CA BEST PRACTICES

CA IT Process Automation Manager Best Practices

Scalability Guidelines

DRAFT DOCUMENT – FEEDBACK WELCOME!

Transforming IT Management

LEGAL NOTICE

This publication is based on current information and resource allocations as of its date of publication and is subject to change or withdrawal by CA at any time without notice. The information in this publication could include typographical errors or technical inaccuracies. CA may make modifications to any CA product, software program, method or procedure described in this publication at any time without notice.

Any reference in this publication to non-CA products and non-CA websites are provided for convenience only and shall not serve as CA's endorsement of such products or websites. Your use of such products, websites, and any information regarding such products or any materials provided with such products or at such websites shall be at your own risk.

Notwithstanding anything in this publication to the contrary, this publication shall not (i) constitute product documentation or specifications under any existing or future written license agreement or services agreement relating to any CA software product, or be subject to any warranty set forth in any such written agreement; (ii) serve to affect the rights and/or obligations of CA or its licensees under any existing or future written license agreement or services agreement relating to any CA software product; or (iii) serve to amend any product documentation or specifications for any CA software product. The development, release and timing of any features or functionality described in this publication remain at CA's sole discretion.

The information in this publication is based upon CA's experiences with the referenced software products in a variety of development and customer environments. Past performance of the software products in such development and customer environments is not indicative of the future performance of such software products in identical, similar or different environments. CA does not warrant that the software products will operate as specifically set forth in this publication. CA will support only the referenced products in accordance with (i) the documentation and specifications provided with the referenced product, and (ii) CA's then-current maintenance and support policy for the referenced product.

Certain information in this publication may outline CA's general product direction. All information in this publication is for your informational purposes only and may not be incorporated into any contract. CA assumes no responsibility for the accuracy or completeness of the information. To the extent permitted by applicable law, CA provides this document "AS IS" without warranty of any kind, including, without limitation, any implied warranties of merchantability, fitness for a particular purpose, or non-infringement. In no event will CA be liable for any loss or damage, direct or indirect, from the use of this document, including, without limitation, lost profits, lost investment, business interruption, goodwill or lost data, even if CA is expressly advised of the possibility of such damages.

COPYRIGHT LICENSE AND NOTICE:

This publication may contain sample application programming code and/or language which illustrate programming techniques on various operating systems. Notwithstanding anything to the contrary contained in this publication, such sample code does not constitute licensed products or software under any CA license or services agreement. You may copy, modify and use this sample code for the purposes of performing the installation methods and routines described in this document. These samples have not been tested. CA does not make, and you may not rely on, any promise, express or implied, of reliability, serviceability or function of the sample code.

TITLE AND PUBLICATION DATE:

CA IT Process Automation Manager Best Practices – Scalability Guidelines
Draft Document - Last Update: April 16, 2010

ACKNOWLEDGEMENTS

Principal Authors and Technical Editors

Gordon Brandyburg
Farid Charkhian
George Curran
Anders Magnusson
Alex Moscoso
Terry Pisauro

CA PRODUCT REFERENCES

This document references the following CA products:

- CA IT Process Automation Manager™ (CA IT PAM)
- CA Embedded Entitlements Manager (CA EEM)

FEEDBACK

Please email us at impcdfeedback@ca.com to share your feedback on this publication. Please include the title of this publication in the subject of your email response. For technical assistance with a CA product, please contact CA Technical Support at http://ca.com/support. For assistance with support specific to Japanese operating systems, please contact CA at http://www.casupport.jp.

Contents

Chapter 1: Introduction

CA IT Process Automation Manager (CA IT PAM) provides a centralized and structured approach to operations management by enabling you to define, build, orchestrate, manage, and report on automated processes spanning across different teams and roles in your organization. By automating routine administrative tasks, CA IT PAM improves operational efficiency and incident response handling, and ensures best practice and regulatory controls compliance.

This document is one in a series of papers providing best practices for making the most of your CA IT PAM implementation. The focus of this paper is on scalability. It uses sample architectures to identify hardware recommendations and implementation guidelines based on size and scope of the deployment (small, medium, large, centralized or decentralized, etc.) and anticipated capacity.

Chapter 2: Components and Design Concepts

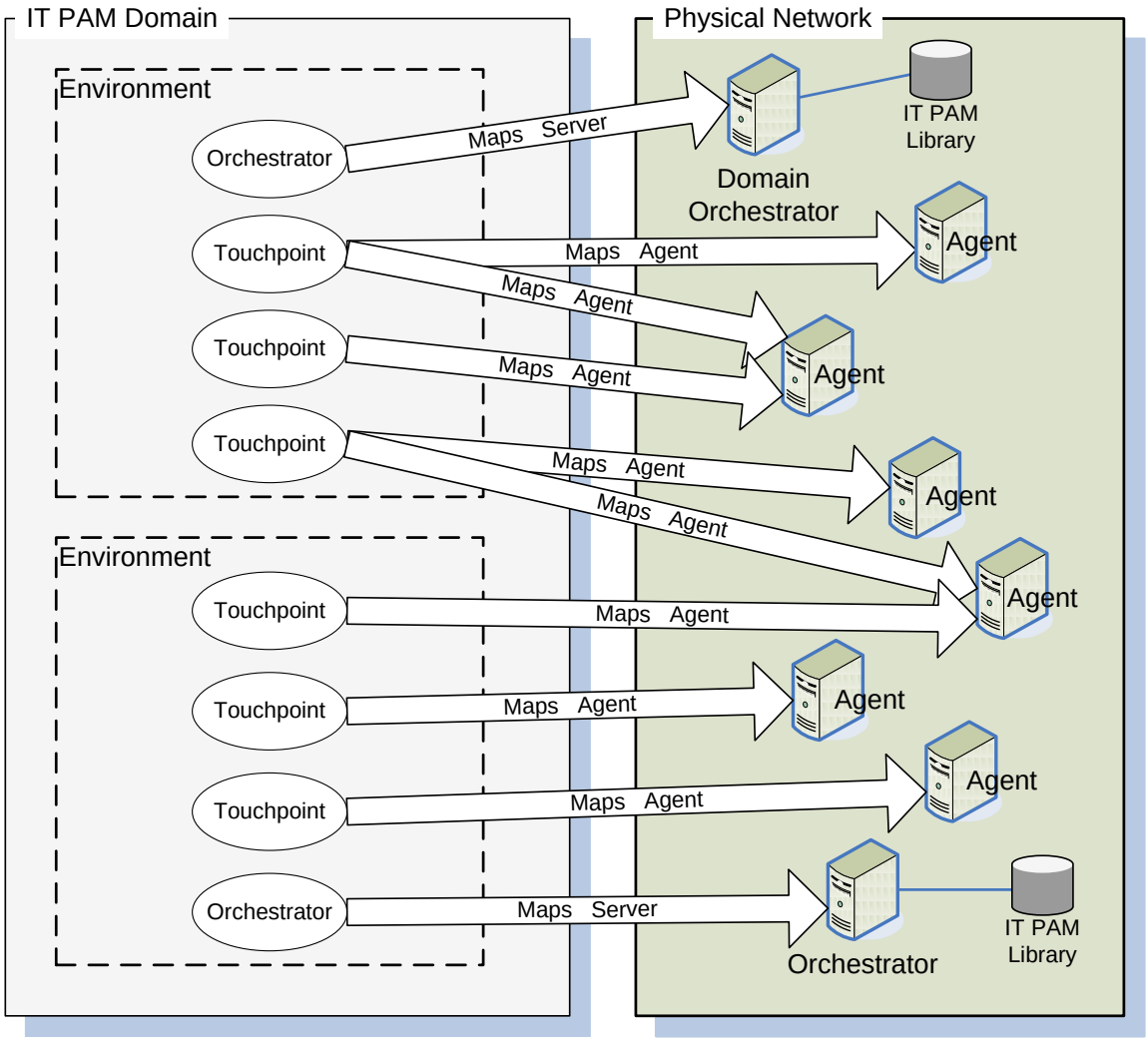
This chapter provides an overview of the CA IT PAM components and their associated sizing considerations. The sizing guidelines provided in this chapter can be used to establish several template architectures – essentially blue prints for a small, medium and large environment deployment. Template architectures utilize a modular approach, establishing a base model that functions as a building block for larger models. Growth from small to medium to large template architectures is managed through the addition of subsequent building blocks.

Information on fault tolerance considerations, such as deployment in a cluster environment, and Disaster Recovery planning are provided in a separate *CA IT PAM Best Practices for Continuous and High Availability* document. Information on port usage for firewall environments is provided in the *CA IT PAM Securability* document.

Let us first take a look at the CA IT PAM architecture and the components that comprise it.

CA IT PAM Components

The CA IT PAM architecture consists of the following components:



- **IT PAM Domain**

  The IT PAM Domain provides the outmost process boundary for IT PAM in the enterprise network.

- **IT PAM Orchestrator**

  Orchestrators support and direct tasks across multiple computers in the environment. Each Orchestrator manages its own Library (called the "IT PAM repository") which contains automation object and state information. One Orchestrator or Orchestrator cluster in the Domain is designated as the **Domain Orchestrator**.

- **IT PAM Environment**

  IT PAM Environments form configuration and operations boundaries around a group of Touchpoints and Orchestrators.

- **IT PAM Touchpoints**

  Touchpoints provide a symbolic reference which maps to one or more IT PAM Agents in an environment. Multiple Agents may be mapped to a single Touchpoint to support failover and load balancing. A Touchpoint exposes IT PAM modules running on the associated-host computer. A Proxy Touchpoint is hosted by an Agent that maps to a remote machine with no Agent daemon running on it.

- **ITPAM Touchpoint Groups**

  You can optionally create your own named groups to group Touchpoints functionally or logically. Logically, Touchpoint groups allow you to organize related Touchpoints and browse more easily among Touchpoints in an Environment. Functionally, Touchpoint groups allow commands and Operators to operate on all Touchpoints in the group.

