

LEM Error Dump and Log User Guide

USER GUIDE

Copyright

© Ericsson AB 2014, 2015. 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	Overview	1
2	Configuring CDCLSV	3
2.1	Custom Packers	3
3	Logging	7
3.1	Log Sizes	8
4	Crash Dumps	11
4.1	Packer Objects for LEM	11
4.2	Collection	12
4.3	Limiting the Number of Stored Dumps	13
5	Commands	15
5.1	CDCLSV Configuration Related Commands	15
5.2	Log Related Commands	15
5.3	Packer Related Commands	16
	Reference List	21





1 Overview

This User Guide describes how to configure and use **Crash Dump and Console Log Collection Service** (CDCLSv) for logging and crash dump handling in LEM. CDCLSv collects component-specific console logs when a Linux core dump is generated. The list of files and directories to be collected and the log file size and the number of kept files are configurable.

CDCLSv is a service that is used for the following tasks:

- Handling CDCLSv, CDSv, CLUSv, ESHSv, and SCSv component-specific console logs and rotate them when a configurable file size is reached. For further details, refer to *LEM System Architecture Description*, Reference [1].
- Collecting the crash dumps created by Linux for the Core MW components of LEM together with all the generated log files. Placing these dumps and log files in a time-stamped and compressed package.



2 Configuring CDCLSV

CDCLSV comes with a default configuration that, in general, suits the LEM services needs. If any configuration changes are to be done, refer to *LEM Parameter Description*.

2.1 Custom Packers

CDCLSV supports creating custom packers. This function is also used by the LEM and vDicos to collect logs when a LEM or vDicos process crashed and dump is generated.

The packers can be defined through the IMM with `CdclsPacker` and `CdclsPackerOption` classes, see Figure 1.

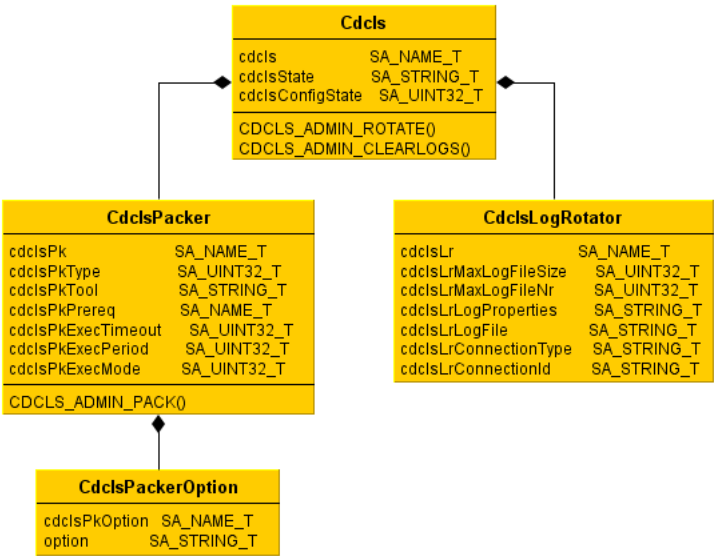


Figure 1 *CdclsPacker and CdclsPackerOption*

The main idea is that the user can configure a command in the packer object that is called periodically with the options that are defined in the packer option object. That user-defined script can determine what to do. For example, in vDicos the packer script checks some predefined directory where dump files can be created, and if new dump files are found, then it starts the packing procedure.



2.1.1 CdclsPacker Object

There are two types of the packers: master and slave. Only the master packer scripts are called periodically. The master can have slave packers that are called before the master is called. The master script is only called when all its slave scripts exited, see Table 1.

Table 1 The CdclsPacker Object

Attribute	Description
cdclsPk	This configuration attribute denotes the name of the <i>Packer</i> . The generated package would also hold this name together with the creation time stamp.
cdclsPkType	Specifies the packer type. It can take one of the following values: <ul style="list-style-type: none">• 1: The packer is a "<i>Master</i>" packer. Such a packer can execute as prerequisites "<i>Slave</i>" packers.• 2: The packer is a "<i>Slave</i>" packer. Such a packer is always executed as a prerequisite for "<i>Master</i>" packers. For such a packer, the <code>cdclsPkExecPeriod</code> and <code>cdclsPkExecTimeout</code> attribute values have no effect.
cdclsPkTool	Specifies the path to the tool that is to be executed as part of the packing activity.
cdclsPkPrereq	Valid for " <i>Master</i> " packers only. It is used to list the ID (RDN) of those " <i>Slave</i> " packers that should be executed as prerequisite for this packing job.
cdclsPkExecTimeout	Valid for " <i>Master</i> " packers only. Specifies the time period a <i>Packer</i> is to wait for the invoked <i>tool</i> to conclude. If the invoked <i>tool</i> does not conclude in the specified time interval, it gets killed by the <i>Packer</i> .



