VMware Smart Assurance UI Platform User and Configuration Guide

VMware Smart Assurance 10.0.0

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Preface

As part of an effort to improve and enhance the performance and capabilities of its product lines, VMware periodically releases revisions of its hardware and software. Therefore, some functions described in this document may not be supported by all versions of the software or hardware currently in use. For the most up-to-date information on product features, refer to your product release notes. If a product does not function properly or does not function as described in this document, please contact your VMware representative.

The purpose of this guide is to provide information about how to use and configure Smart Assurance UI Platform components .

Intended Audience

This guide is intended to be read by any user who needs to use and configure Smart Assurance UI Platform components.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to http://www.vmware.com/support/pubs.

Overview

1

The VMware Smart Assurance UI Platform imports event information from Smarts products through DCF into the Event Store. UI pulls the notification data through API gateway.

Event Store

2

Event Store is a processing system and which reads Notifications published by SAM, process them and persist into backend Elastic Search Database. Event Store is a historical DB of Notifications and Audit Info for anomaly detections and commonality analysis. Process & Persist Chaining with Causes and Caused By Notifications.

This chapter includes the following topics:

Main components of Event Store

Event store uses different components to read the Notifications and process them.

Main components of Event Store are:

- Notification Processor: Following are the main tasks of Notification Processor:
  - Maintains the Causes and Causedby chaining for all the Notifications.
  - Processing for maintaining the History of Notifications.
  - Processing Notification Audits.
  - Archiving Notifications
- Web server: Gin is a high-performance micro-framework that can be used to build web applications and micro services. It makes it simple to build a request handling pipeline from modular, reusable pieces.
- Elastic Search: Elastic search is a modern search and analytics engine which is based on Apache Lucene, provides rich rest API.
- Plug-Ins: Following are the main tasks of Plug-Ins:
  - Get Chaining Causes and Caused By for a Notification.

    - Transform Response Structure.
    - Elastic Search Authentication.
    - Console access is controlled by the user profile that is associated with a user account.
- Kafka: Apache Kafka is publish-subscribe based fault tolerant messaging system. It is fast, scalable and distributed by design.
- Redis: Redis is an open source, advanced key-value store and an apt solution for building high-performance, scalable web applications.
- Rest API: http://<ipaddress>:<port>/eps/swagger/index.html.

How authentication works

In Event Store, there is only component to component authentication.

Authentications are:

- Kafka Authentication
- Elastic Search Authentication
- Redis Authentication

Refer *VMware Smart Assurance UI Installation and Configuration Guide* for authentication detail.

Path for Event Store Log

User can find the Event store log at the following location:

'<Path_to_EventStore_Installation>/logs/'

Event Store configuration

To configure Event Store, user needs to update the event properties and Elastic Search database files.

events.properties

In events.properties file update following:

```
<Path_to_EventStore_Installation>/config/eventstore.properties'

    esUrl= "<ip>:<port>,<ip>:<port>,<ip>:<port>"
    esUsername = "Test"
    esPassword = "ENCRYPTED_PASSWORD"
    esProto = "http" # Elastic search is http/https protocol
     esCertLoc = "", # Elastic search cluster is https enabled then add the certification to the given
location
```

Elastic Search

In Elastic Search database file update the following:

```
'<ip>:<port>/configuration/config/eps/'

    httpPort = "9090" # Event Store http port
    # Elastic Search DB configurations for storing Notifications and Audits
    esUrl = "<ip>:<port>,<ip>:<port>,<ip>:<port>" # Elastic search cluster URL
    esUsername = "Test" # Elastic Search DB user name
    esPassword = "ENCRYPTED_PASSWORD" # Elastic Search DB Password
    # Kafka Configurations for reading Notifications from DCF
    kafkaUrl = "<ip>:<port>,<ip>:<port>,<ip>:<port>" # Kafka URL
    kafkaTopic = "SAMNotifications" # Kafka Topic
    kafkaUser = "kafkaadmin"        # Kafka User Name
    kafkaPassword = "ENCRYPTED_PASSWORD" # Kafka Password
    # Redis used as Cache
    redisCluster = "<ip>:<port>,<ip>:<port>,<ip>:<port>" # Redis Cluster URL
    redisPassword = "ENCRYPTED_PASSWORD" # Redis Password
    # Logger Configuration
    logLocation = "<Path_to_EventStore_Installation>/logs/notifications.log"       # EventStore Log
Location
    logLevel = "ERROR"
    # Archival Notifications for removing from Elastic Search and storing in a file
    purging-scheduler-TimeInterval = "24h" # Scheduler Time Interval
    purgingTimeLimit = "now-1y-0M-0d" # How many days to retain notifications in ES DB
    esdb-repo = "" # Elastic search Repo to Archive Notifications
    #SAM Details for ACK,UNACK,Change Ownership, Release Ownership
    sam-details = "<ip>:<port>" # SAM URL
    # User Session Configurations
    userTokenTTL = "10800"
    userGracePeriod = "300"
    # Https configuratons
    esProto = "http" # Elastic search is http/https protocol
    esCertLoc = "", # Elastic search cluster is https enabled then add the certification to the given
location
    samProto = "http", # SAM is http/https protocol
    samCertLoc = ""  # SAM is https enabled then add the certification to the given location
```

Archiving Notifications

If user configure scheduler to run every month and expiry period is one year then, it creates a file with copy of all the expired notification histories (1 year old) and delete them from the Elastic Search database every month.

To configure scheduler for archiving Notification:

Procedure

1. Create an NFS Shared folder (in any of the 3 ES clusters or any other server), and run below command in all machines:

```
yum install nfs-utils nfs-utils-lib
    create the folder to be shared across cluster in any machine
                Ex: mkdir /var/unixmen_share/
    sudo chown -R elasticsearch:elasticsearch /var/unixmen_share
    edit /etc/exports in the vm from where above folder will be shared
         /var/unixmen_share/     <Cluster1 IP>(rw,sync,no_root_squash,no_all_squash)
        /var/unixmen_share/      <Cluster 2(rw,sync,no_root_squash,no_all_squash)
    service nfs start
    Go to all cluster machines and run below command
        mount -t nfs <nfs_server>:/var/unixmen_share/ /var/unixmen_share/
```

2. All Elastic Search cluster nodes must have access to the shared folder to write data into folder.
3. Verify group id and userid are same for elastic search across clusters, by invoking the below command

```
less /etc/passwd | grep elastic search
```

4. Add repo in the eventstore.properties configuration file:

```
Stop cluster
    add below in all cluster nodes  /etc/elasticsearch/config/elasticsearch.yml
    path.repo: ["/var/unixmen_share/"]
    start the cluster
```

5. Create a repo in cluster:

```
PUT http://<<Any  ES Cluster IP>>:9200/_snapshot/<<repo_name>>
    {
        "type": "fs",
        "settings": {
            "location": "/var/unixmen_share/backup_location",
            "compress": true
        }
    }
```

6. Configure repo, time of expiry and scheduler time period in:

```
// Valid time units are "ns", "ms", "s", "m", "h".

POST http://<<EPS_Server>>:<<EPS port>>:9090/eps/refresh
{
  "doc": {
    "purging-scheduler-TimeInterval": "24h",
    "purgingTimeLimit": "now-1y-0M-0d",
```

```
    "esdb-repo": "smarts-repo"
  }
}
```

7 Restart Eventstore.

Note If esdb-repo is not configured, then it will delete one year old notifications from esdb no backup will be taken.

8 Check Snapshots are getting created using below API:

`GET http://<<Any ES Cluster IP>>:9200/_cat/snapshots/ <<repo_name>>`

9 To delete snapshots from repo, run below REST API:

`DELETE http://<<Any_ES_Cluster_IP>>:`
`9200/_snapshot/<<repo_name>>/<<snapshot_name>>`

cAPI

3

cAPI is a thin layer which consolidates the data from Topology, Performance Metrics and Notification Store.

cAPI:

- Works as a API Gateway redirects the request to corresponding component.
- Is a Load balancer and provides response transformations.
- Is a layer in which Cross-cutting functionality such as authentication, monitoring, and traffic management can be implemented. So that the micro services can remain unaware of these details.

This chapter includes the following topics:

Main components of cAPI

Main components of cAPI are:

- Envoy: Envoy is an open source edge and service proxy, designed for cloud-native applications. Envoy's configuration consists primarily of listeners and clusters.
- Listeners: A listener tells Envoy a TCP port on which it must listen, and a set of filters with which Envoy must process what it hears.
- Clusters: A cluster tells Envoy about one or more backend hosts to which Envoy can proxy incoming requests.

cAPI logs

cAPI provides *admin_access* and *envoy* logs

Path for admin_access.log:

```
access_log_path: <cAPI installation path>/logs/admin_access.log
```

Path for envoy.log:

```
envoy.file_access_log: <cAPI installation path>/logs/envoy.log
```

To change loglevel of console log in envoy, use the following URL:

```
http://<IP>:<port>/logging?level=trace
```

cAPI configuration

This section describes how to configure cAPI in *envoy.yaml* file.

Path for envoy.yaml:

`envoy.yaml: : <cAPI installation path>/config/envoy.yaml`

To change log file location for admin access :

admin- access_log_path:

```
/tmp/admin_access.log # log file location for admin access
```

To change listener port:

```
static_resources:
  listeners:
  - name: listener_0
    address:
      socket_address: { address: 0.0.0.0, port_value: <LISTENER_0_PORT> } # listener port
```

To change listener log file path:

```
static_resources:
  listeners:
  - name: listener_0
    filter_chains:
    - filters:
      - name: envoy.http_connection_manager
        config:
          access_log:
            name: envoy.file_access_log
            config:
              path: /tmp/envoy.log # listener log file
```

To change Elastic Search Username/Password:

```
"esdb_cluster",
                        {
                         ["Authorization"] = "<ES_Authorization_Header>" // add Base 64
encoded                                 username:password
                        },
```

To change Elastic Search database, URL needs to be changed:

```
clusters:
 – name: esdb_cluster
    hosts: [{ socket_address: { address: <ESDB_IP>, port_value: <ESDB_PORT> }}]
```

To change EPS URL:

```
clusters:
 – name: eps_cluster
   hosts: [{ socket_address: { address: <EPS_IP>, port_value: <EPS_PORT> }}]
```

To change VIDM IP or PORT:

```
clusters:
  – name: jwks_cluster
    hosts: [{ socket_address: { address: <VIDM_IP>, port_value: <VIDM_PORT> }}]
```

To change VIDM Host Name, change the host name below:

1 `listeners: – filter_chains: – filters: – config: – route_config : virtual_hosts: routes: route: { host_rewrite: "<VIDM_HOST_NAME>", prefix_rewrite: "/SAAS/auth/oauthtoken",cluster: jwks_cluster }`

2
```
listeners:  – filter_chains:    – filters: – http_filters: – config:
              providers:
                eventstore_auth:
                  issuer: https://<VIDM_HOST_NAME>/SAAS/auth
                  remote_jwks:
                    http_uri:
                      uri: https://<VIDM_HOST_NAME>/SAAS/API/1.0/REST/auth/token?
attribute=publicKey&format=jwks
```

3
```
clusters: – name: tls_context: { sni: www.<VIDM_HOST_NAME> }
```

To add new route:

```
static_resources:
  listeners:
  – name: listener_0
   filter_chains:
    – filters:
    route_config:
     routes:
 – match: { prefix: "/esdb/audit" }
               route:
                 prefix_rewrite: "/audit"
                 cluster: esdb_cluster
                 retry_policy:
                   retry_on: 5xx
                   num_retries: 3
```

```
      per_try_timeout: 0.300s
    request_headers_to_add:
      - header:
          key: "Authorization"
          value: "<ES_Authorization_Header>"
        append: true
```

High Availability and Fault Tolerance for Smarts

4

For providing High Availability(HA) for VMs running Smart Assurance components, vSphere HA is used which creates a cluster for virtual machines by pooling the virtual machines and the hosts they reside on into a cluster. Hosts in the cluster are monitored and in the event of a failure, the virtual machines (running Smart Assurance components) on a failed host are restarted on alternate hosts. For more details on vSphere HA, refer *vSphere 6.7 Availability* documents..

VMware Smart Assurance platform components uses vSphere Fault Tolerance for its virtual machines FT to ensure continuity with higher levels of availability and data protection. vSphere FT ensures availability by having identical virtual machines run on separate hosts. To obtain the optimal results from Fault Tolerance you must be familiar with how it works, how to enable it for your cluster, virtual machines and the best practices for its usage. For more details refer *vSphere 6.7 FT* document.

This chapter includes the following topics:

Prerequisite for HA

Fulfill the following prerequisites before starting the installation:

1. Compatible versions for vSphere, vCenter must be installed and setup for the cluster i.e. vSphere 6.7 and vCenter 6.7.
2. All ESX hosts must have access to same network so that when VM starts on other host it should access to same network.
3. Each host in the HA cluster must be able to do DNS resolution of any other host in the cluster.
4. Minimum 2 ESX hosts are required for setting up HA. However, to ensure redundancy and maximum Fault Tolerance protection, you should have a minimum of three hosts in the cluster.
5. VM files (except for the VMDK files) and other configuration files must be stored on shared storage so that it should be available from any esx hosts. Acceptable shared storage solutions include Fibre Channel, (hardware and software) iSCSI, vSAN, NFS, and NAS.

6 High bandwidth ethernet link between ESX hosts.

Prerequisite for Fault Tolerance

Fulfill the following prerequisites before starting the installation:

1 HA cluster must be setup and enabled before enabling FT.

2 Configure the networking properly i.e. vmkernel adapter must be configured for FT logging and Vmotion.

3 There is no CD/DVD enabled for the VM.

4 Hardware virtualization in BIOS must be enabled for each of the host.

5 All the hosts in the cluster must have supported processors and license for enabling FT.

Setup HA for Smart Assurance Environment

Procedure

1 Install compatible versions of vSphere/vCenter.

2 Create a HA cluster for the hosts and enable HA.

   a From vCenter Console, select **Create a cluster**.

   b For vSphere HA, check out **turn on**.

   Note Also check out , **monitoring** and **admission control**.

   c Click **ok**.

3 Start VMs on the hosts as per the deployment plan and install Smart Assurance components. For more information refer to deployment plan and Smart Assurance UI Installation guide.

4 Ensure whether HA setup is done properly. Follow the steps to verify it:

   a Select **Cluster**.

   b In right pane, select **Monitor** tab.

   c Select **vSphere HA** tab.

   d In the left pane, select **Summary**.

   Verify that Master IP, No. of hosts connected to Master, and Number of protected VMs are correct.

5 Select the VMs for FT.

   Note FT is very resource intensive, it is recommended to enable it only for the mission critical VMs. Otherwise it will have performance impact.

For example in Smart Assurance deployment we have recommended it for main VM running ESM/SAM,DCF controller, UI, and EventStore as we want to ensure zero downtime for this particular VM. Otherwise it may lead to loss of service.

6 Enable the FT for the VMs:

a Right click on the **VM** and click on **Enable Fault tolerance**.

b On the pop up window, select **data store/host** for the secondary host.

c Click **Finish** to complete the FT.

7 Verify whether FT is enabled for the VM: Check whether Primary and secondary VMs are running.

a In the vCenter console, select the FT VM.

In the right pane, choose **Summary** Tab.

On below page, look for widget **Fault Tolerance**

1 Fault Tolerance Status must be **Protected**.

2 Secondary VM location must be another host in cluster.

b By logging in to vSphere console of each host, FT VM must be visible on both Primary/secondary Host.

Known Configuration Issues for HA/FT

Warning for management n/w redundancy

Host currently has no management network redundancy. This message is displayed if the network redundancy configuration within theService Console/VMkernel Port Management Network is incorrect. To prevent this message from appearing, and to comply with proper network redundancy. For resolving the warnings regarding management network redundancy, we need to configure management network redundancy for each host in the cluster.

Workaround : On each host ensure management/vMotion is enable for the vmkernel adapters. Follow below steps:

1 Go to each ESX servers.

2 Click **Network**.

3 Edit *vmkernel adapters*.

4 Edit *management network*.

5 Ensure management(as well vmotion) are checked out.

6 Click **ok** to save the changes.

Warning for number of heartbeat datastores

The number of heartbeat datastores for host is 1, which is less than required: 2

Workaround : This issue occurs if there is no redundancy in shared storage to allow for datastore heartbeating. vSphere HA requires a minimum of two shared datastores shared between all hosts in a cluster for proper datastore heartbeat detection to function.

For resolving this issue, Go to vSphere HA advanced options and set *das.ignoreInsufficientHbDatastore* to *true*. For detailed information, follow the kb article: https://kb.vmware.com/s/article/2004739

Note For troubleshooting other issues refer *vSphere Guides*.

Monitoring Smart Assurance Platform

5

VMware Smart Assurance platform can monitor configured application processes on a host and provide OS level metrics for the host. Typical OS level metrics include metrics related to cpu (ProcessorUtilization), memory(MemoryUtilization), disk, file system etc. These metrics values are available in form of attribute values. In certain scenarios, it can also generate events etc the process is missing(MissingProcess Event), CPU/Memory utilization threshold breached. For further details on os/application/application service group attributes, refer *Chapter 8 of VMware Smarts User and Configuration Guide*.

It also provides the capability to monitor a group of application processes that refers to a service. It can monitor an application group and displays the events under ApplicationServiceGroup. In the topology view, an application instance represents a process and an ApplicationServiceGroup instance represents a group of application processes that provides a service. A user-defined application service group instance can represent a group of processes.

Steps to setup for monitoring any given host:

1 Enabled the Netsnmp on the host: Netsnmp should be enabled for the all the hosts to be monitored. For enabling netsnmp, proper access should be given by updating snmpd.conf file i.e. adding rocommunity public to */etc/snmp/snmpd.conf*.

2 Discover the host to be monitored using VMware Smart Assurance.

Note For monitoring VMware Smarts Assurance platform components, out of the box prebuild template i.e. `apps-emc-smarts.xml` can be used.

Data Collection Controller

6

Data Collection Controller (DCC) is a REST-based orchestrator that takes requests from users to deploy and manage data collection blocks. Data Collection Blocks (usually called 'block') collect data from specified devices, transform, filter, and publish them via multiple interfaces. Currently, a block defines a pipeline that may contain one or more components that will collect, process, and publish data.

