



Customize Installation using Golden ISO

Table 1: Feature History Table

Feature Name	Release Information	Description
Automatic Install of Bridging Bug Fix RPMs	Release 7.5.2	This feature enables an easy one-step, no prompt upgrade, or downgrade, based on GISO. This removes the dependency on having to manually install the bridging bug fix RPMs before performing an upgrade or a downgrade.

Golden ISO (GISO) is a customized ISO that a user can build to suit the installation requirement. The user can customize the installable image to include the standard base image with the basic functional components, and add additional RPMs, SMUs and configuration files based on requirement.

The ease of installation and the time taken to seamlessly install or upgrade a system plays a vital role in a cloud-scale network. An installation process that is time-consuming and complex affects the resiliency and scale of the network. The GISO simplifies the installation process, automates the installation workflow, and manages the dependencies in RPMs and SMUs automatically.

GISO is built using a build script `gisobuild.py` available on the github location [Github](#) location.

From Cisco IOS XR Release 7.5.2, you can use the Automatic Install of Bridging Bug Fix RPMs feature to install the bridging bug fix RPMs that are prerequisite for a system upgrade or a downgrade. You need to add the required Bridging Bug Fix RPMs into the customized ISO built using Cisco Golden ISO (GISO) build script **gisobuild.py**. The GISO can include bridging Bug Fix RPMs for multiple releases, and installs only the specific bridging Bug Fix RPMs required for the target release. The bridging bug fix RPMs can be used in the following scenarios:

- To resolve a bug that might stop upgrade.
- The latest version has new prerequisite requirements that are not met by the earlier version.

When a system boots with GISO, additional SMUs and RPMs in GISO are installed automatically, and the router is pre-configured with the XR configuration in GISO. For more information about downloading and installing GISO, see [Install Golden ISO, on page 12](#).

The capabilities of GISO can be used in the following scenarios:

- Initial deployment of the router

- Software disaster recovery
- System upgrade from one base version to another
- System upgrade from same base version but with additional SMUs
- Install update to identify and update dependant packages
- [Limitations, on page 3](#)
- [Customize Installation using Golden ISO, on page 2](#)
- [Golden ISO Workflow, on page 3](#)
- [Build Golden ISO, on page 4](#)
- [Install Golden ISO, on page 12](#)
- [Install Replace with Golden ISO, on page 15](#)

Limitations

The following are the known problems and limitations with the customized ISO:

- Building and booting GISO for asynchronous package (a package of different release than the ISO) is not supported.
- Verifying the XR configuration is not supported in the GISO build script `gisobuild.py`.
- Renaming a GISO build and then installing from the renamed GISO build is not supported.
- Install operation over IPv6 is not supported.

Customize Installation using Golden ISO

Table 2: Feature History Table

Feature Name	Release Information	Description
Automatic Install of Bridging Bug Fix RPMs	Release 7.5.2	This feature enables an easy one-step, no prompt upgrade, or downgrade, based on GISO. This removes the dependency on having to manually install the bridging bug fix RPMs before performing an upgrade or a downgrade.

Golden ISO (GISO) is a customized ISO that a user can build to suit the installation requirement. The user can customize the installable image to include the standard base image with the basic functional components, and add additional RPMs, SMUs and configuration files based on requirement.

The ease of installation and the time taken to seamlessly install or upgrade a system plays a vital role in a cloud-scale network. An installation process that is time-consuming and complex affects the resiliency and scale of the network. The GISO simplifies the installation process, automates the installation workflow, and manages the dependencies in RPMs and SMUs automatically.

GISO is built using a build script `gisobuild.py` available on the github location [Github](#) location.

From Cisco IOS XR Release 7.5.2, you can use the Automatic Install of Bridging Bug Fix RPMs feature to install the bridging bug fix RPMs that are prerequisite for a system upgrade or a downgrade. You need to add the required Bridging Bug Fix RPMs into the customized ISO built using Cisco Golden ISO (GISO) build script **gisobuild.py**. The GISO can include bridging Bug Fix RPMs for multiple releases, and installs only the specific bridging Bug Fix RPMs required for the target release. The bridging bug fix RPMs can be used in the following scenarios:

- To resolve a bug that might stop upgrade.
- The latest version has new prerequisite requirements that are not met by the earlier version.

When a system boots with GISO, additional SMUs and RPMs in GISO are installed automatically, and the router is pre-configured with the XR configuration in GISO. For more information about downloading and installing GISO, see [Install Golden ISO, on page 12](#).

The capabilities of GISO can be used in the following scenarios:

- Initial deployment of the router
- Software disaster recovery
- System upgrade from one base version to another
- System upgrade from same base version but with additional SMUs
- Install update to identify and update dependant packages

