4

Performing an Active>360 installation

This chapter describes how to install Active>360 software. The topics covered in this chapter apply to both on-premises and cloud-based Active>360 installations.

NOTE: If you want to deploy Active>360 on a cloud-based server (such as an Amazon EC2, Google Compute Engine, Microsoft Azure, or Oracle Cloud instance), please contact Ab Initio Support for advice and assistance before proceeding with the instructions in this chapter.

This chapter covers the following topics:

Step 1. Temporarily disable Co>Operating System variables

Before performing the Active>360 installation, you must temporarily disable several Co>Operating System variables on the host on which you will run the Active>360 installation scripts.

TIP: As part of the installation process, Active>360 installs its own Co>Operating System on each target host and exports required environment variables as appropriate. Because of this, most Co>Operating System variables that are exported in your shell environment prior to performing an Active>360 installation will typically not conflict with Active>360 after the installation has been completed. The purpose of this step is to prevent conflicts during the Active>360 installation process itself.

► To temporarily disable Co>Operating System parameters:

Perform these steps in a Korn shell on the Co>Operating System host on which you will run the Active>360 installation script.

1. Disable the following variables in the Co>Operating System environment for the user account under which the Active>360 installation will be performed:
   - AB_AIR_ROOT
   - AB_EME_REPOSITORIES

   You may need to disable these variables in the installation user's **~/.abinitiorc** file, shell environment, or both.

2. Temporarily rename the installation user's **~/.abinitiorc** file, or comment out variables in the **~/.abinitiorc** file, as follows:

   a. In most cases, to reduce the risk of conflicts with the Active>360 installation scripts, we recommend temporarily renaming the installation user's **~/.abinitiorc** file; for example:

   ```
   mv ~/.abinitiorc ~/not.abinitiorc
   Perform the Active>360 installation.
   mv ~/.not.abinitiorc ~/.abinitiorc
   ```

   TIP: Active>360 itself does not require any values in the installation user's **~/.abinitiorc** file.

   b. If it is not possible or desirable to temporarily rename the **~/.abinitiorc** file, edit the file to comment out at least the following variables:
      - AB_AIR_HOME
      - AB_AIR_ROOT
      - AB_EME_REPOSITORIES
      - AB_HOME

   After completing the Active>360 installation, you can restore the installation user's **~/.abinitiorc** file to its original state.

3. Continue with "Step 2. Set and unset shell environment variables".

Step 2. Set and unset shell environment variables

Before performing the Active>360 installation, you must set or temporarily unset several environment variables on the host on which you will run the Active>360 installation scripts.

► **To set and unset environment variables:**

Perform these steps in a Korn shell on the Co>Operating System host on which you will run the Active>360 installation script.

1. Export (set) the following environment variables in your shell environment:

Variable	Notes
AB_HOME	Path to the Co>Operating System that you will use to run the Active>360 installation script.
AB_APPLICATION_HUB	Path to the Application Hub that you will use to run the Active>360 installation script.
AB_CHARSET	In most cases, set this value to **utf-8**, which is the default value used for the Active>360-specific Co>Operating System environments that are created by the Active>360 installer. **WARNING!** Ensure that the value for AB_CHARSET is the same on all Co>Operating System environments that you will use with Active>360, including the Co>Operating System that you specify with AB_HOME, above. If you need to use an AB_CHARSET value other than **utf-8**, consult with your Ab Initio representative. For more information, see "Choosing a character set (AB_CHARSET)".
JAVA_HOME	Must point to Java version 1.8 or later.
TMP_DIR	(Optional) Path to the system temporary directory; recommended if you want to point to a non-default directory. The user account under which the Active>360 installation will be performed must have read/write access to the directory specified here. For example, by default, the system temporary directory is typically **/tmp**, but the installation user account might not have read/write privileges to that directory, or there might not be enough available disk space in that directory to perform the installation. In such cases, you should set the value of TMP_DIR to a directory other than **/tmp**.
PATH	Must include **$AB_HOME/bin** and **$JAVA_HOME/bin**.