This chapter includes the following topics:

Main components of DCC

Main components of DCC are:

- Web server: Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. It's a pre-fork worker model. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resources, and fairly speedy.
- MongoDB: MongoDB is a cross-platform document-oriented database program. It is a dependency of the DCC. It is classified as a NoSQL database program, MongoDB uses JSON-like documents with schemata. MongoDB is developed by MongoDB Inc.
- Data Collector Framework core: DCF core technology software delivers enterprise and carrier-class cross-domain performance and service level management which transforms data into actionable information, helping assure the delivery of business services.

  DCC interacts with DCF core via its Web Service Gateway, a SOAP-based API that provides a programmable interface to DC Core features.
- DCC endpoints: DCC has two main endpoints:
  - Catalog: Catalog is responsible for block provisioning, configuration, available blocks, and packages.
  - Runtime: Runtime is responsible for operations with blocks, such as start, stop, restart, manage, and un-manage configuration of variable handling filter (VHF).

How authentication works

DCC is configured to be the only entry point of DCF. Once you install it, it allows you to provision and orchestrate any available block in DCF.

Most of the DCC configuration is done through properties in `dc_controller/server/config.py`.

This is typically where an administrator would change the protocol (http/https), the listening address, the listening port, secret key parameters, and a few other advanced settings.

DCC communication is encrypted using HTTPS. The client issues a request using Basic authentication through a HTTPS channel. Therefore, a certificate needs to be generated during the installation process.

DCC logs

DCC logs are present at directory `<Path_to_DCF_Installation>/Tools/Controller/Default/logs/`

The log file under above directory structure : `apg-controller-default.out` - contains controller startup related errors.

Another log file under above directory structure : `dcf.log` - contains controller functionality errors.

DCC configuration

The configuration file `dc_controller/server/config.py` contains all the parameters of DCC.

It contains information about:

- DCC internal credentials, secret key, and path to client certificate.
- MongoDB location and credentials, DC Core location and credentials, maximum number of clients interacting with web service gateway, DCC location, and type of deployment (i.e. Bare metal or virtualized).

This file will be generated as result of the 'create first user' script execution:

```
MONGODB_URI='127.0.0.1:27017'
MONGODB_USERNAME='admin'
MONGODB_PASSWORD='ENCRYPTED_PASSWORD'
MONGODB_DATABASE='admin'
DEPLOYMENT_TYPE='BAREMETAL'
MNR_HOST='127.0.0.1'
MNR_PORT='48443'
MNR_USER='admin'
MNR_PASSWORD='ENCRYPTED_PASSWORD'
MNR_MAX_THREADS=2
SECRET_KEY='SECRET_KEY'
CERTIFICATE_PATH='/opt/DCF/Tools/Controller/Default'
DCF_HOST='0.0.0.0'
DCF_PORT='8443'
USERNAME='admin'
PASSWORD='ENCRYPTED_PASSWORD'
```

```
LOG_LEVEL='INFO' # 'DEBUG', 'ERROR', 'CRITICAL', 'WARNING'

# The following properties are passed to the DCC web server (Gunicorn)
bind='0.0.0.0:8443' # Gunicorn bind socket
workers='4' # Number of async workers
worker_class='gthread' # Gunicorn mode
certfile='/opt/DCF/Tools/Controller/Default/conf/controller-cert.pem'
keyfile='/opt/DCF/Tools/Controller/Default/conf/controller-key.pem'
preload_app='True'

# The following is a Gunicorn server hook, called just after the server is started.
def when_ready(server):
    import dc_controller.wsgi as dcfControllerApp
    dcfControllerApp.create_default_template()
```

The 'config.py' file can be regenerated using below command :

```
python3.6 <DCF_Installation_path>/Tools/Controller/Default/dc_controller/utils/create-first-user.py
<DEPLOYMENT_TYPE> <MongoDB_USERNAME> <MongoDB_PASSWORD> <MONGODB_URI> <MNR_HOST> <MNR_PORT>
<MNR_USERNAME> <MNR_PASSWORD> <MNR_MAX_THREADS> <OUTPUT_CONFIG_FILE> <CERTIFICATE_PATH>
<KEY_FILE_PATH> <DCF_HOST> <DCF_PORT> <DCF_HOST:DCF_PORT> <DCF_USERNAME> <DCF_PASSWORD>
```

VMware Smarts Metrics collector

7

The Smarts metric collector Solution Pack is a DCF Management Pack used to pull all the metrics and topology data from Smarts domain manager and publish the same to KAFKA for consumption. It is fully compatible with versions 6.x, 7.x and 8.x versions of the SMARTS domain managers.

This chapter includes the following topics:

Installation

Smarts metrics collector Solution Pack (smarts-metrics-collect) installation is similar to any other Solution Pack installation and can be done through DCF module manager and also through controller.

Related package is available in DCF module repository by default. For example: `<DCF-Install>/Tools/Module-Repository/smarts-metrics-collect-1.0.pkg`

Installation command: `<DCF-Install>/bin/manage-modules.sh install smarts-metrics-collect -1.0 smarts-metrics-collect`.

Sample Installation questions and output:

```
Required dependencies, in processing order:
   [1]   java '8.0.202' v8.0.202
   [2]   module-manager '1.13u1' v1.13u1
   [3] I collector-manager 'smarts-metrics-collect' (none) => v5.10u1
   [4] I kafka-connector 'smarts-metrics-collect' (none) => v1.0u2
   [5]   jdbc-drivers 'Default' v2.8
   [6] I sm-collector 'smarts-metrics-collect' (none) => v5.9u2
   [7] I smarts-metrics-collect 'smarts-metrics-collect' (none) => v1.0
> 3 not modified, 4 to install
> 26.6 MB space required / 97.7 GB available
 ? Enter the step to modify, 'yes' to accept them, or 'no' to cancel the operation [yes] > yes

Starting installation of Collector-Manager v5.10u1 from collector-manager-5.10u1-linux-x64...
 * Gathering information...
 * 'Collector-Manager v5.10u1' will be registered with instance name 'smarts-metrics-collect'.
 * It will be installed in '/opt/DCF/Collecting/Collector-Manager/smarts-metrics-collect'.
 * Unpacking files...
```

```
 * Installing files... 100%
 * 60 files have been installed.
 * Finalizing installation...
 * Installing service 'collector-manager smarts-metrics-collec...  [ installed ]
Installation complete.

Starting installation of Kafka-Connector v1.0u2 from kafka-connector-1.0u2-linux-x64...
 * Gathering information...
 * 'Kafka-Connector v1.0u2' will be registered with instance name 'smarts-metrics-collect'.
 * It will be installed in '/opt/DCF/Collecting/Kafka-Connector/smarts-metrics-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 17 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Smarts-Collector v5.9u2 from sm-collector-5.9u2-linux-x64...
 * Gathering information...
 * 'Smarts-Collector v5.9u2' will be registered with instance name 'smarts-metrics-collect'.
 * It will be installed in '/opt/DCF/Collecting/Smarts-Collector/smarts-metrics-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 67 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of smarts-metrics-collect v1.0 from smarts-metrics-collect-1.0...
 * Gathering information...
 * 'smarts-metrics-collect v1.0' will be registered with instance name 'smarts-metrics-collect'.
 * It will be installed in '/opt/DCF/Block/smarts-metrics-collect/smarts-metrics-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 8 files have been installed.
 * Finalizing installation...
 ? Smarts Domain Manager hostname or IP address. [localhost] > localhost
 ? Smarts Domain Manager Direct Port Number. [12345] >
 ? Smarts Domain Manager to subscribe. [INCHARGE-AM-PM] >
 ? Username [admin] >
 ? Password [•••••] >
   [1] AM/PM Topology & Metrics
   [2] AM/PM Metrics
   [3] AM/PM Topology
 ? Type [1] >
 ? Polling interval for Topology Data in seconds. [3600] >
 ? Polling interval for Metrics Data in seconds. [240] >
 ? Backend data group [group] >
 ? Thread PoolSize. [3] >
 ? Connection PoolSize. [3] >
 ? More entries? (yes/no) [n] >
 ? Kafka Node Hostname or IP address [localhost] >
 ? Kafka Node Port Number [9092] >
 ? More entries? (yes/no) [n] >
 ? Kafka Topic name to publish Collected Data. [default-topic] >
   [1] Kafka server authentication disabled.
   [2] SASL_PLAINTEXT.
```

```
    [3] SASL_SSL.
    [4] SSL.

 If we select default option 2:-

 ? Kafka server authentication type. [2] > SASL_PLAINTEXT
 ? Kafka server username. [admin] >
 ? Kafka server password. [•••••] >
 ? Do you want to start the installed services now? (yes/no) [n] > n

If we select option 4 [SSL] :-

? Kafka server authentication type. [2] > 4
 ? Location for Kafka Producer/Consumer TrustStore (Java TrustStore).   [../../../Tools/Webservice-
Gateway/Default/conf/truststore] >
 ? Password for Kafka Producer/Consumer TrustStore (Java TrustStore). [•••••] >
 ? Location for Kafka Producer/Consumer KeyStore (Java KeyStore). [../../../Tools/Webservice-
Gateway/Default/conf/clientkeystore] >
 ? Password for Kafka Producer/Consumer KeyStore (Java KeyStore). [•••••] >
 ? Password for KeyStore Key. [•••••] >


? Do you want to start the installed services now? (yes/no) [n] > yes
```

Configuration

Solution Pack related configuration files are located in the directories:

`<DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(smarts-metrics-collect)/conf/`

`<DCF-Install>/ Collecting /Smarts-Collector/<SP-Instance>/conf/`

`<DCF-Install>/ Collecting /Kafka-Connector/<SP-Instance>/conf/`

1 **Collector Manager Configuration**

A Collecting Manager Configuration file can be found at `File: <DCF-Install>/Collecting/Collector-Manager /<INSTANCE>(smarts-metrics-collect)/conf/collecting.xml`.

Sample contents:

```
<config xmlns="http://www.watch4net.com/APG/Collecting"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.watch4net.com/APG/Collecting collecting.xsd ">
        <connectors>
        <connector enabled="true" name="File" type="File-Connector"   config="conf/file-
connector.xml" />
        <connector enabled="true" name="Kafka" type="Kafka-Connector"   config="Kafka-
Connector/smarts-metrics-collect/conf/kafka-connector.xml" />
        </connectors>
```

```
        <collectors>
                <collector enabled="true" name="smarts-INCHARGE-AM-PM-0-metrics" next="Kafka File"
type="Smarts-Collector" config="Smarts-Collector/smarts-metrics-collect/conf/smarts-INCHARGE-AM-
PM-0-metrics.xml" />
                <collector enabled="true" name="smarts-INCHARGE-AM-PM-0-topo" next="Kafka File"
type="Smarts-Collector" config="Smarts-Collector/smarts-metrics-collect/conf/smarts-INCHARGE-AM-
PM-0-topo.xml" />
        </collectors>
</config>
```

This is basic configuration of the Solution Pack which deals with how the processing starts.

2 **SMARTS metrics Configuration**

The following example is a default SMARTS metrics configuration: This file will have all Smarts domain manager configuration for polling.

`File: <DCF-Install>/Collecting/Smarts-Collector//<INSTANCE>(smarts-metrics-collect)/conf/smarts-<domain>-metrics.xml`

```
<config>
        <domain>INCHARGE-AM-PM</domain>
        <host>localhost:12345</host>
        <username>admin</username>

<password>{CFCBACB47C451D42C421C1A2441B636C51D8E0A5FBF2BB9832FD5851C175179184D7D63229D2F5E04A09DB9D
876AFC5E}</password>
        <group>group</group>
        <period>240</period>
        <thread pool-size="3" />
        < dm-connection pool-size ="3" keep-connection="false" creation-grace-time="100" />
        <indicators select="both">conf/pm-metrics.xml</indicators>
        <properties refresh="00:00/86400" send-on-refresh-only="false" />
        <smooth-factor>0.5</smooth-factor>
</config>
```

Supported configuration parameters are:

Parameters	Description
domain	The smarts collector retrieves data from any domain to add historical capabilities to the suite. It is highly flexible and customizable to fit any situation.
host	The host:port to connect to domain.
username	The username to connect to the domain.
password	The password to connect to the domain.
group	The collector will assign the string specified in the group element to generated raw values. This group should match one of your processing component configuration.

Parameters		Description
period		You can choose your polling period in seconds (period element) according to your domain configuration and your needs.
thread pool-size		The thread pool-size attribute let you specify the number of concurrent polling threads which will share a customizable number of connections to smarts domain (specified in dm-connection pool-size).
dm-connection pool-size		The dm-connection pool-size attribute let you specify a customizable number of connections to smarts domain.
dm-connection creation-grace-time		The dm-connection creation-grace-time is the minimum delay between consecutive connection to the smarts domain.
indicators		Indicators let you select the domain descriptor file which defines what is interesting in a smarts domain.
Properties refresh		The refresh attribute controls how often the collector should set the refresh tag on the generated values. It can be either a time with a period (00:00/86400 will refresh properties each day at midnight) or a number of polling loop (360 will refresh properties every 360 polling loops). In both cases, properties will always be refresh on first polling loop, right after the collector startup.
smooth-factor		The smooth-factor control the actual polling rate. For example, if polling period is 240 second and smooth factor is 0.5, the Collector will try to pool the domain in 0.5 * 240 = 120 seconds.
source		A source represents a SMARTS domain instance to monitor. Each source can be composed of a primary source and multiple failover sources, if needed. This is useful for specifying backup SMARTS Domains in case the primary is down.
	primary/failover	Determines if the source is the primary source or just a failover source.
	broker-hostname	The broker hostname of the SMARTS Domain if connecting to the broker as your entry point.
	broker-port	The port on which the SMARTS Domain's broker is accepting connections.
	broker-username (optional)	The username to use to establish the connection with the broker. This password can be in the encrypted form, generated by the crypt-password script. If no authentication is required, omit this element.

Parameters		Description
	broker-password (optional)	The password to use to establish the connection with the broker. This password can be in the encrypted form, generated by the crypt-password script. If no authentication is required, omit this element.
	hostname	The hostname of the SMARTS Domain if you need to directly connect to the manager.
	port	The port on which the SMARTS Domain is accepting connections.
	domain-name	The name of the domain from which metrics & topo data will be fetched.
	username	The username used to connect to the domain manager.
	password	The password used to connect to the domain manager. This password can be in the encrypted form, generated by the crypt-password script.

3 **SMARTS topology Configuration**:

The following example is a default SMARTS topology configuration: This file will have all Smarts domain manager configuration for polling.

`File: <DCF-Install>/Collecting/Smarts-Collector//<INSTANCE>(smarts-metrics-collect)/conf/smarts-<domain>-topo.xml`

```
<config>
        <domain>INCHARGE-AM-PM</domain>
        <host>localhost:12345</host>
        <username>admin</username>

<password>{CFCBACB47C451D42C421C1A2441B636C51D8E0A5FBF2BB9832FD5851C175179184D7D63229D2F5E04A09DB9D
876AFC5E}</password>
        <group>group</group>
        <period>3600</period>
        <thread pool-size="3" />
        <dm-connection pool-size="3" keep-connection="false" creation-grace-time="100" />
        <indicators select="both">conf/pm-topo.xml</indicators>
        <properties refresh="00:00/86400" send-on-refresh-only="false" />
        <smooth-factor>0.5</smooth-factor>
</config>
```

4 **Kafka Connector Configuration**

Kafka server configuration used to publish events from Smarts to Kafka is below:

File : <DCF-Install>/Collecting/Kafka-Connector/<INSTANCE>(smarts-metrics-collect)/conf/kafka-connector.xml)

```
<kafka-connector-config xmlns="http://www.watch4net.com/KafkaConnector"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.watch4net.com/KafkaConnector ../kafka-connector.xsd ">
        <kafka-producer-settings>
            <!-- Servers information -->
                <server host="localhost" port="9092"/>
            <!-- Producer (and topic) information -->
            <producer topic-name="default-topic"
                acks="1"
                retries="0"
                linger-ms="0ms"
                buffer-memory="33554432"
                compression-type="none"
                batch-size="16384"
                max-block-ms="5s"
                max-in-flight-requests-per-connection="5"
                max-request-size="1048576"/>

            <!-- All connection parameters will be attributes except "server" -->
            <connection request-timeout-ms="5s"
                connections-max-idle-ms="9m"
                retry-backoff-ms="100ms"
                reconnect-backoff-ms="50ms"/>

            <!-- Additional properties -->
            <additional-parameters key="metadata.max.age.ms">1000</additional-parameters>
            <additional-parameters key="receive.buffer.bytes">32768</additional-parameters>
            <additional-parameters key="send.buffer.bytes">131072</additional-parameters>

        <additional-parameters key="security.protocol">SSL</additional-parameters>
            <additional-parameters key="ssl.truststore.location">../../../Tools/Webservice-
Gateway/Default/conf/truststore</additional-parameters>
        <additional-parameters
key="ssl.truststore.password">{613FF4F84B7A36EC8D22728760D70A56FF2CB8E9CCCD90367BFEEB74E5B97EAB1B76
5AA92F50F91101E757D131BD5A4D}</additional-parameters>
        <additional-parameters key="ssl.keystore.location">../../../Tools/Webservice-
Gateway/Default/conf/clientkeystore</additional-parameters>
        <additional-parameters
key="ssl.keystore.password">{9F9B74AF79C19897075C1CFEC22D542F018838BEFE763E4DF180850D474320D2EC9595
E36F68D90A1A830175BF42D8A7}</additional-parameters>
        <additional-parameters
key="ssl.key.password">{FE076F391C3C5946DF2EDF230CAF272D601423289B8BFF9C29F246D089C153D01E2E5A0B1A1
F714C814DDF3088F3982F}</additional-parameters>

        </kafka-producer-settings>

        <connector-component-behavior outputJson="true" flush-every-n-rawvalues="5000"/>

</kafka-connector-config>
```

Kafka Connector parameters

Parameters	Description
server	This tag must occur at least once. host: The address of one of the kafka bootstrap servers port: The port that the bootstrap server is listening on.
producer	This tag is optional, but may be used for templating kafka producers. Please refer to the schema for more information.
connection	This tag is optional, but may be used for templating kafka connections. Please refer to the schema for more information.
additional-parameters	This tag is optional, but may be used for configuring kafka options outside of the purview of the previous tags. These options include, but are not limited to SSL connection parameters.

Configuring Kafka Connector:

Parameters	Description
kafka-producer-settings	This tag may be used if we want to customize how the writer writes to kafka. Please refer to the schema for more information.
connector-component-behavior	This tag may be used if we want to control how often we flush to kafka.

