Z/OS ISPF GIT INTERFACE

Easy Access to Git from z/OS ISPF

Abstract

Git is becoming pervasive for source code management and the z/OS developer has been left out until now. The z/OS ISPF Git Interface (aka zigi) provides the z/OS Developer with a very usable Git Client. And best of all this is Open Source – https://zigi.rocks.

Principle Contributors:
Henri Kuiper – zdevops.com
Lionel B. Dyck - 21stcenturysoftware.com

Table of Contents

Revision History

Version	Date	Description of Revision
0.1	11/11/2019	Creation
0.2	11/17/2019	Minor additions
0.3	11/20/2019	Update panels for v1r1 Add Appendix for git install Add Sort and Reset commands for member list display.
0.4	11/21/2019	Add updated GITLOG information
0.5	11/26/2019	Numerous changes for 1.2
0.6	12/05/2019	Numerous changes for 1.4
1.0	02/20/2020	Major updates for version 2 release 1

Overview (what you should know but may not)

Git is a distributed version control system that has been very popular in the distributed environment and thanks to the great team over at Rocket Software it is now available (has been for a few years) for the z/OS environment. The main drawback to using it has been that there has not been a user-friendly, Mainframe based, solution until now. Zigi allows the Mainframe developer to participate in the world of git. There is a wealth of information available on git, just open your favorite browser and do a search, or go directly to github.com where free accounts are available along with many open source projects such as this one.

Zigi was developed to provide the Mainframe developer with a user-friendly interface to git that works the way any other Mainframe user interface works, making it possible for the developer to continue to use the development tools that are familiar, easy to use, and which enhances their productivity. That means that all of the ISPF capabilities are available to the zigi user as well as the ability to integrate their source code management requirements using git.

Pre-Requisites (installing git)

See Appendix A for a checklist for installing git and the additional tools that it requires – all open source and all for free.

Installing zigi

Using git

1. Create an account at GitHub and add your (Mainframe) SSH Public-Key to your account
2. If you haven't added your SSH keys to your GitHub settings, then provide your username and password at the prompt. It is highly recommended that you add your public SSH key to your GitHub account settings to eliminate the prompts for userid and password.
3. Get into OMVS and change to a directory from where you will be installing zigi.
4. If you've your SSH key already generated, and added to your GitHub account execute:

```
git clone git@github.com:wizardofzos/zigi.git
cd zigi
sh install.sh
```

If you haven't added your SSH key to Github clone via https

```
git clone https://github.com/wizardofzos/zigi.git
[ provide GitHub username and password when prompted]
cd zigi
sh install.sh
```

Should you not have the option to connect from your Mainframe directly to GitHub you can download the zip files, unzip them, upload them to your mainframe, and then run the installer. The latest stable version of zigi can be downloaded from https://github.com/wizardofzos/zigi/

When running the install.sh script there will be a few prompts asking for a high-level-qualifier for the zigi ISPF datasets to be created under. The zigi partitioned datasets will be allocated as PDSE partitioned datasets.

After the install.sh completes, exit OMVS and return to native ISPF.

For more information check out the zigi wiki at https://github.com/wizardofzos/zigi/wiki.

Using CBTTape

You can also download zigi from https://www.cbttape.org in file 997. Be sure to check on the Updates page on the left menu for the latest release, and if you don't see it there then go to the CBT page on the left menu.

1. Download FILE 997 to your workstation and unzip.
2. Binary upload the resulting XMIT file to z/OS using RECFM=FB LRECL=80
3. From TSO, or ISPF Option 6, issue RECEIVE INDS(upload-file-name.xmit)
    a. Enter a dataset name or take the default
4. Open the resulting PDS and execute the $INSTALL member which will receive the EXEC and PANELS members into full partitioned datasets.
5. Edit the PDS member STUB to customize for your site and then copy it into a library in your SYSPROC or SYSEXEC allocations under the name ZIGI.
6. Start zigi from any ISPF Panel by entering TSO %ZIGI.

Restrictions and Caveats

Zigi is designed to work under ISPF, using OMVS services, and interfaces to git. There are some restrictions and/or caveats that you should be aware of:

1. Any RECFM=U dataset is not supported, so no load libraries.
2. VSAM, BDAM, and ISAM are not supported.

Getting Started

To start zigi the first time:

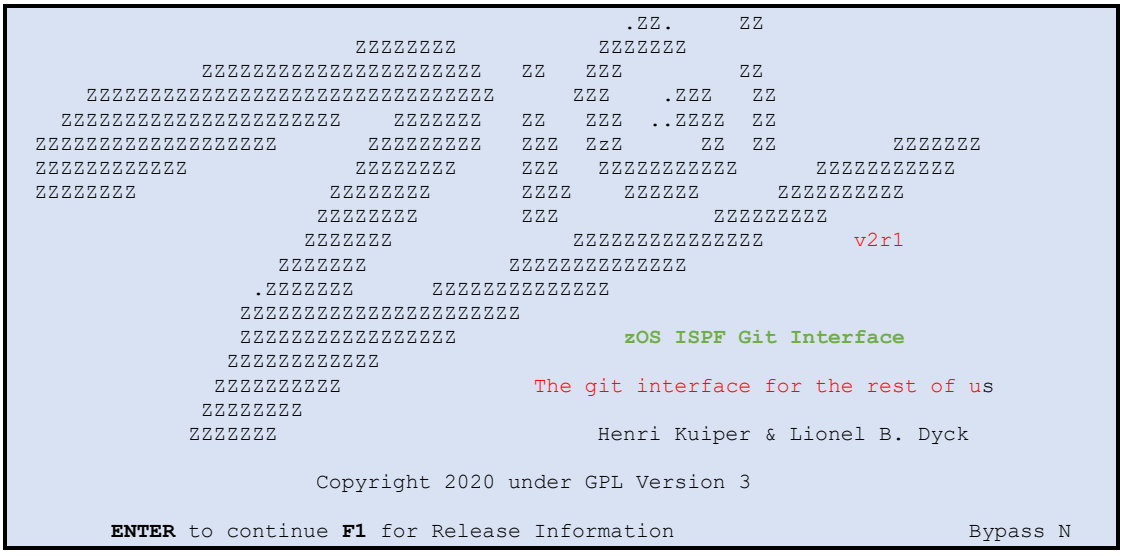
1. In ISPF use option 3.4 and enter the hlq provided above to the installer or enter DSLIST hlq on the ISPF command line.
2. Select the EXEC dataset using Browse, Edit, or View.
3. Note: ZIGI saves its repository ISPF table in the dataset referenced by ISPTABL, but if that DD is not used then it will use the DD ISPPROF as the location for the table.
4. Next to the ZIGI member enter EX to execute it.
    a. This exec is the main zigi code and will dynamically allocate the EXEC library using ALTLIB and the PANELS library using LIBDEF.

```
   Menu  Functions  Confirm  Utilities  Help
 ------------------------------------------------------------------------------
BROWSE            IBMUSER.ZIGI21.ZIGI.EXEC             Row 0000001 of 0000012
Command ===>                                                 Scroll ===> CSR
            Name     Prompt       Size   Created          Changed          ID
_________ GITHELP                 909  2019/12/06  2020/01/27 05:57:00  ZIGI20
_________ ZIGI                   4975  2020/01/18  2020/02/08 01:57:34  ZIGI21
_________ ZIGICKOT                478  2020/01/24  2020/02/04 07:52:14  ZIGI21
_________ ZIGIEM                   56  2019/06/25  2020/01/27 05:54:00  ZIGI20
_________ ZIGIEME                  87  2019/06/25  2020/01/27 05:55:00  ZIGI20
_________ ZIGIEMS                  81  2019/06/25  2020/01/27 05:58:00  ZIGI20
_________ ZIGIGCMD                313  2019/11/27  2020/01/27 05:55:00  ZIGI20
_________ ZIGIMRGM                112  2019/12/10  2020/01/27 05:55:00  ZIGI20
_________ ZIGIOSEL                429  2019/12/10  2020/01/27 05:56:00  ZIGI20
_________ ZIGIRCSI                171  2020/01/23  2020/01/23 05:37:00  ZIGI20
_________ ZIGISTAT                426  2019/11/18  2020/01/27 05:56:00  ZIGI20
_________ ZIGIVMAC                 53  2019/12/16  2020/01/27 05:56:00  ZIGI20
          **End**
```

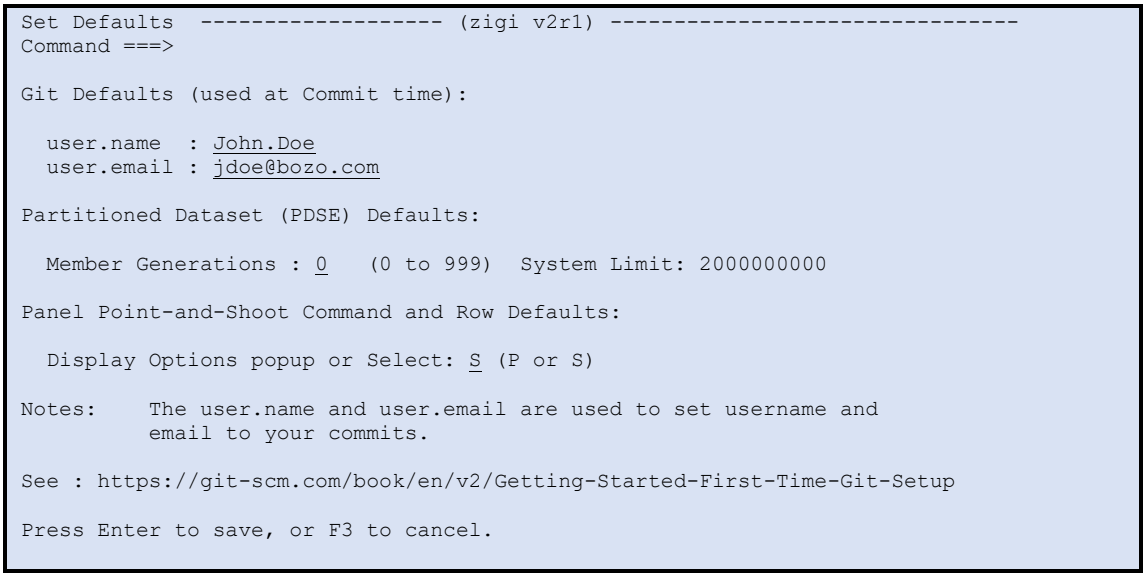
5. A more permanent option is to edit this sample REXX code, make the necessary customizations, and place it in a library in your SYSEXEC concatenation and invoke from any ISPF panel using the command TSO %ZIGI. This sample can be found in the provided EXEC library under the name SAMPLE.

```
/* -------------------- REXX ------------------------------- *
 | This is a REXX exec that may be copied into a system level, |
 | group level, or personal SYSEXEC or SYSPROC library for the |
 | purpose of making it easy to invoke zigi.                   |
 |                                                             |
 | To use customize the exec and panels variables.             |
 * ----------------------------------------------------------- */
 arg opt
 exec   = "'hlq.exec'"                /* <=== update */
 panels = "'hlq.panels'"              /*  <=== update */

 Address TSO 'altlib act application(exec) dataset('exec')'
 Address ISPexec
 'libdef ispplib dataset id('panels') stack'
 "Select CMD(%zigi" opt") Newappl(ZIGI) Passlib Scrname(zigi)"
 'libdef ispplib'
 Address TSO 'altlib deact application(exec)'
```