For example, your exports might be as follows:

```
export AB_HOME=/usr/local/abinitio
export AB_APPLICATION_HUB=/usr/local/abinitio-app-hub
export AB_CHARSET=utf-8
export JAVA_HOME=/etc/alternatives/jre
export TMPDIR=/disk1/tmp
export PATH=$AB_HOME/bin:$JAVA_HOME/bin:$PATH
```

2. Unset the following environment variables:
   - **PG_HOME**
   - **LD_LIBRARY_PATH**
   - **PYTHONPATH**

   For example, your unset commands might be as follows:

   ```
   unset PG_HOME
   unset LD_LIBRARY_PATH
   unset PYTHONPATH
   ```

3. Continue with "Step 3. Obtain and extract the Active>360 installation package".

Step 3. Obtain and extract the Active>360 installation package

Active>360 Version 4.1.6.0 software is distributed as a single GZIP-compressed TAR file named **Active360_V4-1-6-0.tgz**.

► **To obtain and extract the Active>360 installation package:**

1. Contact your Ab Initio representative for instructions on obtaining the Active>360 installation package.
2. Extract the Active>360 product package file to the directory of your choice.

   For example:

   ```
   cp Active360_V4-1-6-0.tgz /disk1/installer
   cd /disk1/installer
   tar zxvf Active360_V4-1-6-0.tgz
   ```

   or

   ```
   tar zxvf Active360_V4-1-6-0.tgz -C /disk1/installer
   ```

   where **-C /disk1/installer** refers to the path under which the **Active360_V4-1-6-0.tgz** file will be extracted. If you use the **-C** option, the directory that you specify (for example, **/disk1/installer**) must already exist.

   Using either of these examples, the Active>360 installer files are extracted to a directory named **Active360_V4-1-6-0**. For the remainder of these instructions, this directory is referred to as *act_installer*.
3. Continue with "Step 4. Prepare an Active>360 installation configuration file".

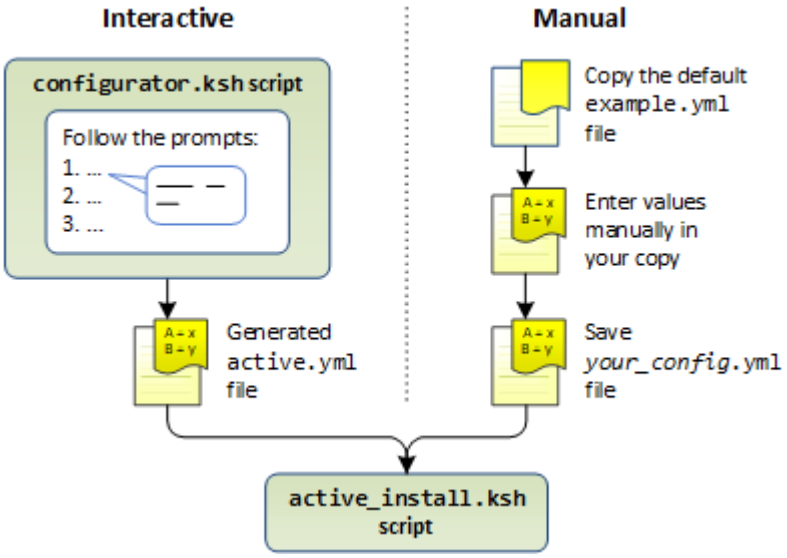
Step 4. Prepare an Active>360 installation configuration file

The Active>360 installation script, **active_install.ksh**, requires a number of deployment-specific configuration parameters. You define these parameters in an installation configuration file, which you then use as input to the Active>360 installation script.

This section contains the following topics:

How to prepare an installation configuration file

You can prepare an installation configuration file either manually, using the *act_installer*/**example.yml** file as a template, or interactively, using the guided **configurator.ksh** script. We strongly recommend using the guided **configurator.ksh** script.



The following are some of the key differences between using the **configurator.ksh** script to generate an **active.yml** file and creating or editing a configuration file manually:

Using configurator.ksh	Manual editing
• You follow guided prompts to enter parameter values using a simplified single-line syntax. In most cases, you do not need to think about formal YAML syntax at all. • The script presents parameter names, displays default values when available, and prompts you accept the given default or enter a new value. • The script performs basic error-checking, and guides you through combinations of parameters that generally do not conflict with each other. • The script reveals only relevant parameters and hides irrelevant parameters, based on your choices as you move through script. • The script automatically generates a well-formed YAML-compliant configuration file, named **active.yml**, that you can use with the **active_install.ksh** script.	• You must understand and strictly adhere to formal YAML syntax rules. The only form of guidance is the *act_installer***/example.yml** file in the Active>360 installation package. • You must consider all parameters and be aware of any potential interactions between them. • Manual editing provides the greatest flexibility, but also presents the greatest risks of typographical errors and conflicting or missing parameter values. For example, in formal YAML syntax, all line breaks, punctuation, indentation levels, and parameter hierarchies are significant; if any of these are wrong, the Active>360 installation will fail. • It is often unclear that your configuration file contains errors until after an Active>360 installation attempt has failed.