Logging

All Event processing logs can be found at below path. This have event spy logging information also, if enabled.

`<DCF-Install>/Collecting/Collector-Manager/<INSTANCE>(ex:smarts-metrics-collect)/logs/collecting-0-0.log`

Note Latest processing log name will have "-0-0" appended to its name like above.

To enable logging, change the collecting configuration to push metrics/topology data to file connector like below. And after changing the configuration, operator need to restart the service to see the logging info in `collecting-0-0.log`.

Edit below file as specified:

```
File : <DCF-Install>/Collecting/Collector-Manager/<INSTANCE>(smarts-metrics-collect)/conf/collecting.xml
---
<connector enabled="true" name="File" type="File-Connector"   config="conf/file-connector.xml" />

---
```

For example : "smarts-metrics-collect" is installed management pack name in the below example:

To check the status of the installed services:

```
<DCF-Install>/bin/manage-modules.sh service status all
….
* Checking collector-manager smarts-metrics-collect'...       [ running ]

To restart the particular service:

<DCF-Install>/bin/manage-modules.sh service restart collector-manager smarts-metrics-collect
 * Stopping collector-manager smarts-metrics-collect'...            [ OK ]
 * Starting collector-manager smarts-metrics-collect'...            [ OK ]
```

Sample output (JSON) of metrics data in Kafka :

```
{
   "groupName":"group",
   "discoveryID":null,
   "jobID":"9999",
   "type":"Processor",
   "timestamp":1554098437,
   "value":0.0,
   "action":"r",
   "properties":{
      "ismanaged":"true",
      "source":"INCHARGE-AM-PM",
      "type":"Processor",
      "datagrp":"SMARTS-PM-METRICS"
   },
   "metrics":{
      "CurrentUtilization":{
         "properties":{
            "name":"CurrentUtilization",
            "unit":"%"
         },
         "value":10.0
      }
   },
   "relations":[

   ],
   "forceRefresh":true,
   "initialized":true,
   "name":"PSR-w1-hs4-i2108.eng.vmware.com/30"
}
```

Sample output (JSON) of topo data in Kafka :

```
{
   "groupName":"group",
   "discoveryID":null,
   "jobID":"9999",
   "type":"Processor",
   "timestamp":1554098437,
```

```
   "value":0.0,
   "action":"r",
   "properties":{
      "ismanaged":"true",
      "source":"INCHARGE-AM-PM",
      "type":"Processor",
      "datagrp":"SMARTS-PM-TOPO"
   },
   "metrics":{
      "id":{
         "properties":{
            "name":"id",
            "unit":"integer"
         },
         "value":30.0
      }
   },
   "relations":[
      {
         "type":"ProcessorGroup",
         "element":"PSRGROUP-w1-hs4-i2108.eng.vmware.com/0",
         "relationName":"PartOf"
      },
      {
         "type":"Host",
         "element":"w1-hs4-i2108.eng.vmware.com",
         "relationName":"PartOf"
      }
   ],
   "forceRefresh":true,
   "initialized":true,
   "name":"PSR-w1-hs4-i2108.eng.vmware.com/30"
}
```


Sample debugging (File Connector) output from collecting-0-0.log:

```
INFO     -- [2019-04-01 02:59:58 EDT] -- CollectorManagerImpl::configure(): Parsing file
'/opt/DCF/Collecting/Collector-Manager/Default/conf/collecting.xml' ...
INFO     -- [2019-04-01 02:59:59 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.FileConnector for connector File...
INFO     -- [2019-04-01 02:59:59 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
connector File (FileConnector v1.0 rM) with /opt/DCF/Collecting/Collector-Manager/Default/conf/file-
connector.xml...
INFO     -- [2019-04-01 02:59:59 EDT] -- CollectorManagerImpl::configureConnectors(): Skipping
disabled connector Kafka.
INFO     -- [2019-04-01 02:59:59 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.SmCollector for collector smarts-INCHARGE-AM-PM-0-
metrics...
INFO     -- [2019-04-01 02:59:59 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
collector smarts-INCHARGE-AM-PM-0-metrics (SmCollector v5.9u2 rM) with /opt/DCF/Collecting/Smarts-
Collector/Default/conf/smarts-INCHARGE-AM-PM-0-metrics.xml...
INFO     -- [2019-04-01 02:59:59 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.SmCollector for collector smarts-INCHARGE-AM-PM-0-topo...
```

VMware Smarts Notification collector

8

The SMARTS Event SP is a DCF Management Pack used to pull all the events from Smarts domain manager and publish the same to KAFKA for consumption. It is fully compatible with version 10.0 of the SMARTS domain managers.

This chapter includes the following topics:

- Installation
- Configuration
- Logging

Installation

Smarts events SP (smarts-notifs-events) installation is similar to any other SP installation and can be done through DCF module manager and also through controller.

Related package are available in DCF module repository by default.

For example: `<DCF-Install>/Tools/Module-Repository/smarts-notifs-events-1.0.pkg`

Installation command :

`<DCF-Install>/bin/manage-modules.sh install smarts-notifs-events`

Sample Installation output :

```
Required dependencies, in processing order:
   [1]   java '8.0.202' v8.0.202
   [2]   module-manager '1.13u1' v1.13u1
   [3] I event-processing-manager ‘smarts-notifs-events' (none) => v1.8u2
   [4] I event-processing-utils ‘smarts-notifs-events' (none) => v1.4u1
   [5] I smarts-listener ‘smarts-notifs-events' (none) => v1.7u2
   [6] I kafka-event-adapter ‘smarts-notifs-events' (none) => v1.0u1
   [7] I smarts-notifs-events ‘smarts-notifs-events' (none) => v1.0
> 2 not modified, 5 to install
> 15.1 MB space required / 26.4 GB available
 ? Enter the step to modify, 'yes' to accept them, or 'no' to cancel the operation [yes] > yes

Starting installation of Event-Processing-Manager v1.8u2 from event-processing-manager-1.8u2-linux-
x64...
 * Gathering information...
 * 'Event-Processing-Manager v1.8u2' will be registered with instance name ‘smarts-notifs-events'.
```

```
 * It will be installed in '/opt/DCF/Event-Processing/Event-Processing-Manager/smarts-notifs-events'.
 * Unpacking files...
 * Installing files... 100%
 * 23 files have been installed.
 * Finalizing installation...
 * Installing service 'event-processing-manager smarts-notifs-...  [ installed ]
Installation complete.

Starting installation of Event-Processing-Utils v1.4u1 from event-processing-utils-1.4u1-linux-x64...
 * Gathering information...
 * 'Event-Processing-Utils v1.4u1' will be registered with instance name ‘smarts-notifs-events'.
 * It will be installed in '/opt/DCF/Event-Processing/Event-Processing-Utils/smarts-notifs-events'.
 * Unpacking files...
 * Installing files... 100%
 * 15 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Smarts-Listener v1.7u2 from smarts-listener-1.7u2-linux-x64...
 * Gathering information...
 * 'Smarts-Listener v1.7u2' will be registered with instance name ‘smarts-notifs-events'.
 * It will be installed in '/opt/DCF/Event-Processing/Smarts-Listener/smarts-notifs-events'.
 * Unpacking files...
 * Installing files... 100%
 * 24 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Kafka-Event-Adapter v1.0u1 from kafka-event-adapter-1.0u1-linux-x64...
 * Gathering information...
 * 'Kafka-Event-Adapter v1.0u1' will be registered with instance name ‘smarts-notifs-events'.
 * It will be installed in '/opt/DCF/Event-Processing/Kafka-Event-Adapter/smarts-notifs-events'.
 * Unpacking files...
 * Installing files... 100%
 * 30 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of smarts-notifs-events v1.0 from smarts-notifs-events-1.0...
 * Gathering information...
 * 'smarts-notifs-events v1.0' will be registered with instance name ‘smarts-notifs-events'.
 * It will be installed in '/opt/DCF/Block/smarts-notifs-events/smarts-notifs-events'.
 * Unpacking files...
 * Installing files... 100%
 * 4 files have been installed.
 * Finalizing installation...
   [1] Smarts broker
   [2] Direct connection
 ? Collection mode [1] >  1
 ? Broker hostname or IP address [localhost] > localhost
 ? Broker port [426] > 426
 ? Are you using broker authentication (yes/no) [n] > n
   Smarts domain to subscribe [INCHARGE-SA-PRES] > INCHARGE-SA-PRES-1
 ? SAM domain username [admin] >
 ? SAM domain password [•••••] >
```

```
 ? Notification list [ALL_NOTIFICATIONS] >
 ? Configure secondary (failover) (yes/no) [n] >
 ? Configure secondary (failover) (yes/no) [n] > yes
   [1] Smarts broker
   [2] Direct connection
 ? Collection mode [1] > 2
 ? SAM Server hostname or IP address. [localhost] >
 ? SAM Server Port Number. [12345] >
 ? Smarts domain to subscribe [INCHARGE-SA-PRES] > INCHARGE-SAM2
 ? SAM domain username [admin] >
 ? SAM domain password [•••••] >
 ? Notification list [ALL_NOTIFICATIONS] >
 ? Kafka Node Hostname or IP address [localhost] > 10.31.119.1
 ? Kafka Node Port Number [9092] >
 ? More entries? (yes/no) [n] > yes
 ? Kafka Node Hostname or IP address [localhost] > 10.31.119.2
 ? Kafka Node Port Number [9092] >
 ? More entries? (yes/no) [n] > yes
 ? Kafka Node Hostname or IP address [localhost] > 10.31.119.3
 ? Kafka Node Port Number [9092] >
 ? More entries? (yes/no) [n] >
 ? Kafka Topic name to publish Collected Data. [default-topic] > eventsTopic
   [1] Kafka server authentication disabled.
   [2] SASL_PLAINTEXT.
   [3] SASL_SSL.
   [4] SSL.

If we select default option 2:-

 ? Kafka server authentication type. [2] > SASL_PLAINTEXT
 ? Kafka server username. [admin] >
 ? Kafka server password. [•••••] >
 ? Do you want to start the installed services now? (yes/no) [n] > yes

If we select option 3 [SASL_SSL]

? Kafka server authentication type. [2] > 3
? Kafka server username. [admin] >
? Kafka server password. [•••••] >
Location for Kafka Producer/Consumer TrustStore (Java TrustStore). [../../../Tools/Webservice-
Gateway/Default/conf/truststore] > (Accept the default)
? Password for Kafka Producer/Consumer TrustStore (Java TrustStore). [â¢â¢â¢â¢â¢] > changeit
? Do you want to start the installed services now? (yes/no) [n] > n

Import all the Kafka certificates into the truststore and start the installed service
<DCF Install Dir>/bin/manage-modules.sh service start event-processing-manager smarts-notifsevents

If we select option 4 [SSL] :-

? Kafka server authentication type. [2] > 4
 ? Location for Kafka Producer/Consumer TrustStore (Java TrustStore).   [../../../Tools/Webservice-
Gateway/Default/conf/truststore] >
 ? Password for Kafka Producer/Consumer TrustStore (Java TrustStore). [•••••] >
 ? Location for Kafka Producer/Consumer KeyStore (Java KeyStore). [../../../Tools/Webservice-
Gateway/Default/conf/clientkeystore] >
```

```
? Password for Kafka Producer/Consumer KeyStore (Java KeyStore). [•••••] >
? Password for KeyStore Key. [•••••] >



? Do you want to start the installed services now? (yes/no) [n] > yes
```

Note Smarts-UI supports only SASL_PLAINTEXT and SASL_SSL

Configuration

Solution Pack related configuration files are located at following directories:

<DCF-Install>/Event-Processing/Event-Processing-Manager/<SP-Instance>(smarts-notifs-events)/conf/

<DCF-Install>/Event-Processing/Smarts-Listener/<SP-Instance>/conf/

<DCF-Install>/Event-Processing/Kafka-Event-Adapter/<SP-Instance>/conf/

1 **Event Processing Manager Configuration**:

A Event Processing Manager Configuration file can be found at:

File: <DCF-Install>/Event-Processing/Event-Processing-Manager/<INSTANCE>(smarts-notifs-events)/conf/processing.xml

Sample contents

```
<processing-manager xmlns="http://www.watch4net.com/Events/DefaultProcessingManager"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.watch4net.com/Events/DefaultProcessingManager
DefaultProcessingManager.xsd ">
        <processing-element name="KAFKA" enabled="true" type="Kafka-Event-Adapter" config="Kafka-
Event-Adapter/smarts-notifs-events/conf/kafka-event-adapter.xml"/>
        <processing-element name="Smarts" enabled="true" config="Smarts-Listener/smarts-notifs-
events/conf/smarts-listener.xml" data="KAFKA"/>
        <processing-element name="EVENT-SPY" enabled="true" type="EventSpy" config="Event-
Processing-Utils/smarts-notifs-events/conf"/>
</processing-manager>
```

This is basic configuration of the Solution Pack which deals with how the processing should start.

2 **SMARTS Event Listener Configuration**:

The following example is a default SMARTS Event Listener configuration: This file will have all Smarts domain manager configurations for polling.

File: <DCF-Install>/Event-Processing/Smarts-Listener//<INSTANCE>(smarts-notifs-events)/conf/smarts-listener.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.watch4net.com/Events/SmartsEventListener"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                           xsi:schemaLocation="http://www.watch4net.com/Events/SmartsEventListener
smarts-listener.xsd ">
        <resync-period>24h</resync-period>
        <connection-check-period>30s</connection-check-period>
        <connection-timeout>60s</connection-timeout>
        <idle-timeout>240h</idle-timeout>
        <source id="INCHARGE-SA-PRES">
                <primary id="primary">
                        <broker-hostname>localhost</broker-hostname>
                        <broker-port>4999</broker-port>
                        <domain-name>INCHARGE-SA-PRES</domain-name>
                        <username>admin</username>

<password>{1BD4D26A81F980A80617601D0EAC255B85C79E7B064E2672F0CBF9EE8BC251A6D2F68C2751691B568BF7D00D
B41E7C61}</password>
                        <notification-list>ICS_NL-ALL_NOTIFICATIONS</notification-list>
                </primary>
        </source>
</configuration>
```

Supported configuration parameters are:

Parameters	Description
resync-period	Period at which the SMARTS Event Listener will initiate a resync with the SMARTS SAM. A resync operation synchronizes data between the SMARTS Event Listener and the notifications that are currently displayed in the SMARTS SAM console. Setting this value to 0 will disable automatic resynchronization.
connection-check-period	Period at which the SMARTS Event Listener will check to make sure that its connection with the SMARTS SAM is still valid.
connection-timeout	Timeout value when trying to establish a connection with the SMARTS SAM.
idle-timeout	If no new notification is received from the SMARTS SAM after that amount of time, the SMARTS Event Listener will disconnect then reconnect to the SMARTS. This checkup is performed at the same time as the connection check. Therefore, the value of this parameter should always be equal or greater than the connection-check-period.

Parameters		Description
source		A source represents a SMARTS SAM instance to monitor. Each source can be composed of a primary source and multiple failover sources, if needed. This is useful for specifying backup SMARTS SAM in case the primary is down.
	primary/failover	Determines if the source is the primary source or just a failover source.
	broker-hostname	The broker hostname of the SMARTS SAM if connecting to the broker as your entry point.
	broker-port	The port on which the SMARTS SAM's broker is accepting connections.
	broker-username (optional)	The username to use to establish the connection with the broker. This password can be in the encrypted form, generated by the crypt-password script. If no authentication is required, omit this element.
	broker-password (optional)	The password to use to establish the connection with the broker. This password can be in the encrypted form, generated by the crypt-password script. If no authentication is required, omit this element.
	hostname	The hostname of the SMARTS SAM if you need to directly connect to the manager.
	port	The port on which the SMARTS SAM is accepting connections.
	domain-name	The name of the domain from which metrics and topo data will be fetched.
	username	The username used to connect to the domain manager.
	password	The password used to connect to the domain manager. This password can be in the encrypted form, generated by the crypt-password script.

Multiple Sources The SMARTS Event Listener can be configured to listen to many sources Simultaneously. This is done by adding more source tags in the configuration file. Each source must have its primary source and can have one or more failover sources.

3 **Kafka Event Adapter Configuration**

Kafka server configuration used to publish events from Smarts/SAM to Kafka is below:

File : <DCF-Install>/Event-Processing/Kafka-Event-Adapter/<INSTANCE>(smarts-notifs-events)/conf/kafka-event-adapter.xml)

```
<?xml version="1.0"?>
<kafka-event-adapter-config xmlns="http://www.watch4net.com/KafkaEventAdapter"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.watch4net.com/KafkaEventAdapter ../kafka-event-adapter.xsd ">
  <cluster>
    <server host="localhost" port="9092"/>
    <additional-parameters key="security.protocol">SSL</additional-parameters>
        <additional-parameters key="ssl.truststore.location">../../../Tools/Webservice-
Gateway/Default/conf/truststore</additional-parameters>
    <additional-parameters
key="ssl.truststore.password">{F871B10293EEB1C941E2EA5466F817546662FD1314591713B73E73A7E39663A99602
55C4B844F536409BD410490E007F}</additional-parameters>
    <additional-parameters key="ssl.keystore.location">../../../Tools/Webservice-
Gateway/Default/conf/clientkeystore</additional-parameters>
    <additional-parameters
key="ssl.keystore.password">{85DF870D632462AF411DB8164B9455741BCCCB1CE493C475B7C121E5CEFA2189A2CDE6
CE65466BE4C2E99175CAEFA6F1}</additional-parameters>
    <additional-parameters
key="ssl.key.password">{DB17AC06BD3C6420FAA350241DFC43BE4504E0173A8BC2FA0C7FC9D79892374392195CD3EB1
5BD3D2D914FD470BF7075}</additional-parameters>
  </cluster>

    <event-writer topic-name="default-topic" stream="data" isJson="true">
        <kafka-producer-settings>
            <producer compression-type="none" />
        </kafka-producer-settings>
    </event-writer>