- **IT PAM Agent\Proxy Agent**

  Configures and runs Agent modules on a single host computer. The Agent modules run tasks on the host computer.

- **IT PAM Client \ User Interface**

  Web based user interfaces, one light weight HTTP based interface that only require an ordinary web browser to provide easy and quick access to the most common tasks. Some more advanced functions such as creating new or modifying existing automation flows requires a thin web distributed java based client, this user interface allow you full access to all functions within CA IT PAM.

In addition, IT PAM modules provide the executable software that implements the functionality defined by IT PAM automation objects. IT PAM modules run on an IT PAM Orchestrator or an Agent host machine. An Orchestrator runs Orchestrator modules in addition to all of the Agent modules.

Domains

At the top level of the CA IT PAM system hierarchy is the **Domain**. A CA IT PAM Domain is defined by all components associated with a single logical **Domain Orchestrator**; however this logical Domain Orchestrator can be comprised of multiple clustered nodes**.**

How Many Domains?

There are several considerations for using multiple domains, including:

- Geographical distribution
- Organizational and/or functional separation
- Separating development, testing and production environments
- Use of different version of CA IT PAM

If a machine will be associated with more than one domain it needs to have multiple installations of the CA IT PAM agent – one agent installed from each Domain Orchestrator.

Note: As long as one common version of CA IT PAM is used a logical separation of the environment (geographical, functional or by organization) can also be done by using separate Environments within the domain, however all components within the domain will regularly communicate with the Domain Orchestrator.

Orchestrators

Orchestrators serve as the "engines" for CA IT PAM processes. Orchestrators run the CA IT PAM server software, determining which tasks should be run, where they should run and when they should run. Each Orchestrator can have its own configuration for CA IT PAM Modules.

Each Orchestrator or Orchestrator cluster also maintains its own distinct Library, which is usually hosted on the company's enterprise database system. The Libraries are where all CA IT PAM automation objects, such as Processes, are stored.

When the first CA IT PAM Orchestrator is installed it automatically becomes the **Domain Orchestrator.** The Domain Orchestrator manages user authentication for the Domain and stores configuration and state information for the Domain. A Domain Orchestrator has the same functionality as any other Orchestrator except that it maintains and distributes the master configuration to other IT PAM components. It cannot be deleted.

Access to Orchestrators is controlled through an existing CA EEM, LDAP or Active Directory system. Orchestrators can share a common Directory Server or they can each use a different Directory Server. Although multiple Directory Servers are typically synchronized, this is not a requirement for CA IT PAM.

Installation of all other CA IT PAM components – agents as well as other Orchestrators – is managed and configured through the Domain Orchestrator. When subsequent Orchestrators are installed each automatically tries to connect to the Domain Orchestrator to identify itself.

All Orchestrators (Domain or standard) can be installed as standalone or clustered servers.

How Many Orchestrators?

After you install the IT PAM Domain Orchestrator you can add additional Orchestrators on other host machines in the network. An IT PAM Server is typically added to maintain an IT PAM Library on a Database Server as well as to run, schedule, and monitor processes and operations on Touchpoints in an Environment. Sometimes a single Server manages operations for all Touchpoints in an environment. At other times, multiple servers may provide for load balancing and failover for an environment.

Environments

Environments are logical segments of the ITPAM Domain. They are typically used to separate release cycle, organizational, functional or geographical areas. For example, you may have a separate "Development" and "Production" Environment. Each Environment must contain one or more Orchestrators; however, each Orchestrator can only be part of one Environment. In multiple tenancy situations you could have multiple Environment which share processes.

While there is no limit to the number of Orchestrators you can add to the Domain, each Orchestrator is limited to participating in a single Environment within the Domain.

How Many Environments?

Because you can define multiple Environments within a Domain, and Agents can be part of multiple environments, you can group computers together in various combinations. In this way, you can use Environments to logically distribute management tasks within different organizational configurations. For example, your organization may be divided into groups such as marketing, development, and human resources. You could create an Environment for each group, in order to run a common set of processes.

Environments have the ability to inherit the Domain's configuration or to override it with configuration specific for each Orchestrator. The same is true at the Orchestrator level. This flexibility gives you an unparalleled level of control over the access to the CA IT PAM system and should be considered as you determine how you want to define your Environments.

Note: In addition to the ability to restrict access to Domains, Environments and Orchestrators by user/group/role, you can also set very granular Permissions (or Access Controls) for any folder or object in the CA IT PAM Library.

Agents and Touchpoints

Touchpoints are the logical representation of "managed resources" in your IT environment. For example, you may have Touchpoints that represents your SAP Server machine, your Primary & Secondary Exchange Servers, and your SQL Server cluster. Touchpoint groups specify any number of IT PAM Touchpoints on which to simultaneously run operations. Touchpoint groups are commonly used to run operations on many similarly configured machines.

Touchpoints are "associated" with these physical systems through Agents. By using this logical representation, you can develop Processes that can be easily exported to different environments with no modifications as long as the Environments contain the same named Touchpoints. Therefore, it is important to use functional names for Touchpoints and avoid using server names or references to release cycles in the name.

Agents are deployed locally on physical servers on which automation operations will be performed. Agents receive operation instructions from the Orchestrator(s), execute them locally, and return results to the Orchestrator. Each agent also maintains the state of its queued and running operations for fault recovery. An Agent can be part of many Environments and can be mapped to multiple Touchpoints within a single Environment.

An Agent Touchpoint runs modules that affect operations on the immediate host. An Agent Touchpoint can be configured as a Proxy Agent Touchpoint to enable management of operations on a remote host. Proxy Agent Touchpoints manage operations on Microsoft Windows or UNIX machines that require a zero IT PAM footprint.

Load balancing and high availability can be achieved by organizing multiple Agents installed on system capable of performing like operations and assigning priorities. For example:

- An available Agent with the highest priority within a Touchpoint will be the primary for an operation within a process assigned to that Touchpoint
- When multiple Agents within a Touchpoint with the same priority are available the workload for operations referencing that Touchpoint will be distributed among those Agents (i.e., load balancing)

- When no Agents within a Touchpoint with the highest priority are available, Agents with the next highest priority will be tasked with operations referencing that Touchpoint (i.e., automatic failover for high availability)

See the "CA IT Process Manager Administration Guide" shipped with the installation for complete details on Agent/Touchpoint configuration options.

How Many Agents and Touchpoints?

An IT PAM Agent is typically added to run IT PAM Agent modules on remote servers in an environment. You can install more than one instance of IT PAM Agent on any particular computer.

CA IT PAM Agent software is installed on those machines where automated tasks will be performed. However, not every machine that is part of the automated Processes will need Agent functionality. When determining whether Agent functionality is needed, think about the tasks in a Process that occur on a specific machine. Do those tasks normally require a user to be physically at the machine or connected via some remote connection mechanism? If so, you will more than likely need to install Agent functionality on that machine. If not, you may not need that functionality.

For example, let's say that we have two servers: Server A and Server B. Server A is an FTP server. As part of the Process you are trying to automate, you need to log on to Server B, download a file from Server A via FTP, and then process that file on Server B. In this case, you only need Agent functionality on Server B. Although you will connect from Server B to Server A via FTP, that action is initiated from Server B.

You can associate more than one Agent to a single Touchpoint. There are two reasons to do this:

- Provide redundancy and failover among Agents on different hosts
- Provide load balancing among Agents on different hosts

When there are multiple Agents associated with a Touchpoint, IT PAM switches operations to a different Agent when an active Agent becomes unreachable. Agents associated with a Touchpoint can be assigned different priority numbers, so Touchpoint operations run preferentially on the highest priority Agent that is currently available.

To provide load balancing among equivalent hosts, you can assign the same priority to multiple Agents assigned to a Touchpoint. For example, a Touchpoint may be mapped to 20 Agents on 20 different host computers. If you assign the same priority to all of the Agents, Touchpoint operations then run randomly on any available Agent.

At least one Touchpoint in an Environment needs to map to an Orchestrator. Note that Orchestrators also contain Agent functionality, so you normally do not need to install Agent software on an Orchestrator. In addition, for scalability reasons, work should typically not be assigned to an agent located on the Orchestrator node.

Touchpoints can also be organized into Touchpoint groups, if these are grouped in a logical way it makes it much easier to keep track and manage your Touchpoints. In addition, you can also schedule IT PAM processes to run on complete Touchpoint groups, this will ensure that the process is executed in parallel on all Touchpoints in the group.

An **IT PAM Proxy Touchpoint** connects to an IT PAM Agent running on supported host and manages IT PAM operations on remote UNIX or Windows machine. The hosting agent can support the following target machines:

- Microsoft Windows versions listed for IT PAM Agent
- UNIX variants listed for IT PAM Agent
- SSH version 2 protocol server is required on a targeted computer.

In addition, the Korn shell must be installed on any UNIX target for the IT PAM SSH Proxy. If it is not, you must either install it or link it to the Bash shell.

IT PAM Client \ User Interface

CA IT PAM has two levels of user interface. The first level, called the Management Console", is the basic webpage that you use to login. The Management Console only requires a supported web browser and allows you to perform a number of basic functions. To perform some of the more advanced functions, such as creating new or modifying existing IT PAM Process definitions, you need to open the IT PAM Client which is a web-distributed, java-based thin client. In addition to a supported browser, the IT PAM Client requires that JRE 1.5 or better is installed on the client.