For more information about the differences between using the guided **configurator.ksh** script to generate an installation configuration file automatically compared to creating or modifying a configuration file manually, see "configurator.ksh syntax versus formal YAML syntax".

Using the configurator.ksh script to generate an installation configuration file

NOTE: We strongly recommend using the interactive **configurator.ksh** script to generate your YAML-compliant Active>360 installation configuration file.

► To use the configurator.ksh script to generate an installation configuration file:

Perform the following steps in a Korn shell on the installation host.

1. Change to the *act_installer* directory; for example:

   ```
   cd /disk1/install/Active360_V4-1-6-0
   ```

2. Start the **configurator.ksh** script; for example:

   ```
   ./configurator.ksh
   ```

   NOTE: You must run the **configurator.ksh** script from within the *act_installer* directory.

   The **configurator.ksh** script starts, and asks if you want to use **basic** mode; for example:

   ```
   ./configurator.ksh
   Unpacking python environment to launch install
   Would you like to use basic mode? y/n:
   ```

3. Specify whether you want run the **configurator.ksh** script in **basic** mode or **custom** mode, as follows:

Response	Mode	Notes
y	**basic**	• You will be prompted to provide values for a limited number of parameters. • Most parameters will use the default values (shown in the *act_installer***/example.yml** file).
n	**custom**	• You will be prompted to provide values for a greater number of parameters. • The specific parameters shown will vary depending on your responses as you move through the script.

4. Provide parameter values, as prompted.

   Unless prompted otherwise, enter each given parameter value on a single line. In cases where you want to provide multiple values for a given parameter, enter the values on a single line, with each value separated by a comma without spaces between; for example:

```
=> Only needed when ab.key.users.source is server. The URLs of the key servers
that are in the key server group specified by AB_KEYSERVER_GROUPS.
=> Enter values separated by commas
=> ab.key.client.users.urls:
abks://keyserver1.bigcorp.com:6151,abks://keyserver2.bigcorp.com:6151
```

   For descriptions, guidelines, and detailed syntax for each installation configuration parameter, see "Active>360 installation script configuration parameter reference".

   When you have finished responding to all prompts, the **configurator.ksh** script automatically generates an Active>360 installation configuration file, named **active.yml**, in the *act_installer* directory.

   NOTE: The generated **active.yml** file will include only those parameters for which you have specified a value or specified a value other than the default — that is, the **active.yml** file will not typically contain all of the parameters that are listed in the **example.yml** file.

   TIP: For your reference, the **configurator.ksh** script saves a complete record of your configuration session in a file named *act_installer***/interactive.txt**.

5. Continue with "Step 5. Run the Active>360 installation script".

Preparing an installation configuration file manually

NOTE: We strongly recommend using the interactive **configurator.ksh** script to generate your YAML-compliant Active>360 installation configuration file. We do not recommend preparing your configuration file manually. For **configurator.ksh** instructions, see "Using the configurator.ksh script to generate an installation configuration file".

► To prepare an installation configuration file manually:

Perform the following steps in a Korn shell on the installation host.

1. Change to the *act_installer* directory; for example:

```
cd /disk1/install/Active360_V4-1-6-0
```

2. Copy the **example.yml** file to a file with a name and location of your choosing; for example:

   ```
   cp example.yml acttest.yml
   ```

   You will use this copy of the **example.yml** file as a template for your own configuration file.

3. Edit the configuration parameters in the copied file, as required for your installation environment.

   For details about each of the configuration parameters, see "Active>360 installation script configuration parameter reference".

4. Continue with "Step 5. Run the Active>360 installation script".

Step 5. Run the Active>360 installation script

After you have prepared an Active>360 installation configuration file, as described in "Step 4. Prepare an Active>360 installation configuration file", you will use that configuration file with the *act_installer*/**active_install.ksh** script to perform the Active>360 installation.

This section contains the following topics:

Running the installation script

Perform the following steps in a Korn shell on the installation host.