</kafka-event-adapter-config>
```

Kafka Event Adapter parameters:

Parameters	Description
cluste	This tag must occur at least once.
server	This tag must occur at least once. host: The address of one of the kafka bootstrap servers port: The port that the bootstrap server is listening on.
producer	This tag is optional, but may be used for templating kafka producers. Please refer to the schema for more information.
consumer	This tag is optional, but may be used for templating kafka consumers. Please refer to the schema for more information.

Parameters	Description
connection	This tag is optional, but may be used for templating kafka connections. Please refer to the schema for more information.
additional-parameters	This tag is optional, but may be used for configuring kafka options outside of the purview of the previous tags. These options include, but are not limited to SSL connection parameters.

Configuring Event Writer:

Parameters	Description
event-writer	This tag is used to define a component that will write to kafka. topic-name : The topic we will write to stream : The event stream to consume from isJson : Json format output is enabled
kafka-producer-settings	This tag may be used if we want to customize how the writer writes to kafka. Please refer to the schema for more information.
connector-component-behavior	This tag may be used if we want to control how often we flush to kafka.
key-encoder	This tag may be used if we want to customize how the kafka key is encoded.
value-encoder	This tag may be used if we want to customize how the kafka value is encoded.

Logging

All Event processing logs can be found at below path. This will have event spy logging info also, if enabled.

`<DCF-Install>/Event-Processing/Event-Processing-Manager/<INSTANCE>(ex:smarts-notifs-events)/logs/processing-0-0.log`

Latest processing log name will have "-0-0" appended to its name like above.

To enable logging, change the event processing configuration to push events to event spy like below. And after changing the configuration, operator need to restart the service to see the logging info in `processing-0-0.log`.

Edit below file as specified below:

```
File : <DCF-Install>/Event-Processing/Event-Processing-Manager/<INSTANCE>(smarts-notifs-
events)/conf/processing.xml
---
<processing-element name="Smarts" enabled="true" config="Smarts-Listener/smarts-notifs-
events/conf/smarts-listener.xml" data="KAFKA EVENT-SPY"/>
---
```

For example: Ex: "smarts-notifs-events" is installed management pack name in below example.

To check the status of the installed services:

```
<DCF-Install>/bin/manage-modules.sh service status all
….
* Checking 'event-processing-manager smarts-notifs-events'...       [ running ]
```

To restart the particular service:

```
<DCF-Install>/bin/manage-modules.sh service restart event-processing-manager smarts-notifs-events
 * Stopping 'event-processing-manager smarts-notifs-events'...            [ OK ]
 * Starting 'event-processing-manager smarts-notifs-events'...            [ OK ]
```

Sample output (JSON) of event in Kafka:

```
{
    "Acknowledged": false,
    "Active": true,
    "AuditTrail": [
        {
            "User": "admin",
            "ActionType": "UNACKNOWLEDGE",
            "SerialNumber": 3,
            "Text": "",
            "Timestamp": 1550654825
        },
        {
            "User": "admin",
            "ActionType": "RELEASE_OWNERSHIP",
            "SerialNumber": 2,
            "Text": "",
            "Timestamp": 1550654778
        },
        {
            "User": "admin",
            "ActionType": "ACKNOWLEDGE ALL",
            "SerialNumber": 1,
            "Text": "",
            "Timestamp": 1550654728
        },
        {
            "User": "DXA",
            "ActionType": "NOTIFY",
```

```
            "SerialNumber": 0,
            "Text": "Server: INCHARGE-AM-PM",
            "Timestamp": 1550654481
        }
    ],
    "Category": "Availability",
    "CausedBy": [],
    "Causes": [
        "NOTIFICATION-Router_10.107.116.235_Unresponsive",
        "NOTIFICATION-Router_200.0.4.1_Unresponsive",
        "NOTIFICATION-HSRPGroup_HSRP-GROUP-10.10.20.5_AllComponentsDown"
    ],
    "Certainty": 100.0,
    "ClassDisplayName": "Router",
    "ClassName": "Router",
    "ClearOnAcknowledge": false,
    "ElementClassName": "Router",
    "ElementName": "10.107.116.235",
    "EventDisplayName": "Down",
    "EventName": "Down",
    "EventState": "ACTIVE",
    "EventText": "Indicates that the root cause is a failed system.",
    "EventType": "DURABLE",
    "FirstNotifiedAt": 1550654416,
    "Impact": 0,
    "InMaintenance": false,
    "InstanceDisplayName": "10.107.116.235",
    "InstanceName": "10.107.116.235",
    "IsProblem": true,
    "IsRoot": true,
    "LastChangedAt": 1550654916,
    "Name": "NOTIFICATION-Router_10.107.116.235_Down",
    "OccurrenceCount": 1,
    "Owner": "admin",
    "PollingState": "SUBSCRIPTION",
    "ProcessedTimeStamp": 1550654916,
    "Severity": 1,
    "Source": "INCHARGE-SA-PRES",
    "SourceDomainName": "INCHARGE-AM-PM",
    "SourceEventType": "PROBLEM",
    "SourceInfo": "",
    "TroubleTicketID": "",
    "UserDefined1": "",
    "UserDefined10": "",
    "UserDefined11": "",
    "UserDefined12": "",
    "UserDefined13": "",
    "UserDefined14": "",
    "UserDefined15": "",
    "UserDefined16": "",
    "UserDefined17": "",
    "UserDefined18": "",
    "UserDefined19": "",
    "UserDefined2": "",
    "UserDefined20": "",
```

```
    "UserDefined3": "",
    "UserDefined4": "",
    "UserDefined5": "",
    "UserDefined6": "",
    "UserDefined7": "",
    "UserDefined8": "",
    "UserDefined9": ""
}
```

Sample debugging (Event-Spy) output from processing-0-0.log

```
INFO     -- [2019-03-13 16:58:14 IST] -- EventSpy$SpyStreamHandler::handleEvent(): From Smarts[data]:
INFO     -- [2019-03-13 16:58:14 IST] -- EventSpy$SpyStreamHandler::handleEvent():
com.watch4net.events.common.data.GenericEvent
(k) Name                = NOTIFICATION-Router_bq-
gwlab.lss.com_DiscoveryError

                                                                                (DEFINITION,STRING)
    ClassName           =
Router


 (DEFINITION,STRING)
    InstanceName        = bq-
gwlab.lss.com


(DEFINITION,STRING)
    EventName           =
DiscoveryError


 (DEFINITION,STRING)
```

VMware Cloudify Orchestrator Collector

9

The Cloudify Orchestrator collect Solution Pack is a DCF management pack used to pull all the Virtual IP Multimedia Subsystem(vIMS) components, VNFs data from Cloudify Orchestrator and publish the same to KAFKA for consumption.

This chapter includes the following topics:

Installation

Cloudify Orchestrator collect SP (cloudify-orchestrator-collect) installation is similar to any other SP installation and can be done through DCF module manager and also through controller.

Related package is available in DCF module repository by default:

Example: `<DCF-Install>/Tools/Module-Repository/ cloudify-orchestrator-collect-1.0.pkg`

Installation command:

```
<DCF-Install>/bin/manage-modules.sh install cloudify-orchestrator-collect <Collector_Instance_Name>
```

Sample Installation questions and output:

```
Required dependencies, in processing order:
   [1]   java '8.0.202' v8.0.202
   [2]   module-manager '1.13u1' v1.13u1
   [3] I collector-manager 'cloudify-orchestrator-collect' (none) => v5.10u1
   [4] I kafka-connector 'cloudify-orchestrator-collect' (none) => v1.0u2
   [5] I stream-collector 'cloudify-orchestrator-collect' (none) => v1.4u2
   [6] I cloudify-orchestrator-collect 'cloudify-orchestrator-collect' (none) => v1.0
> 2 not modified, 4 to install
> 32.5 MB space required / 17.1 GB available
 ? Enter the step to modify, 'yes' to accept them, or 'no' to cancel the operation [yes] >

Starting installation of Collector-Manager v5.10u1 from collector-manager-5.10u1-linux-x64...
 * Gathering information...
```

```
 * 'Collector-Manager v5.10u1' will be registered with instance name 'cloudify-orchestrator-collect'.
 * It will be installed in '/opt/DCF/Collecting/Collector-Manager/cloudify-orchestrator-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 60 files have been installed.
 * Finalizing installation...
 * Installing service 'collector-manager cloudify-orchestrator...  [ installed ]
Installation complete.

Starting installation of Kafka-Connector v1.0u2 from kafka-connector-1.0u2-linux-x64...
 * Gathering information...
 * 'Kafka-Connector v1.0u2' will be registered with instance name 'cloudify-orchestrator-collect'.
 * It will be installed in '/opt/DCF/Collecting/Kafka-Connector/cloudify-orchestrator-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 17 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Stream-Collector v1.4u2 from stream-collector-1.4u2-linux-x64...
 * Gathering information...
 * 'Stream-Collector v1.4u2' will be registered with instance name 'cloudify-orchestrator-collect'.
 * It will be installed in '/opt/DCF/Collecting/Stream-Collector/cloudify-orchestrator-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 46 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of cloudify-orchestrator-collect v1.0 from cloudify-orchestrator-collect-1.0...
 * Gathering information...
 * 'cloudify-orchestrator-collect v1.0' will be registered with instance name 'cloudify-orchestrator-
collect'.
 * It will be installed in '/opt/DCF/Block/cloudify-orchestrator-collect/cloudify-orchestrator-
collect'.
 * Unpacking files...
 * Installing files... 100%
 * 6 files have been installed.
 * Finalizing installation...
 ? Configure an Cloudify Orchestrator (yes/no) [y] >
 ? Hostname or IP address [localhost] > localhost
 ? Username [admin] > admin
 ? Password [•••••] > admin
 ? Cloudify Port [80] > 80
   [1] http
   [2] https
 ? Protocol [1] > 1
 ? More entries? (yes/no) [n] >
 ? Kafka Node Hostname or IP address [localhost] > localhost
 ? Kafka Node Port Number [9092] > 9092
 ? More entries? (yes/no) [n] >
 ? Kafka Topic name to publish Collected Data. [default-topic] > default-topic
 ? Kafka server authentication enabled? (SASL_PLAINTEXT only) (yes/no) [y] >
 ? Kafka server username. [admin] >
 ? Kafka server password. [•••••] >
```

```
 ? Do you want to start the installed services now? (yes/no) [n] > yes
 * Updating service 'collector-manager cloudify-orchestrator-col...  [ updated ]
 * Starting 'collector-manager cloudify-orchestrator-collect'...          [ OK ]
Installation complete.
```

Configuration

Solution Pack related configuration files are located at the following directories:

<DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(cloudify-orchestrator-collect)/conf/

<DCF-Install>/Collecting /Stream-Collector/<SP-Instance>/conf/

<DCF-Install>/Collecting /Kafka-Connector/<SP-Instance>/conf/

1 Collector Manager Configuration

A Collecting Manager Configuration file can be found at below location:

File: <DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(cloudify-orchestrator-collect) /conf/collecting.xml

Sample content:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/APG/Collecting"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.watch4net.com/APG/Collecting collecting.xsd ">
        <runOnce>true</runOnce>
    <connectors>
        <connector enabled="false" name="File" type="File-Connector"  config="conf/file-
connector.xml" />
        <connector enabled="true" name="Kafka" type="Kafka-Connector" config="Kafka-
Connector/cloudify-orchestrator-collect/conf/kafka-connector.xml" />
    </connectors>
        <collectors>
        <collector enabled="true" name="CloudifyDiscovery" next="Kafka" config="Stream-
Collector/cloudify-orchestrator-collect/conf/discovery-cloudify.xml" />
         </collectors>
</config>
```

This is basic configuration of the SP which deals with how the processing should start.

2 Cloudify Orchestrator discovery Configuration

The following example is a default Cloudify Orchestrator discovery configuration. This file will have all properties which are collected.

File: <DCF-Install>/Collecting/Stream-Collector/<SP-Instance>(cloudify-orchestrator-collect)/conf/discovery-cloudify.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<collector-configuration xmlns="http://www.watch4net.com/Text-Collector-Configuration"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.watch4net.com/Text-Collector-
Configuration ../textCollectorConfiguration.xsd ">
    <source>cloudify-orchestrator-collect</source>
    <collecting-group>group</collecting-group>
    <default-character-encoding>UTF-8</default-character-encoding>
    <properties-refresh-periods>10m</properties-refresh-periods>
    <collecting-threads-pool-size>30</collecting-threads-pool-size>

        <collecting-configurations name="cloudify-node-instances-request">
        <include-contexts>conf/context-cloudify.xml</include-contexts>
         <data-retrieval-file>conf/requests/cloudify-requests.xml</data-retrieval-file>
                <data-listeners id="CLOUDIFY-NODES" variable-id="id IPAddress" variable-id-
separator="-">
                <values context-key="CLOUDIFY_HOST">
                <name>CLOUDIFY-NODES-@{CLOUDIFY_HOST}</name>
                <unit>code</unit>
                <replace value=".+"   by="0" pattern="true"/>
            </values>
             <properties context-key="node" property-name="nodeId"/>
             <properties context-key="id" property-name="Name"/>
             <properties context-key="id" property-name="id"/>
             <properties context-key="host_id" property-name="hostId"/>
             <properties context-key="version" property-name="Version"/>
             <properties context-key="ip" property-name="IPAddress"/>
             <properties context-key="deployment_id" property-name="deploymentId"/>
             <properties context-key="tenant_name" property-name="tenantName"/>
             <properties context-key="state" property-name="State"/>
             <properties context-key="CloudifyIP" property-name="cloudifyAgentIPAddress"/>
             <properties context-key="classname" property-name="ClassName"/>
             <properties context-key="classname" property-name="type"/>
             <properties context-key="network_name" property-name="networkName"/>
             <properties context-key="network_id" property-name="networkId"/>
             <hardcoded-properties key="DisplayName">@{DisplayName} [@{ip}]</hardcoded-properties>
             <hardcoded-properties key="Certification">CERTIFIED</hardcoded-properties>
             <hardcoded-properties key="context-name">@{CLOUDIFY_HOST}</hardcoded-properties>
             <dynamic-relations/>
        </data-listeners>
        </collecting-configurations>
</collector-configuration>
```

Supported configuration parameters are:

Parameter	Description
group	The collector will assign the string specified in the group element to generated raw values. This group should match one of your processing component configuration.
threads pool-size	The thread pool-size attribute let you specify the number of concurrent polling threads which will share a customizable number of connections to smarts domain.
Properties refresh	The refresh attribute controls how often the collector should set the refresh tag on the generated values. It can be either a time with a period (00:00/86400 will refresh properties each day at midnight) or a number of polling loop (360 will refresh properties every 360 polling loops). In both cases, properties will always be refresh on first polling loop, right after the collector startup.
source	A source represents a cloudify orchestrator collect instance to discover.

3 Cloudify Orchestrator Requests Configuration

The following example is a default Cloudify Orchestrator requests configuration: This file will have all cloudify orchestrator REST API configuration.

File: `<DCF-Install>/Collecting/Stream-Collector/<SP-Instance>(cloudify-orchestrator-collect) /conf/cloudify-requests.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<data-retrieval-configuration xmlns="http://www.watch4net.com/Text-Parsing-Configuration"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.watch4net.com/Text-Parsing-
Configuration ../textParsingConfiguration.xsd" xmlns:xi="http://www.w3.org/2001/XInclude">
  <retrieving-period>900s</retrieving-period>
  <http-request data-retry="1" data-timeout="300s" connection-timeout="120s" character-
encoding="UTF-8">
       <lock name="perdevicelock-opst-a2-@{CLOUDIFY_HOST}" count="1"/>
       <credentials>
          <username>@{USERNAME}</username>
          <password>@{PASSWORD}</password>
       </credentials>
       <disable-ssl/>
       <request-groups name="RETRIEVE CLOUDIFYNODES">
            <requests options="GET" preAuth="true">
                <url>@{protocol}://@{CLOUDIFY_HOST}:@{PORT}/api/v3.1/node-instances</url>
                <headers key="Content-Type">application/json</headers>
                <headers key="Tenant">default_tenant</headers>
            </requests>
       </request-groups>
       <json-to-xml-transformer>
            <xml-dataset parse-datasets-in-parallel="true">
            <datasets>
                <xpath>/W4N/items/OBJECT</xpath>
                <xml-reader>
                    <extractions xpath-expression="/OBJECT/id">id</extractions>
```

```
                    <extractions xpath-expression="/OBJECT/node_id">node</extractions>
                    <extractions xpath-expression="/OBJECT/runtime_properties/ip">ip</extractions>

                    <extractions xpath-expression="if (substring-before(/OBJECT/node_id,'_host') =
'bono') then ('P-CSCF') else if (substring-before(/OBJECT/node_id,'_host') = 'sprout') then ('I/S-
CSCF') else if (substring-before(/OBJECT/node_id,'_host') = 'dime') then ('HSS/CDF') else if
(substring-before(/OBJECT/node_id,'_host') = 'homer') then ('XDMS') else if (substring-
before(/OBJECT/node_id,'_host') = 'vellum') then ('Vellum') else if (substring-
before(/OBJECT/node_id,'_host') = 'ellis') then ('Ellis') else if (substring-
before(/OBJECT/node_id,'_host') = 'bind') then ('Bind') else if (substring-
before(/OBJECT/node_id,'_host') = 'proxy') then ('Proxy') else (/OBJECT/node_id)" result-
type="string">DisplayName</extractions>

                    <extractions xpath-expression="if (substring-before(/OBJECT/node_id,'_host') =
'bono' or substring-before(/OBJECT/node_id,'_host') = 'sprout') then
('CallSessionControlFunctionsGroup:@{id}-@{ip}') else if (substring-
before(/OBJECT/node_id,'_host') = 'dime') then ('HSSMirrorFunctionGroup:@{id}-@{ip}')  else if
(substring-before(/OBJECT/node_id,'_host') = 'homer') then
('XMLDocumentManagementServersGroup:@{id}-@{ip}') else ('')" result-
type="string">~@RPartOf</extractions>

                    <extractions xpath-expression="if (substring-before(/OBJECT/node_id,'_host') =
'vellum' or substring-before(/OBJECT/node_id,'_host') = 'ellis' or substring-
before(/OBJECT/node_id,'_host') = 'proxy' or substring-before(/OBJECT/node_id,'_host') = 'bind')
then ('IPMultiMediaSubSystem:@{id}-@{ip}') else ('')" result-
type="string">~@RMemberOf</extractions>

                    <extractions xpath-expression="/OBJECT/host_id">host_id</extractions>
                    <extractions xpath-
expression="/OBJECT/runtime_properties/cloudify_agent/broker_ip">CloudifyIP</extractions>
                     <extractions xpath-
expression="/OBJECT/runtime_properties/cloudify_agent/version">version</extractions>
                     <extractions xpath-
expression="/OBJECT/deployment_id">deployment_id</extractions>
                     <extractions xpath-expression="/OBJECT/tenant_name">tenant_name</extractions>
                     <extractions xpath-expression="/OBJECT/state">state</extractions>
                     <extractions xpath-
expression="/OBJECT/runtime_properties/server/meta/cloudify_management_network_name">network_name
</extractions>
                     <extractions xpath-
expression="/OBJECT/runtime_properties/server/meta/cloudify_management_network_id">network_id</extr
actions>

                     <extractions xpath-expression="if (substring-before(/OBJECT/node_id,'_host')
= 'bono' or substring-before(/OBJECT/node_id,'_host') = 'sprout') then
'CallSessionControlFunction' else if (substring-before(/OBJECT/node_id,'_host') = 'homer') then
'XMLDocumentManagementServer' else if (substring-before(/OBJECT/node_id,'_host') = 'dime') then
'HSSMirrorFunction' else if (substring-before(/OBJECT/node_id,'_host') = 'vellum' or substring-
before(/OBJECT/node_id,'_host') = 'proxy' or substring-before(/OBJECT/node_id,'_host') = 'bind' or
substring-before(/OBJECT/node_id,'_host') = 'ellis') then 'VIMSApplicationService' else
('Unknown')" result-type="string">classname</extractions>

                     <release id="CLOUDIFY-NODES"/>
                </xml-reader>
            </datasets>
```

```
        </xml-dataset>
      </json-to-xml-transformer>
 </http-request>
</data-retrieval-configuration>
```

