HSI-HTAR Installation

30 September 2025: Hierarchical Storage Interface 11.3

Table of Contents

Chapter 1. Overview

Starting with HSI-HTAR 10.1, RPM installation became the preferred way to install HSI-HTAR software. Prior to the HSI-HTAR 10.1 release, HSI-HTAR was installed after building from a source tarball. If you still need to build HSI-HTAR from a source tarball, refer to the instructions provided in Appendix A. Note that the products of a source build are RPMs that can then be installed using the RPM installation instructions provided in the following chapters.

The build environment for the RPMs is contained in the file `hsi_pkg_includes`. It is generated by the *Configure* script during the build process, as described in Appendix A of this document. As such, the build environment information for the source tarball is in `config/hsi_pkg_includes`. For binary RPMs, the build environment information can be found in `etc/BUILDCONF.rpms`, which is a copy of the `config/hsi_pkg_includes` file used in building the RPMs.

HSI-HTAR is implemented as a client-server internet application. The installation packaging reflects this implementation. Currently, only Linux distributions are supported for both server and client RPMs. Note that <*distribution*> and <*arch*> below refer to the Linux distribution and the system architecture that the RPM was built for.

Chapter 2. Installation preparation

This section offers pre-installation considerations, as well as a list of prerequisite software for each of the RPMs.

Generally, a list of RPM dependencies can be generated by

```
rpm -qpR <rpm name>
```

If the rpm command does not specify a version for a dependent package that it returns then the Linux distribution default version will work.

To view additional checking via scripts that may be done at the RPM installation, run

```
rpm -qp <rpm name> --scripts
```

2.1. Server RPM

The server RPM contains the HSIGWD binary, which is the HSI Gateway Daemon. This daemon communicates directly with HPSS Core Servers. Consequently, any system the daemon is installed on will also need to have the HPSS instance configuration installed on it as well. This would include the HPSS instance's `HPSS.conf` file (usually found in `/var/hpss/etc`). The runtime configuration for the HSIGWD is placed in this `HPSS.conf` file by the RPM install (unless it already exists). Configuration files for logging (via rsyslog) and systemd are also installed by this RPM (unless the files already exists). These files are system-level configuration files and are located in `/etc` and its subdirectories. Review release notes, verify configurations, and run any site-specific or targeted testing before installing the daemon on a production system.

Because HSIGWD needs a copy of the HPSS instance's `/var/hpss/etc` in order to function with the HPSS instance, there are some sites that run HSIGWD on the HPSS Core system. However, it is possible for you to use a separate system to run the HSIGWD. In fact, this approach is recommended in the case of large or active HPSS instances. The primary requirement is that a non-HPSS Core system has a copy of the HPSS instance's `/var/hpss/etc` directory, prior to installation.

Prerequisite software for the server RPM:

- hpss-lib, version 11.3
- jansson
- krb5-libs (optional)
- libtirpc
- munge-libs
- openssl-libs
- pam

2.2. Client RPM

The client RPM contains the HSI and HTAR binaries. It also contains a basic client configuration in a client-side `HPSS.conf` file, and a basic wrapper script that runs the binaries. These files should be reviewed by the installer or system administrator to verify that both the wrapper and the `HPSS.conf` file will work in the installed environment.

This RPM can be installed on a Linux system which is supported by the RPMs distributed by the HPSS Collaboration.

The symbolic links (*hsi*, *htar*) to the wrapper script are installed in the `/usr/local/bin` directory of the client system. In order for users of the client systems to access HSI and HTAR after installation, `/usr/local/bin` needs be added to the system's PATH variable. If you want to use a different link installation directory, then recreate these links in your preferred directory *after the RPM installation is completed*. Because of the complexities of building these RPMs, the link installation directory is not configurable at installation time.

The HPSS Mover systems also need to be able to reach these client systems. Network access from the HPSS Mover systems should be tested as part of the installation process.