cdclsPkExecPeriod	Valid for "Master" packers only. Specifies the periodic time interval the <i>tool</i> (see <code>cdclsPkTool</code>) is to be invoked within. If the <i>tool</i> execution does not conclude in this time interval, the <i>Packer</i> waits for it (maximum <code>cdclsPkExecTimeout</code> period) before invoking the <i>tool</i> for the next iteration. The value set for <code>cdclsPkExecPeriod</code> is always a multiple of 5 sec .
cdclsPkExecMode	Specifies whether a packing activity is to be executed on unique or on shared load basis. Can take one of the following values: <ul style="list-style-type: none"> • 1: The packer is central. It is executed on Director level. • 2: The packer is shared. It is executed on Agent level.

2.1.2 CdclsPackerOption Object

When a packer is executed by the CDCLSV, the arguments come from these objects. There is one special `cdclsPkOption` that is not used as an argument: if the `cdclsPkOption` is "dumpprefix", then the value given in the option is used as prefix of the name of the packing activity.

This class is child of the `CdclsPacker` class, see Table 2.

Table 2 The *CdclsPackerOption* Object

Attribute	Description
cdclsPkOption	This configuration attribute denotes the name of the <i>PackerOption</i> .
option	Specifies a list of attributes and values the <i>tool</i> specified under <code>cdclsPkTool</code> is to be invoked with. Some special substrings can be used that is to be replaced runtime: <ul style="list-style-type: none"> • [HOSTNAME] Host name where the script is running. • [NODENAME] CMW node name where the script is running.

2.1.3 Running a Packer

When the execution period expires of a packer, then the `CdclsvDirector` starts the packing procedure.



1. Determine the name of the packing activity: `<packer_name>-<timestamp>`.

The name comes from the option attribute of the `CdclsPackerOption` object, where the `cdclsPkOption` is “dumpprefix”. If there is no such object, then the `cdclsPk` attribute of the `CdclsPacker` object is used as name. If manual packing was triggered by calling the `cdclsv-pack` script, then the caller can specify this name through an argument, see Section 5.3 on page 16.

The time stamp is the current time when the activity starts.

2. Check if the packer has prerequisite packer. If so, then call that. The arguments from the packer option for the slave packer are part of the executed command. **Important:** the first two arguments are always `-d <packing_activity_name>`.
3. When all scripts belong to the prerequisite are exited, then the master script is called. The arguments from the packer option for the master packer are part of the executed command. **Important:** the first two arguments are always `-d <packing_activity_name>`.

There can be three ways how a packing activity ends in normal case:

1. Time-out in the slave packers.
2. Time-out in the master packer.
3. Master script ends in time.

When the activity does not conclude in time, that is described in the `cdclsPkExecTimeout` value of the packer object, an alarm is raised. The alarm stops the activity and kills the command if started the execution.

The master scripts exit the return code and are analyzed by the `CdclsvDirector`. Zero means that the script exited successfully and it did packing activity. In this case, if there was alarm raised before, it is cleared. On other return code, the alarm is untouched.

2.1.4 Running The Packer By The User

The user can also perform a packing activity by running the `cdclsv-pack` `<packer>` command. In this case an additional parameter is passed to the packer scripts: `-force`.



3 Logging

All the software components of the vDicos solution log, by using CDCLSV, into a dedicated directory on the service control nodes. This directory can be reached on each node in the cluster through the following link: “/opt/cdclsv/storage/log/”. The log filenames, created under this directory, follow the next template:

```
<process_name>-<log_source><-instance_id><-timestamp>
```

Where:

- *<process_name>*: The name of the process or AMF component writing logs in the log file.
- *<log_source>*: The identity of the node, where the process or AMF component generating the log is running on (the source of the log file).
- *<-instance_id>*: An optional tag that appears if the running component instance number on the same node is more than one.
- *<-timestamp>*: This information is displayed only for rotated log files and represents the file creation time stamp. For a log file that has not yet been rotated, the *<-timestamp>* tag is blank (for example: *<process_name>-<log_source>*).

