

# **z/OS ISPF Git Interface: Easy Access to Git from z/OS ISPF User Guide**

**Software Version: 3.10 | Revision Number: 2.1 | January 2021**

# Contents

<b>About ZIGI.....</b>	<b>5</b>
<b>Revision History.....</b>	<b>5</b>
<b>Overview (What You Should Know but May Not).....</b>	<b>7</b>
Pre-Requisites (Installing Git).....	7
<b>Installing ZIGI.....</b>	<b>8</b>
Using Git.....	8
Additional Options.....	9
Using CBTTape.....	9
<b>Dovetail Enhancements.....</b>	<b>9</b>
<b>Restrictions and Caveats.....</b>	<b>10</b>
<b>Starting ZIGI the First Time.....</b>	<b>10</b>
<b>ISPF Dialog Notes.....</b>	<b>15</b>
<b>The ZIGI Local Repositories Panel.....</b>	<b>15</b>
Action Menu Bar.....	16
The Repository Commands Menu.....	17
The Help Menu.....	17
Clone.....	17
Config.....	19
Connect.....	20
Create.....	20
GITHELP.....	23
Options Menu Assist.....	24
Select Command.....	24
Sort Selections Pop-Up Menu.....	24
SSH Public Key.....	25
Row Selections.....	26
Select Option.....	26
Info Option.....	26
View Option.....	26
Delete Option.....	26
/ Row Selection Prompt.....	27
<b>The ZIGI Current Repository Panel.....</b>	<b>27</b>
Action Bar Menus.....	29
The General Actions Menu.....	29
The Local Repo Actions Menu.....	29
The Remote Repo Actions Menu.....	30
The Handy Functions Actions Menu.....	30
The Help Actions Menu.....	30
AddAll Command.....	31
AddDsn Command.....	31
Select (S) and Add (A) Commands.....	33
Add Binary (AB) Function.....	33
Browse (B).....	33
Browse.....	33
Branch Command.....	33
Check Command.....	34
Commit Command.....	34

Convert Repository (CONVREPO) Command.....	34
DIFF Command.....	35
Extract Command.....	36
Flow Command.....	38
GitCmd Command.....	39
Git Help.....	40
GitLog Command.....	41
Grep Command.....	41
Network.....	43
Options Menu Assist.....	44
Pull.....	44
Push.....	44
Replace.....	44
Remote Command.....	44
Set Command.....	45
Snapshot.....	46
Stash.....	46
Stash List (STASHL) Command.....	47
Branch.....	47
Diff.....	47
Pop.....	48
Pop Conflict Resolution.....	48
Remove.....	48
Show.....	48
Status Command.....	48
Tag.....	48
TagList.....	49
View.....	50
Row Selections.....	50
Select Option.....	50
Add Option.....	51
Browse and View.....	51
Diff.....	51
History.....	51
Undo (U).....	51
Remove (RM).....	51
Rename (RN).....	51
/ Row Selection Prompt.....	51
<b>The ZIGI PDS Member List.....</b>	<b>52</b>
Action Bar Menu.....	53
Locate Command.....	53
Commit.....	54
GitLog.....	54
Grep.....	54
Sort Command.....	54
Status.....	54
Reset-IDs Command.....	55
Options (O) Command.....	55
Member Select Option.....	55
Add Option.....	56
AddBin Option.....	56
AddForce (AF) Option.....	56
Browse Options.....	56
Diff Option.....	56
History Option.....	57
Commit View.....	58

Source View.....	58
Recovery.....	59
Ignore.....	59
Remove and Rename.....	59
Undo.....	59
View.....	60
/ Row Selection Prompt.....	60
<b>Typical ZIGI Activities.....</b>	<b>60</b>
Creating a New Repository (Local and Remote).....	61
Adding a Remote Repository.....	61
Updating a Local Repository.....	61
Pulling Updates from the Remote Repository to Update the Local Repository.....	61
Pushing Updates to the Remote Repository.....	61
Changing to a Different Existing, Or New, Branch.....	62
<b>What is Happening Under the Covers?.....</b>	<b>62</b>
<b>ZG - ISPF Command.....</b>	<b>63</b>
<b>ZGBATCH - Execute ZIGI in Batch.....</b>	<b>65</b>
<b>Timing (For Those Interested).....</b>	<b>66</b>
<b>Appendix A: Installing Git.....</b>	<b>67</b>
<b>Appendix B: Using ZGINSTALL.REX.....</b>	<b>68</b>
<b>Appendix C: Learning Resources.....</b>	<b>69</b>
<b>Appendix D: User Exits.....</b>	<b>69</b>
<b>Appendix E: Common Problems.....</b>	<b>70</b>

# About ZIGI

---

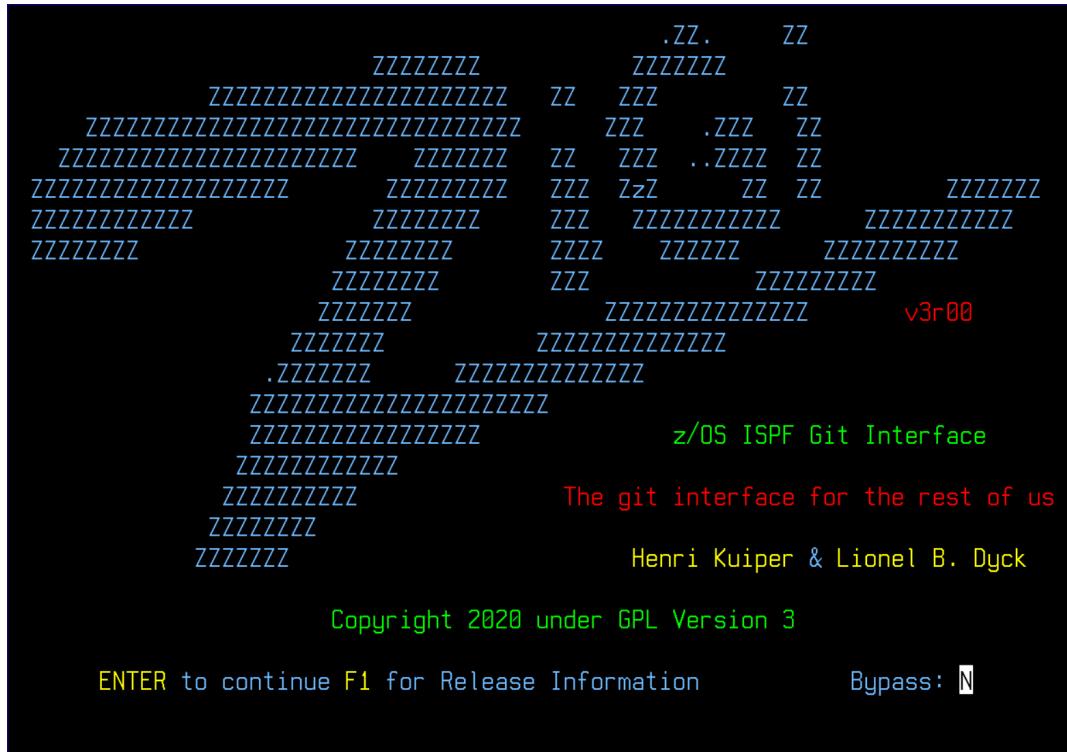
Git is becoming pervasive for source code management (SCM) and the z/OS developer has been left out until now.

The z/OS ISPF Git Interface (AKA ZIGI) provides the z/OS developer with a very useable Git client.

ZIGI is open-source and can be found at <https://ZIGI.rocks>.

**Author:** Henri Kuiper ([henri@zdevops.com](mailto:henri@zdevops.com)): [zdevops.com](https://zdevops.com)

**Author:** Lionel B. Dyck ([lbdyck@gmail.com](mailto:lbdyck@gmail.com))



## Revision History

---

The revision history for ZIGI and the *z/OS ISPF Git Interface: Easy Access to Git from z/OS ISPF User Guide* is listed below. The most recent update is listed first.

Version	Date	Description of Revision
1.14	01/11/2021	Updated Local Repo Actions Menu image. Updated AddDsn Command topic. Added DIFF Command topic.
1.13	12/05/2020	Underwent full editing pass by tech writer. Added Dovetail Enhancements Topic, Connect topic, AddForce Option topic, Ignore Option topic, and Appendix D: User Exits.

<b>Version</b>	<b>Date</b>	<b>Description of Revision</b>
		Updated images. Changes to the Create topic, Delete Options topic, and AddDsn Command topic. Renamed Appendix D: Common Problems to Appendix E: Common Problems.
1.12	08/27/2020	Update with information on Read Only repositories
1.11	08/08/2020	Updates for V3R01 changes
1.10	07/17/2020	Updates for Version 3.00-rc
1.9	07/14/2020	Replace ROLLBACK with extract. Changed install.sh to zgininstall.rex. Added documentation on zgininstall.rex in Appendix B and changed Appendix B to C and Appendix C to D. Added MV as an alias of RN (Rename). Updates for Version 3.00-rc.
1.8	06/09/2020	Add ZGBatch
1.7	05/11/2020	Add Appendix C: Common Issues
1.6	05/08/2020	Add information on support for Load Modules
1.5	05/06/2020	Full user guide editing pass by Tech Docs
1.4	04/20/2020	Updates for V2R8 New ZG command for use with ISPF outside of ZIGI
1.3	03/29/2020	Add Appendix B: Learning Resources
1.2	03/27/2020	Updates for V2R7. Updated Action Bar for Current Repository. New Check command for Current Repository. Support for PDS Member file extensions on the OMVS file copy.
1.1	03/08/2020	Updates for V2R5. Change history in action bar. New Info command for current repository. Current repository SET new option to change Prefix.
1.0	03/01/2020	Major updates for V2R5
0.9	01/27/2020	Updates for version 2 release 0

<b>Version</b>	<b>Date</b>	<b>Description of Revision</b>
0.6	12/05/2019	Numerous changes for 1.4
0.5	11/26/2019	Numerous changes for 1.2
0.4	11/21/2019	Add updated GITLOG information
0.3	11/20/2019	Updated panels for v1r1. Added appendix for Git install. Added Sort and Reset commands for member list display.
0.2	11/17/2019	Minor additions
0.1	11/11/2019	Creation

## **Overview (What You Should Know but May Not)**

---

This topic provides a description of Git and ZIGI.

Git is a version control system that has been very popular in the distributed environment. Thanks to the great team over at Rocket Software, it is available (and has been for a few years) for the z/OS environment. The main drawback is 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. Open your favorite web browser and do a search or go directly to [github.com](https://github.com) where free accounts and many open source projects, including ZIGI, are available.

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 that enhance their productivity. That means that all of the ISPF capabilities are available to the ZIGI user as well as the ability to integrate their SCM requirements using Git.

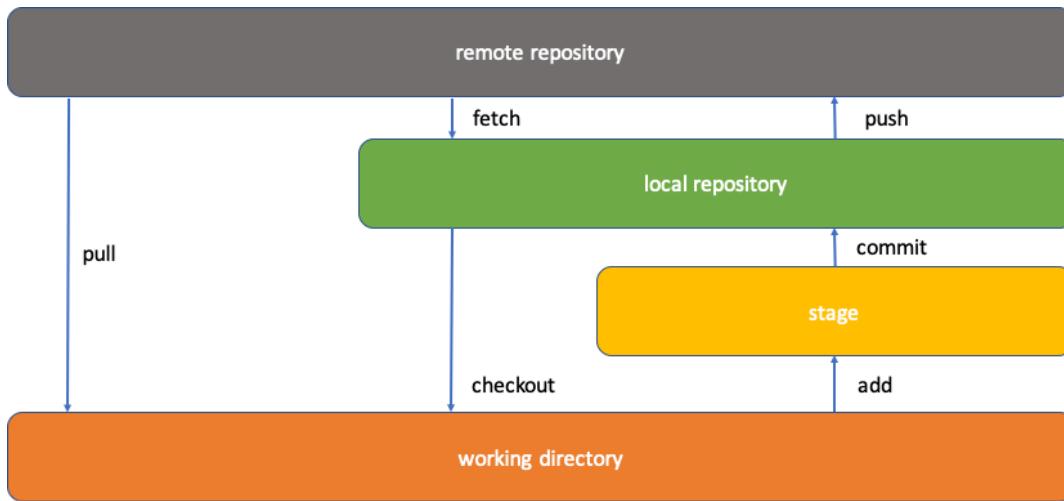
*NEXT TOPIC: [Pre-Requisites \(Installing Git\)](#)*

## **Pre-Requisites (Installing Git)**

---

This topic includes an image that explains the overview of Git processing.

See [Appendix A: Installing Git](#) for a checklist for installing Git and the additional tools it requires, which are open source and free.



*Figure 1: Overview of Git Processing*

NEXT TOPIC: [Installing ZIGI](#)

## Installing ZIGI

This chapter includes the following topics:

- [Using Git](#)
  - [Additional Options](#)
- [Using CBTTape](#)

## Using Git

This topic explains how to use Git.

### Procedure

1. Create an account at GitHub.com, and then add your (Mainframe) SSH Public Key to your account.
2. If you have not added your SSH keys to your GitHub settings, 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. Sign into OMVS, and then change to the directory where you are installing ZIGI.
4. Do one of the following:

- If your SSH key is already generated and added to your GitHub account, execute the following command:

```
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, execute the following command:

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

```
sh install.sh
```

*NEXT TOPIC: Additional Options*

## Additional Options

This topic provides an overview on how to download ZIGI from the internet, unzip the files, upload the files to your mainframe, and then run the installer if you're unable to connect your mainframe to GitHub.

If you do not have the option to connect 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 [here](#).

When running the zgininstall.rex script, there are a few prompts asking for a high-level qualifier (HLQ) for the ZIGI ISPF data sets to be created under. The ZIGI partitioned data sets are allocated as PDSE partitioned data sets. After the zgininstall.rex completes, exit OMVS, return to native ISPF, and then execute the ZGSTAT exec as documented in the zgininstall.rex report. That applies the ISPF statistics to the PDS members of ZIGI.

To use the ZG command, it must be copied from the ZIGI.EXEC library into a library in the standard SYSEXEC, or SYSPROC, allocation.

For more information, check out the [ZIGI wiki](#).

*NEXT TOPIC: Using CBTTape*

---

## Using CBTTape

### Procedure

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).
  1. Enter a data set name or take the default.
4. Open the resulting PDS and execute the \$INSTALL member, which receives the EXEC and PANELS members into full partitioned data sets.
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.
7. To use the ZG command, it must be copied from the ZIGI.EXEC library into a library in the standard SYSEXEC, or SYSPROC, allocation.

### What to do next

Check the [Updates](#) page on the [CBTTape website](#) for the latest release, and if you do not see it there, then go to the **CBT** page under **Downloads** on the left-hand side.

**Tip:** You can also download ZIGI from [this page](#) in file 997.

*NEXT TOPIC: Dovetail Enhancements*

---

## Dovetail Enhancements

ZIGI now supports the Dovetail enhanced getpds and putpds commands. These commands significantly speed up the copying of PDS members between the OMVS filesystem and the z/OS partitioned data sets.

The use of the Co:Z Toolkit is subject to Dovetail's [Community License](#).

Along with the getpds and putpds, which greatly enhances ZIGI, there are also:

- Co:Z SFTP
- Co:Z Launcher
- Co:Z Dataset Pipes
- Co:Z Batch

These commands are not fully exploited with the ZIGI 3.10 release as there are more capabilities available with them.

**Note:** This is optional. The standard OMVS cp command is used if these commands are not available.

*NEXT TOPIC: [Restrictions and Caveats](#)*

## Restrictions and Caveats

---

This topic explains the restrictions and caveats you should know when using ZIGI.

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:

- VSAM, BDAM, and ISAM are not supported.
- While ZIGI utilizes Git, only those git repositories that are configured for ZIGI can be managed by ZIGI. All others may be cloned, and the conversion utility used after some (or a lot of) manual changes.
- File extensions for the OMVS copy of PDS members is supported with the restriction that each PDS may only have one file extension mapped to it.

Load modules are supported from data sets with RECFM=U. The restriction is that for versions of z/OS prior to 2.4, the following PTFs must be installed:

- z/OS 2.2 | UJ01358
- z/OS 2.3 | UJ01356

*NEXT TOPIC: [Starting ZIGI the First Time](#)*

## Starting ZIGI the First Time

---

This task explains how to start ZIGI for the first time.

### About this task

### Procedure

1. In ISPF, use option 3.4 and enter the HLQ provided during the installation process or enter `DSLIST hlq` on the ISPF command line.
2. Select the EXEC data set using Browse, Edit, or View.

**Note:** ZIGI saves its repository ISPF table in the data set referenced by ISPTABL, but if that DD is not used then it uses the DD ISPPROF as the location for the table.

3. Next to the ZIGI member, enter EX to execute it.

This exec is the main ZIGI code and dynamically allocates the EXEC library using ALTLIB and the PANELS library using LIBDEF.

SLBD.ZIGING.ZIGI.EXEC						Row 0000001 of 0000021
Command ==> [ ]						Scroll ==> CSR
Name	Prompt	Size	Created	Changed	ID	
GITHHELP		914	2020/03/11	2020/06/10 06:31:58	ZIGI30	
SAMPLE		17	2020/02/13	2020/02/13 13:06:00	ZIGI30	
ZG		1437	2020/04/30	2020/06/10 06:48:48	ZIGI30	
ZGBATCH		491	2020/06/04	2020/06/11 03:36:38	ZIGI30	
ZGINSTAL		1227	2020/06/11	2020/06/14 12:14:00	ZIGI30	
ZIGI		7798	2020/02/15	2020/06/15 06:35:28	ZIGI30	
ZIGICHG		48	2020/03/11	2020/04/30 05:24:00	ZIGI30	
ZIGICKOT		648	2020/01/24	2020/06/13 16:11:18	ZIGI30	
ZIGICNVT		315	2020/02/27	2020/05/14 04:19:00	ZIGI30	
ZIGIEM		56	2020/03/11	2020/04/27 06:20:00	ZIGI30	
ZIGIEME		87	2020/03/11	2020/04/27 06:20:00	ZIGI30	
ZIGIEMS		81	2020/03/11	2020/04/27 06:20:00	ZIGI30	
ZIGIEXTR		756	2020/06/12	2020/06/14 09:03:42	ZIGI30	
ZIGIGCMD		323	2020/03/11	2020/06/10 06:41:28	ZIGI30	
ZIGIGENI		98	2020/06/11	2020/06/14 12:02:00	ZIGI30	
ZIGIMRGM		112	2020/03/11	2020/04/27 06:21:00	ZIGI30	
ZIGIOSEL		448	2020/03/11	2020/06/10 07:59:23	ZIGI30	
ZIGIRCSI		171	2020/03/11	2020/03/11 07:26:00	ZIGI30	
ZIGIREFD		202	2020/03/17	2020/06/11 13:19:21	ZIGI30	

4. 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

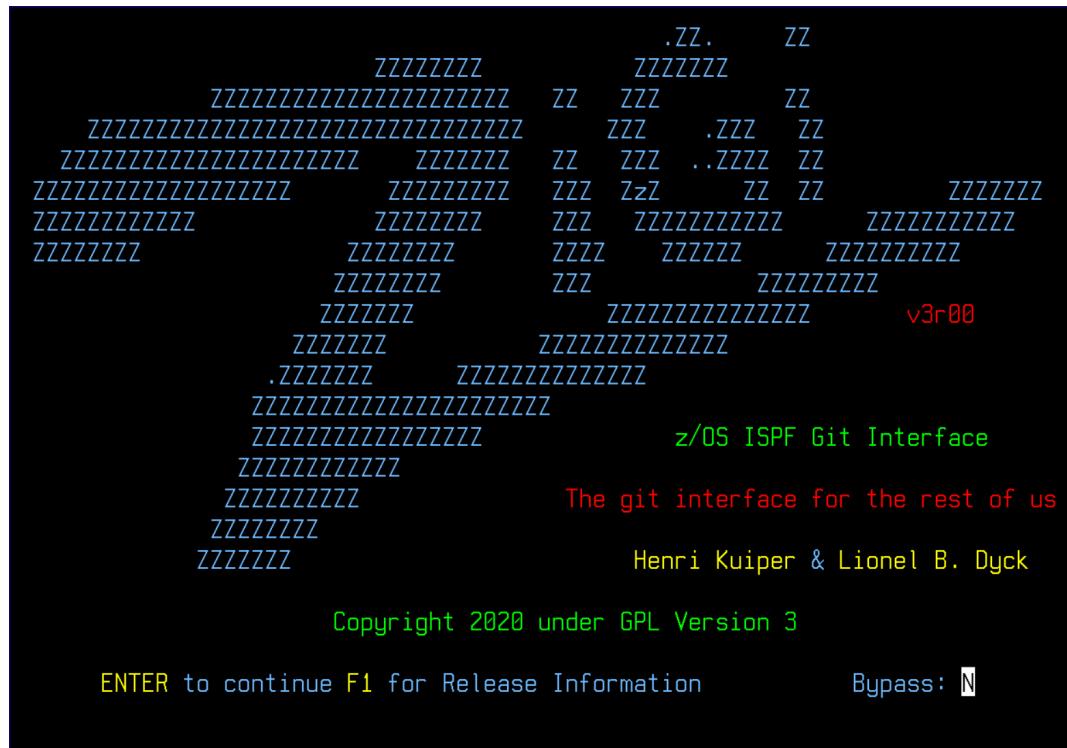
command TSO %ZIGI. This sample can be found in the provided EXEC library under the name SAMPLE.