For more information on supported systems, refer to the HSI-HTAR Release Notes located on the HPSS AdminWiki website:

```
https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=start
```

Prerequisite software for the client RPM:

- krb5-libs (optional)
- libedit
- libtirpc
- munge-libs
- ncurses-libs
- openssl-libs
- pam

Chapter 3. Install HSI-HTAR with RPMs

HSI-HTAR RPMs can be obtained either from the HPSS Admin Wiki, or from the HPSS Collaboration website for registered HPSS customer sites. RPMs are the primary method of installing HPSS software.

- https://hpss-collaboration.clearlake.ibm.com/adminwiki/doku.php?id=start
- http://hpss-collaboration.org

Installation of HSI-HTAR with RPMs uses typical RPM options and commands. As `root`, install the RPM using the following command:

```
rpm -ivh <rpm-name>
```

There are two primary RPM packages for HSI-HTAR:

Table 1. RPM packages

Package	Description
hsihtar-clt	Contains the HSI and HTAR binaries. It also contains a basic client configuration in a client-side `HPSS.conf` file, and a basic wrapper script that runs the binaries.
hsihtar-svr	Contains the HSI Gateway Daemon (HSIGWD) binary. It also contains system-level configuration files for logging (rsyslog) and running with systemd.

For a more complete debugging experience, additional debuginfo RPMs are also available at the above websites. Starting with RHEL8, the debugsource RPM should also be installed. Note that the debuginfo and debugsource RPMs may contain source files. A noarch documentation RPM is also provided that contains installation and user documentation.

The `rpm` commands given below assumes that the current working directory (CWD) is the same directory where the RPMs are located. Give the full path to the RPM, if this is not the case.

3.1. Install Server RPM

The command to install the server RPM is as follows:

```
rpm -ivh hsihtar-svr-11.3-0.<distribution>.<arch>.rpm
```

3.1.1. Post-install tasks

1. Verify the system files.

- /usr/lib/systemd/system/hpss-hsigwd@.service
- /usr/lib/systemd/system/hpss-hsigwd.socket
- /etc/rsyslog.d/hpss-hsigwd.conf

If the SYSTEMD configuration fails to be enabled properly during the RPM install, it can be enabled manually with the following commands

```
   sudo cp -p /hpss_src/hsihtar-11.3-0/etc/hpss-hsigwd@.service
/hpss_src/hsihtar-11.3-0/etc/hpss-hsigwd.socket /usr/lib/systemd/system
   sudo systemctl daemon-reload
   sudo systemctl status hpss-hsigwd.socket
```

2. Verify the HSIGWD configuration in /var/hpss/etc/HPSS.conf. Refer to Appendix D in the *HPSS Installation Guide* for a discussion on the syntax and variables used in this file. In particular, pay close attention to the HSIGWD sub-stanza of the HSI stanza. This contains the actual configuration of the HSIGWD. As an example of things to consider when reviewing the HSIGWD configuration, for sites using Kerberos authentication, make sure that the Server Auth krb5 is turned on. If a krb5 password is being used, make sure that is also turned on. Likewise, for sites using unix authentication, make sure that the Server Authentication Mechanism is set to *unix*. For keytab authentication, select, uncomment, and if necessary edit the pathname. In HPSS.conf, see the following example and remove the ";" to uncomment the configuration lines.

```
850   # Authentication mechanism that server uses to get HPSS creds
851   # Valid settings are "unix" and "krb5"
852   ;Server Authentication Mechanism = unix
853    Server Authentication Mechanism = krb5
854
855   # Authenticator type that server uses to prove its identity
856   # Legal values are auth_none, auth_keytab, auth_keyfile, auth_key,
auth_passwd
857   # Currently, the only supported value is "auth_keytab"
858   Server Authenticator Type = auth_keytab
859
860   # Authenticator that server uses to prove its identity.
861   # The value of this flag depends upon the Server Authenticator Type.
862   # For auth_keytab, it is the pathname of the keytab file for the server
863      ;Server Authenticator = /var/hpss/etc/hpss.unix.keytab
864      Server Authenticator = /var/hpss/etc/hpss.keytab
```

The RPM install should append a default HSIGWD configuration to /var/hpss/etc/HPSS.conf. If this does not happen for some reason, the default HSIGWD configuration is in /hpss_src/hsihtar-11.3-0/etc/hpss-hsigwd.config, and can be appended manually to the end of /var/hpss/etc/HPSS.conf. Without the HSIGWD configuration appended to the end of /var/hpss/etc/HPSS.conf, the HSIGWD service will fail to start.