The *<process_name>* tag of a log file can take the following names in LEM:

- CDCLSV:
 - *cdclsvdirector*: Contains the output of the **CdclsvDirector** component.
 - *cdclsvagent*: Contains the output of the **CdclsvAgent** and **CdclsvLogRotator** components.
- CDSv:
 - *cdsvdirector*: Contains the output of the **CdsvDirector** component.
- CLUSv:
 - *cludirector*: Contains the output of the **CluDirector** component.
- ESHSV:
 - *eshsv*: Contains the output of the **Eshsv** component.
- SCSv:
 - *scsvcollector*: Contains the output of the **ScsvCollector** component.



- `scsvstreamer`: Contains the output of the **ScsvStreamer** component.

The default configuration for these log files is the following:

- `cdclsLrMaxLogFileSize`: 52428800. The maximum log file size is 50 MB. If this size is exceeded, the file is rotated. That is, a new log file is created and the old one is closed and preserved until the number of preserved log files reach a certain threshold.
- `cdclsLrMaxLogFileNr`: 2. The maximum number of rotated files preserved by CDCLSV for a certain logger source.

To change both the number and size of log files for all directors, agents, vms, and so on, modify the attributes of log rotators in IMM, as shown in Example 1. In this example, one log file is set with a size of 30 MB for the CDSv director.

```
immllist cdclsLr=cdsvdirector,cdcls=CDCLSVSite
immcfg -a cdclsLrMaxLogFileNr=1 cdclsLr=cdsvdirector,cdcls=CDCLSVSite
immcfg -a cdclsLrMaxLogFileSize=31457280 cdclsLr=cdsvdirector,cdcls=CDCLSVSite
immllist cdclsLr=cdsvdirector,cdcls=CDCLSVSite
```

3.1 Log Sizes

Though it is possible to configure both the number and size of log files, most of the system runs with the default values. On a fully installed system (with LEM, DBS, and vDicos), the number of logs can reach a high amount, and they can easily populate the available disk space.

To examine the space required for the logs, do the following:

1. Get the number of the nodes and blades (`NODENUM`). It is assumed that all components are running on these nodes.
2. Get all of the log rotators:

```
immfind -c CdclsLogRotator
```

3. Go through all the log rotator objects and get the `cdclsLrMaxLogFileSize` and the `cdclsLrMaxLogFileNr` attributes.
4. Use the following formula to determine the maximum amount of space, in bytes, that the log rotator can log:

$$\text{cdclsLrMaxLogFileSize} \times (\text{cdclsLrMaxLogFileNr} + 1) \times \text{NODENUM}$$

Summarize these numbers for all log rotators to get the maximum size of the logs.

Example 1 shows a script that can give an estimation on the following:

- The size of the logs without rotation (calculating only with the current log files).



- The maximum size of the logs, including the rotated files.
- If `-c` is used as argument, then the current log size (if available).

```
#!/bin/bash

MINSIZE=0
MAXSIZE=0
CURSIZE=0
PRINTCUR=0

if [ "$1" == "-c" ]
then
    PRINTCUR=1
fi

NODENUM=`immfind | grep ^safSu= | grep CdclsVAgent | wc -l`

for LR in `immfind -c CdclsLogRotator`
do
    LR_SIZE=`immlist ${LR} -a cdclsLrMaxLogFileSize | cut -d = -f 2-`
    LR_NUM=`immlist ${LR} -a cdclsLrMaxLogFileNr | cut -d = -f 2-`
    LR_MSG=`immlist ${LR} -a cdclsLrLogProperties | cut -d = -f 2-`
    if [ ${LR_MSG} != "msgwriter" ]
    then
        MINSIZE=$((MINSIZE + LR_SIZE))
        MAXSIZE=$((MAXSIZE + ( LR_SIZE * ( LR_NUM + 1 ) )) )
        if [ $PRINTCUR -eq 1 ]
        then
            LR_FILES=`immlist ${LR} -a cdclsLrLogFile | cut -d = -f 2-`
            for FS in `du -sb ${LR_FILES}* 2> /dev/null | cut -f1`
            do
                CURSIZE=$((CURSIZE + FS))
            done
        fi
    fi
done
echo "Max log size without rotation: ${MINSIZE} / node, =>
      sum: $((MINSIZE * NODENUM)) with ${NODENUM} nodes"
echo "Max log size: ${MAXSIZE} / node, =>
      sum: $((MAXSIZE * NODENUM)) with ${NODENUM} nodes"
if [ $PRINTCUR -eq 1 ]
then
    echo "Current log size: ${CURSIZE}"
fi
```

Example 1 Log Size Estimation Script

For more accurate calculation, determine the exact node number (NODENUM) for each log rotator object.

