



## **Cisco Enterprise Policy Manager DotNet Agent Guide**

Version 3.3.0.0  
April 2009

**Americas Headquarters**  
Cisco Systems, Inc.  
170 West Tasman Drive  
San Jose, CA 95134-1706  
USA  
<http://www.cisco.com>  
Tel: 408 526-4000  
800 553-NETS (6387)  
Fax: 408 527-0883

Text Part Number: OL-19560-01

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCDE, CCSI, CCENT, Cisco Eos, Cisco HealthPresence, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco Nurse Connect, Cisco Stackpower, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0903R)

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

*Cisco Enterprise Policy Manager DotNet Agent Guide*  
© 2009 Cisco Systems, Inc. All rights reserved.



# CONTENTS

## **Preface** v

- Objective v
- Audience v
- Document Organization v
- Document Conventions vi
- Related Documentation vi
- Changes to This Document vi
- Obtaining Documentation and Submitting a Service Request vi

---

## **CHAPTER 1**

### **Introduction** 1-1

- CEPM Overview 1-1
- CEPM Agent in the .NET Application 1-1

---

## **CHAPTER 2**

### **Deploying CEPM Agent in a .NET Application** 2-1

---

## **CHAPTER 3**

### **Deploying the Cisco Agent in a COM Application** 3-1

- Using the COM Agent in VB 3-2
- Using the COM Agent in MFC 3-3

---

## **APPENDIX A**

### **Configuring pep\_config.xml File** A-1

- <cache> A-3
- <type> A-4
- <refresh> A-5
- <interval> A-5
- <prefetch> A-5
- <applicationgroup> A-6
- <application> A-6
- <loadbalance> A-6
- <pdps> A-7
- <http-proxy> A-7
- <apis> A-7
- <adapters> A-8
- <SSL> A-8

APPENDIX B **Configuring log.config File** B-1

APPENDIX C **Load Balancing in PEP** C-1



## Preface

---

This preface describes the objectives, intended audience, and organization and the document conventions in the *Cisco Enterprise Policy Manager DotNet Agent Guide*.

The preface contains the following sections:

- [Objective](#)
- [Audience](#)
- [Document Organization](#)
- [Document Conventions](#)
- [Related Documentation](#)
- [Changes to This Document](#)
- [Obtaining Documentation and Submitting a Service Request](#)

## Objective

This document describes how to deploy the CEPM agent for the .NET applications.

## Audience

This guide is for the administrators who use CEPM and are responsible for resource modelling and entitlement management.

## Document Organization

This guide contains the following chapters and appendixes:

- [Chapter 1, “Introduction”](#)
- [Chapter 2, “Deploying CEPM Agent in a .NET Application”](#)
- [Chapter 3, “Deploying the Cisco Agent in a COM Application”](#)
- [Appendix A, “Configuring pep\\_config.xml File”](#)
- [Appendix B, “Configuring log.config File”](#)

- [Appendix C, “Load Balancing in PEP”](#)

## Document Conventions



### Caution

Means *reader be careful*. You are capable of doing something that might result in equipment damage or loss of data.



### Note

Means *reader take note*. Notes contain helpful suggestions or references to materials not contained in this manual.

## Related Documentation

CEPM\_User\_Guide\_V3.3.0.0.pdf

## Changes to This Document

[Table 1](#) lists the changes made to this document since it was first released.

**Table 1**      **Changes to This Document**

Date	Change Summary
July 7, 2009	Minor edits and template/boilerplate updates for publication to Cisco.com
April 3rd, 2009	Cisco Enterprise Policy Manager (EPM) Release 3.3.0.0

## Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

<http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html>

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.



# CHAPTER 1

## Introduction

---

### CEPM Overview

To meet the demands of security, privacy, and compliance, businesses need to be able to control and audit access to their applications, transactions, content, and data, at a fine-grain level. For example, compliance requires them to maintain "Chinese Walls", enforce separation of duties, and protect the privacy of information assets. Entitlement management, or the ability to consistently administer, enforce, and audit fine-grained access to enterprise assets, has largely been addressed by ad hoc development in different departments by different application owners.

Cisco Enterprise Policy Management (CEPM) addresses this need for consistent and externalized management of authorization decisions with its policy management solution. A typical CEPM deployment consists of the following components:

- The entitlement engine which is a Policy Decision Point (PDP) that evaluates application-specific authorization policies. The PDPs connect with existing information repositories, such as LDAP, AD, databases, and IdM, which are referred to as Policy Information Points (PIPs).
- The administration console, which is a Policy Administration Point (PAP), provides central administration, management, and monitoring of entitlement policies with delegation and integration with an entitlement repository.
- The agent, which is a Policy Enforcement Point (PEP), enforces entitlement policy decisions that are made by the PDP.

This document describes about the architecture of the .NET Agent.