6. The zigi Splash panel will be presented.



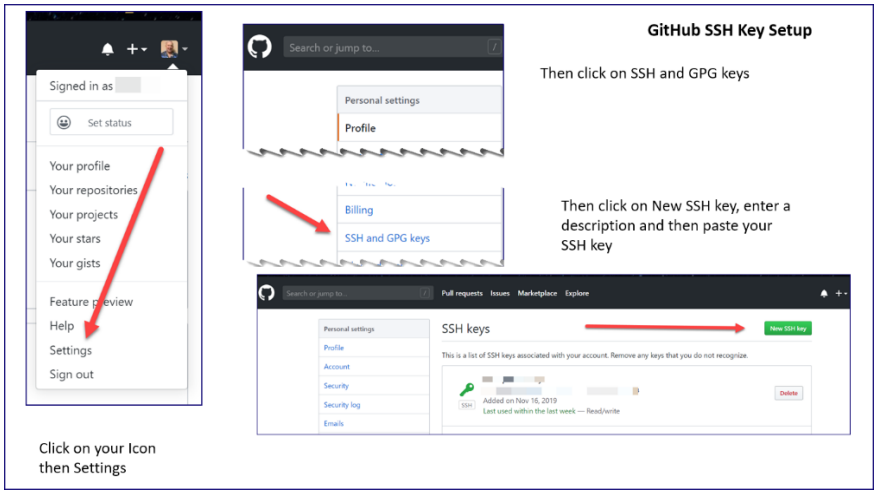
7. Next, the user will be prompted for their username and e-mail along with two zigi processing defaults.
    a. The PDSE member generations default value – it is recommended to use 0 unless you have a tool that supports working with member generations.
    b. The Display Options Popup option will change the behavior of point-and-shoot so that on table panels a click above the table will display the commands popup and anywhere in the table the line selection popup options.



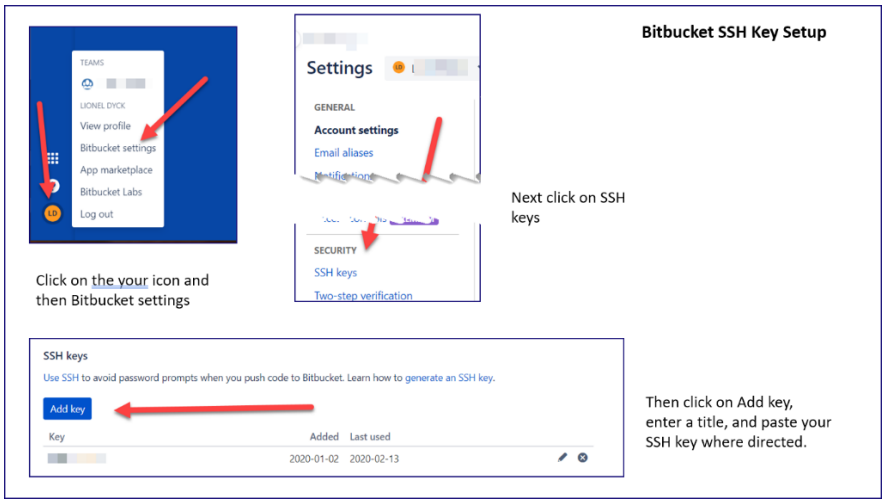
8. After providing your name and e-mail, if you haven't created an SSH key pair yet, zigi will create this for you and will show you the public part of the keypair that you can add to your profile on GitHub (or BitBucket, GitLab or your private enterprise git-server). The public part of the generated key will be shown as follows:

```
--------------------------------- (zigi v2r1) ---------------------------------
 Command ===>

   SSH Key File:  /u/gituser/.ssh/id_rsa.pub

 Copy the records below and paste into Git SSH Key:
Ssh-rsa xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx== developer@ispf.zos



 Press F3 when ready to continue.
```

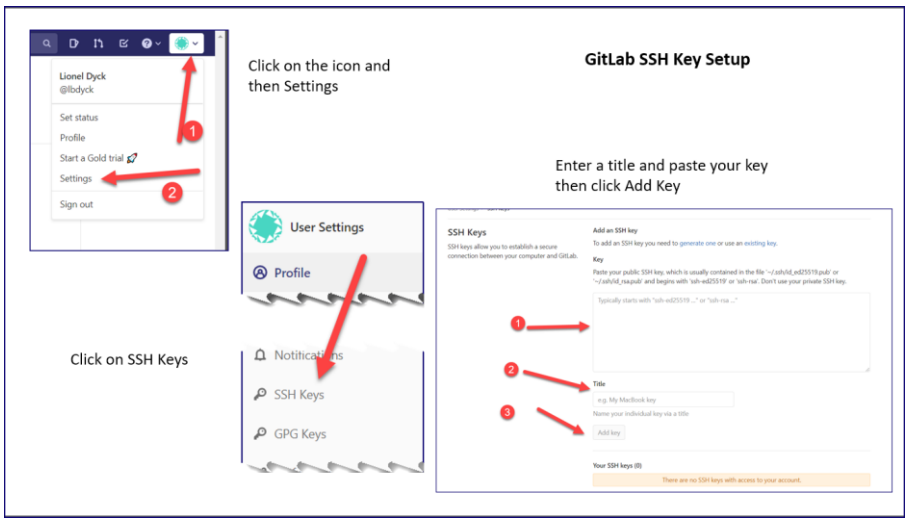
9. Copy the SSH key and paste it into your GitHub (or other) settings under SSH Keys.



Or for Bitbucket:



Or for GitLab:



ISPF Dialog Notes

Many of the ISPF panels support point-and-shoot for commands as well as for row selections. This requires that your TN3270 emulator must support a double click simulating the ENTER key if using the mouse for point-and-shoot. Alternatively, the cursor may be placed in the field or row, and the ENTER key pressed. Changing the ISPF Settings to enable tab to point-and-shoot will also prove useful. The point-and-shoot fields are underscored for easy identification.

The primary table display panels (Local Repository, Current Repository, PDS Member, and Tag List support an ISPF action bar menu. This is in addition to the O command which display a popup menu of commands.

ISPF popups are used to provide information to the user at various points in the dialog where processing may not be instantaneous.

When starting zigi the splash screen will always appear. The splash screen can be disabled by entering a Y to the Bypass option on the splash screen. This will bring up this popup:

```
+---------------------- zigi Status Popup ----------------------+
|                                                               |
|   The splash screen will no longer display per your request.  |
|                                                               |
|   Start zigi with a parm of S to restore.                     |
|   --- Press Enter to continue.                                |
|                                                               |
+---------------------------------------------------------------+
```

The zigi Local Repositories Panel

After the preparatory work above the main zigi ISPF panel will be presented which lists all of the local git repositories that have been added to zigi:

```
   Repository Commands   Help
 ------------------------------------------------------------------------------
 Local Repositories -------------- (zigi v2r1) ---------------- Row 1 to 8 of 8
 Command ===>                                                 Scroll ===> CSR
                                                                          F3
      Repository          Prefix                    Category        Last Access
      ztest               SLBD.ZTEST                testing         20 Feb 2020
      racfadm             SLBD                      tools           19 Feb 2020
****************************** Bottom of data ********************************
```

There are both commands and row selections available for use. Those commands that are **bold** are point-and-shoot fields. That means you can move the cursor to them and press the Enter key or move the cursor there using your mouse and double click with the left mouse button (assuming your TN3270 emulator supports that feature – and most do but you may have to enable that feature).

Action Bar Menu

The action bar menus are there to make it easy to access commands when you forget, or just want to use the mouse for point-and-shoot.

The Repository Commands menu.

```
  Repository Commands   Help
+---------------------------------------+
|    1. Clone Remote repository         |
|    2. Config View/Update Configuration |
|    3. Create Local repository         |
|    4. Find text in table              |
|    5. GIT Help                        |
|    6. Select a Repo to work with      |
|    7. Sort the Table                  |
|    8. View your SSH public key        |
|    9. End                             |
+---------------------------------------+
```

The Help Menu

```
  Help
+------------+
|    1. Help |
+------------+
```

Clone

Clone will prompt to make a copy of a repository that resides on a remote server on your local OMVS filesystem and into a set of z/OS datasets based on a high-level-qualifier that you provide:

```
Clone Repository -------------------- (zigi V2R1) ------------------------------
 Command ===>

 Remote Repository:  URL for reference only
  git@github.com:wizardofzos/zigi.git

 Optional:        :  Specific branch to clone (default is master)


 Local Directory:    ? to browse OMVS folder structure
  /u/gituser/git

 Category for Repository:                (optional)

 PREFIX for datasets:  gituser.zigi                (no quotes)

 Default Push on Commit:   (Y or N)

Default Userid to set prior to Commit:           or blank



                  Press Enter to continue, or F3 to cancel
```

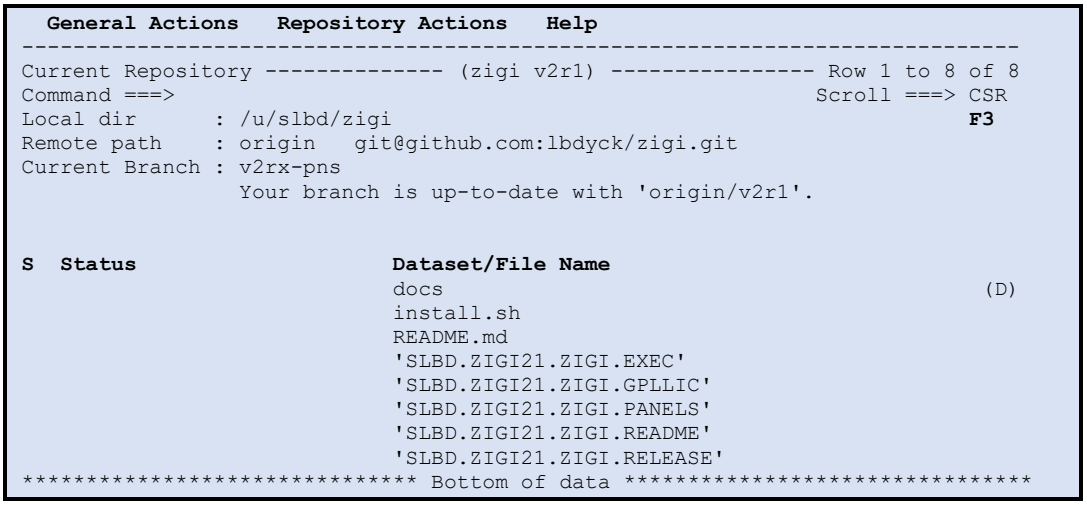
In the above example, the remote repository is the zigi repository on GitHub. We have selected to enter the local OMVS directory manually, and the HLQ (prefix) for the z/OS datasets is specified. If a ? is entered into the Local Directory field then a dialog will be displayed to walk down the OMVS directories to find or make (MKDIR), the directory where the cloned repository will be placed. All but the Optional branch name are required fields.

