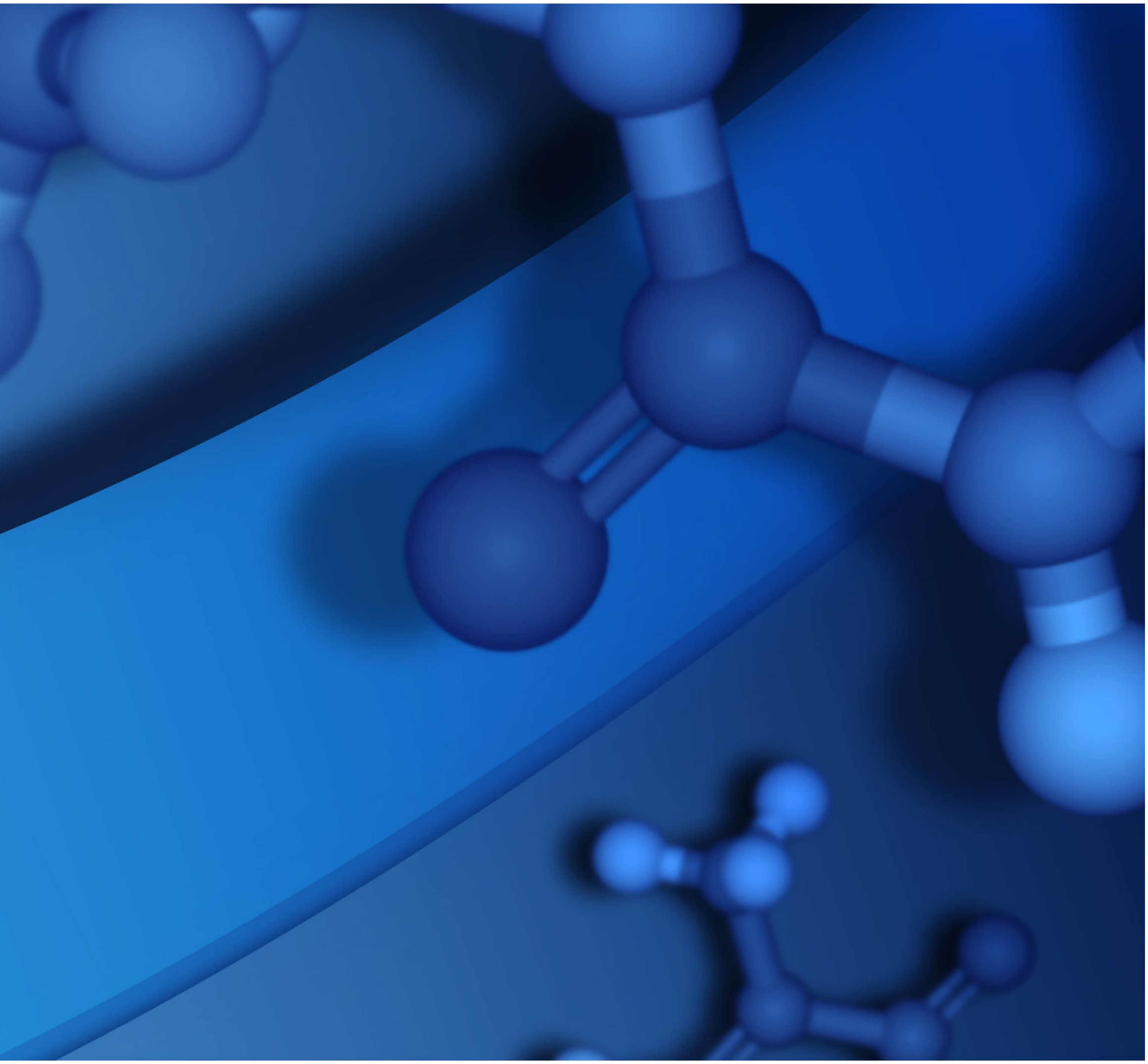
ANSIBLE INSTALLATION GUIDE

BIOVIA 2021



Contents

Chapter 1: Overview

This guide is intended for system administrators who want to use Ansible scripts to install BIOVIA applications.

About Ansible

Ansible is an automation tool that you can use to deploy and configure software in an enterprise setting. BIOVIA provides Ansible playbooks and a template inventory file for you to manage installation of supported BIOVIA applications.

For more information about Ansible, see https://docs.ansible.com.

Supported Applications

Ansible deployment scripts are available for the following BIOVIA applications:

- Foundation Hub (Windows and Linux)
- Pipeline Pilot (Windows and Linux)
- Compose and Capture (Windows)
- Workbook (Windows)
- CISPro (Windows)
- Notebook (Windows)

Notebook Installations

The Ansible installation via Notebook is primarily intended for deployments on single-tenant cloud. It is supported for other environments with the following limitations:

- Support for Oracle only.
- Load balancing deployments are not supported.
- Enabling of SSL and configuration to connect to Foundation Hub must be done manually.
- Support for new installation only. Upgrades should be performed using the Notebook Upgrade application.

Before Starting

Review the system requirements and installation documentation for the BIOVIA applications you plan to install, and review System Requirements for Ansible on page 2.

What's New for 2021

- Added support for CISPro.
- Added support for Notebook.

Chapter 2: Setting up Ansible

To use Ansible to install BIOVIA applications, you must install and configure it on a dedicated server.

System Requirements for Ansible

Ansible Version

Version	Notes
2.8.2	■ Sudo privileges are required. ■ If you are using an older version of Ansible, see Upgrading Ansible on page 4.

Ansible Server Requirements

You must install Ansible on a dedicated server that meets the following requirements.

Hardware

Component	Minimum Hardware
Processor	■ x64 architecture ■ 2 CPU's
RAM	4 GB
Hard disk space	20 GB
Swap Space	N/A
Network	1 GB

Operating System

Operating System	Version	Notes
Red Hat® Enterprise Linux	7	■ 64-bit only ■ Dedicated server with sudo privileges

Target Node Requirements

Ansible installation is available for the following platforms.

Application	Operating Systems
Foundation Hub	■ Windows® Server 2012® R2 and higher ■ RHEL 7

Application	Operating Systems
Pipeline Pilot	■ Windows Server 2012 R2 and higher ■ RHEL 7
Compose and Capture	■ Windows Server 2012 R2 and higher ■ RHEL 7 *is not* supported
Workbook	■ Windows Server 2012 R2 and higher

User Account Requirements

- **Windows:** Administrative privileges (domain user).
- **Linux:** Sudo access.

Installing Ansible

1. Verify you have the supported operating system:

```
cat /etc/redhat-release
```

2. Verify that you have sudo privileges:

```
sudo -l
```

3. Verify that your Ansible machine has the Extras repository enabled and can access the Python-pip installer.

```
sudo yum repolist
```

```
sudo yum info python-pip
```

If there are issues enabling the `Extras` repository, you may be able to install the `EPEL` repository using the following method:

```
sudo rpm -ihv https://dl.fedoraproject.org/pub/epel/epel-release-latest-
7.noarch.rpm
```

4. Install Ansible and the Python libraries to connect to Windows machines. For detailed instructions, see: http://docs.ansible.com/ansible/latest/intro_installation.html.

```
sudo yum -y install python-pip sshpass wget gcc
```

```
sudo pip install --upgrade pip==20.1
```

```
sudo pip install --upgrade ansible==2.8.2 "jinja2>=2.10" pywinrm boto
passlib jmespath ansible-lint yamllint
```

5. Check that package `requests-ntlm` is version 1.1.0 or later.

```
sudo pip list | grep requests-ntlm
```

This should return the following:

```
> requests-ntlm (1.1.0)
```

If not, upgrade `requests-ntlm`.

```
sudo pip install requests-ntlm --upgrade
```

6. Confirm the installation of version 2.8.2 by running the following command:

```
ansible --version
```

Troubleshooting

If you see errors regarding an invalid Python version or similar problems such as `Invalid environment marker: python_version < 'xx'`, you may need to update package dependencies by running the following (or similar) commands:

```
sudo pip install --upgrade setuptools
```

```
sudo pip install --upgrade pip
```

If you encounter problems while updating dependencies, such as: `Cannot uninstall 'requests'.`, this might be because the package is installed with `distutils` and the system cannot accurately determine which files need to be updated. The solution is to ignore the requests package. For example:

1. Run the command:

   ```
   pip install pywinrm --ignore-installed requests
   ```

2. Retry the update.

Upgrading Ansible

Follow these instructions if you are using a version of Ansible older than 2.8.2.

Note: Older versions of BIOVIA playbooks will not work with this version. Download the latest BIOVIA playbooks and example inventory file. See Downloading and extracting the Ansible Scripts for BIOVIA Applications on page 4.