► **To run the Active>360 installation script:**

1. Change to the *act_installer* directory; for example:

   ```
   cd /disk1/install/Active360_V4-1-6-0
   ```

2. Run the **active_install.ksh** script, using the **-config** argument to specify the name of your installation configuration file; for example:

   ```
   ./active_install.ksh -config ./active.yml
   ```

   For **active_install.ksh** syntax, see "active_install.ksh syntax".

   The Active>360 installation tasks will proceed unattended until the installation is complete. Depending on your configuration settings and installation environment, the installation may take several hours.

3. When the installation finishes, continue with "Step 6. Export Active>360 environment variables".

active_install.ksh syntax

```
./active_install.ksh [ -h | --help ] [ -config config_file ]
     [ -release-loc dir_name ] [ -logfile log_file ]
     [ -d | --debug ] [ --verbosity verbosity_level ]
     [ -validation-only ] [ -stage-install-only ]
     [ -suppress-dynamic-param-functions ]
```

All command arguments are optional.

Argument	Description
-h , **--help**	Display help for this command, then exit.
-config *config_file*	The name and directory path of an installation configuration file (see "Step 4. Prepare an Active>360 installation configuration file"). If this argument is omitted, the installation script will look for a file named **active.yml** in the *act_installer* directory. If an **active.yml** file is not found, the installation script will exit.
-release-loc *dir_name*	The directory in which the Active>360 installer package is located (*act_installer*); defaults to the current directory.
-logfile *log_file*	The name and directory path of a file to which installation script logging output should be sent. This log file is an addition to the Ansible playbook log that is generated by default during Active>360 installation. For more information about the Ansible playbook log and other output generated during Active>360 installation, see "Step 7.5 Review Active>360 installer output and installation logs".
-d, **--debug**	Include additional debugging information in the installer log file.
--verbosity *verbosity_level*	Specify the level of verbosity to capture in the Ansible playbook log. The default value is **0**. This value can be a single integer, from **0** through **4**, with higher values capturing greater detail. **NOTE:** The verbosity level is unaffected by the state of the debug (**-d**) argument.
-validation-only	Only validate the parameter values in the Active>360 installation configuration file; do not actually perform the installation.
-stage-install-only	Only stage the Ansible step; do not actually perform the Active>360 installation.
-supress-dynamic-param-funcs	Suppress internal dynamic functions; use only with guidance from Ab Initio Support.

Usage

Note the following:

- You can use relative or absolute paths for all directories and filenames.
- The log file generated with the optional **-logfile** *log_file* argument does not replace the Ansible playbook log that is always generated during Active>360 installation. For more information about the Ansible playbook log, see "Step 7.5 Review Active>360 installer output and installation logs".
- The optional **--debug** argument causes additional debugging information to the Ansible playbook log and the TTY output that is generated during installation script only. Debugging information is not added to the log file that is generated with the **-logfile** *log_file* argument.

Examples

- To start the **active_install.ksh** script from the *act_installer* directory using an installation configuration file named **acttest.yml** that is also in the *act_installer* directory, enter the following:

  ```
  ./active_install.ksh -config ./acttest.yml
  ```

- To use the same **acttest.yml** configuration file, as above, and also generate debugging output and a log file named **acttest.log**, enter the following:

  ```
  ./active_install.ksh -config ./acttest.yml -logfile ./acttest.log -d
  ```

- To start the **active_install.ksh** script from the *act_installer* directory using an installation configuration file with the default name, **active.yml**, enter the following:

  ```
  ./active_install.ksh
  ```

Step 6. Export Active>360 environment variables

The Active>360 installer creates a reference file, named **$act.root/bin/env.sh**, that contains shell environment variables that are specific to your Active>360 deployment.

In order to directly interact with an Active>360 deployment in a command-shell environment, you must first export several deployment-specific variables in that environment. For example, you must export Active>360 environment variables in order to use Co>Operating System and Active>360 utilities like **m_env**, **ab-key**, **act-admin**, **adc**, **ab-db**, **ab-app**, **air**, and so forth.

NOTE: In order for the commands in "Step 7. Verify the Active>360 installation" to succeed, you must export Active>360 environment variables, as described below.

► **To export Active>360 environment variables:**

Perform the following steps in a Korn shell on each Co>Operating System host on which Active>360 will run.

1. (Optional) Open a new terminal shell session on the host on which you performed "Step 5. Run the Active>360 installation script".

   This step is optional but recommended in order to ensure that you start from a "clean" shell environment — for example, so that environment variables like PATH and LD_LIBRARY_PATH do not contain conflicting values or become inordinately long after you export the Active>360 environment variables in the next step, below.