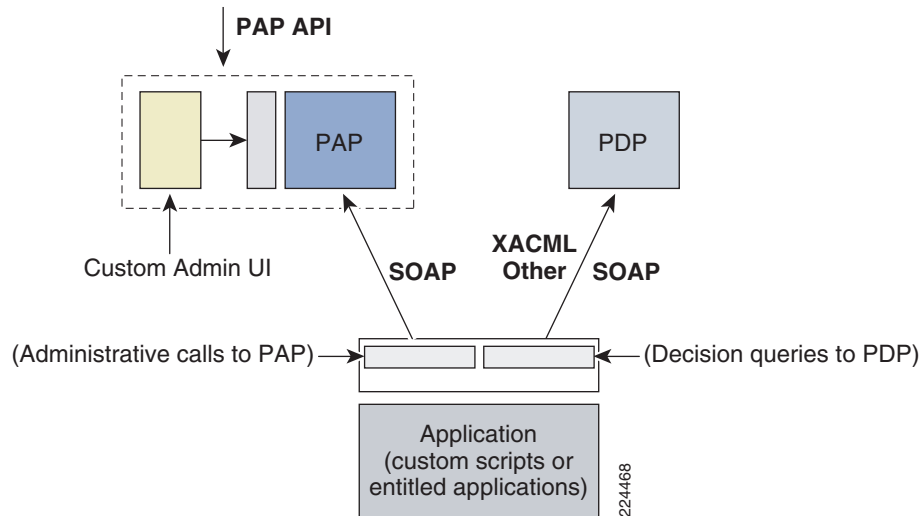
### CEPM Agent in the .NET Application

.NET based applications can be entitled using the PEP for .NET. The agent is a DLL that can be used by any .NET based application (either a desktop or a web-based application). A COM-wrapped agent is also supported for VB, C++, and other Windows-based applications. As with the Java PEPs, the .NET PEP supports libraries for decision and administration APIs.

The two packages of interest are:

- Policy Administration Point (PAP) API
- Policy Decision Point (PDP) API

Figure 1-1 CEPM Deployment Diagram



These packages are intended to be used for customization and easier integration of the components with customer deployments. The following three categories of functionality are supported:

- Creation of custom administrative consoles [**use the PAP APIs**]
- Invocation of decision queries from applications [**use the PDP APIs**]
- Run-time queries to the PAP for creation of automated scripts or policy queries from applications [**use the PEP APIs**]

From V3.3.0.0 onwards, CEPM .NET Agent supports HTTP as transport protocol for PEP-PDP communication. A new set of PEP API has been crafted to support HTTP as well as SOAP. You may find examples elaborating implementation of PEP methods in both old and new authorization model.

Refer to *CEPM Dotnet Developers Guide V3.3.0.0* for more information on using these APIs.



## CHAPTER 2

# Deploying CEPM Agent in a .NET Application

---

CEPM .NET Agent API can be deployed in any .NET application, such as Windows application, console application, and web application.

To deploy the .NET Agent in the existing application:

- 
- Step 1** Unzip the distribution (CEPM\_DotNetAgentV3.3.0.0\_32b\_NCache3.3.zip) in the required directory.
- Step 2** Copy the following files from the unzipped directory into ...\ApplicationName\bin folder (for Windows and Web applications) or into ...\ApplicationName\bin\debug folder (for Console applications):
- Com.Cisco.Agent.Pep.dll
  - Com.Cisco.Epm.Pap.dll
  - Log4net.dll
  - Log4net.xml
  - Log.config
  - pep\_config.xml



**Note** You can download pep\_config.xml file for your application from the PAP UI (choose Home > System Config > Application page and click **Download Agent Config** button for the chosen application). The downloaded pep\_config.xml file may contain only the basic configuration parameters and may not carry the agent related tags. Hence, it is highly recommended to make use of *pep\_config.xml* file that is bundled in the build.)