Limitations

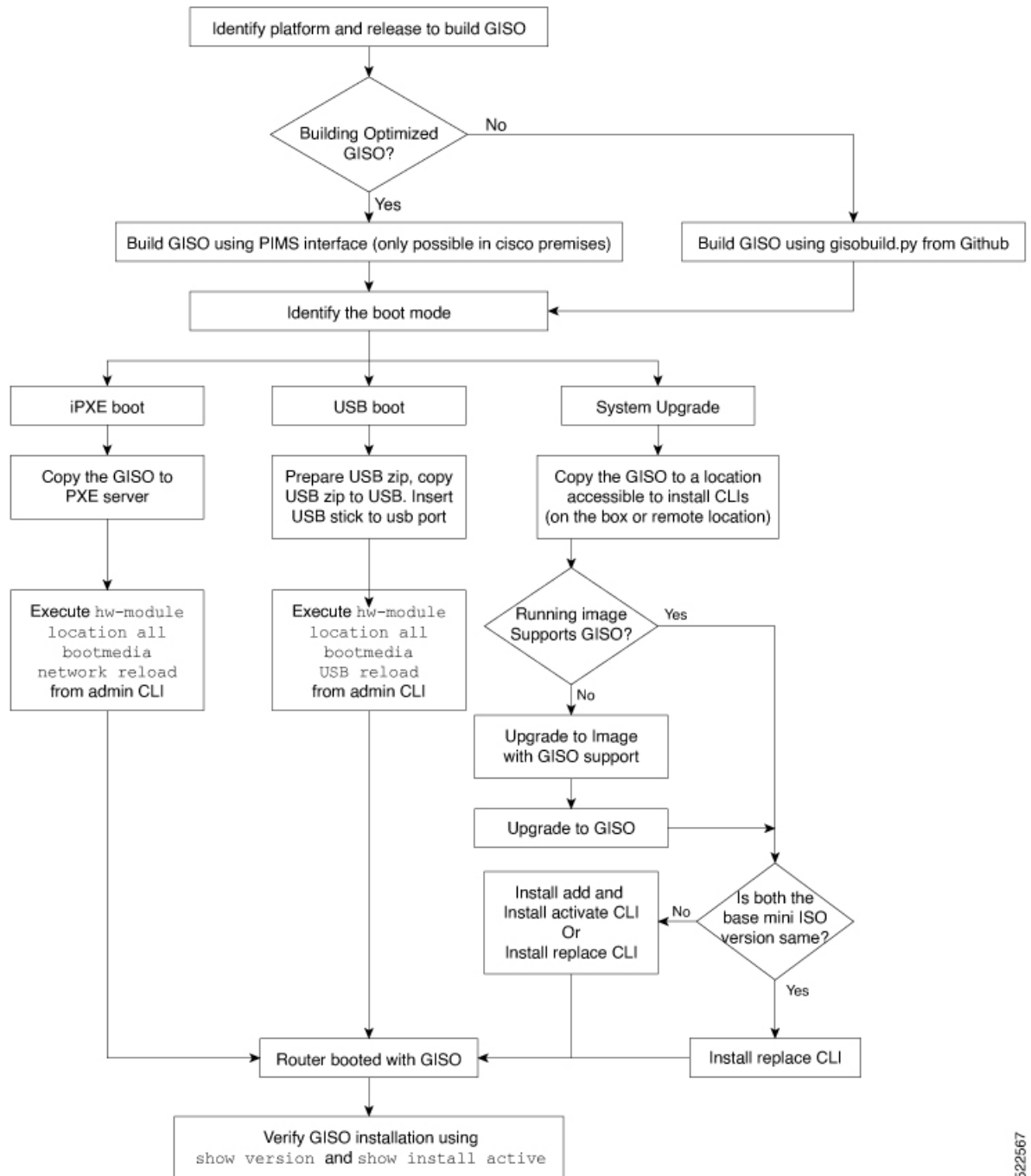
The following are the known problems and limitations with the customized ISO:

- Building and booting GISO for asynchronous package (a package of different release than the ISO) is not supported.
- Verifying the XR configuration is not supported in the GISO build script `gisobuild.py`.
- Renaming a GISO build and then installing from the renamed GISO build is not supported.
- Install operation over IPv6 is not supported.

Golden ISO Workflow

The following image shows the workflow for building and installing golden ISO.

Figure 1: Golden ISO Workflow



522567

Build Golden ISO

The customized ISO is built using Cisco Golden ISO (GISO) build script `gisobuild.py` available on the [Github](#) location.

The GISO build script supports automatic dependency management, and provides these functionalities:

- Builds RPM database of all the packages present in package repository.
- Scans the repositories and selects the relevant Cisco RPMs that matches the input iso.
- Skips and removes third-party RPMs that are not SMUs of already existing third-party base package in mini-x.iso.
- Displays an error and exits build process if there are multiple base RPMs of same release but different versions.
- Performs compatibility check and dependency check for all the RPMs. For example, the child RPM ncs5500-mpls-te-rsvp is dependent on the parent RPM ncs5500-mpls . If only the child RPM is included, the Golden ISO build fails.

Build Golden ISO Using Script

Table 3: Feature History Table

Feature Name	Release Information	Description
Enhanced Golden ISO Build Tool	Release 7.5.1	This enhancement provides you with the flexibility to use the <code>gisobuild.py</code> tool to build GISO images using Cisco IOS XR software commands, YAML-based template file, or docker capability to suit your customized install requirements. When you build a GISO, you can also specify Zero Touch Provisioning (ZTP) initialization file, script initialization file, Cisco IOS XR configuration file, and SMUs in addition to using the base image and optional RPMs to automatically provision the router.

To build GISO, provide the following input parameters to the script:

- Base mini-x.iso (mandatory)
- XR configuration file (optional)
- one or more Cisco-specific SMUs for host, XR and System admin (optional)
- one or more third-party SMUs for host, XR and System admin (optional)
- Label for golden ISO (optional)
- Optional RPMs
- ZTP initialization `ztp.ini` file (optional)

- Script initialization `script.ini` file (optional)

The GISO script does not support verification of XR configuration.



Note To successfully add k9sec RPM to GISO, change the permission of the file to 644 using the **chmod** command.

```
chmod 644 [k9 sec rpm]
```

Cisco IOS XR, Release 7.5.1 introduces enhancements to the `gisobuild.py` GISO build tool. You can also add a `ztp.ini` ZTP initialization and `script.ini` Script initialization file. The ZTP configuration is applied on the router when the current software version is replaced or rolled back to a version with GISO image, and is used whenever ZTP is run to automatically provision the router. The tool supports more than one repository. You can use CLI command, docker, or a YAML file to build GISO.