4 Kafka Connector Configuration

Kafka server configuration used to publish data collected from Cloudify Orchestrator to Kafka is below:

File : <DCF-Install>/Collecting/Kafka-Connector/<Sp-Instance>(cloudify-orchestrator-collect)/conf/kafka-connector.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<kafka-connector-config xmlns="http://www.watch4net.com/KafkaConnector"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.watch4net.com/KafkaConnector ../kafka-connector.xsd ">
        <kafka-producer-settings>
            <!-- Servers information -->
                <server host="localhost" port="9092"/>
            <!-- Producer (and topic) information -->
            <producer topic-name="default-topic"
                acks="1"
                retries="0"
                linger-ms="0ms"
                buffer-memory="33554432"
                compression-type="none"
                batch-size="16384"
                max-block-ms="5s"
                max-in-flight-requests-per-connection="5"
                max-request-size="1048576"/>

            <!-- All connection parameters will be attributes except "server" -->
            <connection request-timeout-ms="5s"
                connections-max-idle-ms="9m"
                retry-backoff-ms="100ms"
                reconnect-backoff-ms="50ms"/>

            <!-- Additional properties -->
            <additional-parameters key="metadata.max.age.ms">1000</additional-parameters>
            <additional-parameters key="receive.buffer.bytes">32768</additional-parameters>
            <additional-parameters key="send.buffer.bytes">131072</additional-parameters>

         <additional-parameters key="sasl.mechanism">PLAIN</additional-parameters>
        <additional-parameters key="security.protocol">SASL_PLAINTEXT</additional-parameters>
        <additional-parameters
key="sasl.jaas.config">org.apache.kafka.common.security.plain.PlainLoginModule required
username="admin"
password="{E7344C553A75234C634E12E19146CE3979270044BDC2859695928B121428BAC4F4336A0F6C9992A84B181372
57E6AA7F}";</additional-parameters>

        </kafka-producer-settings>
```

```
        <connector-component-behavior outputJson="true" flush-every-n-rawvalues="5000"/>

</kafka-connector-config>
```

Kafka Connector parameters:

Kafka Connector parameters

Parameters	Description
server	This tag must occur at least once. host: The address of one of the kafka bootstrap servers port: The port that the bootstrap server is listening on.
producer	This tag is optional, but may be used for templating kafka producers. Please refer to the schema for more information.
connection	This tag is optional, but may be used for templating kafka connections. Please refer to the schema for more information.
additional-parameters	This tag is optional, but may be used for configuring kafka options outside of the purview of the previous tags. These options include, but are not limited to SSL connection parameters.

Configuring Kafka Connector:

Parameters	Description
kafka-producer-settings	This tag may be used if we want to customize how the writer writes to kafka. Please refer to the schema for more information.
connector-component-behavior	This tag may be used if we want to control how often we flush to kafka.

Logging

All logs will be found in below path:

`<DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(cloudify-orchestrator-collect)/logs/collecting-0-0.log`

Latest processing log name will have "-0-0" appended to its name like above.

To enable logging, change the collecting configuration to push metrics/topology data to file connector like below. And after changing the configuration, operator need to restart the service to see the logging info in collecting-0-0.log.

Edit below file as specified below.

File : <DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(cloudify-orchestrator-collect)/conf/collecting.xml

```
---
<connector enabled="true" name="File" type="File-Connector"   config="conf/file-connector.xml" />

---
```

Example: "cloudify-orchestrator-collect" is installed management pack name in below example.

To check the status of the installed services:

```
* Checking collector-manager cloudify-orchestrator-collect'...       [ running ]

To restart the specific service:

<DCF-Install>/bin/manage-modules.sh service restart collector-manager cloudify-orchestrator-collect
 * Stopping collector-manager cloudify-orchestrator-collect'...            [ OK ]
 * Starting collector-manager cloudify-orchestrator-collect '...            [ OK ]
```

Sample output (JSON) of metrics data in Kafka:

```
{
 "groupName":"group",
 "discoveryID":null,
 "jobID":"9999",
 "type":"XMLDocumentManagementServer",
 "timestamp":1552452870,
 "value":0.0,
 "action":"r",
 "properties":{
    "context-name":"10.234.62.8",
    "networkName":"Private-Network",
    "hostId":"homer_host_xcyh8x",
    "source":"cloudify-orchestrator0b66e802-c13f-4df1-82b8-a2606fd059d4",
    "type":"XMLDocumentManagementServer",
    "Certification":"CERTIFIED",
    "cloudifyAgentIPAddress":"172.16.1.9",
    "Name":"homer_host_xcyh8x",
    "tenantName":"default_tenant",
    "Version":"4.5.0",
    "State":"started",
    "deploymentId":"Deployment",
    "DisplayName":"XDMS [172.16.1.11]",
    "ClassName":"XMLDocumentManagementServer",
    "networkId":"9d3569f0-1dc0-4a47-9135-3d2d63af3527",
    "id":"homer_host_xcyh8x",
    "IPAddress":"172.16.1.11",
    "nodeId":"homer_host"
},
 "metrics":{
    "CLOUDIFY-NODES-10.234.62.8":{
```

```
        "properties":{
        "unit":"code",
        "name":"CLOUDIFY-NODES-10.234.62.8"
        },
    "value":0.0
    }
 },
 "relations":[
 {
    "type":"XMLDocumentManagementServersGroup",
    "element":"homer_host_xcyh8x-172.16.1.11",
    "relationName":"PartOf"
 }
 ],
"initialized":true,
"forceRefresh":true,
"name":"homer_host_xcyh8x-172.16.1.11"
}
```

Sample debugging (File Connector) output from collecting-0-0.log

```
INFO     -- [2019-04-11 01:01:36 EDT] -- CollectorManagerImpl::configure(): Parsing file
'/opt/DCF/Collecting/Collector-Manager/cloudify-orchestrator-collect/conf/collecting.xml' ...
INFO     -- [2019-04-11 01:01:39 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.FileConnector for connector File...
INFO     -- [2019-04-11 01:01:39 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
connector File (FileConnector v1.0 rM) with /opt/DCF/Collecting/Collector-Manager/cloudify-
orchestrator-collect/conf/file-connector.xml...
INFO     -- [2019-04-11 01:01:39 EDT] -- CollectorManagerImpl::configureConnectors(): Skipping
disabled connector Kafka.
INFO     -- [2019-04-11 01:01:39 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.ubertext.collector.StreamCollector for collector CloudifyDiscovery...
INFO     -- [2019-04-11 01:01:39 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
collector CloudifyDiscovery (StreamCollector v1.4u2 rM) with /opt/DCF/Collecting/Stream-
Collector/cloudify-orchestrator-collect/conf/discovery-cloudify.xml...
INFO     -- [2019-04-11 01:01:45 EDT] -- CollectorManagerImpl::connect(): Connecting collectors...
INFO     -- [2019-04-11 01:01:45 EDT] -- AbstractCollector::registerNext(): Connecting
CloudifyDiscovery to File...
INFO     -- [2019-04-11 01:01:45 EDT] -- CollectorManagerImpl::configure(): endPacketSendlatch created
with count :1
INFO     -- [2019-04-11 01:01:45 EDT] -- CollectorManagerImpl::configure(): Starting
CollectionEndMsgTask thread...
INFO     -- [2019-04-11 01:01:45 EDT] -- CollectorManagerImpl::start(): Starting Collector Manager
v5.10u1...
INFO     -- [2019-04-11 01:01:45 EDT] -- CollectorManagerImpl::start(): Initializing components...
```

VMware VeloCloud SDWAN collector

10

The VeloCloud SDWAN collect SP is a DCF management pack used to discover and monitor the topology data from VeloCloud SDWAN operator setup. The data is then published into KAFKA for consumption. It is compatible with versions 9.x of the SMARTS domain managers.

This chapter includes the following topics:

Installation

VeloCloud SDWAN collector SP (velocloud-sdwan-collect) installation is similar to any other SP installation and can be done through DCF module manager and also through controller.

Related package will be available in DCF module repository by default:

Example: `<DCF_Install>/Tools/Module_Repository/velocloud_sdwan_collect_1.0.pkg`

Installation via DCF REST API:

Parameters:

- **host** - host where dcf is running.
- **port** - port using which we can access dcf.
- **block-name** - velocloud-sdwan.
- **block-id** - id of the block returned by the provision call.

Provision: To create a new velocloud collect block send the provision call via HTTP POST url: POST: https://{{host}}:{{port}}/dcc/v1/catalog/blocks/{{block-name}}/provision

Sample response: Note that the block-id as it would be used in the subsequent calls.

```
{
    "blockID": "velocloud_sdwan",
    "message": "Block successfully deployed",
    "time": "2019_04_05 21:54:11.976683",
    "instanceId": "velocloud_sdwan2326aa8e_f0e6_4986_856e_7a438f4db23a",
    "params": {
```

```
        "velocloud-sdwan": {
            "collect": {
                "collectormode": "velocloudiscovery"
            },
            "velocloud": {
                "host": [
                    {
                        "hostname": "localhost",
                        "username": "admin",
                        "password":
"{1B414DCCE74DCAF290CBCBD9AE61211A855201DD60FA6C286617522A2B0F55E7F76E7D87AFA37FC8397DA42B751232A4}",
                        "port": "443",
                        "protocol": "https"
                    }
                ]
            },
            "kafka": {
                "cluster": {
                    "node": [
                        {
                            "host": "localhost",
                            "port": "9092"
                        }
                    ],
                    "topic": "default-topic",
                    "auth": "true",
                    "username": "admin",
                    "password":
"{86929A3BEFEC196E5391B202FBE5736D1881D62A685BBFB4A54D2BF17E8B0790251F438D57118BE290650D207B22FEC8}"
                }
            },
            "collection_interval": "90",
            "runOnce_timeout": "3600"
        }
    }
}
```

Set config: Use the response from the provision call as a template for setting the velocloud block config in the POST call to set config. Do note that you can use this call to configure the collector in either discovery or monitoring mode. Make sure to correctly populate all the relevant fields before you send the call.

URL: https://{{host}}:{{port}}/dcc/v1/runtime/blocks/{{block-id}}/config/set

Sample post content:

```
{
        "velocloud-sdwan": {
            "collect": {
                "collectormode": "velocloudiscovery"
            },
            "velocloud": {
                "host": [
                    {
```

```
                        "hostname": "<ip address>",
                        "username": “<username>”,
                        "password": "<password>",
                        "port": "443",
                        "protocol": "https"
                    }
                ]
            },
            "kafka": {
                "cluster": {
                    "node": [
                        {
                            "host": "<kafka-hostname>",
                            "port": "<kafka-port>"
                        }
                    ],
                    "topic": "<kafka-topic>",
                    "auth": "true",
                    "username": "<kafka-username>",
                    "password": "<kafka-password>"
                }
            },
            "collection_interval": "90"
        }
    }
```

Start/Stop/Restart: To start, stop and restart the collector, use the following REST calls:

Start url: POST: https://{{host}}:{{port}}/dcc/v1/runtime/blocks/{{block-id}}/service/start

Stop url: POST: https://{{host}}:{{port}}/dcc/v1/runtime/blocks/{{block-id}}/service/stop

Restart url: POST: https://{{host}}:{{port}}/dcc/v1/runtime/blocks/{{block-id}}/service/restart

Delete: To Delete the block, use the following DEL API:

URL: DEL: https://{{host}}:{{port}}/dcc/v1/catalog/blocks/{{block-id}}

Installation via CLI: Sample Installation questions and output :

```
Required dependencies, in processing order:
   [1]   java '8.0.202' v8.0.202
   [2]   module-manager '1.13u1' v1.13u1
   [3] I collector-manager 'velocloud-sdwan-collect' (none) => v5.10u1
   [4] I kafka-connector 'velocloud-sdwan-collect' (none) => v1.0u2
   [5] I stream-collector 'velocloud-sdwan-collect' (none) => v1.4u2
   [6] I velocloud-sdwan-collect 'velocloud-sdwan-collect' (none) => v1.1
> 2 not modified, 4 to install
> 32.7 MB space required / 37.5 GB available
 ? Enter the step to modify, 'yes' to accept them, or 'no' to cancel the operation [yes] > yes



Starting installation of Collector-Manager v5.10u1 from collector-manager-5.10u1-linux-x64...
 * Gathering information...
 * 'Collector-Manager v5.10u1' will be registered with instance name 'velocloud-sdwan-collect'.
```

```
 * It will be installed in '/opt/DCF/Collecting/Collector-Manager/velocloud-sdwan-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 60 files have been installed.
 * Finalizing installation...
 * Installing service 'collector-manager velocloud-sdwan-colle...  [ installed ]
Installation complete.

Starting installation of Kafka-Connector v1.0u2 from kafka-connector-1.0u2-linux-x64...
 * Gathering information...
 * 'Kafka-Connector v1.0u2' will be registered with instance name 'velocloud-sdwan-collect'.
 * It will be installed in '/opt/DCF/Collecting/Kafka-Connector/velocloud-sdwan-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 17 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Stream-Collector v1.4u2 from stream-collector-1.4u2-linux-x64...
 * Gathering information...
 * 'Stream-Collector v1.4u2' will be registered with instance name 'velocloud-sdwan-collect'.
 * It will be installed in '/opt/DCF/Collecting/Stream-Collector/velocloud-sdwan-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 46 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of velocloud-sdwan-collect v1.1 from velocloud-sdwan-collect-1.1...
 * Gathering information...
 * 'velocloud-sdwan-collect v1.1' will be registered with instance name 'velocloud-sdwan-collect'.
 * It will be installed in '/opt/DCF/Block/velocloud-sdwan-collect/velocloud-sdwan-collect'.
 * Unpacking files...
 * Installing files... 100%
 * 28 files have been installed.
 * Finalizing installation...
   [1] velocloud discovery
   [2] velocloud monitoring
 ? VELOCLOUD Collection mode [1] > 1
 ? Do you want to configure velocloud host?  (yes/no) [y] > yes
 ? Hostname or IP address [localhost] > 1.1.1.1
 ? Username [admin] >
 ? Password [•••••] >
 ? Network Port [443] >
   [1] http
   [2] https
 ? Protocol [2] > 2
 ? More entries? (yes/no) [n] > n
? Kafka Node Hostname or IP address [localhost] >
 ? Kafka Node Port Number [9092] >
 ? More entries? (yes/no) [n] >
 ? Kafka Topic name to publish Collected Data. [default-topic] >
   [1] Kafka server authentication disabled.
   [2] SASL_PLAINTEXT.
   [3] SASL_SSL.
```

```
    [4] SSL.

 If we select default option 2:-

 ? Kafka server authentication type. [2] > SASL_PLAINTEXT
 ? Kafka server username. [admin] >
 ? Kafka server password. [•••••] >
 ? Do you want to start the installed services now? (yes/no) [n] > n

If we select option 4 [SSL] :-

? Kafka server authentication type. [2] > 4
 ? Location for Kafka Producer/Consumer TrustStore (Java TrustStore).   [../../../Tools/Webservice-
Gateway/Default/conf/truststore] >
 ? Password for Kafka Producer/Consumer TrustStore (Java TrustStore). [•••••] >
 ? Location for Kafka Producer/Consumer KeyStore (Java KeyStore). [../../../Tools/Webservice-
Gateway/Default/conf/clientkeystore] >
 ? Password for Kafka Producer/Consumer KeyStore (Java KeyStore). [•••••] >
 ? Password for KeyStore Key. [•••••] >


? Do you want to start the installed services now? (yes/no) [n] > yes
```

Configuration

SP related configuration files can be located in following directories:

<DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(velocloud-sdwan-collect)/conf/

<DCF-Install>/ Collecting /Stream-Collector/<SP-Instance>/conf/

<DCF-Install>/ Collecting /Kafka-Connector/<SP-Instance>/conf/

1 Collector Manager Configuration

A Collecting Manager Configuration file can be found in below location :

File: <DCF-Install>/ Collecting/Collector-Manager /<INSTANCE>(velocloud-sdwan-collect)/conf/collecting.xml

Sample contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/APG/Collecting"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.watch4net.com/APG/Collecting collecting.xsd ">
                        <runOnce>true</runOnce>
    <connectors>
        <connector enabled="false" name="File" type="File-Connector"  config="Collector-
Manager/velocloud-sdwan-collect/conf/file-connector.xml" />
        <connector enabled="true" name="Kafka" type="Kafka-Connector" config="Kafka-
Connector/velocloud-sdwan-collect/conf/kafka-connector.xml" />
    </connectors>
```

```
    <filters>
    </filters>
        <collectors>
        <collector enabled="true" name="VCODiscover" next="File Kafka" config="Stream-
Collector/velocloud-sdwan-collect/velo/conf/discovery-velocloud.xml" />
         </collectors>
</config>
```

This is basic configuration of the SP which deals with how the processing should start.

