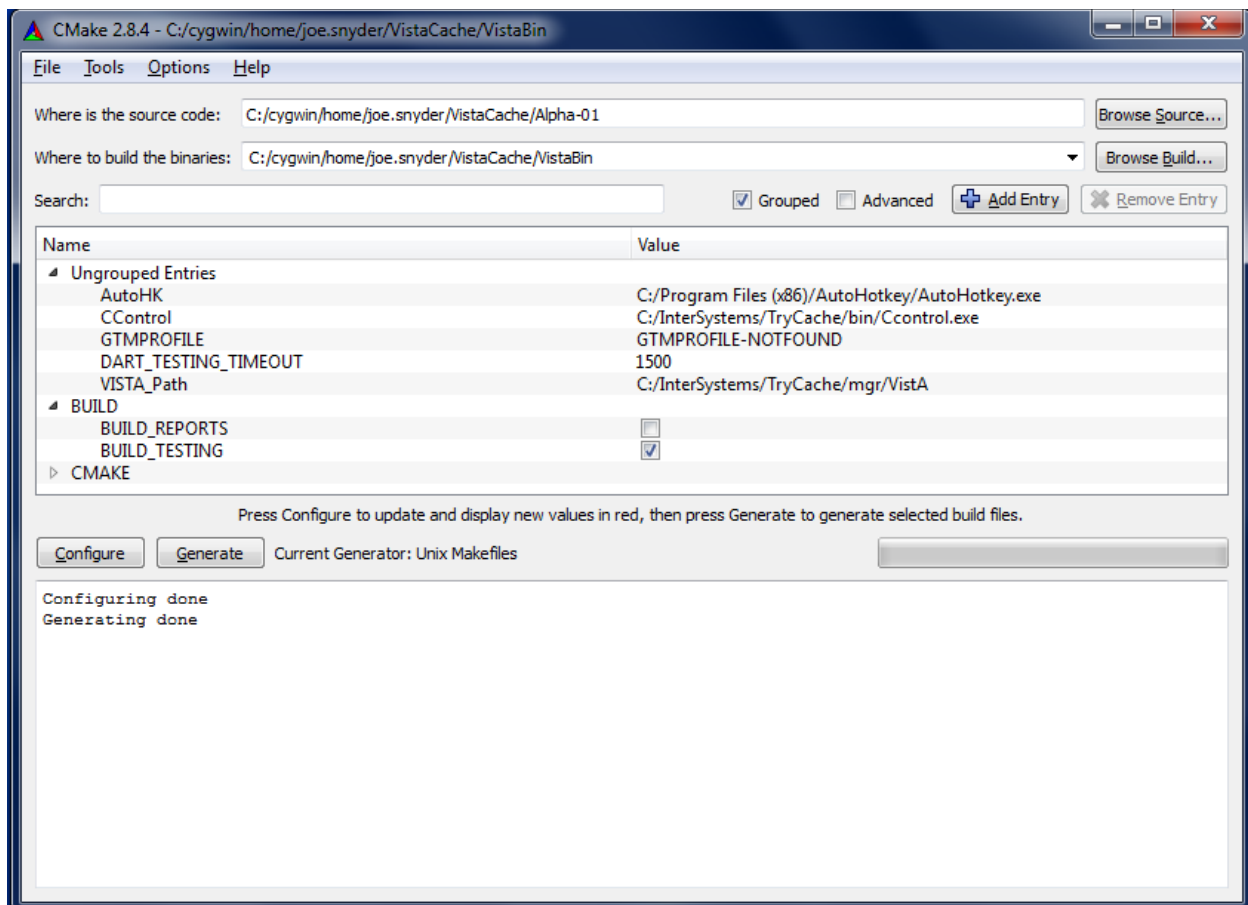


Figure 79 - Setting the VistA variables for GT.M and Linux.

For the Linux example with GT.M (Figure 79), four variables corresponding to EXPECT_EXEC, GTMPROFILE, and VISTA_GLOBALS_DIR, and VISTA_ROUTINES_DIR need to be set or verified according to the values in Figure 78.

There is an option that is not needed to run the testing but may become useful. The CLEAN_CACHE option will show up during configuration on an InterSystems's Caché install. This will utilize scripts to stop the Caché instance, copy in an empty cache.dat, restart the Caché instance, and automatically import the routines and globals from the OSEHRA Code Base. This would all be done during the build phase of a nightly dashboard submission. If you choose not to use this option and have followed the above steps, you can continue to running the tests.