- Note**
- For Cisco NCS 5500 and Cisco NCS 5000 series routers, set the `migration` value to `false`.
 - Set the `clean` option to `true` if you use the same build directory after the first GISO is created. Ensure that you set the option to `true` for every successive GISO build.
 - Set the `docker` option to `true` if you are building GISO using docker.
 - Ensure that the format and syntax of the YAML file is intact to avoid errors when building a GISO. For example, if the `:` symbol is missing, or if an unsupported symbol is used in the template, the GISO build displays errors.

The `gisobuild.py` tool can be run either natively or on systems where docker service is enabled and has the ability to pull published docker images. Prefer building the image using the docker as it does not require additional privileges.



Note The `full-iso` option is used to build a full ISO image `xrv9k-full-x-7.5.1.iso` specific to Cisco IOS XRv 9000 routers. Starting Cisco IOS XR, Release 7.8.1, the full ISO image must not be used to build GISO.

To build GISO, perform the following steps:

Before you begin

- The system where GISO is built must meet the following requirements:
 - System must have Python version 3.6 and later.
 - System must have free disk space of minimum 12 GB.
 - Verify that the Linux utilities `mount`, `rm`, `cp`, `umount`, `zcat`, `chroot`, `mkisofs` are present in the system. These utilities will be used by the script. Ensure privileges are available to execute all of these Linux commands. However, if you are using docker, these utilities are not required.
 - Kernel version of the system must be later than 3.16 or later than the version of kernel of Cisco ISO.

- Verify that a `libyaml` rpm supported by the Linux kernel is available to successfully import `yaml` in the tool.
- User should have proper permission for security rpm(k9sec-rpm) in rpm repository, else security rpm would be ignored for Golden ISO creation.
- The system from where the `gisobuild.py` script is executed must have root credentials. This is not mandatory if you are building the image within a docker container.
- We recommend that you perform a `git pull` operation before you use the `gisobuild.py` script to ensure you obtain the latest version of the script for the Python version.

Procedure

- Step 1** Copy the script `gisobuild.py` from the [Github](#) repository to an offline system or external server where the GISO will be built. Ensure that this system meets the pre-requisites described above in the *Before You Begin* section.
- Step 2** Run the script `gisobuild.py` and provide parameters to build the golden ISO off the router. Ensure that all RPMs and SMUs are present in the same directory or on a repository. The number of RPMs and SMUs that can be used to build the Golden ISO is 64.

```
usage: gisobuild.py [-h] [--iso ISO] [--repo REPO [REPO ...]]
                  [--bridging-fixes BRIDGE_FIXES [BRIDGE_FIXES ...]]
                  [--xrconfig XRCONFIG] [--ztp-ini ZTP_INI] [--label LABEL]
                  [--out-directory OUT_DIRECTORY] [--yamlfile CLI_YAML] [--clean]
                  [--pkglist PKGLIST [PKGLIST ...]] [--script SCRIPT] [--docker]
                  [--x86-only] [--migration]
                  [--remove-packages REMOVE_PACKAGES [REMOVE_PACKAGES ...]]
                  [--skip-usb-image] [--copy-dir COPY_DIRECTORY]
                  [--clear-bridging-fixes] [--verbose-dep-check] [--debug]
                  [--version]
```

Utility to build Golden ISO for IOS-XR.

optional arguments:

```
-h, --help            show this help message and exit
--iso ISO             Path to Mini.iso/Full.iso file
--repo REPO [REPO ...]
                    Path to RPM repository. For LNT, user can specify .rpm, .tgz,
                    .tar filenames, or directories. RPMs are only used if already
                    included in the ISO, or specified by the user via the
                    --pkglist option.
--bridging-fixes BRIDGE_FIXES [BRIDGE_FIXES ...]
                    Bridging rpms to package. For EXR, takes from-release or rpm
                    names; for LNT, the user can specify the same file types as for
                    the --repo option.
--xrconfig XRCONFIG  Path to XR config file
--ztp-ini ZTP_INI    Path to user ztp ini file
--label LABEL, -l LABEL
                    Golden ISO Label
--out-directory OUT_DIRECTORY
                    Output Directory
--yamlfile CLI_YAML  Cli arguments via yaml
--clean              Delete output dir before proceeding
--pkglist PKGLIST [PKGLIST ...]
                    Packages to be added to the output GISO. For eXR: optional rpm
                    or smu to package. For LNT: either full package filenames or
                    package names for user installable packages can be specified.
```

```

Full package filenames can be specified to choose a particular
version of a package, the rest of the block that the package is
in will be included as well. Package names can be specified to
include optional packages in the output GISO.

--docker, --use-container
    Build GISO in container environment. Pulls and run pre-built
    container image to build GISO.

--version
    Print version of this script and exit

EXR only build options:
--script SCRIPT      Path to user executable script executed as part of bootup post
                    activate.
--x86-only           Use only x86_64 rpms even if other architectures are
                    applicable.
--migration          To build Migration tar only for ASR9k

LNT only build options:
--remove-packages REMOVE_PACKAGES [REMOVE_PACKAGES ...]
                    Remove RPMs, specified in a comma separated list. These are are
                    matched against user installable package names, and must be the
                    whole package name, e.g: xr-bgp
--skip-usb-image      Do not build the USB image
--copy-dir COPY_DIRECTORY
                    Copy build artefacts to specified directory if provided. The
                    specified directory must already exist, be writable by the
                    builder and must not contain a previously built artefact with
                    the same name.
--clear-bridging-fixes
                    Remove all bridging bugfixes from the input ISO
--verbose-dep-check   Verbose output for the dependency check.
--debug              Output debug logs to console