```
/* ----- 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 'altnlib act application(exec) dataset('exec')'
Address ISPEexec
'libdef isplplib dataset id('panels') stack'
"Select CMD(%ZIGI" opt") Newappl(ZIGI) Passlib Scrname(ZIGI)"
'libdef isplplib'
Address TSO 'altnlib deact application(exec)'
```

The ZIGI Splash panel is presented.



5. Next, you are prompted for your username and e-mail along with three ZIGI processing defaults:
  1. It is recommended that the **PDSE Member Generations** default value is 0 unless you have a tool that supports working with member generations. See [PDSEGEN on file 969](#).
  2. The **Display Options Popup** option changes the behavior of point-and-shoot so that on table panels, a click above the table displays the commands pop-up and anywhere in the table the line selection pop-up options.
  3. Set the **Bypass File Extension Prompt** if you do not plan to use file extensions for the OMVS files.

```
Set Defaults ----- (ZIGI v3r00)
Command ==> [REDACTED]

Git Defaults (used at Commit time):
  user.name : lionel_dyck
  user.email : lioneld@21csw.com

PDSE Member Generations : 0 (0 to 999) System Limit: 2000000000
Note: Specify 0 unless using a tool that supports generations.

Display Options Popup or Select on Point-and-Shoot: P (P or S)

Bypass File Extension Prompt: N (Y or N)
Set to Y if you do not plan to use file extension for PDS members to bypass being asked when adding partitioned datasets.

Update using the Config command on the Local Repository Menu.

See: https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

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

6. After providing your name and e-mail, if you haven't created an SSH key pair yet, ZIGI creates this for you and shows 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).

```
SSH Key Review ----- (zigi v2r7)
Command ==> [REDACTED]

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

Copy the records below and paste into your Git Account SSH Key:
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQC829P3Kkz0DSYYDrL+EdC5Z8ZJoaazaFdLyJjPNujfb
RnLToBHWMCoPq9DFEdZWK6Y/oOpzLZ052VHeG21b3IIHgTGvQcuijZPSi4MBhARe3s26StA826GIC5+n
Asir1WSn5dKyK0k3E278pkneu0Q8SAAxVRZ3Uoq50WnVp+/ldzmSciXD84e4gbShqLY3kqZ0qAPy6IwR
1QYlukAH16gXfR6hu89iIbE0J9AdFr1hV0ve07Fn0VThsilWGmSnN3ghjGJoKFpANAf1dVS6aNQn0jGr
j9bg4qMjj/X64d61y7jJT0TaSD3t13Lb+f+NvWEc3KVVJ1uR8CHq7mWwXioH SLBD@S0W1

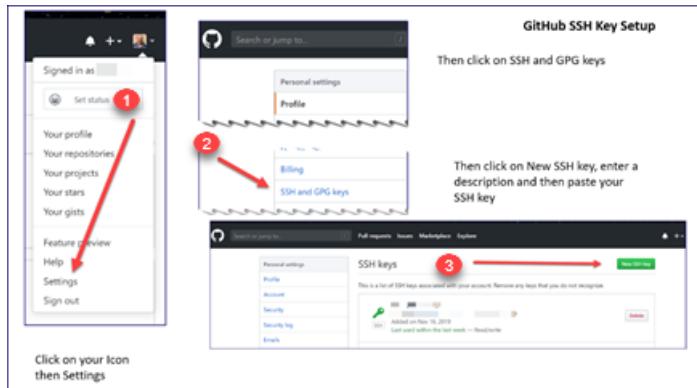
[REDACTED]
[REDACTED]
[REDACTED]
[REDACTED]
[REDACTED]

Press F3 when ready to continue.
```

**7. Copy your SSH key and paste it into one of the following:**

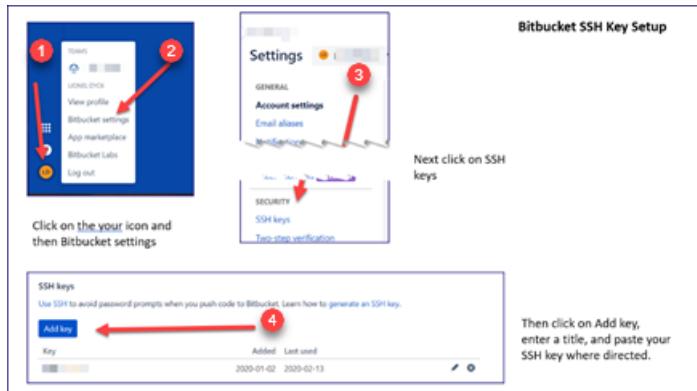
- GitHub settings

1. Click on your icon, and then click **Settings**.
2. Click **SSH and GPG keys**.
3. Click **New SSH key**, enter a description, and then paste your SSH key.



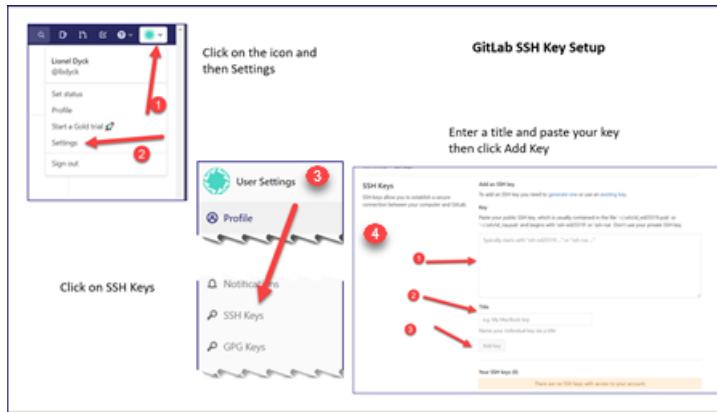
- BitBucket settings

1. Click on your icon.
2. Click **Bitbucket settings**.
3. Click **SSH keys**.
4. Click **Add key**, enter a title, and then paste your SSH key where directed.



- GitLab settings

1. Click the User Settings icon in the upper right-hand corner.
2. From the drop-down menu, select **Settings**.
3. Click **SSH Keys**.
4. Paste your key, enter a title, and then click **Add key**.



[NEXT TOPIC: ISPF Dialog Notes](#)

## ISPF Dialog Notes

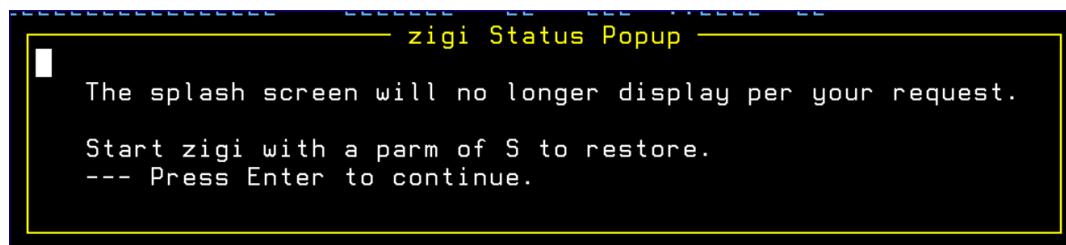
This topic explains the various ISPF dialog notes you may see while using ZIGI.

Many of the ISPF panels support point-and-shoot for commands as well as row selections. This requires that your TN3270 emulator supports 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 also proves 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 displays a pop-up menu of commands.

ISPF pop-ups are used to provide information to you at various points in the dialog where processing may not be instantaneous.

When starting ZIGI, the splash screen always displays. The splash screen can be disabled by entering a **Y** to the Bypass option on the splash screen. This brings up the **zigi Status Popup**:



[NEXT TOPIC: The ZIGI Local Repositories Panel](#)

## The ZIGI Local Repositories Panel

This topic explains the components of the **ZIGI Local Repositories** panel.

After the preparatory work above, the main **Local Repositories** panel displays, which lists all of the local Git repositories that have been added to ZIGI:

Commands Help		Local Repositories ----- (ZIGI v3r00) ----- Row 1 to 18 of 25		
Command ==> █		Scroll ==> CSR F3		
Repository	Prefix	Category	Last Access	
zigi	SLBD.ZIGING	zigi	12 Jul 2020	
TestPull1	SLBD.TESTPULL	testing	12 Jul 2020	
zsync	SLBD.ZSYNCNG	tools	12 Jul 2020	
ztsohelp	SLBD	tools	12 Jul 2020	
sharevar	SLBD.SHAREVAR	tools	12 Jul 2020	
zinstall	SLBD.ZGSTAT	tools	12 Jul 2020	
racfadm	SLBD	RACF	12 Jul 2020	
pdsegen	SLBD	tools	12 Jul 2020	
Runc	SLBD	tools	12 Jul 2020	
TestPull2	SLBD.TESTP2	testing	11 Jul 2020	
TotalStorageBinary	SLBD.ICE170	ipk	10 Jul 2020	
Total_Storage	SLBD.IPK	Total-Storage	10 Jul 2020	
SpaceFinder_Source	SLBD.TSF	SpaceFinder	10 Jul 2020	
sfc	SLBD	tools	7 Jul 2020	
txt2csv	SLBD.TXT2CSV	tools	27 Jun 2020	
zospcre2	SLBD.PCRE2	tools	7 Jun 2020	
body	SLBD	cbt	22 May 2020	
githelp	SLBD	tools	22 May 2020	

There are both commands and row selections available for use. The commands are available via the Action Bar menu and the row selections are available by entering a / in the row selection. This means you can move the cursor to them and press `Enter` or move the cursor there using your mouse and double-click with the left mouse button (this assumes your TN3270 emulator supports that feature; most do but you may have to enable it).

This chapter includes the following topics:

- [Action Menu Bar](#)
- [Clone](#)
- [Config](#)
- [Connect](#)
- [Create](#)
- [GITHELP](#)
- [Options Menu Assist](#)
- [Select Command](#)
- [Sort Selections Pop-Up Menu](#)
- [SSH Public Key](#)
- [Row Selections](#)

## Action Menu Bar

This topic explains the Commands and Help menus.

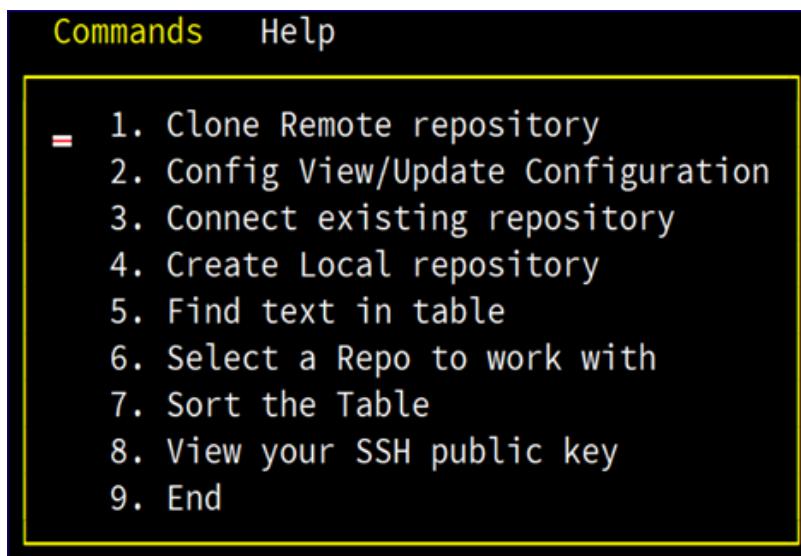
The action bar menus are there to make it easy to access commands when you forget, or want to use the mouse for point-and-shoot.

This section includes the following topics:

- [The Repository Commands Menu](#)
- [The Help Menu](#)

## The Repository Commands Menu

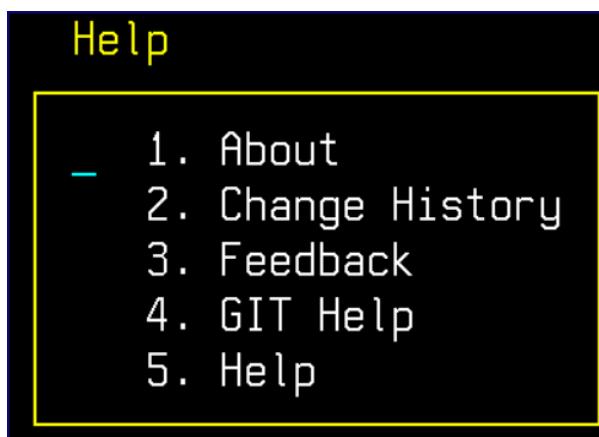
This topic provides a screenshot of the **Commands** menu.



*NEXT TOPIC: [The Help Menu](#)*

## The Help Menu

This topic provides a screenshot of the **Help** menu.



*NEXT TOPIC: [Clone](#)*

## Clone

This topic explains the clone repository functionality.

Clone makes a copy of a repository that resides on a remote server on your local OMVS filesystem and into a set of z/OS data sets based on a HLQ that you provide.

```

Clone Repository ----- (ZIGI v3r00) -----
Command ==> █

Remote Repository: Git Repository URL
git@github.com:wizardofzos/zigi.git

Branch Name: Specific branch to clone (default is master)

PREFIX for datasets: slbd.testzigi (no quotes)
Default Push on Commit: y (Y or N)

Repository Root Directory: ? to browse OMVS folder structure
/u/slbd/testzigi

Repository Name: TestZIGI (optional)
Category for Repository: testing (optional)
Default Userid 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.

The authors have selected to enter the local OMVS directory manually, and the HLQ (prefix) for the z/OS data sets is specified.

If a ? is entered into the **Local dir** field, then a dialog is displayed to walk down the OMVS directories to find or make (MKDIR) the directory where the cloned repository is placed.

All fields except **Branch Name**, **Repository Name**, and **Category for Repository** are required fields.

The **Repository Name** defaults, if blank, to the Git name of the repository, otherwise the name is used for the local OMVS directory name for the clone operation.

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

The **Default Push on Commit** option, if set to Y, is the default on the Commit panel to push after each commit.

The **Default Userid set prior to Commit** 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 data sets in the repository are displayed.

This is also the panel that is displayed when the repository is selected from the primary panel.

If any existing z/OS data sets are about to be replaced by the clone operation, a display is presented during the clone process asking for permission with an option to change the data set prefix for the cloned data sets.

**Note:** ZIGI creates PDSE Version 2 libraries for all partitioned data sets with a default MAXGEN of 0 (see the [Config](#) command to change the default number of generation).

```

General Local Repo Remote Repo Handy Functions Help
Current Repository ----- (ZIGI v3r00) ----- Row 1 to 8 of 8
Command ==> CSR
Local dir   : /u/slbd/zigi
Remote path : origin git@github.com:wizardofzos/zigi.git
Current Branch: devlbd
Your branch is up-to-date with 'origin/devlbd'.

S Status          Dataset/File Name
- zginstall.readme
- zginstall.rex
- README.md
- 'SLBD.ZIGING.ZIGI.EXEC'
- 'SLBD.ZIGING.ZIGI.GPLLIC'
- 'SLBD.ZIGING.ZIGI.PANELS'
- 'SLBD.ZIGING.ZIGI.README'
- 'SLBD.ZIGING.ZIGI.RELEASE'
***** Bottom of data *****
```

NEXT TOPIC: [Config](#)

## Config

This topic explains the Config command.

This command allows you to view and update, if desired, your username and e-mail, the PDSE member generations default, and the display options pop-up.

Another option that is configured here is the option to bypass being asked if a file extension should be used for all members of the added PDS when the members are copied to the repository's OMVS filesystem.