2 Getting configuration from the collector using REST API

a **Config**: The configuration for the collector block can be obtained using the following GET REST request:

URL: GET: https://{{host}}:{{port}}/dcc/v1/catalog/blocks/{{block-name}}/config/get

b **Default Config**: The default configuration for the collector block can be obtained using the following GET REST request url: GET: https:// {{host}}:{{port}}/dcc/v1/catalog/blocks/{{block-name}}/config/default

3 Kafka Connector Configuration

Kafka server configuration used to publish events from velocloud sdwan DCF collector to Kafka is below:

File : `<DCF-Install>/Collecting/Kafka-Connector/<INSTANCE>(velocloud-metrics-collect)/conf/kafka-connector.xml)`

```
<kafka-connector-config xmlns="http://www.watch4net.com/KafkaConnector"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.watch4net.com/KafkaConnector ../kafka-connector.xsd ">
        <kafka-producer-settings>
            <!-- Servers information -->
                <server host="localhost" port="9092"/>
            <!-- Producer (and topic) information -->
            <producer topic-name="default-topic"
                acks="1"
                retries="0"
                linger-ms="0ms"
                buffer-memory="33554432"
                compression-type="none"
                batch-size="16384"
                max-block-ms="5s"
                max-in-flight-requests-per-connection="5"
                max-request-size="1048576"/>

            <!-- All connection parameters will be attributes except "server" -->
            <connection request-timeout-ms="5s"
                connections-max-idle-ms="9m"
                retry-backoff-ms="100ms"
                reconnect-backoff-ms="50ms"/>

            <!-- Additional properties -->
            <additional-parameters key="metadata.max.age.ms">1000</additional-parameters>
```

```
            <additional-parameters key="receive.buffer.bytes">32768</additional-parameters>
            <additional-parameters key="send.buffer.bytes">131072</additional-parameters>

        <additional-parameters key="security.protocol">SSL</additional-parameters>
            <additional-parameters key="ssl.truststore.location">../../../Tools/Webservice-
Gateway/Default/conf/truststore</additional-parameters>
        <additional-parameters
key="ssl.truststore.password">{613FF4F84B7A36EC8D22728760D70A56FF2CB8E9CCCD90367BFEEB74E5B97EAB1B76
5AA92F50F91101E757D131BD5A4D}</additional-parameters>
        <additional-parameters key="ssl.keystore.location">../../../Tools/Webservice-
Gateway/Default/conf/clientkeystore</additional-parameters>
        <additional-parameters
key="ssl.keystore.password">{9F9B74AF79C19897075C1CFEC22D542F018838BEFE763E4DF180850D474320D2EC9595
E36F68D90A1A830175BF42D8A7}</additional-parameters>
        <additional-parameters
key="ssl.key.password">{FE076F391C3C5946DF2EDF230CAF272D601423289B8BFF9C29F246D089C153D01E2E5A0B1A1
F714C814DDF3088F3982F}</additional-parameters>

        </kafka-producer-settings>

        <connector-component-behavior outputJson="true" flush-every-n-rawvalues="5000"/>

</kafka-connector-config>
```

Kafka Connector parameters

Parameters	Description
server	This tag must occur at least once. host: The address of one of the kafka bootstrap servers port: The port that the bootstrap server is listening on.
producer	This tag is optional, but may be used for templating kafka producers. Please refer to the schema for more information.
connection	This tag is optional, but may be used for templating kafka connections. Please refer to the schema for more information.
additional-parameters	This tag is optional, but may be used for configuring kafka options outside of the purview of the previous tags. These options include, but are not limited to SSL connection parameters.

Configuring Kafka Connector:

Parameters	Description
kafka-producer-settings	This tag may be used if we want to customize how the writer writes to kafka. Please refer to the schema for more information.
connector-component-behavior	This tag may be used if we want to control how often we flush to kafka.

Logging

All Event processing logs will be found in below path. This will have event spy logging info also, if enabled.

`DCF-Install>/Collecting/Collector-Manager/<INSTANCE>(velocloud-sdwan-collect)/logs/collecting-0-0.log`

Latest processing log name will have "-0-0" appended to its name like above.

To enable data collection into a file for debugging purposes, change the collecting configuration to push velocloud data to file connector like below instead of just the kafka bus. And after changing the configuration, operator need to restart the service to see the logging info in collecting-0-0.log.

Edit the file as specified below:

File : `<DCF-Install>/Collecting/Collector-Manager/<INSTANCE>(velocloud-sdwan-collect)/conf/collecting.xml`

```
---
<connector enabled="true" name="File" type="File-Connector"   config="conf/file-connector.xml" />

---
```

Example: "velocloud-sdwan-collect" is installed management pack name in below example.

To check the status of the installed services:

```
<DCF-Install>/bin/manage-modules.sh service status all
….
* Checking collector-manager velocloud-sdwan-collect'...       [ running ]
```

To restart the particular service:

```
<DCF-Install>/bin/manage-modules.sh service restart collector-manager velocloud-sdwan-collect
 * Stopping collector-manager velolcoud-sdwan-collect'...            [ OK ]
 * Starting collector-manager velocloud-sdwan-collect'...            [ OK ]
```

Sample output (JSON) of data in Kafka:

```
{"groupName":"group","discoveryID":null,"jobID":"9999","type":"QoeData","timestamp":1554934113,"value":
0.0,"action":"r","properties":{"internalId":"00000004-
e0d1-4c9f-810a-262fbece4ff3","edgeId":"2","linkId":"4","instanceName":"EdgeLinkQoE","attributes":"Perfo
rmance","source":"VC-Collector","enterpriseId":"1","type":"QoeData","totalScore":"10"},"metrics":{"VCO-
Req":{"properties":{"unit":"code","name":"VCO-Req"},"value":0.0}},"relations":
[],"initialized":true,"forceRefresh":true,"name":""}
```

Sample debugging (File Connector) output from collecting-0-0.log

```
INFO     -- [2019-04-10 18:08:39 EDT] -- StreamCollector::sendDiagnostics(): Jobs for Context
10.148.87.69 completed
INFO     -- [2019-04-10 18:08:39 EDT] -- StreamCollector::sendDiagnostics():
Name:vco-monitoring-request Status:SUCESS Total failed jobs:0
```

VMware vIMS SNMP collector

11

This chapter includes the following topics:

Overview

e vIMS (Clearwater) SNMP collect SP is a DCF management pack used to pull the SNMP performance metrics from SNMP enabled Clearwater device and publish the same to KAFKA for consumption. It is fully compatible with versions 6.x, 7.x and 8.x versions of the SMARTS domain managers.

Installation

vIMS (Clearwater) SNMP Perfomance metrics SP (clearwater-collect-1.0.pkg) installation is similar to any other SP installation and can be done through DCF module manager and also through controller.

Related package will be available in DCF module repository by default.

Example: <DCF-Install>/Tools/Module-Repository/clearwater-collect-1.0.pkg

Installation command :

<DCF-Install>/bin/manage-modules.sh install clearwater-collect-1.0 clearwater

Sample Installation questions and output:

```
Required dependencies, in processing order:
   [1]   java '8.0.202' v8.0.202
   [2]   module-manager '1.13u1' v1.13u1
   [3] I collector-manager 'clearwater' (none) => v5.10u1
   [4] I kafka-connector 'clearwater' (none) => v1.0u2
   [5] I variable-handling-filter 'clearwater' (none) => v1.18u2
   [6] I snmp-collector 'clearwater' (none) => v6.8u2
   [7] I clearwater-collect 'clearwater' (none) => v1.0
> 2 not modified, 5 to install
```

```
> 44.2 MB space required / 28.6 GB available
 ? Enter the step to modify, 'yes' to accept them, or 'no' to cancel the operation [yes] > y

Starting installation of Collector-Manager v5.10u1 from collector-manager-5.10u1-linux-x64...
 * Gathering information...
 * 'Collector-Manager v5.10u1' will be registered with instance name 'clearwater'.
 * It will be installed in '/opt/DCF/Collecting/Collector-Manager/clearwater'.
 * Unpacking files...
 * Installing files... 100%
 * 60 files have been installed.
 * Finalizing installation...
 ! WARNING: Systemd version 218 or greater is required. Current version 208 is too old and needs to be
updated to ensure proper shutdown of services when the server shuts down!
 * Installing service 'collector-manager clearwater'...               [ installed ]
Installation complete.

Starting installation of Kafka-Connector v1.0u2 from kafka-connector-1.0u2-linux-x64...
 * Gathering information...
 * 'Kafka-Connector v1.0u2' will be registered with instance name 'clearwater'.
 * It will be installed in '/opt/DCF/Collecting/Kafka-Connector/clearwater'.
 * Unpacking files...
 * Installing files... 100%
 * 17 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Variable-Handling-Filter v1.18u2 from variable-handling-filter-1.18u2-linux-
x64...
 * Gathering information...
 * 'Variable-Handling-Filter v1.18u2' will be registered with instance name 'clearwater'.
 * It will be installed in '/opt/DCF/Collecting/Variable-Handling-Filter/clearwater'.
 * Unpacking files...
 * Installing files... 100%
 * 35 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of SNMP-Collector v6.8u2 from snmp-collector-6.8u2-linux-x64...
 * Gathering information...
 * 'SNMP-Collector v6.8u2' will be registered with instance name 'clearwater'.
 * It will be installed in '/opt/DCF/Collecting/SNMP-Collector/clearwater'.
 * Unpacking files...
 * Installing files... 100%
 * 387 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of clearwater-collect v1.0 from clearwater-collect-1.0...
 * Gathering information...
 * 'clearwater-collect v1.0' will be registered with instance name 'clearwater'.
 * It will be installed in '/opt/DCF/Block/clearwater-collect/clearwater'.
 * Unpacking files...
 * Installing files... 100%
 * 14 files have been installed.
 * Finalizing installation...
```

```
? Configure an Clearwater Sprout Node (yes/no) [y] > y
 ? Hostname or IP address of the Clearwater Sprout Node [localhost] > localhost
 ? SNMP Port  [161] >
   [1] udp
 ? Select SNMP Protocol from the above list [1] >
 ? SNMP Community String Value [•••••] >
   [1] v2c
 ? Select SNMP Version from the above list [1] >
 ? Deployment Id of the vIMS [default] > ClearwaterDeployment
 ? Orchestrator Hostname or IP address [default] > localhost
 ? Timeout [30] >
 ? More entries? (yes/no) [n] >
 ? Configure an Clearwater Bono Node (yes/no) [y] >
 ? Hostname or IP address of the Clearwater Bono Node [localhost] > localhost
 ? SNMP Port  [161] >
   [1] udp
 ? Select SNMP Protocol from the above list [1] >
 ? SNMP Community String Value [•••••] >
   [1] v2c
 ? Select SNMP Version from the above list [1] >
 ? Deployment Id of the vIMS [default] > ClearwaterDeployment
 ? Orchestrator Hostname or IP address [default] >
 ? Timeout [30] >
 ? More entries? (yes/no) [n] >
 ? Kafka Node Hostname or IP address [localhost] >
 ? Kafka Node Port Number [9092] >
 ? More entries? (yes/no) [n] >
 ? Kafka Topic name to publish Collected Data. [default-topic] > clearwater-snmp
 ? Kafka server authentication enabled? (SASL_PLAINTEXT only) (yes/no) [y] > y
 ? Kafka server username. [admin] > admin
 ? Kafka server password. [•••••] >
 ? Advanced configurations (yes/no) [y] > y
 ? Do you want to change the Polling Period (Default: 5 mins) (yes/no) [y] > y
 ? Sprout data Polling period (in Seconds) Range:60 to 3600 [300] > 600
 ? Bono data Polling period (in Seconds) Range:60 to 3600 [300] > 600
 ? Do you want to start the installed services now? (yes/no) [n] > y
 ! WARNING: Systemd version 218 or greater is required. Current version 208 is too old and needs to be
updated to ensure proper shutdown of services when the server shuts down!
 * Updating service 'collector-manager clearwater'...                  [ updated ]
 * Starting 'collector-manager clearwater'...                              [ OK ]
Installation complete.
```

Configuration

SP related configuration files can be located in following directories:

<DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(clearwater)/conf/

<DCF-Install>/ Collecting /SNMP -Collector/<SP-Instance>/conf/

<DCF–Install>/ Collecting /Kafka–Connector/<SP–Instance>/conf/

1 Collector Manager Configuration

A Collecting Manager Configuration file can be found in below location:

File: <DCF–Install>/ Collecting/Collector–Manager /<INSTANCE>(clearwater)/conf/collecting.xml

Sample contents :

```
<?xml version="1.0" encoding="UTF–8"?>
<config xmlns="http://www.watch4net.com/APG/Collecting"
xmlns:xsi="http://www.w3.org/2001/XMLSchema–instance"
        xsi:schemaLocation="http://www.watch4net.com/APG/Collecting collecting.xsd ">
    <connectors>
        <connector enabled="false" name="File" type="File–Connector"   config="conf/file–
connector.xml" />
        <connector enabled="true" name="Kafka" type="Kafka–Connector"   config="Kafka–
Connector/clearwater/conf/kafka–connector.xml" />
    </connectors>
    <filters>
        <filter enabled="true" name="Param–CLEANUP" next="Kafka" config="Variable–Handling–
Filter/clearwater/conf/vhf–param–cleanup.xml"/>
    </filters>
        <collectors>
        <collector enabled="true" name="CLEARWATER–SNMP" next="Param–CLEANUP" config="SNMP–
Collector/clearwater/conf/snmpcollector.xml" />
        </collectors>
  </config>
```

This is basic configuration of the SP which deals with how the processing should start.

2 SNMP Metrics Collecting Configuration Files

To configure the SNMP collector, the following configuration files are required:

a Snmpcollector.xml: This file specifies the file path for other configuration files.

```
<?xml version="1.0" encoding="UTF–8"?>
<config>
    <slave–snmp–poller–file >conf/slave–snmp–poller.xml</slave–snmp–poller–file>
    <snmp–masks–file >conf/snmp–masks.xml</snmp–masks–file>
    <snmp–polling–distribution–file >conf/snmp–polling–distribution.xml</snmp–polling–
distribution–file>
    <translations–file>conf/translations.xml</translations–file>
    <mib–files–directory >mibs</mib–files–directory>
</config>
```

b slave-snmp-poller.xml: This file associates SNMP Agents to SnmpMasks in PollingGroups.

```
<?xml version="1.0" encoding="UTF–8"?><slave–snmp–poller name="ClearwaterCollector"
pollerPrefix="SNMP–CLEARWATER" groupName="group" oidsThreads="100" tablesThreads="100"
alwaysPushProperties="true" handleCounterWrapAround="true" usePollerPrefixInVariable="false"
oidAsPropertyInRawValues="false" configVersion="6.6"
```

```
xmlns:xi="http://www.w3.org/2001/XInclude">
  <local-communication-settings hostIpAddress="localhost" communicationPort="52006"
communicationInterfaceIPAddress="0.0.0.0" maxConcurrentSessions="98">
    <out-interface-ip-addresses>
      <out-interface-ip-address>0.0.0.0</out-interface-ip-address>
    </out-interface-ip-addresses>
    <accept-commands-from-list/>
  </local-communication-settings>
  <xi:include  href="polling-groups.xml"/>
</slave-snmp-poller>
```

polling-groups.xml:

```
<?xml version="1.0" encoding="UTF-8"?><polling-groups>
  <polling-group name="PG_CLEARWATER-BONO" pollingPeriod="300" tableDiscoveryTime="00:42:00"
groupName="group" uniformTimestamps="false" startAtDefinedPeriod="false" enabled="true"
startTimeOffset="0">
    <polling-masks>
      <polling-mask-name>CLEARWATER-BONO</polling-mask-name>
    </polling-masks>
    <polled-snmp-agents>
      <snmp-agents-explicit-group-name>PG_CLEARWATER-BONO</snmp-agents-explicit-group-name>
    </polled-snmp-agents>
  </polling-group>
  <polling-group name="PG_CLEARWATER-SPROUT" pollingPeriod="300" tableDiscoveryTime="00:42:00"
groupName="group" uniformTimestamps="false" startAtDefinedPeriod="false" enabled="true"
startTimeOffset="0">
    <polling-masks>
      <polling-mask-name>CLEARWATER-SPROUT</polling-mask-name>
    </polling-masks>
    <polled-snmp-agents>
      <snmp-agents-explicit-group-name>PG_CLEARWATER-SPROUT</snmp-agents-explicit-group-name>
    </polled-snmp-agents>
  </polling-group>
</polling-groups>
```

Polling Group attributes are:

1 **name**: The polling group's name.

2 **pollingPeriod**: The polling group's polling period, specified in seconds. This value indicate the time interval between each polling.

3 **groupName**: This attribute overrides the retention group name specified in the slave-snmp-poller element for this polling group and all of the ones defined under it.

4 **polling-masks**: This element set contains polling-mask-name elements. At least one pollingmask- name element must be provided per polling-group.

5 **polled-snmp-agents**: This element set contains at least one snmp-agent-ip-address, snmpagents- ip-group-name or snmp-agents-explicit-group-name element. These elements actually indicate which SNMP agent to poll in this polling group.

c Snmp-masks.xml: SNMP collecting process defined in the snmp-masks.xml file. Clearwater uses SNMP-table polling method.

Sample snmp-masks file:

```
<snmp-mask name="CLEARWATER-SPROUT" doNotEdit="true" version="1.0.0.0">
    <snmp-properties>
      <snmp-property id="vendor" name="vendor" value="Clearwater"/>
      <snmp-property id="type" name="type" value="Sprout"/>
      <snmp-property id="datagrp" name="datagrp" value="PM"/>
    </snmp-properties>
    <snmp-tables>
      <snmp-table id="ICSCFSessionEstablishmentNetworkTable"
name="ICSCFSessionEstablishmentNetworkTable" oid=".1.2.826.0.1.1578918.9.3.37" tableIndex=".
1.1" tableColNameOid=".1.1" indexEntryMask="oid._._.1" useSnmpV2cMultipleRowPolling="true"
cacheAllSnmpReferenceTables="false" sortOnOid="false" matchNFirstOidSuffixOnly="0">
        <columns-and-filters>
          <snmp-column name="ICSCFSessionEstablishmentNetworkAttempts" appendOid=".1.2"
unit="integer" refreshOnPropertyChange="false"/>
          <snmp-column name="ICSCFSessionEstablishmentNetworkSuccesses" appendOid=".1.3"
unit="integer" refreshOnPropertyChange="false"/>
          <snmp-column name="ICSCFSessionEstablishmentNetworkFailures" appendOid=".1.4"
unit="integer" refreshOnPropertyChange="false"/>
          <snmp-column name="ICSCFSessionEstablishmentNetworkSuccessPercent" appendOid=".1.5"
unit="integer" refreshOnPropertyChange="false"/>
        </columns-and-filters>
      </snmp-table>
     </snmp-tables>
    <mib-files>
      <mib-file-name>INET-ADDRESS-MIB.txt</mib-file-name>
      <mib-file-name>PROJECT-CLEARWATER-MIB</mib-file-name>
    </mib-files>
```

Snmp-masks attributes are:

1 **id**: The table's id, which must be unique among all snmp-oid and snmp-table elements.