```

Example

Example: Build Docker-Based GISO Image

In this example, a GISO image is built using docker.

```

[root@xr src]# ./gisobuild.py --docker --iso /auto/ncs5500giso/ncs5500-mini-x-7.5.1.iso
--repo /auto/ncs5500giso --pkglst ncs5500-bgp-2.0.0.0-r751.x86_64.rpm
ncs5500-eigrp-1.0.0.0-r751.x86_64.rpm ncs5500-isis-2.1.0.0-r751.x86_64.rpm
ncs5500-k9sec-3.1.0.0-r751.x86_64.rpm
ncs5500-li-1.0.0.0-r751.x86_64.rpm ncs5500-mcast-3.0.0.0-r751.x86_64.rpm
ncs5500-mgb1-3.0.0.0-r751.x86_64.rpm
ncs5500-mpls-2.1.0.0-r751.x86_64.rpm ncs5500-mpls-te-rsvp-3.1.0.0-r751.x86_64.rpm
ncs5500-ospf-2.0.0.0-r751.x86_64.rpm
ncs5500-parser-2.0.0.0-r751.x86_64.rpm --label dockerbasedgiso

```

```

Local System requirements check [PASS]
Pulling gisobuild image from hub. Please wait...
\
Done...
System requirements check [PASS]

```

```
Platform: ncs5500 Version: 7.5.1
```

```
Scanning repository [/auto/ncs5500giso]...
```

```
Building RPM Database...
```

```
Total 11 RPM(s) present in the repository path provided in CLI
```

```
[ 1] ncs5500-mpis-2.1.0.0-r751.x86_64.rpm
[ 2] ncs5500-mgbl-3.0.0.0-r751.x86_64.rpm
[ 3] ncs5500-bgp-2.0.0.0-r751.x86_64.rpm
[ 4] ncs5500-parser-2.0.0.0-r751.x86_64.rpm
[ 5] ncs5500-isis-2.1.0.0-r751.x86_64.rpm
[ 6] ncs5500-mcast-3.0.0.0-r751.x86_64.rpm
[ 7] ncs5500-mpis-te-rsvp-3.1.0.0-r751.x86_64.rpm
[ 8] ncs5500-ospf-2.0.0.0-r751.x86_64.rpm
[ 9] ncs5500-li-1.0.0.0-r751.x86_64.rpm
[10] ncs5500-eigrp-1.0.0.0-r751.x86_64.rpm
[11] ncs5500-k9sec-3.1.0.0-r751.x86_64.rpm
```

Following XR x86_64 rpm(s) will be used for building Golden ISO:

```
(+) ncs5500-ospf-2.0.0.0-r751.x86_64.rpm
(+) ncs5500-bgp-2.0.0.0-r751.x86_64.rpm
(+) ncs5500-parser-2.0.0.0-r751.x86_64.rpm
(+) ncs5500-mcast-3.0.0.0-r751.x86_64.rpm
(+) ncs5500-li-1.0.0.0-r751.x86_64.rpm
(+) ncs5500-eigrp-1.0.0.0-r751.x86_64.rpm
(+) ncs5500-mgbl-3.0.0.0-r751.x86_64.rpm
(+) ncs5500-mpis-2.1.0.0-r751.x86_64.rpm
(+) ncs5500-mpis-te-rsvp-3.1.0.0-r751.x86_64.rpm
(+) ncs5500-isis-2.1.0.0-r751.x86_64.rpm
(+) ncs5500-k9sec-3.1.0.0-r751.x86_64.rpm
```

...RPM signature check [PASS]

Skipping following rpms from repository since they are already present in base ISO:

```
(-) ncs5500-parser-2.0.0.0-r751.x86_64.rpm
(-) ncs5500-bgp-2.0.0.0-r751.x86_64.rpm
```

...RPM compatibility check [PASS]

Building Golden ISO...

Summary

XR rpms:

```
ncs5500-mcast-3.0.0.0-r751.x86_64.rpm
ncs5500-mgbl-3.0.0.0-r751.x86_64.rpm
ncs5500-isis-2.1.0.0-r751.x86_64.rpm
ncs5500-mpis-te-rsvp-3.1.0.0-r751.x86_64.rpm
ncs5500-eigrp-1.0.0.0-r751.x86_64.rpm
ncs5500-mpis-2.1.0.0-r751.x86_64.rpm
ncs5500-ospf-2.0.0.0-r751.x86_64.rpm
ncs5500-li-1.0.0.0-r751.x86_64.rpm
ncs5500-k9sec-3.1.0.0-r751.x86_64.rpm
```

...Golden ISO creation SUCCESS.

Golden ISO Image Location: /var/tmp/giso/gisobuild-toolkit-master/src/output_gisobuild/
ncs5500-golden-x-7.5.1-dockerbasedgiso.iso

View that the GISO file is created successfully.

```
[root@xr src]# ls
exrmod gisobuild.py lntmod output_gisobuild utils

[root@xr src]# cd output_gisobuild/
[root@xr output_gisobuild]# ls
img_built_name.txt logs ncs5500-golden-x-7.5.1-dockerbasedgiso.iso
rpms_packaged_in_giso.txt
```

Example: Build YAML-Based GISO Image

YAML is a markup file that serves as a template to provide the package list and manage the build options.

