

# Frequently Asked Question About CEMF Configuration

Document ID: 24804

---

## Questions

- How do I define an object specification file?
  - Is there a quick way to delete all alarms in the system?
  - How does the Cisco Element Management Framework Auto Discovery application know what type of managed object to create when it discovers a device?
  - How do I customize the default Performance Manager configuration to keep the Raw and Summary data?
  - Can I configure Auto Discovery to name discovered network objects with their DNS name instead of their IP addresses?
  - How can I hide values such as community strings that I enter on the Deployment Wizard GUI?
  - Can traps be forwarded to different third party Network Management Systems based on the source of the traps?
  - Can traps from the same device be forwarded to multiple third party Network Management Systems?
  - If an EM is installed that results in a change to the environment (such as adding a script to config/env), when does this change take affect?
  - How can the 'Rename Object' service be removed?
  - How can I change the maximum size of a log file?
  - How can I modify labels for the attributes in the "Monitored Attributes" list in Performance Manager?
  - How do I use the objectFileParser tool for CEMF 3.0.1 to import configuration data in terms of sites/groups/devices into CEMF?
  - How should I configure my ObjectStore installation for CEMF?
- 

**Q. How do I define an object specification file? (Products affected : CEMF 3.x)**

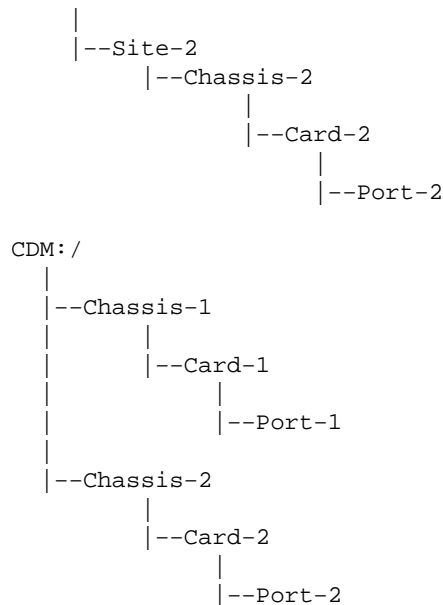
**A.** The general format for an object specification file entry is as follows:

```
OBJECT
  NAME <name of object>
  CLASS <class of object>
  CONTAINMENT <containment path for object>
  :
  ATTRIBUTE <attribute name> (<attribute value>) <attribute section>
  :
ENDOBJECT
```

**Note:** You can specify as many CONTAINMENT and ATTRIBUTE clauses as you want within the Object Specification file.

The following example illustrates how to define a real object specification. It assumes that the following objects are deployed within Cisco Element Management Framework (CEMF):

```
Physical:/
|
|--Site-1
|   |
|   |--Chassis-1
|       |
|       |--Card-1
|           |
|           |--Port-1
```



Assume that you want to create a permanent virtual circuit (PVC) object called PVC-A, which appears in both the Physical and Content Distribution Manager (CDM) views, with two parents, Port-1 and Port-2, in each view. You also want to initialize an attribute of this object (for example: **PVCctrl-MIB.customerName** to **ABC**). The controller that manages your PVC objects is **PVCctrl** and PVC objects are of class **ciscoPVC**. The following object specification file is required for this purpose:

```

OBJECT
  NAME          PVC-A
  CLASS         ciscoPVC
  CONTAINMENT   Physical:/Site-1/Chassis-1/Card-1/Port-1/PVC-A
  CONTAINMENT   Physical:/Site-2/Chassis-2/Card-2/Port-2/PVC-A
  CONTAINMENT   CDM:/Chassis-1/Card-1/Port-1/PVC-A
  CONTAINMENT   CDM:/Chassis-2/Card-2/Port-2/PVC-A
  ATTRIBUTE     PVCctrl-MIB.customerName (ABC) PVCctrl
ENDOBJECT

```

**Q. Is there a quick way to delete all alarms in the system? (Products affected : CEMF 3.x)**

**A.** Yes, you can use the **alarmDeleter** utility in the **<cemf\_root>/bin** directory. You must configure the **[AlarmDeleter]** section in the **<cemf\_root>/config/init/alarmDeleter.ini** configuration file. Specify the age of the alarms you want deleted. For example, if you want to delete all alarms, set the age to **0** (that is, set all attributes in **alarmDeleter.ini** beginning with **age** to **0** (under section **[AlarmDeleter]**)).