1. Install Ansible and the Python libraries. For detailed instructions, see: http://docs.ansible.com/ansible/latest/intro_installation.html.

   ```
   sudo yum -y install python-pip sshpass wget gcc
   ```

   ```
   sudo pip install --upgrade pip==20.1
   ```

   ```
   sudo pip install --upgrade ansible==2.8.2 "jinja2>=2.10" pywinrm boto
   passlib jmespath ansible-lint yamllint
   ```

2. Check that package `requests-ntlm` is version 1.1.0 or later.

   ```
   sudo pip list | grep requests-ntlm
   ```

   This should return the following:

   ```
   > requests-ntlm (1.1.0)
   ```

   If not, upgrade `requests-ntlm`.

   ```
   sudo pip install requests-ntlm --upgrade
   ```

3. Confirm the installation of version 2.8.2 by running the following command:

   ```
   ansible --version
   ```

Downloading and extracting the Ansible Scripts for BIOVIA Applications

1. Create a `BIOVIADeployment` directory on the Ansible server. If you are using a previous version of the Ansible Scripts for BIOVIA applications, rename your existing `BIOVIADeployment` directory.
2. Locate `BIOVIA_<version>.BIOVIADeployment<version>.tgz`, which is available alongside the Foundation Hub installation files.

3. Extract the `.tgz` file:

```
cd BIOVIADeployment
```

```
tar xvf BIOVIA_<version>.BIOVIADeployment<version>.tgz
```

4. Ensure the working directories are not world-writable. Ansible does not support working directories that are world-writable. The `ansible.cfg` file will be ignored if it is found in such a directory. You will be prompted with the following warning:

```
[WARNING] Ansible is in a world writable directory [directory], ignoring
it as an ansible.cfg source.
```

   In this case, remove write permissions for all users on the directory:

```
<sudo> chmod 775 [directory]
```

5. Make a copy of the template inventory file, `inventories/EXAMPLE_ENVIRONMENT/inventory`, so that the original is available if you want to create a new version..

```
mkdir inventories/newtestenvironment
```

```
cp -r inventories/EXAMPLE_ENVIRONMENT inventories/newtestenvironment
```

Changes to the Inventory File Template

Some variables that were available in previous example inventory files have been deprecated and will cause an error if you try to use them with the newer versions of the playbooks. You can configure the new inventory example file, or remove the deprecated variables from the previous version. For Foundation Hub, `hub_upgrade_only` and `hub_force_configuration` have been deprecated and should be removed from your inventory file if you plan to use an older version.

Copying BIOVIA Application Files

Copy the application installer and license files as described here. If you are installing CISPro, you will also need a copy of the Oracle Client installation zip file. You will set variables that point to these files when you configure the inventory file.

1. Copy the installer files to the appropriate server:
   - For applications other than Pipeline Pilot, copy the compressed application installer files to the Ansible server.
   - For Pipeline Pilot, set up an Apache server and copy the compressed installer files there. See Setting up an Apache Server for Pipeline Pilot on page 6.
2. Copy the license files for all applications, including Pipeline Pilot, to the Ansible server.
3. For CISPro installations, copy the Oracle Client zip file to the Ansible server.

Setting up Target Nodes

Windows nodes require additional configuration. For Windows and Linux nodes, you can use a "ping" utility playbook to check connectivity.

Windows

1. Remote desktop to the Windows Server using an account with Administrator privileges.
2. Open a command prompt as administrator.

3. Execute the following command to download and execute the `ConfigureRemotingForAnsible.ps1` script:

   ```
   @powershell -NoProfile -ExecutionPolicy bypass -command "iex ((New-
   Object System.Net.Webclient).DownloadString
   ('https://raw.githubusercontent.com/ansible/ansible/devel/examples/scrip
   ts/ConfigureRemotingForAnsible.ps1'))"
   ```

   If successful, you will receive a response that a self-signed SSL certificate was generated.

   Alternatively, you can download the `ConfigureRemotingForAnsible.ps1` script from https://github.com/ansible/ansible/blob/devel/examples/scripts/ and execute it separately.

4. Test the connection between Ansible and the Windows nodes. Enter the user credentials for the target nodes when prompted.

   ```
   ansible-playbook -i inventories/newtestenvironment/inventory ping-
   win.yml
   ```

Linux

Linux nodes do not require additional configuration. You can test connectivity using the ping-linux.yml playbook.

```
ansible-playbook -i inventories/newtestenvironment/inventory ping-linux.yml
```

Setting up an Apache Server for Pipeline Pilot

For large installations such as Pipeline Pilot, it is recommended that you provision an Apache server with the Ansible server to distribute installer files to target nodes.
If you cannot use this method, copy the installer to the Ansible server and in the inventory file, set `plp_req_installer_zip_loc` to that location and set `plp_req_get_pps_from_apache` to false.

1. In the Ansible deployment scripts directory, run the `provision.yml` playbook to setup the local Apache server:

   ```
   sudo ansible-playbook provision.yml -i <inventory file>
   ```

   where `<inventory file>` is the relative path from the `provision.yml` file.

2. Verify that the Apache index page opens from a machine on the same network as the Ansible server. Navigate to: `http://<Ansible server IP/Hostname>`, where `<Ansible server IP/Hostname>` is the IP address of the server or its fully qualified hostname. For example `http://ansiblehost.localdomain` or `http://127.0.0.1`.

3. Copy the uncompressed Pipeline Pilot installer files to the Apache server.

   ```
   mv ./<pp installer>.7z /opt/biovia_installers/PipelinePilot/
   ```

4. Verify that the installer file is visible from the Apache URL. Using any machine on the same network as the Ansible server, navigate to `http://{{ Ansible Control Machine IP/Hostname }}/PipelinePilot`. You should see the installer file that you moved into the `/opt/biovia_installers/PipelinePilot/` directory in the Apache server's `/PipelinePilot` index.

Chapter 3:
Configuring the Inventory File

Inventory files contain variables used for the installation of the applications. The inventory file defines the target nodes (hosts), host variables that pertain to specific applications being installed, and global variables that apply to all nodes and applications.

Inventory File Syntax

- Comments are delimited by hashtags #.
- In the template, optional fields are commented, and required fields are not commented.
- Some variables have default values that are used when the variable is commented out. The descriptions of the variables in this chapter list default values.
- Variable with double curly brackets have values that are replaced by Ansible. For example, `{{ ansible_fqdn }}` will be replaced with the fully qualified domain name of the Ansible server.
- **DO NOT** include line breaks between entries within a group. The following is an example of an invalid entry:

  ```
  [hub-windows]
  hub1.website.com
  hub2.website.com

  hub3.testsite.com
  ```

Parts of the Example Inventory File and Variable Groups

The example inventory file contains sections for each of the applications. Within each application section are groups of variables to define the hosts that will be the target of the installation and groups of variables that are specific to the application being installed. Additionally there are global variables that apply to all hosts and applications, and Ansible variables that control how Ansible is run.

Hosts Groups

The `[<application name>-<operating system>]` header defines the hosts groups for the application and supported operating systems. Include the fully qualified domain name of your target nodes in these groups.

Example

```
[hub-windows]
hub1.website.com
hub2.website.com
hub3.testsite.com
```

Notes on Load Balancing

For *Foundation Hub* and *Compose and Capture*, all target nodes in the inventory file must either not be load-balanced or must belong to the same load-balanced cluster. This is not a limitation for *FoundationSSO* or *Pipeline Pilot*.

Variable Groups

Variable groups contain groups of variables that are applied to the hosts when running a playbook. These groups contain `:vars` in the name.

- `[<application name>:vars]`: Variables in this group apply to all hosts, regardless of the platform they belong to. For example, variables in the `[hub:vars]` group apply to all the hosts defined in `[hub:windows]` and `[hub:linux]` groups.
- `[<application name>-<operating system>:vars]`: Variables in this group apply only to hosts for the specified operating system. For example, variables in the `[hub-windows:vars]` group only apply to the hosts defined in the `[hub:windows]` group.

See the sections for each application in this chapter for a description of the host variables for each application.

Ansible Variables

The variables below the `ANSIBLE VARIABLES - DO NOT EDIT BELOW THIS LINE` section contain global variables that apply to all applications, Ansible variables that are set to prompt for credentials when the playbook is run, and variables that define the groups and group hierarchy of the inventory file.

Customizing the Example Inventory File

BIOVIA provides a single template inventory file with sections and variables for all of the supported applications. You can create variations of the inventory file for specific applications and deployments as long as you keep the variable groups and Ansible Variables section intact.

Ansible Vault

Ansible Vault allows you to keep sensitive data such as passwords or keys in encrypted files, rather than as plain text in a playbook or inventory file. This section discusses how to create a password protected Ansible Vault file that you can use to maintain sensitive values for variables and how to reference them from the inventory file. For details about Ansible Vault, see the Ansible Vault section of the *Ansible User Guide*: https://docs.ansible.com/ansible/latest/user_guide/vault.html.

