The issue I worked on:

- zopen info -r not showing Build Dependencies

```
09:39:35 RC=(0) [SYSA] bash-5.2$ zopen info git
==> git: STABLE 2.48.1
Description:         git on z/OS
==> Package
Version:             2.48.1
Release:             20250128_192801
Buildline:           STABLE
Categories:          development source_control
GitHub:              https://github.com/zopencommunity/gitport
License:             https://github.com/zopencommunity/gitport/blob/main/patches/LICENSE
zopen license:       https://github.com/zopencommunity/gitport/blob/main/LICENSE
==> Installation Details
Installed:           Yes
Installation Path:   /hewitt/zopentools/zopen_repo/usr/local/zopen/git/git-heads.v2.48.1.20250128_192801.zos
Installation Size:   210.803 MB
==> Test Status
Passed:              89/92 (96%)
==> Package Details
Download Size:       52.67 MB
Commit SHA:          be3a731b0efd1c47adf8259f09c1e584c493c109
Community SHA:       f93ff170b93a1782659637824b25923245ac9dd1
==> Dependencies
Build:               autoconf, automake, bash, check_clang, check_python, coreutils, curl, curl, diffutils, expat, gawk, gettext, git, git, gzip, help2man, jq,
libpcre2, m4, make, ncurses, openssl, perl, sed, sed, tar, tar, texinfo, xz, zlib, zoslib, zoslib
Runtime:             bash, less, ncurses, perl

09:39:40 RC=(0) [SYSA] bash-5.2$ zopen info git -r
==> git (Build 2873) - (STABLE) (Not Installed)
==> Package
Version:             2.48.1.0
Release:             20250128_192801
Buildline:           _gitport_2873
Categories:          development source_control
GitHub:              https://zopencommunity/gitport/releases/download
Installed:           No
==> Test Status
Passed:              89/92 (96%)
==> Package Details
Download Size:       25.80 MB
Install Size:        52.67 MB
Community SHA:       f93ff170b93a1782659637824b25923245ac9dd1
==> Dependencies
Runtime:             bash less ncurses perl
```

  - Link: https://github.com/zopencommunity/meta/issues/954

- Description of issue: When using the zopen info command, the build dependencies will appear. However, when we add -r at the end, it will not show the build dependencies.

The initial solution I implemented:

- Since we know that -r will force a remote lookup, it means that when we run the “zopen info -r” it will access the remote JSON_CACHE to retrieve the build dependencies data.

- When looking into the zopen-info file with the code for that command, I saw that there was no code allowing build dependencies to be displayed in a remote lookup. To fix this, I first parsed the build and runtime dependencies field from the remote_data JSON object using the jq tool.
  - This is seen here:
    - build_dependencies=$(echo "${remote_data}" | jq -r '.assets[0].build_dependencies | join(", ")')

        - runtime_dependencies=$(echo "${remote_data}" | jq -r '.assets[0].runtime_dependencies')

- I then created a Dependencies section that formatted and displayed both the build and runtime dependencies. I also added a condition that checked if either of these fields was missing or empty which would then display “None”.
    - Here is how it looks:
        - printHeader "==> Dependencies"
        - if [-n "${build_dependencies}"]; then
        - printf "%-20s %s\n" "Build:" "${build_dependencies}"
        - else
        - printf "%-20s %s\n" "Build:" "None"
        - fi
        - if [-n "${runtime_dependencies}"]; then
        - printf "%-20s %s\n" "Runtime:" "${runtime_dependencies}"
        - else
        - printf "%-20s %s\n" "Runtime:" "None"
        - fi

- This should hopefully fix the issue.

More info on the code created:

- build_dependencies=$(echo "${remote_data}" | jq -r '.assets[0].build_dependencies | join(", ")')
    - build_dependencies= will assign the output to the shell variable
    - echo "${remote_data}" will take the content of the remote_data variable (JSON string) and pass it as the input for next command in the pipeline
    - jq -r will process the JSON file making sure that the output doesn’t include quotes around strings or escape special characters.
    - '.assets[0].build_dependencies | join(", ")') will access the first object of the assets array which which comes from the build_dependencies.Then it will use the “|” to pass the result from that to the join function so that if there are an array of strings it combines them into a single string.
- printHeader "==> Dependencies"

    - Will use function printHeader to display header labeld "==> Dependencies"
    - Used to keep with what the normal output had.

- if [-n "${build_dependencies}"]; then
    - Checked if the build_dependencies variable contains a non-empty value
    - -n
        - Test operator that evaluates to true if string length is greater than 0

- printf "%-20s %s\n" "Build:" "${build_dependencies}"
    - %-20s will allocate 20 characters for the first string with “-” being used for it to be left aligned. This is for the “Build: ” label
    - %s will display the second string immediately after the label
    - /n will add a newline at the end of the printed line

- else
- printf "%-20s %s\n" "Build:" "None"
    - If build_dependencies is empty or undefined then it will print “Build: None” showing no build dependencies exist

- fi
    - Marks the end of the if statement block (Shell doesn’t know conditional block finished without this)