The following example shows a sample YAML template:

```
# Options below correspond to the tool input options.
# --iso ISO          Path to Mini.iso/golden.iso file
# --repo REPO [REPO ...]
#                   Path to list of RPM repositories. RPMs are only used if already
#                   included in the ISO, or specified by the user via the --pkglist
#                   option.
# --pkglist PKGLIST [PKGLIST ...]
#                   Optional list of rpm or smu to add to the ISO.
# --remove-packages REMOVE_PACKAGES [REMOVE_PACKAGES ...]
#                   Remove named RPMs, specified in a space separated list. Valid build
#                   option for LNT only. eXR builds simply ignores this option.
# --bridging-fixes BRIDGE_FIXES [BRIDGE_FIXES ...]
#                   Bridging rpms to package. Takes from-release (supported for eXR)
#                   or rpm names.
# --xrconfig XRCONFIG Path to XR config file
# --ztp-ini ZTP_INI    Path to user ztp ini file
# --script SCRIPT      Path to user executable script executed as part of
#                   bootup post activate. Valid build option for eXR only.
#                   LNT builds simply ignores.
# --label LABEL        Golden ISO Label
# --out-directory OUT_DIRECTORY
#                   Output Directory. Built GISO and logs will be available post
#                   gisobuild.
# --copy-directory COPY_DIRECTORY
#                   Copy built artefacts to specified directory if provided. Valid build
#                   option for LNT only. eXR build ignores this option.
# --yamlfile CLI_YAML  Cli arguments via yaml.
# --clean              Delete output dir before proceeding.
# --migration          To build Migration tar only for ASR9k. Valid build option for eXR
#                   only.
#                   LNT builds simply ignore this option.
# --docker             Load and run pre-built docker image. Valid build option for eXR
#                   only.
#                   LNT builds simply ignore this option.
# --x86-only           Use only x86_64 rpms even if other architectures are applicable.
#                   Valid build
#                   option for eXR only. LNT builds simply ignore this option.
# --version            Print version of this script and exit

packages:
  iso: <path-to-iso>
  repo:
    - <path-to-repo1>
    - <path-to-repo2>
  pkglist:
    - <pkg1>
    - <pkg2>
  bridge-fixes:
    upgrade-from-release:
      - <dotted-release-1>
      - <dotted-release-2>
    rpms:
      - <pkg1>
      - <pkg2>
  remove_packages:
    - <pkg1>
    - <pkg2>

user-content:
```

```

script: <path-to-script-sh>
xrconfig: <path-to-router.cfg>
ztp-ini: <path-to-ztp.ini>

output:
  label: <giso-label>
  out-directory: <path-to-output-directory>
  clean: <true/false>

options:
  docker: <true/false>
  migration: <true/false>
  x86-only: <true/false>

```

In this example, you configure a YAML file with the required files:

```

packages:
  iso: /auto/751_repo/ncs5500-mini-x-7.5.1.iso
  repo:
    - /auto/751_repo/
  pkglist:
    - ncs5500-bgp-2.0.0.0-r751.x86_64.rpm
    - ncs5500-eigrp-1.0.0.0-r751.x86_64.rpm
    - ncs5500-isis-2.1.0.0-r751.x86_64.rpm
    - ncs5500-li-1.0.0.0-r751.x86_64.rpm
    - ncs5500-mcast-3.0.0.0-r751.x86_64.rpm
    - ncs5500-mgbl-3.0.0.0-r751.x86_64.rpm
    - ncs5500-mppls-2.1.0.0-r751.x86_64.rpm
    - ncs5500-mppls-te-rsvp-3.1.0.0-r751.x86_64.rpm
    - ncs5500-ospf-2.0.0.0-r751.x86_64.rpm
    - ncs5500-parser-2.0.0.0-r751.x86_64.rpm
    - ncs5500-k9sec-3.1.0.0-r751.x86_64.rpm
    - ncs5500-mcast-3.0.0.1-r751.CSCxr33333.x86_64.rpm
    - ncs5500-os-5.0.0.1-r751.CSCxr11111.x86_64.rpm
    - ncs5500-sysadmin-hostos-7.5.1-r751.CSCho99999.admin.x86_64.rpm
    - ncs5500-sysadmin-hostos-7.5.1-r751.CSCho99999.host.x86_64.rpm
    - ncs5500-sysadmin-topo-7.5.1-r751.CSCcv55555.x86_64.rpm
    - ncs5500-sysadmin-system-7.5.1-r751.CSCcv44444.x86_64.rpm
    - openssh-scp-6.6p1.p1-r0.5.0-r751.CSCtp11111.xr.x86_64.rpm
    - cisco-klm-zermatt-0.1.p1-r0.0-r751.CSCtp11111.xr.x86_64.rpm

  remove_rpms: []

user-content:
  script: script.sh
  xrconfig: /auto/751_repo/gisoxrconfig.cfg
  ztp-ini: /auto/751_repo/ztp.ini

output:
  label: 751_yaml_install
  out-directory: /auto/751_repo/
  clean: true

options:
  docker: false
  full-iso: false
  migration: false
  x86-only: false

```

If you do not want to specify the list of packages and parameters via CLI, you can use the YAML file template.

```
[directory-path]$ ./src/gisobuild.py --yamlfile <input-yaml-cfg>
```

To override any input in the YAML configuration file, use the corresponding CLI options.

```
[directory-path]$ ./src/gisobuild.py --yamlfile <input-yaml-cfg> --label <new-label>
```

This new label overrides the label specified in the YAML file.

When the host machine does not have its package dependencies met, but allows pulling and running docker images, enable the docker option in YAML file to `true` and run the command:

```
[directory-path]$ ./src/gisobuild.py --yamlfile <input-yaml-cfg>
```

where, the `input-yaml-cfg` has the docker option set to `true`.

What to do next

Install the GISO image on the router.

Install Golden ISO

Golden ISO (GISO) automatically performs the following actions:

- Installs host and system admin RPMs.
- Partitions repository and TFTP boot on RP.
- Creates software profile in system admin and XR modes.
- Installs XR RPMs. Use **show install active** command to see the list of RPMs.
- Applies XR configuration. Use **show running-config** command in XR mode to verify.