The IT PAM Client provides the primary administrative interface for IT PAM and provides all necessary functions required to configure, administer and define automated production, including:

- Administration of the IT PAM Domain
- Server/Agent installation and configuration
- Environment creation and configuration
- Touchpoint mapping and configuration
- Proxy Touchpoint mapping and configuration
- Browsing IT PAM Libraries
- Creating and administering automation-objects in IT PAM Libraries
- Managing security for IT PAM components and database objects.
- Administering Processes and operations

Clustering

Using a front-end Load Balancer, such as an Apache HTTP Server can provide High Availability and enable you to load balance user sessions across multiple CA IT PAM Orchestrator nodes in a cluster. The Load Balancer communicates with the Tomcat Server configured as part of each Server (i.e., the "Worker Node" referenced in the clustering section of the CA IT PAM Server install). The "Primary" server will function as the sole provider of Agent Engine and Policy Engine within the cluster and provide a queue server that acts as a dispatcher of operation requests and updates. It will then load balance these messages across the queue clients sitting on other servers in the cluster. JBoss internally initiates heartbeat messages within the cluster to determine the state of member servers. If the Primary Server then goes down, after a timeout period JBoss will mark it as "down" and redeploy its "services" onto one of the other Servers in the cluster.

Additional information about establishing High Availability\Failover in CA IT PAM is provided in a separate IT PAM Best Practices paper.

Chapter 3: Template Architectures

What is a Template Architecture?

Template architectures are essentially blue prints based on documented best practices for the deployment of key interrelated components that can be used as is or easily adapted to meet the specific needs of your business. For the purposes of this paper, a modular approach was taken to establish base models that function as building blocks for larger models. As your business grows in size or complexity your IT PAM can be adjusted to keep pace though the snapping additional building blocks to the original model.

How to Decide Which Fits Best?

Unfortunately no simple mathematical formula exists that accepts a handful of readily available inputs and returns results that point to the exact design that will work best for you. However, by following a basic process to narrow down the options based on some pertinent facts about your current business practices, plans for automation and integration you should be able to identify optimum choices.

Key Entities

Before you begin the process it is important you understand the role and relationships of the following key IT PAM entities:

- Domain
- Environment
- Touchpoint

The Domain is the highest level entity in the IT PAM hierarchy. Each Domain is completely autonomous and essentially represents a separate implementation IT PAM. Domains interact independently with the security provider (LDAP or CA Embedded Entitlements Manager (EEM)) and manage object level security within their scope via the Domain Orchestrator. Objects can be exchanged between Domains by exporting and importing definitions.

An Environment is a logical entity within a Domain used to segregate objects and operations by lifecycle stage (i.e., test, production), business unit or location (logical or physical). Security is managed by the parent Domain. To be completely autonomous, each Environment should have its own Orchestrator and Library. As with the Domain, objects can be exchanged between Environments by exporting and importing definitions.

Touchpoints are logical representations of an Agent or multiple Agents. Load balancing and high availability can be achieved by deploying Agents on multiple systems then organizing those Agents into a Touchpoint (see Agents and Touchpoints in Chapter 2: Components and Design Concepts).

Key Considerations

In designing your architecture, business requirements should take priority over capacity or performance. For the initial design isolation of operations and data (autonomy) and the need for high availability should be taken into consideration first. In general, capacity or performance can be increased by adding instances of Orchestrators and Agents to a well defined Domain architecture that is aligned with the needs of the business.

Autonomy

The first step in the process is to identify the number of Domains (see discussion of Domains under CA IT PAM Components earlier in this document) you will be implementing and the purpose of each of them. Begin by asking how IT PAM will be managed. For example:

- A single Domain with a single Environment. This is not a typical configuration but could be suitable for small deployments targeted at single solution usage.
- A single centralized Domain with multiple autonomous Environments aligned with lifecycle stages, business units or geographic regions to orchestrate all procedures in the enterprise
- Multiple Domains with multiple Environments

The critical factors in making this decision are related to your needs for management autonomy and data isolation. While it is true that a single domain can be broken down into logical Environments (see Environments discussion under CA IT PAM Components earlier in this document).

High Availability

CA IT PAM is an enterprise-class product that can be implemented as highly available. However, as with any application, additional hardware and software is required. In deciding whether to implement high availability for CA IT PAM, you must weigh the additional costs against the cost of a potential outage (for maintenance or due to a disaster). If the cost of any such potential outage would exceed the cost of the required hardware and software you should select a highly available template architecture as the basis for your model.

Communications

Slow or high latency network links present a challenge for designing an architecture for any distributed application. In the case of CA IT PAM, keep the following in mind:

- Communication between the Domain Orchestrator, its Library (database) and other components is limited to requests for configuration and security access information and responses to those requests and is, in general, less demanding on network resources and more tolerant of latency.
- Communications between an Orchestrator, its Library (database) and Agents is more demanding and network speed/latency will have a measurable impact on performance.

In general, Orchestrators should be located as close (electronically) as possible to their Libraries (databases) and to the Agents they interact with. If you need to choose between locating the Orchestrator close to the Libraries or close to the Agents, it is preferable to locate them closer to the Libraries for better performance.

Scalability

Assuming you have identified the number of Environments and Orchestrators in your Domain, you should now turn your attention selecting the template architecture which will be used as the starting point for your implementation. To do this, you will need to estimate the following important metrics:

- Number of Service Level Agreements (SLAs) related to the procedures to be automated
- Average and peak number of procedures that must execute per unit time
- Average number of operations per procedure (sometimes referred to as "transitions") and the degree of parallelism of (correlation between) operations
- Average duration of operations (not as critical for initial planning)

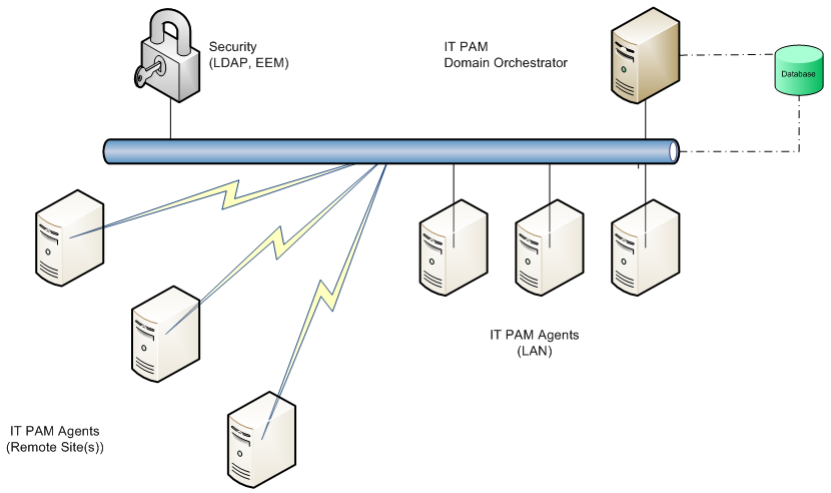
These metrics can then be translated into values that can be compared to capacity ranges for the small, medium and large template architectures. See "Appendix C: Performance Guidelines" for guidelines on using these metrics. Understandably, you might find gathering the information needed to estimate the anticipated load a tedious and difficult task. The better the estimates are the better the chances of "right sizing" your implement initially, however, should you underestimate adding additional component instances to increase capacity is possible later.

Template Architectures for Domains

The Domain is the top level CA IT PAM entity (see Domains section in Chapter 2). Every deployment will require at least one Domain instance. In more complex deployments where complete autonomy for distinct lifecycle stage, business unit or geographical region is required multiple Domains may be implemented. When such extreme isolation is required each Domain should be approached as a separate architecture since no dependencies on common components need be considered.

Standalone Domain

As the most basic of all template architectures, the "Standalone Domain" architecture is ideal for a non-production ("lab") environment or as a low-budget production deployment option when high-availability is not a requirement. Here is an example of a Standalone Domain architecture:



In a "Standalone Domain" architecture, a single "Domain Orchestrator" is deployed. The associated CA IT PAM "Library" database and "Reporting" database may be hosted by a DBMS instance on the same system or a remote system. Agents only are deployed on other systems where operations will need to be executed.

Security

Security is provided by either an LDAP compliance directory or an instance of CA Embedded Entitlements Manager (CA EEM) which may be installed locally or remotely.

Database Specifications

Consult the DBMS vendor specifications for hardware and operating system requirements. Typically 2 CPUs, a minimum of 4 GB memory and a minimum 40 GB or free space would be recommended.

Domain Orchestrator System Specifications

A CA IT PAM Orchestrator is configured as the primary Domain Orchestrator. See Appendix A for hardware recommendations and Appendix B for prerequisite software requirements.

Autonomy

Autonomy can be achieved in any sized IT PAM Domain by defining multiple Environments (see Environments under CA IT PAM Components) and Libraries then defining appropriate users, roles and access rules (see "CA IT Process Automation Manager Best Practices: Securability Guidelines"). However, additional hardware will be required to host the additional instances of IT PAM Orchestrators and Libraries (databases) required.

High Availability

The "Standalone Domain" architecture template is a low cost option. Redundancy required for a highly available solution is eliminated to reduce hardware, software, implementation and maintenance costs. Therefore, "Standalone Domain" architecture should not be considered whenever business requirements dictate that CA IT PAM be highly available.

Scalability

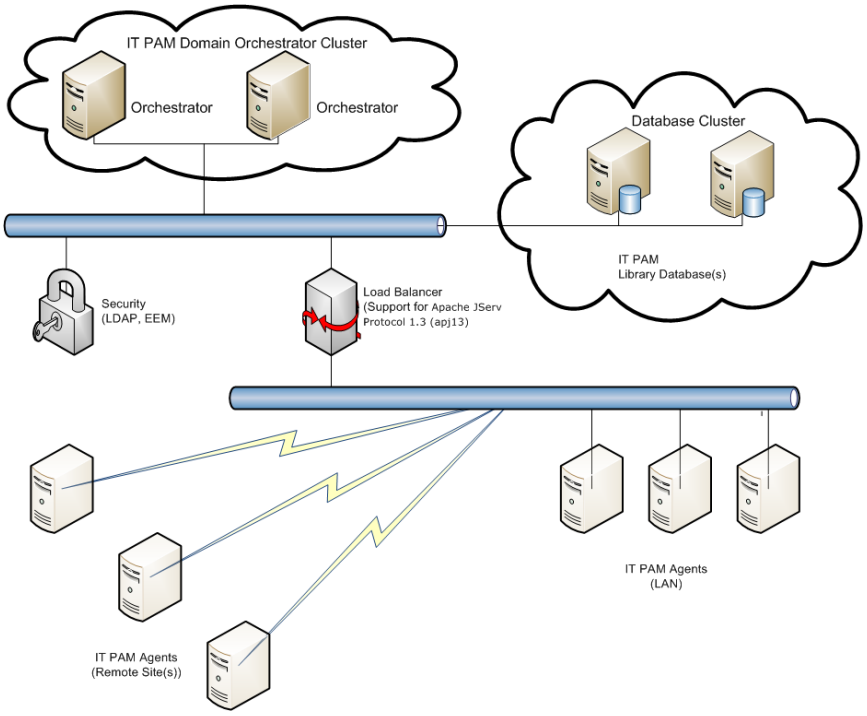
Even with the limited hardware resources used in the "Standalone Domain" template architecture, Orchestrator performance typically exceeds requirements with normal usage. Because the CA IT PAM design allows for flexibility and expansion, adding additional standalone or clustered Orchestrators is always possible to increase capacity.