In addition, the **alarmDeleter.ini** file contains the **deleteAllAlarms** variable, which can be set to **0** to clear only cleared alarms, and set to **1** to clear active and cleared alarms (all alarms). If you want all alarms to be deleted (not just cleared alarms), then you also need to set **deleteAllAlarms** to **1**.

See page 80 of the Cisco Element Management Framework Physical Architecture, for more information on alarm deletion.

**Q. How does the Cisco Element Management Framework Auto Discovery application know what type of managed object to create when it discovers a device? (Products affected : CEMF 3.x)**

**A.** The Auto Discovery application fetches the **sysDescr** (System Description) and **sysObjectID** (System Object ID) attributes from any Simple Network Management Protocol device it discovers. Auto Discovery then uses the mapping file(s) found under **<CEMFROOT>/config/oidMappings/** to map these attributes to a Cisco EMF class in order to create a managed object within the system to manage the device. These OID mapping files use regular expressions for mapping and the OID mapping file of a particular technology is installed with the Element Manager of that technology. The format of the OID mapping file is as follows:

```

MAPPING name
    ENTERPRISE-ID [<enterprise-oid>]
    SYS-DESCRIPTION [<description>]
    AV-CLASS-NAME [<classname>]
END-MAPPING

```

Where `enterprise-oid` and `system description` are regular expressions.

The following are two simple examples of OID mappings:

### Example 1:

Using the Enterprise OID as the mapping.

```

MAPPING enterprise
    ENTERPRISE-ID 9\.1\.*
    AV-CLASS-MAPPING ciscoRouter
END-MAPPING

```

The mapping above creates a CEMF object of class **ciscoRouter** if the regular expression is matched with the contents of the **sysObjectID** attribute fetched from an auto discovered device. It is important to note that the **ENTERPRISE-ID** should only contain the portion of the **sysObjectID** from the enterprise identifier (the Cisco identifier in this example).

### Example 2:

Using the System Description as the mapping.

```

MAPPING description
    SYS-DESCRIPTION .*[Ss]parc.*
    AV-CLASS-MAPPING sun
END-MAPPING

```

The mapping above creates a CEMF object of class **sun** (sun workstation) when the regular expression is matched with the contents of the **sysDescr** attribute fetched from an auto discovered device.

***Q. How do I customize the default Performance Manager configuration to keep the Raw and Summary data? (Products affected : CEMF 3.x)***

**A.** There are two files that affect the way Performance Data and Summary Data are shown in the Performance Manager application. The **<CEMFROOT>/config/init/attributeHistoryServer.ini** and the **historyCriteria** files in **<CEMFROOT>/config/dataload/historyCriteria**. Their setup and customization options are shown below.

```
attributeHistoryServer.ini setup:
```

This file defines how much raw (or not summarized) data we store. The default values at the top of the file are as follows:

```

minValueCount = 50
maxValueCount = 1000
minRawDataAge = 60

```

**minValueCount** defines the minimum amount of raw data samples that are kept in the database. Once this number of samples has been reached. There will never be fewer raw samples than this

value.

**maxValueCount** defines the maximum number of raw samples that are kept in the database. At 1:00 a.m. every morning the system checks this value against the number of raw samples available. If it is exceeded then all raw samples between min and max are summarized as per the summary rules in the associated criteria file.

**minRawDataAge** value stops data being lost when there is a very low poll period (for example 30 sec perf polling). Do not change this value.

#### Formulas:

◇ **maxValueCount** = (Number of required days of raw data) x (Number of daily polls)

◇ **minValueCount** = (Number of required raw data samples) x (Number of daily polls)

Ignore the minRawDataAge. Allow for a full midnight-to-midnight period between min and max (i.e. at least two days).

#### Example:

Using a 15 minute polling requirement, with a maximum four days raw data, with two days always available, based on the above example, results in the following:

```
minValueCount = 192
maxValueCount = 384
minRawDataAge = 60
```

This means that raw data collects for day one, two, three, and four. At midnight on day five, raw data older than 192 samples old is summarized. After this, the available raw data samples are saw-toothed between 192 and 384 samples ( with two days or 192 samples always available).

Criteria file setup:

Each type of object (such as linecard, interface, and PVC) has a unique class name. The criteria file for each class shows the summarization rules used during the summarize process described above. It describes how many raw samples are summarized into each summary sample, and also describes how the raw data are summarized. At the end of each criteria file are the following lines:

```
SUMMARYINTERVAL 120
SUMMARYRULE average
SUMMARYRULE peak
SUMMARYRULE trough
```

**SUMMARYINTERVAL** is the period of time over which each summary sample takes raw values (in the above, a summary is performed of all data over a 120 second period).

