Creation:
January 11, 1993
Revised:
January 22, 1993

Environment Maintenance Tool Installation Guide

David Markley

Computing Services
Carnegie Mellon University

This document describes how to install the Environment Maintenance Tool.

Familiarity with AFS is assumed. The document listed in section 1.0 will provide background information.

1.0 Related Documents

Colyer, Wallace; Held, Mark; Markley, David; and Wong, Walter, *Software Management in the Andrew System*, AFS Users Group Proceedings, Summer 1992.

2.0 Installation Overview

Installing EMT requires an understanding of your current software management situation; knowledge of **vos** and **pts** commands; access to an admin instance in your cell; installation of the **site**, **depot**, and **ADM** applications; and a desire for simplified software management. Once you have all these things, installation should be fairly straightforward.

3.0 Before You Begin

Before diving into compiling EMT, it would be best if you started setting up the software management structure at your site to conform to a structure that EMT will be able to manage. EMT makes software management an almost trivial task, and it does so by imposing a solid structure. Before using EMT, you will have to create the various areas it required “by hand.” See Appendix D for an example of how you might set up your software management structure.

4.0 Compiling EMT

Obviously, the first things you will need are working binaries for the server and client of EMT. This will require modifying a few files contained in the sources you have. After you have completed these modifications, you may use **make, build**, or **smake** to compile the EMT sources.

4.1 Modify rxemt.xg

Contained in the rx subdirectory, this file contains the defines that determine tell EMT a great deal about its working environment.

4.1.1 EMTD_RPC_PORT

This defines the port that the EMT server will bind to. Use the default, or pick a port that no other service on the host machine will use.

4.1.2 EMTD_RPC_SERVICE

This defines the service number that the EMT server will use in binding its socket.

4.1.3 EMTD_RPC_NAME

This defines the name of the service that EMT will use in binding its socket.

4.1.4 EMTD_WORK_NAME

This is the name of the user that EMT will authenticate as, when doing all of its work. This should be an account that has been specifically created for EMT. A real user should not be using this account. If you are not using MIT kerberos, you will need to set the key for the user and place this in the KeyFile within the CONF directory.

4.1.5 EMTD_WORK_INSTANCE

This is the instance of the user that EMT will authenticate as. There is no need for EMT to use any special instance (such as an admin instance), so you may simply leave this as an empty string.

4.1.6 EMTD_WORK_CELL

This is the cell that the account EMT will be using resides in. This should be set to your local cell name. You may set this to be NULL as well.

4.1.7 EMTD_WORK_ID

This is the ID of the account that EMT will be using.

4.1.8 EMTD_SERVER_CELL

This is the name of the cell that the EMT server will be working in. Set this to be the name of your cell. This should not be set to NULL.

4.1.9 EMTD_WORKING_DIRECTORY

This is the path to the directory from which EMT will read its configuration files, and under which it will write its logs.

4.1.10 EMTD_CONF_PATH_DEFAULT

This is the default configuration path. It is where EMT will look for its KeyFile and other standard AFS configuration files. If you are using MIT kerberos, you can set this to be an empty string.

4.1.11 EMTD_DB_DIRECTORY

This is the path to the database directory that EMT uses. This should point to a database volume, for EMT, that is mounted somewhere in AFS.

4.1.12 EMTD_SRVR_DUMP_PATH

This is the path to a directory containing nightly dumps from the file servers. These dumps are done through a local modification to AFS at CMU. Since this is only used in the **vclean** command, it is unlikely that you will need to set this to anything other than an empty string. If you would like more information about this, please contact EMT support at CMU (markley+@cmu.edu).

4.1.13 EMTD_SCRIPTS_DIRECTORY

This is the path to the area containing the scripts used by EMT in its execution. These scripts came with the EMT source, and are initially located in the **scripts** subdirectory. It doesn’t matter whether you place these on the local disk or on AFS, so long as the path is correctly set here.