To utilize this option, the CLEAN_CACHE checkbox must also be checked to tell CMake to configure the correct files. You will also need to create a new cache.dat using the steps from earlier (starting at Figure 15) and set the PRISTINE_CACHE_DAT_PATH to point to the location of that newly created cache.dat.

[Return to the Beginning](#)

Running the tests and uploading testing results to the OSEHRA Software Quality Dashboard

The configuration process sets up the test environment and prepares the system to perform the test functions. Once configuration is complete the tests are ready to be run.

Testing is run from a command prompt, such as the *Terminal* in Linux and either the *Cygwin Shell* or the *Windows command prompt* in Windows. From any of these options, changing directory (`cd`) to the **Testing Binary Directory** of your test installation and entering “ctest” will run all the tests. Other ctest options allow specific tests or groups of tests to be run. To see usage information for ctest along with the list of available options enter “`ctest --help`” (Figure 80), which provides usage information.

```
~ /VistaCache/VistaBin
joe.snyder@tuchanka ~
$ cd /home/joe.snyder/VistaCache/VistaBin/
joe.snyder@tuchanka ~/VistaCache/VistaBin
$ ctest --help
ctest version 2.8.4
Usage
    ctest [options]

Options
  -C <cfg>, --build-config <cfg>
    = Choose configuration to test.
  -U, --verbose
    = Enable verbose output from tests.
  -UU, --extra-verbose
    = Enable more verbose output from tests.
  --debug
    = Displaying more verbose internals of CTest.
  --output-on-failure
    = Output anything outputted by the test program
      if the test should fail. This option can
      also be enabled by setting the environment
      variable CTEST_OUTPUT_ON_FAILURE
  -F
    = Enable failover.
  -Q, --quiet
    = Make ctest quiet.
  -O <file>, --output-log <file>
    = Output to log file
  -N, --show-only
    = Disable actual execution of tests.
  -L <regex>, --label-regex <regex>
    = Run tests with labels matching regular
      expression.
  -R <regex>, --tests-regex <regex>
    = Run tests matching regular expression.
  -E <regex>, --exclude-regex <regex>
    = Exclude tests matching regular expression.
  -LE <regex>, --label-exclude <regex>
    = Exclude tests with labels matching regular
      expression.
  -D <dashboard>, --dashboard <dashboard>
    = Execute dashboard test
  -M <model>, --test-model <model>
    = Sets the model for a dashboard
  -I <action>, --test-action <action>
    = Sets the dashboard action to perform
  --track <track>
    = Specify the track to submit dashboard to
  -S <script>, --script <script>
    = Execute a dashboard for a configuration
  -SP <script>, --script-new-process <script>
    = Execute a dashboard for a configuration
  -A <file>, --add-notes <file>
    = Add a notes file with submission
  -I [Start,End,Stride,test#,test#]!Test file!, --tests-information
    = Run a specific number of tests by number.
  -U, --union
    = Take the Union of -I and -R
  --max-width <width>
    = Set the max width for a test name to output
  --interactive-debug-mode [0|1]
    = Set the interactive mode to 0 or 1.
  --no-label-summary
    = Disable timing summary information for
      labels.
  --build-and-test
    = Configure, build and run a test.
  --build-target
    = Specify a specific target to build.
  --build-nocmake
    = Run the build without running cmake first.
  --build-run-dir
    = Specify directory to run programs from.
  --build-two-config
    = Run CMake twice
  --build-exe-dir
    = Specify the directory for the executable.
  --build-generator
    = Specify the generator to use.
  --build-project
    = Specify the name of the project to build.
  --build-makeprogram
    = Specify the make program to use.
  --build-noclean
    = Skip the make clean step.
  --build-config-sample
    = A sample executable to use to determine the
      configuration
  --build-options
    = Add extra options to the build step.
  --test-command
    = The test to run with the --build-and-test
      option.
  --test-timeout
    = The time limit in seconds, internal use only.
  --tomorrow-tag
    = Nightly or experimental starts with next day
      tag.
  --ctest-config
    = The configuration file used to initialize
      CTest state when submitting dashboards.
  --overwrite
    = Overwrite CTest configuration option.
  --extra-submit <file>[!<file>]
```

Figure 80 - "ctest --help" output.