**SUMMARYRULE** is a list of samples taken over each summary period.

#### Example:

Consider six hour summaries with the average, minimum, and maximum value stored for each period. Combined with the example given for the **attributeHistoryServer.ini**, would mean that you would have 15 minute raw samples for the previous two days, but anything older than two days only shows six hour averages/peaks/troughs. This requires the following set of rules:

```
SUMMARYINTERVAL 21600
SUMMARYRULE average
SUMMARYRULE peak
SUMMARYRULE trough
```

Alternatively, if the above four lines are removed then only raw data is kept. If this is done then no summaries are created at all and any raw data older than the maxValueCount is discarded.

***Q. Can I configure Auto Discovery to name discovered network objects with their DNS name instead of their IP addresses? (Products affected: CEMF 3.x)***

**A.** Auto Discovery, by default, names discovered objects with their IP address, but you can change it to name the objects with their DNS looked up names. For example, if myServer.cisco.com has the IP address 19.19.19.19, then the discovered object (the Cisco EMF managed object that models the discovered device) is named 19.19.19.19. However, if DNS naming is enabled, then the discovered object is named myServer.cisco.com.

**Note:** Objects within the Network View are always named with their IP address, and this is not configurable. An auto discovery of the above object with DNS enabled results in the following containments being created:

```
Network:/19.0.0.0/19.19.19.19
Physical:/myServer.cisco.com
genericObjects:/myServer.cisco.com
```

And without DNS enabled :

```
Network:/19.0.0.0/19.19.19.19
Physical:/19.19.19.19
genericObjects:/19.19.19.19
```

To enable DNS naming, change the following setting in the **<CEMFROOT>/config/init/discoveryGUI.ini** in the [discovery] section to:

```
dnsLookup = 1
```

The updated setting is picked up by the discovery GUI by closing and then re-opening it to enable the **dnslookup** option.

***Q. How can I hide values such as community strings that I enter on the Deployment Wizard GUI? (Products affected : CEMF 3.x)***

**A.** You can use deployment templates and configure data entry fields displayed on the Deployment Wizard to mask (hide) the value entered by the user. As the user types, asterisk characters are displayed in the data entry field.

You achieve this configuration by modifying the prompt specification to include a data entry mode. The data entry mode can be either NORMAL or HIDDEN. Append a comma followed by either NORMAL or HIDDEN to the prompt line. The following excerpt from a deployment template file illustrate how this is done:

```
PARAMETER variable.someVariable
PROMPT "Please enter some Variable: "
DEFAULT "someDefault"
END_PARAMETER
```

This configuration results in a Deployment Wizard data entry field displayed with the prompt text "Please enter some Variable: ." The value entered by the user in response to this prompt is shown in the data entry field. To hide this value the parameter specification is changed to the following:

```
PARAMETER variable.someVariable
PROMPT "Please enter some Variable:", HIDDEN
DEFAULT "someDefault"
END_PARAMETER
```

The Deployment Wizard dialog now displays an asterisk for each character that the user types in as a response to this prompt.

You can only use NORMAL or HIDDEN options for the prompt text entry argument, and NORMAL is the default value. You can only set the data entry mode to HIDDEN for text fields, or for a DEFAULT clause for a text field, where you want the default value masked when displayed. If you set this mode for other field types (for example, integer) you will get a parse error.

**Note:** The asterisk character is hard coded as the masking character and is not configurable.

***Q. Can traps be forwarded to different third party Network Management Systems based on the source of the traps? (Products affected : C EMF 3.x)***

**A.** Traps cannot currently be forwarded on the basis of which object they come from. The decision to forward a trap is based on the enterprise, generic, and specific ids of the incoming trap. Specific combinations of these may be forwarded to a specific address. Therefore, if two different kinds of devices generate different combinations of enterprise, generic, and specific ids, they may then be forwarded to different Network Management Systems. Otherwise, this cannot be done at the moment.

***Q. Can traps from the same device be forwarded to multiple third party Network Management Systems? (Products affected : C EMF 3.x)***

**A.** Yes, this can be done by adding multiple lines to the trap forwarding file, causing traps to be redispached to multiple destinations.