4.1.14 EMTD_RSH_PATH

This is the path to the **rsh** executable. If you are not using MIT kerberos or don’t plan to use **rsh**, you can set this to be an empty string.

4.1.15 EMTD_OPLOG_COMMAND

This is the command line invoked for logging operations that EMT has performed. At CMU, we use an application named **oplog** (see later description). EMT execs this command line, and writes to it through its stdin. You may use any application to log the completion of EMT commands in this way. You could set this to be a dummy shell script, or simply set it to send mail to someone or some bboard.

4.1.16 EMTD_DEFAULT_ADM_SERVER

This is the name of the default ADM server host machine. Set this to be the name of the machine that the ADM server is running on.

4.1.17 EMTD_DEFAULT_ADM_PORT

This is the default name of the port that the ADM server will be listening to.

4.1.18 EMTD_DEFAULT_HOST

This is the name of the default EMT server host machine. Set this to be the name of the machine that the EMT server will be running on.

4.1.19 EMTD_MAX_MONITORS

Monitors are the connections to EMT that are watching a single command’s execution. This is the maximum number of monitors that EMT will allow on each command. Each monitor only takes about 8 bytes, but 10 monitors seems to be more than enough.

4.2 Modify the top-level Makefile

If all goes well, you will only have to modify the top-level Makefile. This Makefile contains several variable definitions that determine the functionality that will be present in the EMT server and client.

4.2.1 GNU Readline

Allowing USE_GNURL to be defined will enable the client to take advantage of the gnu-readline functionality.

4.2.2 MIT Kerberos

If you do not wish to use MIT kerberos, you must comment out the definitions of both USE_MITKERB and KRBLIBS.

4.2.3 Remote Shell (RSH)

If you are using MIT kerberos, and wish to take advantage of **rsh** through EMT, you must allow USE_RSH to be defined. Comment it out, if you don't want this feature.

4.2.4 Operation Logging

If you want EMT to log the operations it performs through the utility you defined with EMTD_OPLOG_COMMAND in rxemt.xg, you must allow USE_OPLOG to be defined. Comment it out, if you don't want things logged, except to file on the local disk.

4.2.5 Timestamps

If you plan to use timestamps on links in your depot areas under your software management scheme, you must allow USE_TIMESTAMPS to be defined. Comment it out, if you don't want to use timestamps.

4.2.6 Destination Directory

DESTDIR is the path that will be used when installing EMT. You should set it to point to the volume you wish to install EMT into. This volume should be mounted in the destination area initially, according to your software management directory structure.

4.2.7 Root Directory

ROOTDIR must be set to the root path of the area under which common include and library directories will be found. At CMU, this is /usr/local/.

4.2.8 Library Directories

LIBDIRS must be set to point to the directories that your libraries will be found in.

4.2.9 auth Library

LIBAUTH must be set to indicate the correct "auth" library for your site. If your are running AFS 3.2, this should be libauth.krb.a.

4.2.10 ADM Library

LIBADM must be set to the path for the ADM library at your site.

4.3 Make the executables

After having edited the necessary files and setting things the way that your site requires them, you should simply need to run make in the top level of the EMT source area. If you run into errors along the way, you may have to check the way you have modified the files. If it seems to be a more general problem, you may wish to contact EMT support at CMU (markley+@cmu.edu).

5.0 Software Management Structure

In order for EMT to maintain your software, it will require knowledge about your software management structure. EMT reads this information from the file named **emtd.config**, which must be placed in the directory you indicated by setting EMTD_WORKING_DIRECTORY (in rxemt.xg). This file describes the structure of your software management in terms of the trees, environments, and system types you will be using. A sample file is contained in the **local** subdirectory of the EMT source are, and can be found in Appendix A of this document. This describes the structure at CMU, which you might use as an example of how to structure your site.

The emtd.config file has a very simplistic format. Each line starts with a key word followed by a colon. This indicates what information is present on that line. There ordering of these key words is important in defining your structure. The following is a list of the key words, and the information that should be on the lines that follow them.

