Day Zero Configuration

Day Zero Configuration

The initial or day 0 configuration of a VNF is based on the VM type. A VNF administrator configures the initial template for each VM type at the time of VNF deployment. The same configuration template is applied to all deployed and new VMs of that VM type. The template is processed at the time of individual VM deployment. The day 0 configuration continues to persists, so that all initial deployment, healing and scaling of VMs have the same day 0 template.

Some of the day 0 configuration tasks include bringing up the interface, managing the network, support for static or dynamic IP (DHCP, IPAM), SSH keys, and NetConf enabled configuration support on VNF.

Note ESC does not support day 0 configuration of interfaces added during service update. In case of recovery for day 0 configuration, all the interfaces with Network Interface Card IDs will be configured.

Day Zero in the Configuration Data Model

The day 0 configuration file can be specified in different ways in the data model, but you can use only one of the options at a time.

- <file> url </file>—The url specifies a file on the ESC VM file system or file hosted on report http server. ESC downloads the file specified by the URL. This file is used as a template to replace the tokens specified in this template with the values specified in the variables section. This template is used to generate the day 0 configuration.
- <data> inline config content </data>—Specifies URL for the template. This allows the use of inline text as the template.
- <encrypted_data> inline config content</encrypted_data>—The inline configuration content will be encrypted based on the data.

- <file_locators> list of file locators </file_locators>—Similar to file, a file_locator defines file to download from a remote server with basic authentication (if required).

Note The <file_locators> is deprecated in ESC Release 4.0.

- <file_locator_name> deployment defined file_locator </file_locator_name>—Similar to file, the file_locator_name is used to download the file from a remote server with basic authentication (if required).

Day 0 configuration is defined in the datamodel under the config_data tag. Each user data and the configuration drive file is defined under the configuration tag. The contents are in the form of a template. ESC processes the template through the Apache Velocity Template Engine before passing to the VM.

The config_data tag is defined for each vm_group. The same configuration template is applied to all VMs in the vm_group. The template file is retrieved and stored at deployment initialization. Template processing is applied at time of VM deployment. The content of the config file can be retrieved from the file or data.

```
<file> url </file>
<data> inline config content </data>
```

A destination name is assigned to the config by <dst>. User Data is a treated as a special case with <dst>--user-data</dst>.

A sample config data model,

```
<config_data>
    <configuration>
        <file>file://cisco/userdata_file.txt</file>
        <dst>--user-data</dst>
        <variable>
            <name>CUSTOM_VARIABLE_FOR_USERDATA</name>
            <val>SOME_VALUE_XXX</val>
        </variable>
    </configuration>
    <configuration>
        <file>file://cisco/config.sh</file>
        <dst>config.sh</dst>
        <variable>
            <name>CUSTOM_VARIABLE_FOR_CONFIG</name>
            <val>SOME_VALUE_XXX</val>
        </variable>
    </configuration>
</config_data>
```

Custom variable can be specified in the variables tag within the configuration. Zero or more variables can be included in each configuration. Each variable can have multiple values. Multiple values are only useful when creating more than one VM per vm_group. Also, when performing scale in and scale out, additional VMs can be added and removed from the VM group.

Note Note the following while providing multiple values for the variable tag.

- The variable values assigned to the initially deployed VMs are unique and from the pool. There is no order followed for assigning the values from the pool. That is, the first VM can use the second value from the pool.
- A scaled out VM should have a unique variable value and from the pool.

- A recovered VM (after undeploy or redeploy) must retain the same value it had before.

The contents of <file> are a template that is processed by the Velocity Template Engine. ESC populates a set of variables for each interface before processing the configuration template:

NICID_n_a_IP_ALLOCATION_TYPE	string containing FIXED | DHCP
NICID_n_a_NETWORK_ID	string containing neutron network uuid
NICID_n_a_IP_ADDRESS	ipv4 or ipv6 address
NICID_n_a_MAC_ADDRESS	string
NICID_n_a_GATEWAY	ipv4 or ipv6 gateway address
NICID_n_a_CIDR_ADDRESS	ipv4 or ipv6 cidr prefix address
NICID_n_a_CIDR_PREFIX	integer with prefix-length
NICID_n_a_NETMASK	If an ipv4 CIDR address and prefix are present, ESC will automatically calculate and populate the netmask variable. This is not substituted in the case of an IPv6 address and should not be used.
NICID_n_a_ANYCAST_ADDRESS	string with ipv4 or ipv6
NICID_n_a_IPV4_OCTETS	string with last 2 octets of ip address, such as 16.66, specific to CloudVPN

Where n is the interface number from the data model, for example, 0, 1, 2, 3

Note The interface number, n starts with 0 for OpenStack, and 1 for VMware.