```

Set Defaults ----- (ZIGI v3r00) -----
Command ==> CSR

Git Defaults (used at Commit time):
user.name : lionel_dyck
user.email : lioneld@21csw.com

PDSE Member Generations : 0 (0 to 999) System Limit: 2000000000
Note: Specify 0 unless using a tool that supports generations.

Display Options Popup or Select on Point-and-Shoot: P (P or S)

Bypass File Extension Prompt: N (Y or N)
Set to Y if you do not plan to use file extension for PDS members to bypass
being asked when adding partitioned datasets.

Update using the Config command on the Local Repository Menu.

See: https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

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

In the future, this is also where other, user-customizable options can be configured. At this point, the **PDSE Member Generations** field should be 0 unless there is a tool available that can work with Member Generations (see PDSEGEN for an open-source solution).

*NEXT TOPIC:* [Connect](#)

## Connect

---

This topic explains the Connect command.

The Connect command adds an existing local ZIGI-formatted Git repository to the ZIGI ISPF Table for ZIGI management.

```

Connect Repository (ZIGI v3r10)
Command ==> _____
Local Directory: ? to browse OMVS folder structure
-
PREFIX for datasets: _____ no quotes
Default Push on Commit: _____ Y or N
Read Only: _____ Y or blank
Category for Repository: _____ optional
Default Userid to set prior to Commit: _____ or blank for no change
Specify zos-workingtree-encoding: _____ ibm-1047 if blank
-
Press Enter to continue, or F3 to cancel

```

The local repository should point to the existing OMVS filesystem directory where the Git repository exists, and the repository must conform to ZIGI standards (with a .zigi subdirectory).

The other options are identical to the Clone and Create panels.

**Note:** The repository name is dynamically determined by the name of the repository directory.

*NEXT TOPIC:* [Create](#)

## Create

---

This topic explains the Create command.

The Create command creates a new local repository that can be added to a remote repository.

```
Create New Repository ----- (ZIGI v3r02) -----
Command ==> [ ]
```

Name: test4zigi

Local Directory: ? to browse OMVS folder structure  
/u/slbd/t

PREFIX for datasets: slbd.test4zip no quotes  
Default Push on Commit: n Y or N  
Read Only: n Y or blank

Category for Repository: testing optional  
Default Userid to set prior to Commit: \_\_\_\_\_ or blank for no change  
Specify zos-workingtree-encoding: \_\_\_\_\_ ibm-1047 if blank

Press Enter to continue, or F3 to cancel

In the above example, a test repository is created in your test directory.

Using a ? in the **Local Directory** field brings up a file directory list from which a directory may be selected or created (MKDIR) and then selected.

The default push on commit option, if set to Y, is the default on the Commit panel to push after each commit. A read only repository is one in which the z/OS data sets are never updated by ZIGI and are intended to track changes to z/OS data sets. All changes are tracked by ZIGI and Git. All ZIGI features that could update or delete z/OS data sets are blocked. These repositories can be pushed to a Git server and then cloned; however, clone requires a data set prefix that should prevent a clone from replacing any existing z/OS data sets.

The **Category for Repository** is used for the ZIGI Local Repositories display to categorize the local repositories for sorting and easier identification.

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

The encoding is the character set to be used for all data added to the repository. The default is ibm-1047.

The Read Only option creates a read-only repository. This repository is configured such that ZIGI does not allow any updates to the z/OS data sets from within ZIGI. This may be used for a shared local repository where several users access it for reference while preventing updates.

The repository may still be updated if the z/OS data sets are updated outside of ZIGI which causes ZIGI to update the OMVS files and thus the Git repository.

Edit, Merge, Branch, Replace, and any other ZIGI action that could update the z/OS data sets is prevented, but working outside of ZIGI does not prevent the updates from occurring.

```
OMVS Selection ----- (ZIGI v3r00) ----- Row 1 to 18 of 24
Command ==> F3

Directory: /u/slbd/

S T OMVS File Date Perm
d . Jun 11 10:29 755
d .. Jun 11 10:25 555
- .gitconfig Feb 11 16:18 644
- .gitignore Mar 11 17:35 644
- .profile Apr 13 12:15 700
- .setup Oct 11 2019 700
- .sh_history May 12 17:41 600
d .ssh Jun 11 12:59 777
d .zigi Jun 11 19:50 755
d Only Mar 10 13:21 777
d cbt Feb 12 18:59 777
d cmt Apr 11 19:32 777
d git Jun 11 17:53 755
d mnt Apr 10 2019 755
d omvscopy Apr 10 16:58 777
d pdsegen Mar 10 17:34 777
d pmr Jun 10 11:39 755
d racfadm Jun 11 11:34 777
```

After the Create process completes, the **Add Datasets** panel displays to facilitate populating the repository with z/OS data sets:

```
Add Datasets ----- (ZIGI v3r00) -----
Command ==> Scroll ==> CSR F3

Fixed Dataset Prefix: SLBD.TEST4ZIG
Ignore 1st n qualifiers in prefix: (0 to 8)

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

S Dataset Name Volser Status
***** Bottom of data *****
```

This can also be achieved using the ADDDSN command to add existing z/OS data sets to the repository. See the [AddDsn Command](#) command, which is not to be confused with the add row selection, for more information.

*NEXT TOPIC: [GITHELP](#)*

## GITHHELP

---

This topic explains the GITHHELP ISPF dialog.

GITHHELP is an ISPF dialog that provides simplified access to the Git help documentation. If you have a screen wider than 80 columns, then an alternate display is used where there is one row per command.

Syntax: GITHelp optional command (e.g. gith commit)

```
----- git manpages ----- Row 1 of 146
Command ==> [ ] Scroll ==> CSR

Commands: Find string Only string Refresh
Line: 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
  An overview of recommended workflows with Git
- git-add
  Add file contents to the index.
```

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

Selecting a command displays the command's manpage:

```
Menu Utilities Compilers Help
BROWSE SYS20061.T140239.RA000.ZDOLD.R0100447 Line 0000000000 Col 001 080
Command ==> [ ] Scroll ==> CSR
***** Top of Data *****
GITEVERYDAY(7) Git Manual GITEVERYDAY(7)

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 purposes
  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.

  * If you work with other people, you will need commands listed in the
    Individual Developer (Participant) section as well.
```

[NEXT TOPIC: Options Menu Assist](#)

## Options Menu Assist

This topic explains the options menu assist pop-up menu.

Use `o` on the command line to bring up the **zigi Local Repository Commands** menu of all available commands. This is useful if you selected the Pro or Hidden menu.



*NEXT TOPIC:* [Select Command](#)

## Select Command

This topic explains the Select command.

If you don't want to Tab to a row, do one of the following:

- Move the cursor to a row
- Use point-and-shoot
- 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.

**Tip:** For the Select command, the search is case insensitive.

*NEXT TOPIC:* [Sort Selections Pop-Up Menu](#)

## Sort Selections Pop-Up Menu

This topic explains the **zigi Sort Selections** pop-up menu.

Sort the table based on the columns.

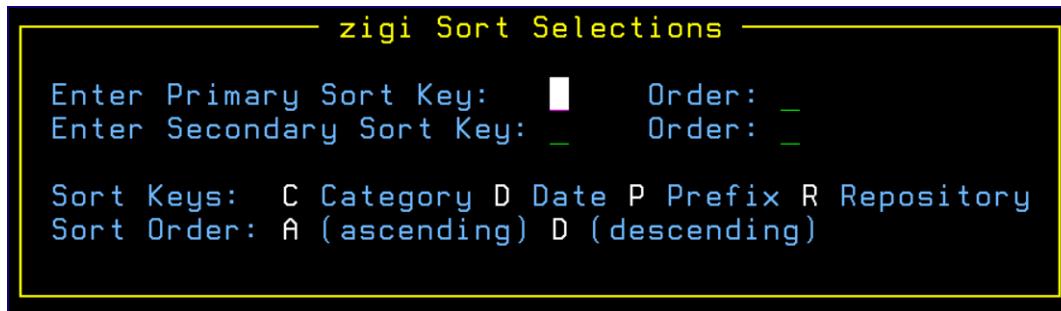
Syntax: `Sort column1 order1 column2 order 2`

Abbreviation of `SO` is allowed.

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

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

If only the command **SORT** is entered, the **zigi Sort Selections** pop-up displays to assist.



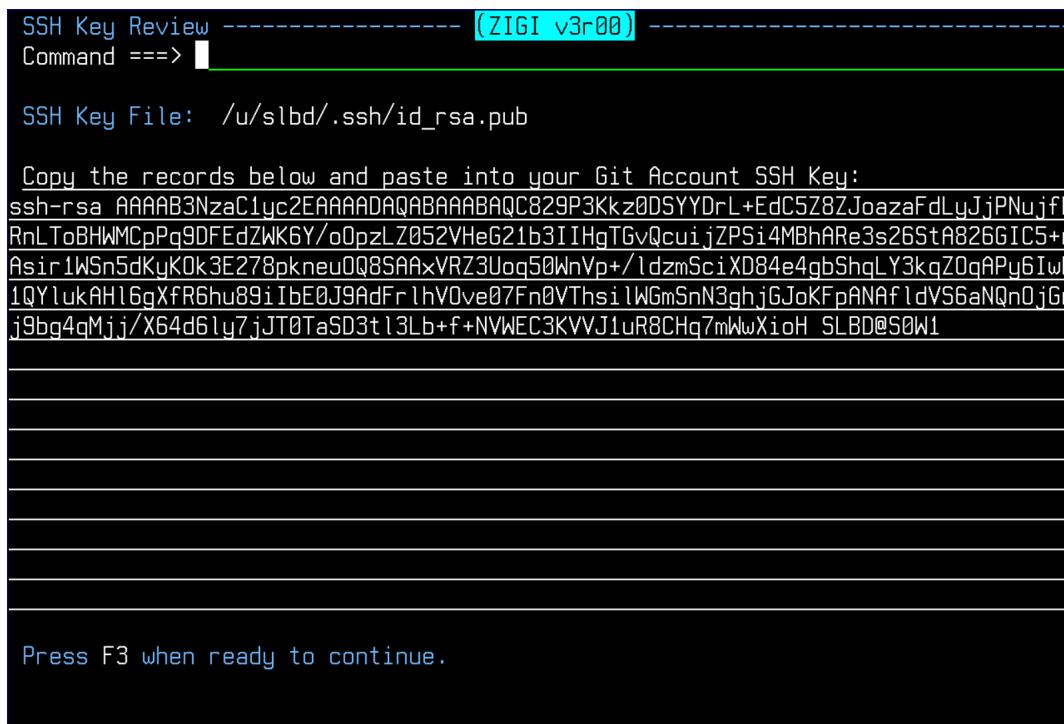
**Note:** The last sort order is retained and used the next time ZIGI starts.

*NEXT TOPIC: [SSH Public Key](#)*

## SSH Public Key

This topic explains how to display your SSH public key.

This command displays 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. There is no concern about sharing your public key since it must be used in concert with your private key, which should never be shared with anyone.



*NEXT TOPIC: [Row Selections](#)*

## Row Selections

---

This section includes the following topics:

- [Select Option](#)
- [Info Option](#)
- [View Option](#)
- [Delete Option](#)
- [/ Row Selection Prompt](#)

### Select Option

Selecting a repository using S (or point-and-shoot) opens the repository.

*NEXT TOPIC: [Info Option](#)*

### Info Option

The info option displays a short summary of information about the repository.

```
----- ZIGI Local Repository Summary Info -----
Name      : racfadm
Repository : racfadm
Branch    : master
Remote    : git@github.com:lbdyck/racfadm.git

Home Directory: /u/slbd/racfadm
Category     : Tools
z/OS HLQ     : SLBD
Ignore Qual  : 1
Last Access   : 11 Oct 2020
Default Userid:
Default Push  : Y
Read Only    : No
```

*NEXT TOPIC: [View Option](#)*

### View Option

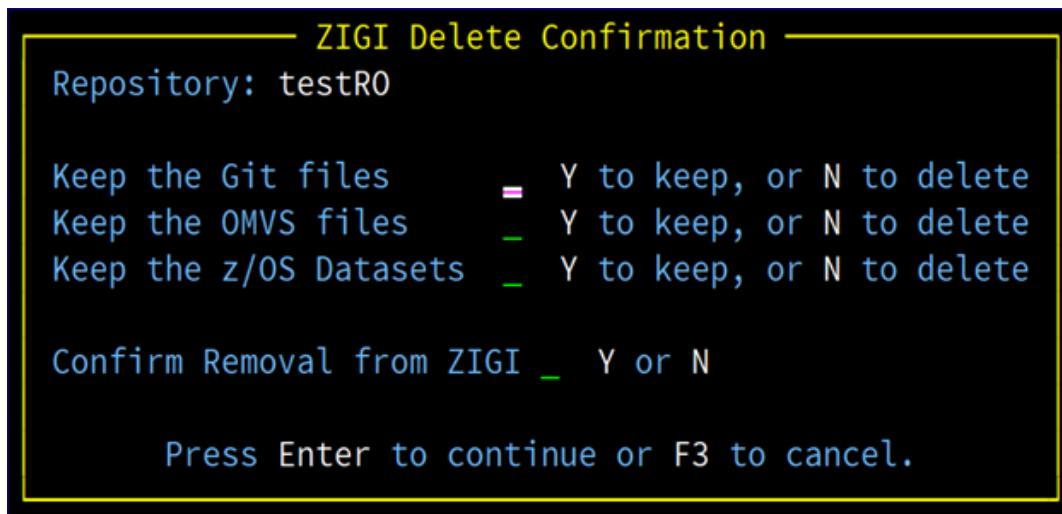
Using View (v) opens the ISPF 3.17 (UDList) on the repository's OMVS filesystem.

*NEXT TOPIC: [Delete Option](#)*

### Delete Option

The Delete option removes the repository from ZIGI management by removing it from the ZIGI local repository ISPF Table. Options are provided to keep, or delete, the Git files, OMVS files, and z/OS data sets.

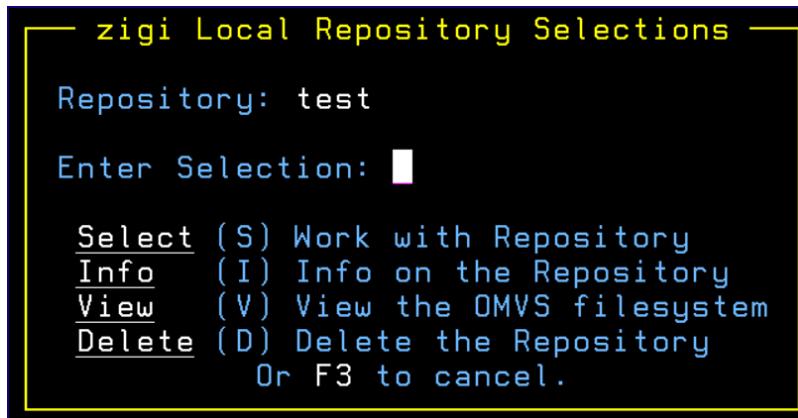
Confirmation on the **ZIGI Delete Confirmation** screen is required to proceed as a failsafe.



*NEXT TOPIC: / Row Selection Prompt*

### / Row Selection Prompt

If / is entered for a row selection, the **zigi Local Repository Selections** menu displays to assist you:



*NEXT TOPIC: The ZIGI Current Repository Panel*

## The ZIGI Current Repository Panel

---

This topic explains the components of the **ZIGI Current Repository** panel.

When a repository is selected from the Local Repositories panel, the z/OS data sets associated with the repository are displayed:

The screenshot shows the z/OS ISPF Git Interface window. At the top, there's a menu bar with tabs: General, Local Repo, Remote Repo, Handy Functions, and Help. Below the menu, it says "Current Repository ----- (ZIGI v3r00) ----- Row 1 to 8 of 8". It also shows the command line "Command ==> [ ]" and scroll options "Scroll ==> CSR F3". The local directory is "/u/slbd/zigi" and the remote path is "origin git@github.com:wizardofzos/zigi.git". The current branch is "devlbd" and it is up-to-date with "origin/devlbd".

S	Status	Dataset/File Name
—		zgininstall.readme
—		zgininstall.rex
—		README.md
—		'SLBD.ZIGING.ZIGI.EXEC'
—		'SLBD.ZIGING.ZIGI.GPLLIC'
—		'SLBD.ZIGING.ZIGI.PANELS'
—		'SLBD.ZIGING.ZIGI.README'
—		'SLBD.ZIGING.ZIGI.RELEASE'
***** Bottom of data *****		

For demonstration purposes, this user guide uses the ZIGI repository to start with.

This section includes the following topics:

- [Action Bar Menus](#)
- [AddAll Command](#)
- [AddDsn Command](#)
- [Branch Command](#)
- [Check Command](#)
- [Commit Command](#)
- [Convert Repository \(CONVREPO\) Command](#)
- [DIFF Command](#)
- [Extract Command](#)
- [Flow Command](#)
- [GitCmd Command](#)
- [Git Help](#)
- [GitLog Command](#)
- [Grep Command](#)
- [Network](#)
- [Options Menu Assist](#)
- [Pull](#)
- [Push](#)
- [Replace](#)
- [Remote Command](#)
- [Set Command](#)
- [Snapshot](#)
- [Stash](#)
- [Stash List \(STASHL\) Command](#)
- [Status Command](#)
- [Tag](#)
- [TagList](#)
- [View](#)
- [Row Selections](#)

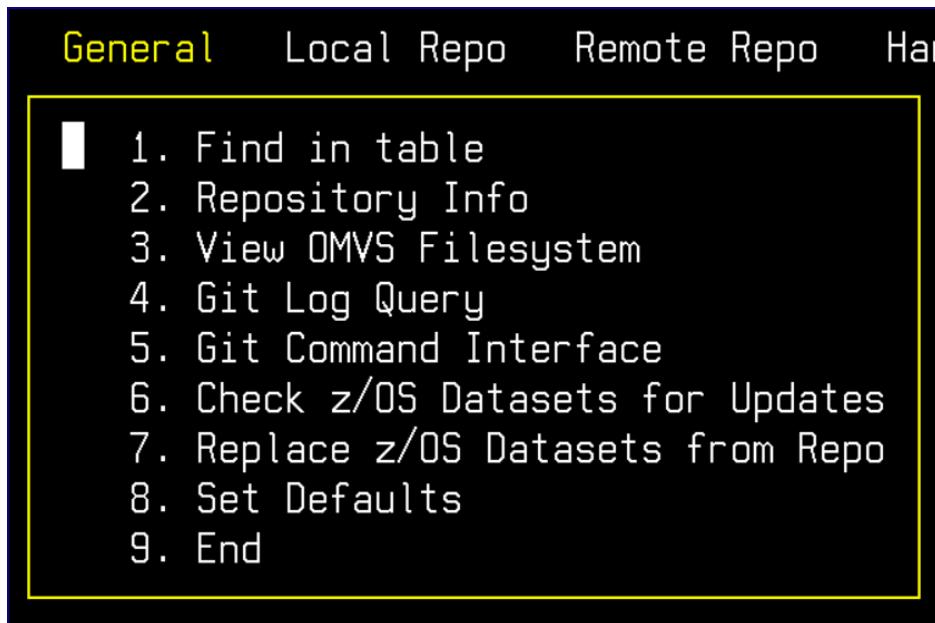
## Action Bar Menus

This section includes the following topics:

- [The General Actions Menu](#)
- [The Local Repo Actions Menu](#)
- [The Remote Repo Actions Menu](#)
- [The Handy Functions Actions Menu](#)
- [The Help Actions Menu](#)

### The General Actions Menu

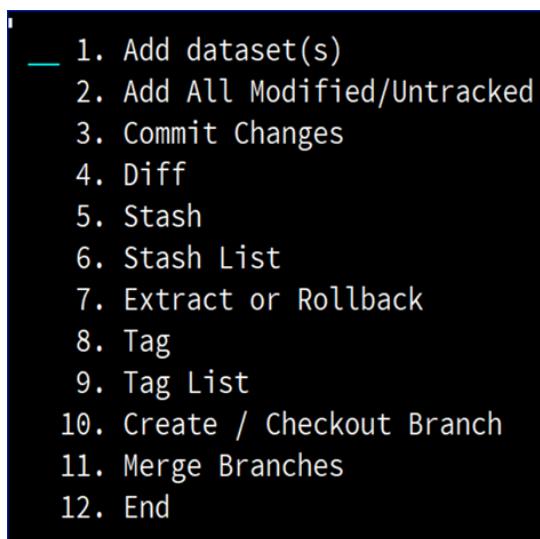
This topic provides a screenshot of the **General** actions menu.