---



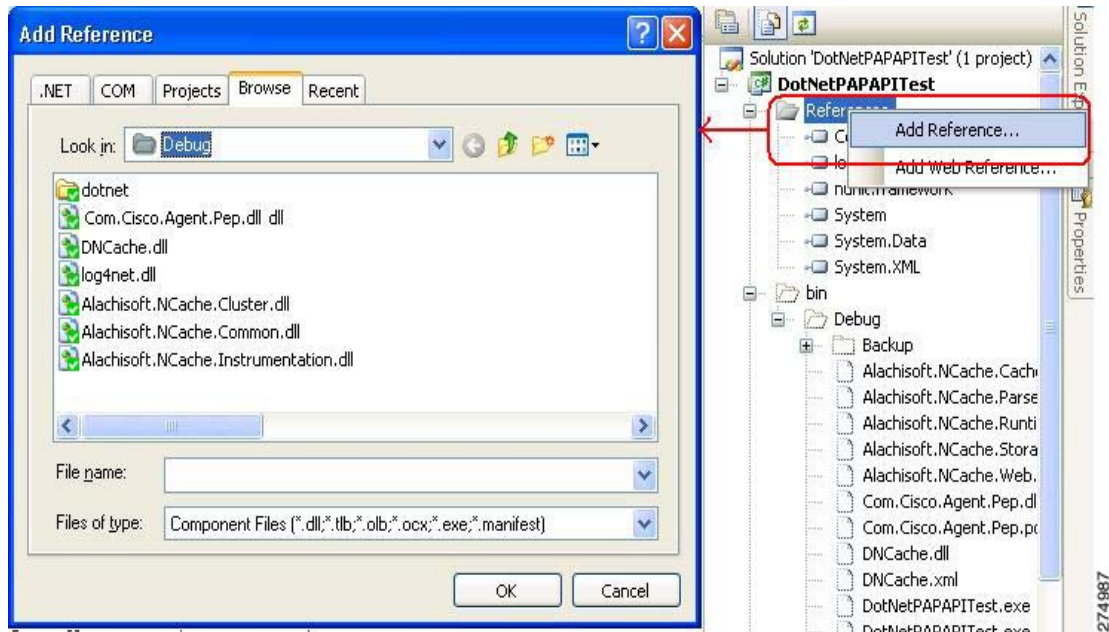
**Note** The pep\_config.xml and log.config files must be updated according to your current requirement before proceeding toward final deployment of the .NET Agent. Refer to [Appendix A, “Configuring pep\\_config.xml File”](#) and [Appendix B, “Configuring log.config File”](#) for information about configuring these files.

---

- Step 3** Launch Visual Studio, choose **File > Open > Projects** and open the required application in which you need to deploy the .NET Agent.

- Step 4** In the **Solution Explorer** window, right-click **References** and click **Add Reference**. The **Add Reference** window appears.

**Figure 2-1 Add Reference**



- Step 5** Click **Browse** to choose the required component.
- Step 6** Locate **Com.Cisco.Agent.Pep.dll** file.
- Step 7** Click **OK**, which deploys the .NET Agent in the selected application.
- Step 8** Add the following piece of code to define entitlements:

To utilize the old PEP APIs (which support only SOAP):

```
IAuthorizationManager manager = SoapAuthorizationManager.GetInstance();
bool result = manager.IsUserAccessAllowed
("Username", "CEPMAgentTest:TestWebApp3:WebForm1", "any");
```

To utilize the new PEP APIs (which support both HTTP and SOAP):

```
IAuthorizationManager manager =
AuthorizationManagerFactory.GetInstance().GetAuthorizationManager();
bool result = manager.IsUserAccessAllowed
("Username", "CEPMAgentTest:TestWebApp3:WebForm1", "any");
```



## CHAPTER 3

# Deploying the Cisco Agent in a COM Application

While dealing with COM applications (for example, applications developed using VC++, ATL, VB, C++, and C), a COM agent (PEP agent for COM) is used. This COM agent is a wrapper that provides a wide range of configuration properties, which can be used to configure the application in appropriate way from the command line.

To deploy the COM agent in a COM application:

---

**Step 1** Convert the dll file, Com.Cisco.Agent.Pep.dll, into a tlb file, Com.Cisco.Agent.Pep.tlb.



**Note**

In the build process of COM agent, create and register a type library (Com.Cisco.Agent.Pep.tlb) of Com.Cisco.Agent.Pep.dll using the following command lines.

```
regasm Com.Cisco.Agent.Pep.dll/tlb <folderpath>/Com.Cisco.Agent.Pep.tlb  
regasm Com.Cisco.Agent.Pep.dll/codebase/tlb <folderpath>/Com.Cisco.Agent.Pep.tlb
```

This type library is created using the Assembly Registration tool (Regasm.exe) so as to enable the COM client to use the Assembly seamlessly. This tool reads the metadata within an assembly and adds the necessary entries to the registry, which allows COM clients to create .NET Framework classes transparently. The Assembly Registration tool can generate and register a type library when you apply the /tlb: option. COM clients require that the type libraries to be installed in the Windows registry. Without this option, Regasm.exe only registers the types in an assembly, not the type library. This type library (Com.Cisco.Agent.Pep.tlb) is created in the same folder where the corresponding dll is located.

**Step 2** Copy Com.Cisco.Agent.Pep.tlb file into the ...\ApplicationName folder of the application.

**Step 3** Copy pep\_config.xml and Log.config files into \ApplicationName\Debug folder.



**Note**

You must set the application name and application group name in the pep\_config file. Other attributes like username, password, and URL related to the PDP must be mentioned in the pep\_config file.

**Step 4** To test whether the deployment is successful or not, refer to the following example, which states how to use the COM agent with VB and VC++.