3. Restart the system services.
   - rsyslog

     ```
     sudo systemctl start rsyslog
     sudo systemctl status rsyslog
     ```

   - hpss-hsigwd.socket

     ```
     sudo systemctl start hpss-hsigwd.socket
     sudo systemctl status hpss-hsigwd.socket
     ```

4. Create the COS list used by HSI and move it into `/var/hpss/etc`. The `/opt/hpss/bin/lshpss` executable needs to be on the machine that the `make_cos` runs on, as that script calls **lshpss**. As such, if HSIGWD is being run on a separate system, then `make_cos` typically needs to be moved and run on the HPSS Core system, and the resulting COS List (*cos* file) needs to be moved back to the system running HSIGWD. Be sure to review any resulting COS List before copying it to `/var/hpss/etc`. Assuming that HSIGWD is running on the HPSS Core, then the command to generate the COS List is:

   ```
   /hpss_src/hsihtar-11.3-0/bin/make_cos
   cp cos /var/hpss/etc
   ```

3.1.2. Uninstall command

To uninstall the server RPM, the command is as follows:

```
rpm -evh hsihtar-svr-11.3-0.<distribution>.<arch>
```

Since hsihtar-svr may (or may not) install system-level configuration files, these files are not deleted or touched when this RPM is uninstalled. It will be the responsibility of the system administrator to analyze and modify any modifications made by the RPM installation of this package to these configuration files.

3.2. Install Client RPM

The command to install the client RPM is as follows:

```
rpm -ivh hsihtar-clt-11.3-0.<distribution>.<arch>.rpm
```

3.2.1. Post-install tasks

1. Verify symbolic links for HSI and HTAR in `/usr/local/bin`.

```
ls -l /usr/local/bin/hsi /usr/local/bin/htar
env | grep PATH
```

So that users can access these symbolic links, you may have to add `/usr/local/bin` to the system PATH variable.

2. Check and set, the value of HPSS_SERVER_HOST and other variables in /hpss_src/hsihtar-11.3-0/bin/wrapper. The completion of this task is absolutely needed, since the installed value of HPSS_SERVER_HOST is the machine on which the client RPM was built. The client processes will not be able to contact the HSIGWD otherwise.

You may need to change the variable value WRAPPER_USE_KEYTAB, if there is a site-specific way of handling keytab authentication for HTAR. By default, WRAPPER_USE_KEYTAB is set to *on* for the RPM installed. Set this to *off*, in order to avoid using the logic in this wrapper script to handle HTAR's keytab authentication, and allow site-specific logic to be used instead.

3. Review *Network Options* in /hpss_src/hsihtar-11.3-0/etc/HPSS.conf. Recall that this is the client's `HPSS.conf` file, and is read by the HSI and HTAR processes on the client systems. Refer to Appendix D in the *HPSS Installation Guide* for a discussion on the syntax, configuration variables, and values used in this stanza.

Another stanza to review in /hpss_src/hsihtar-11.3-0/etc/HPSS.conf is *PFTP Client Interfaces*. This stanza (commented out by default) can aid in determining which network interface the client will use when transferring data to and from the HPSS instance. Because specific network interfaces can be configured in *PFTP Client Interfaces*, this stanza may be useful in resolving connection issues between the client and the HPSS mover(s). If this stanza is not configured, then the client's default network interface will be used to transfer data to and from HPSS.

4. Add /hpss_src/hsihtar-11.3-0/man to system envirnoment variable MANPATH.

3.2.2. Uninstall command

To uninstall the client RPM, the command is as follows:

```
rpm -evh hsihtar-clt-11.3-0.<distribution>.<arch>
```

Appendix A: Building HSI-HTAR from source

Overview

The source build of HSI-HTAR uses the build automation tool CMAKE. In order to reduce the complexity of building this package, two build scripts have been developed, namely *Configure* and *Compile*.