Example

```
NICID_0_NETWORK_ID=0affdc19-60fd-4a4f-a02b-f062d7a66c27
NICID_0_MAC_ADDRESS=fa:16:3e:4d:c5:f8

NICID_0_0_IP_ALLOCATION_TYPE=DHCP
NICID_0_0_IP_ADDRESS=1.1.22.133
NICID_0_0_GATEWAY=1.1.0.1
NICID_0_0_CIDR_ADDRESS=1.1.0.0
NICID_0_0_CIDR_PREFIX=16
NICID_0_0_NETMASK=255.255.0.0

NICID_0_1_IP_ALLOCATION_TYPE=DHCP
NICID_0_1_IP_ADDRESS=fd04:1::a03
NICID_0_1_GATEWAY=fd04:1::1
NICID_0_1_CIDR_ADDRESS=fd04:1::/64
NICID_0_1_CIDR_PREFIX=64
```

By default, ESC substitutes the $ variable in the day 0 configuration file with the actual value during deployment. You can enable or disable the $ variable substitution for each configuration file.

Add the following field to the configuration data model:

```
<template_engine>VELOCITY | NONE</template_engine> field to configuration
```

where,

- VELOCITY enables variable substitution.
- NONE disables variable substitution.

If no value is set the default option is VELOCITY, and the $ variable substitution takes place. When set to NONE, the $ variable substitution does not take place.

You must follow these tips while processing the template through the velocity template engine.

- To escape dollar sign in the template insert,

  ```
  #set ( $DS = "$" )
  ```

  then replace the variable with

  ```
  passwd: ${DS}1${DS}h1VxC40U${DS}uf2qLUwGTjHgZplkP78xA
  ```

- To escape a block in the template, insert #[[and #]]. For example,

  ```
  #[[ passwd: $1$h1VxC40U$uf2qLUwGTjHgZplkP78xA ]]#
  ```

File Locator

To fetch external configuration files, a file locator is added to the day 0 configuration. The file locator contains a reference to the file server, and the relative path to the file to be downloaded.

Note The file locator attribute is defined at the deployment level, that is, directly under the deployment container instead of policy actions and day 0 configuration sections. For updated data model see Fetching Files From Remote Server.

Example of day 0 configuration with a file locator:

```
<?xml version="1.0" encoding="UTF-8"?>
<esc_datamodel xmlns="http://www.cisco.com/esc/esc">
  <tenants>
    <tenant>
      <name>sample-tenant</name>
      <deployments>
        <deployment>
          <name>sample-deployment</name>
          <vm_group>
            <name>sample-vm-group</name>
            <config_data>
              <!-- exisiting configuration example - remains valid -->
              <configuration>
                <file>file:///cisco/config.sh</file>
                <dst>config.sh</dst>
              </configuration>
              <!-- new configuration including use of file locators -->
              <configuration>
                <dst>ASA_config_0</dst>
                <file_locators>
                  <file_locator>
                    <name>configlocator-1</name>
                    <!-- unique name -->
                    <remote_file>
```

```
                      <file_server_id>server-1</file_server_id>
                      <remote_path>/share/users/configureScript.sh</remote_path>
                      <!-- optional user specified local silo directory -->
                      <local_target>day0/configureScript.sh</local_target>
                      <!-- persistence is an optional parameter -->
                      <persistence>FETCH_ALWAYS</persistence>
                      <!-- properties in the file_locator are only used for
                           fetching the file not for running scripts -->
                      <properties>
                        <property>
                          <!-- the property name "configuration_file" with value "true"
indictates this is the
                               script to be used just as using the <file> member case of
the configuration -->
                          <name>configuration_file</name>
                          <value>true</value>
                        </property>
                        <property>
                          <name>server_timeout</name>
                          <value>120</value>
                          <!-- timeout value in seconds, overrides the file_server property
 -->
                        </property>
                      </properties>
                    </remote_file>
                    <!-- checksum is an optional parameter.
                         The following algorithms are supported: SHA-1, SHA-224, SHA-256,
SHA-384, SHA-512 -->
                      <checksum>SHA256 (configureScript.sh) =
dd526bb2c0711238ec2649c4b91598fb9a6cf1d2cb8559c337c5f3dd5ea1769e</checksum>
                  </file_locator>
                  <file_locator>
                    <name>configlocator-2</name>
                    <remote_file>
                      <file_server_id>server-2</file_server_id>
                      <remote_path>/secure/requiredData.txt</remote_path>
                      <local_target>day0/requiredData.txt</local_target>
                      <persistence>FETCH_ALWAYS</persistence>
                      <properties />
                    </remote_file>
                  </file_locator>
                </file_locators>
              </configuration>
            </config_data>
          </vm_group>
        </deployment>
      </deployments>
    </tenant>
  </tenants>
</esc_datamodel>
```