Using Ansible Vault

1. Run the following command to create the vault file in the same directory as the plain text inventory file.
   ```
   ansible-vault create inventories/EXAMPLE_ENVIRONMENT/inventory_secrets
   ```
2. Create a new Ansible Vault password.
   ```
   New Vault password:
   Confirm New Vault password:
   ```
   The new vault file opens in a vi editor.
3. Edit the vault file.
   - Use the variable names defined in the inventory prepended with `ansible_vault_`. For example, for the inventory file variable `hub_dataSource_password_plaintext`, name the Ansible Vault file variable `ansible_vault_hub_dataSource_password_plaintext`.
   - The variables must be organized into the same groups used in the inventory file. For example, `hub_dataSource_password_plaintext` is a member of the `[hub:vars]` group in the

inventory file. The corresponding AnsibleVault variable, `ansible_vault_hub_dataSource_password_plaintext` must also be a member of a `[hub:vars]` group.

Example

```
[hub:vars]
ansible_vault_hub_dataSource_password_plaintext=yourpassword
ansible_vault_hub_keystore_password_plaintext=yourpassword
```

4. Press **Esc** then `:wq!` to write the file and quit the vi editor. Confirm the file is encrypted by using the following command.

```
more inventories/EXAMPLE_ENVIRONMENT/inventory_secrets
```

Example

```
$ANSIBLE_VAULT;1.1;AES256

39626337366262616464646565623134653531353261623333666465396262333062646465313936

3334353237663136643031346165633137393436643332640a64386238363963623863313937626

5

6133306436303266666134643262333866393663323931336639623063663135643162613330333

7

3832376239346236660a64343036323035313032666164336262346138333363343830633439626

4
373334376662376266636439346465386232396261626534393165333066326630 61
```

5. Edit the inventory file variables to reference the variables you defined in the Ansible Vault file.

```
[hub:vars]
hub_dataSource_password_plaintext="{{ ansible_vault_hub_dataSource_
password_plaintext }}"
hub_keystore_password_plaintext="{{ ansible_vault_hub_dataSource_
password_plaintext }}"
```

6. When you run the playbook for the inventory that references the Ansible Vault file, include the flag, `--ask-vault-pass`.

Example

```
ansible-playbook -i inventories/EXAMPLE_ENVIRONMENT ping-win.yml --ask-
vault-pass
```

You will be prompted for your Ansible Vault password.

```
Vault password:
```

Editing an Ansible Vault File

To make changes to an existing encrypted Ansible Vault file.

1. Run the following command:

```
ansible-vault edit inventories/EXAMPLE_ENVIRONMENT/inventory_secrets
```

2. At the prompt, enter your Ansible Vault password.

```
Vault password:
```

3. Make changes in the vi editor.
4. Press **Esc** then `:wq!` to write the file and quit the vi editor.

Foundation Hub Variables

Configure the following inventory file variables for installations of Foundation Hub.

[hub-windows]

The `[hub-windows]` group defines the fully qualified domain names for Windows target nodes for Foundation Hub.

Example

```
[hub-windows]
hub-server1.example.domain
hub-server2.example.domain
```

IMPORTANT! For load-balanced deployments, all target nodes must belong to the same load-balanced cluster (point to the same load balancer). Do not include nodes from multiple load balancers in the same inventory file.

[hub-linux]

The `[hub-linux]` group defines the fully qualified domain names for Linux target nodes for Foundation Hub.

Example

```
[hub-linux]
hub-server3.example.domain
hub-server4.example.domain
```

IMPORTANT! For load-balanced deployments, all target nodes must belong to the same load-balanced cluster (point to the same load balancer). Do not include nodes from multiple load balancers in the same inventory file.

[hub-windows:vars]

The variables in the `[hub-windows:vars]` group apply to the Windows target nodes for Foundation Hub.

Variable	Description
`hub_win_local_source_path`	Linux path to the Foundation Hub Windows installer file on the Ansible control machine. **Required** New installation, Upgrade
`hub_win_install_dir`	Directory on the Windows target node to install Foundation Hub. **Default** `C:\Program Files\BIOVIA\Foundation`

Variable	Description
`hub_win_temp_dir`	Temporary directory on the target node where the Foundation Hub installer will be copied. **Default** `C:\temp`

[hub-linux:vars]

The variables in the `[hub-linux:vars]` group apply to the Linux target nodes for Foundation Hub.

Variable	Description
`hub_linux_local_source_path`	Linux path to the Foundation Hub Linux installer file on the Ansible server. **Required** New installation, Upgrade
`hub_linux_install_dir`	Directory on the Linux target node to install Foundation Hub. Do *not* set this to a user home directory. **Default** `/opt/BIOVIA/Foundation`
`hub_linux_tmp_dir`	Temporary directory on the target node where the Foundation Hub installer will be copied. Do *not* set this to a user home directory. **Default** `/tmp/foundation_ansible`
`hub_linux_backup_location`	Linux host backup directory. Do *not* set this to a user home directory. **Default** `/tmp/hub_config_temp`
`hub_install_user`	User that owns the hub installation folder. This must be a local user. The username will be created if it does not exist. **Default** `biovia`
`hub_install_group`	Group that owns the hub installation folder. **Default** `biovia`

[hub:vars]

The variables in the `[hub:vars]` group apply to all Foundation Hub target nodes.

Deployment

Variable	Description
`hub_license_path`	Linux path to the Foundation Hub license file on the Ansible control machine, not the target machine (for example, `/path/hub.lic`). **Required** New installation
`hub_server_user`	Administrator user for Foundation Hub. **Default** `scitegicadmin`
`hub_server_password`	Administrator password for Foundation Hub. **Default** `passwordforscitegicadmin`
`hub_iq_log`	Toggles generation of IQ reporting content. **Default** `true`

Application Configuration

The Application Configuration variables affect settings that are stored in the `app-config.groovy` file for Foundation Hub. These variables are only used when you specify the `hub_ansible_action=reconfigure` action when running the playbook. The original `app-config.groovy` file will be backed up and overwritten.

Variable	Description
`hub_dataSource_url`	URL for the Oracle database to be used for Foundation Hub. **Required** New installation **Default** `jdbc:oracle:thin:@//database.host:1521/orcl`
`hub_dataSource_username`	Username for the Oracle database. **Required** New installation
`hub_dataSource_password`	Encrypted password for the Oracle database. **Required** New installation
`hub_dataSource_password_plaintext`	Unencrypted password credential for the Oracle database. **Required** New installation

Variable	Description
`hub_selfRegistration_rootUrl`	If not load-balanced, this is the fully qualified domain name and HTTP port of the Foundation Hub server (for example, `http://hub.dsone.3ds.com:9954`). If load-balanced, this is the fully qualified domain name and HTTPS port of the load balancer (for example, `https://loadbalancer.dsone.3ds.com:8853`). **Default** `http://{{ ansible_fqdn }}:9954`
`hub_selfRegistration_sslRootUrl`	If not load-balanced, this is the fully qualified domain name and HTTPS port of the Foundation Hub server (for example, `http://hub.dsone.3ds.com:9953`). If load-balanced, this is the fully qualified domain name and HTTPS port of the load balancer (for example, `https://loadbalancer.dsone.3ds.com:8853`). **Default** `https://{{ ansible_fqdn }}:9953`
`hub_selfRegistration_nodeRootUrl`	If not load-balanced, leave this empty. If load-balanced, this is the fully qualified domain name and HTTPS port of the Foundation Hub server (for example, `https://hub.dsone.3ds.com:9953`).
`hub_selfRegistration_nodeSslRootUrl`	If not load-balanced, leave this empty. When load-balanced, this is the fully qualified domain name and HTTP port of the Foundation Hub server (for example, `http://hub.dsone.3ds.com:9954`).
`hub_session_globalTimeout`	Foundation Hub setting for session timeout regardless of activity. **Default** `'8h'`
`hub_session_inactivityTimeout`	Foundation Hub setting for session timeout after a period of inactivity. **Default** `'30m'`
`hub_environment_hosted`	Whether the deployment will be integrated with the 3DS platform. This configuration setting removes the header bar from the Foundation home page. **Default** `'false'`
`hub_environment_label`	Environment label that displays in the user interface. **Default** `'Development'`

Variable	Description
`hub_additionalConfiguration`	You can use this to add additional configuration settings to the `app-config.groovy` file.

Tomcat Properties

The Tomcat Properties variables affect settings that are stored in the `tomcat.properties` file for Foundation Hub. These variables are only used when you specify the `hub_ansible_action=reconfigure` action when running the playbook. The original `tomcat.properties` file will be backed up and overwritten.