See also Appendix D for tuning recommendations to optimize performance.

Highly Available Domain

If, after analyzing business requirements (see How to Decide Which Fits Best?), you determine that an outage, planned or unplanned, would severely impact business you should use the "Highly Available Domain" template architecture as the starting point for planning your deployment.

Following is an example of a highly available CA IT PAM architecture:



In the "Highly Available Domain" architecture, the Library is stored in a clustered instance of the DBMS. A primary and secondary node for the Domain Orchestrator are implemented on separate systems and accessed via a load balancer. Should one of the DBMS or Domain Orchestrator host systems be taken off line or fail, the outage would have little or no impact.

ITPAM Domain Orchestrator System Specifications

The Domain Orchestrator is configured as a cluster via JBoss Application Server (see Guidelines for Continuous and High Availability for details). See Appendix A for hardware recommendations and Appendix B for software requirements.

Security

Security is provided by either an LDAP compliance directory or an instance of CA Embedded Entitlements Manager (EEM) which would be installed locally or remotely.

Database Cluster Specifications

All supported DBMS can be implemented as highly available clusters. Hardware and software requirements will vary dependent upon the DBMS you choose to implement. Consult the documentation published by the DBMS vendor for details.

Load Balancer Specifications

This architecture requires a load balancer (hardware or software) that supports the Apache JServ Protocol 1.3 (ajp13) (see CA IT Process Automation Manager Best Practices for Continuous and High Availability Guide).

High Availability

The "Highly Available Domain" template architecture requires a greater investment in resources and is slightly more complex than a simple "Standalone" model. The benefit, however, is that critical components are redundant, thereby eliminating single points of failure and potential outages. Maintenance can be applied without the need to interrupt critical business services.

Autonomy

Autonomy can be achieved in any sized IT PAM Domain by defining multiple Environments (see Environments under CA IT PAM Components) and Libraries then defining appropriate users, roles and access rules (see "CA IT Process Automation Manager Best Practices: Securability Guidelines"). Unlike the "Standalone Domain" model, the systems hosting the Domain Orchestrators are not serving double-duty as DBMS hosts so additional resources are available to support additional Orchestrator instances. Any additional Library databases that are required can be hosted in the DBMS. If the individual Environment must be highly available the associated Orchestrators should also be clustered (see "Highly Available Autonomous Environment" for details).

Scalability

Since the Domain Orchestrator is comprised of "active/active" cluster nodes you can expect better performance compared to a "Standalone" implementation. Resource contention is also reduced since the systems hosting those components are not also tasked with providing resources for a DBMS.

See also Appendix D for tuning recommendations to optimize performance.

Template Architectures for Autonomous Environments

In the CA IT PAM entity hierarchy, Environments are next level down from Domains (see Environments in Chapter 2: Components and Design Concepts). The "Default Environment" will be created automatically when a Domain is deployed that will include the Domain Orchestrator. Although the primary mission of the Domain Orchestrator is to interact with the default (Domain) security provider (LDAP or EEM) to manage the default configuration for various modules, it may also be used to manage process execution in non-production or a small deployment. Such an approach limits capabilities in terms of performance but, more importantly autonomy. Consider the following:

- It is highly recommended that procedures at different lifecycle stages (DEV, Test, Production) be isolated. The recommended isolation can be achieved by implementing multiple Domains or autonomous Environments within a single Domain - each with its own Orchestrator and associated Library.
- Data and processes for different sub-divisions of the enterprise may be required to be kept separate – either by law or as a requirement of internal business policies. Depending on the degree of separation mandated, it may suffice to deploy autonomous Environments within a single Domain.

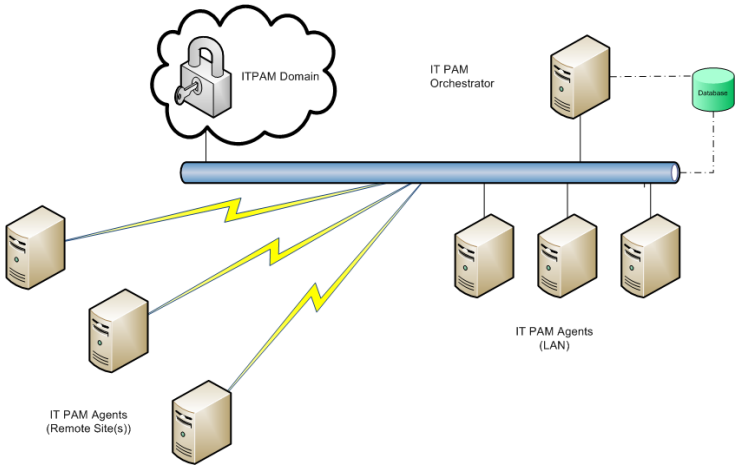
Compared to deployment of multiple Domains, implementing autonomous Environments within a single Domain provides the following benefits:

- Reduced implementation costs: some systems and resources can support multiple instances of components associated with different environments (for example, a powerful DBMS host or cluster can support multiple IT PAM Libraries (databases).
- Simplified administration: Security can be defined and administered across a single Domain eliminating redundant user and role definitions

Given the above, production deployments that rely solely upon the "Default Environment" would be atypical.

Standalone Autonomous Environment

The "Standalone Autonomous Environment" provides isolation by supporting and interacting with an autonomous instance of an IT PAM Library through an independent instance of Orchestrator.



In the "Standalone Autonomous Environment" a single Orchestrator is deployed. It is associated with an IT PAM Library (database) instance hosted by a DBMS instance running on the same system or a remote system. Agents only are deployed on other systems where operations will need to be executed.

Security

The default is inherited from the IT PAM Domain or it may be overridden at the Environment or Orchestrator level.

Database Specifications

Consult the DBMS vendor specifications for hardware and operating system requirements. Typically 2 CPUs, a minimum of 4 GB memory and a minimum 40 GB or free space would be recommended.

Orchestrator System Specifications

The Orchestrator is configured as member of a previous installation. See Appendix A for hardware recommendations and Appendix B for software requirements.

Autonomy

Since each autonomous Environment will have its own instance of Orchestrator and Library, modifications to one environment will have no impact on operations of another environment. In the case where distinct environments are implemented for each lifecycle stage, objects developed and tested in one environment can be "promoted" to the next stage using the simple "Export" and "Import" functions available. Likewise, common objects can be distributed to any number of environments using the same technique.

High Availability

Like the "Standalone Domain", the "Standalone Autonomous Environment" is a low cost option. Redundancy required for a highly available solution is eliminated to reduce hardware, software, implementation and maintenance costs. The "Standalone Autonomous Environment" should not be considered in situations where high availability is required.

Scalability

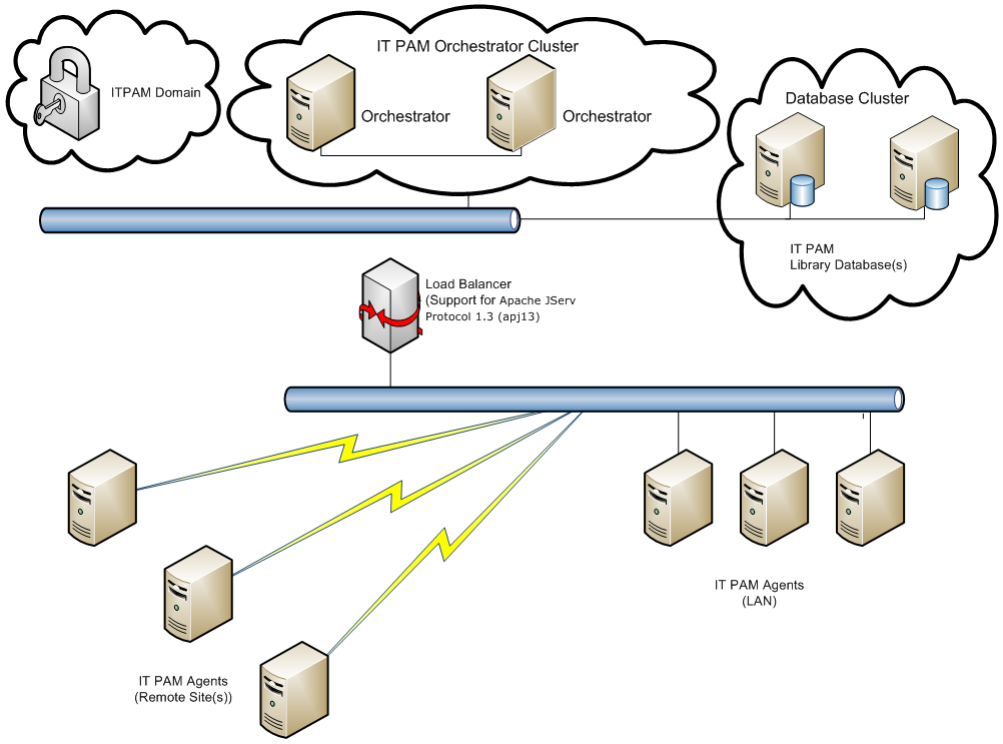
Orchestrator performance, even with the limited hardware resources used in the "Standalone Domain" template architecture, typically exceeds requirements with normal usage. Because the IT PAM design allows for flexibility and expansion, adding additional standalone or clustered Orchestrators is always possible to increase capacity.