Table 2. Build scripts

Script	Description
Configure	Configures the build environment for HSI-HTAR, based on local build environments. Sets various build variables used by **Compile**. This script writes these values out to `config/hsi_pkg_includes`.
Compile	Runs CMAKE with the build environment values specified in `config/hsi_pkg_includes`.

The following list identifies the development software packages that must be installed on the build system prior to building HSI-HTAR from source code:

Depending upon whether the client, server (HSIGWD), or documentation is being built, not all of the packages listed may need to be installed on the build system.

Prerequisite software for the build of HSI-HTAR Source Tree

- asciidoctor with asciidoctor-pdf (needed for document build)
- cmake, version 3.7 or later
- docbook-style-xsl, version 1.70 or later (needed for document build)
- hpss-lib, version 11.3 (needed for server build)
- hpss-lib-devel, version 11.3 (needed for server build)
- jansson-devel
- java-<version>-openjdk, version 1.8.0 or later (needed for document build)
- javapackages-filesystem, version 5.3.0 or later (needed for document build)
- krb5-devel
- libedit-devel
- libtirpc-devel
- munge-devel
- ncurses-devel
- openssl-devel
- pam-devel
- python3, version 3.6 or later

- python3-distro
- python3-ruamel-yaml

The general steps for building the HSI-HTAR interface are similar to other software packages. They are as follows and provide the organization of this appendix:

1. Obtain HSI-HTAR source tree.
2. Run **Configure**.
3. Perform the build by running **Compile**.

HSI-HTAR Source Tree

Obtain the HSI-HTAR source tree from HPSS support.

Run Configure

The **Configure** script has two modes that can be run in, namely *batch* and *interactive*. The default mode is batch. In order to run **Configure** in the default mode, in the root of the source tree, type:

```
./Configure
```

Configure behaves similar to the `configure` script that is used by GNU's Autotool, in that it tries to automatically detect the build environment of the local machine. For example, it finds the default compiler, as well as the CMAKE program. Like `configure`, values for certain build variables can be specified on the command line. For example, in order to specify the exact path for CMAKE under `/opt`, the **Configure** command is:

```
./Configure --with-cmake /opt/bin/cmake
```

Using the `--with-*` command line options will allow build customization without having to run *Configure* in interactive mode. For example, to use a non-standard SSL version for the build. the **Configure** command would look like:

```
./Configure --with-openssl-base <SSL Base path> --with-openssl-libpath <SSL Library
path>
```

To see a complete list of all command line options for **Configure** type:

```
./Configure --help
```

Another build environment item that **Configure** detects is the presence (or absence) of HPSS headers and libraries. These are typically found in `/opt/hpss/include` and `/opt/hpss/lib`. If they

exist on the build system, the **Configure** will allow for the configuration of the server (HSIGWD) build, as well as the client (hsi, htar) build. If the HPSS files do not exist, then **Configure** will only configure the build environment for the client build. It is possible to specify the configuration of either the client build, or the server build, like so:

```
./Configure --client
```

or

```
./Configure --server
```

However, in the case of specifying the server build, the HPSS headers and and libraries need to be on the build system. Otherwise, **Configure** will fail.

If you wish to run **Configure** interactively (in the non-default mode), you will be presented with a series of menus that will allow you to set the build environment variables. To turn on the interactive mode, run **Configure** as follows:

```
./Configure -i
```

A series of line menus will be displayed, like the one below:

```
++++++++++++++++++++++++++++++++++++++++++++++++
Configuring BUILD TOOLS AND LIBRARIES
++++++++++++++++++++++++++++++++++++++++++++++++

This script has made its best guess as to where required build tools
and libraries are located. This screen allows you the chance to
review and change any of these findings before the rest of the
build configuration and actual build takes place.

Once you have made all the changes that you wish to make (if any),
enter "a" at the prompt to continue.
============================================================
0. CC ........................... /usr/bin/cc
1. CMAKE ........................ /usr/bin/cmake
2. CPACK ........................ /usr/bin/cpack
3. HSI_LIBEDIT_SUPPORT .......... on
4. KEYTAB_TMPL .................. %%H/private/.ktb_%%U
5. LIBEDIT_LIBPATH .............. /usr/lib64
6. LIBEDIT_INCPATH .............. /usr/include
7. OPENSSL_BASE ................. /usr
8. OPENSSL_LIBPATH .............. /usr/lib64
============================================================
Please enter 'a', 'h [num]', or [num] < 9
Enter selection:
```