Variable	Description
`hub_http_port`	HTTP port for the Foundation Hub node. **Default** `9954`
`hub_https_port`	HTTP port for the Foundation Hub node. **Default** `9953`
`hub_keystore_path`	Path to the keystore file on the target node (for example, `C:\BIOVIA\Foundation\keystore.jks`).
`hub_keystore_path_local`	Linux path to the keystore file on the Ansible machine (for example, `/ansible/directory/keystore.jks`).
`hub_keystore_password`	Encrypted password for the target node keystore.
`hub_keystore_password_plaintext`	Unencrypted password for target node keystore.
`hub_keystore_keyAlias`	Keystore alias for the target node keystore.
`hub_javamelody_enabled`	Melody allows you to diagnose performance issues, determine system load, and so on. **Default** `'false'`
`hub_accesslogging_enabled`	Creates logs of the remote host/IP address URL and response codes in `<hub_install>/logs`. **Default** `'false'`

Pipeline Pilot Variables

Configure the following inventory file variables for installations of Pipeline Pilot.

[plp-windows]

The `[plp-windows]` group defines the fully qualified domain names for Windows target nodes for Pipeline Pilot.

Example

```
[plp-windows]
plp-server1.example.domain
plp-server2.example.domain
```

[plp-linux]

The `[plp-linux]` group defines the fully qualified domain names for Linux target nodes for Pipeline Pilot.

Example

```
[plp-linux]
plp-server3.example.domain
plp-server4.example.domain
```

[plp-windows:vars]

The variables in the `[plp-windows:vars]` group apply to the Windows target nodes for Pipeline Pilot.

Variable	Description
`plp_req_win_temp`	Temporary directory on the target node where the Pipeline Pilot installer will be copied. **Default** `C:\temp`

[plp-linux:vars]

The variables in the `[plp-linux:vars]` group apply to the Pipeline Pilot Linux target nodes only.

Variable	Description
`plp_req_lnx_temp`	Temporary directory on the target node where the Pipeline Pilot installer will be copied. Do *not* set this to a user home directory. **Default** `/tmp`

[plp:vars]

The variables in the `[plp-linux:vars]` group apply to all Pipeline Pilot target nodes.

Server

Variable	Description
`plp_req_lic_path`	Linux path to the Pipeline Pilot license file on the Ansible server (for example, `/path/hub.lic`). **Required** New installation
`plp_req_zip_host_url`	URL of the fully qualified domain name of the Apache server directory that contains the compressed Pipeline Pilot

Variable	Description
	installer file (for example, `http://{{ Ansible Control Machine IP/Hostname }}/PipelinePilot`. Make sure `plp_req_get_pps_from_apache` is set to true. **Required** New installation, Upgrade **Note:** If you cannot use this method, copy the installer to the Ansible server and in the inventory file, set `plp_req_installer_zip_loc` to that location and set `plp_req_get_pps_from_apache` to false.
`plp_req_installer_zip_loc`	Linux path to the zipped installer file on the Ansible machine. Use this only if you cannot use plp_req_zip_host_url. To use this method, set `plp_req_get_pps_from_apache=false`.
`plp_req_install_location`	Location to install the application on the target node. Do *not* set this to a user home directory. **Default** `E:\Program Files\BIOVIA\PPS`
`plp_req_http_port`	HTTP port for Pipeline Pilot. **Default** `9944`
`plp_req_https_port`	HTTPS port for Pipeline Pilot. **Default** `9943`
`plp_req_server_admin_user`	Administrator username for Pipeline Pilot. This does not create a new user. Uncomment and set if the default user name has been changed. **Default** `scitegicadmin`
`plp_req_server_admin_password`	Administrator password for Foundation Hub. This does not create a new user. Uncomment and set if the default password has been changed. **Default** `passwordforscitegicadmin`
`plp_req_generate_iq_report`	Toggles generation of IQ reporting content. **Default** `true`

Foundation Hub Registration

These variable are used when installing Pipeline Pilot as part of a Foundation Hub deployment. In this case, set `plp_req_hub_auto_register` to `true` and define the other variables in this section.

Variable	Description
`plp_req_hub_auto_register`	Automatically register the Pipeline Pilot installation with the Foundation Hub defined by `plp_req_hub_url`. **Default** `false`
`plp_req_hub_url`	URL to the Foundation Hub server (for example, `https://HubServer.domain:9953/foundation/hub`).
`plp_req_hub_user`	Admin username for Foundation Hub.This must be a local user. The username will be created if it does not exist. **Default** `scitegicadmin`
`plp_req_hub_password`	Admin password for Foundation Hub. **Default** `passwordforscitegicadmin`

SSL Security

Variable	Description
`plp_req_hub_ssl_security_config`	Whether to set the SSL Security Level specified in `plp_req_hub_ssl_security_level`. For details, see "SSL Security Level" in the *Pipeline Pilot Admin Portal Guide*. **Default** `false`
`plp_req_hub_ssl_security_level`	Sets the SSL Security Level. For details, see "SSL Security Level" in the *Pipeline Pilot Admin Portal Guide*. **Defaut** `Intermediate_Java6`

Reverse Proxy

These variables are typically used for load-balanced configurations where the reverse proxy is the load balancer.

Variable	Description
`plp_req_rproxy_url`	Fully qualified domain name for the reverse proxy machine. Set only if you are defining a reverse proxy.
`plp_req_rproxy_http_port`	HTTP port for the reverse proxy machine.

Variable	Description
`plp_req_rproxy_ssl_port`	HTTPS port for the reverse proxy machine. **Default** `443`
`plp_req_rproxy_loadbalanced_onoff`	Toggle for load balancing. **Default** `false`
`plp_req_rproxy_aliases`	Semicolon separated list of aliases. Do not include blank spaces.

Domain Authentication

Variable	Description
`plp_req_domain_auth`	Pipeline Pilot Authentication Method. For details, see the *Pipeline Pilot Admin Portal Guide*. **Default** `domain`
`plp_req_domain_auth_config`	Whether to change the Pipeline Pilot Authentication Method specified by `plp_req_domain_auth`. **Default** `false`

Utility Settings

Variable	Description
`plp_req_get_pps_from_apache`	download the Pipeline Pilot installer from the URL for the Apache web server specified by `plp_req_zip_host_url`. **Default** `true`
`plp_req_preserve_ports`	Retail the existing port numbers during an upgrade. **Default** `true`

SSL Certificates

Variable	Description
`plp_req_ssl_import_flag`	Import new SSL certificates. **Default** `false`
`plp_req_ssl_crt_file_location`	Location of a new SSL Certificate CRT file.

Variable	Description
`plp_req_ssl_key_file_location`	Location of a new SSL Certificate KEY file.
`plp_req_ssl_chain_file_location`	Location of a new SSL Certificate CHAIN file.

Configuring the Inventory for the FoundationSSO Package for Pipeline Pilot

The FoundationSSO package for Pipeline Pilot is required for certain versions of Foundation Hub, and must be installed after Pipeline Pilot. See the *Foundation Hub Product Release Document* to determine if you need to install this package.

FoundationSSO is installed using a Pipeline Pilot installer. Use the same variable settings and playbook used for the Pipeline Pilot installation. You will set `plp_req_installer_zip_loc` to point to the FoundationSSO installer. Otherwise, keep the other variables the same.

Compose and Capture Variables

Configure the following inventory file variables for installations of Compose and Capture.

[compose-windows]

The `[compose-windows]` group defines the fully qualified domain names for Windows target nodes for Compose and Capture.

Example

```
[compose-windows]
compose-server1.example.domain
compose-server2.example.domain
```

[compose-windows:vars]

The variables in the `[compose-windows:vars]` group apply to the Windows target nodes for Compose and Capture.

Variable	Description
`compose_win_local_source_path`	Linux path to the Compose and Capture Windows installer file on the Ansible control machine. **Required** New installation, Upgrade
`compose_win_install_dir`	Directory on the target node to install Compose and Capture. **Default** `C:\Program Files\BIOVIA\Compose`
`compose_win_temp_dir`	Temporary directory on the target node where the Compose and Capture installer will be copied. **Default** `C:\temp`

[compose:vars]

The variables in the `[compose:vars]` group apply to all Compose and Capture target nodes.

Deployment

Variable	Description
`compose_upgrade_only`	**Default** `true`
`compose_force_configuration`	If true, Ansible will overwrite the `app-config.groovy` and `tomcat.properties` files using the variable settings in the Compose and Capture Application Configuration and Tomcat Properties sections. **Default** `false`
`compose_license_path`	Linux path to the Compose and Capture license file on the Ansible server. **Required** New installation
`compose_iq_log`	Toggles generation of IQ reporting content. **Default** `true` **Note:** The playbook for Compose and Capture will not set the Pipeline Pilot Reporting Server label. For details, see Pipeline Pilot Reporting Server label for Compose and Capture on page 42.