See also Appendix D for tuning recommendations to optimize performance.

Highly Available Autonomous Environment

As with the "Highly Available Domain", when an outage, either planned or unplanned, would severely impact business use the "Highly Available Autonomous Environment" template architecture.

Following is an example of a Highly Available Autonomous Environment:



The "Highly Available Autonomous Environment" is very similar to the "Highly Available Domain" in design. But while the primary mission of the Domain Orchestrators is to manage security the primary mission of the Orchestrators in an autonomous Environment is to manage the Library and operations within its scope.

The Library is stored in a clustered instance of the DBMS. Depending on the resources of the DBMS cluster systems and network topology this may be a separate cluster or it may be shared with the Domain and/or other autonomous Environments by implementing multiple DBMS instances in the cluster or by simply assigning unique database names when installing IT PAM Orchestrators for different Environments.

Primary and secondary Orchestrators are implemented on separate systems and accessed via a load balancer. Should one of the DBMS or Domain Orchestrator host system be taken off line or fail, the outage would be have little or no impact.

Security

The default is inherited from the IT PAM Domain or it may be overridden at the Environment or Orchestrator level.

Database Cluster Specifications

All supported DBMS can be implemented as highly available clusters. Hardware and software requirements will vary dependent upon the DBMS you choose to implement. Consult the documentation published by the DBMS vendor for details.

Load Balancer Specifications

This architecture requires a load balancer (hardware or software) that supports the Apache JServ Protocol 1.3 (ajp13) (see CA IT Process Automation Manager: Quick Start Guide for Clustered Domains).

ITPAM Orchestrator System Specifications

In this configuration the Orchestrator is configured as member of a cluster (see Guidelines for Continuous and High Availability for details). See Appendix A for hardware recommendations and Appendix B for software requirements.

High Availability

The "Highly Available Autonomous Environment" template architecture requires a greater

investment in resources and is slightly more complex than a simple "Standalone" model. The benefit, however, is that critical components are redundant eliminating single points of failure and potential outages. Maintenance can be applied without the need to interrupt critical business services.

Autonomy

Since each autonomous Environment will have its own Library and Orchestrator instances, modifications to one Environment will have no impact on operations of another Environment. In the case where distinct Environments are implemented for each lifecycle stage, objects developed and tested in one Environment can be "promoted" to the next stage using the simple "Export" and "Import" functions available. Likewise, common objects can be distributed to any number of Environments using the same technique.

Scalability

Orchestrators are essentially "active/active" cluster nodes so you can expect better performance compared to a "Standalone" implementation. Testing indicates transitions/minute (measure of the ability to detect completion of one operation or step in a process and start the next) improve approximately 35% for each Orchestrator instance deployed in a cluster. Resource contention is also reduced since the systems hosting those components are not also tasked with providing resources for a DBMS.

See also Appendix D for tuning recommendations to optimize performance.

Extending Template Architectures

Over time, business requirements may change. For example:

- New business units may be added or acquired, requiring additional autonomous Environments
- Demands for automation may increase requiring additional resources to increase capacity

Because CA IT PAM is based on a modular design these situations can be easily handled by simply adding Domains or new Environments to an existing Domain or additional Orchestrator instances (clustered or non-clustered) to an existing Environment.

Note: If there is a possibility that the Orchestrator will be clustered in the future the recommendation is to install it in a single node cluster initially. It is much easier to add additional node as needed than to add an existing Orchestrator to a new cluster.

Adding Domains or Environments

The planning, design and implementation processes for adding Domain or Environments are no different than those presented in the preceding sections for new implementations. The key considerations and associated template architectures are exactly the same. Once the hardware and prerequisite software are available a new Environment or even a new Domain can quickly and easily be implemented.

Adding Orchestrators to Increase Capacity or Availability

It is often difficult to accurately project the demand for services and, therefore, the capacity required. When you find that the load in a particular Environment may be impacting performance and response times it is always possible to add additional Orchestrators by creating or extending an existing Orchestrator cluster.

See also Appendix D for tuning recommendations to optimize performance.

Adding Agents to Touchpoints to Increase Capacity or Availability

Since management activities (i.e., initiating processes, dispatching operations, and maintaining status) typically consume far fewer resources that the actual execution of the operations it is more likely that additional Agents will be deployed. Agent capacity is impacted primarily by the resources and time to execute operations assigned. While most operations complete very quickly certain operations (ex. script execution, query execution) can be time consuming and/or resource demanding. Additional Agent instances can be easily added to any existing Touchpoint. Depending on the priorities set for the member Agents within a Touchpoint, the additional instances can be used to better distribute load between members or as stand-in instances to when high availability is a concern (see Agents and Touchpoints in Chapter 2: Components and Design Concepts).

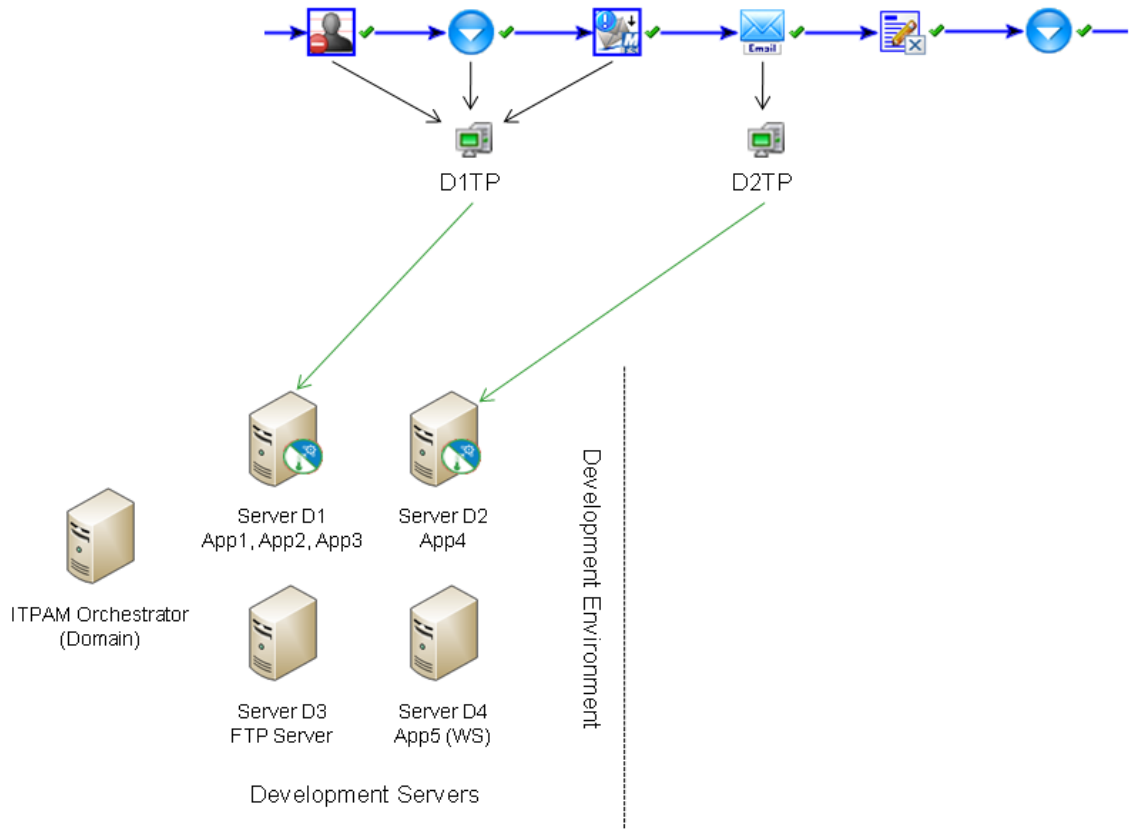
See also Appendix D for tuning recommendations to optimize performance.

Chapter 4: Examples

This chapter contains examples of various CA IT PAM deployment configurations.

Using Touchpoints

Here is an example of a simple CA IT PAM architecture, consisting of a single Development Environment:



In this example, there are four development servers that will be part of an automated process: D1, D2, D3 and D4. Two of these servers (D1 and D2) host an application that needs to be accessed locally via command execution. Server D3 is an FTP server and hosts files that need to be downloaded to the Orchestrator machine while Server D4 hosts an application that can be accessed via Web Services. To configure this environment, we do the following:

1. Identify which machine will serve as the Orchestrator and install the CA IT PAM software on it.

   This machine will automatically become the Domain Orchestrator.