*NEXT TOPIC: [The Local Repo Actions Menu](#)*

### The Local Repo Actions Menu

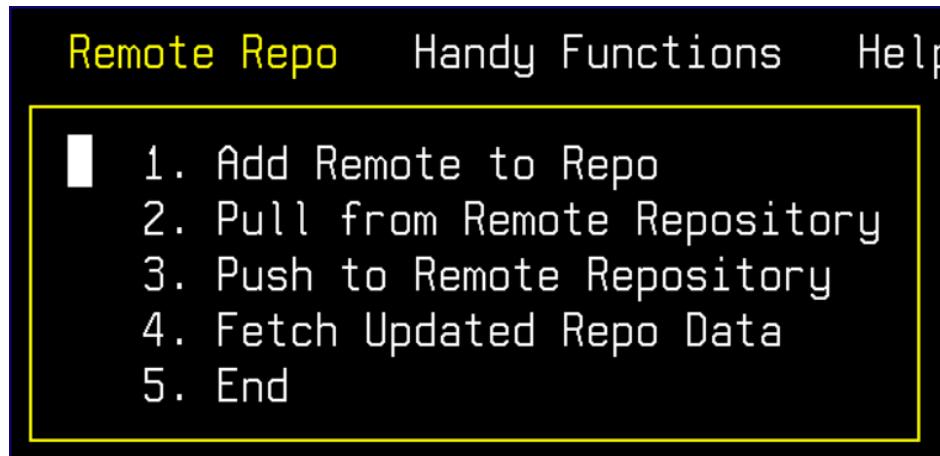
This topic provides a screenshot of the **Local Repo** actions menu.



*NEXT TOPIC: [The Remote Repo Actions Menu](#)*

## The Remote Repo Actions Menu

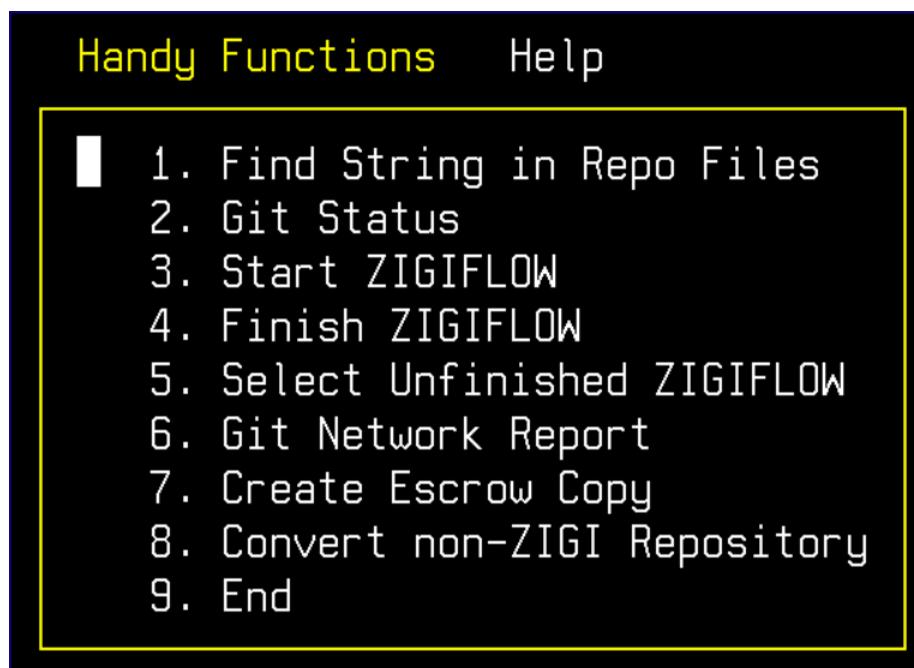
This topic provides a screenshot of the **Remote Repo** actions menu.



*NEXT TOPIC: [The Handy Functions Actions Menu](#)*

## The Handy Functions Actions Menu

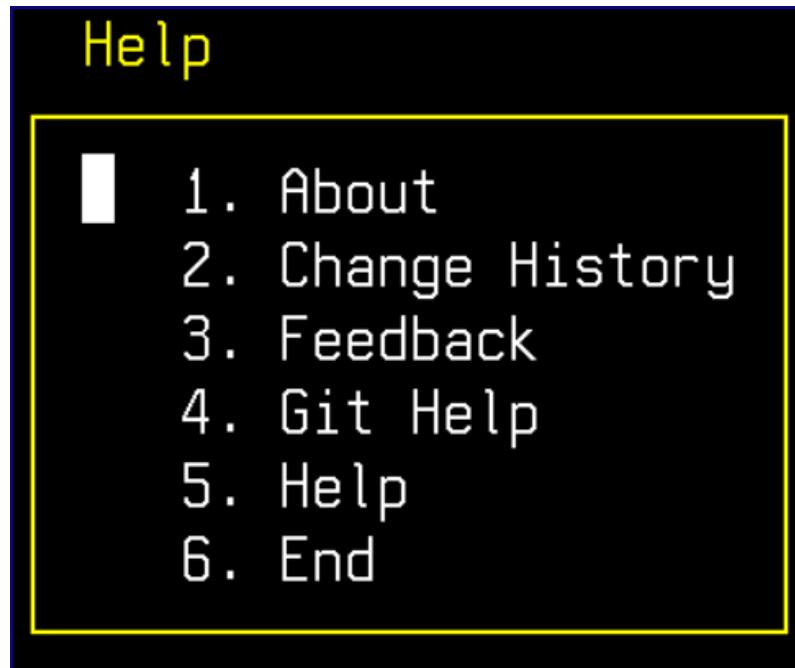
This topic provides a screenshot of the **Handy Functions** action menu.



*NEXT TOPIC: [The Help Actions Menu](#)*

## The Help Actions Menu

This topic provides a screenshot of the **Help** actions menu.



*NEXT TOPIC:* [AddAll Command](#)

## AddAll Command

This topic explains the functionality of the AddAll command.

The AddAll command adds all modified and untracked data sets and files to the Git staging index, which readies them for the next Git Commit.

This is performed using the git add . command.

*NEXT TOPIC:* [AddDsn Command](#)

## AddDsn Command

This topic explains the functionality of the AddDsn command.

The AddDsn command makes it easy to add additional z/OS data sets to the local repository.

```

Add Datasets ----- (zigi v2r5) ----- Row 1 of 9
Command ==> █
Fixed Dataset Prefix: ZDOLD.TEST
Ignore 1st n qualifiers in prefix: 1
Scroll ==> CSR
F3

Command: Find
Line: S Add A Add AB Add binary B Browse

S   Dataset Name                      Status
--- ZDOLD.TEST.$AB#DE.PDS             Added
--- ZDOLD.TEST.PDS                   Added
--- ZDOLD.TEST.PDSX
--- ZDOLD.TEST.PDSX.CLONE
--- ZDOLD.TEST.PDSZ
--- ZDOLD.TEST.TEXT                  Added
--- ZDOLD.TEST.TEXT.CLONE
--- ZDOLD.TEST.TEXTZ
--- ZDOLD.TEST.TEXTZ.CLONE
***** Bottom of data *****

```

In the figure above, four data sets were already added to the local repository and others are available to add.

Note that only z/OS data sets that have been added are managed by ZIGI as z/OS data sets. This adds an entry for the data set into the .zigi/dsn control file. If a data set is not included in .zigi/dsn, then it is not copied from z/OS to OMVS for Git management. If a file is in the OMVS filesystem and managed by Git, it is not copied to a z/OS data set unless it is found in the control file.

If the repository is new and there are no z/OS data sets included, then the **Fixed Dataset Prefix** and **Ignore 1st n qualifiers in prefix** are blank and must be filled in before the list of z/OS data sets may be presented. The qualifiers to ignore should be at least 1 but may be 0.

The **Fixed Dataset Prefix** field is the HLQ for the data sets to be presented, from which one, or more, may be added to the repository. Be aware that adding a binary data set using the A or S option adds the data set as text data and not binary. If the data set contains binary data, then add it using the AB (add binary) option.

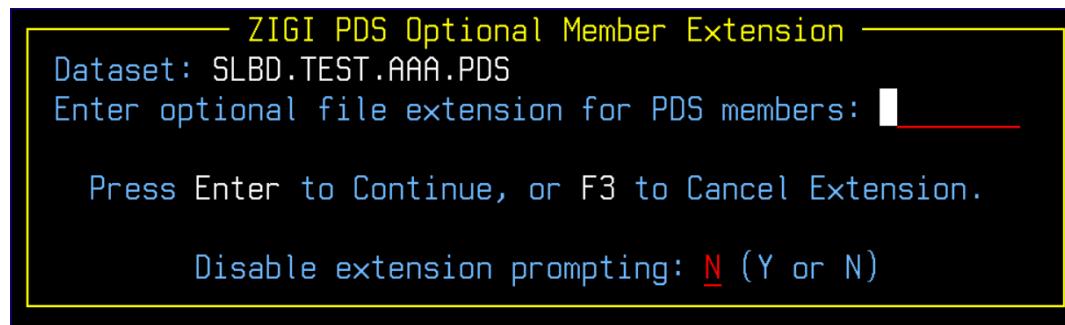
If the data set is a load library (RECFM=U), then it is added as binary and as an executable load module regardless of the selection (S, A, AB). If there are aliases associated with any load module then those are retained as well in the copy operation from z/OS to OMVS and restored in the OMVS to z/OS copy.

The **Ignore 1st n qualifiers in the prefix** field instructs ZIGI to ignore those qualifiers when creating the OMVS file, or directory, for the data set. Thus GITUSER.ZIGI.ZIGI.README is created in the OMVS filesystem as ZIGI.README. Then when someone else clones this repository, the file is uploaded using the HLQ that they provide, which results in a z/OS data set of hlq.ZIGI.README. Make sure that the ignore count does not exceed the number of qualifiers of the data set 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 is      TOOL.JCL
When someone clones with an HLQ of HENRI.REPOS.CLONES, the z/OS Dataset Name is
HENRI.REPOS.CLONES.TOOJCL
```

When adding a PDS, you are prompted to enter an optional file extension to be used when copying the PDS members to the OMVS filesystem. This enables the use of workstation clients to clone and update a repository by using the file extensions to easily identify the language type within the workstation editors (for example: source.c instead of SOURCE). If the optional extension is not provided, then the member names continue to be copied in uppercase to the OMVS filesystem without any file extensions.



There is a restriction that only one file extension is allowed per PDS. This restriction is in place to prevent member name collisions from two files with the same file name but different file extensions (for example: source.c and source.jcl). This prompt may be disabled for the current set of adds by entering a Y to disable.

**Note:** The **ZIGI PDS Optional Member Extension** prompt may be bypassed by setting the bypass option to Y (yes) on the config settings panel from the **Local Repository** panel.

*NEXT TOPIC: [Select \(S\) and Add \(A\) Commands](#)*

## Select (S) and Add (A) Commands

This topic explains the Select and Add commands.

The Select (**S**) and Add (**A**) selections add the data set as text data, or if RECFM=U, as a binary executable that includes tagging the file as binary and updating the .gitattributes file accordingly.

*NEXT TOPIC: [Add Binary \(AB\) Function](#)*

## Add Binary (AB) Function

This topic explains the Add Binary (AB) function.

The Add Binary (**AB**) function adds the entire data set as binary data. This includes tagging the OMVS copy of the data set with the binary tag (ctag) and updating the .gitattributes file. If the data set is a PDS that contains both text and binary members: **S** or **A** the entire data set, select the PDS from the **Current Repository** panel, and then from the Member list **AB** the individual members that are in binary format.

*NEXT TOPIC: [Browse \(B\)](#)*

## Browse (B)

Browse (**B**) allows you to verify the data before adding it using ISPF Browse.

*NEXT TOPIC: [Browse](#)*

### Browse

Browse browses the member using ISPF Browse.

*NEXT TOPIC: [Branch Command](#)*

---

## Branch Command

This topic explains the functionality of the Branch command.

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

```
----- [zigi v2r5] ----- Row 1 of 2
Command ==> █
Current Repository : zigi
Current HLQ       : ZDOLD.ZIGI25
Current Branch    : v2rn-dev

Type C to checkout an existing branch
Type D to locally delete a branch
Press F3 to exit

Or create a new branch: _____
S  Branch          Status
  master           Remote
  v2r5-rc          Local/Remote
***** Bottom of data *****
```

**Note:** The **Status** column indicates if the branch is only on the **Remote** server, or if there is a local copy (**Local/Remote**).

*NEXT TOPIC: [Check Command](#)*

## Check Command

This topic explains the functionality of the Check command.

The Check command checks all z/OS data sets, based on the last reference date from the data set control block (DSCB), to determine if the data has changed since the repository was opened. This is useful if the repository has been open for a period of time and the repository's data sets have been updated outside of ZIGI.

*NEXT TOPIC: Commit Command*

## Commit Command

This topic explains the functionality of the Commit command.

The Commit command commits 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.

```
Git Commit ----- (zigi v2r5) ----- Row 1 of 15
Command ==> _____ Scroll ==> CSR
Commit Title: [ ] F3
Optional Tag: _____ Push after: Y (Y/N)
Command: Insert or Insert # Line: I insert I# insert # D delete D# delete
S Commit Message Text
-----
-----Use F3 to Continue - Cancel if Title blank-----
```

1. Enter a title, which is limited to 50 characters, that summarizes the changes being committed.
2. Enter as many lines of text as required. Initially 15 lines are available for text entry; however, more can be inserted using the Insert command (for example: I 10) or the line command I (for example: I 9).
3. A line may be deleted using the delete (D) line command.
4. Optionally, entering a Tag generates a git tag command to tag the repository.

*NEXT TOPIC: Convert Repository (CONVREPO) Command*

## Convert Repository (CONVREPO) Command

This topic explains the functionality of the CONVREPO command.

The convert repository (CONVREPO) command converts a repository that has been cloned into your OMVS filesystem into a repository that ZIGI can manage. There are numerous requirements for ZIGI to manage a repository, primarily because Git has no awareness of the z/OS data sets. All Git files are stored in an OMVS filesystem.

A ZIGI-managed repository requires the following:

- A .zigi directory in the repository root directory.
- A file with the name of dsn in the .zigi directory with information on the z/OS data sets managed by ZIGI:

```
# Default DSORG and DCB info
* PO FB 80 32720
SAMPLE.REXX PO VB 255 32760
SAMPLE.PDS PO FB 80 32760
SAMPLE.TEXT PS FB 121 27951
```

- Each z/OS sequential data set must have a copy in the repository root directory.
- Each z/OS partitioned data set must have a directory in the repository root.
- Each PDS member must have a copy in the directory associated with the full PDS.
- Only files that are uppercase and conform to z/OS data set naming conventions are accepted by ZIGI to be copied to z/OS.

The conversion process:

1. Creates a .gitattributes file in the repository root directory if one does not exist. If a .gitattributes exists in the repository then it is untouched, but you should consider updating it with some of the .gitattributes that you can find in ZIGI managed repositories.
2. Creates a .zigi subdirectory.
3. Creates a .zigi/dsn file with a set of defaults.
4. Identifies files in the repository root that conform to z/OS data set naming conventions, after being translated to uppercase, and copies them to z/OS data sets using the HLQ defined when the repository was cloned.
5. Identifies directories in the repository root that conform to z/OS data set naming conventions after being translated to uppercase.
  1. Creates a PDS in z/OS.
  2. Renames the members within to remove any suffix to files with uppercase names assuming the file conforms to z/OS PDS member naming conventions.
  3. Copies renamed files to the PDS as members.
6. Files that end with .bin are treated as binary files and the .bin suffix is removed when copied to z/OS.

*NEXT TOPIC: [DIFF Command](#)*

## DIFF Command

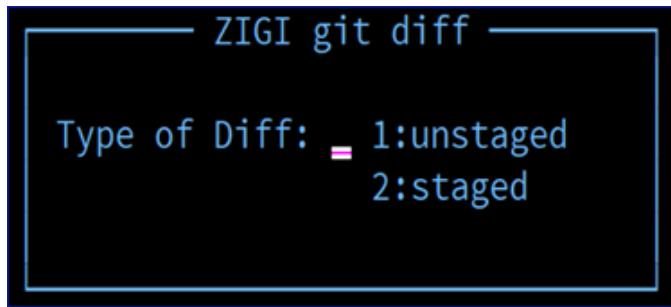
This topic explains the functionality of the DIFF command.

The DIFF command displays the pending changes prior to staging (add) or pending changes prior to a commit.

The syntax is:

- Diff (to be prompted)
- Diff 1 (for prior to staging)
- Diff 2 (for prior to commit)

The prompting panel is:



*NEXT TOPIC: [Extract Command](#)*

## Extract Command

This topic explains the functionality of the Extract command.

The **Repository Extract** screen provides you with several options working within the current repository.

The extract selection display is:

