

Data Collection Guideline for vMRF

Virtual Multimedia Resource Function

Operating Instructions

Copyright

© Ericsson AB 2016–2019. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document [Trademark Information](#).

Contents

1	Introduction	1
2	Collect Data for vMRF Troubleshooting	2
3	Overview of Data Collection	3





1 Introduction

The purpose of this document is to instruct how to collect troubleshooting data that is to be enclosed in a Customer Service Request (CSR) in case a problem is experienced with the Virtual Multimedia Resource Function (vMRF).



2 Collect Data for vMRF Troubleshooting

This section describes how to collect all troubleshooting data with the `collectData.py` script. Other possible options for the script are described in [Overview of Data Collection](#) on page 3.

Prerequisites

- An incident has happened that requires opening a CSR.
- There is enough free disk space available in the `/tmp` and `cluster/storage` directories, that is, more than 500 MB free disk space, or more than 50% of total disk space capacity of the disk partition.

Steps

1. Open an SSH connection to the Operation And Maintenance (O&M) IP address of the Virtual Multimedia Resource Function (vMRF) Virtualized Network Function (VNF) instance using the following command:

```
ssh <user_ID>@<O&M_IP_address>
```

Note: This command opens an SSH connection to the active SC VM, where data collection is performed. It is not recommended to log into a PL VM to collect troubleshooting data.

2. Run the following command:

```
collectData.py -a
```

Note: It is recommended to run `collectData.py -a` option to collect all available troubleshooting data. The reason is that the script runs for a reasonable amount of time even with all options enabled, and collecting all information maximizes the help given for troubleshooters. with the

`dcdgm` and `getinfo.sh` are aliases to `collectData.py -a`, and running them provides the same result.

3. Attach the resulting `tar.gz` file to the CSR. The `.tar.gz` file is located in `cluster/storage/collectdata/<date_time>/`.



3 Overview of Data Collection

To collect troubleshooting data, the `collectData.py` script is used. The script collects troubleshooting information as specified by the options used when running the script.

Each command execution, except for options `-h` and `-v`, generates a `tar.gz` file containing the requested information from the VNF in the following output format:

```
<hostname>_<machine_timestamp>_collectdata.tar.gz
```

The collected data can be analyzed after unpacking the `tar.gz` file located in `/cluster/storage/collectdata/<date_time>/`. To unpack the collected data, run the following commands:

- `gunzip <hostname>_<machine_timestamp>_collectdata.tar.gz`
- `tar xvf <hostname>_<machine_timestamp>_collectdata.tar`

The script accepts any of the following options, and it is also possible to combine multiple options (for example, `collectData.py -bcd1`):

- | | |
|-----------------------|---|
| -h, --help | This option shows the help message without collecting any data. It also lists the commands run by other options. For further information on Command Line Interface (CLI) tool commands, see CLI Commands . |
| -a, --all | <p>This option collects all of the troubleshooting information, that is, Performance Management (PM) data, system logs, and the result of CLI tool and system commands. A detailed description of data collected is provided in the description of options below.</p> <p>Note: Binaries are not collected with this option due to their large size. However, <code>-b</code> can be combined with <code>-a</code> when needed.</p> |
| -b, --binaries | This option collects all the binaries under the <code>opt/mrf_agent</code> , <code>opt/mrf_director</code> , and <code>opt/ip_pipeline</code> directories needed, for example, for debugging crash dumps. |
| -c, --commands | This option automatically runs a set of CLI tool and system commands and collects data in the <code>commands.log</code> file in each VM. The list of commands run is printed out when using this option. |



Note: The CLI and system commands are executed to all the VMs within the cluster by using the `cluster run` option.

Option `-c` collects data by running the following commands:

— CLI tool commands:

- `mrf_appl status`
- `mrf_appl internals`
- `mrf_appl compute-resource`
- `mrf_appl overload-control`
- `mrf_appl context-info`
- `mrf_appl sctp-status`
- `mrf_appl h248-counters -t`
- `mrf_appl sctp-pm-counters`
- `mrf_appl service-pm-counters`
- `mrf_appl announcement-status -s`
- `mrf_appl cache-test`
- `mrf_appl license-info`
- `mrf_appl announcement-counters`
- `mrf_appl h248interface-counters`
- `mrf_appl smms-counters`
- `ipp conf`
- `ipp pm-counters`
- `ipp debug-counters`
- `ipp error-counters`
- `ipp discard -counters`
- `ipp signal -counters`
- `ipp dpdk-counters -m`
- `ipp ethdev -counters -x`



- `ipp mpd -usage`
- `ipp neigh`
- `mpd internals`
- System commands:
 - `verify_mrfv_node_status.py`
 - `cluster list`
 - `cat /usr/share/image/package_build_data`
 - `ip -a`
 - `free -m`
 - `ls -ltr /cluster/storage/dumps`
 - `dmesg -T`
 - `timedatectl status`
 - `cat /etc/systemd/timesyncd.conf`
 - `df -ha`
 - `ps eLo psr,pid,ppid,tid,pcpu,rss,time,comm,cmd`
 - `uptime`
 - `last reboot`
 - `mountpoint /media/cluster_sync`
 - `/bin/systemctl status --all`
 - `tipc link list`
 - `tipc link statistics show`
 - `cat /etc/issue`
 - `cat /proc/cpuinfo`

-d, --data

This option collects the following information:

- CLISS configuration

Note: The output of the CLISS command `show -r -v` is collected in file `clissConfig.log`.

- Performance Monitoring data



Performance Monitoring data is generated periodically after Report Output Periods (ROP files). ROP files are stored at `/cluster/storage/no-backup/com-apr9010443/PerformanceManagementReportFiles/`.

— Alarm and alert logs

These files are stored in the saflog file at `/cluster/storage/no-backup/coremw/var/log/saflog`.

COM logs are stored at `/cluster/storage/clear/com-apr9010443`.

COM SA (Support Agent) traces are stored at `/storage/clear/comsa_for_coremw-apr9010555`.

Middleware OpenSaf logs are stored at `/var/log/opensaf`.

— Crash dump files

These files are stored at `/cluster/storage/dumps/`.

— Network configuration data

— Cloud init data

The following data is collected depending on the deployment environment:

- VMware:
 - `ovf-env.xml`
- OpenStack:
 - `vendor_data.json`
 - `user_data`
 - `network_data.json`
 - `meta_data.json`

Note: Cloud init data is fetched only for users with root privileges.

-l, --logs

This option collects system logs from the `journalctl` command.



The output of collected system logs is stored into a `journal_messages` file in the corresponding data collection folder.

-n, --node

This option collects troubleshooting information only in a vMRF VNF specified by its IP address. The option must be combined with at least one data-collecting option as shown below:

`collectData.py -n <VNF_IP address> -<option(s)>`.

For example, **`collectData.py -n <VNF_IP address> -lb`** collects system logs and binaries on the specified vMRF VNF.

-o, --output

This option specifies the output directory.

-s, --show-results

This option prints the summary of data collection to the screen. The option must be combined with other data collection options.

-v, --verbose

This option prints the output of several CLI tool and system commands, that is, in the current VM only. No `tar .gz` file is generated, only an on-screen printout. If combined with other options, `-v` ignores them, and the script runs as if `-v` was the only option.

The option collects the same data as option `-c`, in addition to the following system command:

— `mpstat -P ALL 5 1`