Application Configuration

The Application Configuration variables affect settings that are stored in the `app-config.groovy` file for Compose and Capture. These variables are only used when you set `compose_force_configuration=true`. If you set any of these variables, the original `app-config.groovy` file will be backed up and overwritten.

Variable	Description
`compose_hubServer_user`	Admin username for the Foundation Hub server to which Compose and Capture will be registered. This must be a local user. The username will be created if it does not exist. **Default** `scitegicadmin`
`compose_hubServer_password`	Encrypted admin password for the Foundation Hub server to which Compose and Capture will be registered.

Variable	Description
`compose_hubServer_password_plaintext`	Unencrypted admin password for the Foundation Hub server to which Compose and Capture will be registered. **Default** `passwordforscitegicadmin`
`compose_hubServer_rootUrl`	The HTTP URL for the Foundation Hub server to which Compose and Capture will be registered (for example, `http://hub.hostname:9954/foundation/hub`).
`compose_hubServer_sslRootUrl`	The HTTPS URL for the Foundation Hub server to which Compose and Capture will be registered (for example, `https://hub.hostname:9953/foundation/hub`).
`compose_dataSource_username`	Username for the Oracle database.
`compose_dataSource_password`	Encrypted password for the Oracle database. Leave this and the `compose_dataSource_password_plaintext` variable blank to be prompted for the password when running the playbook.
`compose_dataSource_password_plaintext`	Unencrypted password credential for the Oracle database. Leave this and the `compose_dataSource_password` variable blank to be prompted for the password when running the playbook.
`compose_dataSource_url`	URL for the Oracle database to be used for Compose and Capture. **Default** `jdbc:oracle:thin:@//database.host:1521/orcl`
`compose_applicationServer_rootUrl`	HTTP URL for the Compose server. **Default** `http://{{ ansible_fqdn }}:9964`
`compose_applicationServer_sslRootUrl`	HTTPS URL for the Compose server. **Default** `https://{{ ansible_fqdn }}:9963`
`compose_proxyServer_rootUrl`	For load-balanced deployments, provide the HTTP URL for the load balancer (for example, `http://loadbalancer.dsone.3ds.com:8864`).
`compose_proxyServer_sslRootUrl`	For load-balanced deployments, provide the HTTPS URL for the load balancer (for example, `https://loadbalancer.dsone.3ds.com:8863`).
`compose_displayTheme_nohead`	Whether the deployment will be integrated with the 3DS platform. This configuration setting removes the header bar from Compose and Capture home page. **Default**

Variable	Description
	`'false'`

Tomcat Properties

The Tomcat Properties variables affect settings that are stored in the `tomcat.properties` file for Compose and Capture. These variables are only used when you set `compose_force_configuration=true`.The original `tomcat.properties` file will be backed up and overwritten.

Variable	Description
`compose_http_port`	HTTP port for Compose and Capture. **Default** 9964
`compose_https_port`	HTTPS port for Compose and Capture. **Default** 9963
`compose_keystore_path`	Path to the keystore file on the target node (for example, `C:\BIOVIA\Compose\keystore.jks`).
`compose_keystore_path_local`	Linux path to the keystore file on the Ansible machine (for example, `/ansible/directory/keystore.jks`).
`compose_keystore_password`	Encrypted password for the target node keystore.
`compose_keystore_password_plaintext`	Unencrypted password for target node keystore.
`compose_keystore_keyAlias`	Keystore alias for the target node keystore.

Workbook Variables

Configure the following inventory file variables for installations of Workbook.

[workbook-windows]

The `[workbook-windows]` group defines the fully qualified domain names for Windows target nodes for Workbook.

Example

```
[workbook-windows]
workbook-server1.example.domain
workbook-server2.example.domain
```

[workbook-windows:vars]

The variables in the `[workbook-windows:vars]` group apply to the Windows target nodes for Workbook.

General Installation

Variable	Description
`wb_installroot`	Windows path to install Workbook. **Default** `C:\Program Files (x86)\BIOVIA`
`wb_log_path`	Windows path to save Vault logs. **Default** `C:\VaultLogs`

Installer Location

Variable	Description
`wb_win_local_source_path`	Linux path to the temporary folder to download the installer on the Ansible server. **Default** `/tmp/WorkbookBuilds/`
`wb_installer_file_name`	Name of Workbook installer zip file. **Default** `BIOVIA_Workbook_2020.zip`

Temporary Installation Folder

Variable	Description
`wb_dir_path`	Path to the directory where the Workbook installer file is unzipped. **Default** `C:\BIOVIA\Workbook`
`wb_win_temp_dir`	Temporary directory for saving Workbook configuration files. **Default** `C:\tmp`

Oracle Data Access Components (ODAC) Client

Variable	Description
`workbook_oracle_client_local_source_path`	**Default** `/tmp/ODTwithODAC122011.zip`
`workbook_oracle_client_tmp_directory`	**Default** `C:\temp`

Variable	Description
`workbook_oracle_client_ oracleHome`	**Default** `C:\oracle\product\12.2.0\client`
`workbook_oracle_client_ oracleBase`	**Default** `C:\oracle`
`workbook_oracle_client_ download_version`	**Default** `12.2.0`
`workbook_oracle_client_ config_listener`	**Default** `true`
`workbook_oracle_client_tns_ name`	**Default** `youroracletnsalias`
`workbook_oracle_client_tns_ host`	**Default** `youroraclehostname.example.domain`
`workbook_oracle_client_tns_ port`	**Default** `1521`
`workbook_oracle_client_tns_ service_name`	**Default** `youroracleservicename`

Config File

Variable	Description
`wb_config_file`	Name of the Workbook config file. **Default** `workbook_config.xml`

Foundation Hub Connection

Variable	Description
`wb_hub_sslRootUrl`	Foundation hub URL - HTTPS. **Default** `https://intg-server.intg.local:9953`
`wb_hub_username`	Admin username for Foundation Hub. This must be a local user. The username will be created if it does not exist. **Default** `scitegicadmin`
`wb_hub_password`	Admin password for Foundation Hub. **Default** `passwordforscitegicadmin`

E-mail Settings

Variable	Description
`wb_smtp_server`	Workbook SMTP server name. **Default** `mail.none.com`
`wb_smtp_port`	Workbook SMTP server port. **Default** `25`
`wb_email_from`	Workbook email from address. **Default** `none@none.com`
`wb_email_to`	Workbook email to address. **Default** `none@none.com`

SSL Certificate

Variable	Description
`workbook_ssl_pfx_file_common_name`	Name of the certificate file to be used for Workbook install. **Default** `intg.local.pfx`
`workbook_pfx_file_local_path`	Path of certificate file on the Ansible server. This certificate will be copied to Workbook target server. **Default** `C:\temp`
`wb_cert_path`	Path to copy the certificate file on the Vault target nodes. **Default** `C:\temp`
`wb_cert_password`	Certificate password. **Default** `certificatepassword`

Vault Endpoint

Variable	Description
`wb_vault_endpoint`	Vault endpoint. For a load balanced environment provide the load balancer URL.

Vault Deployment

Variable	Description
`wb_vault_admin_domain`	Vault domain name.
`wb_vault_admin_user`	Vault admin username.
`wb_vault_admin_fname`	First name of the Vault admin user.
`wb_vault_admin_lname`	Last name of the Vault admin user.
`wb_vault_admin_password`	Vault admin password.
`wb_deployment_type`	Whether the deployment will be "New", "Upgrade", or "auto", which will detect an existing installation to upgrade or install a new version if not. Note that the values are case-sensitive. **Default** `auto`
`wb_upgrade_version`	Specify the Workbook version to upgrade from or "auto", which will look for the version of an existing installation. Leave this blank for a new installation. `auto`
`wb_globaladmin_group_name`	Foundation Hub group name. **Default** `Global Administrators`

Oracle Database Schema

Variable	Description
`wb_db_server`	Database server name to host Vault database. **Default** `youroraclehostname.example.domain`
`wb_db_port`	Database server port name. **Default** `1521`
`wb_db_service`	Database server service name. **Default** `youroracleservicename`
`wb_db_username`	Database admin username. **Default** `system`