The file locator parameters include:

- name—used as the key and identifier for a file locator.
- local_file or remote_file—choice of file location. Local file is used to specify a file existing on the ESC VM file system already. The remote_file is used to specify a file to fetch from a remote server.
  - file_server_id—id of the File Server object to fetch the file from.
  - remote_path—path of the file from the base_url defined in the file server object.
  - local_target—optional local relative directory to save the file.

- properties—name-value pairs of of information that may be required.
- persistence—options for file storage. Values include CACHE, FETCH_ALWAYS and FETCH_MISSING (default).

- checksum—optional BSD style checksum value to use to validate the transferred file's validity.

For more information, see Fetching Files From Remote Server.

To encrypt the files see, Encrypting Configuration Data.

Day 0 Configuration for vCD Deployment

The day 0 configuration for vCD deployment can be passed in different ways:

- Constructing an ISO file
- OVF properties
- Pre-existing ISO file in a catalog (OOB ISO file)

Note

- For initial deployment, the number of VM group(s) defined in the datamodel must be the same as the number of VM(s) in the vApp template. In a deployment, the image value of each VM group should be unique.
- The out of band (OOB) ISO file cannot be used along with constructing an ISO file method, as the VM can consider any one. The ovf property can be used with OOB ISO or constructing ISO together.

Day 0 configuration through constructing an ISO file:

```
</rules>
            <config_data>
              <!-- take content from the file path and save it as config.sh into the ISO
file -->
              <configuration>
                <dst>config.sh</dst>
                <file>file:///cisco/config.sh</file>
              </configuration>
             <!-- take content from the file path, replace variables with values, and save
 it as data/config.sh into the ISO file -->
              <configuration>
                <dst>data/params.cfg</dst>
                <file>file:///cisco/template.cfg</file>
                <variable>
                  <name>CF_VIP_ADDR</name>
                  <val>10.0.0.9</val>
                </variable>
                <variable>
                  <name>CF_DOMAIN_NAME</name>
                  <val>cisco.com</val>
                </variable>
                <variable>
                  <name>CF_NAME_SERVER</name>
                  <val>172.16.180.7</val>
```

```
              </variable>
             </configuration>
             <!-- take the data section as the content of the file, replace variables with
 values, and save it as user-data.txt into the ISO file-->
             <configuration>
               <dst>user-data.txt</dst>
               <data>#cloud-config
manage_etc_hosts: true
hostname: $HOST_NAME
local-hostname: $HOST_NAME
</data>
               <variable>
                 <name>$HOST_NAME</name>
                 <val>something.cisco.com</val>
               </variable>
             </configuration>
           </config_data>
```

Day 0 configuration through OOB ISO file:

```
         </rules>
            <config_data>
              <configuration>
                <!-- ISO file stored in catalog-1 -->
                <dst>vcdCatalog:catalog-1</dst>
                <data>h2.iso</data>
              </configuration>
            </config_data>
```

Day 0 configuration through OVF properties:

```
           <config_data>
              <configuration>
                <!-- ovf properties as day0 -->
                <dst>ovfProperty:mgmt-ipv4-addr</dst>
                <data>$NICID_0_IP_ADDRESS/24</data>
              </configuration>
            </config_data>
```

For information on deploying VNFs on vCD, see Deploying Virtual Network Functions on VMware vCloud Director (vCD).