### **Valid Trap Forwarding Configuration File**

The trap forwarding configuration file should be placed in **<CEMF\_ROOT>/config/data/trapForwardFile**. The example below shows a trap forwarding file that forwards traps onto two hosts. The first host receives all the traps regardless of the values of the trap data fields. The second host receives only those traps that have a generic id of 6 and an enterprise OID that is equal to 1.3.6.1.4.1.9.1.14:

```
System Traps

hostname1 -1:-1:

hostname2 6:-1: 1.3.6.1.4.1.9.1.14
```

Notes regarding the format of the above file:

- ◇ The line that specifies the field headings (System Traps) must be present before the forwarding rules.
- ◇ The recipient host can be specified by hostname or IP address.
- ◇ The wildcard value for generic and specific ids is -1.
- ◇ The wildcard value for the enterprise OID is an empty string.
- ◇ The hostname and the trap pattern must be separated by a space character followed by a tab character.

◇ The last forwarding rule line must have a line–return at the end.

### **SNMPv2 Notification Forwarding**

Incoming v2c notifications are dispatched as v1 traps starting with CEMF 3.0.4 Patch 08 (not in CEMF 3.1 FCS). This is due to a design flaw in the SNMPv2c protocol such that v2c traps are forwarded as v1 traps. There is no field in v2c traps for the IP address of the source agents. The address is taken from the IP packet's header. Therefore, the traps sent on by a manager appear to have been raised by the manager itself.

◇ The outgoing v1 traps have the exact varbinds that were on the original v2c trap. This includes varbinds for the timeticks and the trap OID.

◇ The generic id is 6 and the specific id is set to the last number of the trap OID of the v2c trap.

◇ The enterprise field of the outgoing v1 trap has the OID of the original v2c trap OID with the last number removed. Note that the full OID is included within the varbinds.

***Q. If an EM is installed that results in a change to the environment (such as adding a script to config/env), when does this change take affect? (Products affected : CEMF 3.x)***

**A.** This change does not take affect unless CEMF is stopped and restarted.

In V2, environment variables were used to pass settings to CEMF processes during their initialization. These settings were placed in the environment and not in a configuration file because the settings were needed before any configuration file could be read. The only variables originally passed were the port to bind to and the location of the configuration directory. Although other variables were not supported, other variables were added by developers because they are easy to set.

Processes inherit their environment from their parent process. In CEMF the **sysmgr** process is the parent of all server processes. Since the **sysmgr** process cannot be stopped without stopping CEMF, no changes to the environment are picked up until CEMF is restarted.

This was acceptable in V2 since any changes to the system were performed while the system was down. The system would have to be started after the changes were made.

V3 uses self management to make changes to the system while it is still running. This has the effect that the **sysmgr** process never restarts, and so never rereads its environment. Environment variables cannot be used to pass settings around the system. The reliance on the port and configuration environment variables was removed by adding dynamic port allocation, and a static configuration directory setting. All other non–supported settings must be removed from the environment and placed into configuration files.

### **Recommendation**

The **systemConfig** class is the supported method of retrieving settings from configuration files. This class gives type safe access to settings held in a process's configuration file (held in **<CEMFROOT>/config/init**). **systemConfig** settings are read during process initialization and are reread when the process restarts.

Configuration file settings are grouped into sections, and have the form **name=value**. The value can be retrieved by using the appropriate **systemConfig** accessor and giving the setting's section and name.

***Q. How can the 'Rename Object' service be removed? (Products affected : CEMF 3.x)***

**A.** There are a number of ways that a service can be removed, depending on the level of installation.

◇ For a fresh installation:

1. Go to **CEMFROOT/config/partitioning** directory.
2. Go to the **objectGui.partspec** file.
3. Remove the **SERVICE** definition and its entry in the **FEATURELIST**.
4. Start CEMF for the first time.

◇ In a system that is already running CEMF do one of the following:

· Create a new file:

1. Create a new file containing the definition of the **SERVICE** that is to be removed.
2. Run the **partitioningTool** on this file using the delete option (**partitioningTool -d /fullpath/myFile.partSpec**).

This removes from the system only those services that are contained within your new definition file.

· Change an existing file:

1. Change the **SERVICETYPE** in the original file so that the feature is not available on any objects.
2. Run the **partitioningTool** using the **update** option (**-u**).

This will not remove it but does ensure it is not presented to the user.

**Q. How can I change the maximum size of a log file?** (*Products affected : CEMF 2.1.x, 3.x*)

**A.** The default size of a log file is 1Mb. Add an entry for **maxLogfileSize** in the **logger** section of the **.ini** file. Its value specifies the maximum size of the log file for that process in kilobytes.