Variable	Description
`wb_db_password`	Database admin password. **Default** `sys`
`wb_siteconnection_username`	Workbook Site schema username. **Default** `vaultsite`
`wb_siteconnection_password`	Workbook Site schema password. **Default** `vaultsite`
`wb_fileservice_username`	Workbook File Service schema username. **Default** `vaultfile`
`wb_fileservice_password`	Workbook File Service schema password. **Default** `vaultfile`
`wb_vaultsite_ro_username`	WorkbookVault Site readonly username. **Default** `vaultsitero`
`wb_vaultsite_ro_password`	WorkbookVault read-only password. **Default** `vaultsitero`
`wb_Workflow_username`	Workbook Workflow schema username. **Default** `vaultwftools`
`wb_Workflow_password`	Workbook Workflow schema password. **Default** `vaultwftools`
`wb_RAS_username`	Workbook RAS schema username. **Default** `symyxdb`
`wb_RAS_password`	Workbook RAS schema password. **Default** `symyxdb`
`wb_RASUser_username`	Workbook RAS schema user username. **Default**

Variable	Description
	`symyxdbuser`
`wb_RASUser_password`	Workbook RAS schema user password. **Default** `symyxdbuser`
`wb_home_repo_username`	Workbook home repository schema username. **Default** `vaultuser`
`wb_home_repo_password`	Workbook home repository schema password. **Default** `vaultuser`
`wb_home_repo_name`	Workbook home repository name. **Default** `Home Repository`
`wb_direct_user`	Workbook Direct schema name. If nothing is specified, script will look for latest Direct schema installed. **Default** `C$DIRECT2020`

Oracle Database Tablespace

Variable	Description
`wb_db_default_tablespace`	Default tablespace name. **Default** `/opt/u02/orcl/pdborcl/`
`wb_vault_tablespace`	Vault tablespace name. **Default** `Vault`
`wb_vault_temp_tablespace`	Vault temp tablespace name. **Default** `VaultTemp`
`wb_vault_idx_tablespace`	Vault IDX tablespace name. **Default** `VAULTIDX`
`wb_vault_lob_tablespace`	Vault LOB tablespace name. **Default** `VAULTLOB`

Variable	Description
`wb_file_service_tablespace`	Vault File Service tablespace name. **Default** `FileService`
`wb_ras_default_tablespace`	Workbook RAS tablespace name. **Default** `symyxdb`
`wb_ras_user_tablespace`	Workbook RAS user tablespace name. **Default** `symyxuser`
`wb_ras_index_tablespace`	Workbook RAS index tablespace name. **Default** `symyxind`
`wb_ras_lob_tablespace`	Workbook RAS LOB tablespace name. **Default** `symyxlob`
`wb_ras_tmp_tablespace`	Workbook RAS temp tablespace name. **Default** `symyxtemp`
`wb_ras_audit_tablespace`	Workbook RAS audit tablespace name. **Default** `symyxaudit`
`wb_ras_audit_index_tablespace`	Workbook RAS audit index tablespace name. **Default** `symyxauditindex`
`wb_ras_audit_lob_tablespace`	Workbook RAS audit lob tablespace name. **Default** `symyxauditlob`

Oracle Database Repository

Variable	Description
`wb_nb[1-20]_repo_username`	Vault Repository schema username. Workbook supports up to 20 schemas. **Default** `vaultn1`
`wb_nb[1-20]_repo_password`	Vault Repository schema password. Workbook supports up to 20 schemas.

Variable	Description
	Default `vaultn1`
`wb_nb[1-20]_repo_name`	Vault Repository name. Workbook supports up to 20 schemas. **Default** `General`

Workbook Templates

Variable	Description
`wb_exp_templates_repo_path`	Workbook repo path, where Experiment templates are to be published. **Default** `General\Experiment_Templates_2020`

ADM Store Manager

Variable	Description
`wb_old_profile_name`	Profile name set in the rules file before upgrade.
`wb_new_profile_name`	Profile name to be set in the rules file after upgrade.

IQ Report

Variable	Description
`wb_iq_direct_pass`	Direct schema password for IQ reporting. **Default** `directpassword`
`wb_iq_plp_url`	Pipeline Pilot URL HTTPS for IQ reporting. **Default** `https://intg-server.intg.local:9503/admin`
`wb_iq_global_admin`	Workbook Administrators group name for IQ reporting. **Default** `Global Administrators`
`wb_iq_vault_version`	Workbook version to be searched in Hub for IQ reporting. **Default** `'2020'`

Variable	Description
`wb_iq_tls_regpath`	Registry path to TLS 1.2 on the Workbook serverfor IQ reporting. **Default** `HKLM:\SYSTEM\CurrentControlSet\Control\ SecurityProviders\SCHANNEL\Protocols\`
`wb_iq_htttp_endpoint`	Workbook endpoint HTTP for IQ reporting. **Default** `http://intg-server.intg.local`
`wb_iq_htttps_endpoint`	Workbook endpoint HTTPS for IQ reporting. **Default** `https://intg-server.intg.local`
`wb_iq_apppools`	IIS App pool names for Workbookfor IQ reporting. **Default** `'["Vault", "ADM", "VaultRest", "FileService"]'`
`wb_iq_host_group`	Group name for Workbook target serversfor IQ reporting. **Default** `{{ groups['workbook-windows'] | list }}`
`wb_iq_hub_host_group`	Group name for Foundation Hub servers for IQ reporting. **Default** `{{ groups['hub-windows'] | list }}`
`wb_iq_plp_host_group`	Group name of Pipeline Pilot servers for IQ reporting. **Default** `{{ groups['plp-windows'] | list }}`
`wb_registry_path`	This is registry path to Vault install **Default** `HKLM:\SOFTWARE\Wow6432Node\Microsoft\Window s\ CurrentVersion\Uninstall\ {c22b8b97-9d2b-4643-a6b7-5e5bc62d024f}\`
`wb_iq_log_location`	Location on the Ansible server where the iq report content will be saved **Default** `./iq_reports`

CISPro Variables

Configure the following inventory file variables for installations of CISPro.

[cispro-windows]

The `[cispro-windows]` group defines the fully qualified domain names for Windows target nodes for CISPro.

Example

```
[cispro-windows]
cispro-server1.example.domain
cispro-server2.example.domain
```

[cispro-windows:vars]

The variables in the `[cispro-windows:vars]` group apply to the Windows target nodes for CISPro.

Installer Location

Variable	Description
`cispro_win_local_source_path`	Linux path to the installer file on the Ansible server.

Installation Directory

Variable	Description
`cispro_win_install_dir`	Path to the directory where CISPro is installed. **Default** `C:\BIOVIA\cispro`

License

Variable	Description
`cispro_LicenseKey`	License key for CISPro.

DMP File Configuration

Variable	Description
`cispro_data_pump_dir_name`	Name of the data pump directory. **Default** `cispro_test_dump_dir`

SSL Certificate

Variable	Description
`cispro_ssl_pfx_file_common_name`	SSL certificate common name.
`cispro_ssl_pfx_file_password`	SSL certificate password.
`cispro_ssl_pfx_file_local_path`	SSL certificate Linux path on Ansible server.

Oracle Data Access Components (ODAC) Client

Variable	Description
`cispro_oracle_client_download_version`	ODAC client version. **Note:** ODAC client is required for CISPro. **Default** `12.2.0`
`cispro_oracle_client_local_source_path`	Linux path to ODAC client installation zip file on the Ansible server.
`cispro_oracle_client_oracleBase`	Windows base directory path to ODAC client on target machine. **Default** `C:\oracle`
`cispro_oracle_client_tns_name`	ODAC client TNS name.
`cispro_oracle_client_tns_host`	ODAC client TNS host's fully qualified domain name.
`cispro_oracle_client_tns_port`	Oracle client TNS port. **Default** `1521`
`cispro_oracle_client_tns_service_name`	ODAC client TNS service name.

Oracle Database Schema

Note: These fields are not used for upgrades.

Variable	Description
`cispro_oracle_username`	Oracle Database user name (required for new installations only.)
`cispro_oracle_password`	Oracle Database password (required for new installations only.)

Foundation Hub Integration

Variable	Description
`cispro_RegisterWithHub`	Register CISPro with Foundation Hub. **Default** `false`
`cispro_RunPreSync`	Run Pre-Sync between Foundation Hub and CISPro. **Default** `false`

Variable	Description
`cispro_RunSync`	Run Sync between Foundation Hub and CISPro. **Default** `false`
`cispro_HubServerURL`	URL to the Foundation Hub server. **Default** `https://hub-server1.example.domain:9953/foundation/hub`
`cispro_HubUsername`	Admin username for Foundation Hub. **Default** `scitegicadmin`
`cispro_HubPassword`	Admin password for Foundation Hub. **Default** `passwordforscitegicadmin`
`cispro_hub_database`	Foundation Hub database schema username.
`cispro_installationrooturl`	CISPro installation root URL. **Default** `http://{{ ansible_fqdn }}`
`cispro_installationrooturlsecure`	CISPro installation root URL secure. **Default** `https://{{ ansible_fqdn }}`

Direct Integration

Variable	Description
`cispro_direct_user`	Direct database schema username. **Default** `DIRECT2021`

Creating an Inventory and Host Files for Notebook

For Notebook installations, you will need to create an inventory, inventory secret, and host file.