2. Identify which machines require the CA IT PAM agent.

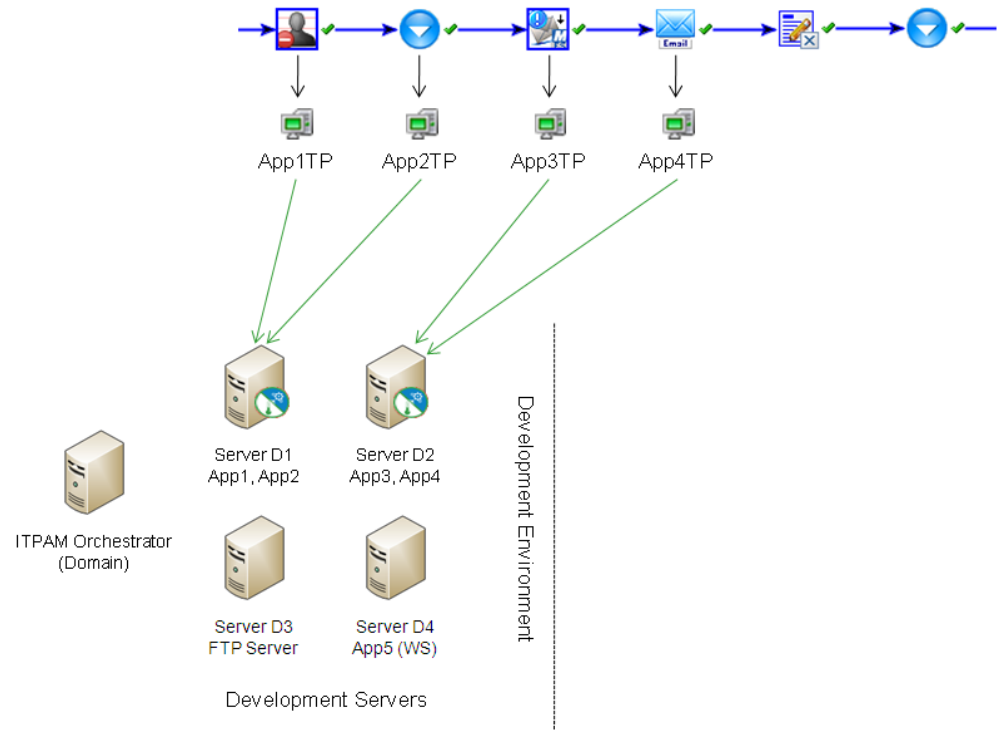
   Since we do not need to perform tasks locally on servers D3 and D4, we do not need to install agents on these machines. We only need to install agents on servers D1 and D2.

3. Optionally, create and IT PAM Environment named "Development." and install a autonomous Orchestrator to manage this environment.
4. Create Touchpoints to map the agents on Server D1 and Server D2 to the Environment. You might be tempted to call these D1TP and D2TP, however due to reasons outlined below this is NOT recommended.

Now let's say that we create a Process with six steps. There are 5 applications and 1 FTP server. The six steps touch each component in that order, so it's App1, App2, App3, App4, App5, then FTP. For each step – or Operator in the Process – we have to identify which Touchpoint to run on. For the first three steps, which involve App1, App2 and App3, we go to Touchpoint D1TP. For step 4, we go to Touchpoint D2TP. The remaining steps are performed from the Orchestrator since we do not have to do anything local to those machines.

This configuration may work fine but, what if there is a change in the IT environment. For example, what happens if you move App3 to Server D2 because the load on Server D1 is too much? In this case the third Operator in the Process, which was supposed to be performed on App3, is actually running on the wrong machine. To resolve this we could go to that Operator and change the target Touchpoint but, what if there were another 50 Processes which include an Operator that is supposed to be performed on App3? Does this mean that we have to search through all Processes to identify where we should be working with App3 but are using the wrong Touchpoint?

One way to better handle changes in the IT environment is to re-think how Touchpoints are defined. Remember that a Touchpoint is "an abstraction of a managed resource." In our first example, Touchpoints were defined based on the machine in use. Now, consider defining Touchpoints based on their function rather than the machine they are performed on. To do this, let's take another look at our example. This time, instead of creating Touchpoints that represent Servers D1 and D2, we're going to create Touchpoints that represent App1, App2, App3 and App4.



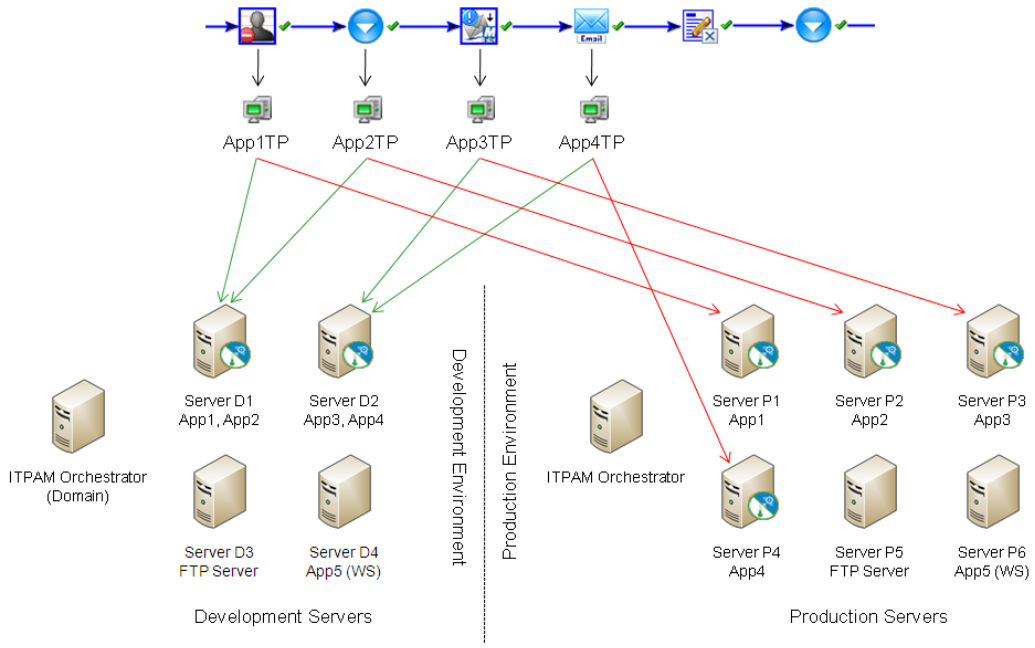
Originally, three of our Touchpoints mapped to the Agent on Server D1, and the other Touchpoint map to the Agent on Server D2. Operators in the Process will be configured to run on Touchpoints that represent the applications, not the servers. This time, when the third step in the process is moved to accommodate heavier load on the initial server, instead of having to reconfigure our Process, and possibly dozens or hundreds of other Processes that have an Operator that needs to perform on App3, we just change the mapping of our Touchpoint.

Next, let us consider how Touchpoints can be used in a production environment. It is not uncommon for there to be differences between development and production servers, however, our procedures will be similar. The first step is to create a new Environment for the production servers and, for the sake of simplicity, call it "Production Environment." Since each Orchestrator can only belong to a single CA IT PAM Environment, we need to identify a new server in the Production Environment on which to install the Orchestrator software. This server will point back to the existing Domain Orchestrator. Again, Agents will be installed locally, pointing web browsers to the Domain Orchestrator.

The naming convention for Touchpoints used in the In the Development Environment reflected the function\application that was used. Since this will be the same in the Production Environment, we'll use the same names. In fact, we could copy the Process from the Development Environment to the Production Environment and it would work immediately – with no changes – even though the names used by the actual development servers and production servers are different.

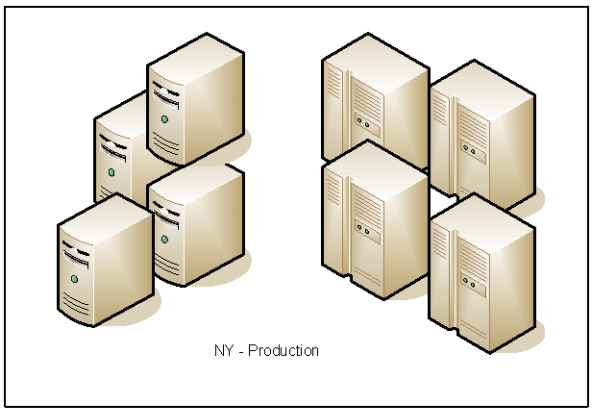
The first Operator in our Process executes on Server D1 in the Development Environment. In the Production Environment, this would be Server P1. Similarly, the fourth Operator, which executed on Server D2 in the Development Environment, operates on Server P4 in the Production Environment.

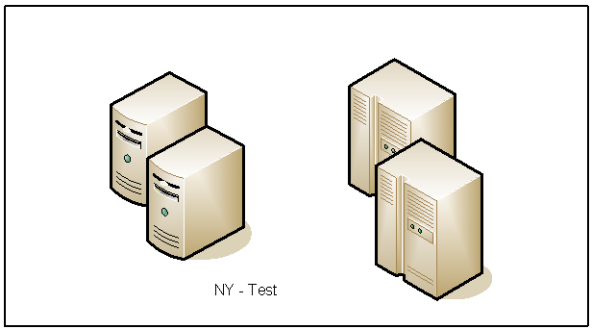
Here you can see an example of this environment:



Scalable Deployments Example

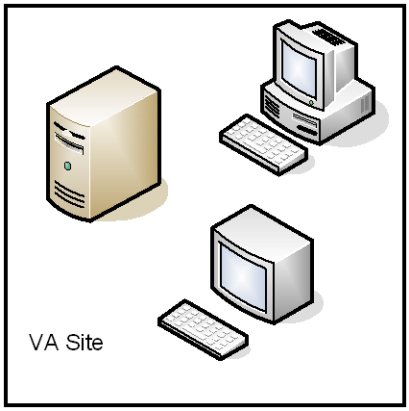
In the following example there are 3 sites to which CA IT PAM will be deployed. The first site is located in New York. This site is the primary location for the organization's Systems Management and mission critical systems and applications.



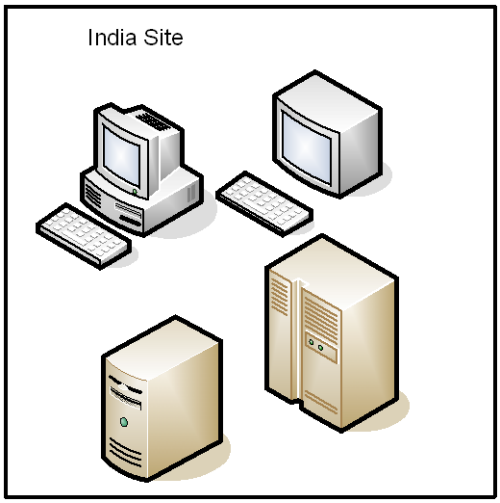


It consists of a production environment that contains over 400 servers that support live operations and a smaller test environment that contains 50 servers which are used to stage deployment of new applications and servers.

