

Resource Manager

A Guide to Setting Up CMS Networks

User Guide

© Copyright 1996 Eurotherm Controls Limited

All rights strictly reserved. No part of this document may be stored in a retrieval system or transmitted, in any form or by any means without prior written permission from Eurotherm Controls Limited

HA24105C005 3 Mike Fox

CHANGE RECORD

New Revision	Date	Section Changed	Description
1	10/03/94		Initial Release
2	20/12/94		Update for Version 1.2
3	27/08/96		Update for Version 1.3 (Change from latex to Word)

TABLE OF CONTENTS

1. SCOPE	4
2. RELATED DOCUMENTS.....	4
3. TOPOLOGY	4
4. ROUTERS	4
4.1 Definition File Format.....	5
4.2 Media	5
4.2.1 UDP	6
4.2.1.1 UDP Name Server	6
4.3 Serial	7
4.3.1 New Method	7
4.3.2 Old Method.....	8
4.4 Application Entities	8
4.5 Logging	9
5. PROXIES.....	10
5.1 How to Set up Proxies	11
5.1.1 Explicit (router)	11
5.1.2 Explicit (roumsg)	11
5.1.3 Automatic (roumsg).....	12
6. ROULOG	12
7. ROUMSG.....	13
8. ROUNET.....	15

1. SCOPE

This document describes how a network of CMS (Version 1.3) nodes may be established.

All messages sent between different nodes are sent via routers. Each CMS node has one (and only one) router. Establishing a CMS network consists of setting up each of the media connections on each router, and then establishing any indirect connections with the use of Router proxies.

This document is then only concerned with the router process and its associated processes

2. RELATED DOCUMENTS

	Title	Document Number	Revision and date
[1]	Resource Manager - A Guide to Tuning CMS	HA24015C004	2

3. TOPOLOGY

Before beginning to establish a CMS network it will be necessary to determine what the topology is, that is what nodes are present and over which media (currently only UDP 4.2.1 and Serial 4.3) they communicate.

If a network is analysed the following can be determined.

- The total number of nodes.
- The total number of nodes that each node can talk to.
- The total number of nodes that each node cannot talk to.

In particular this guide will explain how the last item can be dealt with such that all nodes are in fact able to talk to each other (via other nodes).

4. ROUTERS

Each router (cmsrout) in a network needs to learn how to reach other nodes (ie which medium to use). It also needs to learn what application entities (AE) are located at which nodes. In order to retain all this information the router maintains i) a set of media, ii) a medium to node cross-reference, iii) an AE to node cross-reference. Each of these cross-references are fixed in size when the router is loaded so it is important that these are large enough to hold all the envisaged data.

In versions 1.3 onwards it is possible to import and export this information in definition files.

cmsrout accepts the following options :

- [**<ProcessName>**] Change the default CMS process name from “Router”.
- [**-closed**] Close the network. Prevent access to/from nodes other than those described in the definition file.
- [**-d <FileName>**] Input definition file name
- [**-i <InitData>**] CMS buffer set, default is “4:1024 12:4096”
- [**-a <MaxAEs>**] Maximum number of AEs, default is 64.
- [**-m <Medium> [=<Name>] [<StartPort>] [<StartName>]**] Select medium **<Medium>** with start up parameters **<StartName>** (a string) and **<StartPort>** (a number). These 2 start up parameters will vary from medium to medium. If **<Name>** is not specified then the medium takes the name **<Medium>**.
- [**-n <MaxNodes>**] Maximum number of nodes, default is 32.
- [**-p <MaxProxies>**] Maximum proxies (section 5), default is 8.
- [**-u**] Auto detect AEs.
- [**-v <Period>**] Medium recovery period in seconds, default is 20 seconds (section 4.2)
- [**-x <FileName>**] Exit file name for definition file.
- [**-y <Key>**] CMS shared memory key.

4.1 Definition File Format

The router definition file format is line orientated where each line is one of the following forms :-

- **NODE : <Node> <Medium>**
- **AE: <Node>!<Protocol>.<AE Name>**
- **Proxy: <Node> <Proxy>**

Any line commencing “#” is considered to be a comment and all characters after (and including) a “;” are considered to be a comment.

4.2 Media

Each of the router media is started with 2 parameters, one a string the other an integer. The default values and significance of these is entirely media dependent.