```
Repository Extract ----- (ZIGI v2r10) ----- Row 1 of 198
Command ==> [ ]                                     Scroll ==> CSR
                                                               F3
Commits to Review: 1000 (1 to 9999)

Commands Only limit table Refresh table
Options: S Display X Extract R Rollback

Sel Date/Time          Tag
    Title

- 2020/06/12 16:39:17 e5b24e7
  Remove obsolete panels
- 2020/06/12 07:34:43 752834a
  V28R2 update ----- 12 June 2020 - V28R2      - Other Items      -- Add
- 2020/06/12 02:26:25 df904e1
  V28R1 - Skels ----- Add the skels dataset that was missed before
- 2020/06/12 02:16:42 dbb5030
  V28R1 update ----- 11 June 2020 - V28R1      - User (Option 1)
- 2020/06/10 02:38:14 97f7edc
  V28R0 update ----- 09 June 2020 - V28R0      - General Resources (Opti
- 2020/06/09 08:25:22 d40647f
  V27R9 update ----- 09 June 2020 - V27R9      - User, Group, Dataset an
```

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

Use REFRESH (R) to restore the full display.

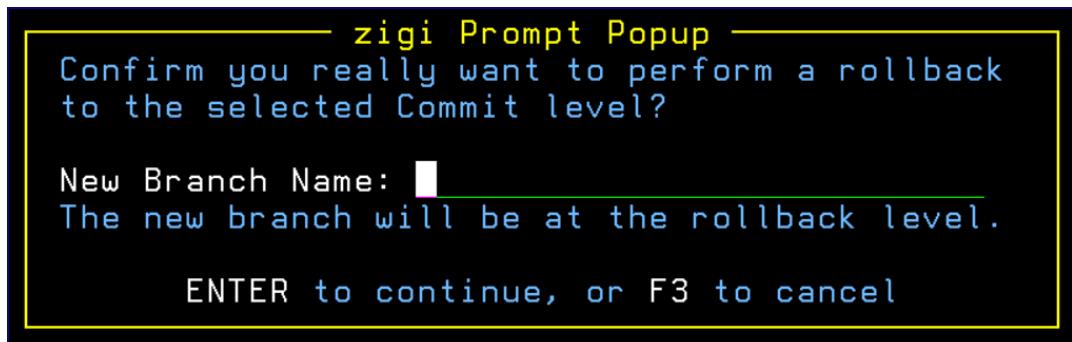
Select the individual commit to view it:

```

VIEW      ZDOLD.WORK.ZIGI.GITDATA          Columns 00001 00072
Command ==> █
***** **** Top of Data ****
000001 commit 8527c58dc08873cd714f42a5110a75181382faa4
000002 Author: lionel dyck <lbdyck@gmail.com>
000003 Date:   Sun Mar 1 09:56:37 2020 -0600
000004
000005     Update interface to GitHelp for a command
000006 -----
000007     New syntax from within ZIGI for GitHelp:  GitHelp optional-command
000008     e.g. GITH commit
000009
000010 diff --git a/.zigi/ZIGI.EXEC b/.zigi/ZIGI.EXEC
000011 index 5440b93..0e22aa1 100644
000012 --- a/.zigi/ZIGI.EXEC
000013 +++ b/.zigi/ZIGI.EXEC
000014 @@ -1,6 +1,6 @@
000015 GITHELP 19/12/06 20/02/18 01 10 08:07    909    766      0 ZIGI21
000016 SAMPLE  20/02/13 20/02/13 01 00 13:06      17      17      0 ZIGI21
000017 -ZIGI   20/02/15 20/03/01 01 99 11:35  6517  5341      0 ZIGI25
000018 +ZIGI   20/02/15 20/03/01 01 99 11:51  6517  5341      0 ZIGI25
000019 ZIGICKOT 20/01/24 20/02/29 01 18 16:17    492    470      0 ZIGI25
000020 ZIGICNVT 20/02/27 20/02/29 01 20 16:17    311    163      0 SLBD
000021 ZIGIEM  19/06/25 20/01/27 01 09 05:54      56      27      0 ZIGI21
000022 diff --git a/.zigi/ZIGI.PANELS b/.zigi/ZIGI.PANELS
000023 index 6c8f646..e1653a9 100644
000024 --- a/.zigi/ZIGI.PANELS
000025 ... L/.zigi/ZIGI.PANELS

```

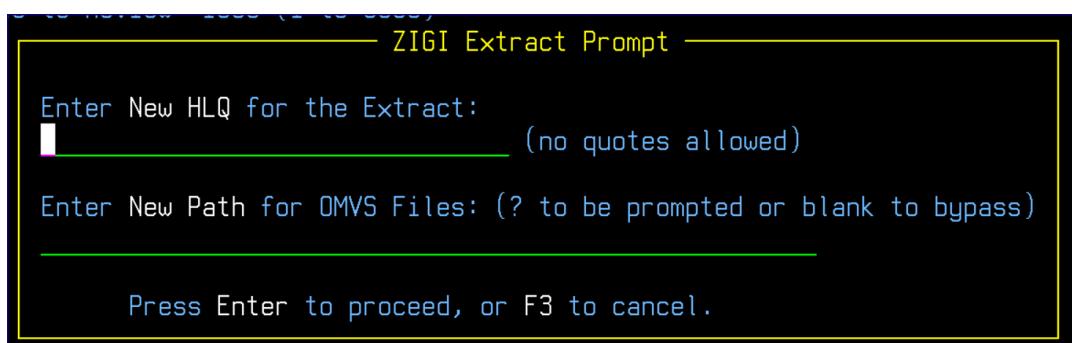
Or select using R to Rollback to that commit level:



The **zigi Prompt Popup** requires the name of a new branch and that the word ROLLBACK be entered in uppercase letters to confirm that you want to do it.

Option X is used to extract all the changed elements from all the selected commits into a set of non-Git managed data sets and OMVS directory. This can be used for packaging to distribute changes to others (for example: create a PTF). When multiple commits are selected, all changed elements from all selected commits is collected, then the most recent commit level is checked out to a temporary branch from which the changed elements are copied from.

The **ZIGI Extract Prompt** displays:



Enter an HLQ for the extracted z/OS data sets and an optional OMVS directory (must not currently exist as it is created) into which the changed elements are copied.

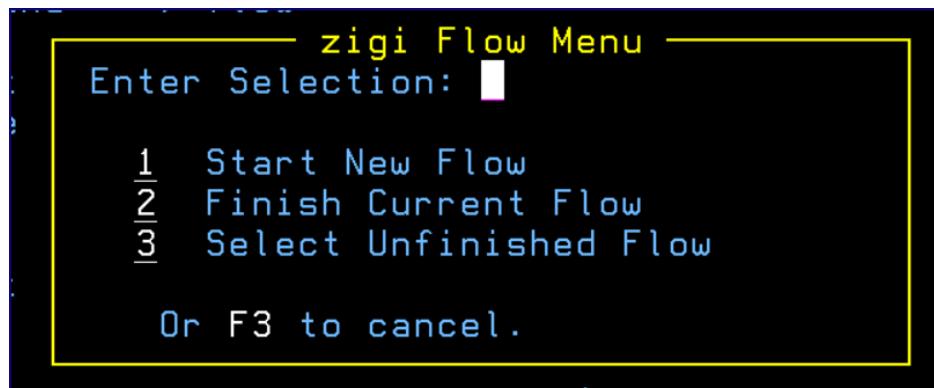
*NEXT TOPIC: [Flow Command](#)*

## Flow Command

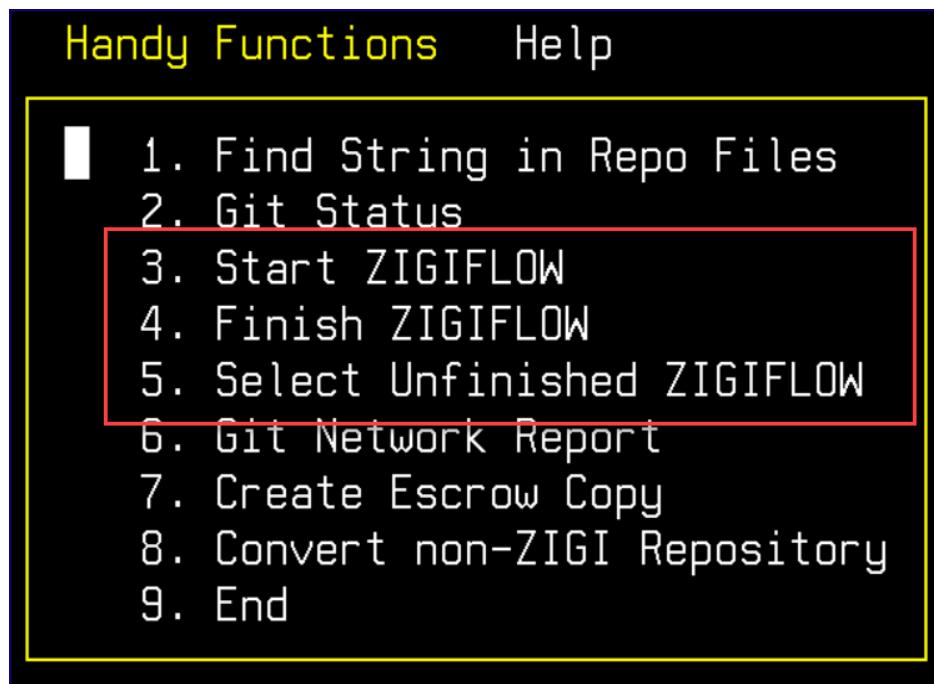
---

This topic explains the functionality of the Flow command.

To assist you in your workflow, ZIGI helps you via the ZIGI flow commands. The FLOW command displays the **zigi Flow Menu**:



The **Handy Functions** action bar menu is similar:



A new ZIGI flow can be started at any time if the workspace is clean. In other words, there can be no local changes that are not committed to the current branch.

```

----- zigi Starting new flow on repository -----
This will start a new 'zigiflow'.
Two branches will be created : zigiflow-<flowname> and
<flowname> .

You can create new flows at any time, just remember they
will all be based off of the 'master' -branch.

Once you're ready, end the flow and all commits from work
will be squashed into the feature-branch.

Flowname: flow-test (no spaces)
          (the flowname is used as branch name)

Or F3 to cancel.

```

When starting a new ZIGI flow, ZIGI prompts you for a flow name. This is used internally as a branch name so spaces are not allowed.

Upon starting a flow, ZIGI checks out the master branch, create a new branch called `zigiflow-yourFlowName`, and also creates a `yourFlowName` branch. Both of these branches are identical to master.

Then you can make changes, commit, and once done, finish the flow. Upon finishing the flow, all the commits from the `zigiflow` branch are squashed into the final branch at which point this can be merged into the master or pushed out to a remote repository so a pull request might be issued.

Due to the way `zigiflow` manages the internal branching, it's easy for you to (once the workspace for their current work is clean, for example: work-in-progress is committed to the flowbranch) start another `zigiflow` (always based on master) to perform some efixing. Once that flow is done, the resulting branch can be merged to master (or pushed for a pull request) and also be merged into the other workflow.

Switching between workflows is done via option 3 and can only be done in a clean workspace.

```

----- (zigi v2r5) ----- Row 1 of 1
Command ==> S                                         Scroll ==> CSR
Current Repository : test                           F3
Current HLQ      : ZDOLD.TEST
Current Branch   : master

Type S to work on selected flow
Press F3 to exit

S Flow                               Status
_ zigiflow-flow-test           Local
***** Bottom of data *****
```

NEXT TOPIC: [GitCmd Command](#)

## [GitCmd Command](#)

This topic explains the functionality of the GitCmd command.

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 v2r5) ----- Row 1 of 9
Command ==> █
                                         Scroll ==> CSR
                                         F3
Git repo dir: /home/zdold/test/test
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 -1 --name-only --pretty=format:""
log -m -1 --name-only
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 side of the screen).

Any Git command may be entered and it is remembered the first 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:** Git is already in the command, so you do 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.

The Git command table is shared among all repositories.

To execute multiple commands, including OMVS commands, string them together using &&. See the following figure for an example:

```

Git Commands ----- (zi
Command ==>

Git repo dir: /home/zdold/test/test

Enter any Git command (without git):
git status && ls -lat█

```

*NEXT TOPIC: [Git Help](#)*

## Git Help

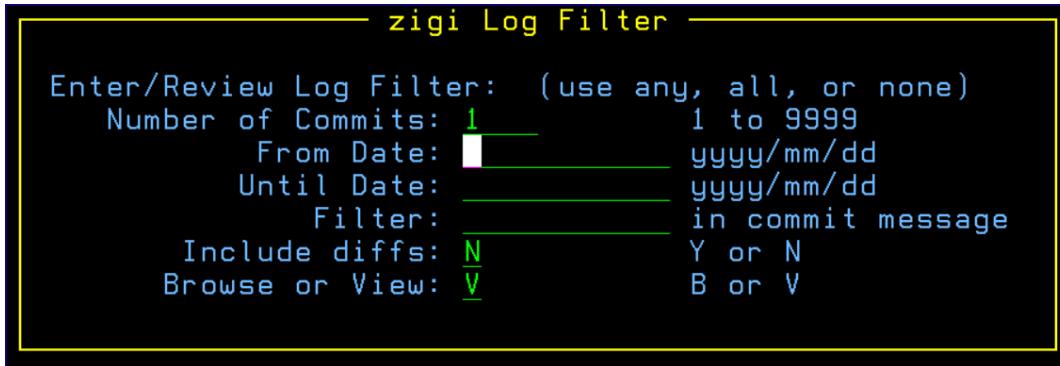
See [GITHELP](#) for more information.

*NEXT TOPIC: [GitLog Command](#)*

## GitLog Command

This topic explains the functionality of the GitLog command.

This command displays the Git log information. When requested, the **zigi Log Filter** pop-up is presented to request the amount of log information to report:



**Note:** The date format is yyyy/mm/dd, which is translated to yyyy-mm-dd and 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 is used by grep to filter commits to that string. No blanks allowed. The values entered are remembered in your ISPF Profile for repeated use. The Git log command is executed using the --cc and -m options along with the appropriate options for the above values, after which the ISPF view (or browse – no special colors with browse) is invoked on the results:

```

VIEW      ZDOLD.WORK.ZIGI.GITDATA          Columns 00001 00072
Command ==> █
***** **** Top of Data ****
000001 commit b93455ed1892c5834bc511118b12b8afb213e558
000002 Author: Lionel Dyck <lbdyck@gmail.com>
000003 Date:   Sun Mar 1 12:15:54 2020 -0600
000004
000005     1st commit
000006
000007 diff --git a/.zigi/TEST.$AB#DE.PDS b/.zigi/TEST.$AB#DE.PDS
000008 new file mode 100644
000009 index 0000000..31f458d
000010 --- /dev/null
000011 +++ b/.zigi/TEST.$AB#DE.PDS
000012 @@ -0,0 +1,5 @@
000013 +$AB@D#X 20/01/17 20/01/17 01 00 20:21    23    23    0 ZDOLD
000014 +$ABC#X 20/01/17 20/01/17 01 00 20:21    7     7    0 ZDOLD
000015 +DE$ABC 20/01/17 20/01/17 01 00 20:21    6     6    0 ZDOLD
000016 +DEF     20/01/17 20/01/17 01 00 20:21    5     5    0 ZDOLD
000017 +JKL     20/01/17 20/01/17 01 00 20:21    5     5    0 ZDOLD
000018 diff --git a/.zigi/TEST.PDS b/.zigi/TEST.PDS
000019 new file mode 100644
000020 index 0000000..83cdd0e
000021 --- /dev/null
000022 +++ b/.zigi/TEST.PDS
000023 @@ -0,0 +1,17 @@

```

[NEXT TOPIC: Grep Command](#)

## Grep Command

This topic explains the functionality of the Grep command.

The **Grep Command Entry** screen prompts for a string and then searches the local repository for that string.

```
Grep Command Entry ----- (zigi v2r5) -----
Command ==> _____
```

Git grep: bpxwunix

Dataset/File or Report View: d [D or R)

Note: The grep is NOT case sensitive.

Press F3 to exit grep prompt

The results display on the **Grep Datasets/Files** view:

```
Grep Datasets/Files ----- (zigi v2r5) ----- Row 1 of 10
Command ==> █                               Scroll ==> CSR
                                                F3
```

Grep Command: bpxwunix

Options: S Edit E Edit B Browse V View

Sel	Status	Dataset/File	Count
-	-	'ZDOLD.ZIGI25.ZIGI.EXEC(GITHHELP)'	5
-	-	'ZDOLD.ZIGI25.ZIGI.EXEC(ZIGI)'	50
-	-	'ZDOLD.ZIGI25.ZIGI.EXEC(ZIGICKOT)'	6
-	-	'ZDOLD.ZIGI25.ZIGI.EXEC(ZIGICNVT)'	7
-	-	'ZDOLD.ZIGI25.ZIGI.EXEC(ZIGIGCMD)'	6
-	-	'ZDOLD.ZIGI25.ZIGI.EXEC(ZIGIOSEL)'	4
-	-	'ZDOLD.ZIGI25.ZIGI.EXEC(ZIGISTAT)'	8
-	-	'ZDOLD.ZIGI25.ZIGI.PANELS(ZIGICC)'	1
-	-	'ZDOLD.ZIGI25.ZIGI.PANELS(ZIGIMDIR)'	1
-	-	'ZDOLD.ZIGI25.ZIGI.PANELS(ZIGINEW)'	1
***** Bottom of data *****			

This view presents each of the data sets, or data set(member), where a hit was detected along with reporting how many hits were found in each. You have the ability to Browse, Edit, or View each.

The following image shows the results for the Report view:

```
Browse                                zigi Git Messages          Lines  0000000000
Command ==> █                         Scroll ==> CSR
***** Top of Data *****
ZIGI.EXEC/GITHHELP: x = bpxwunix(cmd,,so.,se.,env.)
ZIGI.EXEC/GITHHELP: | bpxwunix commands. Needed for git. |
ZIGI.EXEC/GITHHELP: x = bpxwunix(cmd,,so.,se.)
ZIGI.EXEC/GITHHELP: rv=bpxwunix('env','env. ,,,1)
ZIGI.EXEC/GITHHELP: x = bpxwunix(c,,o.,e.)
ZIGI.EXEC/ZIGI: |                                     - Improve bpxwunix performance |
ZIGI.EXEC/ZIGI: x = bpxwunix(cmd,,so.,se.)
```

The **zigi Local Repository Summary Info** screen displays a short summary of information about the repository.