To keep the size of the logs within a reasonable limit, configure the `cdclsSizeLimit`. That parameter tries to keep the log size below the limit by removing the old rotated files. For more information, refer to *LEM Parameter Description*.

If the `cdclsSizeLimit` is set below the maximum log size without rotation, and if the size of all log files is near to `cdclsLrMaxLogFileSize`, and if there are no more rotated log files, then the size of the log can go over the limit without the file getting deleted. Only rotated files are deleted if the size of the log is above limit.

`cdclsSizeLimit` can be set below that value, because the components can log in different quantity. If one component produces larger logs and other components produce smaller logs (or none at all), then the rotated log files of



the component with the larger logs get deleted. This way the size of the log files can be kept below the limit.



4 Crash Dumps

The LEM applications are Core MW processes. When a process of LEM crashes, a Linux core dump is generated by the Linux system. These dumps are generated in the `/cluster/dumps` directory.

4.1 Packer Objects for LEM

The `CdclsPacker` object tells the CDCLSV what to do with a crash file. The CDCLSV implementation comes with a default *Packer task* implementation that effectuates the following:

- Checks the apparition of a file or collection of files on the file system matching a predefined file type template. The template description is configurable through the `CdclsPkOption` child object of the packer.
- Upon fulfillment of their trigger condition, for example, crash dump apparition, it collects the CDCLSV logs and LEM-related crash dumps. It then packs them into a time-stamped `*.tgz` file. Backtraces for the LEM-related crash dumps are generated as well.

The `CdclsPacker` objects for the LEM components are configured, as shown in the following example, where two packer objects are created at installation of the CDCLSV.

```
cdclsPk=vDicos,cdcls=CDCLSVSite
```

Name	Type	Value(s)
cdclsPkType	SA_UINT32_T	1 (0x1)
cdclsPkTool	SA_STRING_T	/opt/cdclsv/bin/cdclsv_packer.sh
cdclsPkPrereq	SA_NAME_T	vDicosPrereq (12)
cdclsPkExecTimeout	SA_UINT32_T	295 (0x127)
cdclsPkExecPeriod	SA_UINT32_T	60 (0x3c)
cdclsPkExecMode	SA_UINT32_T	1 (0x1)
cdclsPk	SA_NAME_T	cdclsPk=vDicos (14)
SaImmAttrImplementerName	SA_STRING_T	CdclsPackerImplementer
SaImmAttrClassName	SA_STRING_T	CdclsPacker
SaImmAttrAdminOwnerName	SA_STRING_T	<Empty>

```
cdclsPk=vDicosPrereq,cdcls=CDCLSVSite
```

Name	Type	Value(s)
cdclsPkType	SA_UINT32_T	2 (0x2)
cdclsPkTool	SA_STRING_T	/opt/cdclsv/bin/cdclsv_prereq_packer.sh
cdclsPkPrereq	SA_NAME_T	<Empty>
cdclsPkExecTimeout	SA_UINT32_T	295 (0x127)
cdclsPkExecPeriod	SA_UINT32_T	60 (0x3c)
cdclsPkExecMode	SA_UINT32_T	2 (0x2)
cdclsPk	SA_NAME_T	cdclsPk=vDicosPrereq (20)
SaImmAttrImplementerName	SA_STRING_T	CdclsPackerImplementer
SaImmAttrClassName	SA_STRING_T	CdclsPacker
SaImmAttrAdminOwnerName	SA_STRING_T	<Empty>

The first object is the master (`cdclsPkExecMode` attribute is one), and the second is the slave (`cdclsPkExecMode` attribute is two).



These predefined packers are also configurable through the `CdclsPkOption` objects. Each component has its own `CdclsPkOption` object to determine the core dump files that trigger a collection of a dump through the option attribute. This attribute is the argument of the `cdclsPkTool` command in the `CdclsPacker` object.