2. Source the contents of the **$act.root/bin/env.sh** file, where **$act.root** is the root directory in which Active>360 is installed.

   For example, if Active>360 is installed in a directory named **/disk1/acttest**, you would enter the following command:

   ```
   . /disk1/acttest/bin/env.sh
   ```

3. (Optional) Export the PG_HOME environment variable on each Active>360 Co>Operating System host so it points to the relevant Active>360 Co>Operating System installation.

   This step is optional but recommended to avoid conflicting PG_HOME paths with non-Active>360 Co>Operating System installations.

   a. Export the PG_HOME variable; for example:

      ```
      export PG_HOME=$AB_HOME/lib/postgresql
      ```

   b. Update the LD_LIBRARY_PATH variable to include PG_HOME; for example:

      ```
      export LD_LIBRARY_PATH=PG_HOME/lib:$LD_LIBRARY_PATH
      ```

4. Continue with "Step 7. Verify the Active>360 installation".

Step 7. Verify the Active>360 installation

After exporting Active>360 environment variables, as described in "Step 6. Export Active>360 environment variables", perform the following steps to verify the Active>360 installation:

- Step 7.1 Verify Active>360 Co>Operating System environments
- Step 7.2 Verify core Active>360 services
- Step 7.3 Verify Active>360 application and observability status
- Step 7.4 Verify general Active>360 health status
- Step 7.5 Review Active>360 installer output and installation logs
- Step 7.6 Review Active>360 port assignments and URLs

Step 7.1 Verify Active>360 Co>Operating System environments

The Active>360 installer creates an Active>360-specific Co>Operating System environment on each target host in the Active>360 installation. For summaries of the various target hosts in an Active>360 installation, see "Planning your Active>360 hosts".

TIP: The Active>360 installation target hosts are defined in your Active>360 installation configuration file, in the parameters under the **act.hosts** parent parameter. For more information, see "Step 4. Prepare an Active>360 installation configuration file".

The purpose of this step is to verify that your AB_HOME and AB_APPLICATION_HUB Co>Operating System variables point to locations in your Active>360 installation directories, that the Ab Initio bridge and bridge security are configured correctly and running, and that your Co>Operating System environments are properly keyed.

► **To verify Active>360 Co>Operating System environments:**

Perform these steps in a Korn shell on each Active>360 installation target host, starting with the Active>360 administration host.

1. Verify that the paths for AB_HOME and AB_APPLICATION_HUB begin with the root directory in which Active>360 is installed (**act.root** installation configuration parameter).

   These paths should be the same on the Active>360 administration host and all target installation hosts.

   For example, if **$act.root** is a directory named **/disk1/acttest**, use the **m_env** command to verify the values for AB_HOME and AB_APPLICATION_HUB, as follows:

```
m_env AB_HOME AB_APPLICATION_HUB

Variable              Set  Value or Resulting Action
--------------------  ---  ------------------------------------------
AB_APPLICATION_HUB     *   /disk1/acttest/usr/local/abinitio-app-hub \
  (from environment)
AB_HOME                *   /disk1/acttest/usr/local/abinitio-V4-1-5-4 \
  (from environment)
```

If the paths shown for AB_HOME and AB_APPLICATION_HUB do not begin with the directory corresponding to **$act.root**, verify that you have exported Active>360 environment variables correctly, as described in "Step 6. Export Active>360 environment variables".

2. Verify that the Co>Operating System is keyed correctly; for example:

```
ab-key show -current
A key file for this installation was found:

Path: /disk1/acttest/var/abinitio/data/abkc/data/kdkeystore.xml
Key server: abks://keyserver2.bigcorp.com:6153
Offline grace period ends: 2023-01-26 18:35:37 (in 1 day and 56 minutes)
Key ID: template.002.desc.txt
Products: All Products (expires 2023-02-01, warns 3 days ahead)
```

   Active>360 requires an "All Products" key on each host.

3. Check the status of the Ab Initio bridge that was created during Active>360 installation.

   a. Show the bridge and bridge security names; for example:

```
ab-bridge list
bridge-29070          29070 Ab Initio Bridge configuration
ab-bridge security list
act-security-config   Ab Initio Bridge security configuration
bridge-tools          'Ab
```

   b. Show bridge details and status:

```
ab-bridge show bridge_name
ab-bridge status -name bridge_name
```

      where *bridge_name* is the name of the bridge that was created by the Active>360 installer.