The Category is helpful as a sort option on the Local Repository table display.

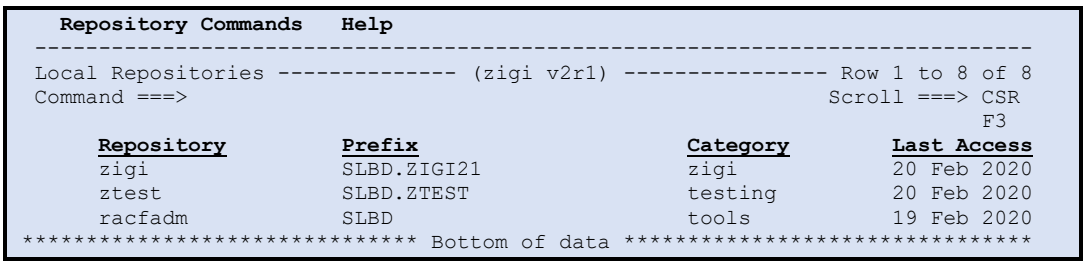
The default Push on Commit option, if set to Y, will be the default on the Commit panel to push after each commit.

The Default Userid is set if there is a requirement to change the ISPF statistics userid for each PDS member before a commit.

After the cloning is complete the z/OS datasets in the repository are displayed. This is also the panel that is displayed when the repository is Selected from the Primary panel. Note that zigi creates PDSE Version 2 libraries for all partitioned datasets with a default MAXGEN of 0 (see the Config command to change the default number of generation).

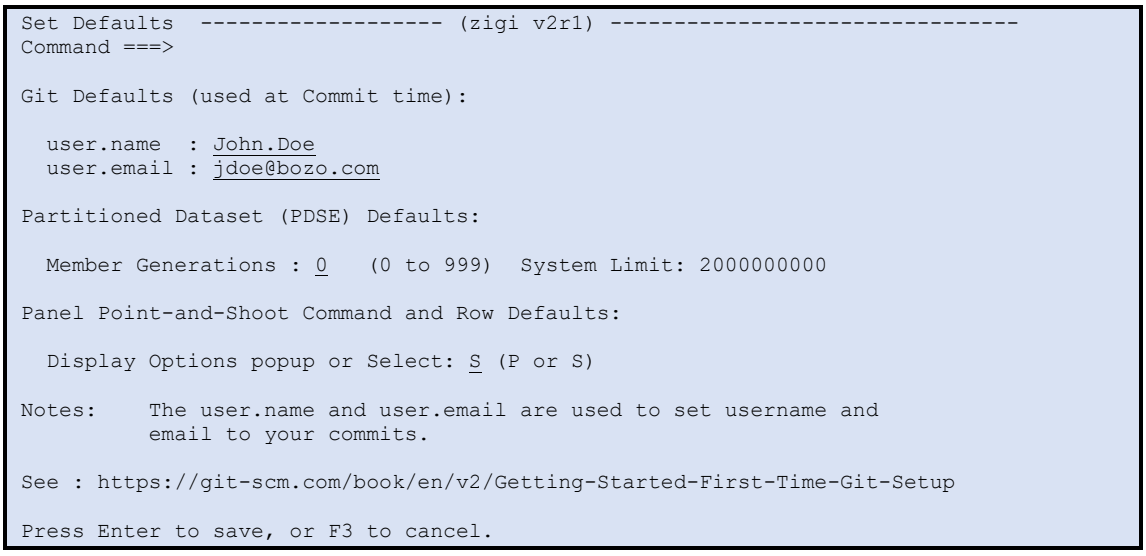


And upon returning to the Primary panel we can see the repository:



Config

This command will allow you to view and update if desired, your username and e-mail, the PDSE member generations default and the display options popup.



In the future, this will also be where other, user-customizable, options will be configured.

Create

Create will create a new local repository that can then later be added to a remote repository.

```
Create New Repository --------------- (zigi V2R1) ------------------------------
Command ===>

Name:         Test


Local Directory:    ? to browse OMVS folder structure
 /u/gituser/test

Default Push on Commit:   (Y or N)

Default Userid to set prior to Commit:           or blank









                  Press Enter to continue, or F3 to cancel
```

In the above example, a test repository will be created in the user's test directory.

The default Push on Commit option, if set to Y, will be the default on the Commit panel to push after each commit.

The Default Userid is set if there is a requirement to change the ISPF statistics userid for each PDS member before a commit.

After the Create process completes the Current Repositories panel will display:

```
  General Actions   Repository Actions   Help
--------------------------------------------------------------------------------
Current Repository -------------- (zigi v2r1) ---------------------------------
Command ===>                                                  Scroll ===> CSR
Local dir      : /u/slbd/test/Test                                     F3
Remote path    : <no remote defined>
Current Branch : master
                 nothing to commit
                 working tree clean

S  Status                    Dataset/File Name
****************************** Bottom of data ********************************
```

Notice that the local directory uses the name of the repository and is case sensitive.

At this point, the ADDDSN command should be used to add existing z/OS datasets to the repository. See below for more on the ADDDSN command, which is not to be confused with the Add row selection.

GITHELP

GITHELP is an ISPF dialog that provides simplified access to the git help documentation. If the user has a screen wider than 80 columns, then an alternate display will be used where there is one row per command.

```
-------------------------------- git manpages ------------- Row 1 to 9 of 146
Command ===>                                                  Scroll ===>

Commands:   Only display only rows with string Refresh after an Only
Selections: S select to view

S  Command / Description
   git
   git manpage
   attributes
   Defining attributes per path
   everyday
   Everyday Git With 20 Commands Or So
   glossary
   A Git glossary
   ignore
   Specifies intentionally untracked files to ignore
   modules
   Defining submodule properties
   revisions
   Specifying revisions and ranges for Git
   tutorial
   A tutorial introduction to Git (for version 1.5.1 or newer)
   workflows
```

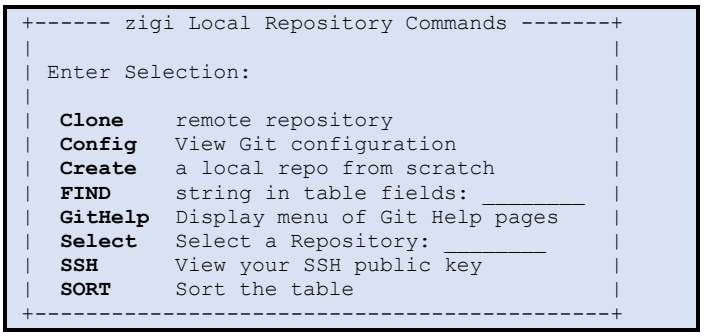
The Only command, entered with a string, will limit the display of available commands to those that match the string in either the command name or the description.

Selecting a command will display that commands man page:

```
BROWSE    SYS19316.T061327.RA000.GITUSER.R0121520     Line 0000000000 Col 001 076
 Command ===>                                          Scroll ===> PAGE
******************************** Top of Data ********************************
GITEVERYDAY(7)                   Git Manual                    GITEVERYDAY(



NAME
       giteveryday - A useful minimum set of commands for Everyday Git

SYNOPSIS
       Everyday Git With 20 Commands Or So

DESCRIPTION
       Git users can broadly be grouped into four categories for the purpose
       of describing here a small set of useful command for everyday Git.

       *   Individual Developer (Standalone) commands are essential for
           anybody who makes a commit, even for somebody who works alone.
. . .
```

Options Menu Assist

Use O on the command line to bring up a popup menu of all available commands. This is useful if the user has selected to use the Pro or Hidden menu.



Select command

For those who don't want to tab to a row, or move the cursor to a row, and use point-and-shoot or enter S to select a repository there is a command option to directly select a repository to be opened.

Syntax: Select repository-name

Select may be abbreviated as S and must be followed by the repository name. For the Select command the search is case insensitive.

Sort

Sort the table based on the columns.

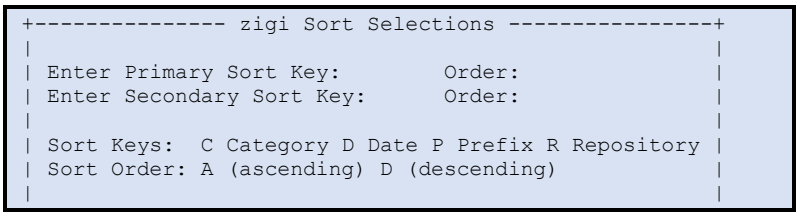
Syntax: Sort column1 order1 column2 order 2

Abbreviation of SO is allowed.

Sort column: Repository (R), Prefix (P), Category (C), or Last (L) or Date (D)

Sort order: Ascending (A) or Descending (D)

If only the command SORT is entered a prompt will appear to assist:



Note that the last sort order is retained and used the next time zigi starts.

SSH

This command will display your SSH public key so that it can be easily copied and then pasted into your GitHub (or equivalent) user profile settings to enable easy access to the remote repository.

```
SSH Key Review-------------------------- (zigi V2R1) ---------------------------------
 Command ===>

   SSH Key File:  /u/gituser/.ssh/id_rsa.pub

 Copy the records below and paste into Git SSH Key:
Ssh-rsa xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx== developer@ispf.zos



 Press F3 when ready to continue.
```

Row Selections

Select option

Selecting a repository using S (or point-and-shoot) will open the repository (see the next section).

View option

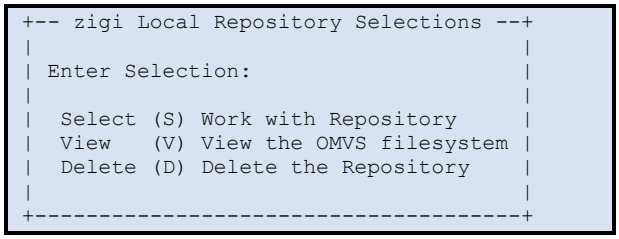
Using V (view) will open the ISPF 3.17 (UDList) on the repositories OMVS filesystem.

Delete option

The Delete option will delete the local repository from the OMVS filesystem while leaving the z/OS datasets untouched.