Each medium will be in one of the following states

- [**Uninitialised**] Not yet started by the router.
- [**Started**] Initialisation started but not yet completed.
- [**Active**] Initialised and capable of sending and receiving messages.
- [**Failed**] Medium is no longer capable of sending or receiving messages.

The router attempts to start all of its media on start up. Once start up is complete and at least one medium is active then the router will attempt to either complete initialisation (from the started state) or re-initialise (from the failed state) all other media every router recovery period.

4.2.1 UDP

This section describes the establishment of the router "UDP" medium.

The first stage of establishing the UDP network is to determine where to locate the UDP name server (`udp_serv`), there should be one, and only one, such name server on the network. Unless memory or processing power is very tight, any node will do as the name server is relatively small and is generally dormant. Each router that is to support the UDP medium must be started with the `-m UDP <UDP Name Server HostName> [<Port>]` option, the port parameter is only required if the port number is changed from the default.

4.2.1.1 UDP Name Server

The UDP name server maintains the list of CMS nodes against internet addresses. This list is saved to disk on any change. The name server uses port 7000 by default. If this port is being used by any other UDP or TCP/IP product then it may be changed by using the `-p <Port>` option.

`udp_serv` accepts the following options

- `[-expiry <$Time>]` Delete any entries in the file (see `-file`) that are older than `<Time>` days.
- `[-file <FileName>]` Save the state of the located nodes in `<FileName>`, default is `nodes.CMS`.
- `[-port <Port>]` Change the default port of the UDP name server from 7000 to `<Port>`
- `[-quiet]` Suppress informational messages.

4.3 Serial

This section describes the establishment of the router serial media ("Serial", "Serial8", "Serial7", "Serial6"). These media shall be generically referred to as the serial medium. The Serial medium is a simple point-to-point serial connection.

The characteristics of the media are :-

Medium	Data Bits	Characteristics
Serial	7	An ASCII protocol. This has a low bandwidth but is well suited to use over insecure links.
Serial8	8	A DLE protocol. This is the most efficient protocol. Its only defect is that actual message size is a function of data values. This is the preferred medium to use.
Serial7	8	A DLE protocol. This is the most efficient protocol. Its only defect is that actual message size is a function of data values. This is the preferred medium to use.
Serial6	7	A 6 bits per 7 bit byte compressed protocol. This has a higher bandwidth than that of "Serial" over 7 bit links but puts a larger overhead on the processor.

Other communications parameters may be set to choice.

The serial medium may be initialised in one of two ways.

4.3.1 New Method

The preferred way of initialising the medium is for is to take a <StartName> only.

This is a comma separated list of parameters and values each of which is of the form parameter=value.

These are :-

Parameter	Values	Default	Description
port	"A"-	"A"	Port number
baud	75-115200	9600	Baud rate
stop	1, 2	1	Stop bits
parity	"e", "o", "n"	"e"	Parity
mintimeout	1- maxtimeout	5	Minimum message timeout (in seconds)
maxtimeout	mintimeout-	10	Maximum message timeout (in seconds)
node	CMS Node	None	Node number of other end.

4.3.2 Old Method

The old way to initialise the medium is for it to take a <StartName> which the node at the other end of the serial line if known. If not supplied then the medium will interrogate the other end. The <StartPort> is a numeric value which is a combination of the baud rate and the port number. The last digit indicates the port number eg 9601 implies 9600 baud on port "B", 2400 implies 2400 baud on port "A". Baud rates from 300 to 115200 are valid, but it is a local issue if they are implemented. Port values are a local issue. This results in effective timeouts of 5 seconds (or slightly more) per message. The communications parameters are always 1 stop bit, even parity.

4.4 Application Entities

If the -u option is supplied to the router then it will periodically update its list of AEs for each node.

If no input definition file is supplied this is the **only** way that a router can establish the location of AEs and therefore the -u option is **mandatory** in these cases.

It should be noted that the -closed option may be used with -u, i.e. the network is closed to other nodes but not to other AEs on those nodes.

4.5 Logging

Each router is capable of supporting up to 4 logging devices (section 6) which receive router log messages when generated.