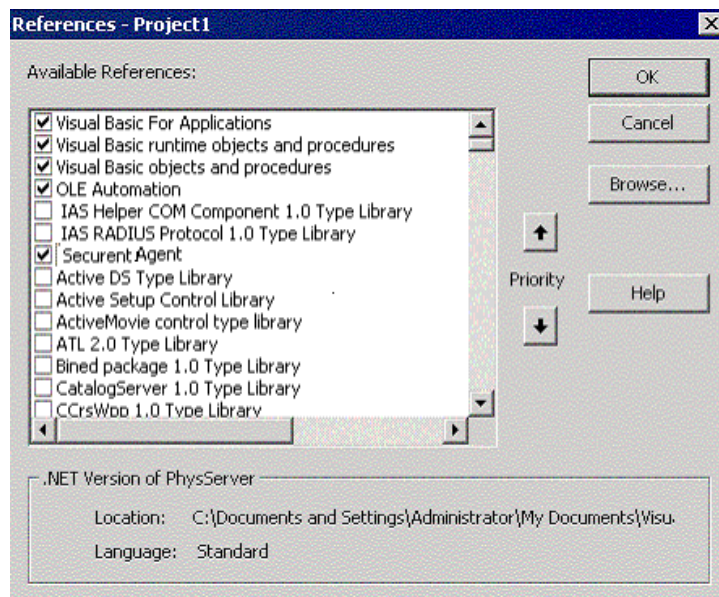
---

## Using the COM Agent in VB

To deploy the COM agent in a VB application:

- Step 1** Open VB console.
- Step 2** Create a new Standard EXE or open an existing application
- Step 3** Choose **Project > References**.
- Step 4** Find your class in the available references list (see [Figure 3-1](#)).

**Figure 3-1** Reference Class



To define entitlement, select a button (for example) and add the following code:

```
Private Sub Command1_Click()
    Dim i As Cepm.SoapAuthorizationManager
    Dim result As Boolean
    Set i = Cepm.SoapAuthorizationManager.GetInstance
    result = com_manager->IsUserAccessAllowed
    ("username", "ApplicationGroupName:ApplicationName:PageName", "any");
End Sub
```

This code deploys the COM agent in your application.



**Note** To test whether the deployment is successful or not, run the VB application. The specified button is rendered only if you are authorized to view it.

## Using the COM Agent in MFC

The implementation of COM agent in MFC includes the #import "CepmAgent3.3.tlb" statement in your header file as well as using namespace CepmAgent.

Open your application in which you want to integrate the agent and copy the following code into the pages or controls for which you want to define entitlement:

```
void CNickDlg::OnOK()
{
    CString strMessage;
    CoInitialize(NULL);
    CEPM::IAuthorizationManager *com_manager;
    CEPM::IAuthorizationManagerPtr
    p1(__uuidof(SoapAuthorizationManager.GetInstance));
    com_manager = p1;
    VARIANT_BOOL result =
    com_manager->IsUserAccessAllowed("username", "ApplicationGroupName:ApplicationName:PageName", "any");
    strMessage.Format("Result is: %d", result);
    MessageBox(strMessage, "Result", MB_ICONASTERISK | MB_ICONINFORMATION);
}
```

This code deploys the COM agent in your MFC application.



---

**Note** To test whether the deployment is successful or not, run the MFC application. The specified button is rendered only if you are authorized to view it.

---





# APPENDIX A

## Configuring pep\_config.xml File

This appendix describes the structure of the PEP configuration file (*pep\_config.xml*) that is embedded into the protected .NET application to avail fine grained authorization provided by CEPM and gives guidelines for configuring it to suit the requirements of your application.