5.1 Tree Description

FIGURE 1

```
tree: <tree name>
```

The largest overall structure in our software management scheme is called a tree. To start the definition of a tree, you must first have a line with the **tree** key word, followed by the name of the tree. Everything following this line in the configuration file will be read as part of this tree, until another **tree** key word is encountered.

5.2 Source Path

FIGURE 2

```
src_path: <source path>
```

EMT needs to know where the source area is for the tree. You must place the path to the top level of your source area after this key word.

5.3 Database Path

FIGURE 3

```
db_path: <database path>
```

EMT must have an area that where it can store and update information. You must place the path to the database area you have created for EMT after this keyword.

5.4 Group Ownership

FIGURE 4

```
owner: <owning pts group>
```

You should place the name of the **pts** group that should own all other **pts** groups created for this tree after this key word.

5.5 Environment Description

FIGURE 5

```
environment: <environment name>
```

This is the next largest structure of software management after the tree. You must place the name of the environment you are defining after this keyword. Every line that follows this key word will be associated with the environment named, until another environment or tree key word is reached. Note that there are two special environments that must be defined for every tree: **obj** and **dest**. These environments are defined in the same way as all other environments, but they have special meaning to EMT. They specify the object and destination areas that EMT will use for the current tree in all its operations on that tree.

5.6 System Description

FIGURE 6

```
system: <system name>,<path>,<depot type>,<replicated>,<unrep
days>,<expire days>
```

This key word describes the smallest part of our software management: the system. Information contained all the lines following this key word will be associated with the system being defined, until another system, environment, or tree key word is reached.

5.6.1 <system name>

Should be filled with the name of the system being defined.

5.6.2 <path>

Should be filled with the path to the top level for this system, under the current tree and environment. When working with the object and destination environments, this will be the path to the working areas for these environments. For all other environments, this should be the path to the depot area, without “depot” on the end.

5.6.3 <depot type>

This is used to indicate which version of depot is to be run. This is only really significant at CMU. You should always set this to be non-zero.

5.6.4 <replicated>

Should be set to indicate whether this system should have collections replicated, when they are released to this environment and tree. Setting this to be non-zero will indicate that collections should be replicated. This has no meaning within the object and destination environments.

5.6.5 <unrep days>

Should be set to indicate the number of days that will be allowed to pass before old, unreleased replicated volumes will be unreplicated (in order to save disk space).

5.6.6 <expire days>

Should be set to indicate the number of days that will be allowed to pass before old mountpoints in the depot area will be removed. These old mountpoints result when EMT is using the timestamp routines.

5.7 Replicated Volumes

FIGURE 7

```
volume_rep: <volume name>
```

This key word indicates the name of a volume that should be released when changes are made to the current system's area. Any number of lines with this key word can come after a system key word. This has no meaning for systems within the destination or object environments.

5.8 Hosts used by rsh

FIGURE 8

```
rsh_host: <host name>
```

This defines the host that should be used when performing an **rsh** operation for some function operating on the current system. Any number of these key word lines can be listed after a system key word. The host names will be stored with the system in the order they are supplied. If **rsh** fails to connect with a host, it will try the next host on the list. This has no meaning in the object and destination environments, or if you do not plan to use **rsh**.

6.0 Tree Configuration

In order to allow for different configurations for each tree, EMT reads a configuration file for each tree. This file is named **config.<tree name>**, and must be placed in the same directory as **emtd.config** (see previous section). There is an example config-local found in Appendix B. The keywords that can be used are described as follows:

6.1 Limiting Quota

FIGURE 9

```
quota.<collection name>: <maximum quota>
```

Should you decide to allow collection maintainers to perform certain commands on their collection, this is how you can limit the amount of quota they can have for each volume in the collection. If they want more than this amount, they will have to contact the Gatekeeper. Setting <maximum quota> to be zero will give the maintainer access to unlimited quota for their collection. The <collection name> is the name of the collection which this quota restriction will affect. If "*" is used as the collection name, or no collection name is given, the maximum quota specified will be the default for every collection within the tree that is not explicitly defined.