The log messages that may be issued are :

- **[Node <Node> : Cannot find a medium]** - No medium can be found for node.
- **[Node <Node> : Failed to route on]** - Message to node was not routed over its medium.
- **Node <Node> : Failed to find media for pending queue]** - Message placed on the pending queue is now lost due to failure of the medium.
- **Node <Node> : Not found, failed to route on]** - No medium could be found for a message to node that has arrived from another router.
- **[Node <Node> : Router starting]** - The router is starting and now attempting to initialise all media.
- **[Medium <Medium> : Initialised]** - Medium is now initialised.
- **[Medium <Medium> : Failed to initialise]** - Medium failed to initialise.
- **[Medium <Medium>]: Initialisation started]** - Initialisation of Medium has begun but not yet completed.
- **[Medium <Medium> : Failed]** - Medium has failed, and the router now attempts to re-initialise.
- **[Medium <Medium> : Failed to receive]** - Message not read from Medium when Medium identified message to be read.
- **Medium <Medium> : Failed to route message]** - A message was not routed over Medium.
- **[Medium <Medium> : Could not route intra-resource]** - A message was not routed within the Resource over the Medium.
- **[AE <AENAME> : Cannot locate node]** - Node for AE not yet detected.
- **[Node list full]** - Node could not be added to cross-reference list as it is full.
- **[Proxy list full]** - Proxy could not be allocated as list is full.
- **[Unsupported request]** - Message received as a router message is not a supported type.
- **[No buffer available to formulate a response]** - Router could not respond to a router message due to lack of buffers.
- **[Message too short]** - Only a fragment of a message has been received from a medium.
- **[Bad destination address]** - Could not decode address.
- **[Bad destination address mode]** - Unsupported addressing mode used.
- **Bad format message received for routing]** - The message received from a local process has been encoded incorrectly.
- **[Incorrect length message received for routing]** - The message received from a local process is not of the correct length.

- [Unintelligible delivered message] - The router message header is garbled.
- [Bad address mode for routing] - Could not decode address.
- [Unsupported logical mode] - Node = 0.0.0.0, illegal.
- [Bad format message] - Message not in expected universal format.
- [Unloaded] - Router has been unloaded.
- [Exiting due to signal <Number>] - Unix router crash due to the specified signal.

5. PROXIES

For every node which a router cannot talk to, a proxy router should be established. The proxy router is a router which is capable of reaching both other routers. In effect both of the pair of routers which cannot talk to each other need to be told to use another router which can talk to both. That is all messages between these two nodes are sent via the proxy router.

Depending on the topology there may be one or more routers capable of acting as Proxies. If there is more than one router capable of acting as proxy then there is no requirement that the same proxy router is used between 2 nodes in both directions.

Requiring a Resource to act as proxy does put an additional burden on the router(s) in those Resources, and so it may prove beneficial to spread the burden of proxy among as many Resources as possible.

If it is known in advance that certain Resources will NEVER be required to talk to certain other Resources for which a proxy would be required, then it is possible to omit setting up these proxies.

Having determined which Resources are to act as proxies for other Resources it is now possible to decide how to set up the proxies.

5.1 How to Set up Proxies

Proxies are set up by issuing a special router message to the source router informing it that in order to reach the destination Resource that it cannot talk to directly that it should send all such messages to another Resource (the proxy) which can talk to both source and destination Resources.

There are 3 methods of setting up the required proxies, two requiring explicit statements on how connections should be made and the other automatic which allows for connections to be made any valid way.

5.1.1 Explicit (router)

The router input definition files may be used to explicitly set up required proxies. This must be done locally for each router and cannot be managed centrally.

5.1.2 Explicit (roudmsg)

By compiling a file of router messages for input to the roudmsg tool (section 7) it is possible to set up the required proxies. The same operation could be done by manually typing in the required messages, but this would not allow for an automatic start up.

For example if the messages have been stored in a unix file called proxies then the command

```
roumsg < proxies
```

could be included in a start up script.

Each such proxy message is of the form

```
<Node> SetProxy <Node> <Proxy>
```

There are 2 basic approaches to setting up proxies this way

- [Centrally] All proxies are set up from one Node
- [Locally] Each node sets up its own proxies

Setting up the proxies centrally has the advantage that all the information is contained in one place and is easy to maintain. It does however have several disadvantages in that it requires the whole network to be up to complete its task and in the event of a router being unloaded any proxies set up for that node will be lost.