The following is a sample *pep\_config.xml* file:

```
<pep_config version="3.3">
<cache decisionCacheEnabled="true" cacherefreshtype="onlyupdated" decisionsOnly="false"
provider="net.securent.agent.sdk.cache.CacheProvider"
implementor="Com.Cisco.Epm.Cache.Impl.NCache"
eventProvider="Com.Cisco.Epm.Cache.Event.EventProvider" >
  <type>TTL</type>
  <refresh enable="false">update</refresh>
  <interval>20</interval>
  <prefetch enable="false" type="user" bulkUsersPerRequest="10">
    <prefetchForApis>
      <api name="isUserAccessAllowed">
        <!-- Prefetch based on message attributes -->
        <!-- Attribute for Context -->
        <!--<message-attribute name="urn:cisco.cepm:3.3:xacml:context-name"
override="true">TestAppGrp:TestApp:Context1</message-attribute> -->
        <!-- Attribute for Role Bundle -->
        <!--<message-attribute name="urn:cisco.cepm:3.3:xacml:rolebundle-name"
override="true">Rb1</message-attribute>-->
        <!-- Attribute for Resource Type -->
        <!--<message-attribute name="urn:cisco.cepm:3.3:xacml:resource-type-name"
override="true">TestAppGrp:TestApp:ResType</message-attribute>-->
        <!-- Attribute for any attributes -->
        <!--<message-attribute name="Key1" override="true">Value1</message-attribute>-->
      </api>
      <!-- api name="getDecisionsByResourceTypeForAnyAction"/-->
    </prefetchForApis>
    <!-- Selective prefetch for configured Groups, Roles, Resources entities-->
    <groups>
      <!-- FQN of GroupName -->
      <!--<group>TestAppGrp:TestApp:Grp1</group> -->
    </groups>
    <roles>
      <!-- FQN of RoleName -->
      <!-- <role>Prime group:Prime portal:Internal Dev:Internal Dev Tokyo</role> -->
    </roles>
    <resources>
      <!-- FQN of ResourceName -->
      <!-- <resource>TestAppGrp:TestApp:Res3</resource> -->
    </resources>
  </prefetch>
</applicationgroup>Prime group</applicationgroup>
```

```

    <application>Prime portal</application>
</cache>

<logs enable="false" records="20" logsttl="10"/>

<loadbalance enabled="false">
  <algorithm>roundrobin</algorithm>
  <refreshtime>10</refreshtime>
  <timeout>10000</timeout>
</loadbalance>
<pdp>
  <pdp>
    <protocol>soap</protocol>
    <username>admin</username>
    <password>h1BYu+lcwM=</password>
    <url>http://localhost:8080/pdp/services/AuthorizationService</url>
    <timeout>1000</timeout>
  </pdp>
  <!--<pdp>
    <protocol>http</protocol>
    <username>admin</username>
    <password>h1BYu+lcwM=</password>
    <url>http://localhost:8080/pdp/AuthorizationEndPoint</url>
    <timeout>10000</timeout>
  </pdp-->
</pdp>

<http-proxy>
  <host></host>
  <port></port>
</http-proxy>

<apis>
  <api>
    <url>http://localhost:8080/cepm</url>
    <username>superuser</username>
    <password>h1BYu+lcwM=</password>
    <repositoryname>Default Domain</repositoryname>
    <timeout>10000</timeout>
  </api>
</apis>

<adapters>
  <soap>Com.Cisco.Epm.Soap.SoapTransportAdaptor</soap>
  <http>Com.Cisco.Epm.Http.HttpTransportAdaptor</http>
</adapters>

<ssl truststoreFile="C:\OpenSSL\bin\PEM\demoCA\keypair.p12"
truststorePass="changeit"></ssl>

  <dnocache>
    <cachepath>C:\DotnetCache\DNCache.xml</cachepath>
    <cachename>Cache</cachename>
  </dnocache>
  <ncache>
    <cachename>TestCache</cachename>
  </ncache>
</pep_config>

```

## <cache>

CEPM supports caching for the .NET application using NCache and DNCache. To avail the caching facility for your .NET application, you must put *DNCache.dll* and *dncache.xml* files (which are available in the distribution) in the ...ApplicationName\bin\debug folder. Caching works only for the API methods `IsUserAccessAllowed`, `IsRoleAccessAllowed`, `IsGroupAccessAllowed`, and `GetDecisions` and their overloaded methods. The configuration to make use of the cache is done in the following tag:

```
<cache decisionCacheEnabled="true" cacherefreshtype="onlyupdated" decisionsOnly="false"
provider="net.securent.agent.sdk.cache.CacheProvider"
implementor="Com.Cisco.Epm.Cache.Impl.NCache"
eventProvider="Com.Cisco.Epm.Cache.Event.EventProvider" >
```

Update this tag by mentioning the provider and the implementor details. If you set this tag to false, caching is disabled.

The cache tags and their corresponding parameters can be updated according to the user requirement. A brief explanation of various parameters and child tags to be updated for caching follows.

- decisionCacheEnabled attribute

To enable the PEP cache mechanism, this attribute must be set to true.

- cacheRefreshType attribute

This attribute value can be set to `onlyupdated` or `all`. When set to `onlyupdated`, then during the cache refresh process, only the changed data is copied from the PDP and is updated in the PEP cache. When set to `all`, then during the cache refresh process, all the data from the PDP cache gets copied to the PEP cache.

- decisionOnly attribute

This is a Boolean expression. If set to false, it stores the entire cache objects which contain decision, decision lifetime, and every entity attributes related to the decisions. If set to true, it stores only the decisions. Because the lifetime will not be persistent, the cache for any policy will not be updated when its lifetime expires. It is important to note that for better performance, it is highly recommended to set this attribute to true.

- Provider attribute

Keep this tag unchanged.

- implementor attribute

The value of this implementor class is set according to the type of cache used. For DNCache, the implementor class must be set to:

```
Com.Cisco.Epm.Cache.Impl.DNCacheImpl
```

To enable DNCache for your PEP, you must install the DNCache in your system and create your own server instance.