2 **name**: The table's name.

3 **oid**: The table's base oid. i.e.: ifTable is .1.3.6.1.2.1.2.2.

4 **tableIndex**: The table's index columns. The form MUST BE AS FOLLOWS: ".1.1 .1.2" This indicates column 1 and 2 of the table.

5 **tableColNameOid**: Specifies which column values must be used, in the table, to set the part property for all polled columns. i.e.: tableColNameOid=".1.1 .1.2" to use column 1 and 2 values.

6 **indexEntryMask**: This describes the table's rows oid structure and is used to generate index pseudo-columns when the table does not have explicit index columns.

7 **partType**: Specifies the raw value's property parttype. This is user defined.

8 **name**: the oid name. It may or may not be the same as the oid name specified in a MIB. This is the name property in the resulting rawValue.

9 **appendOid**: This specifies the table's column to poll. It's format is ALWAYS ".1.X", where X is the column number.

10 **MIB Files**: Mib files are used to get textual values from integer enumerated values. The textual value will be set as the property value if polled as a property.

```
<mib-files>
      <mib-file-name>INET-ADDRESS-MIB.txt</mib-file-name>
      <mib-file-name>PROJECT-CLEARWATER-MIB</mib-file-name>
    </mib-files>
```

d agents-groups.xml

```
<?xml version="1.0" encoding="UTF-8"?><agents-groups>
  <snmp-agent ipAddress="localhost" transportProtocol="udp" snmpPort="161" timeout="30000"
retries="2" snmpVersion="v2c"
snmpCommunityOrUserV3Name="{5563E4FF395EB0585419E36F21AA940ECB70A8EEBEE91682A2C57A9CA4C4E0F31F5
1CD48E0EBC0B099F148A88002B4AF}" maxBulkSize="50" devtype="vIMS"
doNotUseGetBulkRequests="false" onFailTryWith="v1"/>
  <snmp-agents-explicit-group name="PG_CLEARWATER-BONO">
    <agent-ip-address>localhost:161</agent-ip-address>
  </snmp-agents-explicit-group>
  <snmp-agents-explicit-group name="PG_CLEARWATER-SPROUT">
    <agent-ip-address>localhost:161</agent-ip-address>
  </snmp-agents-explicit-group>
</agents-groups>
```

Agents-groups attributes are:

1 **ipAddress**: This is the ipAddress of the SNMP agent (Clearwater).

2 **transportProtocol**: udp is by default.

3 **snmpPort**: This is the SNMP port to use when polling the SNMP agent. Port 161 is set by default.

4 **snmpVersion**: the SNMP version to use when polling the agent. Defaults to v2c for clearwater.

5 **snmpCommunityOrUserV3Name**: SNMP v2c community string.

6 **snmp-agents-explicit-group**: Specifies SNMP agent group name.

7 **agent-ip-address**: SNMP agent IP address and its port.

3 Kafka Connector Configuration: Kafka server configuration used to publish SNMP performance metrics from clearwater to Kafka is below:

File : <DCF-Install>/Collecting/Kafka-Connector/<INSTANCE>(clearwater)/conf/kafka-connector.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<kafka-connector-config xmlns="http://www.watch4net.com/KafkaConnector"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.watch4net.com/KafkaConnector ../kafka-connector.xsd ">
        <kafka-producer-settings>
            <!-- Servers information -->
                <server host="localhost" port="9092"/>
            <!-- Producer (and topic) information -->
            <producer topic-name="clearwater-snmp"
                acks="1"
                retries="0"
                linger-ms="0ms"
                buffer-memory="33554432"
                compression-type="none"
                batch-size="16384"
                max-block-ms="5s"
                max-in-flight-requests-per-connection="5"
                max-request-size="1048576"/>

            <!-- All connection parameters will be attributes except "server" -->
            <connection request-timeout-ms="5s"
                connections-max-idle-ms="9m"
                retry-backoff-ms="100ms"
                reconnect-backoff-ms="50ms"/>

            <!-- Additional properties -->
            <additional-parameters key="metadata.max.age.ms">1000</additional-parameters>
            <additional-parameters key="receive.buffer.bytes">32768</additional-parameters>
            <additional-parameters key="send.buffer.bytes">131072</additional-parameters>

                <additional-parameters key="sasl.mechanism">PLAIN</additional-parameters>
        <additional-parameters key="security.protocol">SASL_PLAINTEXT</additional-parameters>
        <additional-parameters
key="sasl.jaas.config">org.apache.kafka.common.security.plain.PlainLoginModule required
username="admin"
password="{BCB52C90E273915E2727C2D77EDFFED922A6608165C43784D1FB5B24FBB18A2434C8260D30DF673B67D17BA5
870177A2}";</additional-parameters>

        </kafka-producer-settings>

        <connector-component-behavior outputJson="true" flush-every-n-rawvalues="5000"/>

</kafka-connector-config>
```

Kafka Connector parameters

Parameters	Description
server	This tag must occur at least once. host: The address of one of the kafka bootstrap servers port: The port that the bootstrap server is listening on.
producer	This tag is optional, but may be used for templating kafka producers. Please refer to the schema for more information.
connection	This tag is optional, but may be used for templating kafka connections. Please refer to the schema for more information.
additional-parameters	This tag is optional, but may be used for configuring kafka options outside of the purview of the previous tags. These options include, but are not limited to SSL connection parameters.

Configuring Kafka Connector:

Parameters	Description
kafka-producer-settings	This tag may be used if we want to customize how the writer writes to kafka. Please refer to the schema for more information.
connector-component-behavior	This tag may be used if we want to control how often we flush to kafka.

Note Few raw metrics from kafka are directly read by vROps management pack, and few other raw metrics are read by K4M and are processed back to Kafka, which in-turn read by vROps. Refer *vRealize Operations Management Pack for Smart Assurance Adapter Guide* and *VMware Smart Assurance K4M KPI Designer User Guide* for more information.

Logging

All Event processing logs will be found in below path. This will have event spy logging info also, if enabled.

```
<DCF_Install>/Collecting/Collector_
Manager/<INSTANCE>(ex:clearwater)/logs/collecting_0_0.log
```

Latest processing log name will have "-0-0" appended to its name like above.

To enable logging, change the collecting configuration to push metrics/topology data to file connector like below. And after changing the configuration, operator need to restart the service to see the logging info in collecting-0-0.log.

Edit below file as specified below.

File : <DCF-Install>/Collecting/Collector-Manager/<INSTANCE>(clearwater)/conf/collecting.xml

```
---
<connector enabled="true" name="File" type="File-Connector"   config="conf/file-connector.xml" />

---
```

To check the status of the installed services:

```
<DCF-Install>/bin/manage-modules.sh service status all
* Checking 'collector-manager clearwater'...                          [ running ]

To restart the particular service:

<DCF-Install>/bin/manage-modules.sh service restart collector-manager clearwater
* Stopping 'collector-manager clearwater'...                             [ OK ]
 * Starting 'collector-manager clearwater'...                             [ OK ]
```

Sample JSON output:

```
{"groupName":"group","discoveryID":null,"jobID":"9999","type":"Sprout","timestamp":1554880287,"value":
0.0,"action":"r","properties":
{"oidIndex":"1","ip":"10.106.230.99","part":"scopePrevious5SecondPeriod","source":"SNMP-
CLEARWATER","type":"Sprout","pollgrp":"PG_CLEARWATER-SPROUT","devdesc":"Linux cw-aio 3.16.0-30-generic
#40~14.04.1-Ubuntu SMP Thu Jan 15 17:43:14 UTC 2015
x86_64","devtype":"vIMS","orchestrator":"default","vendor":"Clearwater","deploymentid":"default","locat
ion":"","model":"","table":"ICSCFSessionEstablishmentNetworkTable","datagrp":"PM"},"metrics":
{"ICSCFSessionEstablishmentNetworkSuccesses":{"properties":
{"unit":"integer","name":"ICSCFSessionEstablishmentNetworkSuccesses"},"value":
0.0},"ICSCFSessionEstablishmentNetworkAttempts":{"properties":
{"unit":"integer","name":"ICSCFSessionEstablishmentNetworkAttempts"},"value":
0.0},"ICSCFSessionEstablishmentNetworkFailures":{"properties":
{"unit":"integer","name":"ICSCFSessionEstablishmentNetworkFailures"},"value":
0.0},"ICSCFSessionEstablishmentNetworkSuccessPercent":{"properties":
{"unit":"integer","name":"ICSCFSessionEstablishmentNetworkSuccessPercent"},"value":
1000000.0}},"relations":
[],"initialized":true,"forceRefresh":true,"name":"10.106.230.99.ICSCFSessionEstablishmentNetworkTable.s
copePrevious5SecondPeriod"}
```

Sample debugging (File Connector) output from collecting-0-0.log

```
INFO     -- [2019-04-10 03:11:24 EDT] -- CollectorManagerImpl::configure(): Parsing file
'/opt/DCF/Collecting/Collector-Manager/clearwater/conf/collecting.xml' ...
INFO     -- [2019-04-10 03:11:25 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.FileConnector for connector File...
INFO     -- [2019-04-10 03:11:25 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
connector File (FileConnector v1.0 rM) with /opt/DCF/Collecting/Collector-Manager/clearwater/conf/file-
connector.xml...
INFO     -- [2019-04-10 03:11:25 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.snmpCollector.SnmpCollectorSlave for collector CLEARWATER-
SNMP...
INFO     -- [2019-04-10 03:11:25 EDT] -- W4nHeavySnmpSession::<clinit>(): SnmpOidSessionTTL value set
```

```
to (ms): 600000
INFO     -- [2019-04-10 03:11:25 EDT] -- SnmpCollectorSlave::<clinit>(): Single oid polling session
time to live set to 600,000 ms.
INFO     -- [2019-04-10 03:11:25 EDT] -- SnmpCollectorSlave::<clinit>(): Table polling session time to
live set to 1,200,000 ms.
INFO     -- [2019-04-10 03:11:25 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
collector CLEARWATER-SNMP (SnmpCollectorSlave v6.8u2 rM) with /opt/DCF/Collecting/SNMP-
Collector/clearwater/conf/snmpcollector.xml...
INFO     -- [2019-04-10 03:11:25 EDT] -- SnmpCollectorSlave::configureLogLevel(): SNMP packet decoding
logging set to FATAL
INFO     -- [2019-04-10 03:11:25 EDT] -- CollectorManagerImpl::connect(): Connecting collectors...
INFO     -- [2019-04-10 03:11:25 EDT] -- AbstractCollector::registerNext(): Connecting CLEARWATER-SNMP
to Param-CLEANUP...
INFO     -- [2019-04-10 03:11:25 EDT] -- CollectorManagerImpl::start(): Starting Collector Manager
v5.10u1...
```

VMware vROPs Alerts collector 12

This chapter includes the following topics:

Overview

The vROPs Alerts SP is a DCF management pack used to pull the alerts from vROPs and publish the same to KAFKA for consumption.

Installation

vROPs Alerts SP (vrops-collect) installation is similar to any other SP installation and can be done through DCF module manager and also through controller.

Related package will be available in DCF module repository by default.

Example: `<DCF-Install>/Tools/Module-Repository/vrops-collect -1.0.pkg`

Installation command:

`<DCF-Install>/bin/manage-modules.sh install vrops-collect <Instance-Name (e.g. vrops-alerts)>`

Sample Installation questions and output :

```
Required dependencies, in processing order:
   [1]   java '8.0.202' v8.0.202
   [2]   module-manager '1.13u1' v1.13u1
   [3] I collector-manager 'vrops-alerts' (none) => v5.10u1
   [4] I failover-filter 'vrops-alerts' (none) => v5.5
   [5] I cross-referencing-filter 'vrops-alerts' (none) => v1.7u2
   [6] I kafka-connector 'vrops-alerts' (none) => v1.0u2
   [7] I variable-handling-filter 'vrops-alerts' (none) => v1.18u2
   [8] I stream-collector 'vrops-alerts' (none) => v1.4u2
```

```
   [9] I vrops-collect 'vrops-alerts' (none) => v1.0
> 2 not modified, 7 to install
> 34.3 MB space required / 17.1 GB available
 ? Enter the step to modify, 'yes' to accept them, or 'no' to cancel the operation [yes] >

Starting installation of Collector-Manager v5.10u1 from collector-manager-5.10u1-linux-x64...
 * Gathering information...
 * 'Collector-Manager v5.10u1' will be registered with instance name 'vrops-alerts'.
 * It will be installed in '/opt/DCF/Collecting/Collector-Manager/vrops-alerts'.
 * Unpacking files...
 * Installing files... 100%
 * 60 files have been installed.
 * Finalizing installation...
 * Installing service 'collector-manager vrops-alerts'...          [ installed ]
Installation complete.

Starting installation of FailOver-Filter v5.5 from failover-filter-5.5-linux-x64...
 * Gathering information...
 * 'FailOver-Filter v5.5' will be registered with instance name 'vrops-alerts'.
 * It will be installed in '/opt/DCF/Collecting/FailOver-Filter/vrops-alerts'.
 * Unpacking files...
 * Installing files... 100%
 * 4 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Cross-Referencing-Filter v1.7u2 from cross-referencing-filter-1.7u2-linux-
x64...
 * Gathering information...
 * 'Cross-Referencing-Filter v1.7u2' will be registered with instance name 'vrops-alerts'.
 * It will be installed in '/opt/DCF/Collecting/Cross-Referencing-Filter/vrops-alerts'.
 * Unpacking files...
 * Installing files... 100%
 * 10 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Kafka-Connector v1.0u2 from kafka-connector-1.0u2-linux-x64...
 * Gathering information...
 * 'Kafka-Connector v1.0u2' will be registered with instance name 'vrops-alerts'.
 * It will be installed in '/opt/DCF/Collecting/Kafka-Connector/vrops-alerts'.
 * Unpacking files...
 * Installing files... 100%
 * 17 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of Variable-Handling-Filter v1.18u2 from variable-handling-filter-1.18u2-linux-
x64...
 * Gathering information...
 * 'Variable-Handling-Filter v1.18u2' will be registered with instance name 'vrops-alerts'.
 * It will be installed in '/opt/DCF/Collecting/Variable-Handling-Filter/vrops-alerts'.
 * Unpacking files...
 * Installing files... 100%
 * 35 files have been installed.
```

```
 * Finalizing installation...
Installation complete.

Starting installation of Stream-Collector v1.4u2 from stream-collector-1.4u2-linux-x64...
 * Gathering information...
 * 'Stream-Collector v1.4u2' will be registered with instance name 'vrops-alerts'.
 * It will be installed in '/opt/DCF/Collecting/Stream-Collector/vrops-alerts'.
 * Unpacking files...
 * Installing files... 100%
 * 46 files have been installed.
 * Finalizing installation...
Installation complete.

Starting installation of vrops-collect v1.0 from vrops-collect-1.0...
 * Gathering information...
 * 'vrops-collect v1.0' will be registered with instance name 'vrops-alerts'.
 * It will be installed in '/opt/DCF/Block/vrops-collect/vrops-alerts'.
 * Unpacking files...
 * Installing files... 100%
 * 9 files have been installed.
 * Finalizing installation...
 ? Configure vROps  Server (yes/no) [y] > yes
 ? Hostname or IP address [localhost] > localhost
 ? Username [admin] > admin
 ? Password [•••••] >
 ? vROps Port [443] > 443
   [1] http
   [2] https
 ? Protocol [2] > 2
 ? More entries? (yes/no) [n] >
 ? Kafka Node Hostname or IP address [localhost] > localhost
 ? Kafka Node Port Number [9092] > 9092
 ? More entries? (yes/no) [n] >
 ? Kafka Topic name to publish Collected Data. [default-topic] > default-topic
 ? Kafka server authentication enabled? (SASL_PLAINTEXT only) (yes/no) [y] >
 ? Kafka server username. [admin] > admin
 ? Kafka server password. [•••••] >
 ? Do you want to start the installed services now? (yes/no) [n] > yes
 * Updating service 'collector-manager vrops-alerts'...              [ updated ]
 * Starting 'collector-manager vrops-alerts'...                          [ OK ]
Installation complete.
```

Configuration

SP related configuration files can be located under following directories:

<DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(vrops-alerts)/conf/

<DCF-Install>/ Collecting /Stream-Collector/<SP-Instance>/conf/

<DCF-Install>/ Collecting /Kafka-Connector/<SP-Instance>/conf/

<DCF-Install>/Collecting/Cross-Referencing-Filter/<SP-Instance>/conf/

<DCF-Install>/Collecting/Variable-Handling-Filter/<SP-Instance>/conf/

1 Collector Manager Configuration: A Collecting Manager Configuration file can be found at below location:

File: <DCF-Install>/Collecting/Collector-Manager/<SP-Instance>(vrops-alerts) /conf/collecting.xml

Sample content:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="http://www.watch4net.com/APG/Collecting"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.watch4net.com/APG/Collecting collecting.xsd ">
    <runOnce>false</runOnce>
    <connectors>
        <connector enabled="false" name="File" type="File-Connector"   config="conf/file-
connector.xml" />
        <connector enabled="true" name="Kafka" type="Kafka-Connector" config="Kafka-
Connector/vrops-alerts/conf/kafka-connector.xml" />
    </connectors>
    <filters>
        <filter enabled="true" name="VHF-vrops" next="Kafka" config="Variable-Handling-
Filter/vrops-alerts/conf/vhf-vrops.xml" />
        <filter enabled="true" name="CRF-vrops" next="VHF-vrops" config="Cross-Referencing-
Filter/vrops-alerts/conf/crf-vrops.xml" />
    </filters>
    <collectors>
        <collector enabled="true" name="VROPSAlerts" next="CRF-vrops" config="Stream-
Collector/vrops-alerts/conf/vrops-alerts.xml" />
    </collectors>
</config>
```

2 vROPs Alerts Configuration: vROPs Alerts Configuration file can be found in below location :

File: <DCF-Install>/Collecting/Stream-Collector/<SP-Instance>/conf/vrops-alerts.xml