**Q. How can I modify labels for the attributes in the "Monitored Attributes" list in Performance Manager?** (*Products affected : CEMF 3.x*)

**A.** The default full **section:module.tag** for attributes is not very clear for users. The file directory **<CEMFROOT>/config/dbm/common/** contains files that map attribute names into a format more suitable for displaying to the user. Cisco EMF SDK generates the mapping files in this directory from the attributes that Element Manager GUIs use. The mappings contained within this directory can be used by the Performance Manager and other Applications to display those mapped strings.

You can add additional mapping files by putting them in their Element Managers' **config/user** directory, and specifying the path above for their destination in the user file list. The user can also manually edit the mapping files to change the name that is displayed in the GUI for their attributes (by default Cisco EMF SDK uses the actual attribute name specified in the MIB).

**Note:** The attribute names contained within the directory must be unique, it is the responsibility of the user to ensure that no name clash problems occur, otherwise you may get unexpected attribute name mapping behavior.

**Q. How do I use the objectFileParser tool for CEMF 3.0.1 to import configuration data in terms of sites/groups/devices into CEMF?** (*Products affected : CEMF 3.x*)

**A.** Using an object spec file is simpler than using APIs directly. The format of an object spec file is best demonstrated by examples with comments.

## File 1

```
OBJECT
NAME Example
```

```

IDTYPE 0x0a
CLASS managedObjectTree

ATTRIBUTE contTreeVisibility (1) containment
ATTRIBUTE contTreeAccess (2) containment
ATTRIBUTE contTreeDesc ("Example view") containment
ATTRIBUTE contTreeConnectivity (0) containment
ATTRIBUTE contTreeIsPropagating (1) containment

ENDOBJECT

```

This example creates a containment tree (a view). The IDTYPE clause is an instruction to CEMF to create a different type of object, in this case a containment view. Note that this is one detail of object spec files that may change. In general, for object import, you do not need to use this clause, because your EM has already created the appropriate view where you create objects. The values of the attributes shown is documented in the CEMF–SDK documentation.

**Note:** The containment view definition **MUST** have the NAME clause and the name must be unique in all views.

The following is the format of the attribute clause:

```

ATTRIBUTE <possibly-module-scoped-name> (value) <section>

```

## File 2

```

OBJECT
  CLASS site
  CONTAINMENT Physical:/Site1
  CONTAINMENT Example:/Site1
  ATTRIBUTE AMAF-MGMT-MIB.Comment (A dummy site) LocalDB
ENDOBJECT

OBJECT
  CLASS snmpAgent
  CONTAINMENT Physical:/Site1/Agent One
  CONTAINMENT Example:/Site1/Agent One In Example View
  ATTRIBUTE AMAF-MGMT-MIB.ipaddress (194.131.185.1) LocalDB
  ATTRIBUTE SNMP-ATTRIBUTES-MIB.snmpv1-read-community (public) LocalDB
  ATTRIBUTE SNMP-ATTRIBUTES-MIB.snmpv1-write-community (private) LocalDB
ENDOBJECT

OBJECT
  CLASS snmpAgent
  CONTAINMENT Physical:/Site1/Agent Two
  CONTAINMENT Example:/Site1/Agent Two In Example View
  ATTRIBUTE AMAF-MGMT-MIB.ipaddress (194.131.185.2) LocalDB
  ATTRIBUTE SNMP-ATTRIBUTES-MIB.snmpv1-read-community (public) LocalDB
  ATTRIBUTE SNMP-ATTRIBUTES-MIB.snmpv1-write-community (private) LocalDB
ENDOBJECT

```

This example creates a site object and two SNMP agent objects supporting SNMPv1.

Managed objects **MUST NOT** define the name clause. Instead, they define containment paths, all the way down to the name of the leaf node. If this name is not unique in the context of a level of containment, object creation fails. An object does not need the same leaf node name in all trees.

The order of objects in the file is significant. You must specify the objects in an order that defines parents prior to children. The **objectFileParser** rejects other orders.

Create containment views before you place objects in a view, using a separate run of the **objectFileParser**.

---

## Related Information

- [CEMF Knowledge Base](#)

---

All contents are Copyright © 1992—2002 Cisco Systems, Inc. All rights reserved. Important Notices and Privacy Statement.

---

Updated: Jul 31, 2007

Document ID: 24804

---