For NCache, the implementor class must be set to:

```
Com.Cisco.Epm.Cache.Impl.NCache
```

### Enabling NCache for PEP



**Note** The following procedures assume that you have NCache installed on your system.

To enable NCache:

- 
- Step 1** Go to **Start > All Programs > NCahce > NCache Manager**.
  - Step 2** In the NCacheManager console, create a new project by clicking **File > New > Project**.
  - Step 3** In the NCache Explorer (left navigation pane), right click **NCacheManagement** and click **Create New Cache**. The New Cache Wizard appears.
  - Step 4** Enter the server name of the system where this new cache is created. Click **Next**.
  - Step 5** Enter the Cache Name in *web.config* file in the following format:
 

```
<appSettings>
  <add key="CacheName" value="cepmcache" />
</appSettings>
```

The same name must be mentioned in *client.nconf* file. Click **Next**.
  - Step 6** Set the heap size. Click **Next**.
  - Step 7** Follow the wizard guidelines for setting up advanced options.
  - Step 8** Click **Finish**. The new cache is displayed in the NCache Explorer.
  - Step 9** Right click the newly created cache and click **Start**. This starts the selected cache instance.
  - Step 10** Open *pep\_config.xml* file and update the <cache> tag by setting the value of implementor class as:
 

```
net.securent.admin.sdk.cache.impl.DNCacheImpl
```

Save and close the config file.
  - Step 11** Copy *client.nconf* file into the classpath of the PEP. Update this file by entering the server name and cache ID as configured in Step 3 and Step 4. Save and close the file.
- 

### Configuring DNCache for PEP

To enable DNCache for your PEP, you must add the <appSettings> tag (key-value pairs) in your *application.config* file (or *web.config* file for web application). A sample tag is given here:

```
<appSettings>
  <add key="CachePath" value="C:\DotnetCache\DNCache.xml" />
  <add key="CacheName" value="Cache" />
</appSettings>
```

You must replace the values in this tag as per your requirements

The above values must match with the values given in the DNCache tag in *pep\_config.xml* file. For example:

```
<dnocache>
  <cachepath>C:\DotnetCache\DNCache.xml</cachepath>
  <cachename>Cache</cachename>
</dnocache>
```

<type>

This tag can be set to TTL (time to live) or session.

Set its value to TTL.

## <refresh>

The enable attribute value can be set to true or false. If set to false, PEP will never refresh its cache (except during start-up if the <prefetch> element's enable attribute is set to true). If set to true, the PEP refreshes its cache after every few second's time interval as specified in the <interval> element (as explained in the <interval> element below).

The <refresh> element can be set to invalidate or update. Setting <refresh> to invalidate erases the cache during the refresh cycle. Setting <refresh> to update, updates the PEP cache according to the value set for cacherefreshtype attribute (as explained earlier).

If the PEP is deployed in the clustered cache mode, then if the <refresh> element's enable attribute value is set to true, the PEP acts as a primary cache for the other PEPs in the same cluster. As primary cache, PEP refreshes its cache with the changed data and also refreshes all the PEP caches that are present in the same cache cluster. If this attribute value is set to false, then the PEP acts as a secondary cache and it never refreshes its own cache, but depends upon the primary cache PEP/PDP to refresh its cache.

## <interval>

This tag defines the periodicity (in seconds) of cache refresh activity. For example, if you want to refresh the cached data after every 60 seconds, then set <interval> value to 60.

## <prefetch>

This element is configured to prefetch the policy information and store in the PEP cache, when the PEP component gets loaded in Java Virtual Machine (JVM).

- enable attribute

The enable attribute value can be set to true or false. If set to true, all the data is refreshed during the PEP startup. If set to false, the PEP does not refresh its data during startup.

- type attribute

The type attribute can be set to user or resource.

If set to user, then all the resources belonging to all the users are refreshed in the cache. The user value can be set when there are fewer number of users compared to the number of resources, thus minimizing the number of API calls and reducing network traffic.

If set to resource, then all the users belonging to all the resources are refreshed in the cache. The resource value can be set when there are few number of resources compared to the number of users.

- bulkUsersPerRequest attribute

The bulkUsersPerRequest attribute value is applicable only when the type attribute value (described earlier) is set to user. The intention is to reduce the network traffic. The bulkUsersPerRequest attribute value specifies the batch size of the number of users for which the prefetch will be done.

For example, if there are total of 100 users and the bulkUsersPerRequest attribute value is set to 10, the PDP fetches the policy information (allowed/denied resources) of the 100 users in the batches of 10 users each because the policy information is fetched in ten batches, i.e. tenrequests.

<prefetch> can have multiple <api> subelements. Each <api> subelement contains a PEP API method name to call during prefetch operation as per the requirement.

- <prefetchForApis>

This element defines the selective prefetch mechanism to filter out with respect to the groups, roles and resources.