An example of the CDSv:

```
cdclsPkOption=cdsOptions,cdclsPk=vDicos,cdcls=CDCLSVSite
```

Name	Type	Value(s)
option	SA_STRING_T	-w /cluster/dumps/Cdsv* -c /opt/cdsv/storage/*
cdclsPkOption	SA_NAME_T	cdclsPkOption=cdsOptions (25)
SaImmAttrImplementerName	SA_STRING_T	CdclsPackerOptionImplementer
SaImmAttrClassName	SA_STRING_T	CdclsPackerOption
SaImmAttrAdminOwnerName	SA_STRING_T	<Empty>

4.2 Collection

In every minute, the CDCLSV checks if a crash occurs (`cdclsPkExecPeriod` attribute of the packer object). If so, then it collects the logs in a time-stamped, compressed file in the `/opt/cdclsv/storage/dumps/` directory. The `cdclsv_packer.sh` script collects the logs and dumps if a file appears with a name that matches at least one of the filters in the watch list (`-w` or `-wi` parameter). Before collection, it runs the slave packers, defined in the `cdclsPrereq`. The `cdclsv_prereq_packer.sh` script also matches the filenames. It differentiates the following two kinds of files:

- l** For Linux core dumps. The `gdb` command is used to analyze them.
- spec** For special dump files that require a script as first parameter, which can analyze the specific dump files.

The analyzed files and the output of the analysis is placed in the dump directory that gets compressed. The master packer also watches the file in this directory so it collects the analyzed files.

A manual way is available to trigger the collection of the logs on to a compressed package, as described in Section 5 on page 15.

4.2.1 Structure of the Collected Compressed File

The content of the compressed file depends on the packer configuration by the `CdclsPackerOption` objects. By default, it contains the following information:

- `/opt/cdclsv/storage/log/`: *Log* of the LEM components.
- `/var/log/<node_name>/messages`: *Messages* files from all nodes.



- `/var/log/<node_name>/messages.1`: *First rotated messages* files from all nodes.
- `/opt/cdsv/storage/`: CdsV configuration of the cluster.

4.3 Limiting the Number of Stored Dumps

The default value for the number of stored dumps is 50. To change this limit, set attribute `-k`, as shown in the following example.

```
immcfg -a option="-k 49 -w /cluster/dumps/CdclsV* -wi /opt/cdclsV/storage/autocc/* \
-c /opt/cdclsV/storage/log -c /var/log/*/messages -c /var/log/*/messages.1 \
-c /cluster/storage/no-backup/coremw/var/log/saflog/vdicos/vdlog/*" \
cdclsPkOption=baseCdclsVOptions,cdclsPk=vDicos,cdcls=CDCLSVSite
```

Setting this attribute to a too low value (less than 10) can be disadvantageous. It is possible that the logs of a trigger event that caused a chain of crashes later can be lost.

Note: This method is deprecated by the new `CdclsConfigAttribute` called `maxNumOfDumps`.





5 Commands

This section briefly describes the most useful commands to control the CDCLSV.

5.1 CDCLSV Configuration Related Commands

This section describes commands that are in relation with CDCLSV configuration.

cdclsv-cfgattr-create

Creates a `CdclsConfigAttribute` object with the given name and value.

Synopsis: `cdclsv-cfgattr-create <parameter_name> <parameter_value>`

Example: creating the `cdclsSizeLimit` parameter and giving it 1G as value:

```
cdclsv-cfgattr-create cdclsSizeLimit 1G
```

cdclsv-cfgattr-get

Prints the current value of a given attribute.

Synopsis: `cdclsv-cfgattr-get <parameter_name>`

cdclsv-cfgattr-list

Lists the already created attributes.

Synopsis: `cdclsv-cfgattr-list`

cdclsv-cfgattr-set

Sets a new value for an already created attribute.

Synopsis: `cdclsv-cfgattr-set <parameter_name> <parameter_value>`

cdclsv-cfgattr-unset

Removes an existing attribute object.

Synopsis: `cdclsv-cfgattr-unset <parameter_name>`

5.2 Log Related Commands

This section describes commands that are in relation with logs.

**cdclsv-add-marker**

Adds text message to all log files.

Synopsis: `cdclsv-add-marker [<text message>]`

cdclsv-clear

Clears all log files.

Synopsis: `cdclsv-clear [--force] [--reset-only] [--help]`

Options:

- *--force*: Clears buffers also in the memory. Logs in the buffer of the director waiting for write. Logs in the buffer of log rotators waiting for send to the director.
- *--reset-only*: If you use this option, the command does not remove files, only reset some internal data that is used for communication between director and rotators.
- *--help*: Displays help.

cdclsv-rotate

Rotates all log files manually.