Procedure

Step 1 Download GISO image to the router using one of the following options:

- **PXE boot:** when the router is booted, the boot mode is identified. After detecting PXE as boot mode, all available ethernet interfaces are brought up, and DHCPClient is run on each interface. DHCPClient script parses HTTP or TFTP protocol, and GISO is downloaded to the box.

When you bring up a router using the PXE boot mode, existing configurations are removed. To recover smart licensing configurations like Permanent License Reservation (PLR), enable these configurations after the router comes up.

```
Router#configure
Router(config)#license smart reservation
Router(config)#commit
```

- **USB boot or Disk Boot:** when the USB mode is detected during boot, and GISO is identified, the additional RPMs and XR configuration files are extracted and installed.
- **System Upgrade:** when the system is upgraded, GISO can be installed using **install add**, **install activate**, or using **install replace** commands.

Important

To replace the current version and packages on the router with the version from GISO, note the change in command and format.

- In versions prior to Cisco IOS XR Release 6.3.3, 6.4.x and 6.5.1, use the **install update** command:

```
install update source <source path> <Golden-ISO-name> replace
```

- In Cisco IOS XR Release 6.5.2 and later, use the **install replace** command.

```
install replace <absolute-path-of-Golden-ISO>
```

Note

To create a Bootable External USB Disk, do the following:

- Ensure that the USB Boot Disk has a minimum storage of 8GB, and that you have root/admin or appropriate permission to create bootable disk on linux machine.

- a. Copy and execute usb-install script on the Linux machine to create a bootable external USB.

```
Router#admin
```

```
Router#./linux/misc/scripts/create_usb_zip ncs5500 ncs5500-goldenk9-x.iso-7.5.01.v1
adding: EFI/ (stored 0%)
adding: EFI/boot/ (stored 0%)
adding: EFI/boot/grub.cfg (deflated 66%)
adding: EFI/boot/bootx64.efi (deflated 67%)
adding: boot/ (stored 0%)
adding: boot/install-image.iso (deflated 1%)
Zip file created - usb_boot.zip
Router# ls -ltr usb_boot.zip
-rw-r--r-- 1 user eng 1448680576 Sep 14 04:13 usb_boot.zip
Router#
```

- b. Reset the RSP/RP and plug in bootable USB to RSP/RP's front panel. The USB will get detected in ROMMON. Note that when the system is in ROMMON, and if you add a front panel external USB, the USB will not be detected until the RSP/RP is reset.

The options to upgrade the system are as follows:

- **system upgrade from a non-GISO (image that does not support GISO) to GISO image:** If a system is running a version1 with an image that does not support GISO, the system cannot be upgraded directly to version2 of an image that supports GISO. Instead, the version1 must be upgraded to version2 mini ISO, and then to version2 GISO.
- **system upgrade in a release from version1 GISO to version2 GISO:** If both the GISO images have the same base version but different labels, **install add** and **install activate** commands does not support same version of two images. Instead, using **install source** command installs only the delta RPMs. System reload is based on restart type of the delta RPMs.

Using **install replace** command performs a system reload, irrespective of the difference between ISO and the existing version.

- **system upgrade across releases from version1 GISO to version2 GISO:** Both the GISO images have different base versions. Use **install add** and **install activate** commands, or **install replace** command to perform the system upgrade. The router reloads after the upgrade with the version2 GISO image.

Step 2

Run the **show install repository all** command in System Admin mode to view the RPMs and base ISO for host, system admin and XR.

```
sysadmin-vm:0_RP0#show install repository all
Admin repository
-----
```

```

ncs5500-sysadmin-6.2.2
ncs5500-sysadmin-hostos-6.2.2-r622.CSCcv10001.admin.x86_64
ncs5500-sysadmin-hostos-6.2.2-r622.CSCcv10001.admin.arm
ncs5500-sysadmin-system-6.2.2-r622.CSCcv10005.x86_64
ncs5500-sysadmin-system-6.2.2-r622.CSCcv10005.arm
....
XR repository
-----
ncs5500-iosxr-mgbl-3.0.0.0-r622.x86_64
ncs5500-xr-6.2.2
....
Host repository
-----
host-6.2.2

```

Step 3 Run the **show install package <golden-iso>** command to display the list of RPMs, and packages built in GISO.

Note

To list RPMs in the GISO, the GISO must be present in the install repository.

```
Router#show install package ncs5500-goldenk9-x64-6.2.2
```

This may take a while ...

```

ISO Name: ncs5500-goldenk9-x64-6.2.2
ISO Type: bundle
ISO Bundled: ncs5500-mini-x64-6.2.2
Golden ISO Label: temp
ISO Contents:
  ISO Name: ncs5500-xr-6.2.2
  ISO Type: xr
  rpms in xr ISO:
    iosxr-os-ncs5500-64-5.0.0.0-r622
    iosxr-ce-ncs5500-64-3.0.0.0-r622
    iosxr-infra-ncs5500-64-4.0.0.0-r622
    iosxr-fwding-ncs5500-64-4.0.0.0-r622
    iosxr-routing-ncs5500-64-3.1.0.0-r6122

```

```

ISO Name: ncs5500-sysadmin-6.2.2
ISO Type: sysadmin
rpms in sysadmin ISO:
  ncs5500-sysadmin-topo-6.2.2-r622
  ncs5500-sysadmin-shared-6.2.2-r622
  ncs5500-sysadmin-system-6.2.2-r622
  ncs5500-sysadmin-hostos-6.2.2-r622.admin
...