- **<groups>**  
This element is configured to prefetch the users that belong to the configured group. For example there are 10 users (User1.....to User10). User1 to User5 are mapped to Group1. Specify the **<group>** that is (**<group>SampleGroup:SampleApplication:Group1</group>**).The **<prefetchForApis>** fetches only 5 users (User1 to User5), since only 5 users are mapped to Group1.
- **<roles>**  
This element is configured to prefetch the users that belong to the configured role. For example there are 10 users (User1.....to User10). User1 to User5 are mapped to Role2. Specify the **<role>** that is (**<role>SampleGroup:SampleApplication:Role2</role>**).The **<prefetchForApis>** fetches only 5 users (User1 to User5), since only 5 users are mapped to Role2.
- **<resources>**  
This element is configured to prefetch the users that belong to the configured resource. For example there are 10 users (User1.....to User10). User4 to User9 are mapped to resource Send Trades. Specify the **<resources>** that is (**<resource>Prime group:Prime portal:Send Trades</resource>**).The **<prefetchForApis>** fetches only 6 users (User4 to User9), since only 6 users are mapped to Send Trades.  
  
The **<prefetchForApis>** element works in conjunction with **<groups>**,**<roles>** and **<resources>**. For example there are 10 users (User1.....to User10). User1 to User5 are mapped to Grp1.User3 to User7 are mapped to role1 and User4 to User9 are mapped to resource Send Trades.The **<prefetchForApis>** fetches only 2 users (User4 and User5), since only 2 users are commonly mapped to Grp1,role1 and Send Trades (Grp1 union Role1 union SendTrades).

## <applicationgroup>

Set this tag value to the application group name for which the PEP is deployed.

## <application>

Set this tag value to the application name for which the PEP is deployed.

## <loadbalance>

When the enabled attribute value is set to true, the PEP component implements the load-balancing mechanism while referencing various PDPs. If set to false, the load-balancing mechanism is not implemented.



### Note

When multiple PDPs are associated with a single PEP, the load-balancing mechanism can be used. (Refer to the *Cisco Enterprise Policy Manager Developer Guide* for more information about the load-balancing mechanism in the PEP.)

The **<loadbalance>** element contains the following subelements:

- **<algorithm>**  
This subelement decides the type of algorithm to run while the load-balancing mechanism is implemented. Set it to roundrobin.
- **<refreshtime>**

This subelement decides the time interval (in minutes) after which PEP should check the status of all the PDPs configured in the <pdps> element and refresh its cache with the individual PDP's status (that is active or inactive).

- <timeout>

This element decides the time interval (in milliseconds) for which the PEP component should wait for the response after making a request to PDP, so as to assess PDPs status as active or inactive. For example, if the value is set to 1000, when the PEP makes a request to PDP for status check and if the PEP does not receive response within 1000 milliseconds, the PEP sets the status of that PDP to inactive in its own cache.

## <pdps>

This tag contains the configuration details for all the PDPs that need to be referenced from this PEP component. Each PDP configuration needs to be provided in a separate <pdp> tag.

The following child tags that should be configured for each PDP entry:

- <protocol>

This tag decides the protocol for PEP-PDP communication. The .NET PEP Agent uses SOAP or HTTP protocol to communicate with the PDP.

- <username>

This tag contains the user name value to connect to the PDP.

- <password>

This tag contains the password value for the above mentioned user name.

- <url>

This tag contains the URL of the PDP service.

- <timeout>

This tag decides the time interval (in seconds) for which PEP should wait for the response after making a request to PDP. If a response is not received by the PEP from the PDP in this configured time interval, the PEP considers it as an error request.

## <http-proxy>

This tag contains the configuration details for the proxy server. If proxy server is not being used, then give the empty values for its child tags. The following child tags should be configured to connect to the proxy-server.

- <host>

This tag contains the machine IP address of the proxy server.

- <port>

This tag contains the port number for the proxy service.

## <apis>

This tag contains the configuration details of the PAP server. These values are used by the PEP API to access the PAP application. The following child tags that should be configured to access the PAP server.

- <url>

This tag contains the URL of the PAP server.

- <username>

This tag contains the user name to connect to the PAP server.

- <password>

This tag contains the password for the above user.

- <repositoryname>

This tag contains the name of the repository to connect to in the PAP server.

## <adapters>

This tag contains the name of the adapter that is used by the PEP application to access PDP. .NET PEP uses SOAP and HTTP protocols to communicate with PDP. Do not update this value, which are by default set as shown below.

```
<adapters>
<soap>Com.Cisco.Epm.Soap.SoapTransportAdaptor</soap>
<http>Com.Cisco.Epm.Http.HttpTransportAdaptor</http>
</adapters>
```

## <SSL>

If you enable SSL for PEP-PDP communication, use this tag to set the directory path of the truststore certificate and the truststore password.



# APPENDIX **B**

## Configuring log.config File

---

This appendix describes the structure of PEP log configuration file (log.config) and guides you through the procedures of configuring it.