      For example, using the bridge name that is highlighted in Step **3.a**, you should see output similar to the following:

```
ab-bridge show bridge-29070
Name:                        bridge-29070
Description:                 Ab Initio Bridge for Active>360
Protocol:                    HTTP
Port:                        29070
Bridge user:                 actadmin
User switching mechanism: Local default (Impersonation script)
Log verbosity:               Terse: RPC invocation, terse, audit, warning,
error, and informational messages
Status page authentication: Status pages disabled
Security configurations:  act-security-config
EME authentication mode:  Enabled

ab-bridge status -name bridge-29070
Bridge 'bridge-29070' is running and listening on port 29070.
```

   The important things to verify in this step are that the bridge and bridge security names listed in Step **3.a**, above, match the names listed in Step **3.b**. Also make certain that the name of the bridge user is the same username that was used for the **ab.aiuser_name** parameter in the Active>360 installation configuration file.

   TIP: By default, the name and port for the Active>360 bridge is **bridge-***listener_port*, where *listener_port* is the value of **act.base_port_prefix** plus **70**. For example, if the value of **act.base_port_prefix** is **290**, then the Active>360 bridge would be **bridge-29070**. For more information, see "Understanding Active>360 port assignments and URLs".

4. Verify that the Active>360 EME Technical Repository is running and contains the primary top-level directories; for example:

```
air ls Projects/abinitio
directory /Projects/abinitio/: 1 elements
        project: stdenv
```

Step 7.2 Verify core Active>360 services

The purpose of this step is the verify that the core Active>Data and Active>360 Messaging services are running.

► To verify core Active>360 services:

Perform these steps in a Korn shell on the Active>360 administration host.

1. Verify that all engines in the Active>360 Active>Data cluster are running; for example:

```
cd $ACT_HOME/sand/abinitio/act/main/run
adc info engines lobby
acthost.control acthost.bigcorp.com 17733  up
acthost.data.01 acthost.bigcorp.com 17788  up
acthost.data.02 acthost.bigcorp.com 17791  up
acthost.publisher.01 acthost.bigcorp.com 17792  up
```

   TIP: If an Active>Data engine is listed as **down**, you can use the **adc engine start** command to start it. For example, in the listing above, if the engine named **acthost.data.02** was listed as **down**, you would use the following command to start it:

```
adc engine start ./lobby acthost.data.02
```

2. Check the status of the Active>360 Messaging subsystem.

   a. Ensure that the message bus **kafka** processes are running; for example:

```
pgrep -l kafka
17781 adc-kafka-consu
17792 adc-kafka-publi
```

   You should see at least two entries: one for the message consumer parent process (**adc-kafka-consu**), and one for the message publisher parent process (**adc-kafka-publi**).

   b. List the **kafka** brokers; for example:

```
cd $ACT_HOME/kafka
./bin/zookeeper-shell.sh acthost:29090 ls -R /brokers/ids
Connecting to acthost:29090

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
/brokers/ids
/brokers/ids/0
```

c. Display details about the **kafka** brokers; for example:

```
./bin/zookeeper-shell.sh acthost:29090 get -s /brokers/ids/0
Connecting to acthost:29090

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
{"features":{},"listener_security_protocol_map":{"INTERNAL":"PLAINTEXT", \
  "CLUSTER":"PLAINTEXT","EXTERNAL":"PLAINTEXT"}, \
  "endpoints":["INTERNAL://acthost:29091","CLUSTER://acthost:29092", \
  "EXTERNAL://acthost:29093"],"jmx_port":29042,"port":29091, \
  "host":"acthost","version":5,"timestamp":"1674601719730"}
cZxid = 0x19
ctime = Tue Jan 24 18:08:39 EST 2023
mZxid = 0x19
mtime = Tue Jan 24 18:08:39 EST 2023
pZxid = 0x19
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x100004dd2d70000
dataLength = 307
numChildren = 0
```

TIP: The **zookeeper** port (**29090** in these examples) that you should use for these commands is derived by appending **90** to the value of the **act.base_port_prefix** installation configuration parameter. For example, if the value of **act.base_port_prefix** that was used during the Active>360 installation is **290**, the **zookeeper** port that you should use is **290 + 90 = 29090**. For more information see, "Understanding Active>360 port assignments and URLs".