/ Row Selection Prompt

If / is entered for a row selection the following popup menu will be presented to assist the user:



The zigi Current Repository Panel

When a repository is selected from the Local Repositories panel the z/OS datasets associated with the repository will be displayed:

```
  General Actions   Repository Actions   Help
------------------------------------------------------------------------------
Current Repository -------------- (zigi v2r1) ---------------- Row 1 to 5 of 5
Command ===>                                                  Scroll ===> CSR
Local dir      : /u/slbd/git/racfadm                                      F3
Remote path    : origin   git@github.com:lbdyck/racfadm.git
Current Branch : master
                 Your branch is up-to-date with 'origin/master'.


S  Status                    Dataset/File Name
                             'SLBD.RACFADM.EXEC'
                             'SLBD.RACFADM.MSGS'
                             'SLBD.RACFADM.PANELS'
                             'SLBD.RACFADM.README'
                             README.md
****************************** Bottom of data ********************************
```

For demonstration purposes we will use the zigi repository to start with.

Action Bar menus

The General Actions menu

```
  General Actions
+---------------------------------+
|    1. Add dataset               |
|    2. Find in table             |
|    3. GIT Commands              |
|    4. GIT Help                  |
|    5. Git Log Query             |
|    6. Find String in Repo Files |
|    7. Replace z/OS Datasets     |
|    8. Set Defaults              |
|    9. View OMVS Filesystem      |
|   10. End                       |
+---------------------------------+
```

The Repository Actions menu

```
  Repository Actions   Help
+--------------------------------------+
|    1. Add Modified/Untracked         |
|    2. Change to another Branch       |
|    3. Create Escrow Copy             |
|    4. Commit Changes                 |
|    5. Git Network Report             |
|    6. Git Status                     |
|    7. Merge Branches                 |
|    8. Perform a Git Fetch            |
|    9. Pull from Remote Repository    |
|   10. Push to Remote Repository      |
|   11. Add Remote to Repo             |
|   12. Rollback to Prior Commit Level |
|   13. Git Tag                        |
|   14. List all Tags                  |
|   15. End                            |
+--------------------------------------+
```

And the Help menu is self-explanatory.

AddAll

The AddAll command will add all Modified and UnTracked datasets and files to the Git staging index which makes them ready for the next Git Commit.

AddDsn

AddDsn makes it easy to add additional z/OS datasets to the local repository.

```
Add Datasets -------------------- (zigi V2R1) ---------------- Row 1 to 6 of 6
 Command ===>                                                  Scroll ===> CSR

 Fixed Dataset Prefix:              GITUSER.ZIGI
 Ignore 1st n qualifiers in prefix: 2

Enter a prefix to add datasets under that prefix to the repo via the list
below. Optionally change the number of qualifiers to ignore in the prefix
for the OMVS file name or directory.

Command: Find
Line: S Add  A Add  AB Add binary  B Browse SA Selective Member Add

 S  Dataset Name                                Status
    GITUSER.ZIGI.V1RN.EXEC
    GITUSER.ZIGI.V1RN.PANELS
    GITUSER.ZIGI.ZIGI.GPLLIC                      Added
    GITUSER.ZIGI.ZIGI.README                      Added
    GITUSER.ZIGI.ZIGI.V1R1.EXEC                   Added
    GITUSER.ZIGI.ZIGI.V1R1.PANELS                 Added
 ****************************** Bottom of data ********************************
```

In the above example, four datasets were already in the local repository and two others are available to add.

If the repository is new, that is there are no z/OS datasets included, then the Fixed Dataset Prefix and Ignore 1st n qualifiers will be blank and must be filled in before the list of z/OS datasets may be presented.

The Fixed Dataset Prefix field is the high-level-qualifier (HLQ) for the datasets to be presented, from which one, or more, maybe selected to add to the repository. Be aware that adding a binary dataset using the A, or S, option will add the dataset as text data and not binary. If the dataset contains binary data then add it using the AB (add binary) option.

The Ignore 1st n qualifiers in the prefix field instructs zigi to ignore those qualifiers when creating the OMVS file, or directory, for the dataset. Thus GITUSER.ZIGI.ZIGI.README will be created in the OMVS filesystem as ZIGI.README. Then when someone else clones this repository the file will be uploaded using the HLQ that they provide results in a z/OS dataset of hlq.ZIGI.README. Make sure that the 'ignore count' does not exceed the number of qualifiers of the dataset you want to add.

To clarify:

Local z/OS Dataset name: **HLQ.MLQ.TOOL.JCL**

Set ignore qualifier to 2 and the file added to the local repository will be **TOOL.JCL**

When someone clones with an HLQ of HENRI.REPOS.CLONES the z/OS Dataset Name will be **HENRI.REPOS.CLONES.TOOL.JCL**

Select (S) and Add (A)

These selections will add the dataset as text data.

Add Binary (AB)

Add Binary will add the entire dataset as binary data. This includes tagging the OMVS copy of the dataset with the binary tag (chtag) and updating the .gitattributes file for the file. If the dataset is a partitioned dataset that contains **both** text and binary members then Select or Add the entire dataset and then from the Current Repository display Select the PDS and from the Member list AB the individual members that are in binary format.

Browse (B)

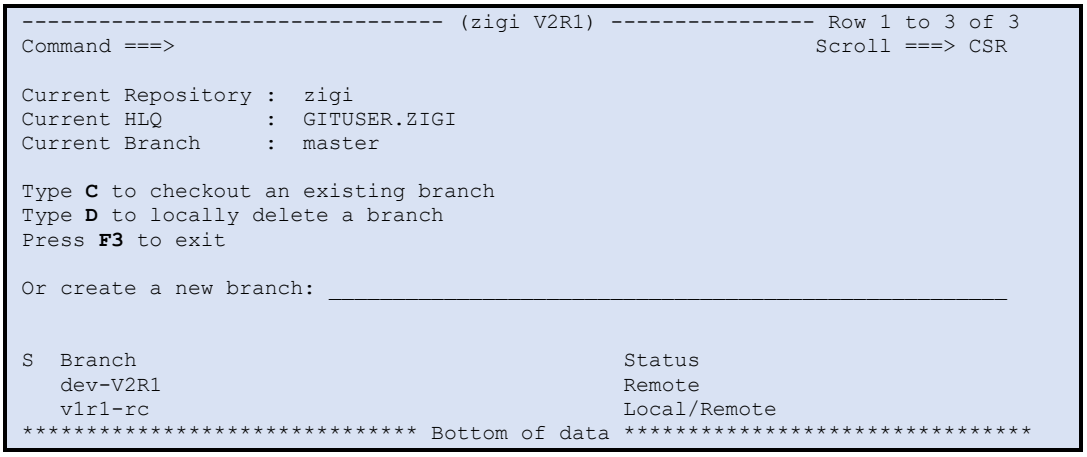
Browse is here to allow the user to verify the data before adding by browsing it using ISPF Browse.

Browse

Browse will browse the member using ISPF Browse

Branch

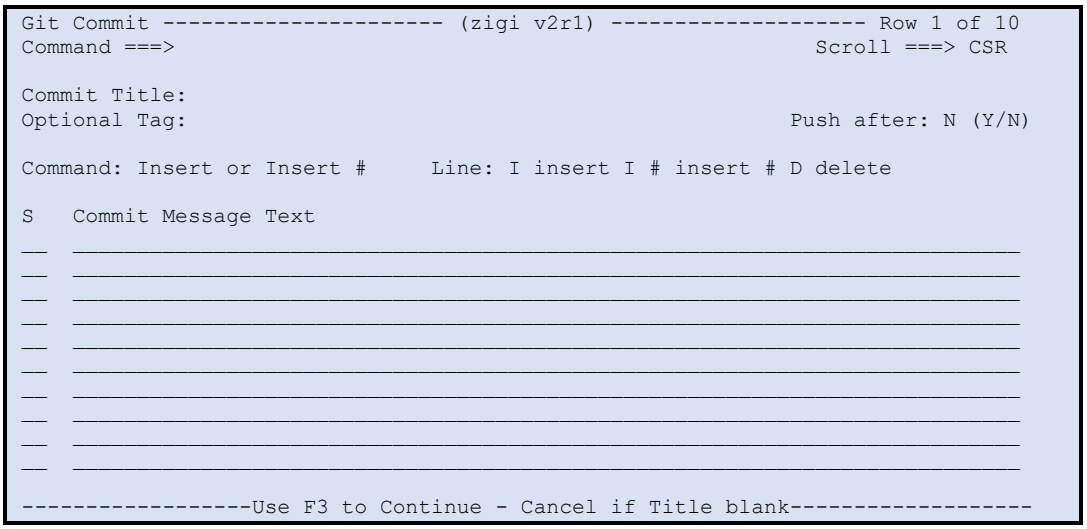
The Branch command allows the user to change to a different repository. If the branch does not exist the user may create it, (effectively performing a **checkout -b**) if it exists the user may check out that branch with the C line-command.



Note the status column indicates if the branch is only on the Remote server, or if there is a copy locally (Local/Remote).

Commit

The Commit command will commit the current set of changes that have been added to the git index to the local repository. This does **not** Push the updates to the remote repository unless the Push after field is set to Y.



Enter a title, which is limited to 50 characters, that summarizes the changes being committed. Then enter as many lines of text as required. Initially 10 lines are available for text entry. However, more can be inserted using the INSERT command (e.g. I 10), or the line command I (e.g. I9). Line may be deleted using the delete (D) line command.

Optionally enter a Tag that will generate a **git tag** command to tag the repository.

GitCmd

Git Command is an ISPF dialog, similar to ISPF Option 6, which makes it easy to enter git commands from within zigi, assuming there is something you want to do that zigi doesn't already do.

```
Git Commands -------------------- (zigi V2R1) --------------------- Row 1 of 7
Command ===>                                                 Scroll ===> CSR

Git repo dir: /home/zdold/git/zigi

Enter any git command (without git):    Browse/View: B (B or V)
git



Hint: Use the GitHelp command to review the available git commands and syntax.

S Command history ( D delete S Select for edit and use X eXecute now)
   log -m                                                                  >
   log -n 3                                                                >
   status                                                                  >
   status && ls -la                                                        >
   branch -a                                                               >
   log --graph --oneline --format="%h %<(80,trunc)%f"                      >
   log --graph --oneline                                                   >
****************************** Bottom of data ********************************
```

The dialog displays better in a wide screen, however if using a standard 80 column display then the command history fields are scrollable (as indicated by the > symbol on the right).

Any git command may be entered, and it will be remembered the 1st time it is used. The history is a push-down stack of previous commands from which you can delete a command you no longer want to keep, select a command to be placed into the command entry field where it can be tailored and executed, or execute a command and have it placed into the command entry field after execution.

The results of all commands are displayed using ISPF Browse or View, depending upon the selection.

Note that git is already in the command, so the user does not have to enter it in the command entry field.

The CLEAR command is available to clear out the table or individual commands in the table may be removed using the **D** selection option.