6.2 Limiting the number of Versions

FIGURE 10

```
versions: <maximum versions>
```

To limit the number of versions that can exist for a collection at any one time, you must set <maximum versions>. This is useful in ensuring that maintainers delete old versions before allocating space for new versions. If necessary, the Gatekeeper can override this by using the “-force” commands on creates in the EMT client.

6.3 Limiting Command Access

FIGURE 11

```
auth.<command name>: <security level>+
```

As mentioned earlier, it is possible to allow the maintainer to perform certain functions on their collection. Through the auth keyword, the level of security that can call individual commands can be specified.

6.3.1 <command name>

This must be the command name for which authorization is being set. If “*” is used as the command name, or no command name is specified, the security levels listed become the default for any command not otherwise explicitly defined. The current command names that the EMT server recognizes are: create, delete, release, remove, update, setquota, access, migrate, and expunge.

6.3.2 <security level>+

There are three levels of security: global, gatekeeper, and maintainer. The global security level is authorized to perform any command, so it should not be listed. The other levels should be listed, separated by whitespace.

6.4 Limiting the Times Commands are allowed

FIGURE 12

```
time.<command name>.<environment name>: <restricted time>+
```

This keyword allows the software manager to restrict the time that commands can be run under certain environments. This is useful if large releases of volumes cause a significant and noticeable slowdown of access to the file servers.

6.4.1 <command name>

This must be the command for which a time restriction is being set. If “*” is used as the command name, the time restriction becomes the default for all commands within the environment that are not explicitly defined for that environment. The current command names that the EMT server recognizes are: create, delete, release, remove, update, setquota, access, migrate, and expunge.

6.4.2 <environment name>

This must be the environment in which the time restriction for the command is being set. If “*” is used as the environment name, the time restriction becomes the default for the command in any environment where it is not explicitly set. If both the command and environment names are set to “*” or not present, the time restriction becomes the default for any command in any environment for which time restrictions have not been explicitly set.

6.4.3 <restricted time>+

This must be set to indicate the restricted time(s) for the command and environment specified. The format is as follows:

FIGURE 13

```
(<day>+, <start time>-<end time>)
```

`<day>+` is the number of the day for which the restriction should apply. If it should occur on more than one day, the number for each day should appear separated by whitespace. Zero is Sunday, etc.

`<start time>` is the time the restriction should begin. It should be the hour and minute of the 24 hour clock separated by a colon.

`<end time>` is the time the restriction should end. It should be the hour and minute of the 24 hour clock separated by a colon.

More than one of these restrictions can be listed after the time keyword. They should be placed on the same line, separated by whitespace.

7.0 Security Levels

There are three levels of security that EMT recognizes: global, gatekeeper, and maintainer. Each of these levels is allowed a varying degree of access to EMT commands. Before the EMT server will process a command, it will determine which group or groups the calling user is in. If the calling user is not in a group that is allowed to perform the command, EMT will report that the user in unauthorized.

The format for the files that EMT reads to determine the Global and Gatekeeper levels is fairly straightforward. For each user in the group, simply list on separate lines the user’s name, instance, and cell. There are examples of these files in the local subdirectory of the sources, and an example of an admin-global file can be found in Appendix C of this document.

7.1 Global

The Global users are capable of performing any command in any tree on any collection. This is usually the individual or individuals that are acting as the overall software manager. EMT determines who is a Global user by looking in the file named admin-global that should be placed in EMT’s working directory. Obviously, access to this file should be very restricted.

7.2 Gatekeeper

The Gatekeeper is normally capable of performing any command within a specific tree on any collection in that tree. This individual or group of individuals area acting as software managers for the specified tree. EMT determines who is a Gatekeeper by looking in the file named admin-<tree name> that should be placed in EMT's working directory. Again, access to this file should be very restricted.

