Contents

Introduction

This document describes how to troubleshoot MallocLite memory leaks on Cisco IOS®software platforms.

It also specifies the information you should gather before you open a Cisco Technical Assistance Center (TAC) case or reload the device. Collect the outputs mentioned in this document, and attach them to the TAC case in order to help expedite problem resolution.

Background Information

MallocLite is used by the memory manager in order to allocate small, fixed size pieces of memory, known as chunks, for allocations less than or equal to 128 bytes. Small memory allocations do not have the overhead of a block header for every allocation. This feature is supported for processor memory pools only.

Every memory block header takes about 48 bytes of memory, and the smallest block takes about 24 bytes. With a traditional approach in Cisco IOS software for each allocation, you would consume at least 72 (48 + 24) bytes of memory, even if you need to allocate only 8 bytes of actual data.

With MallocLite, this overhead can be reduced by the use of chunks. There is still some overhead, because the chunks have to be managed. However, since the chunks are fixed size, they are managed in a different way than blocks, and the overhead is less.

It is the responsibility of the applications that use the MallocLite memory to free it properly. MallocLite masks the user of the memory.

Troubleshoot

Note: The Cisco CLI Analyzer (registered customers only) supports certain **show** commands. Use the Cisco CLI Analyzer in order to view an analysis of **show** command output.

Identify Application Responsible for Leak

It is usually difficult to identify an existing bug if you search only by the *malloclite* keyword.

This example shows that the *MallocLite* process is holding an abnormal amount of memory:

```
#show processes memory sorted

Processor Pool Total:  1614282720  Used:  1544726580  Free:   69556140
     I/O Pool Total:  313524224   Used:  115564032   Free:  197960192

PID TTY  Allocated      Freed    Holding    Getbufs    Retbufs Process
 0   0          0          0  1476043512         0          0 *MallocLite*
```

You need to identify the exact application that is responsible for leak. Three possible identification methods are:

- Decode allocator PC.
- Investigate MallocLite memory statistics.
- Disable MallocLite.

Decode Allocator PC

Even with MallocLite turned on, you can usually see what function asked for the memory. Output from the **show memory allocating-process totals** command might show different PC values even though the name reported is MallocLite:

```
#show processes memory sorted

Processor Pool Total:  1614282720  Used:  1544726580  Free:   69556140
     I/O Pool Total:  313524224   Used:  115564032   Free:  197960192

PID TTY  Allocated      Freed    Holding    Getbufs    Retbufs Process
 0   0          0          0  1476043512         0          0 *MallocLite*
```

A Cisco TAC engineer can decode the PC values from the top of the list (with the highest total). This helps identify the application that has the memory leak.

Investigate MallocLite Memory Statistics

Among the enhancements added in Cisco IOS software release 15.1T was a new CLI that displays the summary of MallocLite memory allocated by each PC. The **show memory lite-chunks** command can help you identify applications that are using a large amount of MallocLite blocks.

```
#show processes memory sorted

Processor Pool Total:  1614282720  Used:  1544726580  Free:   69556140
     I/O Pool Total:  313524224   Used:  115564032   Free:  197960192

PID TTY  Allocated      Freed    Holding    Getbufs    Retbufs Process
 0   0          0          0  1476043512         0          0 *MallocLite*
```

Refer to the command reference for details of the **show memory lite-chunks** command.

```
#show processes memory sorted

Processor Pool Total:  1614282720  Used:  1544726580  Free:   69556140
     I/O Pool Total:  313524224   Used:  115564032   Free:  197960192

PID TTY  Allocated      Freed    Holding    Getbufs    Retbufs Process
```

```
 0   0           0           0  1476043512          0           0 *MallocLite*
```

Examples of output from this command include:

```
#show processes memory sorted

Processor Pool Total:  1614282720  Used:  1544726580  Free:   69556140
     I/O Pool Total:  313524224   Used:  115564032   Free:  197960192

PID TTY  Allocated      Freed     Holding    Getbufs    Retbufs Process
 0   0           0           0  1476043512          0           0 *MallocLite*
```

Again, the TAC engineer can decode PC values with the highest total and identify the application that is leaking the memory.

Disable MallocLite

The MallocLite feature is enabled by default. In order to investigate the MallocLite leak, you can disable MallocLite:

```
#show processes memory sorted

Processor Pool Total:  1614282720  Used:  1544726580  Free:   69556140
     I/O Pool Total:  313524224   Used:  115564032   Free:  197960192

PID TTY  Allocated      Freed     Holding    Getbufs    Retbufs Process
 0   0           0           0  1476043512          0           0 *MallocLite*
```

The leaked memory will still be under MallocLite until the next reload; however, you can start to monitor further leaks with the **show processes memory sorted** and **show memory allocating-process totals** commands. The leaks will now appear under the real process.

If the device runs very low on memory, you must save the configuration and reload the device in order to release the memory:

```
#show processes memory sorted

Processor Pool Total:  1614282720  Used:  1544726580  Free:   69556140
     I/O Pool Total:  313524224   Used:  115564032   Free:  197960192

PID TTY  Allocated      Freed     Holding    Getbufs    Retbufs Process
 0   0           0           0  1476043512          0           0 *MallocLite*
```

The memory might deplete again over time, so use the **show processes memory sorted** and **show memory allocating-process totals** commands in order to monitor memory usage from that point forward.

> **Note**: If you effectively disable MallocLite with the **no memory lite** command and reload the device, output from the **show memory lite-chunks** command will be empty.

Refer to the command reference for details of the **memory lite** command.