To execute multiple commands, including OMVS commands, string them together using &&:

```
Enter any Git command (without git):    Browse/View: V (B or V)
git status && ls -la
```

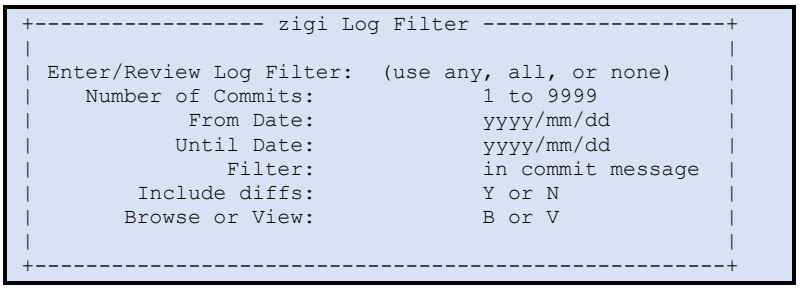
Git Help

See Githelp for more information.

GitLog

This command will display the git log information.

When requested a popup will be presented to request the amount of log information to report:

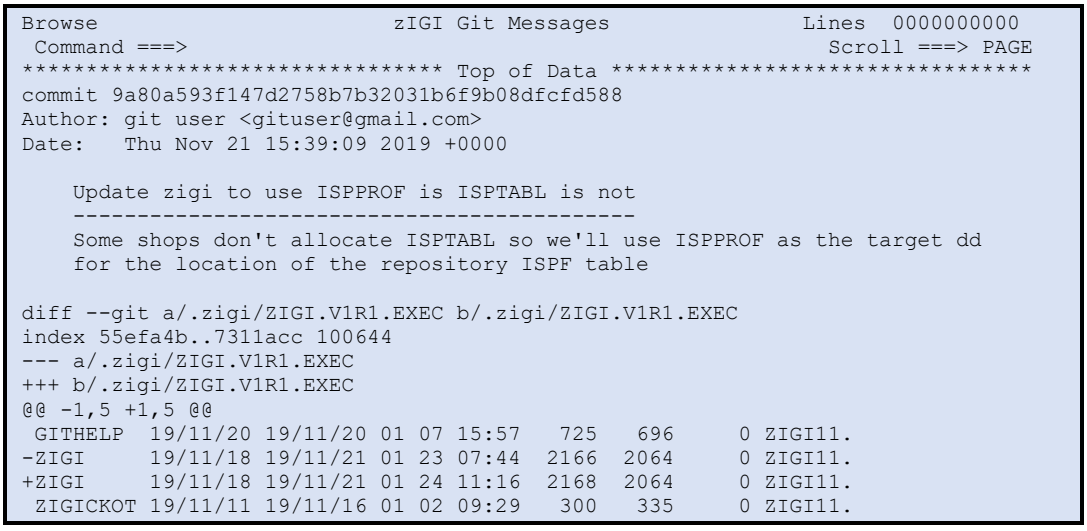


Note that the date format is yyyy/mm/dd which will be translated to yyyy-mm-dd which is required by git. The reason for this is that we are using the ISPF Panel verify for standard dates to verify the date.

The Filter is a string that will be used by grep to filter commits to that string. No blanks allowed.

The values entered will be remembered in the user's ISPF Profile for repeated use.

The git log command will be executed using the --cc and -m options along with the appropriate options for the above values. After which ISPF browse will be invoked on the results:



Grep

The Grep command will prompt for a string and then search the local repository for that string, displaying the results:

```
-------------------------------- (zigi V2R1) -------------------------------
Command ===>


Git grep for :  bpxwunix

Dataset or Report View:   (D or R)

Note: The grep is NOTE case sensitive.









                         Press F3 to exit grep prompt
```

And the results in Dataset view:

```
-------------------------------- (zigi V2R1) ---------------- Row 1 to 5 of 5
Command ===>                                                 Scroll ===> CSR

Grep Command: bpxwunix

Options: S Edit E Edit B Browse V View

Sel      Status     Dataset                                      Count

                    'GITUSER.V12.ZIGI.V1R1.EXEC(GITHELP)'           5
                    'GITUSER.V12.ZIGI.V1R1.EXEC(ZIGI)'             34
                    'GITUSER.V12.ZIGI.V1R1.EXEC(ZIGICKOT)'          1
                    'GITUSER.V12.ZIGI.V1R1.EXEC(ZIGIOSEL)'          4
                    'GITUSER.V12.ZIGI.V1R1.EXEC(ZIGISTAT)'          8
****************************** Bottom of data ********************************
```

The Dataset view will present each of the datasets, or dataset(member), where a hit was detected along with reporting how many hits were found in each. Browse, Edit, or View each.

And the results for the Report view:

```
Browse                       zIGI Git Messages              Lines  0000000000
 Command ===>                                                 Scroll ===> PAGE
******************************** Top of Data ********************************
ZIGI.V1R1.EXEC/GITHELP:  x = bpxwunix(cmd,,so.,se.,env.)
ZIGI.V1R1.EXEC/GITHELP:  | bpxwunix commands.
ZIGI.V1R1.EXEC/GITHELP:  x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/GITHELP:  rc = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/GITHELP:    x = bpxwunix(c,,o.,e.)
ZIGI.V1R1.EXEC/ZIGI:  |                          - Improve bpxwunix performance
ZIGI.V1R1.EXEC/ZIGI:  x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:    x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:    x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:    x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:        x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:          x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:    x = bpxwunix(cmd,,do.,de.)
ZIGI.V1R1.EXEC/ZIGI:    x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:    x = bpxwunix(cmd,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:      x = bpxwunix('rm -rf 'localrep'/'zigirep,,so.,se.)
ZIGI.V1R1.EXEC/ZIGI:    x = bpxwunix(copcm,,so.,se.)
```

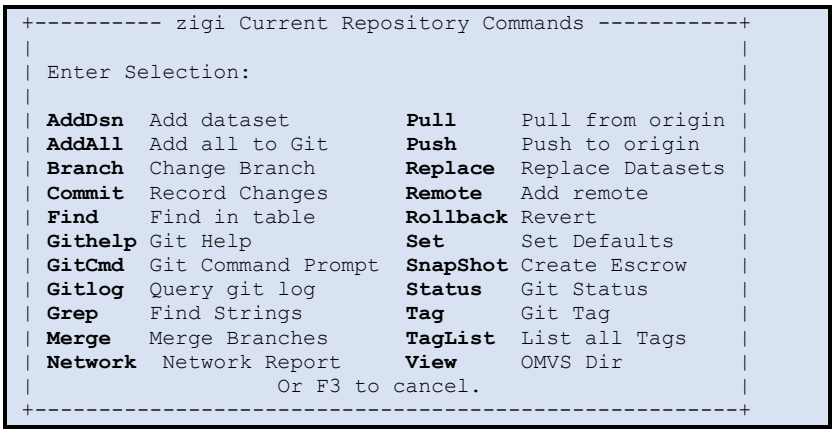
Network

Displays the git log graph report which can be helpful to see the branch structure of the commit history.

```
Browse                       zIGI Git Messages              Lines  0000000015
 Command ===>                                                 Scroll ===> CSR
* ed89496 Remove-Edit-Hiliting-for-merge-conflict-files
* b1e6bd1 Make-Menu-an-alias-of-Basic-and-Pro
* 02799c2 Add-help-panel-for-repository-delete-popup
* 144f713 Update-Release-notes
* c6ad3b1 Remove-extraneous-control-display-save
*   59b007a Clean-from-merge-with-wizardofzos-2.0-branch
|\
| *   8d4c12c Merge-pull-request-2-from-wizardofzos-V2R1-dev
| |\
| | *   7ff201e Merge-branch-V2R1-dev-into-V2R1-dev
| | |\
| | |/
| |/|
| * | 54540ae Update-release-notes-with-more-things-in-2.0
| | * 3bdd616 Weirdest-thing-ever
| | *   0c5941b Merge-remote-tracking-branch-origin-v1r4-rc-into..
| | |\
| | | * d8d88c3 Another-version-bump
| | | * 8cd1afd We-ve-got-version-indicators-everywhere
| | | * 5835aa4 Update-for-version-bump
| | | * 6722416 Update-installer-to-reflect-1.4
```

Options Menu Assist

Use O on the command line to bring up a popup menu of all available commands. This is useful if the user has selected to use the Pro or Hidden menu.



Pull

Pull will copy from the remote repository and replace the updated files in the local OMVS filesystem. After the pull, the updated z/OS datasets will be automatically refreshed.

Push

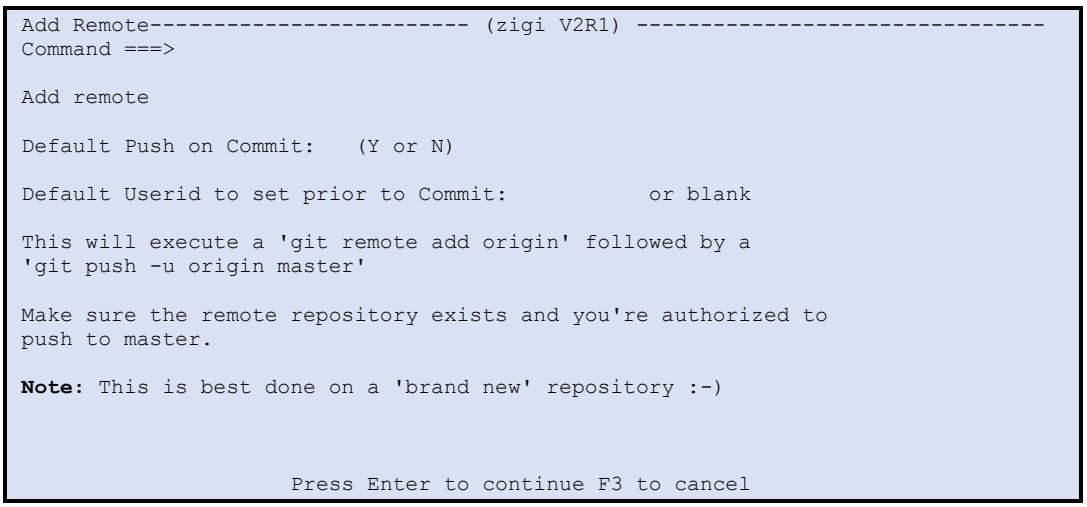
Push will push the current branch to the remote repository.

Replace

Replace will replace all of the current z/OS datasets in the repository with the data in the local repositories OMVS filesystem, including maintaining the ISPF statistics when those datasets were added to the OMVS filesystem. All refreshed partitioned datasets will be allocated as PDSE version 2 libraries with the default MAXGEN. If the refreshed library is a PDS, and more than 25% of the members are being updated, then the PDS will be re-allocated as a PDSE Version 2 with the default MAXGEN. If the refreshed library is a PDSE then it will not be reallocated and only the updated members will be refreshed.