Creating an Inventory File for Notebook

1. From the Ansible server, navigate to inventory directory and create a `notebook` directory:
   ```
   cd inventories
   ```
   ```
   mkdir notebook
   ```
2. Create an inventory file in the `notebook` directory:
   ```
   vi inventory
   ```
3. Add a line for each host that defines the host alias, IP address, and username.
   ```
   <host alias> ansible_host=x.x.x.x ansible_user=username
   ```

Element	Description
`<host alias>`	Alias for the host node.
`ansible_host`	IP address of host node.
`ansible_user`	User name of host node used for installing notebook application.

4. Add nodes under node group.

```
[notebook_windows]
<host alias 1>
<host alias 2>
...
```

5. Define the following variables for the group:

```
[notebook_windows:vars]
ansible_connection=winrm
ansible_port=5986
ansible_winrm_server_cert_validation=ignore
ansible_winrm_transport=ntlm
notebook_build_zip_path=
notebook_build_temp_dir=
notebook_win_temp_dir=
company_logo_text=
```

Variable	Description
`ansible_connection`	Connection type to the between Ansible and the host. This can be the name of any of Ansible's connection plugins. **Example:** `winrm`
`ansible_port`	Connection port number. **Example:** `5986`
`ansible_winrm_server_cert_validation`	Server certificate validation mode (`ignore` or `validate`). Unless verifiable certificates have been configured on the WinRM listeners, set this to `ignore`. **Example:** `ignore`
`ansible_winrm_transport`	Authentication type. Separate multiple types with commas. For more information about the options, see https://docs.ansible.com/ansible/latest/user_guide/windows_winrm.html#authentication-options. **Example:** `ntlm`

Variable	Description
`notebook_build_zip_path`	Full path of the Notebook installation zip file on the Ansible server. **Example:** `/opt/tmp/ElnBuild/BIOVIA_NotebookServer_2021.zip`
`notebook_build_temp_dir`	Path of the temporary directory on the Ansible server. This directory is used by Ansible to execute temporary file operations. **Example:** `/opt/tmp/temp`
`notebook_win_temp_dir`	Path of temporary directory on the host node where the Notebook installation zip file will be copied by Ansible. Use backslashes for the path. **Example:** `c:\temp`
`company_logo_text=`	Text that appears in the PDF header of an experiment. It can be maximum of 13 ASCII characters. If not specified, "BIOVIA" is used. This value is saved in `<notebook installation>\ElnService\SystemSettings.xml`.

Creating an Inventory Secret File

Create an inventory secret file to store the passwords of the host node.

1. Navigate to the `inventories/notebook` folder you created for the inventory file:

   ```
   cd inventories/notebook
   ```

2. Create a file named `inventory_secrets`:

   ```
   vi inventory_secrets
   ```

3. Add passwords for the host nodes.

   ```
   <host node 1> ansible_password=**********
   <host node 2> ansible_password=**********
   ```

4. Navigate to the directory of the `inventory_secrets` file and execute the following command to encrypt it. You will be prompted to enter the password.

   ```
   ansible-vault encrypt inventory_secrets
   ```

   Note: The password you use to encrypt the inventory secret is used while executing the Ansible playbook.

For details about working with inventory secret files, see Ansible Vault on page 8.

Creating a Host Files for Notebook

Host files contain variables and values for the individual hosts. Create a separate file for each host you specified in the inventory file.

1. From the Ansible server, navigate to the `inventories/notebook` directory and create a `host_vars` directory:

   ```
   cd inventories/notebook
   ```

   ```
   mkdir host_vars
   ```

2. Create a host file using the form `<host alias>.yml`where `<host alias>` is the name of the host specified in the inventory file.
3. Add the following lines to indicate the beginning and ending of the `.yml` file.

   ```
   ---
   ```

   ```
   ...
   ```

   Add the variables and values between these lines.
4. Add host variables:

Variable	Description

`notebook_execute_oracle_scripts`	Executes Oracle scripts on the node.
	Example:
	`### Database execution`
	`notebook_execute_oracle_scripts: true`

`notebook_CreateOracleDatabase_DBA_appSettings`	Updates SYSDBA credentials in the `web.config` file of `CreateOracleDatabase.exe`. The Directory key indicate the directory path of tablespace.sql. To execute `tablespace.sql` for Notebook and DataArchive, provide two paths separated by a semicolon (;).
	Example:
	`### Add the SYSDBA Credentials used to run tablespace.sql`
	`notebook_CreateOracleDatabase_DBA_appSettings:`
	`- { key: "DATABASESERVERNAME", value: "hostservername:1521/ORCL" }`
	`- { key: "DATABASEUSERNAME", value: ""}`
	`- { key: "DATABASEPASSWORD", value: ""}`
	`- { key: "Directory", value: "{{ notebook_win_temp_dir }}\\notebookDeployment\\Database_Oracle;{{ notebook_win_temp_dir }}\\notebookDeployment\\Databases\\Oracle_DataArchive"}`
	`- { key: "SysDbaConnection",value: "True"}`

Variable	Description

`notebook_CreateOracleDatabase_appSettings`	Updates database user credentials in `web.config` file of `CreateOracleDatabase.exe` to execute Oracle scripts for notebook application.
	Example:
	`### Add database user credentials used to run the Oracle database scripts`
	`notebook_CreateOracleDatabase_appSettings:`
	`- { key: "DATABASESERVERNAME", value: " hostservername:1521/ORCL" }`
	`- { key: "DATABASEUSERNAME", value: "eln"}`
	`- { key: "DATABASEPASSWORD", value: ""}`
	`- { key: "Directory", value: "{{ notebook_win_temp_dir }}\\notebookDeployment\\Database_Oracle"}`
	`- { key: "SysDbaConnection",value: "false"}`

`notebook_CreateOracleDatabase_dataArchive_appSettings`	Updates database credentials in `web.config` file of `CreateOracleDatabase.exe` to execute oracle scripts for the DataArchive application.
	Example:
	`### Add database user credentials used to run the Data Archive Oracle database scripts`
	`notebook_CreateOracleDatabase_dataArchive_appSettings`
	`- { key: "DATABASESERVERNAME", value: "hostservername:1521/ORCL" }`
	`- { key: "DATABASEUSERNAME", value: "DATAARCHIVE"}`
	`- { key: "DATABASEPASSWORD", value: ""}`
	`- { key: "Directory", value: "{{ notebook_win_temp_dir }}\\notebookDeployment\\Databases\\Oracle_DataArchive"}`
	`- { key: "SysDbaConnection",value: "false"}`
	`- { key: "DataPackageFolder",value: "c:\\Biovia-DA\\Packages"}`

`notebook_ria_appSettings`	Updates the `appSettings` of RIA `web.config` file.
	Example:
	`### Add RIA app settings`
	`notebook_ria_appSettings:`
	`- { key: "DATABASESERVERNAME", value: " hostservername:1521/ORCL" }`
	`- { key: "DATABASEUSERNAME", value: "elnruntime"}`
	`- { key: "DATABASEPASSWORD", value: ""}`
	`- { key: "DefaultSchema", value: "ELN"}`
	`- { key: "DATABASETYPE",value: "ORACLE"}`

`notebook_da_appSettings`	Updates the `appSettings` of DataArchive `web.config` file. **Example:** `### Add Data Archive app settings`

Variable	Description

	`notebook_da_appSettings:`
	`- { key: "DATABASESERVERNAME", value: " hostservername :1521/ORCL" }`
	`- { key: "DATABASEUSERNAME", value: "dataarchiveruntime"}`
	`- { key: "DATABASEPASSWORD", value: ""}`
	`- { key: "DefaultSchema", value: "ELN"}`
	`- { key: "DATABASETYPE",value: "ORACLE"}`
	`- { key: "NotebookUsername",value: "superuser"}`
	`- { key: "NotebookPassword",value: ""}`

`notebookLoginService_appSettings`	(optional) Updates the `appsettings` in Notebook login service.
	Example:
	`### Add Notebook login service settings`
	`notebookLoginService_appSettings:`
	`- { key: "AcceptedReturnUrls", value: ""}`
	`- { key: "DefaultDomain", value: ""}`
	`- { key: "FoundationHost", value: ""}`

5. Repeat steps 2 and 3 for the remaining hosts.

Chapter 4: Running the Playbook

BIOVIA provides Ansible playbooks that you can use to manage your deployment of BIOVIA applications.

IMPORTANT! The machine used to initiate connection and supply login credentials to the Ansible server MUST maintain network connection throughout the time that a playbook executes. If this remote connection is interrupted, the playbook execution will stop. For example, if a laptop launches a PuTTY window and is used to execute a playbook, closing the laptop prematurely will likely terminate the Ansible script.

