

Using Remote Process Monitor Console (Procmon)

Document ID: 20422

Introduction

Prerequisites

Requirements

Components Used

Conventions

Usage

Procmon Tracing

Procmon Earlier Than ICM 4.0

Procmon in ICM 4.0 and Later

Related Information

Introduction

The Process Monitor Console (**procmon**) is the main interface console process. You can use **procmon** to query information indirectly from the Automatic Call Distributor (ACD) of your site. The Cisco Intelligent Contact Management (ICM) processes that reside on the Peripheral Gateway (PG) allow you to:

- List agents, skills, services, and call data
- Increase tracing on an ICM process
- Query for ACD-specific information

Prerequisites

Requirements

Cisco recommends that you have knowledge of this topic:

- The troubleshooting and support of ICM PG

Components Used

The information in this document is based on ICM version 4.6.2 and later.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

Refer to Cisco Technical Tips Conventions for more information on document conventions.

Usage

You can run **procmon** from a Telnet session or a DOS command prompt. The **procmon** process supports both local and remote commands. Local commands are defined within **procmon**, while you need to program remote commands into the monitored process. This section provides a list of basic **procmon** commands and

process-specific commands for use with processes such as:

- Peripheral Interface Manager (PIM)
- Computer Telephony Integration (CTI) Server (CTISVR)
- Open Peripheral Controller (OPC)

Here is an example:

```
Syntax: c:\>procmon /?
Version: Release 4.6.2, Build 08799
Usage: PROCMon CustomerName NodeName ProcessName [SystemName] [/f InputFile]
       [/wait] [/stop] [/help] [/?]
```

Note: The final line of this output displays over two lines due to space limitations.

In order to display a basic list of commands, issue **help**. A list like this displays:

Command	Definition
echo	Controls echo of command lines
emsmon	Controls remote EMS ¹ monitor process (start, stop, pause, resume)
error_stop	Controls setting of stop on error flag
help, ?	Displays help
monitor_help, mhelp	Displays Monitor Server help
monitor_sleep, msleep	Sleeps for specified seconds or milliseconds
quit, q	Ends the program
read_file, read	Directs command input to another input file

¹ EMS = Event Management System

This table provides a list of commands that you can use when you troubleshoot:

Command	Definition
pim_list_agents, la	Lists agents that are currently configured by PIM
pim_list_services, ls	Lists services that are currently configured by PIM
pim_list_skill_groups, lsg	Lists skill groups that are currently configured by PIM
acd_debug, debug	Turns on/off the debug trace
pim_list_trace, ltrace	Lists the current PIM trace bit settings
pim_trace, trace	Sets or resets PIM trace bits
pim_dump_periph, acdperiph	Dumps the contents of the peripheral object

Quit

Ends proemon

Each command has separate syntax. In order to determine the syntax, issue the command and follow it with **/?**.

Note: Each peripheral type contains a different set of commands. For a list of commands that are associated with each peripheral, issue **mhhelp**.

Here is sample output:

```
>>>>1a
SkillTarget ID    Periph#    C          Ext#          Inst#          ActGroups
      5000          6000      Y Yes    3000(3000) <1/ 1> [BO] [265436], <2/ 2> [BO][866278]
      5001          6001      Y Yes    3001(3001) <1/ 1> [AV] [59704], <2/ 2> [AV] [59704]
      5002          6002      Y No      -1(-1)
      5003          6003      Y No      -1(-1)
      5028          6030      Y No      -1(-1)
```

```
>>>>1s
SkillTarget ID    Periph#    C    SerMem    Pri    SerTH    SLType    PSLType    Ext#
      5017          6500      Y      1         2      30       1         4
      5018          6501      Y      2         1      30       1         4
      5019          6502      Y      3         1      30       1         4
```

In this output, **SLType** indicates the default value for the **ServiceLevelType** field for each service that is associated with the peripheral. This value indicates how ICM calculates the service level. You can override the default for individual services.

PSLType indicates the default value for the **PeripheralServiceLevelType** for each service that is associated with the peripheral. You can override the default for individual services.

```
>>>>1sg
  Periph#    Pri    C    SkillTarget ID    Ext#
      1         0    Y      5007          6900
      1         1    Y      5008          6900
      1         2    Y      5009          6900
      2         0    Y      5010          6901
      2         1    Y      5011          6901
      2         2    Y      5012          6901
      3         0    Y      5013          6902
      3         1    Y      5014          6902
      3         2    Y      5015          6902
      3         3    Y      5016          6902
```

```
>>>>debug /?
Usage: acd_debug [/noagent] [/agent] [/agent+] [/agent++] [/nobri] [/bri] [/bri+]
        [/nocall] [/call] [/call+] [/nocms] [/cms] [/cms+] [/csc]
        [/csc+] [/nocsc] [/noconfig] [/config] [/nocv] [/cv] [/noerror]
        [/error] [/nohb] [/hb] [/noopc] [/opc] [/nopost] [/post] [/nosim]
        [/sim] [/notg] [/tg] [/notimer] [/timer] [/notp] [/tp] [/tp+]
        [/trace] [/novq] [/vq] [/warning] [/nowarning] [/all] [/noall]
        [/set UserSetBit] [/help] [/?]

>>>>debug /call+ /post /agent
Trace: AGENT CALL+ POST
UserTraceLevel=0xE848200003FFFFFF800E000000000000000000000000040
Time stamp: 09/13/02
```