Remote

The Remote command is used for newly created local repositories to be associated with a remote repository.



RollBack

Rollback will provide the user with the option to back out to any commit level. The rollback will be performed in a new branch.

The rollback selection is:

```
Repository Rollback------------ (zigi V2R1) --------------- Row 1 to 7 of 10
Command ===>                                                 Scroll ===> CSR

Commits to Review: 10 (1 to 99)

Commands Only limit table Refresh table
Options: S Display R RollBack

Sel Date/Time            Tag
    Title

    2019/11/26 13:55:45  523e3983730b57eedc934d9837f3eb8868e41d3d
    Translate member with special chars back for use
    2019/11/26 12:31:17  d282c862d50410b33624583bc6a530fddaab9683
    Update to combine messages with multiple A(dds)
    2019/11/26 12:29:48  649032e81fe5634c08249d229cfd24eff82018d2
    Get remote in sync with local
    2019/11/26 12:16:25  2166caf3e31f86e4957ff93fdc02062e6ce8db2f
    Merge branch 'zigi' of github.com:testing/zigi into zigi
    2019/11/26 12:15:59  68511ae012002d29f5e8e90dab2cd60c1c86bd7d
    Update non-zigi clone process
    2019/11/26 06:04:14  1d9e12f784920be034cebfdaac721c4596f14dae
    Delete zigipoph
    2019/11/25 20:45:45  45ca50c8720ac473adff4591c43559c515bf4b14
    Update to non-manage panels (cleanup)
```

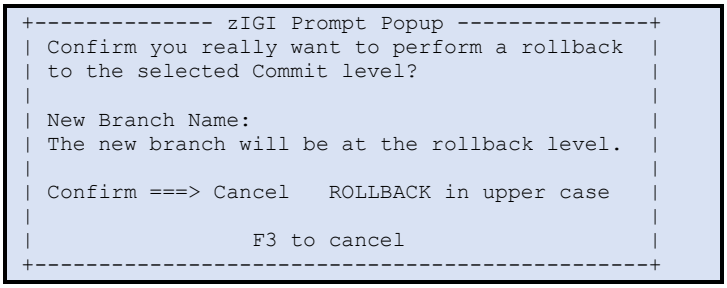
Use ONLY (abbreviation O) followed by a short string to limit the display of items to those that match the string in the commit title.

Use REFRESH (abbreviation R) to restore the full display.

Select the individual commit to view it:

```
View                          zIGI Git Messages              Lines  0000000000
Command ===>                                                  Scroll ===> PAGE
******************************** Top of Data ********************************
commit 523e3983730b57eedc934d9837f3eb8868e41d3d
Author: git user <gituser@company.com>
Date:   Tue Nov 26 13:55:45 2019 +0000

    Translate member with special chars back for use

diff --git a/.zigi/ZIGI.V1R1.EXEC b/.zigi/ZIGI.V1R1.EXEC
index a7003bc..50c3e53 100644
--- a/.zigi/ZIGI.V1R1.EXEC
+++ b/.zigi/ZIGI.V1R1.EXEC
@@ -1,5 +1,5 @@
 GITHELP  19/11/20 19/11/21 01 08 19:06   737   696     0 ZIGI11.
-ZIGI     19/11/18 19/11/26 01 86 08:28  2556  2064     0 ZIGI12.
+ZIGI     19/11/18 19/11/26 01 88 09:52  2563  2064     0 ZIGI12.
 ZIGICKOT 19/11/11 19/11/16 01 02 09:29   300   335     0 ZIGI12.
 ZIGIOSEL 19/11/11 19/11/15 01 03 09:23   418   409     0 ZIGI12.
 ZIGISTAT 19/11/18 19/11/25 01 14 12:31   367   374     0 ZIGI12.
diff --git a/ZIGI.V1R1.EXEC/ZIGI b/ZIGI.V1R1.EXEC/ZIGI
index 1f48a88..64b38ed 100644
```

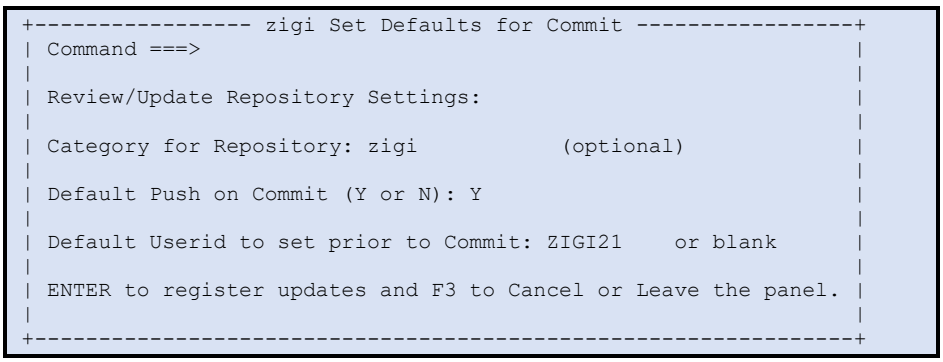
Or select using R to ROLLBACK to that commit level:



This popup requires the name of a new branch and that the word ROLLBACK be entered in UPPER CASE to confirm that you want to do it.

Set

Set is used to define default actions for the user:



The Commit panel has an option to Push after the Commit. This setting provides a default, which the user may over-ride on the Commit panel anytime.

The Userid is what all PDS members ISPF Stats will be set to when they are processed by Commit. Only those members will have their ISPF Stats updated to the specified userid.

NOTE: Do not use 8-character userids unless your system has enabled their use.

Snapshot

Snapshot will create an Escrow set of datasets and OMVS files from the current repository outside of Git. This can then be used for packaging and distribution, as well as an archive or backup for auditing and safekeeping purposes.

```
SnapShot ------------------------ (zigi V2R1) -------------------------------
Command ===>

Enter a git tag to identify this milestone:

Enter New HLQ for the Snapshot datasets:
                                          (no quotes required)

Enter New Path for OMVS Files:

Tag the current level of the repository as a milestone for easy future
reference.

Create a snapshot of the entire repository consisting of all z/OS datasets and
OMVS files. The HLQ must be empty and the path should not exist (it will be
created).

This snapshot may be considered an Escrow copy or a Locked Archive as it is
not manageable by git. Use this when a milestone has been reached such as a
version, release, maintenance level, or patch.


                  Press Enter to proceed, or F3 to cancel.
```

Status

The Status command will display the git status command results:

```
Browse                       zIGI Git Messages              Lines  0000000000
 Command ===>                                                 Scroll ===> PAGE
******************************** Top of Data ***********************************
On branch master
Your branch is up-to-date with 'origin/master'.

nothing to commit, working tree clean
******************************** Bottom of Data ********************************
```

Tag

Tag will add a Git Tag to the current repository and the tag will be pushed to the remote server.

```
+---------- zigi Tag Repository -----------+
|                                          |
| Enter Tag ID:                            |
| Optional Text:                           |
|                                          |
| Press Enter to continue or F3 to Cancel. |
+------------------------------------------+
```

A lightweight tag, one without the optional text, is very much like a branch that doesn't change — it's just a pointer to a specific commit.

Annotated tags, those with the optional text, however, are stored as full objects in the Git database. They're check summed; contain the tagger name, email, and date; have a tagging message; and can be signed and verified with GNU Privacy Guard (GPG). It's generally recommended that you create annotated tags so you can have all this information; but if you want a temporary tag or for some reason don't want to keep the other information, lightweight tags are available too.

TagList

TagList will display a selection table of all the Tags.

```
  Tag Commands   Help
------------------------------------------------------------------------------
Tag Summary  -------------------- (zigi v2r1) --------------- Row 1 to 8 of 14
Command ===>                                                 Scroll ===> CSR
                                                                        F3

Sel Date/Time            Tag
    Title

    2020 Feb 19 13:41:19 v.32
    V32 update ---------- 1) Menu Option 1 (Userid)      -- Display 'SE' line c
    2020 Feb 19 06:38:58 v3.1
    V3.1 Update ---------- - Menu Option 0 (Settings)   * Increased size of 'S
    2020 Feb 19 06:04:03 v3.0
    V3.0 Updates ------------ Backup, delete and copy members... Changes: - Add
    2020 Feb 18 14:01:14 v2.9
    V2.9 Update ---------- Menu Options 1/2/3/4 - When a general user... only
    2020 Feb 18 12:00:42 v2.8
    V2.8 update ----------   - Option 1 (Settings)     -- Changed Administrato
    2020 Feb 18 11:36:01 v2.7
    Minor updates ------------  Option 0 (Settings), re-arranged fields and up
    2020 Feb 18 10:28:39 v2.6
    V2.6 Update ---------- - Option 0 (Settings)    -- Added 'Status Interval'
    2020 Feb 18 06:35:26 v2.4
    Updates ------- - Option 0    -- Capitalized 'Filter' in field description
```

The table may be filtered using the Only command (e.g. only x) and Refresh will recreate the table with all entries.

Row selection of S will display the tag details while selection of C will create a new branch based on the tag level of the repository. This will not only change the OMVS filesystem files to match the tag level file, it will also replace all z/OS datasets to the tag level.

A confirmation popup will ask for the name of a new branch for the tag level code:

```
 +----------- zigi Tag Recovery Prompt -----------+
 | Confirm you really want to create a new branch |
 | from the selected Tag level?                   |
 |                                                |
 | New Branch Name:                               |
 | The new branch will be at the Tag level.       |
 |                                                |
 |       ENTER to continue, or F3 to cancel       |
 +------------------------------------------------+
```

Action Bar menu

```
  Tag Commands   Help
+-------------------------------+
| 1  1. Only - filter the table |
|    2. Refresh the table       |
|    3. GIT Commands            |
|    4. GIT Help                |
|    5. End                     |
+-------------------------------+
```

View

The View command will invoke the ISPF 3.17 (UDList) service on the active local repository filesystem.

Row Selections

Select option

The select option, or point-and-shoot, will perform one of two different actions depending upon the type of dataset being selected.

Edit a sequential dataset. After the edit, if the data has changed, it will be copied down to the OMVS file.

Open an Edit member list for a PDS – see the next section for more information.

If the selected file is an OMVS directory then zigi will open that directory and display it using the Current Repository routines.

Add option

The Add option will add the dataset, if sequential, or any modified members if a partitioned dataset, to the git index which makes them available to be committed.

Browse and View

These options will use ISPF Browse or ISPF View on the requested dataset or file. If the requested dataset or file is an OMVS subdirectory then the action is the same as Select.

Diff

Display the git differences between the current dataset, or file, and the git commit version of the file. See Diff option for more information.

History

View a history of commits for the dataset, or file, with an option to view the source as of the historical point in time. See the History option for more information.

Undo (U)

Undo will revert any changes made to a dataset or OMVS file before a commit. This means the status is [M}, { M], or [MM]. Note that Undo will not work on a full PDS – it will only work on PDS members.