To display the tests that are currently available without actually running the tests, enter "ctest -N."

```
~/.VistaCache/VistaBin
joe.snyder@tuchanka ~/.VistaCache/VistaBin
$ ctest -N
cygwin warning:
MS-DOS style path detected: C:/cygwin/home/joe.snyder/VistaCache/VistaBin
Preferred POSIX equivalent is: /home/joe.snyder/VistaCache/VistaBin
CYGWIN environment variable option "nodosfilewarning" turns off this warning.
Consult the user's guide for more details about POSIX paths:
  http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
Test project /home/joe.snyder/VistaCache/VistaBin
Test #1: ATest
Test #2: BTest
Test #3: CTest
Test #4: DTest
Test #5: ETest
Test #6: FTest
Test #7: GTest
Test #8: HTest
Test #9: ITest
Test #10: JTest
Test #11: KTest
Test #12: LTest
Test #13: MTest
Test #14: NTest
Test #15: OTest
Test #16: PTest
Test #17: QTest
Test #18: RTest
Test #19: STest
Test #20: TTest
Test #21: UTest
Test #22: VTest
Test #23: WTest
Test #24: XTest
Test #25: YTest
Test #26: ZTest

Total Tests: 26
joe.snyder@tuchanka ~/.VistaCache/VistaBin
$ -
```

Figure 81 - "ctest -N" output.

Entering the command "ctest -R <string>" (Figure 82) allows you to specify a test by regular expression match of the string that is passed along with it.

```
~/.VistaCache/VistaBin
joe.snyder@tuchanka ~/.VistaCache/VistaBin
$ ctest -R BTest
Test project /home/joe.snyder/VistaCache/VistaBin
Start 2: BTest
1/1 Test #2: BTest ..... Passed    5.36 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =  5.93 sec
joe.snyder@tuchanka ~/.VistaCache/VistaBin
$
```

Figure 82 - "ctest -R output"

Among the most useful options to ctest is "-D." The command "ctest -D <Configuration>," with <Configuration> set to either *Experimental*, *Nightly*, or *Continuous*, will perform the testing and submit the test results to the OSEHRA Dashboard, <http://code.osehra.org/CDash/index.php?project=OSEHR>. Options can be combined and Figure 83 shows an example of combining the "-D" option for test execution and reporting with the "-R" option for selectively executing a set of tests.

```

C:\. ~/VistaCache/VistaBin
joe.snyder@tuchanka ~
$ cd /home/joe.snyder/VistaCache/VistaBin/
joe.snyder@tuchanka ~/VistaCache/VistaBin
$ ctest -R BTest -D Experimental
cygwin warning:
MS-DOS style path detected: C:/cygwin/home/joe.snyder/VistaCache/VistaBin
Preferred POSIX equivalent is: /home/joe.snyder/VistaCache/VistaBin
CYGWIN environment variable option "nodosfilewarning" turns off this warning.
Consult the user's guide for more details about POSIX paths:
  http://cygwin.com/cygwin-ug-net/using.html#using-pathnames
Site: TUCHANKA
Build name: Win32-
Create new tag: 20110919-2238 - Experimental
Configure project
  Each . represents 1024 bytes of output
  . Size of output: 0K
Build project
  Each symbol represents 1024 bytes of output.
  '!' represents an error and '*' a warning.
  .. Size of output: 2K
  0 Compiler errors
  0 Compiler warnings
Test project /home/joe.snyder/VistaCache/VistaBin
  Start 2: BTest
1/1 Test #2: BTest ..... Passed      4.54 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =  5.65 sec
Performing coverage
Cannot find any coverage files. Ignoring Coverage request.
Submit files (using http)
Using HTTP submit method
Drop site:http://code.osehra.org/CDash/submit.php?project=OSEHR
Uploaded: C:/cygwin/home/joe.snyder/VistaCache/VistaBin/Testing/20110919-2238
/Build.xml
Uploaded: C:/cygwin/home/joe.snyder/VistaCache/VistaBin/Testing/20110919-2238
/Configure.xml
Uploaded: C:/cygwin/home/joe.snyder/VistaCache/VistaBin/Testing/20110919-2238
/Test.xml
Submission successful

joe.snyder@tuchanka ~/VistaCache/VistaBin
$ -
```

Figure 83 - "ctest -R BTest -D Experimental" output.

Currently, the tests are organized differently on the two systems. The Linux testing is robust and fast enough such that each routine can be provided to the XINDEX function individually. Caché testing requires substantially more time and is fragile to parallel execution. Because of this, Caché testing is performed on groups of routines organized by first letter. *i.e.* all routines starting with "A" are tested as a group, followed by all routines starting with "B", and so on through to all routines starting with "Z". Future changes will standardize the testing between the different operating systems.

Note For Linux Users:

The GT.M version doesn't automatically source the gtmprofile for manual testing. It is recommended that you explicitly source the *GTMEnvironment.sh*, found in the *Testing Binary Directory*, from your command line prompt prior to testing. This is created during the configure step, and will set up the proper environment variables for testing.