The reason this level of security is called Gatekeeper comes from the concept that releasing software to an environment requires the opening of a "gate." Since the user or users that have this level of security are the ones that perform the releases, they can be considered keepers of the gate, or Gatekeepers.

7.3 Maintainers

Maintainers are usually not capable of performing any EMT commands that require interaction with the server. This means that they are very limited on the client side of EMT. If the Gatekeeper of Global chooses, the Maintainers may be granted access to some EMT server commands. Maintainers are that individual or group of individuals that are working to in some way support a collection. EMT determines if a user is a maintainer by looking for them in the **pts** group for the collection, named: sysmaint:<collection name>.commanders.

8.0 KeyFile vs. Ticket Files

If you are planning to use MIT kerberos, you will need to set up an **/etc/srvtab** on the host machine for the EMT server. Use "ksrvutil add" to add the key for the user that EMT will run under.

If you plan on running EMT without MIT kerberos, you will need to know the key for the user that EMT will run under. Use the EMT server to add the key to the KeyFile in the CONF directory you designated in **rxemt.xg**. The command should look resemble "emtd addkey -key XXXXXXXX." You will also have to make sure there are links in your specified CONF directory to the CellServDB, ThisCell, and UserList files. These can be usually be found in /usr/afs/etc.

9.0 EMT Overview

EMT was written to automate software management in a distributed computing environment, and thereby reduce the amount of time required to create, delete, release and upgrade software in the system. Any operation in the environment is initiated and handled through EMT. By allowing EMT to control the environment, multiple volume operations, or groups of commands that need to be run to create, delete, or release a collection, are collapsed into a single command. This command can include operations for multiple architectures.

It is easy to incorporate EMT into any Software Management system. The policies and configurations are specified through a configuration file; the EMT server reads the configuration file on start-up and sends each client a copy upon connection. The client verifies that the user's commands are correct before making requests to the server. Currently, changes to the configuration file require the server to be restarted. This is done because a dynamically changing configuration requires all clients to be informed of each change. If clients are not informed of changes, the current user commands might be invalid under the new configuration.

EMT uses the standard AFS common error and command line libraries.

10.0 Related Software

EMT requires several supporting applications to be able to make use of all its funtionality.

10.1 site

This application supplies EMT with the server/partition pairs, when EMT needs to know where to create, replicate, and unreplicate volumes. You may specify theses locations on the command line of the client commands that require it, but **site** provides an automated generation of the correct locations.

10.2 oplog

EMT always keeps daily logs of the operations it has performed, but it can be set to use oplog to log the operations to a bboard and to a zephyr instance. This application can be replaced by any application that will accept data that is piped into its stdin. This means that you could simply replace the EMTD_OPLOG_COMMAND with "mail username," and EMT operations would be mailed to the username you specified.

11.0 Timestamps

One of the options you have, when setting up EMT, is whether or not you want to use timestamps. The difference is simple. If you decide to use timestamps, each time you release a different version of a collection to an environment, the volume is mounted in the depot area as `<collection name>.<timestamp>`. If you decide not to use timestamps, the collection is re-mounted as `<collection name>` in the depot area. The old version, that was mounted there is unmounted first. In both cases, when **depot** is run in the depot area, the latest release of the collection is linked in. When using timestamps, EMT will also write to a file in the database area that revords all releases to environtments.

Using timestamps is useful, if you plan on using depot to set up areas on the local disk (Local Disk Depot). If timestamps were not used, the volume that a workstation might have a link to could change when a different version is released. This could cause problems by creating conflicts on the workstation because of differences in the versions of a collection. By using timestamps, the links that depot creates on a workstation point dirrectly to a particular version of a collection. Even if a different version of a collection is released, the workstation will not show the change until depot is run on its local disk again.

The only major problem with timestamps are that the depot area can quickly fill up with multiple mounts for each collection. To solve this problem there is tool that can be used to remove mountpoints that are older than a certain number of days. The number of days is determined by you in the **emtd.config** file, when defining the system areas. The tool used to remove these old mointpoints comes as part of a software collection named tsdb (contains Time Stamp Database tools). Also in this collection is the tool that unreplicates unreleased volumes.