Remove (RM)

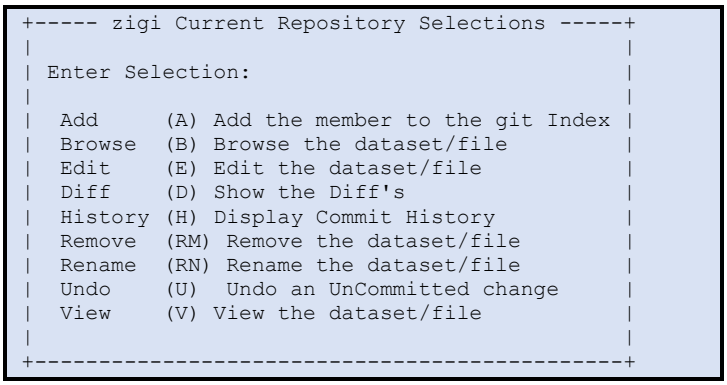
Remove will delete the z/OS dataset, the OMVS file, or the OMVS directory. It will also delete it from git management. This can be undone before, or after, commit processing using Rollback.

Rename (RN)

Rename will rename the z/OS dataset, the OMVS file, or the OMVS directory and update git. Note that git will NOT retain any of the git histories from the original name.

/ Row Selection Prompt

If a / is entered for a row selection the following prompting popup will aid the user:



The zigi PDS Member List

From the Repository display (previous section) if a dataset is a partitioned dataset then the member list will be displayed:

```
  Menu   Help
------------------------------------------------------------------------------
PDS Member List ----------------- (zigi v2r1) -------------- Row 1 to 15 of 17
Command ===>                                                  Scroll ===> CSR
Current Dataset    :  'SLBD.RACFADM.EXEC'                                 F3
Current Repository :  /u/slbd/git/racfadm
Current Branch     :  master

S   Member    Status               Type  Size    Mod-Date  Mod-Time  Userid
    $DIR                            T      24    20/02/16     05:21  RACFADM
    RACFADM                         T      46    20/02/19     05:38  RACFADM
    RACFAUTH                        T     203    20/02/19     13:40  RACFADM
    RACFCHGE                        T      10    20/02/19     13:40  RACFADM
    RACFCHGS                        T      11    20/02/19     13:40  RACFADM
    RACFCLSG                        T     858    20/02/19     13:40  RACFADM
    RACFCLSR                        T     167    20/02/19     13:40  RACFADM
```

Note that the Type column contains a T if the member is a text member and a B if it is binary.

Action Bar menu

```
  Menu   Help
+----------------------------------+
|    1. Add All Modified/Untracked |
|    2. Commit changes             |
|    3. GIT Commands               |
|    4. GIT Help                   |
|    5. GitLog                     |
|    6. Grep - search for string   |
|    7. Locate a Member            |
|    8. Sort the table             |
|    9. Status of the Repository   |
|   10. Reset Userids              |
|   11. End                        |
+----------------------------------+
```

Locate command

The locate command will find a member that starts with or matches the string provided (e.g. L GIT will find the member GITHELP).

Commit command

See the Commit command above.

GitLog command

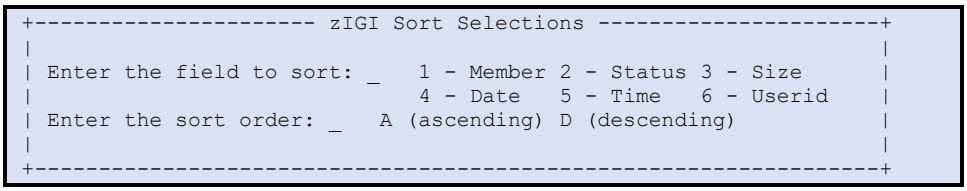
See the GitLog command above.

Grep command

See the Grep command above.

Sort command

The Sort command will sort one column in either ascending (A) or descending (D) order. When entered without parameters a popup panel will assist the user:



The sort syntax is:

SORT field order

If the order parameter is blank, then it defaults to Ascending.

The field names may be abbreviated thus:

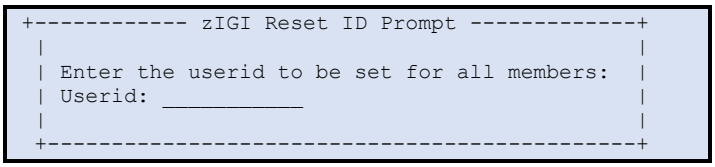
Field	Abbreviation	Field	Abbreviation
MEMBER	ME	DATE	DA
STATUS	ST	TIME	TI
SIZE	SI	USERID	US

Status command

See the Status command above.

Reset-IDs command

This option will prompt to change the userids of all members to a new userid. This is useful to hide the users userid or to make all userids conform to a standard for promoted elements.



Options (O) command

The O command displays a popup command menu to assist those using the terse or hidden menu option.

```
+---------- zigi PDS Member Commands -----------+
|                                               |
| Enter Selection:                              |
|                                               |
|  AddAll    Add all Untracked to Git           |
|  Commit    Register all changes with Git      |
|  Gitlog    Review the Git log                 |
|  Grep      Search the repository for a string |
|  Locate     ________Locate a member           |
|  Reset-IDs Reset all userids                  |
|  Sort      Sort the member list               |
|  Status    Get current Git status             |
|               Or F3 to cancel.                |
+-----------------------------------------------+
```

Member Select option

S, or point-and-shoot, will select the member to be opened using ISPF Edit. If the member is changed during the Edit session, then it will be copied from the z/OS dataset to the OMVS filesystem.

Add option

The Add option will add the member, as text, to the git index making it eligible to be committed.

AddBin option

The AddBin option will add the selected member to git in binary format and will tag the OMVS file copy of the member as binary. The .gitattributes file will be updated with the OMVS directory and file name so that git will always treat it as binary. This is useful for object decks and other data that should never be converted from EBCDIC to ASCII and back.

Browse options

Browse will open the member using ISPF Browse.

Diff option

Compares the active file in the OMVS filesystem with the last index for the file.

```
View                          zIGI Git Messages              Lines  0000000000
 Command ===>                                                  Scroll ===> PAGE
******************************** Top of Data ********************************
diff --git a/ZIGI.V1R1.EXEC/GITHELP b/ZIGI.V1R1.EXEC/GITHELP
index b8477c8..179875b 100644
--- a/ZIGI.V1R1.EXEC/GITHELP
+++ b/ZIGI.V1R1.EXEC/GITHELP
@@ -224,7 +224,7 @@ popup:
  parse arg msg1
  msg2 = 'Patience . . .'
   call pfshow 'off'           /* make sure pfshow is off */
- 'addpop'
+  'Addpop row(4) column(12)'
  'Control Display Lock'
  'Display Panel(gitpop)'
  'rempop'
****************************** Bottom of Data ******************************
```

The diff is displayed using ISPF View that has been enhanced to hilite the additions and deletions in the file. If there have been no changes since the last commit, then there will be nothing to compare.

History option

History shows a selectable list of all commits for the selected member. From this list, the full source from that point in time may be viewed, or the commit with diffs can be viewed:

```
--------------------------------- (zigi V2R1) -------------------- Row 1 of 15
Command ===>                                                  Scroll ===> CSR

Selections: S View historical source C View Commit Info R Recover source

Dataset/File: RACFADM.EXEC/RACFADM

Sel Date/Time of Commit               Author

    Fri 2019 Dec 6  11:43:34 -0600    george washington
    Fri 2019 Dec 6  10:51:38 -0600    george washington
    Sun 2019 Dec 1  10:44:08 -0600    george washington
    Sun 2019 Dec 1  10:35:09 -0600    george washington
    Fri 2019 Nov 22 11:09:53 +0000    george washington
    Wed 2019 Nov 20 19:58:29 +0000    george washington
    Wed 2019 Nov 20 16:32:08 +0000    george washington
    Wed 2019 Nov 20 10:16:35 -0600    John Smith
```

The row selection of / will display a popup to guide the selection of the row command:

```
+----- zigi History Selection Options ------+
|                                           |
| Enter Selection:                          |
|                                           |
|  Commit  (C) View Commit history          |
|  Recover (R) Recover the historical Level |
|  Select  (S) Browse the dataset/file      |
|                                           |
```

Commit view

```
VIEW       ZDOLD.WORK.ZIGI.DATA                           Columns 00001 00072
Command ===>                                                Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 commit 6d9a1af3cbf65a82a98b8c249d28fa1ad6a6dee1
000002 Author: george washington <zigiuser@company.com>
000003 Date:   Sun Dec 1 10:35:09 2019 -0600
000004
000005     Small update if Only result is empty table
000006
000007 diff --git a/.zigi/GITHELP.EXEC b/.zigi/GITHELP.EXEC
000008 index 4462f5b..56be5c9 100644
000009 --- a/.zigi/GITHELP.EXEC
000010 +++ b/.zigi/GITHELP.EXEC
000011 @@ -1 +1 @@
000012 -GITHELP  19/11/20 19/11/22 01 08 07:09   737   696     0 ZDOLD
000013 +GITHELP  19/11/20 19/12/01 02 99 12:34   738  2696     0 GITHELP
000014 diff --git a/GITHELP.EXEC/GITHELP b/GITHELP.EXEC/GITHELP
000015 index 67d8500..4407c41 100644
000016 --- a/GITHELP.EXEC/GITHELP
000017 +++ b/GITHELP.EXEC/GITHELP
```

This view can be used to see the specific changes that were made with that specific commit.

Source view

```
VIEW       GITUSER.WORK.ZIGI.DATA                          Columns 00001 00072
Command ===>                                                   Scroll ===> CSR
****** **************************** Top of Data ***************************
==MSG> Historical view of element: ZDOLD.GITHELP.GITHELP.EXEC(GITHELP)
==MSG> From: george washington on Sun 2019 Dec 1  10:35:09 -0600
==MSG> All ISPF View commands are available, including Compare, Create and Copy
==MSG> -------------------------------------------------------------------------
000001   /* ------------------  rexx procedure  -------------------- *
000002   | Name:      GitHelp                                        |
000003   |                                                           |
000004   | Function:  Display a menu (table) of git commands and     |
000005   |            allow the user to view the selected help,      |
000006   |            filtering using Only xxx and a Refresh         |
000007   |            options are provided.                          |
000008   |                                                           |
000009   | Syntax:   %Githelp                                        |
000010   |                                                           |
000011   | Usage Notes:                                              |
000012   |       1. git must be installed on the local OMVS          |
```

This view of the source is from that specific commit. It is placed into a z/OS dataset to allow the user to use the ISPF Edit Compare, Create, and Copy commands, among others.