Step 7.3 Verify Active>360 application and observability status

The Active>360 installation creates and starts several applications and an observability stack on the Active>360 administrative host and all Active>360 target hosts. The following procedure describes how to check the status of these applications and the observability stack.

Perform the following tasks in a Korn shell on the Active>360 administrative host and on all relevant Active>360 target hosts..

► **To verify Active>360 application and observability status:**

Perform these steps in a Korn shell on the relevant Active>360 installation target hosts, starting with the Active>360 administration host.

1. Check the status of the Active>360 **ab-app** and **ab-db** deployments that were created during Active>360 installation; for example:

```
ab-app status
Instance: act_mhub, on port: 29010, is running.
Instance: activeportal, on port: 29000, is running.
Instance: ag, on port: 29020, is running.
Instance: controlcenter, on port: 29060, is running.

ab-db status
Instance Name         Port       Status     Owner
-------------         ----       ------     -----
act_mhub              29014      Running    actadmin
ag                    29024      Running    actadmin
controlcenter         29064      Running    actadmin
```

   If you installed the demonstration version of Authorization Gateway or Control>Center with Active>360, those applications and databases should also appear in this list. If you installed standalone versions of these products, they will not be shown in this list.

2. Check the status of the components in the Active>360 observability stack; for example:

```
cd $ACT_HOME/bin
./ab-grafana.sh status
running
./ab-prometheus.sh status
running
./ab-timescaledb.sh status
running
```

3. (Optional) Show additional information about the TimescaleDB background worker processes; for example:

```
pgrep -a postgres | grep -i timescale
7005 postgres: TimescaleDB Background Worker Launcher
7165 postgres: TimescaleDB Background Worker Scheduler
```

4. (Optional) If installed, show the status of Active>360 Reference Application (RFA); for example:

```
$ACT_HOME/rfa/bin/rfa.ksh status
RFA status: running
```

   TIP: If the RFA status is **down**, you can use the following command to start the RFA:

```
$ACT_HOME/rfa/bin/rfa.ksh start
```

Step 7.4 Verify general Active>360 health status

Active>360 includes an administration utility, named **act-admin**), in the **$ACT_HOME/bin** directory. Among other tasks, you can use the **act** command to check the general health of your Active>360 deployment.

► **To verify general Active>360 health:**

Perform these steps in a Korn shell on the Active>360 administration host.

1. Change to the **$ACT_HOME/bin** directory; for example:

```
cd $ACT_HOME/bin
```

2. Run the **act-admin status** command; for example:

```
act-admin status
============================================================
Running Ansible... This will take a few minutes (up to 30)
============================================================
============================================================
{"localhost": {"changed": 0, "failures": 0, "ignored": 0, \
  "ok": 2, "rescued": 0, "skipped": 0, "unreachable": 0}, \
  "acthost": {"changed": 11, "failures": 0, "ignored": 0, \
  "ok": 554, "rescued": 0, "skipped": 81, "unreachable": 0}}
============================================================
```

   In addition to summary information, like that shown above, the command returns detailed information about numerous Active>360 processes. This detailed information is displayed in your console window as raw, unformatted text.

3. In the summary information returned by the **act-admin status** command, as shown above, check the status of the various Ansible tasks; for example, you might want to look for any failures.

4. If the summary information includes any failures, check the Ansible playbook log for details; for example:

```
less $ACT_HOME/utils/ansible/ansible-playbooks/logs/ansible-playbook.log
```

   TIP: The Ansible playbook log typically contains a lot of information, starting with information about the Active>360 installation run itself. To jump to the location in the log at which the results for the **act-admin status** command begin, search for the date (and optionally the time) at which you ran the command.

   For example, if you ran the **act-admin status** command on January 26, 2023, and you use the **less** command to view the Ansible playbook log, you would enter the following command in the **less** command bar:

```
/2023-01-26
```

Step 7.5 Review Active>360 installer output and installation logs

The following sections describe the various types of logs that are generated by the Active>360 installer. If you encounter any errors when installing Active>360, these log files can often be helpful in determining the cause of the errors.

- Ansible playbook log
- Console (TTY) output
- Optional installation log file
- Optional debugging output
- interactive.txt file
- inv.yml file
- run_install.ksh file
- .release.config and release_config.*timestamp*.cfg files

Ansible playbook log

The primary installation actions performed by the Active>360 installer are implemented as sets of Ansible automation playbook tasks. The results of these Ansible playbook tasks are recorded in an Ansible playbook

log file. Compared with the other log types, the playbook log provides the most detailed information about the results of an Active>360 installation run.