The following is a sample log.config file.

```
<configuration>
  <configSections>
    <section name="log4net"
      type="System.Configuration.IgnoreSectionHandler" />
  </configSections>
  <log4net>
    <appender name="RollingLogFileAppender"
      type="log4net.Appender.RollingFileAppender">
      <param name="File" value="DotnetAgent.log" />
      <param name="AppendToFile" value="true" />
      <param name="MaxSizeRollBackups" value="10" />
      <param name="MaximumFileSize" value="10MB" />
      <param name="RollingStyle" value="Size" />
      <param name="StaticLogFileName" value="true" />

      <layout type="log4net.Layout.PatternLayout">
        <param name="Header" value="\r\n\r\n-----\r\n" />
        <param name="Footer" value="\r\n-----\r\n\r\n" />
        <param name="ConversionPattern" value="%d [%t] %-5p -%m%n" />
      </layout>
    </appender>
    <appender name="ColoredConsoleAppender"
      type="log4net.Appender.ColoredConsoleAppender">
      <mapping>
        <level value="ERROR" />
        <foreColor value="White" />
        <backColor value="Red" />
      </mapping>
      <mapping>
        <level value="DEBUG" />
        <backColor value="Green" />
      </mapping>
      <mapping>
        <level value="INFO" />
        <foreColor value="White" />
      </mapping>
      <layout type="log4net.Layout.PatternLayout">
        <param name="ConversionPattern" value="%-5p: %m%n" />
      </layout>
    </appender>
  </root>
  <level value="ALL" />
  <appender-ref ref="RollingLogFileAppender" />
</configuration>
```

```
        <appender-ref ref="ColoredConsoleAppender" />
    </root>
</log4net>
</configuration>
```

The following are the general configurations that can be made in the log.config file.

### 1. Updating the File Appender

- Change the name of the log file (named as DotnetAgent.log in the sample file), update the Name parameter in the <Appender> tag.
- Change the maximum file size for back up, update the MaximumFileSize parameter with the required value. This enables the application to create back up on exceeding the specified size.
- Change the header and footer style of the log file, update the Header and Footer parameters in the <layout> tag.

### 2. Updating the Console Appender

- Change the colors of individual log levels, update the corresponding <mapping> tags available for different log levels.

### 3. Updating the Root

- Update the <Root> tag to specify which appender to be available in the log file. If you want to view both appenders, keep this tag unchanged. If you want to keep only one appender, then uncomment the other. For example, if you want to keep only the File appender then comment the Console Appender and if you want to keep only the Console appender then comment the File Appender.

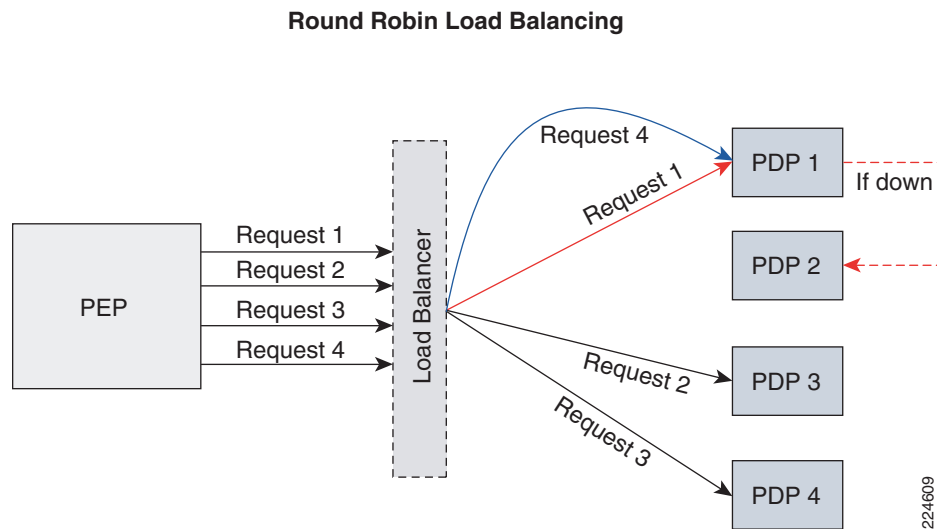


## Load Balancing in PEP

When multiple PDPs are configured to communicate with the PEP, the request- response time may be considerably higher because the PDP is down or the PDP is busy handling other requests. Load-balancing addresses this problem by allowing the protected application server to distribute the load to different PDPs based on a load-balancing policy or algorithm that is configured for the PEP.

The CEPM implements the round robin load-balancing policy.

**Figure C-1** Loadbalancing in PEP



When the PEP sends a request to the PDP, the load-balancer component decides to send the request to the first PDP registered within the cluster. If the specified PDP is down (not alive) or busy handling any other request, the request is redirected to the second PDP as per the round robin policy configured in the PEP component.

In this way, when the last PDP (PDP 4 in the diagram) is alive and responds to the PEP Request 3, the very next request (Request 4) is transferred to the first PDP (that is PDP 1) as per the round robin algorithm.

Unlike in Java PEP, the load-balancing process for the .NET PEP is not protocol-independent. The rule is implemented within any number of the PDPs having SOAP as the transport protocol.