Sample content:

```
<collector-configuration xmlns="http://www.watch4net.com/Text-Collector-Configuration"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.watch4net.com/Text-Collector-
Configuration ../textCollectorConfiguration.xsd ">
    <source>VROPS-Collector</source>
    <collecting-group>group</collecting-group>
    <default-character-encoding>UTF-8</default-character-encoding>
    <properties-refresh-periods>10m</properties-refresh-periods>
    <collecting-threads-pool-size>30</collecting-threads-pool-size>

        <collecting-configurations name="vrops-login-request">
        <include-contexts>conf/context-vrops.xml</include-contexts>
         <data-retrieval-file>conf/requests/vrops-requests.xml</data-retrieval-file>
            <data-listeners id="VROPS-TOKEN" variable-id="token">
                <values context-key="VROPS_HOST">
                  <name>VROPS-REQUEST-@{VROPS_HOST}</name>
```

```
                  <unit>code</unit>
                  <replace value=".+"   by="0" pattern="true"/>
                </values>
                <properties context-key="token" property-name="token"/>
             </data-listeners>

            <data-listeners id="VROPS-ALERTS" variable-id="AlertId" variable-id-separator="-">
                <values context-key="VROPS_HOST">
                  <name>VROPS-ALERTS-@{VROPS_HOST}</name>
                  <unit>code</unit>
                  <replace value=".+"   by="0" pattern="true"/>
                </values>
                <properties context-key="alertId" property-name="AlertId"/>
                <properties context-key="alertDefinitionName" property-name="Name"/>
                <properties context-key="alertDefinitionId" property-name="EventDisplayName"/>
                <properties context-key="resourceId" property-name="InstanceId"/>
                <properties context-key="status" property-name="Status"/>
                <properties context-key="alertLevel" property-name="Severity"/>
                <hardcoded-properties key="objType">alertDetail</hardcoded-properties>
                <hardcoded-properties key="ExternalSource">VROPS</hardcoded-properties>
                <hardcoded-properties key="Certification">CERTIFIED</hardcoded-properties>
                <hardcoded-properties key="context-name">@{VROPS_HOST}</hardcoded-properties>
             </data-listeners>
             <data-listeners id="VROPS-RESOURCES" variable-id="InstanceId" variable-id-
separator="-">
                <values context-key="VROPS_HOST">
                  <name>VROPS-RESOURCES-@{VROPS_HOST}</name>
                  <unit>code</unit>
                  <replace value=".+"   by="0" pattern="true"/>
                </values>
                <properties context-key="identifier" property-name="InstanceId"/>
                <properties context-key="resourceName" property-name="InstanceName"/>
                <properties context-key="adapterKindKey" property-name="adapterKindKey"/>
                <properties context-key="resourceKindKey" property-name="ClassName"/>
                <properties context-key="resourceKindKey" property-name="type"/>
                <hardcoded-properties key="objType">resourceDetail</hardcoded-properties>
                <hardcoded-properties key="ExternalSource">VROPS</hardcoded-properties>
             </data-listeners>

       </collecting-configurations>
</collector-configuration>
```

This is basic configuration of the SP which deals with how the processing should start.

3 vROPs Alerts REST API requests Configuration: The following example is a default vROPs Alerts requests configuration. This file will have configurations for polling.

File: `<DCF-Install>/Collecting/Stream-Collector/vrops-alerts/conf/requests/vrops-requests.xml`

Sample content:

```
<?xml version="1.0" encoding="UTF-8"?>
<data-retrieval-configuration xmlns="http://www.watch4net.com/Text-Parsing-Configuration"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.watch4net.com/Text-Parsing-
Configuration ../textParsingConfiguration.xsd" xmlns:xi="http://www.w3.org/2001/XInclude">
  <retrieving-period>900s</retrieving-period>
    <http-request data-retry="1" data-timeout="300s" connection-timeout="120s" character-
encoding="UTF-8">
         <lock name="perdevicelock-opst-a2-@{VROPS_HOST}" count="1"/>
               <disable-ssl/>
               <request-groups name="VROPS LOGIN">
                 <requests options="POST">
                    <url>@{protocol}://@{VROPS_HOST}:@{PORT}/suite-api/api/auth/token/acquire</url>
                    <headers key="Accept">application/json</headers>
                    <headers key="Content-Type">application/json</headers>
                    <post-content-file>conf/requests/data-post.json</post-content-file>
                 </requests>
               </request-groups>
                 <json-to-xml-transformer include-json-name="true">
                <xml-dataset>
                   <datasets>
                   <xpath>/W4N</xpath>
                    <xml-reader>
                        <extractions xpath-expression="concat('vRealizeOpsToken ',/W4N/token)"
result-type="string">token</extractions>
                        <http-request data-retry="1" data-timeout="300s" connection-timeout="120s"
character-encoding="UTF-8">
                           <disable-ssl/>
                           <request-groups name="RETRIEVE ALERTS">
                              <requests options="GET">
                               <url>@{protocol}://@{VROPS_HOST}:@{PORT}/suite-api/api/alerts</url>
                               <headers key="Content-Type">application/json</headers>
                               <headers key="Accept">application/json</headers>
                               <headers key="Authorization">@{token}</headers>
                              </requests>
                           </request-groups>
                           <json-to-xml-transformer>
                              <xml-dataset parse-datasets-in-parallel="true">
                                <datasets>
                                  <xpath>/W4N/alerts/OBJECT</xpath>
                                    <xml-reader>
                                      <extractions xpath-
expression="/OBJECT/alertId">alertId</extractions>
                                      <extractions xpath-expression="if (/OBJECT/alertLevel =
'CRITICAL') then '1' else if(/OBJECT/alertLevel = 'IMMEDIATE') then '2' else if
( /OBJECT/alertLevel = 'WARNING') then '3' else if ( /OBJECT/alertLevel = 'INFO') then '4' else
('5') " result-type="string">alertLevel</extractions>
                                      <extractions xpath-
expression="/OBJECT/resourceId">resourceId</extractions>
                                      <extractions xpath-expression="if (/OBJECT/status =
'ACTIVE') then 'ACTIVE' else ('INACTIVE') " result-type="string">status</extractions>
                                      <extractions xpath-
expression="/OBJECT/alertDefinitionId">alertDefinitionId</extractions>
```

```
                                    <extractions xpath-
expression="/OBJECT/alertDefinitionName">alertDefinitionName</extractions>
                                    <release id="VROPS-ALERTS"/>
                                    <http-request data-retry="1" data-timeout="300s" connection-
timeout="120s" character-encoding="UTF-8">
                                       <disable-ssl/>
                                       <request-groups name="RETRIEVE RESOURCES">
                                          <requests options="GET">
                                           <url>@{protocol}://@{VROPS_HOST}:@{PORT}/suite-
api/api/resources/@{resourceId}</url>
                                           <headers key="Content-Type">application/json</headers>
                                           <headers key="Accept">application/json</headers>
                                           <headers key="Authorization">@{token}</headers>
                                          </requests>
                                       </request-groups>
                                       <json-to-xml-transformer>
                                         <xml-dataset parse-datasets-in-parallel="true">
                                           <datasets>
                                           <xpath>/W4N</xpath>
                                           <xml-reader>
                                             <extractions xpath-
expression="/W4N/identifier">identifier</extractions>
                                             <extractions xpath-
expression="/W4N/resourceKey/name">resourceName</extractions>
                                             <extractions xpath-
expression="/W4N/resourceKey/adapterKindKey">adapterKindKey</extractions>
                                             <extractions xpath-
expression="if(/W4N/resourceKey/resourceKindKey = 'HostSystem') then 'Host' else
(/W4N/resourceKey/resourceKindKey)" result-type="string">resourceKindKey</extractions>
                                             <release id="VROPS-RESOURCES"/>
                                            </xml-reader>
                                           </datasets>
                                         </xml-dataset>
                                       </json-to-xml-transformer>
                                    </http-request>
                                 </xml-reader>
                              </datasets>
                           </xml-dataset>
                        </json-to-xml-transformer>
                     </http-request>
                  </xml-reader>
               </datasets>
            </xml-dataset>
         </json-to-xml-transformer>
    </http-request>
</data-retrieval-configuration>
```

4 Kafka Configuration: Kafka server configuration used to publish alerts data from vROPs to Kafka is below:

File : `<DCF-Install>/ Collecting/Kafka-Connector/<SP_Instance>/conf/kafka-connector.xml`

Sample content:

```
<?xml version="1.0" encoding="UTF-8"?>
<kafka-connector-config xmlns="http://www.watch4net.com/KafkaConnector"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.watch4net.com/KafkaConnector ../kafka-connector.xsd ">
        <kafka-producer-settings>
            <!-- Servers information -->
                <server host="localhost" port="9092"/>
            <!-- Producer (and topic) information -->
            <producer topic-name="default-topic"
                acks="1"
                retries="0"
                linger-ms="0ms"
                buffer-memory="33554432"
                compression-type="none"
                batch-size="16384"
                max-block-ms="5s"
                max-in-flight-requests-per-connection="5"
                max-request-size="1048576"/>

            <!-- All connection parameters will be attributes except "server" -->
            <connection request-timeout-ms="5s"
                connections-max-idle-ms="9m"
                retry-backoff-ms="100ms"
                reconnect-backoff-ms="50ms"/>

            <!-- Additional properties -->
            <additional-parameters key="metadata.max.age.ms">1000</additional-parameters>
            <additional-parameters key="receive.buffer.bytes">32768</additional-parameters>
            <additional-parameters key="send.buffer.bytes">131072</additional-parameters>

                <additional-parameters key="sasl.mechanism">PLAIN</additional-parameters>
        <additional-parameters key="security.protocol">SASL_PLAINTEXT</additional-parameters>
        <additional-parameters
key="sasl.jaas.config">org.apache.kafka.common.security.plain.PlainLoginModule required
username="admin" password="admin-secret";</additional-parameters>

        </kafka-producer-settings>

        <connector-component-behavior outputJson="true" flush-every-n-rawvalues="5000"/>

</kafka-connector-config>
```

Kafka Connector parameters

Parameters	Description
server	This tag must occur at least once. host: The address of one of the kafka bootstrap servers port: The port that the bootstrap server is listening on.
producer	This tag is optional, but may be used for templating kafka producers. Please refer to the schema for more information.

Parameters	Description
connection	This tag is optional, but may be used for templating kafka connections. Please refer to the schema for more information.
additional-parameters	This tag is optional, but may be used for configuring kafka options outside of the purview of the previous tags. These options include, but are not limited to SSL connection parameters.

Configuring Kafka Connector:

Parameters	Description
kafka-producer-settings	This tag may be used if we want to customize how the writer writes to kafka. Please refer to the schema for more information.
connector-component-behavior	This tag may be used if we want to control how often we flush to kafka.

5 vROPs Cross Referencing Filter Configuration: The following example is a default vROPs Alerts cross referencing filter configuration.

File: `<DCF-Install>/ Collecting/Cross-Referencing-Filter/vrops-alerts/conf/crf-vrops.xml`

Sample content:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.watch4net.com/CrossReferencingFilter"
xsi:schemaLocation="http://www.watch4net.com/CrossReferencingFilter/cross-referencing-filter.xsd">
    <blind-write>true</blind-write>
    <data-
accessor>com.watch4net.apg.v2.collector.plugins.crossreferencingfilter.data.mem.MemoryHashDataAcces
sor</data-accessor>
    <reference>
        <condition type="filter">ExternalSource=='VROPS'</condition>
        <match-on>InstanceId</match-on>
        <include-properties>true</include-properties>
        <get-property>InstanceId</get-property>
        <get-property>AlertId</get-property>
        <get-property>Name</get-property>
        <get-property>EventDisplayName</get-property>
        <get-property>Status</get-property>
        <get-property>Severity</get-property>
    </reference>
</configuration>
```

6 vROPs Variable Handling Filter Configuration: The following example is a default vROPs Alerts variable handling filter configuration.

File: <DCF-Install>/ Collecting/Variable-Handling-Filter/vrops-alerts/conf/vhf-vrops.xml

Sample content:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns="http://www.watch4net.com/VariableHandlingFilter"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.watch4net.com/VariableHandlingFilter variable-
handling-filter.xsd ">
        <handling-configuration id="FilterMetrics">
                <matcher
class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.matchers.APGFilterMatcher">
                        <parameter name="filter">!(objType=='resourceDetail')</parameter>
                </matcher>
                <handler
class="com.watch4net.apg.v2.collector.plugins.variablehandlingfilter.handlers.BlockAllHandler"/>
        </handling-configuration>
</configuration>
```

Logging

All logs will be found in below path.

<DCF-Install>/Collecting/Collector-Manager/<SP-Instance>/logs/collecting-0-0.log

Latest processing log name will have "-0-0" appended to its name like above.

To enable logging, change the collecting configuration to push metrics/topology data to file connector like below. And after changing the configuration, operator need to restart the service to see the logging info in collecting-0-0.log.

Edit below file as specified below.:

File : <DCF-Install>/Collecting/Collector-Manager/<SP-Instance>/conf/collecting.xml

```
---
<connector enabled="true" name="File" type="File-Connector"   config="conf/file-connector.xml" />

---
```

Example: "vrops-alerts" is installed management pack name in below example.

To check the status of the installed services:

```
<DCF-Install>/bin/manage-modules.sh service status all
….
* Checking collector-manager vrops-alerts'...       [ running ]
```

To restart the specific service:

```
<DCF-Install>/bin/manage-modules.sh service restart collector-manager vrops-alerts
 * Stopping collector-manager vrops-alerts'...            [ OK ]
 * Starting collector-manager vrops-alerts'...            [ OK ]
```

Sample output (JSON) of event in Kafka:

```
{
  "groupName" : "group",
  "discoveryID" : null,
  "jobID" : "9999",
  "type" : "VirtualMachine",
  "timestamp" : 1548910968,
  "value" : 0.0,
  "action" : "r",
  "properties" : {
    "Status" : "ACTIVE",
    "adapterKindKey" : "VMWARE",
    "InstanceId" : "2bdb8c22-6f54-4089-824b-e4e3551795e3",
    "context-name" : "10.234.24.66",
    "EventDisplayName" : "AlertDefinition-VMWARE-VMRunningOnSnapshotsForMoreThan2Days",
    "Severity" : "3",
    "source" : "VROPS-Collector",
    "type" : "VirtualMachine",
    "Certification" : "CERTIFIED",
    "Name" : "Virtual machine is running on snapshots for more than 2 days",
    "InstanceName" : "strs-vm-174.lss.emc.com",
    "AlertId" : "a7e66b66-a2b3-4875-835e-7012003345e1",
    "ExternalSource" : "VROPS",
    "ClassName" : "VirtualMachine",
    "objType" : "alertDetail"
  },
  "metrics" : {
    "VROPS-ALERTS-10.234.24.66" : {
      "properties" : {
        "unit" : "code",
        "name" : "VROPS-ALERTS-10.234.24.66"
      },
      "value" : 0.0
    }
  },
  "relations" : [ ],
  "initialized" : true,
  "forceRefresh" : true,
  "name" : "a7e66b66-a2b3-4875-835e-7012003345e1"
}
```

Sample debugging (File Connector) output from collecting-0-0.log

```
INFO     -- [2019-04-11 01:52:32 EDT] -- CollectorManagerImpl::configure(): Parsing file
'/opt/DCF/Collecting/Collector-Manager/vrops-alerts/conf/collecting.xml' ...
INFO     -- [2019-04-11 01:52:33 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.FileConnector for connector File...
INFO     -- [2019-04-11 01:52:33 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
connector File (FileConnector v1.0 rM) with /opt/DCF/Collecting/Collector-Manager/vrops-
alerts/conf/file-connector.xml...
INFO     -- [2019-04-11 01:52:34 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.kafka.connector.KafkaConnector for connector Kafka...
INFO     -- [2019-04-11 01:52:34 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
connector Kafka (KafkaConnector v1.0u2 rM) with /opt/DCF/Collecting/Kafka-Connector/vrops-
alerts/conf/kafka-connector.xml...
INFO     -- [2019-04-11 01:52:34 EDT] -- KafkaConnector::setRecordMetadataGetTimeoutMs(): Set value of
"kafkaconnector.requestfuture.timeout.ms" to: 60,000
INFO     -- [2019-04-11 01:52:34 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.VariableHandlingFilter for filter VHF-vrops...
INFO     -- [2019-04-11 01:52:34 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
filter VHF-vrops (VariableHandlingFilter v1.18u2 rM) with /opt/DCF/Collecting/Variable-Handling-
Filter/vrops-alerts/conf/vhf-vrops.xml...
INFO     -- [2019-04-11 01:52:35 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.v2.collector.plugins.CrossReferencingFilter for filter CRF-vrops...
INFO     -- [2019-04-11 01:52:35 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
filter CRF-vrops (CrossReferencingFilter v1.7u2 rM) with /opt/DCF/Collecting/Cross-Referencing-
Filter/vrops-alerts/conf/crf-vrops.xml...
INFO     -- [2019-04-11 01:52:35 EDT] -- CollectorManagerImpl::connect(): Connecting filters...
INFO     -- [2019-04-11 01:52:35 EDT] -- AbstractCollector::registerNext(): Connecting VHF-vrops to
File...
INFO     -- [2019-04-11 01:52:35 EDT] -- AbstractCollector::registerNext(): Connecting CRF-vrops to
VHF-vrops...
INFO     -- [2019-04-11 01:52:35 EDT] -- CollectorManagerImpl::getComponentInstance(): Instantiating
class com.watch4net.apg.ubertext.collector.StreamCollector for collector VROPSAlerts...
INFO     -- [2019-04-11 01:52:35 EDT] -- CollectorManagerImpl::getComponentInstance(): Configuring
collector VROPSAlerts (StreamCollector v1.4u2 rM) with /opt/DCF/Collecting/Stream-Collector/vrops-
alerts/conf/vrops-alerts.xml...
INFO     -- [2019-04-11 01:52:40 EDT] -- CollectorManagerImpl::connect(): Connecting collectors...
INFO     -- [2019-04-11 01:52:40 EDT] -- AbstractCollector::registerNext(): Connecting VROPSAlerts to
CRF-vrops...
INFO     -- [2019-04-11 01:52:40 EDT] -- CollectorManagerImpl::start(): Starting Collector Manager
v5.10u1...
INFO     -- [2019-04-11 01:52:40 EDT] -- CollectorManagerImpl::start(): Initializing components...
```