[*Return to the Beginning*](#)

Reviewing the results

Executing “ctest -D Experimental” or “ctest -D Nightly,” for example, will upload the test results to the OSEHRA Software Quality Dashboard

(<http://code.osehra.org/CDash/index.php?project=Open+Source+EHR>) as shown in Figure 84. Note that the entries are organized by sections for different test types, Nightly or Experimental. The site name is the name of the machine running the tests, and build name is the operating system.

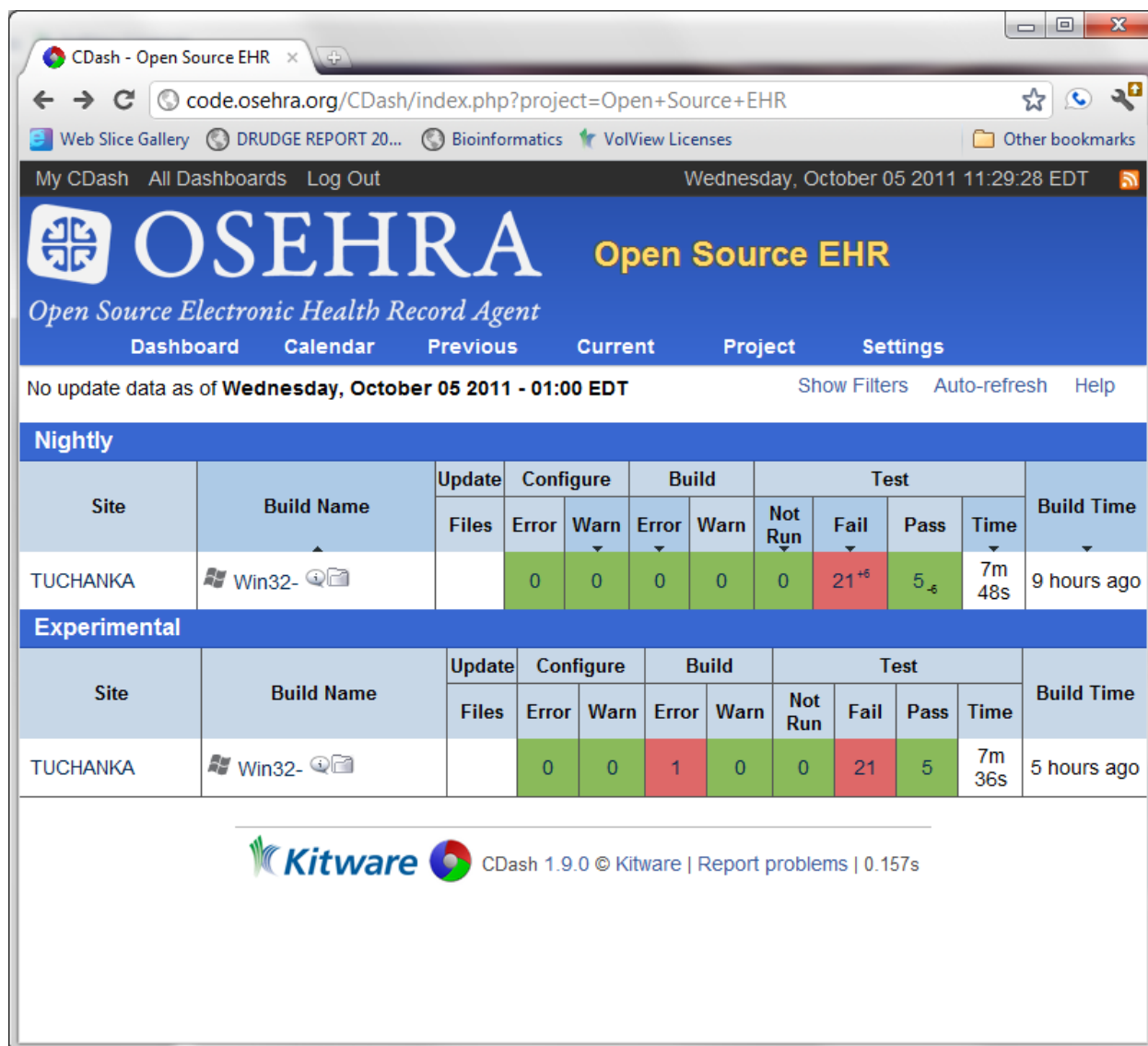


Figure 84 - OSEHRA Software Quality Dashboard

[Return to the Beginning](#)