```

```

ISO Name: host-6.2.2
ISO Type: host
rpms in host ISO:
  ncs5500-sysadmin-hostos-6.2.2-r622.host

```

```

Golden ISO Rpm:
  xr rpms in golden ISO:
    ncs5500-k9sec-x64-2.2.0.1-r622.CSCxr33333.x86_64.rpm
    openssh-scp-6.6p1.p1-r0.0.CSCtp12345.xr.x86_64.rpm
    openssh-scp-6.6p1-r0.0.xr.x86_64.rpm
    ncs5500-mpis-x64-2.1.0.0-r622.x86_64.rpm
    ncs5500-k9sec-x64-2.2.0.0-r622.x86_64.rpm

  sysadmin rpms in golden ISO:
    ncs5500-sysadmin-system-6.2.2-r622.CSCcv11111.x86_64.rpm
    ncs5500-sysadmin-system-6.2.2-r622.CSCcv11111.arm.rpm
    openssh-scp-6.6p1-r0.0.admin.x86_64.rpm
    openssh-scp-6.6p1-r0.0.admin.arm.rpm

```

```

openssh-scp-6.6p1.p1-r0.0.CSCtp12345.admin.x86_64.rpm
openssh-scp-6.6p1.p1-r0.0.CSCtp12345.admin.arm.rpm
ncs5500-sysadmin-hostos-6.2.2-r622.CSCcv10001.admin.x86_64.rpm
ncs5500-sysadmin-hostos-6.2.2-r622.CSCcv10001.admin.arm.rpm

host rpms in golden ISO:
openssh-scp-6.6p1-r0.0.host.x86_64.rpm
openssh-scp-6.6p1-r0.0.host.arm.rpm
openssh-scp-6.6p1.p1-r0.0.CSCtp12345.host.x86_64.rpm
openssh-scp-6.6p1.p1-r0.0.CSCtp12345.host.arm.rpm

```

The ISO, SMUs and packages in GISO are installed on the router.

Install Replace with Golden ISO

Table 4: Feature History Table

Feature Name	Release Information	Description
Check Integrity of Golden ISO (GISO) Files	Release 7.5.1	This feature enables an automated check during install [package] replace operations to ensure that the files in GISO has not been corrupted. It does so by calculating the md5sum of the files and comparing it against md5sum value that is contained within the GISO that was calculated when the image was built.
Automatic Bridging of Bug Fix RPMs	Release 7.5.2	This release enables an easy, one-step, no-prompt upgrade or downgrade based on GISO. This removes the dependency on manually installing RPMs that bridge bug fixes across release versions.

Golden ISO (GISO) upgrades the router to a version that has a predefined list of software maintenance update (SMUs) with a single operation. However, to update to the same version with a different set of SMUs requires a two-step process.

To avoid this two-step process, use the **install replace** command to replace the currently active version with the full package including the image and SMUs in the newly added GISO.

The process involves upgrading the GISO to add the delta SMUs, and manually deactivating the SMUs that are not in use. In addition, this is the only method to upgrade to GISO containing different optional RPMs, which is a subset of the running set of optional RPMs. For example, consider V1 of GISO is the running version with V1 mini and optional RPMs V1 mpls, V1 mpls-te, V1 mgbl, and V1 k9sec. If V2 of GISO does not contain V2 k9sec, then use **install replace** to upgrade to the optional RPMs in V2.

From Cisco IOS XR Release 7.5.x, you can now add bridging bug fixes through the GISO, and this helps you to automatically install the bridging bug fix RPMs that are prerequisite for a system upgrade or a downgrade. The GISO can include bridging SMUs for multiple source releases, and installs only the specific bridging SMUs required for the target release.

The bridging bug fix RPMs can be used in the following scenarios:

- To resolve a bug that might stop upgrade.
- The latest version has new prerequisite requirements that are not met by the earlier version.

The **install replace** command adds and applies the highest version of any bridging bugfix RPMs for the currently active XR version.



Important

To replace the current version and packages on the router with the version from GISO, note the change in command and format.

- In versions prior to Cisco IOS XR Release 6.3.3, 6.4.x and 6.5.1, use the **install update** command:

```
install update source <source path> <Golden-ISO-name> replace
```

- In Cisco IOS XR Release 6.5.2 and later, use the **install replace** command.

```
install replace <absoulte-path-of-Golden-ISO>
```



Note

The **replace** keyword in **install update** command is supported only with GISO, but not with .mini and .rpm packages directly.



Note

A service pack (SP) is a tar file consisting of bugfix SMUs used for only SMU upgrades. Whereas, a GISO image is used for upgrading the iso images, feature packages, and SMUs on the router. It is recommended to install GISO image because GISO provides customization options during the build process and an enhanced install upgrade experience. GISO provides benefits such as label-based install, automatic management of dependencies, filtering out fully superseded SMUs, handling optional RPMs during GISO build and decreased install upgrade schedule.

A standard image consisting of the latest SMU set, optional RPMs, and base iso packages are combined to build a GISO with a frequency matching the regular service pack release schedule for 32-bit platforms. For 64-bit platforms, download the standard GISO from [Software Download Center](#) and perform the following steps. Alternatively, to build a GISO consisting of SMU RPMs using customized GISO.