5.3 Packer Related Commands

This section describes commands that are in relation with packers:

cdclsv-list-packers

Lists packer objects in the IMM. The class of packer object is `CdclsPacker`.

cdclsv-get-pack-timeout

Gets the time-out value of a specific packer.

Synopsis: `cdclsv-get-pack-timeout [<packer_object_DN> | -h]`

Gets time-out of `<packer_object_DN>` in seconds. `<packer_object_DN>` is optional, default value is `cdclsPk=vDicos,cdcls=CDCLSVSite`.



cdclsv-set-pack-timeout

Sets the time-out value of a specific packer.

Synopsis: `cdclsv-set-pack-timeout [-h] |`
`[<packer_object_DN>] [<timeout_in_seconds>]`

Sets time-out of `<packer_object_DN>` to `<timeout_in_seconds>`. The `<packer_object_DN>` is optional, the default value is `cdclSPk=vDicos,cdcls=CDCLSVSite`. `<timeout_in_seconds>` is optional, the default value is 295.

cdclsv-get-pack-period

Gets the period of a specific packer.

Synopsis: `cdclsv-get-pack-period`
`[<packer_object_DN>|-h]`

Gets period of `<packer_object_DN>` in seconds. `<packer_object_DN>` is optional, default value is `cdclSPk=vDicos,cdcls=CDCLSVSite`.

cdclsv-set-pack-period

Sets the period of a specific packer.

Synopsis: `cdclsv-set-pack-period [-h] |`
`[<packer_object_DN>] [<period_in_seconds>]`

Sets period of `<packer_object_DN>` to `<period_in_seconds>`. The `<packer_object_DN>` is optional, the default value is `cdclSPk=vDicos,cdcls=CDCLSVSite`. `<period_in_seconds>` is optional, the default value is 60. The period must be a multiple of five. It can also be 0 meaning that the packer is turned off and the tool is not called.

**cdclsv-pack**

Executes a packer activity.

Synopsis: `cdclsv-pack [-h] [<packer_object_DN>|<packer_object_name>] [--prefix|-p|--name|-n <dump_file_name_prefix>]`

Executes a packer activity on <packer_object_DN>. This results in executing the command given in the `cdclsPkTool` attribute of the <packer_object_DN> with the proper options collected from `CdclsPackerOption` objects. <packer_object_DN> is optional, the default value is `cdclsPk=vDicos,cdcls=CDCLSVSite`.

If the <dump_file_name_prefix> argument is given, then the name of packing activity contains this name as a prefix instead of the name of the packer or the name that was defined in the `CdclsPackerOption`, see Section 2.1.3 on page 5.

cdclsv-pack-status

Prints the status of a packing activity.

Synopsis: `cdclsv-pack-status [<packer_object_name>|<packer_object_DN>|-a|--all|-am|--all-masters]`

Options:

- `-a, --all`: Prints status for all packers, including the prereq packers.
- `-am, --all-masters`: Prints status for all non-prereq and master packers.

If no option is given, the command prints the state of the `cdclsPk=vDicos,cdcls=CDCLSVSite` packer by default.

The following commands are available from within the COM CLI:

cdclsv-list

Lists the Distinguished Name (DN) of the available master objects.

cdclsv-get-timeout

Displays the time-out value of a specific master object.

Synopsis: `cdclsv-get-timeout [-h] | <object_DN>`

Displays the time-out of <object_DN> in seconds.

**cdclsv-set-timeout**

Sets the time-out value of a specific master object.

Synopsis: `cdclsv-set-timeout [-h] | <object_DN> <timeout_in_seconds>`

Sets time-out of *<object_DN>* to *<timeout_in_seconds>*.

cdclsv-invoke

Executes the activity defined for a specific master object.

Synopsis: `cdclsv-invoke [-h] | <object_DN> [--prefix|-p <dump_file_name_prefix>]`

Executes the activity defined for *<object_DN>*.

If the *<dump_file_name_prefix>* argument is given, then the name of packing activity contains this name as a prefix instead of the name of the packer or the name that was defined in the `CdclsPackerOption`, see Section 2.1.3 on page 5.

cdclsv-status

Prints the status of the activity defined for a specific master object.

Synopsis: `cdclsv-status [-h] | <object_DN>`

Prints the status of the activity defined for *<object_DN>*.





Reference List

Documents

- [1] *LEM System Architecture Description*, 1/155 53-CXP 902 5257/4