Installation Order

BIOVIA applications must be installed in the following order:

1. Foundation Hub
2. Pipeline Pilot
3. FoundationSSO (if applicable)
4. Remaining applications

Playbooks

The following playbooks are available for installing BIOVIA applications:

Playbook	Description
`install-hub-basic.yml`	Installs or upgrades a single instance of Foundation Hub.
`install-hub-loadbalanced.yml`	Installs or upgrades the Foundation Hub nodes one at a time. All targeted nodes should belong to the same load-balanced cluster (point to the same load balancer). Do not include nodes from multiple load balancers in the same inventory file.
`install-plp-basic.yml`	Installs or upgrades a single instance of Pipeline Pilot. **Note:** The `plp_req_install_location` variable is used for both Windows and Linux installations. Be sure to change the path as appropriate.
`install-compose-basic.yml`	Installs or upgrades a single instance of Compose and Capture.
`install-compose-loadbalanced.yml`	Installs or upgrades the Compose and Capture nodes one at a time. All targeted nodes should belong to the same load-balanced cluster (point to the same load balancer). Do not include nodes from multiple load balancers in the same inventory file.

Playbook	Description
`install-workbook-basic.yml`	Installs or upgrades a single instance of Workbook. This playbook does not generate an IQ report for Workbook. Run `workbook-LB-iq-report.yml` separately.
`workbook-LB-iq-report.yml`	Generates an IQ report for Workbook.
`install-cispro-basic.yml`	Installs or upgrades a single instance of CISPro.
`cispro-iq-report.yml`	Generates an IQ report for a CISPro installation. **Note:** Ansible CISPro scripts do not support creating a new Oracle user.
`install_notebook.yml`	Installs Notebook. Upgrading is not supported.

Utility Playbooks

Playbook	Description
`ping-win.yml`	Pings the Windows nodes to test for connectivity. See Setting up Target Nodes on page 5.
`ping-linux.yml`	Pings the Linux nodes to test for connectivity. See Setting up Target Nodes on page 5.
`provision.yml`	Locally installs and deploys an Apache server for file hosting. See Setting up an Apache Server for Pipeline Pilot on page 6.

Playbook Commands

The command for running a playbook is:

```
ansible-playbook -i <path> <playbook.yml>
```

where <path> can point to a folder that contains the inventory file, or to the specific inventory file itself, and <playbook>`.yml` is the name of the playbook you want to run.

Note: You will be prompted for the target node administrator credentials when you run the playbook.

Example

```
ansible-playbook -i inventories/EXAMPLE_ENVIRONMENT/inventory install-hub-
basic.yml
```

Using Foundation Hub Playbook Actions

Additional actions are available when you run one of the Foundation Hub playbooks. See Foundation Hub Playbook Actions on page 42.

Running a Playbook with Ansible Vault

If you are using an Ansible Vault file to encrypt some of your variables, you must point to the folder that contains the inventory file and the encrypted file when running the command, not to a specific inventory file. The command must include an `--ask-vault-pass` flag.

```
ansible-playbook -i <path> <playbook.yml> --ask-vault-pass
```

Example

```
ansible-playbook -i inventories/EXAMPLE_ENVIRONMENT install-hub-basic.yml --
ask-vault-pass
```

For more information about using this feature, see Ansible Vault on page 8.

Note: You will be prompted for your Ansible Vault password.

Pipeline Pilot Reporting Server label for Compose and Capture

The Pipeline Pilot Reporting Server label used by Compose and Capture is not set when using Ansible to install a new version of Compose and Capture, and will be reported as an error in the IQ report. You must set the Pipeline Pilot Reporting Server label manually as a post-install configuration step. See the *Compose and Capture Installation Guide* for details about configuring this setting.

Foundation Hub Playbook Actions

Foundation Hub supports specific actions when running the playbook such as stopping or starting services or forcing an upgrade. To use an action, add a `-e hub_ansible_action={{ Action }}` flag to your playbook command.

Example

```
ansible-playbook -i inventories/EXAMPLE_ENVIRONMENT install-hub-basic.yml  -
e hub_ansible_action=Upgrade --ask-vault-pass
```

Supported Ansible Actions for Foundation Hub

Action	Description
`Auto`	Automatically determines whether to perform an upgrade or a new Install based on the presence of a previously installed version. This option is run by default when you run one of the Foundation Hub playbooks without a flag.
`Install`	Performs a clean install of Foundation Hub. See Foundation Hub Variables on page 10 for the required variables. **Results** ■ A new version of Foundation Hub is installed on the target nodes. ■ An IQ report is generated.

Action	Description
`Upgrade`	Performs an upgrade of an existing installation of Foundation Hub. See Foundation Hub Variables on page 10 for the required variables. **Results** ■ Foundation is upgraded on the target nodes. ■ Backups of the original `app-config.groovy` and `tomcat.properties` configuration files are created: ■ For the Ansible host machine, the backup files are generated in the folder set by the `hub_backup_archive_location` variable (default is `<ansible scripts directory>\hub_backup_configurations`). ■ For Windows targets, the backup files are generated in the folder set by the `hub_win_backup_location` variable (default is `C:\temp\hub_config_temp`). ■ For Linux targets, the backup files are generated in the directory set by the hub_linux_backup_location variable (default is `/tmp/hub_config_temp`). ■ An IQ report is generated.
`Uninstall`	Uninstalls Foundation Hub from the target nodes.

Action	Description
`Reconfigure`	Overwrites the original Foundation Hub `app-config.groovy` and `tomcat.properties` configuration files. **Results** ■ The configuration files are overwritten using values defined in the Foundation Hub Application Configuration and Tomcat Properties sections of the inventory file and default values from the Foundation Hub playbook. None of the previous values are saved. See Foundation Hub Variables on page 10. ■ The Foundation Hub service is restarted. ■ Backups of the previous configuration files are created: ■ For the Ansible host machine, the backup files are generated in the folder set by the `hub_backup_archive_location` variable (default is `<ansible scripts directory>\hub_backup_configurations`). ■ For Windows targets, the backup files are generated in the folder set by the `hub_win_backup_location` variable (default is `C:\temp\hub_config_temp`). ■ For Linux targets, the backup files are generated in the directory set by the hub_linux_backup_location variable (default is `/tmp/hub_config_temp`). ■ An IQ report is generated
`Stop`, `Start`, and `Restart`	Stops, starts, or restarts Foundation Hub services.
`Status`	Returns the status of Foundation Hub services.
`Precheck`	Checks that the nodes meet the system requirements for Foundation Hub.
`Iq`	Generates an IQ report on the Ansible host machine.

Installing FoundationSSO

The FoundationSSO package is a set of Pipeline Pilot components and protocols required by Foundation Hub. This package is included in some versions of Pipeline Pilot and does not need to be installed separately. See the *Foundation Hub Product Release Document* to find out if you need to install this package. You can use the same inventory file variables that you set for Pipeline Pilot, changing the `plp_req_installer_zip_loc` variable to point to the FoundationSSO installer package.

Post Installation Steps

If you have installed the FoundationSSO package, manually refresh the Pipeline Pilot Applications page.

1. Sign in to the Pipeline Pilot Admin Portal.
2. Open **Reports > Foundation Applications**.

3. Click **Update Application**.

Reporting

The Ansible scripts for BIOVIA applications supports the generation of Installation Qualification (IQ) report content. You can use this content in official IQ reports for your company.

To use the IQ report generation features, set the appropriate inventory file variables for your application and run the playbook for your application. Note that the Workbook playbook does not generate IQ report content. However, you can run a separate playbook to generate IQ report content for Workbook.

Tip: For Foundation Hub you can run the playbook with a flag to generate IQ report content without performing an install. See Foundation Hub Playbook Actions on page 42.

Application	Variables	Playbooks
Foundation Hub	`hub_iq_log`	`install-hub-basic.yml` `install-hub-loadbalanced.yml`
Pipeline Pilot	`plp_req_generate_iq_report`	`install-plp-basic.yml`
Compose and Capture	`compose_iq_log`	`install-compose-basic.yml` `install-compose-loadbalanced.yml`
Workbook	`wb_iq_*`	`workbook-LB-iq-report.yml`

Note: The playbook for Compose and Capture will not set the Pipeline Pilot Reporting Server label, which will be reported as an error in the IQ report. You must manually set the label as a post-installation step. For details, see Pipeline Pilot Reporting Server label for Compose and Capture on page 42.

Checking the IQ Reports

After running the playbook, check the pre- and post-IQ reports in the `BIOVIADeployment` extracted directory: `/iq_reports/<node name>/<application name>-iq-<year>-<month>-<day>-<hour>-<minute>/<application name>-iq-report.txt`. The reports are separated by node and application and are timestamped.