Procedure

Step 1 **install replace** <GISO-location> [**commit** | **noprompt**]

Example:

```
Router#install replace harddisk:/<giso-image>.iso
```

```
+++++
```

```

Install operation 11 started by root:
exec-timeout is suspended.
No install operation in progress at this moment
Label = More_Pkgs
ISO <giso-iso-image>.iso in input package list. Going to upgrade the system to

version <new-giso-image>.
System is in committed state
Current full-label: <giso-image>_R_Commit
Current only-label: R_Commit
Current label: R_Commit
Updating contents of golden ISO
Scheme : localdisk
Hostname : localhost
Username : None
SourceDir : /ws
Collecting software state..
Getting platform
Getting supported architecture
Getting active packages from XR
Getting inactive packages from XR
Getting list of RPMs in local repo
Getting list of provides of all active packages
Getting provides of each rpm in repo
Getting requires of each rpm in repo
Fetching .... <giso-image>.iso
Label within GISO: More_Pkgs
Skipping <platform>-mgb1-3.0.0.0-<release>.x86_64.rpm from GISO as it's active
Adding packages
    <platform>-golden-x-<release>-<Label>.iso
RP/0/RP0/CPU0:Jun 20 14:43:59.349 UTC: sdr_instmgr[1164]: %INSTALL-INSTMGR-2-OPERATION_SUCCESS :

Install operation 12 finished successfully
Install add operation successful
Activating <platform>-golden-x-<release>-<Label>
Jun 20 14:44:05 Install operation 13 started by root:
    install activate pkg <platform>-golden-x-<release>-<Label> replace noprompt
Jun 20 14:44:05 Package list:
Jun 20 14:44:05     <platform>-golden-x-<release>-<Label>.iso
Jun 20 14:44:29 Install operation will continue in the background
exec-timeout is resumed.
Router# Install operation 13 finished successfully
Router: sdr_instmgr[1164]: %INSTALL-INSTMGR-2-OPERATION_SUCCESS :

Install operation 13 finished successfully

Router#install replace <path-to-image> <platform-name-golden-x-<version>-<label>.iso
Tue Mar 17 08:07:15.176 UTC
+++++
Mar 17 08:07:24 Install operation 46 started by root:
Mar 17 08:07:24   install replace source <path-to-image> <platform-name-golden-x-<version>-<label>.iso
Mar 17 08:07:24 No install operation in progress at this moment
Mar 17 08:07:24 Checking system is ready for install operation
Mar 17 08:07:24 'install replace' in progress
Mar 17 08:07:24 Label = GISO_IMAGE_XRV9K <version>
Mar 17 08:07:24 ISO xrv9k-goldenk9-x-<version>-<label>.iso in input package list. Going to upgrade
the system to version <new-version>
Mar 17 08:07:25 Scheme : http
Mar 17 08:07:25 Hostname : 10.x.x.x
Mar 17 08:07:25 Collecting software state..
Mar 17 08:07:25 Getting platform
Mar 17 08:07:25 Getting supported architecture
Mar 17 08:07:25 Getting active packages from XR
Mar 17 08:07:25 Getting inactive packages from XR

```

```

Mar 17 08:07:28 Getting list of RPMs in local repo
Mar 17 08:07:28 Getting list of provides of all active packages
Mar 17 08:07:28 Getting provides of each rpm in repo
Mar 17 08:07:28 Getting requires of each rpm in repo
Mar 17 08:07:36 Fetching .... xrv9k-goldenk9-x-<version>-<label>.iso
Mar 17 08:08:02 Adding packages
      xrv9k-goldenk9-x-<version>-<label>.iso
Router:Mar 17 08:09:03.487 UTC: sdr_instmgr[1281]: %INSTALL-INSTMGR-2-OPERATION_SUCCESS : Install
operation 47 finished successfully
Mar 17 08:09:03 Install add operation successful
Mar 17 08:09:08 Activating xrv9k-goldenk9-x-<version>-<label>
Mar 17 08:09:10 Install operation 46 started by root:
      install activate pkg xrv9k-goldenk9-x-<version>-<label> replace
Mar 17 08:09:10 Package list:
Mar 17 08:09:10      xrv9k-goldenk9-x-<version>-<label>
This install operation will reload the system, continue?
[yes/no]:[yes] yes
Mar 17 08:10:30 Install operation will continue in the background
Mar 17 08:10:30 Activate operation ID is: 46 for 'install source' ID:46

Router# Install operation 46 finished successfully
%INSTALL-INSTMGR-2-OPERATION_SUCCESS : Install operation 46 finished successfully
sdr_instmgr[1150]: %INSTALL-INSTMGR-2-SYSTEM_RELOAD_INFO : The whole system will be reloaded to
complete install operation 46

```

Note

The md5sum of the GISO files is checked automatically during this operation to ensure that the image has not been corrupted. A mismatch in md5sum value indicates that the file is manipulated, and the operation fails.

For Cisco IOS XR Release 7.1.1, use the command `install replace harddisk:/<dir>/<giso-image>.iso`.

Important

For versions earlier than Cisco IOS XR Release 6.5.2, use the following command:

install update source *<absolute-path-of-Golden-ISO>* **replace**

For example,

```
Router#install update source harddisk:/ <giso-image>.iso replace
```

The version and label of the newly added GISO is compared with the version and label of the currently active version. If a mismatch is identified, a new partition is created and the full package is installed. After installation, the system reloads with the image and packages from the newly added GISO.

Note

Activating or deactivating on a system that has a valid label invalidates the label. This action is irreversible. For example, running **show version** command on the system displays the label 6.3.3_633rev1005. If any SMU is activated or deactivated on the system, the label 633rev1005 is invalidated, and the `show version` command displays only 6.3.3 as the label.

Step 2 **show version****Example:**

```

Router#show version
Wed Jun 20 15:06:37.915 UTC
Cisco IOS XR Software, Version <new-giso-image>
Copyright (c) 2013-2018 by Cisco Systems, Inc.

Build Information:
Built By      : <user>
Built On     : <date>

```

```
Build Host      : <host-name>
Workspace      : <workspace-name>
Version        : <version>
Location       : <path>
Label          : <label-name>

cisco <platform> () processor
System uptime is 3 hours 51 minutes
```

The system loads with the image and packages from the newly added GISO.