```
----- zigi Local Repository Summary Info -----
Repository      : zigi
Branch         : v2r6-pre
Remote          : origin git@github.com:lbddyck/zigi.git
Home Directory: /home/zdold/zigi
Category        : zigi
z/OS HLQ       : ZDOLD.ZIGING
Ignore Qual    : 2
Last Access   : 8 Mar 2020
Default Userid: ZIGI26
Default Push   : Y
```

*NEXT TOPIC:* [Network](#)

## Network

---

This topic explains the functionality of the Network command.

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

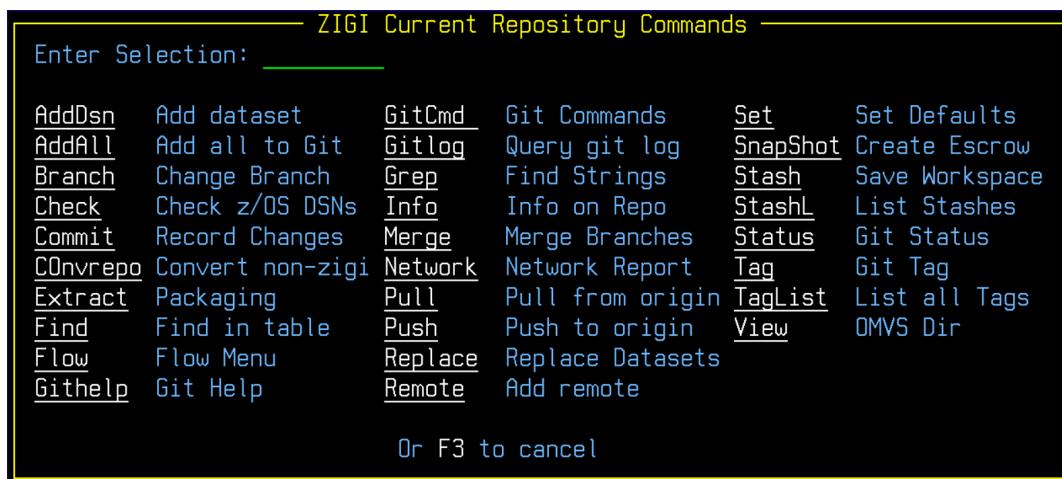
```
VIEW      SLBD.WORK.ZIGI.GITDATA          Columns 00001 00000
Command ==>                                         Scroll ==> CSR
000030 * ec0b1f8 Colorize-sorted-columns-in-ZIGIPRIM
000031 * 26f54cf Fix-ZG-bug-with-file-suffix-and-update-Release
000032 * 17a68df Release-update-wording
000033 * 3867d8f Update-ZG-usage
000034 * 1232376 Fixed-the-merge-conflicts
000035 \
000036 * f6ea68e Fix-merge-errors-on-lbd-dev-merge
000037 |
000038 // 
000039 * 04996b4 REXXFORM-ZIGI-Exec-for-consistent-flow
000040 * b800030 Update-ZIGIAU-authors-contributors-Tutorial
000041 * 46741f8 Merge-pull-request-4-from-mfsysprog-lbd-dev
000042 \
000043 * 4f81142 refactored-Work_with_repo_file
000044 * 9ed6bbb refactored-work_with_repo
000045 * 1702e52 move-code-to-functions
000046 /
000047 * 73355dd Update-to-ZG
000048 * a3b6987 Update-ZG-to-honor-the-Reset-Userid-if-defined
000049 * 29fa47f Update-to-remove-inused-code-from-ZG
```

*NEXT TOPIC:* [Options Menu Assist](#)

## Options Menu Assist

This topic explains the functionality of the **ZIGI Current Repository Commands** pop-up.

Use **o** on the command line to bring up a pop-up menu of all available commands. This is useful if you have selected to use the Pro or Hidden menu.



*NEXT TOPIC: [Pull](#)*

## Pull

Pull copies from the remote repository and replaces the updated files in the local OMVS filesystem. After the pull, the updated z/OS data sets are automatically refreshed.

*NEXT TOPIC: [Push](#)*

## Push

Push pushes the current branch to the remote repository.

*NEXT TOPIC: [Replace](#)*

## Replace

Replace replaces all of the current z/OS data sets in the repository with the data in the local repository's OMVS filesystem, including maintaining the ISPF statistics when those data sets were added to the OMVS filesystem. All refreshed partitioned data sets are 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 is reallocated as a PDSE Version 2 with the default MAXGEN. If the refreshed library is a PDSE then it is not reallocated and only the updated members are refreshed.

*NEXT TOPIC: [Remote Command](#)*

## Remote Command

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

```
Add Remote ----- (zigi v2r5) -----
Command ==> _____
Add remote [ ] _____
Default Push on Commit: Y (Y or N)
Default Userid to set prior to Commit: ZIGI25 or blank
This will execute a 'git remote add origin' followed by a
'git push -u origin master'.
Make sure the remote repository exists and you're authorized to
push to master.

Note: This is best done on a 'brand new' repository :-)

Press Enter to continue F3 to cancel
```

NEXT TOPIC: [Set Command](#)

## Set Command

This topic explains the functionality of the Set command.

The Set command is used to define default actions for you.

```
ZIGI Set Defaults for Repository
Command ==> _____
Review/Update Repository Settings:

Name of the Repository: TestZIGI
Category for Repository: testing (optional)
PREFIX for datasets: SLBD.TESTZIGI (no quotes)
Default Push on Commit: Y (Y or N)
Default Userid to set prior to Commit: [ ] _____ or blank
Disable File Extension Prompt on Dataset Add: N (Y or N)
Prime Repository with zginstall.rex: N (Y or N)

ENTER to register updates and F3 to Cancel or Leave the panel.
```

**Name of the Repository** is the name of the OMVS directory where the repository resides and if changed only results in the name of the repository in the Local Repositories table. The directory is not changed.

**Category for Repository** is an arbitrary description for the repository that allows you to categorize, or group, repositories.

**PREFIX for datasets** is used to define the z/OS data set HLQ that was used when the repository was created or cloned. The Prefix may be changed, and all z/OS data sets are renamed, and depending upon the qualifiers to ignore (see the INFO display) it is possible the OMVS files and directories may be renamed as well. To change the Prefix requires that the new prefix have the same number of qualifiers as the original prefix. If OMVS files and/or directories are changed, then git is also updated to be aware of the rename.

**Default Push on Commit** has an option to push after the commit. This setting provides a default, which you may override on the Commit panel at anytime.

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

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

The option to **Disable File Extension Prompt on Dataset Add** can be enabled or disabled here.

If the repository is to be distributed to others who may not have ZIGI installed, use the **Prime Repository with zgininstall.rex** option to place a zgininstall.readme and the zgininstall.rex into the repository. See [Appendix B: Using ZGINSTALL.REX](#) for additional information.

NEXT TOPIC: [Snapshot](#)

## Snapshot

Snapshot creates an Escrow set of data sets 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 v2r5] -----
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.
```

NEXT TOPIC: [Stash](#)

## Stash

Stash saves (pushes) your current Git workspace and restores the workspace to the last commit level. This allows you to quickly switch what you're working on and work on something else without forcing a commit.

**Note:** A stash is never pushed to the remote server. Enter a short description on the **zigi Prompt Popup** screen since you may have multiple stashes in your project.

```
----- zigi Prompt Popup -----
Enter a comment for the stash:
=> _____
```

NEXT TOPIC: [Stash List \(STASHL\) Command](#)

## Stash List (STASHL) Command

This topic explains the functionality of the Stash List (STASHL) command.

The Stash list (STASHL) command lists all stashes in the current repository and provides a number of options.

```
Commands Help
Stash Summary ----- (zigi v2r ----- Row 1 of 1
Command ==> █                               Scroll ==> CSR
                                                F3
Sel Stash-ID   Comments
stash@{0}  On master: 1 Mar 2020 14:29:31 test stash
***** Bottom of data *****
```

Available commands with Stash List are currently limited to one at this time. The Clear command removes all stashes in the current repository. Use this only if you are positive you don't need them anymore. Enter a / in the row selection field to view available line selections:

```
zigi Stash Row Selection Options
Enter Selection: █
Branch (B) Create a new branch from stash
Diff (D) Show the diff's for the stash
Pop (P) Restore a stash
Remove (R) Remove (Drop) the stash
Show (S) Show stash summary
Or F3 to cancel.
```

This section includes the following topics:

- [Branch](#)
- [Diff](#)
- [Pop](#)
  - [Pop Conflict Resolution](#)
- [Remove](#)
- [Show](#)

### Branch

Branch creates and checks out a new branch starting from the commit at which the *stash* was originally created and then applies the changes recorded in the stash to the new working tree and index. If that succeeds, it then drops the stash. This is useful if the branch on which you ran git stash save has changed enough that git stash apply fails due to conflicts. Since the stash entry is applied on top of the commit that was HEAD at the time git stash was run, it restores the originally stashed state with no conflicts.

*NEXT TOPIC:* [Diff](#)

### Diff

Diff uses the Stash Show command with option -p to display the stash summary along with the differences between the last commit and the stash.

*NEXT TOPIC:* [Pop](#)

## Pop

Pop removes a single stashed state from the stash list and applies it on top of the current working tree state (for example: do the inverse operation of Git stash save). The working directory must match the index. For simplicity sake, ZIGI validates that the current workspace is clean before performing a stash pop. This is because numerous conflicts could arise that would require manual intervention and the authors were unable to find a reliable methodology to resolve those conflicts programmatically.

*NEXT TOPIC:* [Pop Conflict Resolution](#).

### Pop Conflict Resolution

There are two potential solutions that may be used to create a clean workspace:

- Create a new stash from the workspace, and then pop the desired stash.
- Perform the following actions:
  1. AddAll to add all untracked and modified data sets and files to the Git index staging area.
  2. Commit those changes.

After performing either of the above, the pop should work.

*NEXT TOPIC:* [Remove](#)

## Remove

Remove drops the selected stash from the current repository.

*NEXT TOPIC:* [Show](#)

## Show

Show displays a summary of the differences between the stash level and the most recent commit.

*NEXT TOPIC:* [Status Command](#)

## Status Command

---

The Status command displays the Git status command results:

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

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

*NEXT TOPIC:* [Tag](#)

## Tag

---

Tag adds a Git Tag to the current repository and the tag is pushed to the remote server.



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 are 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.

*NEXT TOPIC:* [TagList](#)

## TagList

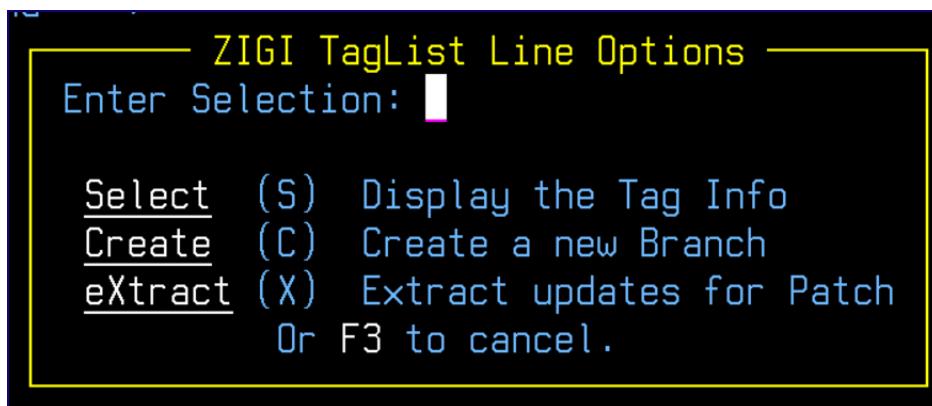
---

This topic explains the functionality of the TagList command.

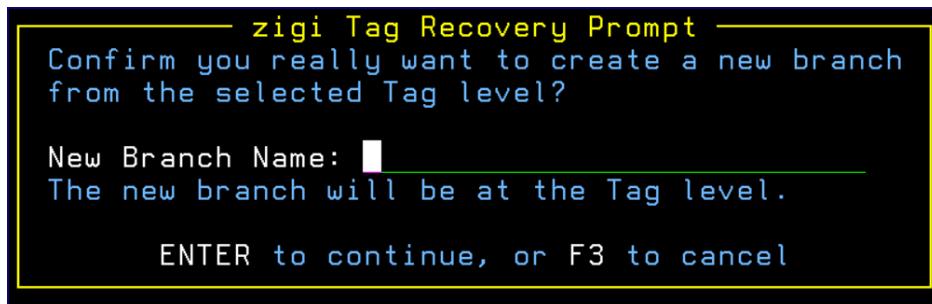
TagList displays a selection table of all the Tags.

Commands	Help			
Tag Summary	(zigi v2r5)	Row 1 of 48		
Command ==>		Scroll ==>	CSR	F3
Sel	Date/Time	Tag		
	Title			
-	2020 Mar 1 08:32:16	v7r3		
	V7R3 Update	----- 01 Mar 2020 - V7R3	- User (Option 1)	* Added
-	2020 Feb 28 13:48:47	v7r2		
	v7r2 updates	----- 28 Feb 2020 - V7R2	- User (Option 1)	* Tr
-	2020 Feb 28 11:25:52	v7r1		
	V7R1 Update	----- 28 Feb 2020 - V7R1	- User (Option 1)	* Plac
-	2020 Feb 28 10:02:58	v7r0		
	v7r0 updates	----- 27 Feb 2020 - V7R0	- User (Option 1)	* Ad
-	2020 Feb 28 08:33:55	v6.9		
	V6R9	---- 27 Feb 2020 - V6R9	- User (Option 1)	* Added code to allo
-	2020 Feb 28 06:25:28	v6.8		
	V6R8 Updates	----- 27 Feb 2020 - V6R8	- User (Option 1)	- Ad
-	2020 Feb 28 01:21:52	v6.7		
	V6R7 Update	----- 27 Feb 2020 - V6R7	- User (Option 1)	* Adju
-	2020 Feb 27 14:02:48	v6.5		
	V6R5 update	----- 27 Feb 2020 - V6R5	- User (Option 1)	* Remo
-	2020 Feb 27 11:59:20	v6.4		

The table may be filtered using the Only command (for example: only x) and Refresh recreates the table with all entries. A row selection of S displays the tag details while a selection of C creates a new branch based on the tag level of the repository. This not only changes the OMVS filesystem files to match the tag level file, it also replaces all z/OS data sets to the tag level. Selection X extracts all the changed elements from all the selected tags into a non-Git managed set of z/OS data sets and OMVS directory. This can be used for packaging to distribute only the updated elements (for example: PTF). A row selection option of / brings up the **ZIGI TagList Line Options** pop-up:



The **zigi Tag Recovery Prompt** pop-up asks for the name of a new branch for the tag level code:



*NEXT TOPIC: [View](#)*

## View

---

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

## Row Selections

---

This section includes the following topics:

- [Select Option](#)
- [Add Option](#)
- [Browse and View](#)
- [Diff](#)
- [History](#)
- [Undo \(U\)](#)
- [Remove \(RM\)](#)
- [Rename \(RN\)](#)
- [/ Row Selection Prompt](#)

### Select Option

The Select option, or point-and-shoot, performs one of two different actions depending upon the type of data set being selected. After editing a sequential data set, if the data has changed, it is copied down to the OMVS file.

Open an Edit member list for a PDS. See [The ZIGI PDS Member List](#) for more information.

If the selected file is an OMVS directory, then ZIGI opens that directory and displays it using the Current Repository routines.

*NEXT TOPIC: [Add Option](#)*

## Add Option

The Add option adds the data set (if sequential), or any modified members of a partitioned data set, to the Git index making them available to be committed.

*NEXT TOPIC: [Browse and View](#)*

## Browse and View

The Browse and View options use ISPF Browse or ISPF View on the requested data set or file. If the requested data set or file is an OMVS subdirectory, then the action is the same as Select.

*NEXT TOPIC: [Diff](#)*

## Diff

Diff displays the Git differences between the current data set, or file, and the Git commit version of the file. See [Diff Option](#) for more information.

*NEXT TOPIC: [History](#)*

## History

History views a history of commits for the data set, or file, with an option to view the source as of the historical point in time. See [History Option](#) for more information.

*NEXT TOPIC: [Undo \(U\)](#)*

## Undo (U)

Undo (U) reverts any changes made to a data set or OMVS file before a commit. This means the status is [M], {M}, or [MM].

**Note:** Undo does not work on a full PDS, it only works on PDS members.

*NEXT TOPIC: [Remove \(RM\)](#)*

## Remove (RM)

Remove (RM) deletes the z/OS data set, the OMVS file, or the OMVS directory. It also deletes it from Git management. This can be undone before, or after, commit processing using Rollback.

*NEXT TOPIC: [Rename \(RN\)](#)*

## Rename (RN)

The Rename (RN) command renames the z/OS data set, the OMVS file, or the OMVS directory and then updates Git.

**Note:** Git does not retain any of the Git histories from the original name.

*NEXT TOPIC: [/ Row Selection Prompt](#)*

## / Row Selection Prompt

If a / is entered for a row selection, the **zigi Current Repository Selections** pop-up aids you:

```
----- zigi Current Repository Selections -----
DSN/File: 'ZDOLD.RACFADM.EXEC'

Enter Selection: █

Add      (A) Add the dataset/file to the git Index
Browse   (B) Browse the dataset/file
Edit     (E) Edit the dataset/file
Diff     (D) Show the Diff's
History  (H) Display Commit History
Remove   (RM) Remove the dataset/file
Rename   (RN) Rename the dataset/file
Select   (S) Edit the dataset/file
Undo    (U) Undo an UnCommitted change
View    (V) View the dataset/file
Or F3 to cancel.
```

NEXT TOPIC: [The ZIGI PDS Member List](#)

## The ZIGI PDS Member List

---

From the Repository display, if the selected data set is a PDS, then the **PDS Member List** displays:

Menu    Help	
PDS Member List -----	(zigi v2r5) ----- Row 1 of 17
Command ==> █	Scroll ==> CSR
Current Dataset : 'ZDOLD.RACFADM.EXEC'	F3
Current Repository : /home/zdold/racfadm	
Current Branch : master	
S	Member    Status
	Type    Size    Mod-Date    Mod-Time    Userid
	T    27    20/02/21    13:58    RACFADM
	T    49    20/03/01    08:29    RACFADM
	T    230    20/03/01    08:29    RACFADM
	T    13    20/03/01    08:29    RACFADM
	T    25    20/03/01    08:29    RACFADM
	T    1013    20/03/01    08:29    RACFADM
	T    240    20/03/01    08:29    RACFADM
	T    307    20/03/01    08:29    RACFADM
	T    850    20/03/01    08:29    RACFADM
	T    730    20/03/01    08:29    RACFADM
	T    38    20/03/01    08:29    RACFADM
	T    46    20/03/01    08:29    RACFADM
	T    229    20/03/01    08:29    RACFADM
	T    16    20/03/01    08:29    RACFADM
	T    989    20/03/01    08:29    RACFADM
	T    217    20/03/01    08:29    RACFADM
	T    54    20/03/01    08:29    RACFADM
***** Bottom of data *****	

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

This chapter includes the following topics:

- [Action Bar Menu](#)
- [Locate Command](#)
- [Commit](#)
- [GitLog](#)
- [Grep](#)
- [Sort Command](#)
- [Status](#)
- [Reset-IDs Command](#)

- Options (O) Command
- Member Select Option
  - Add Option
  - AddBin Option
  - Browse Options
  - Diff Option
  - History Option
  - Commit View
  - Source View
  - Recovery
- Remove and Rename
- Undo
- View
- / Row Selection Prompt

## Action Bar Menu

---

This topic provides a screenshot of the action bar menu.