The Ansible playbook log is written to a file named **$ACT_HOME/utils/ansible/ansible-playbooks/logs/ansible-playbook.log**.

TIP: When you run the **active_install.ksh** script, you can use the **--verbosity** argument to specify the level of detail that you want to capture in the playbook log. For more information, see "active_install.ksh syntax".

For troubleshooting Active>360 installation errors, it is often useful to jump to the **PLAY RECAP** section at the end of the Ansible playbook log, which indicates the number of failed playbook tasks, if any; for example:

```
tail -3 $ACT_HOME/utils/ansible/ansible-playbooks/logs/ansible-playbook.log
2023-01-24 20:36:38,026 p=10485 u=abinitio n=ansible | PLAY RECAP ****
2023-01-24 20:36:38,027 p=10485 u=abinitio n=ansible | acthost :  ok=1216  \
  changed=385  unreachable=0  failed=1  skipped=2525 rescued=0  ignored=37
```

Console (TTY) output

During Active>360 installation, the installer writes a continuous stream of console (TTY) output to the command shell from which the installation script was run.

The contents of this TTY output are mostly the same as the contents of the Ansible playbook log. The primary difference between the two log types is the inclusion of timestamps in the playbook log showing the start and finish datetime for each playbook task.

Optional installation log file

You can generate an optional installer log file by running the Active>360 installation script with the **-logfile** *log_file* option.

The contents of this optional installer log file show the progress of the Active>360 installation before, during, and after the Ansible playbook task stage.

For more information about generating an installer log file, see "active_install.ksh syntax".

Optional debugging output

If you choose to generate an installer log file, you can use the **-d** option to include additional debugging information in the console TTY output and optional installer log. This optional debugging argument does not affect the details that are captured in the Ansible playbook log.

interactive.txt file

When you use the **configurator.ksh** script to generate an **active.yml** installation configuration file, the **configurator.ksh** script saves a complete record of your configuration session in a file named *act_installer***/interactive.txt**.

The **interactive.txt** file contains all prompts and responses from your configuration session in the order in which they occurred. This information can be useful for double-checking your configuration choices.

inv.yml file

The **active_install.ksh** script automatically generates a file named **inv.yml** file in the *act_installer* directory.

This **inv.yml** file contains your installation script configuration parameters plus a number of additional derived parameters in an Ansible-friendly YAML format.

When troubleshooting an Active>360 installation, it can sometimes be useful to review the contents of the **inv.yml** file to verify the actual installation parameters that were passed to Ansible.

run_install.ksh file

Related to the **inv.yml** file, the **active_install.ksh** script also automatically generates a script named **run_install.ksh** in the *act_installer* directory.

This **run_install.ksh** script prepares the shell environment for the Active>360 installation, and invokes the **$ACT_HOME/bin/ansible-playbook** command, passing the **inv.yml** file as input.

As with the **inv.yml** file, it can sometimes be useful to review the contents of the **run_install.ksh** script to verify that shell environment and Ansible invocation command are correct.

.release.config and release_config.*timestamp*.cfg files

These are not log files, but contain metadata about the Active>360 installation package, including subsystem components and minimum software requirements for the specific version of Active>360 that you are installing.

The **.release.config** file is located in the *act_installer* directory. The **release_config.***timestamp***.cfg** is located in the **$ACT_HOME/config** directory. Both files contain exactly the same content.

When troubleshooting an Active>360 installation, it can sometimes be useful to review either of these files to verify that your installation environment matches the list of required components and versions.

Step 7.6 Review Active>360 port assignments and URLs

After a successful Active>360 installation run, the Ansible playbook log displays a list of URLs for various Active>360 applications and services; for example:

```
tail $ACT_HOME/utils/ansible/ansible-playbooks/logs/ansible-playbook.log
  msg:
  - 'RFA URL: http://acthost:29005/rfa'
  - 'Active>Portal URL: http://acthost:29000/activeportal'
  - 'Active MHub URL: http://acthost:29010/act_mhub'
  - 'AG URL: http://acthost:29020/ag'
  - 'Runtime Service URL: http://acthost:29078'
  - 'Kafka Bootstrap Servers: acthost:29092'
```

For information about the port assignments and URLs that are generated automatically during Active>360 installation, see "Understanding Active>360 port assignments and URLs".