This may be overcome in one of 2 ways :

- the -persist option can be used ensuring that roudmsg successfully completes one message delivery before proceeding to the next.
- the sequence proxy messages can be periodically repeated, the routers effectively ignore any repeat requests.

Keeping all proxies local has the disadvantage that the information is spread about, but the advantages that it is easy to maintain all the local proxies and that they can be reloaded on start up.

5.1.3 Automatic (roumsg)

The automatic method uses the `rounet` tool (section 8) in the active mode. This leaves the decision about which nodes to use as proxies up to the tool. It would be perfectly acceptable to have many `rounet` tools running on the network, this would probably achieve the same result slightly faster at the expense of increasing network traffic.

`rounet` needs to be started with `-nodes` to force the finding of nodes or started at a node whose router supports which multiple media (if there is no such node then no proxies are required).

6.ROULOG

The `roulog` tool collects router logging messages from a set of routers and prints them on `stdout`. `roulog` accepts the following options

- [**<Name>**] CMS process name, default "RouterLogger".
- [**-nodes <Nodes>**] Nodes to log messages from, default "this" node. Maximum of 40 nodes.
- [**-y <Key>**] CMS shared memory key.
- [**-z <Time>**] Time in between re-establishing `roulog` as a logger in seconds, default is 10 seconds. This caters for CMS nodes that are unloaded and then reloaded.

Example roulog output

```
1.1.1.1 "Node    5.5.5.5      : Not found, failed to route on"
4.4.4.4 "Node    6.6.6.6      : Not found, failed to route on"
2.2.2.2 "Node    7.7.7.7      : Not found, failed to route on"
1.1.1.1 "Proxy list full"
```

7.ROUMSG

The `roumsg` tool reads from `stdin` a single router message, parses it and if ok sends it to the destination, waits for the reply and prints the result on `stdout`.

`roumsg` accepts the following options

- [**<Name>**] CMS process name, default "RouterMessenger".
- [**-echo**] Echo requests on `stdout`.
- [**-line**] Format each response as a single line of text, default is multi-line for lists.
- [**-persist**] Continue send this message until a response is received.
- [**-retries <Retries>**] Number of retries for any message which times out, default 0.
- [**-wait <Seconds>**] Maximum period to wait for a reply, default 10 seconds.
- [**-y <Key>**] CMS shared memory key.

Each router message is of the form :

`<Node> <MessageType> [<Message arguments>]`

where the message body is one of

- [**ListNodes**] List the nodes found.
- [**ListMedia**] List the media supported.
- [**MediumStatus <Medium>**] Report the status of a medium and the number of messages sent and received across it.
- [**ListProxies**] List all proxies
- [**SetProxy <Node> <Proxy>**] Use Proxy as a proxy for Node.
- [**ListAEs**] List all AEs found.
- [**ListLocalAEs**] List all AEs on the routers node.
- [**SetLogger**] Set this tool to be a router logger.

`<Node>` may also be either `"^"` - the last node to which a messages was sent or `"."` - this node.

Example roumsg Session

```
. listnodes
1.1.1.1 Response ListNodes
    2.2.2.2 on Medium1
    3.3.3.3 on Medium2
3.3.3.3 mediumstatus Medium2
3.3.3.3 Response MediumStatus Medium2:UDP Active
    Inits = 1
    Sends = 236
    Receives = 237
3.3.3.3 listnodes
3.3.3.3 Response ListNodes
    4.4.4.4 on Medium3
    1.1.1.1 on Medium2
^ listproxies
3.3.3.3 Response ListProxies
    2.2.2.2 by 1.1.1.1
. listproxies
1.1.1.1 Response ListProxies
    4.4.4.4 by 3.3.3.3
```

8.ROUNET

The `rounet` tool builds up a picture of the CMS network by sending router messages (the same messages as those that can be issued by `roumsg` (section 7). This picture consists of a map of the Routers, their media and (optionally) their Aes. It also produces a picture of the connections between nodes, via which media and with which proxies. It may also be used in an active mode to set proxies for those connections that need proxies but have not had them set already.