*NEXT TOPIC:* [Locate Command](#)

## Locate Command

---

The locate command finds a member that starts with or matches the string provided (for example: L GIT finds the member GITHELP).

*NEXT TOPIC:* [Commit](#)

## Commit

---

See [Commit Command](#).

*NEXT TOPIC:* [GitLog](#)

## GitLog

---

See [GitLog Command](#).

*NEXT TOPIC:* [Grep](#)

## Grep

---

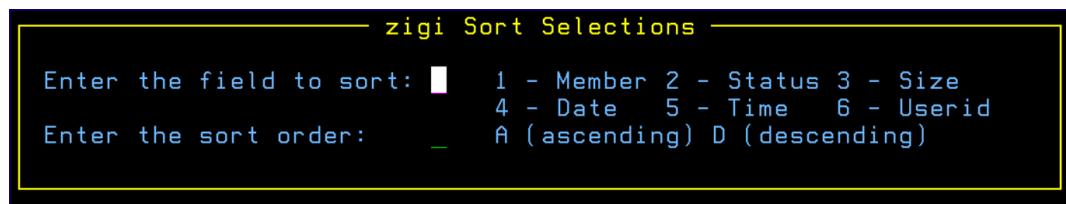
See [Grep Command](#) command.

*NEXT TOPIC:* [Sort Command](#)

## Sort Command

---

The Sort command sorts one column in either **A (ascending)** or **D (descending)** order. When entered without parameters, the **zigi Sort Selections** panel assists you:



The sort syntax is: SORT field order

If the order parameter is blank, then it defaults to **A (ascending)**.

The field names may be abbreviated:

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

*NEXT TOPIC:* [Status](#)

## Status

---

See [Status Command](#) for more information.

*NEXT TOPIC:* [Reset-IDs Command](#)

## Reset-IDs Command

The Reset-IDs command prompts to change the userids of all members to a new userid. This is useful to hide your userid or to make all userids conform to a standard for promoted elements.

```
----- zigi Reset ID Prompt -----
Enter the userid to be set for all members:
Userid: █
```

*NEXT TOPIC: Options (O) Command*

## Options (O) Command

The Options (O) command displays the **ZIGI PDS Member Commands** menu to assist you if you are 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
GitCmd       Issue Git Commands
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.
```

*NEXT TOPIC: Member Select Option*

## Member Select Option

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

This section includes the following topics:

- [Add Option](#)
- [AddBin Option](#)

- [AddForce \(AF\) Option](#)
- [Browse Options](#)
- [Diff Option](#)
- [History Option](#)
  - [Commit View](#)
  - [Source View](#)
  - [Recovery](#)
- [Ignore](#)
- [Remove and Rename](#)
- [Undo](#)
- [View](#)
- [/ Row Selection Prompt](#)

## Add Option

The Add option adds the member, as text, to the Git index making it eligible to be committed.

*NEXT TOPIC:* [AddBin Option](#)

## AddBin Option

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

*NEXT TOPIC:* [AddForce \(AF\) Option](#)

## AddForce (AF) Option

The AddForce (AF) option forcefully adds a member if the member had been previously ignored.

*NEXT TOPIC:* [Browse Options](#)

## Browse Options

Browse opens the member using ISPF Browse.

*NEXT TOPIC:* [Diff Option](#)

## Diff Option

Diff option compares the active file in the OMVS filesystem with the last index for the file.

```

VIEW      ZDOLD.WORK.ZIGI.GITDATA          Columns 00001 00072
Command ==> █
***** **** Top of Data ****
000001 diff --git a/TEST.PDS/AA1 b/TEST.PDS/AA1
000002 index 77122da..a36ce9c 100644
000003 --- a/TEST.PDS/AA1
000004 +-- b/TEST.PDS/AA1
000005 @@ -2,10 +2,6 @@ Line 1 added
000006 Line 1 added
000007 Line 1 added
000008 Line 1 added
000009 -Line 1 added
000010 -Line 1 added
000011 -Line 1 added
000012 -Line 1 added
000013 Line 2 added
000014 Line 3 added
000015 Line 4 added
***** **** Bottom of Data ****

```

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

*NEXT TOPIC: History Option*

## 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:

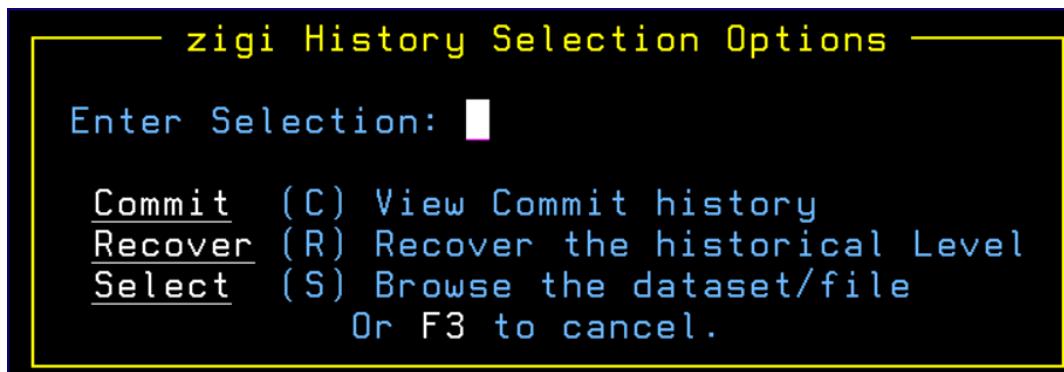
```

Help
----- (ZIGI v3r00) ----- Row 1 to 15 of 261
Command ==> █ Scroll ==> CSR
F3
Dataset/File: RACFADM.EXEC/RACFADM

Sel Date/Time of Commit Subject
-
2020/07/12 03:23:05 Update-zginstall.rex
-
2020/07/08 11:51:30 V32R7-update
-
2020/07/06 10:33:56 V32R6-Update
-
2020/06/30 15:37:31 V31R8-update
-
2020/06/30 04:36:16 V31R7-update
-
2020/06/29 09:45:45 V31R1-update
-
2020/06/27 07:16:18 Use-improved-zginstall.rexx
-
2020/06/25 08:42:50 V30R6-update
-
2020/06/24 14:24:05 V30R5-update
-
2020/06/22 14:05:19 V30R2-update
-
2020/06/22 11:13:27 V30R1-update
-
2020/06/22 11:13:27 V30R1-update
-
2020/06/21 05:56:10 V30R0-update
-
2020/06/20 09:02:02 V29R9-update
-
2020/06/20 06:58:41 Add-zginstall-readme

```

The row selection of / displays the **zigi History Selection Options** to guide the selection of the row command:



*NEXT TOPIC:* [Commit View](#)

### Commit View

The Commit view can be used to see the changes made with a specific commit.

```

VIEW      ZDOLD.WORK.ZIGI.GITDATA          Columns 00001 00072
Command ==> █                                         Scroll ==> CSR
***** **** Top of Data ****
000001 commit cdad252bbf6a57d14c9a0804ba40c689004e1a48
000002 Author: lionel dyck <lioneld@21cs.w.com>
000003 Date: Thu Feb 6 09:11:51 2020 -0600
000004
000005     Correct file contents
000006     -----
000007     accidentally put rexx in panels and panels in rexx
000008     ---- fixing
000009
000010 diff --git a/.zigi/RACFADM.EXEC b/.zigi/RACFADM.EXEC
000011 new file mode 100644
000012 index 0000000..810ccf4
000013 --- /dev/null
000014 +++ b/.zigi/RACFADM.EXEC
000015 @@ -0,0 +1,15 @@
000016+$DIR 20/01/11 20/01/21 01 09 20:50    15    13    0 RACFADM
000017 +RACFADM 20/01/20 20/02/06 01 07 09:10    46    37    0 SLBD
000018 +RACFAUTH 20/01/03 20/02/06 20 38 09:06   204   217    0 RACFADM
000019 +RACFCLSG 20/01/03 20/02/06 20 67 09:06   848   790    0 RACFADM
000020 +RACFCLSR 20/01/03 20/02/06 20 41 09:06   168   184    0 RACFADM
000021 +RACFCLSS 20/01/03 20/02/06 20 39 09:06   226   244    0 RACFADM
000022 +RACFDNSN 20/01/03 20/02/06 20 88 09:06   670   627    0 RACFADM
000023 +RACFGRP 20/01/03 20/02/06 20 99 09:06   622   504    0 RACFADM
000024 +RACFMSGC 20/01/03 20/02/06 20 33 09:06   33    39    0 RACFADM
000025 +RACFMSGS 20/01/03 20/02/06 20 27 09:06   44    47    0 RACFADM
000026 +RACFPRM 20/01/20 20/02/06 20 21 09:06   222   221    0 RACFADM

```

*NEXT TOPIC:* [Source View](#)

### Source View

This view of the source is from a specific commit. It is placed into a z/OS data set to allow you to use the ISPF Edit Compare, Create, and Copy commands, among other commands.

```

VIEW      ZDOLD.WORK.ZIGI.GITDATA          Columns 00001 00072
Command ==> █
***** **** Top of Data ****
==MSG> Historical view of element: ZDOLD.RACFADM.EXEC(RACFADM)
==MSG> From: lionel dyck on Fri 2020 Feb 21 05:38:38 -0600
==MSG> All ISPF View commands are available, including Compare, Create and Copy
==MSG> -----
000001 /*%NOCOMMENT===== REXX =====*/
000002 /* PURPOSE: RACFADM - Display Menu - LIBDEF and ALTLIB datasets */
000003 /*-----*/
000004 /* FLG YYMMDD USERID DESCRIPTION */
000005 /* --- ----- ----- */
000006 /* @A1 200120 LBD     Added code for sites using qual. PLIB/MLIB */
000007 /* @A0 200120 LBD     Created REXX */
000008 /*-----*/
000009 Address ISPEexec
000010 call Setup
000011 'Select Panel(racfmenu)'
000012 'libdef ispmlib'
000013 'libdef isppplib'
000014
000015 Address TSO 'altnlib deact app(exec)'
000016 EXIT 0
000017 /*-----*/

```

*NEXT TOPIC:* [Recovery](#)

## Recovery

Recovery prompts for a recovery data set if that data set should be allocated new and a confirmation of the recovery. Recovery can occur in the active data set if desired but with an additional confirmation prompt.

```

----- zigi History Recovery -----

Recover a historical level of:
'ZDOLD.RACFADM.EXEC(RACFADM)'

Enter the recovery dataset:
'ZDOLD.RACFADM.EXEC(RACFADM)'

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.

```

*NEXT TOPIC:* [Ignore](#)

## Ignore

The Ignore (I) option causes a member to be ignored by Git. The member is not included in the repository when it is pushed, cloned, or merged.

*NEXT TOPIC:* [Remove and Rename](#)

## Remove and Rename

See [Remove \(RM\)](#) and [Rename \(RN\)](#) which applies, in this case, just to the PDS Member.

*NEXT TOPIC:* [Undo](#)

## Undo

Undo reverts any changes made to a PDS member before a commit.

This means the status is [M], {M}, or [MM].

NEXT TOPIC: [View](#)

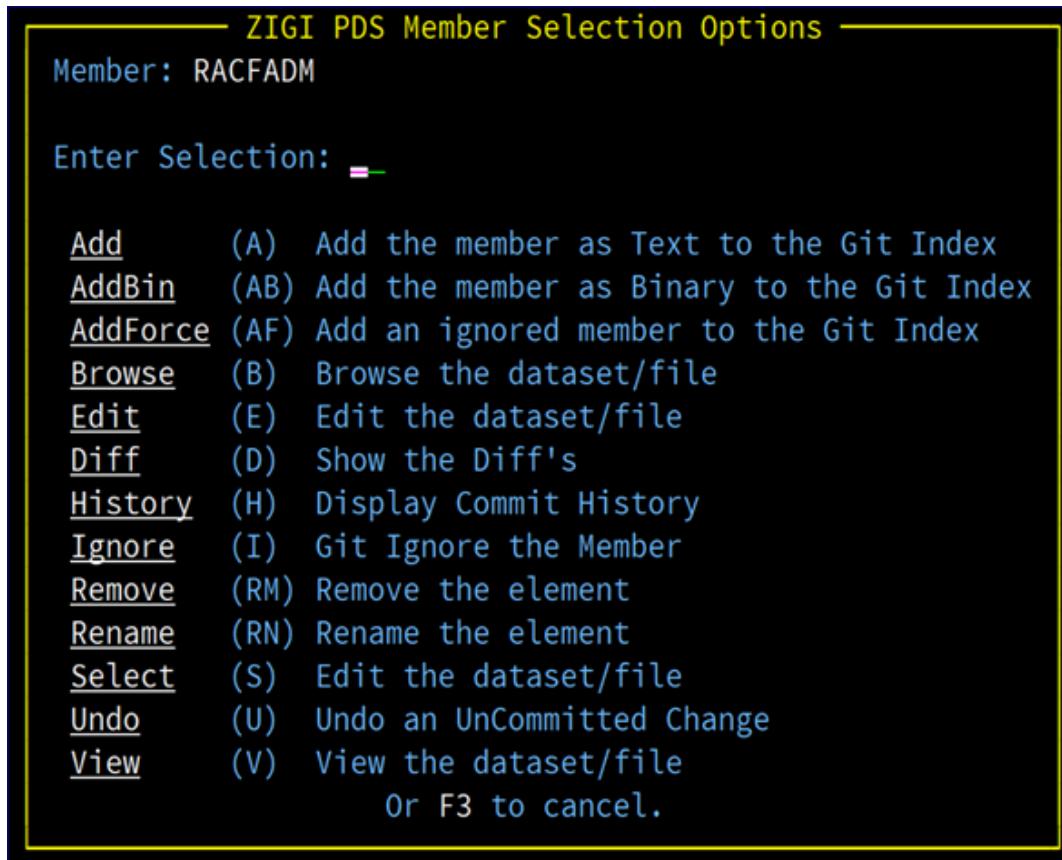
## [View](#)

View the data set or file using ISPF View.

NEXT TOPIC: [/ Row Selection Prompt](#)

## [/ Row Selection Prompt](#)

If a / is entered for a row selection, the **ZIGI PDS Member Selection Options** menu aids you:



NEXT TOPIC: [Typical ZIGI Activities](#)

## [Typical ZIGI Activities](#)

---

ZIGI is intended to be used as the interface to Git. All development activity should be performed outside of ZIGI using your favorite ISPF-based tools. Once a development phase is complete, ZIGI can be activated to commit and push the update.

This chapter includes the following topics:

- [Creating a New Repository \(Local and Remote\)](#)
- [Adding a Remote Repository](#)
- [Updating a Local Repository](#)
- [Pulling Updates from the Remote Repository to Update the Local Repository](#)
- [Pushing Updates to the Remote Repository](#)
- [Changing to a Different Existing, Or New, Branch](#)

## Creating a New Repository (Local and Remote)

---

There are two ways of creating a (Git-based) repository with ZIGI:

- Cloning a remote repository or creating one from scratch locally.
- Using the create command to create a new (local) repository.

This performs a git init *repoName* in the chosen directory. A basic .gitattributes is created and added/committed to your repository. This is there to ensure that the encoding of the various Git-managed files are in the correct format. After creating the local repo (ZIGI takes care of your .gitattributes file) you can ADD data sets to the repository. (See [AddAll Command](#) command).

Adding files to ZIGI is not the same thing as adding them to the Git repository. Once a data set has been added to ZIGI, the data sets (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 add (A) line command.

*NEXT TOPIC:* [Adding a Remote Repository](#)

## Adding a Remote Repository

---

ZIGI currently only supports a single remote (origin) per repository. 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 is performed.

*NEXT TOPIC:* [Updating a Local Repository](#)

## Updating a Local Repository

---

### About this task



**Attention:** This topic is intentionally left blank. The authors are looking for input from those who use the ZIGI product to write the information for this topic.

### Procedure

## Pulling Updates from the Remote Repository to Update the Local Repository

---

### About this task



**Attention:** This topic is intentionally left blank. The authors are looking for input from those who use the ZIGI product to write the information for this topic.

### Procedure

## Pushing Updates to the Remote Repository

---

## About this task



**Attention:** This topic is intentionally left blank. The authors are looking for input from those who use the ZIGI product to write the information for this topic.

## Procedure

### Changing to a Different Existing, Or New, Branch

Changing to a different branch requires the current branch be clean. Clean means that all changes have been committed. Once the branch is clean, use the Branch command and either select an existing branch using C (for checkout) or enter the name of a new branch.