Appendix A Example emtd.config

```
tree: local
src_path:/afs/andrew.cmu.edu/system/src/local
db_path:/afs/.andrew.cmu.edu/data/db/emt
owner: sysmaint:localboss
environment: obj
system: pmax_42,/afs/andrew.cmu.edu/system/obj/pmax_42/local,1,0,0,0
system: sun4411,/afs/andrew.cmu.edu/system/obj/sun4411/local,7,0,0,0
environment: dest
system: pmax_42,/afs/andrew.cmu.edu/system/dest/pmax_42/local,1,0,0,0
system: sun4411,/afs/andrew.cmu.edu/system/dest/sun4411/local,1,0,0,0
environment: beta
system: pmax_42,/afs/.andrew.cmu.edu/system/beta/pmax_42/local,1,0,10,30
volume_rep: pmax_42.blocal.011
system: sun4411,/afs/.andrew.cmu.edu/system/beta/sun4411/local,1,0,10,30
environment: gamma
system: pmax_42,/afs/.andrew.cmu.edu/system/gamma/pmax_42/local,1,1,10,30
volume_rep: pmax_42.local.011
rsh_host: pmax.wash.acs.cmu.edu
system: sun4411,/afs/.andrew.cmu.edu/system/gamma/sun4411/local,1,1,10,30
volume_rep: sun4411.local.011
rsh_host: sun4c.wash.acs.cmu.edu
tree: contributed
src_path: /afs/andrew.cmu.edu/system/src/contributed
db_path: /afs/.andrew.cmu.edu/data/db/emt
owner: sysmaint:contriboss
environment: obj
system: pmax_42,/afs/andrew.cmu.edu/system/obj/pmax_42/contributed,1,0,0,0
system: sun4411,/afs/andrew.cmu.edu/system/obj/sun4411/contributed,1,0,0,0
environment: dest
system: pmax_42,/afs/andrew.cmu.edu/system/dest/pmax_42/contributed,1,0,0,0
system: sun4411,/afs/andrew.cmu.edu/system/dest/sun4411/contributed,1,0,0,0
environment: gamma
system: pmax_42,/afs/.andrew.cmu.edu/system/gamma/pmax_42/contributed,1,1,10,30
volume_rep: pmax_42.cont.012
rsh_host: pmax.wash.acs.cmu.edu
system: sun4411,/afs/.andrew.cmu.edu/system/gamma/sun4411/contributed,1,1,10,30
volume_rep: sun4411.cont.012
rsh_host: sun4c.wash.acs.cmu.edu
```

Appendix B Example config.local

quota.atk-atk: 80000
quota.gnu-emacs: 80000
quota.tsdb: 5000
quota.emt: 0
quota: 20000
versions: 4
auth.create: gatekeeper maintainer
auth.setquota: gatekeeper maintainer
auth.access: gatekeeper
auth: gatekeeper
time.release.gamma: (1 2 3 4 5, 10:00-17:00)
time.update.gamma: (1 2 3 4 5, 10:00-17:00) (0, 2:00-6:00)

Appendix C Example admin-global

wally@ANDREW.CMU.EDU
dm3e@ANDREW.CMU.EDU
vanryzin@ANDREW.CMU.EDU

Appendix D Example Software Management Structure

The illustration below shows an example of how the structure of your software management area might look. Items that are in boxes are mountpoints for a volume. Items that are in ovals are simply directories.

EMT needs to know the exact path to each of the volumes that appear just below the thick gray line. These paths are defined in **emtd.config**. Once these areas are created and defined, EMT can then maintain the area below the gray line. You can define the area above the gray line in any way you choose. This is merely an example.

Please note that this illustration is not complete. For example, the same structure should appear under the boxes with "/sun4411" in them as appears under the boxes with "/pmax_42" in them. "emacs" and "lindo" are only examples of possible collection names, and more versions than those shown can be mounted in each collection's area.