`rounet` accepts the following options

- [**<Name>**] CMS process name, default "Networker".
- [**-active**] Make indirect connections that have not been made using proxies.
- [**-connections <FileName>**] Write the connections output to the file `<FileName>`, default is `stdout`.
- [**-destruct <Times>**] Destroy the information every `<Times>` cycles, default is 10.
- [**-listAEs**] Report the AEs in the map, default is not to.
- [**-map <FileName>**] Write the network map to the file `<FileName>`, default is to `stdout`.
- [**-nodes <Nodes>**] Force nodes `<Nodes>` to be found, maximum of 5 nodes.
- [**-timeout <Timeout>**] Set message timeout to `<Timeout>` seconds, default is 30 seconds.
- [**-x**] Exit when network picture is complete.
- [**-y <Key>**] CMS shared memory key.
- [**-z <Sleep>**] Delay `<Sleep>` seconds in between each cycle, default is 60 seconds.

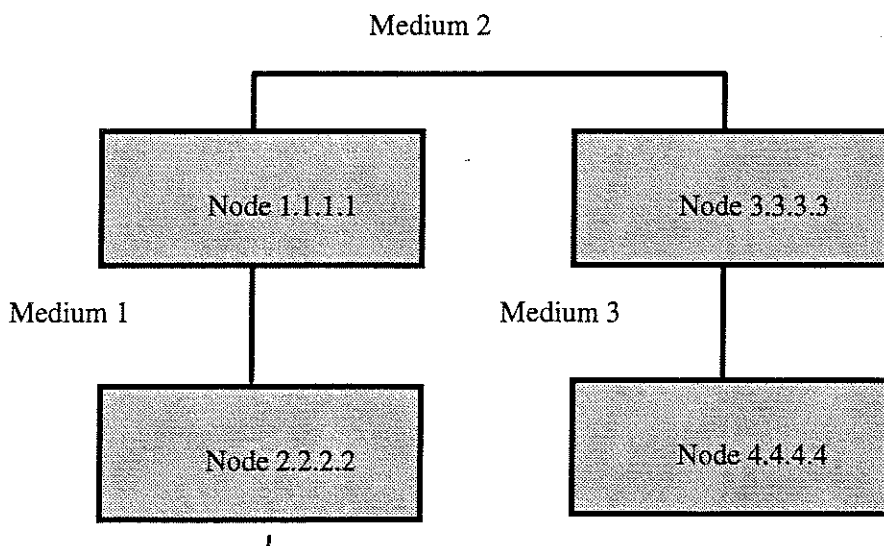


Figure 1: An Example Topology

This would if rounet was run with -active result in the following output.

This topology (figure 1) shows a network where no node has a direct connection to all nodes, and some nodes have 2 levels of indirection to reach other nodes.

Example Map

```
Router: 1.1.1.1
  Medium: Medium2
    Node: 3.3.3.3
  Medium: Medium1
    Node: 2.2.2.2
  Proxies:
    Node: 4.4.4.4 with 3.3.3.3
Router: 2.2.2.2
  Medium: Medium1
    Node: 1.1.1.1
  Proxies:
    Node: 3.3.3.3 with 1.1.1.1
Router: 3.3.3.3
  Medium: Medium2
    Node: 1.1.1.1
  Medium: Medium3
    Node: 4.4.4.4
  Proxies:
    Node: 2.2.2.2 with 1.1.1.1
Router: 4.4.4.4
  Medium: Medium3
    Node: 3.3.3.3
  Proxies:
    Node: 1.1.1.1 with 3.3.3.3
```


Example Connections

1.1.1.1=>(Medium1)=>2.2.2.2
1.1.1.1=>(Medium2)=>3.3.3.3
1.1.1.1=>(Medium2@3.3.3.3)=>(Medium3)=>4.4.4.4
2.2.2.2=>(Medium1)=>1.1.1.1
2.2.2.2=>(Medium1@1.1.1.1)=>(Medium2)=>3.3.3.3
2.2.2.2=>(Medium1@1.1.1.1)=>(Medium2@3.3.3.3)=>(Medium3)=>4.4.4.4
3.3.3.3=>(Medium2)=>1.1.1.1
3.3.3.3=>(Medium2@1.1.1.1)=>(Medium1)=>2.2.2.2
3.3.3.3=>(Medium3)=>4.4.4.4
4.4.4.4=>(Medium3@3.3.3.3)=>(Medium2)=>1.1.1.1
4.4.4.4=>(Medium3@3.3.3.3)=>(Medium2@1.1.1.1)=>(Medium1)=>2.2.2.2
4.4.4.4=>(Medium3)=>3.3.3.3

—oOo—