Note: The **debug** feature only remains active while the process remains active. When the process exits, the **debug** utility no longer functions. In order to make the trace permanent, add the hexadecimal number that you find in the **UserTraceLevel** line to the EMS trace in the registry.

```

>>>>acdperiph
BuildNum: 08799 (Rel 4.6.2) Time: 06/11/02 16:27:40
  SwitchTime=08/26/02 13:56:22, DefRoute=CTIVarMap-NNNNNNNNNN (y=PIM access)CTIString=
  CVBridge=[G3MsgRecvCnt=169239 (0x29517) Min/AllBrisUp=1/1 NumMonitored=1
  PhysBris=0x1 RtBris=0x0 BadBris=0x0]
Bri[0] State=ACTIVE GoIdle=0
  [NtwrkCngstn[Forced=F Switch=F]
  Window=10000 MsgDlyTime=500
  BriCfgParams(Exp.) = [*CvHost[0]=taclablg3 CvHost[1]= ]
  Msgs [Sent=157095 (0x265a7) Recv=169239 (0x29517) ] [SA0id=314182 LastSA0IdRecvd=31482
  Msgs [SendQ=0x0 SentQ=0x0 RecvQ=0x0 ]
  Msgs [PriSendQ=0x0 RecvQ=0x0 ]
  [ActiveAssoc[Avail=2033 Locked=11] OutstandingSent=0x0 Reg{MaxAllowed=4 ChkMtrs=1
  ChkMsgRates=1
  [Meters/Sec (Enabled: Min 0.00 Avg 0.17 Max 2.45 (Tot 28840.16 Samples 229013
  SumAvg 0.13)]
  [NotEnabled]
  Timers=[3PMC=4 ACDSplt=61 AgntCls=30 AgntSt=240 BriHB=60
  CfgRtry=900 StlBriMsg=10 SwtchTm=30 TG=60 StatMntr=28800 StatMntrInit=120]
  SwitchTime=08/26/02 13:56:22
  NumActiveCalls=0 NumAgentsSeen=2
ProcessName=pim1 ShutdownType=1 Duplex=1 Side=0
  GeoTelBaseDir=C:\icr\lab1\PG1B RegistryBase=ICR\lab1\PG1B DMPSYSTEMID=1
  MDSConnections=1 MDSPIMHandle=33 MDSOPCHandle=1 PIMHeartBeatTime=-1
  CTIRestarts=0
  RoutingClientState=SHUTDOWN
  State=ACTIVE StateInitTime=08/27 10:06:55 (16.9 day)
Time stamp: 09/13/02 10:32:36
>>>>

```

Note: For more information on **acdperiph**, refer to Troubleshooting Avaya Definity G3 using Procmon.

Procmon Tracing

Procmon Earlier Than ICM 4.0

- You can use **procmon** to turn up tracing on the PIM, MIS, and CTISVR processes.

Syntax **procmon** *custid nodeid processname* .

Example usage is **procmon bt pg1a pim1**.

- Type **mhhelp** at the >> prompt to access help for **Procmon**. For example, >> **mhhelp**.
- Add Tracing In order to add tracing, use the **sxtrace**, **scrtrace**, and **satrace** commands; use with **/all**. Example usage for **sxtrace** is >>**sxtrace /all**. You must also save the trace by issuing the **svxtrace**, **svertrace**, and **svatrace** commands. It is recommended that you add and save all three trace levels when you troubleshoot Spectrum issues.
- Remove Tracing In order to remove tracing, use the **cxtrace**, **ccrtrace**, and **catrace** commands; use with **/all**. Example **cxtrace** usage is >>**cxtrace /all**. It is always better to remove tracing upon completion of troubleshooting.
- Ems logs With all tracing, you should increase the EmsLogFileMax and EmsAllLogFilesMax settings in regedt32. The path to these values is:

```

HkeyLocalMachine\Software\Geotel\ICM\custid\PGxx\EMS\CurrentVersion\
Library\Processes\processid

```

Note: This value is displayed over two lines due to space limitations.

Procmon in ICM 4.0 and Later

- You can use **procmon** to turn up tracing on the PIM, MIS, and CTISVR processes.

Syntax **procmon custid nodeid processname**. Example usage is **procmon bt pg1a pim1**.

- In order to access help for **Procmon**, type **mhhelp** at the >> prompt; for example, >> **mhhelp**.
- Tracing The **ltrace** command displays all the available tracing options. Apply Transaction Link tracing (**sxtrace**) by typing **trace xact*** at the >> prompt. Apply Agent tracing with the **trace spectrum*** command.

Related Information

- [Turning Up Tracing](#)
- [Using the OPCTest Command-Line Utility](#)
- [Turning Up Tracing](#)
- [IPCC Troubleshooting Guide](#)
- [How to Use the Dumplog Utility](#)
- [Troubleshooting Avaya Definity G3 using Procmon](#)
- [Release Notes for Cisco ICM Software Release 4.6.2](#)
- [Technical Support & Documentation – Cisco Systems](#)

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Nov 02, 2006

Document ID: 20422