```
----- (zigi v2r5) ----- Row 1 to 2 of 2
Command ==> I                                         Scroll ==> CSR
Current Repository : test                           F3
Current HLQ      : SLBD.TEST
Current Branch   : test1

Type C to checkout an existing branch
Type D to locally delete a branch
Press F3 to exit

Or create a new branch: _____
```

S	Branch	Status
	master	Local
	test2	Local

\*\*\*\*\* Bottom of data \*\*\*\*\*

When changing to an existing branch:

1. Git updates the Git index and all the files in the workspace to the state in the selected branch.
2. ZIGI then replaces all the existing z/OS data sets with the OMVS files that correspond to the branch.

When changing to a new branch:

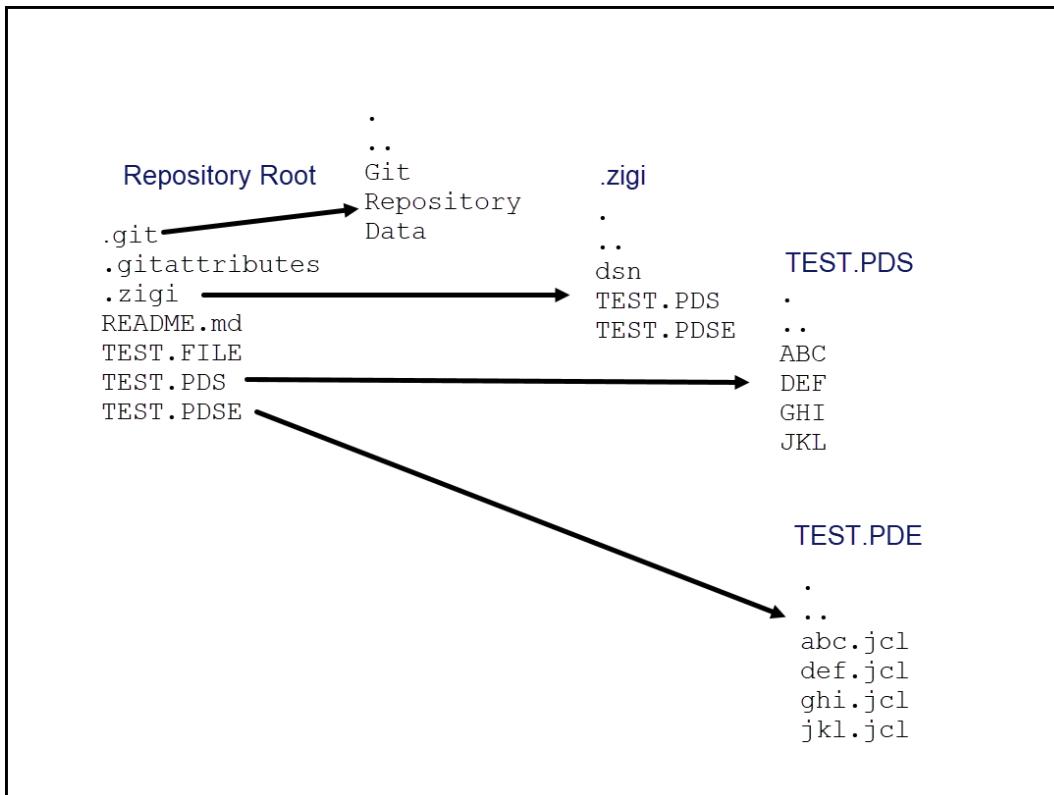
1. Git creates a new Git index for the new branch with the state of the current files.
2. ZIGI updates its status to use the new branch.

*NEXT TOPIC: [What is Happening Under the Covers?](#)*

## 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, you are informed that the current branch is behind and needs updating (pull). ZIGI also checks all the z/OS data sets and the OMVS filesystem for updates, marking the z/OS data sets if there are modifications in play that need to be added to the index, committed, and pushed. If the data set is a sequential data set, it is always copied to an OMVS file so that Git can detect changes. For a PDS, if the member is new then it is copied to the OMVS directory that maps to the PDS. If a file extension has been defined for the PDS, then the member is copied using the file extension. The extension is defined when the data set is added to ZIGI and stored in the .zigi/dsn file.

This picture provides an example of a ZIGI OMVS filesystem layout.



When a PDS 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 any new or changed members and then flags them. 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 data sets are added to ZIGI, an OMVS file is updated with the DCB information for that data set. This file is then used during a replace, clone, or pull operation to correctly allocate the z/OS data sets. The space for those data sets is dynamically determined based on the OMVS filesystem usage and are allocated as PDSE version 2 libraries with a zero MAXGEN.

*NEXT TOPIC: [ZG - ISPF Command](#)*

## ZG - ISPF Command

There are times when an element is being edited (updated) within a PDS that is part of a ZIGI managed repository. After completing the update, just that element needs to be processed by Git. To edit an element, use the ZG command.

The syntax is: **ZG *data set(member)*** option

Where: *data set(member)* is the element to be quickly processed by Git.

Options:

Blank	Prompt for Option
A	Git Add
AC	Git Add and Commit
ACP	Git Add, Commit, and Push

For ease of use, `zg` may be entered on the ISPF member list row command next to the member name to be processed.

<u>Menu Functions Confirm Utilities Help</u>						
EDIT SLBD.ZIGING.ZIGI.EXEC			Row 0000001 of 0000017			
Command ==> <u>zg</u>			Scroll ==> CSR			
Name	Prompt	Size	Created	Changed	ID	
GITHHELP		909	2020/03/11	2020/03/11 07:26:00	ZIGI26	
SAMPLE		17	2020/02/13	2020/02/13 13:06:00	ZIGI21	
ZG		1403	2020/04/07	2020/04/07 07:58:16	ZIGI28	
<u>zg</u>	<u>ZIGI</u>	7135	2020/02/15	2020/04/04 15:48:04	SLBD	
ZIGICHG		47	2020/03/11	2020/03/18 07:29:00	ZIGI26	
ZIGICKOT		516	2020/01/24	2020/03/27 09:19:00	ZIGI27	
ZIGICNVT		6999	2020/02/27	2020/03/28 03:22:00	ZIGI27	
ZIGIEM		56	2020/03/11	2020/03/11 07:26:00	ZIGI26	
ZIGIEME		87	2020/03/11	2020/03/11 07:26:00	ZIGI26	
ZIGIEMS		81	2020/03/11	2020/03/11 07:26:00	ZIGI26	
ZIGIGCMD		313	2020/03/11	2020/03/11 07:26:00	ZIGI26	
ZIGIMRGM		112	2020/03/11	2020/03/11 07:26:00	ZIGI26	
ZIGIOSEL		444	2020/03/11	2020/03/11 07:26:00	ZIGI26	
ZIGIRCSD		171	2020/03/11	2020/03/11 07:26:00	ZIGI26	
ZIGIREFD		155	2020/03/17	2020/03/18 11:04:00	ZIGI27	
ZIGISTAT		432	2020/03/06	2020/03/18 08:24:00	ZIGI26	
ZIGIVMAC		56	2020/03/11	2020/03/11 07:26:00	ZIGI26	
**End**						

If the `zg` command has not been used before for this data set, then a list of all ZIGI-managed repositories are presented to select from:

Update Git Repository						
Command ==> <u>zg</u>			Row 1 to 9 of 20			
Selection: A Add AC Add/Commit ACP Add/Commit/Push 0 Open zigi						
Repository	Category	Last Date				
racfadm	racfadm	10 Apr 2020				
omvscopy	tools	7 Apr 2020				
zigi	zigi	7 Apr 2020				
pdsegen	pdsegen	6 Apr 2020				
sfc	sfc	4 Apr 2020				
test	testing	1 Apr 2020				
ztest	zTest	30 Mar 2020				
dbb	dbbgitispf	30 Mar 2020				
lbd_test	testing	28 Mar 2020				

If `zg` has been used with that data set before, then the repository prompt is bypassed, and the **zigi Git Option** pop-up displays:



There is an additional option available via the pop-ups, O, which opens the repository.

*NEXT TOPIC: ZGBATCH - Execute ZIGI in Batch*

## ZGBATCH - Execute ZIGI in Batch

There may be projects where there is a need to schedule ZIGI to check for updated and new elements in a repository's z/OS data sets, which is where ZGBATCH is used.

The following is sample JCL:

```

//ZGBATCH JOB XXX,'ZIGI',REGION=0M,NOTIFY=&SYSUID
//OUT    OUTPUT DEFAULT=YES,JESDS=ALL,OUTDISP=(HOLD,HOLD)
//*- -----
//*- INVOKE ISPF IN BATCH *
//*- -----
//ISPF    EXEC PGM=IKJEFT1B,DYNAMNBR=50
//*- -----
//** SYSEXEC IS WHERE THE CMD RESIDES *
//*- -----
//SYSEXEC DD DISP=SHR,DSN=zigi.exec
//*- -----
//** MLIB AND TLIB FOR THE REAL ISPF *
//*- -----
//ISPMLIB DD DISP=SHR,DSN=ISP.SISPMENU
//ISPTLIB DD DISP=SHR,DSN=ISP.SISPTENU
//*- -----
//** TEMPORARY PANELS, SKELETONS, AND TABLES ISPF LIBRARIES *
//** ALLOCATED TO VIO *
//*- -----
//ISPPLIB DD DISP=(,DELETE),SPACE=(TRK,(1,1,1)),UNIT=VIO,
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//ISPSLIB DD DISP=(,DELETE),SPACE=(TRK,(1,1,1)),UNIT=VIO,
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//ISPPROF DD DISP=(,DELETE),SPACE=(TRK,(1,1,1)),UNIT=VIO,
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//SYSTSPRT DD SYSOUT=*
//*- -----
//** SYSTSIN - THE BATCH TSO COMMANDS TO INVOKE ISPF *
//*- -----
//SYSTSIN DD *
PROFILE PREFIX(userid)
ispf cmd(%zgbatch repo-prefix qualignr repo-directory P
/*
//COMMIT DD *      (Optional)
Commit subject line
Text line 1
Text line 2
/*

```

The zgbatch syntax is:

Parameter	Description
Repo-prefix	This is the ZIGI-defined z/OS prefix for the z/OS data sets within the repository.
Qualignr	The # of qualifiers to ignore when constructing the OMVS filenames.
Repo-directory	The OMVS directory where the repository is located.
C or P	C to commit only or P to commit and push.

NEXT TOPIC: [Timing \(For Those Interested\)](#)

## 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. When ZIGI runs, time stamps for various routines and functions are recorded in a file in your home directory. When ZIGI ends, if the DD was allocated, the file is listed on the **z/OS UNIX Directory List** screen:

```

Menu Utilities View Options Help
z/OS UNIX Directory List List 1 of 1
Command ==> [ ] Scroll ==> PAGE
List . . . : /u/slbd1/zigidebug*
EUID . . . : 38155432
Command Pathname Message Type Permission Audit Ext Fmat
----- /u/slbd1/zigide File rw-rw-rw- fff--- --s- ----
**End**

```

The file may be browsed, viewed, edited, or deleted.

```

Menu Utilities Compilers Help
BROWSE /u/slbd1/zigidebug.d20063.t075013 Line 0000000000 Col 001 080
Command ==> [ ] Scroll ==> CSR
***** Top of Data *****
3 Mar 2020 07:50:13 : Starting collection of env variables
3 Mar 2020 07:50:17 : Finished collection of env variables
3 Mar 2020 07:50:21 : Getting binfiles: ztest
3 Mar 2020 07:50:22 : Got 3 binfiles
3 Mar 2020 07:50:22 : Selecting repo: ztest
3 Mar 2020 07:50:22 : Working with Repo: ztest
3 Mar 2020 07:50:23 : fetching remote
3 Mar 2020 07:50:23 : getting the remotes
3 Mar 2020 07:50:28 : ...done fetching remotes
3 Mar 2020 07:50:28 : getting all files in repo-1
3 Mar 2020 07:50:28 : ...done
3 Mar 2020 07:50:29 : iterating...
3 Mar 2020 07:50:29 : check file or dir
3 Mar 2020 07:50:30 : done
3 Mar 2020 07:50:30 : its a dir...lmmstat all the things SLBD1.ZTEST.PDS
3 Mar 2020 07:50:34 : check file or dir
3 Mar 2020 07:50:35 : done

```

The active debug log may be viewed from several of the panels (Local Repository, Current Repository, and PDS Member List) by using the VIEWD command.

For those interested, here is a short EXEC that allocates and/or frees 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'

```

NEXT TOPIC: [Appendix A: Installing Git](#)

## Appendix A: Installing Git

---

Appendix A lists the actions needed to install Git.

Check	Action
	Create an OMVS user account with SuperUser (su) capabilities (for installing Git systemwide).
	Allocate a good size ZFS (one that we are using is 500-600 cylinders).
	Get 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. Go to <a href="https://www.rocketsoftware.com/product-categories/mainframe/git-for-zos">https://www.rocketsoftware.com/product-categories/mainframe/git-for-zos</a> to 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.
	Go to <a href="https://gist.github.com/wizardofzos/897b243d4cbe9fbc471ec1396fbbe174">https://gist.github.com/wizardofzos/897b243d4cbe9fbc471ec1396fbbe174</a> and download the installer into the same directory as the things you downloaded from Rocket Software.
	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., is installed. This may be your personal home directory if the filesystem is large enough (or can grow enough).
	Mount the 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 the git packages were uploaded to.
	Run the installer script, and when prompted, provide the path to the mounted ZFS that is home to git and tools.
	Review the installation.
	Find the environment.sh file and copy the contents into your /etc/profile file. If you installed into a path that is not the production path, then be VERY CAREFUL as you 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 then enter the command git version – if it works you have Git.
	When happy: <ul style="list-style-type: none"> <li>Remove _dist.sh script to remove the downloaded files</li> </ul>

Check	Action
	<ul style="list-style-type: none"> <li>Then use <code>uninstall.sh</code> to:           <ul style="list-style-type: none"> <li>Remove the <code>environment.sh</code></li> <li>Remove the <code>remove_dist.sh</code></li> <li>Remove the other installation files</li> </ul> </li> </ul>
	Now, unmount 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 is mounted at each IPL: <pre>MOUNT      FILESYSTEM('OMVS.GIT.ZFS') TYPE(ZFS) MODE(READ) MOUNTPOINT('/usr/ported')</pre>

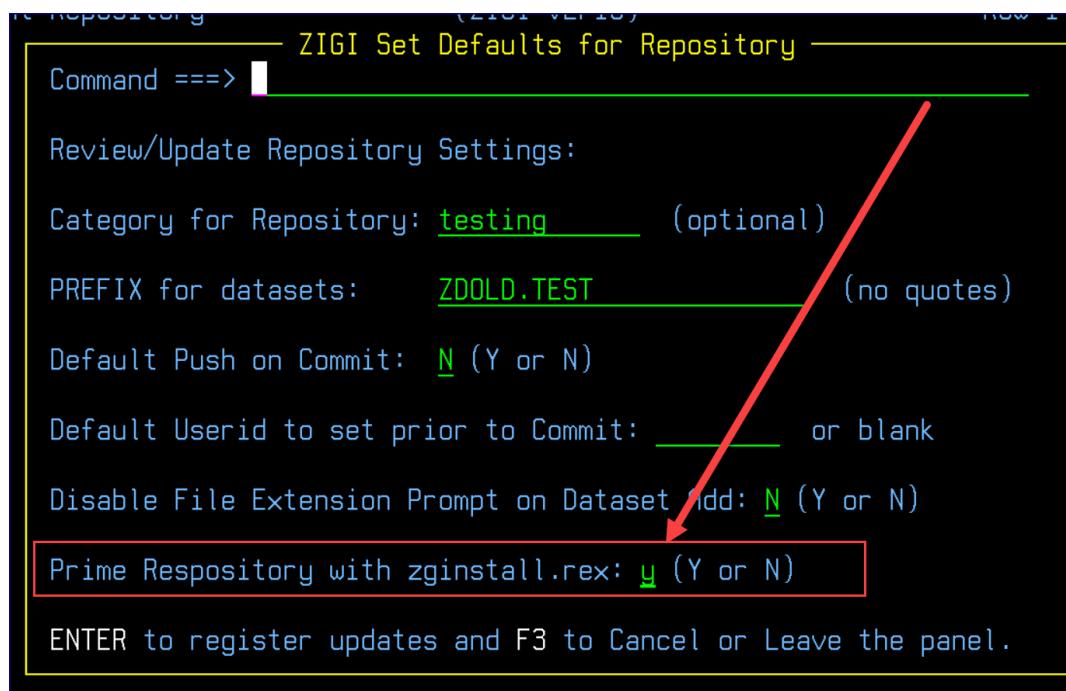
NEXT TOPIC: [Appendix B: Using ZGINSTALL.REX](#)

## Appendix B: Using ZGINSTALL.REX

---

The `zgininstall.rex` is a generic installation tool that works with any ZIGI-managed repository to recreate the z/OS data sets when a repository is cloned outside of ZIGI. It is intended for use when a site does not have ZIGI installed on z/OS to allow ZIGI-managed repositories to be installed easily.

To add `zgininstall.rex` to a repository, use the SET command in the Current Repository and update the settings:



This generates a short and long message on the **ZIGI Git Messages** screen:

```

Browse ZIGI Git Messages Lines 0000000000
Command ==> [ ] Scroll ==> CSR
***** Top of Data *****
warning: zgininstall.readme added file have been automatically tagged IBM-1047 because
warning: zgininstall.rex added file have been automatically tagged IBM-1047 because
***** Bottom of Data *****

The ZIGI Generalized Installation Tool has been copied into the Git
repository. Both zgininstall.rex and zgininstall.readme have been created.
-----
A COMMIT with a PUSH should now be performed.

.
.
```

A git add is performed automatically. Perform a Commit and Push to lock in this update.

*NEXT TOPIC:* [Appendix C: Learning Resources](#)

## Appendix C: Learning Resources

---

The following are some links to webpages to learn more about Git:

- <https://danielmiessler.com/study/git/>
- <https://www.atlassian.com/git>
- <https://git-scm.com/>
- <http://rogerdudler.github.io/git-guide/>
- <https://guides.github.com/introduction/git-handbook/#basic-git>
- <https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>
- Additional links: <https://try.github.io/>

*NEXT TOPIC:* [Appendix D: User Exits](#)

## Appendix D: User Exits

---

ZIGI version 3.10 includes new user exits. All exits are provided as samples that must be updated by each site and then installed using the correct exit name.

Available exits include:

User Exit	Description
ZIGIXITO (SAMPXIT0)	Return the location of the Dovetail getpds and putpds commands if they are not found in the PATH.

The exit driver is ZIGIEXIT, which is called passing the exit number:

Return = zigiexit(#)

OR

Return = zigiexit(# *data*)

The # maps to the exit number (for example: 0 to ZIGIXIT0) and *data* is any information that the exit may require. Each exit returns the requested information.

*NEXT TOPIC:* [Appendix E: Common Problems](#)

## Appendix E: Common Problems

---

This is a partial list of common problems you may encounter:

- Failure in the cp command attempting to copy a z/OS data set, or z/OS PDS member, to USS. This is typically caused by the z/OS data set being opened (for example: ISPF Browse).
- Failure using the Clone or Remote ZIGI commands due to an access failure. This may be caused by your ssh public key on the active LPAR not being registered with the remote server.