Recovery

The historical recovery will prompt for a recovery dataset if that dataset should be allocated new, and then a confirmation of the recovery. Recovery can occur into the active dataset if desired but with an additional confirmation prompt.

```
+------------------- zigi History Recovery --------------------+
|                                                              |
| Recover a historical level of:                               |
| 'IBMUSER.ZIGI20.ZIGI.GPLLIC'                                 |
|                                                              |
| Enter the recovery dataset:                                  |
| 'IBMUSER.ZIGI20.ZIGI.GPLLIC'                                 |
|                                                              |
| Allocate Recovery dataset if it does not exist: N (Y or N)   |
|                                                              |
| Confirm Recovery:   (Y or N)                                 |
|                                                              |
| Press Enter to continue or F3 to Cancel.                     |
+--------------------------------------------------------------+
```

Remove and Rename

See the previous discussion about Remove and Rename which applies, in this case, just to the PDS Member.

Undo

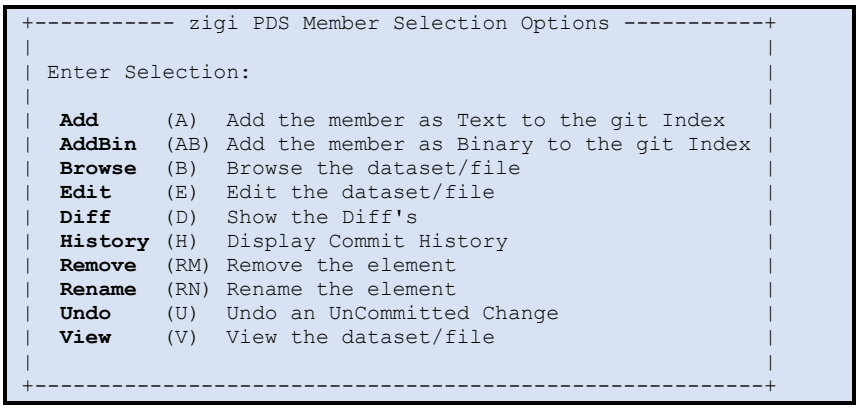
Undo will revert any changes made to a PDS member before a commit. This means the status is [M}, { M], or [MM].

View

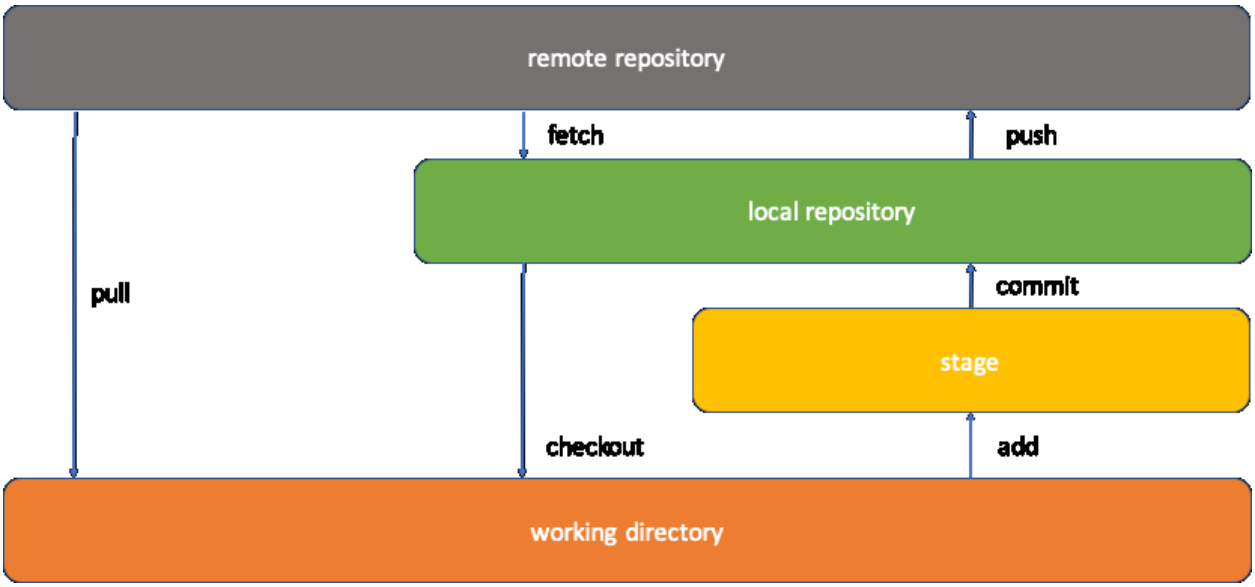
View the dataset or file using ISPF View.

/ Row Selection Prompt

If a / is entered for a row selection a popup prompt menu will aid the user:



Git status cheat sheet ☺



Typical zigi Activities

Zigi is intended to be used as the interface to git. All development activity is anticipated to be performed outside of zigi using your favorite ISPF based tools. Once a development phase is complete then zigi can be activated to commit and push the update.

Creating a new Repository (Local and Remote)

There are two ways of creating a (git-based) repository with zigi. Cloning (see further) a remote repository or creating one "from scratch" locally. Via the 'create' command a new (local) repository van be created. This will perform a 'git init <repoName>' in the chosen directory. A 'basic' .gitattributes will be created and added/committed to your repository. This is there to make sure that the encoding of the various git-managed files will be in the correct format.

After creating the local repo (remember zigi takes care of your .gitatrtibutes file) you can ADD datasets to the repository. (see the Add command on page xxx).

Adding files to zigi is **not the same thing** as adding them to the git-repository. Once a dataset has been added to zigi the datasets (in file-format) are *only* available in the working-space of the git repository. (they are present in /repofolder/reponame). Adding to the git-repo is done via the 'A' (add) line command.

Adding a Remote Repository

zigi currently only supports a single 'remote' (named origin). Once you have a local repository you can use the 'remote' command to add this (single) remote to your local repository. To do so, the remote repository (GitHub, BitBucket, GitLab, etc.) needs to associate the public part of your (Mainframe) ssh-key-pair with the userID for the remote repository service.
Once you've set up the repository on the remote side, you can add the remote within zigi. A "push –u origin master" will be performed.

Updating a Local Repository

Need prose

Pulling Updates from the Remote Repository to Update the Local Repository

Need prose

Pushing Updates to the Remote Repository

Need prose

Creating a New Branch

Need prose

Changing to a Different Branch

Need prose

What is Happening Under the Covers?

When zigi opens a repository the first action is to check the remote server for updates. If there are updates, then the user will be informed that the current branch is behind and needs updating (pull). Zigi also checks all of the z/OS datasets and the OMVS filesystem for updates, marking the z/OS datasets if there are modifications in play that need to be added to the index, committed, and

pushed. If the dataset is a sequential dataset, it is always copied to an OMVS file so that Git can detect changes. For a partitioned dataset, if the member is new then it will be copied to the OMVS directory that maps to the PDS.

When a partitioned dataset is updated, the ISPF statistics for all members are captured and an OMVS file is updated to accurately reflect the current statistics. This file is then used to compare the members to identify if there are new, or changed, members and then flag them as such. The file is also used during a replace, clone, or pull, operation to reset the ISPF statistics as git has zero awareness of them.

As datasets are added to zigi, an OMVS file is updated with the DCB information for that dataset. This file is then used during a replace, clone, or pull, operation to correctly allocate the z/OS datasets. The space for those datasets is dynamically determined based on the OMVS filesystem usage and are allocated as PDSE version 2 libraries with a zero MAXGEN.

Timing (for those interested)

For those who may be interested in some of the internal timings it is as easy as allocating a DD with the name of ZIGIDEBG. Then when zigi runs, time stamps for various routines and functions will be recorded in a file in the user's home directory. Then when zigi ends, if the DD was allocated, the file will be opened in Browse:

```
BROWSE    /u/giruser/zigidebug.txt                  Line 0000000000 Col 001 059
 Command ===>                                                 Scroll ===> CSR
******************************** Top of Data ********************************
13 Nov 2019 08:01:02 : Starting collection of env variables
13 Nov 2019 08:01:16 : Finished collection of env variables
13 Nov 2019 08:01:32 : start of update_repo_metadata
13 Nov 2019 08:01:33 : fetching remote
13 Nov 2019 08:01:34 : getting the remotes
```

For those interested here is a short EXEC that will allocate and/or free the ZIGIDEBG DD:

```
  /* ------------------- REXX ---------------------- *
  | A toggle exec that will allocate the ZIGIDEBG DD |
  | if it does not exist and if it does then it will |
  | be freed.                                        |
  * ------------------------------------------------ */
  x = listdsi('ZIGIDEBG' 'FILE')
  if x > 0 then if sysreason = 3 then 'Free F(ZIGIDEBG)'
  else 'Alloc f(zigidebg) dummy reuse'
```

Appendix A – Installing git

Check	Action
	Need an OMVS user account with SuperUser (su) capabilities (for installing git 'systemwide')
	Need to allocate a good size ZFS (one that we are using is 500-600 cylinders)
	Access to PARMLIB member BPXPRMxx or someone who can update it for you to add the appropriate MOUNT statement for the new ZFS
	Download and install the git package from Rocket Software. Jump on over to https://www.rocketsoftware.com/product-categories/mainframe/git-for-zos and get started. Be sure to download git, bash, gzip and perl and bring all those files to one directory in USS. • Per the git README.ZOS you may also need to download: unzip and vim.
	Then head on over to https://gist.github.com/wizardofzos/897b243d4cbe9fbc471ec1396fbbe174 and download that installer into the same directory as the things you downloaded from rocket.
	Upload the files to a ZFS on z/OS. This ZFS does not have to be the same as the intended ZFS where git, et.al., will be installed. This may be your personal home directory if the filesystem is large enough (or can grow enough).
	Mount the above-created ZFS to a mount point. This is the target for git, and related, tools.
	Get into OMVS
	Go to the directory where the installer was placed and where the git packages were uploaded to.
	Run the installer script and when prompted provide the path to the mounted ZFS that will be home to git and tools
	Review the installation.
	Find the **environment.sh** file and copy the contents into your **/etc/profile** file. **WARNING**: If you installed into a path that will not be the production path then be VERY CAREFUL as you will have to edit the environment.sh values before placing into /etc/profile. Suggest also adding to the profile **export GIT_PAGER=more** unless you have installed a ported version of less.
	Exit OMVS and get back into OMVS and enter the command **git version** – if it works you've got git.
	When happy run: 1) The remove_dist.sh script to rm the downloaded files 2) The uninstall.sh to: a) Remove the environment.sh b) Remove the remove_dist.sh c) Remove other installation files
	Now unmount the git and tools zfs from the temporary mount point.
	Create a new, permanent mount point and mount it.

	Update PARMLIB BPXPRMxx with the MOUNT statement so that it will be mounted at each IPL:
	`MOUNT FILESYSTEM('OMVS.GIT.ZFS')`
	`TYPE(ZFS)`
	`MODE(READ)`
	`MOUNTPOINT('/usr/ported')`