As indicated, standard responses to these menus are:

Table 3. Configure menu responses

Response	Description
a	Accept all variable values listed in the current menu. This response will display the next menu for consideration.
h *num*	Print help text for the variable at line *num* of the menu
num	Allows the value of the variable at line *num* of the menu to be changed.
q	Quits or aborts the current run of **Configure** without saving the build environment file.

At the end of an interactive session, after **Configure** has written out the build environment in `config/hsi_pkg_includes`, it will ask you if you desire to continue with the build, like so:

```
Saving old build parameter cache to config/hsi_pkg_includes.old
Command to build: ./Compile -c config/hsi_pkg_includes -ssl /usr -server -client
Run the build/compile command?
```

Respond either *y* (for yes) or *n* (for no). If your response is *y*, it will run **Compile** with the specified command line.

As indicated above, the **Configure** script will save the build environment settings in the file `config/hsi_pkg_includes`. If it exists, this file is read on subsequent runs of **Configure**, so the settings will not have to be entered again. The `config/hsi_pkg_includes` file can be updated manually, saved off, and used to automate the configuration and build process, if needed.

Run Compile

The **Compile** reads the `config/hsi_pkg_includes` file and applies the build environment using CMAKE. If a `config/hsi_pkg_includes` file exists, it is not necessary to rerun **Configure** in order to run the **Compile** script.

By default, **Compile** creates a build directory under the source root directory. The default name of this build directory is `./bld-<hostname>-<arch>-<dist version>`. For example, after running **Compile**, the following build directory may exist in the source tree root:

```
<HSI-HTAR Source Tree>/bld-io-linux_x86_64-rhel8.8
```

All temporary objects and build products are placed in this build directory. In order to clean up from a build, or to remove a build, this directory is the only object you need to delete from the source tree. **Compile** will create this directory, if it does not exist.

Compile also contains logic to do all the packaging of the HSI-HTAR product. Specifically, **Compile** is used to generate the RPMs discussed in the main part of this installation document.

Like **Configure**, **Compile** takes command line switches. However, in its most basic invocation, no arguments are needed, like so:

```
./Compile
```

This command will simply build all of the executables, build targets, and configuration files, and place them in the build directory. Options to the **Compile** script target or add to the products it builds. In order to see what options **Compile** accepts, run:

```
./Compile --help
```

Output from such a command will look similar to:

```
usage: Compile [-h] [-a ARCH] [-b BDIR] [-c CFILE] [-client] [-docs] [-noarch]
               [-pkg] [-server] [-ssl SSLDIR] [-v]

Compile -- wrapper to build HSI/HTAR software

optional arguments:
  -h, --help   show this help message and exit
  -a ARCH      Build Platform Architecture
  -b BDIR      User Build Directory. Should be a scratch directory or not
               exist
  -c CFILE     A Configuration file. Works best as an absolute path
  -client      Build Client
  -docs        Build Formatted Documentation
  -noarch      Creates the noarch RPM packages associated with HSI-HTAR. This
               option implies -pkg
  -pkg         Creates the native installation package (i.e. RPM) for the
               build platform.
  -server      Build Server (hsigwd)
  -ssl SSLDIR  OpenSSL Installation Directory
  -v           Verbose Build Output
```

Particular options to note are `-c CFILE` and `-b BDIR`. The `-c` option allows for the specification of a different build configuration than `config/hsi_pkg_includes`. The `-b` option allows for a different name or location of the build directory used in the build. Also, the `-docs` option generates *all* of the documentation, including the *HSI-HTAR Reference Manual*.