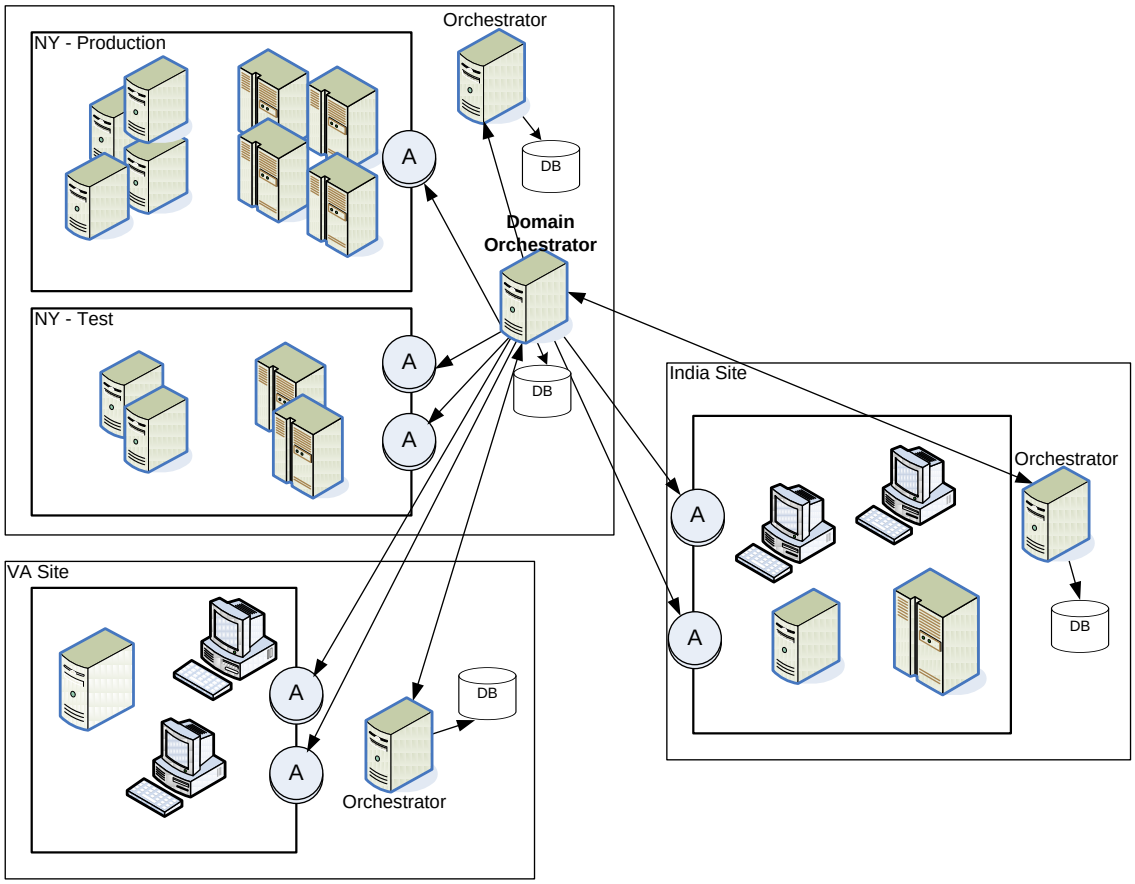
The second site is located in Virginia. It contains a dozen servers which support local financial business systems, along with several workstations and servers which are used to conduct routine maintenance and administrative tasks.



The third site is located in India. It contains over 200 server and workstations that are used by the local development teams. Although none of these servers is mission critical there are several administrative and maintenance processes that are performed against them.

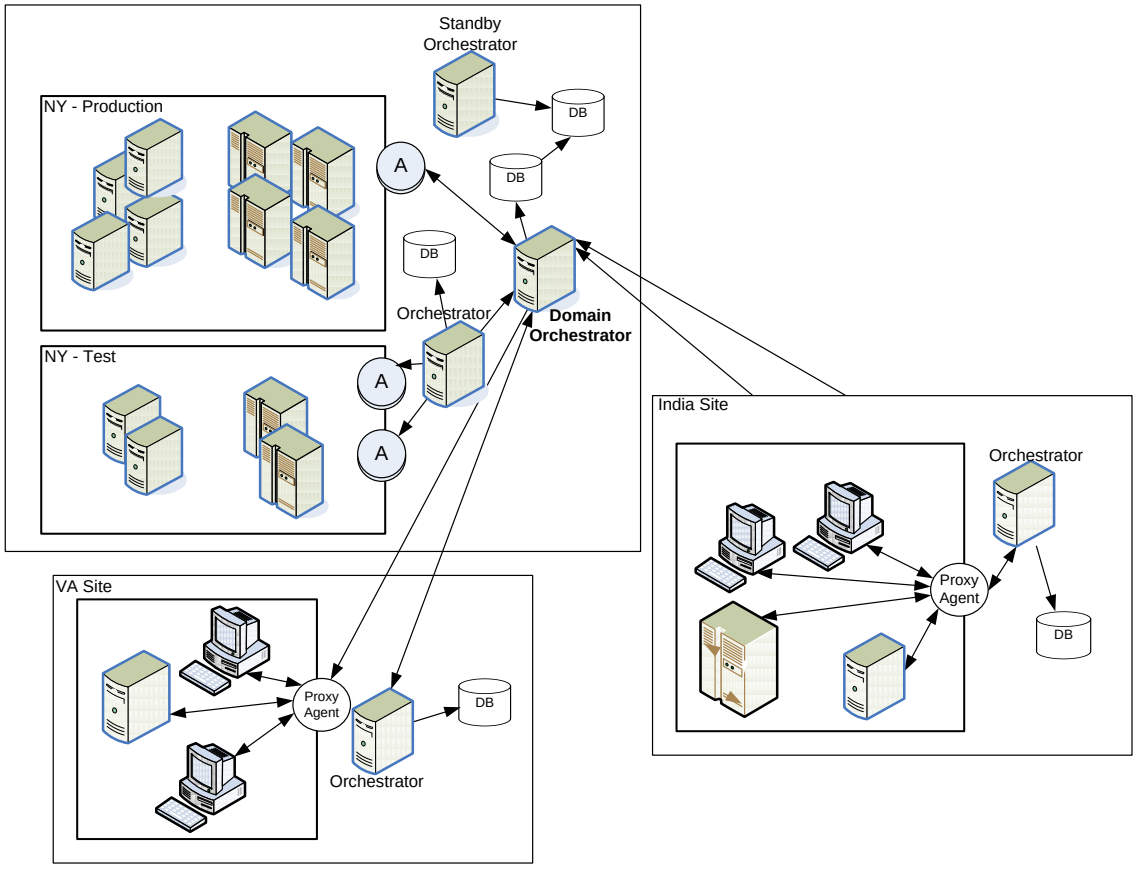


We start by installing the Domain Orchestrator at the NY site and deploy additional Orchestrators to support the remaining environments in Virginia and India. Then, we install Agents on several systems and servers in all three locations. Upon installation the Orchestrators and Agents identify themselves to the Domain Orchestrator.. :



CA IT PAM Environments are created to support the individual environments – Test and Production – that are used at each site. When the Orchestrators are added to the Environments and Agents are associated with Touchpoints in those Environments, they join a new communication group which is used to exchange process operation information.

Although this deployment is functional, it lacks fault tolerance and load balancing in the mission critical environments. Furthermore, there are more Agents than necessary given the type of systems that are in use. As a result, the unnecessary agents can cause extraneous network traffic. To address this shortfall we can deploy clustered Orchestrators for High Availability (HA) and load balancing. The database for the clustered (i.e., secondary Orchestrator) can be clustered or replicated from the live (i.e., Primary Orchestrator) Domain Orchestrator's database. Furthermore, Proxy Agents can be used for non-critical locations, such as in the India site, in order to minimize the CA IT PAM footprint and reduce the network demand.


Standby
Orchestrator
DB
NY - Production
A
DB
DB
Orchestrator
Domain
Orchestrator
NY - Test
A
A
India Site
Orchestrator
Proxy
Agent
DB
VA Site
Proxy
Agent
DB
Orchestrator

Appendix A: Hardware Recommendations

The recommendations listed in the following sections were current at the time this document was posted. Recommendations may change with new releases and/or certifications. To view the most accurate up-to-date information consult the "CA IT Process Automation Manager Installation and Configuration Guide" distributed with the installation media.

Orchestrator

The following recommendations assume the Library is hosted by a DBMS running on a separate, remote system. When the DBMS will be co-hosted (running on the same system as the Domain Orchestrator) the resource requirements for the DBMS should be added to the resource requirements recommended below:

- Minimum 2 CPU minimum
- Minimum 4 GB memory
- Minimum 40 GB minimum free space
- Minimum 100 Mbps network connection (1000 Mbps recommended)

Appendix B: Prerequisite Software

The prerequisites listed in the following sections were current at the time this document was posted. The list of the supported platforms might change through later patches and/or certification. To view the most accurate up-to-date information consult the Compatibility Matrix that can be accessed from the CA IT PAM page on Support Online (See https://support.ca.com/irj/portal/anonymous/phpdocs?filePath=0/8237/8237_compmatrix.html)

Orchestrator

Prerequisite software not provided with the installation media:

- Supported operating system
  - Microsoft Windows 2003, 2008 Server
  - Linux RedHat 3.0/4.0/5.0, SuSE 9/9.1/9.2/10/11
  - UNIX: HPUX 11i, Solaris 9/10 or AIX 5.3+
- Sun JDK r1.6+
- Access to a supported DBMS
  - MySQL r4.2+
  - MS SQL 2005, MS SQL 2008

Third party software provided with the installation media:

- Appropriate JDBC driver for selected DBMS
- JBoss Application Server (Provided with CA IT PAM installation media)
- Hibernate
- TAPI Driver (optional)

Agent

Prerequisite software not provided with the installation media:

- Supported operating system
  - Microsoft Windows XP, Windows Vista, Windows 7, 2003 Server, 2008 Server
  - Linux RedHat 3.0/4.0/5.0, SuSE 9/9.1/9.2/10/11
  - UNIX: HPUX 11i, Solaris 9/10 or AIX 5.3+
- Sun JRE 1.5+

Client

Prerequisite software no provided with the installation media:

- Supported browser
  - IE Explorer 6.0+
  - Mozilla FireFox 2.x+
- Sun JRE 1.5+

Appendix C: Performance Guidelines

Following are some preliminary guidelines that can be used to estimate general scalability based on expected performance. For example, let's assume the following:

- On average, you expect to trigger roughly 50 automated procedures per hour during regular processing and 100 per hour during peak periods
- Each procedure executes 10 operations (steps) to complete the procedure (i.e., opens a change order, sends emails, etc.)
- The expectation is that the implementation be capable of "queuing" the procedure, "transitioning" all the operations in the average procedure and "completing" the procedure under within 15 minutes of being triggered.

