Razor 6 Scripting

Razor 6 supports scripting on both Unix Servers and Windows Servers. It can run native scripts that were developed for Razor 5 on a Windows Server with minimal modifications and run scripts in existing Razor 5 universes with some modifications required to distinguish where the Razor 6 Server is running.

Special Note for Razor 6 Scripting Integration:

If you are implementing Razor 5 scripts on a Windows Server for use in Razor 6 Git repositories that currently access the following directories for either data or updating files, these will no longer work. All of the following directories are now implemented in either the database or the Git repository.

For Issues, Versions and Baselines (Threads):

1. Any files in the Latest Directory
2. Any files in the Attachments Directory
3. Any files in the History Directory
4. Any files in the Activity Directory
5. Any files in the Info Directory
6. Any files in the Objects Directory

Visible Systems recommends that, for now, during BETA testing you comment out access to these files. Visible Systems will be updating this document on how to access the data that previously resided in these directories and files.

Initial Configuration for Razor 6 Scripting

Windows Server Configuration

Visible Systems recommends a clean install of a Windows Server 2016 or later. Make sure that it is fully patched and Anti Virus software is installed. In all of our testing, we have not had any problems with firewalls but the 2 ports required for Razor 6 are 8888 and the database port number. In most cases the database port number is 3306 but this can be changed by the user and accommodations may have to be made in the firewall.

The recommended size for the primary drive (C:\) is at least 150GB with a second drive for the Git repositories of 50GB or better. RAM, processor numbers do not matter too much so a minimum configuration may be 4GB of RAM and 2 cores. Razor 6 Server is extremely lightweight.

The second drive for the Git repositories, lets use D:\ in this case, should be renamed to something like GitRepositories or Razor6Universes to distinguish it from other drives that may exist on this server. Once this is done navigate to this drive and create a folder

razorScripts

Or some name that denotes where the scripts will reside. If you are going to be using an existing set of Razor 5 scripts that are different in multiple universes you can create multiple folders for each scripting type at this time.

What does this mean?
Let’s say that currently in Razor 5 you have 2 universes, universeA and universeB and UniverseA uses an entirely different set of scripts than universeB. Therefore, on the new Windows Sever, you would create 2 folders where each set of scripts will reside.

Once the server is configured, it is time to install the Administration Tool. Installation of the Administration tool is covered in the Administration tool document (https://resources.visiblesystemscorp.com/razor-6-beta-release) . Make sure that encryption is set up and decide which authorization service is to be used.

Razor 6 on a Unix Server

Configuring a Unix server to run Razor 6 is much the same as configuring Razor 6 on a Windows Server. There are a few cautionary notes that must be mentioned. If you decide to use the same server that is also running a Razor 5 server you must create a separate drive or folder for the Git repositories. Also, you must create the “razorScript” or subfolder in the drive where the Git repositories reside. This is because even though Razor 6 can access and use Razor 5 created universes and scripts the Git side of Razor 6 is unaware of the directory structure of Razor 5 and cannot access these scripts.

Preparing Razor 6 for Scripts

If you are a current Razor 5 user and have an Issue, Versions and Thread form that you wish to use in a new Git universe, then this section is for you. If you are new to Razor 6 there are default forms developed that should meet your short term needs. Visible Systems is in the process of developing a form editing package which will allow you to design a form and import that design to be used in Git universes. This package will be released at a later time.

Importing a Razor 5 configuration (Defaults, Roles, Permissions etc.)

This section discusses how to import your current Razor 5 configurations for use in Razor 6 Git universes. The process is exactly the same for imports into either Unix or Windows.

The first step is to create a directory on the Razor 5 server such as Razor5Imports. It can be anywhere on the server. Next copy either the Wdefaults or Xdefaults file to this location and rename it to defaults.

Next create 3 directories called Issues, Versions and Threads. Now the caveat with the Issues directory is this. Look in the defaults file for a line the looks like this:

```
Issues*IssueFormissue_form1*TitleLabel*x:        0
```

Note that the form name is issue_form1. Now rename the Issues directory created above to issue_form1. This is a very important step to take. Once this is complete, then copy the following files into the issues_form1 directory

1. Attributes
2. Roles
3. Permissions
4. Insensitive
5. Actions
6. Commands

Do the same for Versions and Threads. Note: Bitmaps will be implemented at a later date. For now Visible Systems is providing a default set that may fit your needs at this time. We need to develop a procedure for converting existing bitmap files to a format that can be used by Razor 6.

The next is to use the Razor 6 Administration Tool to import the configuration into the Database. See the documentation on the Razor 6 Administration Tool on how to accomplish this step.

Onto Scripting for Razor 6!

Getting Razor 5 Scripts ready

You can now make changes to your existing Razor 5 scripts to get ready for Razor 6. These changes are needed for Razor 6, but will not impact how your current Razor 5 server will execute them. Doing these updates now, will help ease the conversion to Razor 6 in the future.

The first step is to look at each script and locate any calls to the Razor 5 server. Once you have found one, the changes are as follows:

```
if [[ -z "${RAZOR6}" ]]
then
   requested_date=`razor rz_time_val "$date_required" "%b %d, %Y:%T"`
else
   if [ "${RAZOR6}" == "1" ]
   then
          requested_date=`"${RAZOR_HOME}"/Razor6 Server"/RazorServer.exe rz_time_val
"$date_required" "%b %d, %Y:%T"`
   else
          requested_date=`/usr/bin/mono “${RAZOR_HOME}”/RazorServer.exe rz_time_val
"$date_required" "%b %d, %Y:%T"`
   fi
fi
```

Note: This example uses a sh, ksh or bash shell script

Ok, so what are these changes? First, the new environmental variable is RAZOR6 which is controlled by the Razor 6 server. So if you are running a Razor 5 server, that variable is not set. So the script will use the original Razor 5 call to the Razor Server. If your Razor 6 Server is running under Windows, the server sets RAZOR6 to a “1”, and executes the command against the server on the Windows side. On the other hand if you are running the Razor 6 Server on Unix, the last path is taken.

Remember in the first page you created the razorScripts directory in the Git Universe drive? Now navigate to that directory and create the following sub-directories:

1. The issues directory. Make sure it is the same name as created above when you imported the setting. I.E. issues_form1
2. Next create the versions_tool directory
3. Finally create the baselines_tool directory

Under each of these newly created directories create a directory called Scripts (case matters) especially for Unix.

Now copy your existing scripts into each of their respective directories.

Scripting modification for Git universes.

In most cases, you should have used the standard before and after scripts which call scripts to be run. Therefore each before and after script will need to be modified as follows:

```
$RAZOR_UNIVERSE_DIR/DOMAIN_01/++ISSUES++.issues_form1/Scripts/issues_form1_preload $*
```

to:

```
$RAZOR_SCRIPTS_DIR/task_requests/Scripts/issue_form1_preload $*
```

That’s the first step. Now in each script, the following must be changed to accommodate possible spaces in the directory structure in Windows. Any commands that source commands from the system setup script needs to be modified such as:

```
$nawkexec
```

Change to

```
“${nawkexec}”
```

The quotes and braces are required to handle spaces in the directory name. Also, for debugging purposes the -x added to the heading for #!/bin/sh is honored in Razor 6 and will display a debugging output of the script. Just remember to use exit 11 from within the script.

At this time Visible systems have not seen any problems with scripts that use commands such as echo the sourced system_setup executables.