Based on the above information we can conclude the following about the capacity required:

- On average, the implementation must be able to "queue" 8.3 procedures/minute, 16.6 procedures/minute during peak periods
- On average, the implementation must be able to "transition" 12.5 operations/minute, 25 operations/minute during peak periods

Obviously, values for the metrics above may not be readily available and estimating them may be difficult if you are unfamiliar with CA IT PAM. Following are some suggestions to get you started:

- If CA IT PAM will be implemented to replace an existing workflow process manager, examine the defined procedures, break them down into component steps (operations) and gather performance information on the execution of those steps if available.
- Install a simple IT PAM Environment (See *CA IT PAM Quick Start Guide*) and mock up some typical procedures that will be needed to understand the number of operations that will be required and the times of operations that may be "long running".
- Engage CA services to assist with research and assessment of capacity required

See also "Appendix D: Tuning for Increased Scalability" for information on what changes can be made to improve performance.

Appendix D: Tuning for Increased Scalability

The right tuning practices can significantly improve both CA IT PAM performance and scalability. Following are several areas that can be considered for tuning:

- Hardware Recommendations
- OS Parameters
- Network Parameters
- Database Parameters
- JVM Parameters
- JBOSS Parameters
- Web Services and UI Parameters
- IT PAM Orchestrators

Keep in mind that proper tuning requires identifying an acceptable balance between different factors affecting the environment. Changing certain parameters may negatively impact other areas. Therefore, it is important that you test any changes you plan to make in a non-production environment prior to deployment.

OS Tuning Parameters

CA IT PAM components are supported on a number of different operating systems. The type and degree of OS tuning parameters will depend on the specific operating system, version and patch level in use. The recommendations provided in this section are based on IT PAM Orchestrators running under the following OS:

- Microsoft Windows Server 2003 SP2 and SP3
- Red Hat Enterprise Linux v 5.0
- Sun Solaris 10 on SPARC

Use of 64 bit OS/Hardware is recommended for many reasons:

- Data path between memory and CPU is doubled, which improves performance of memory-bound applications
- 64-bit memory addressing theoretically gives unlimited heap allocation
- Removes some artificial memory address restrictions that exist in the OS
- Faster data encryption\decryption algorithms within applications may execute 3-5 times faster

Use of 64-bit JVM is highly recommended for applications that run with more than 1.5 Gig of memory.

Tuning Recommendations for Microsoft Windows Server 2003

The following tuning recommendations apply to Microsoft Windows Server 2003. Similar recommendations are valid for most operating system; however, there might be minor variations:

- Disk
  - The system page file, the operating system and the data must be on separate physical disks if possible
  - Ensure Allocation Unit Size is appropriate for the size of the volume
  - Need at least 1.5Gig for the OS and another 3 – 5Gig for the future service packs, and other optional components, etc.
  - Review the recommended registry settings listed in the table below for tuning the Storage and File system.
- Memory
  - Avoid page file on the same drive as the operating system
  - Avoid page file on a fault-tolerant drive, such as mirrored volume
  - Do not place multiple page files on different partitions on the same physical disk drive
  - Set Paging file size to be at least 1.5 times the amount of RAM
  - Setting the paging file's initial size and maximum size to the same value increases efficiency because the operating system does not need to expand the file during processing. Setting different values for initial and maximum size can contribute to paging file overhead and disk fragmentation.
  - Memory dump file needs to be at least as large as the physical memory available plus 1Mb.
- Network
  - Confirm network adapter settings are optimal
  - Bind each network adapter to a CPU
  - Check the following Network Parameters for SMB (File Server Communications): NumTcbTablePartitions, Tcp1323Opts, TcpWindowSize, and MaxHashTableSize. See details on this in the "Registry Setting for Tuning Network on Windows Server 2003" section below.

Registry Setting for Tuning File System and Memory on Windows Server 2003

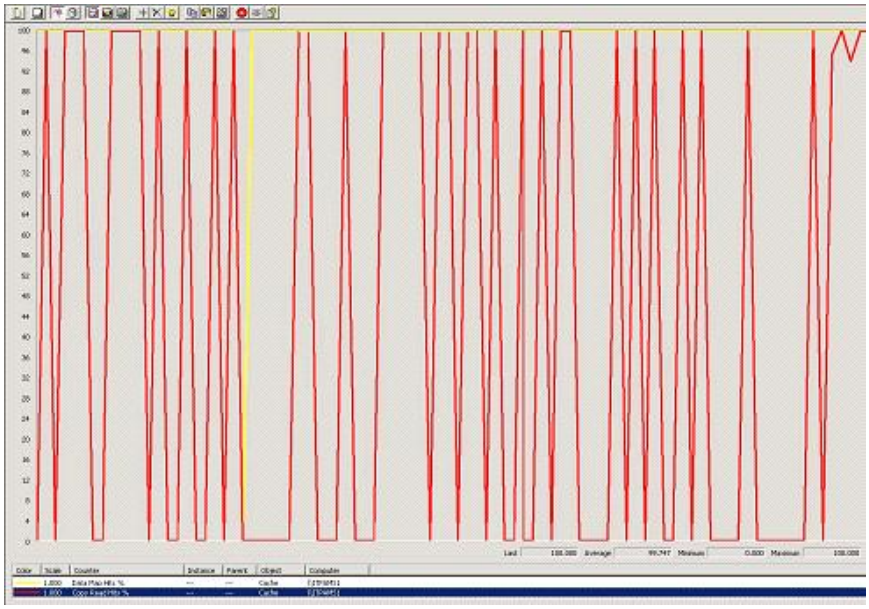
The following registry settings can be used to tune file systems and memory performance:

- **PagedPoolSize**

  Paged pool is an area of system memory (physical memory used by the operating system) for objects that can be written to disk when they are not being used. File cache space and paged pool space share a common area in system virtual address. Limiting the size of the paged pool to 192,000,000 lets the system expand the file system virtual address space up to 960 MB, allowing for a larger file system cache, which causes more content to be cached and consequently faster serving of files. To set aside extended virtual address space for the file system cache, edit this entry (or create a new one if one doesn't already exist) and set the value to 192,000,000. **LargeSystemCache** must be set to 1 and **SystemPages** set to 0.

  Warning: Maximizing the size of the system cache (in other words, setting LargeSystemCache to 1, PagedPoolSize to 192,000,000, and SystemPages to 0) must be avoided on Terminal Services machines (where only 2GB of system memory is shared by all users) or when the /3GB switch is used (as there is only 1GB of system memory).

  The Cache object performance counters Copy Read Hits % and Data Map Hits % provide an overview of the cache's effectiveness – showing files being accessed fitting nicely into the real memory available for caching. In the example below, the average Copy Read Hits % is 99.747 while the average Data Map Hits % is 100.



  System Cache tells how much memory is being used as a disk cache, in other words, avoiding accessing the physical hard drive. Of course, there must be a balance between System Cache and Available Physical Memory since a reasonable amount of free memory is required, but free memory is also wasted memory and can be used as disk cache whenever possible. The Memory object performance counter System Cache Resident Bytes reports the amount of real memory currently in use by the file cache. As the number of System Cache Resident Bytes increases, the various measures of hit ratio are expected to will also increase.

File cache space and paged pool space share a common area in system virtual address. Limiting the paged pool size allows for larger system cache, which causes more content to be cached and allows for faster serving of files.

Location:

<HKLM>\System\CurrentControlSet\Control\MemoryManagement

Registry Setting for Tuning Network on Windows Server 2003

The following settings can be used to tune network performance:

- **NumTcbTablePartitions**

  This setting controls the number of TCB table partitions. The TCB table may be partitioned to improve scalability on multiprocessor systems, by reducing contention on the TCB table. The default value is 0x4. Edit (or create if non-existent) this entry and set it to 8. The value should always be a power of two, that is, 2, 4, 8, 16, 32, etc.

  Note: On multiprocessor systems, change the number of partitions to four times the number of processors in your system.

  Location:

  <HKLM>\System\CurrentControlSet\Tcpip\Parameters\Interface

- **MaxHashTableSize**

  Hash table holding the state of TCP connections. The default value is 128 multiplied by the square of the number of processors in the system. When a large concurrent load is expected then this can be set to a maximum value of 65536. Keep in mind that the table uses non-paged pool. Therefore, do not set too high a value for the parameter if the server does not have much available non-paged pool, which can be monitored with the Pool Nonpaged Bytes counter of the Memory object in System Monitor.

  Location:

  <HKLM>\System\CurrentControlSet\Services\Parameters

- **TCPWindowSize**

  Determines the maximum number of bytes that can be outstanding on the network at any given time. In other words, the sending host can send only that amount of data before it must wait for an acknowledgment and window update from the receiving host. You may need to increase the window size to above 64K as larger receive windows improve performance especially over **high-delay** (high RTT), **high-bandwidth** networks. Otherwise, the sending host will never be able to send enough packets to fill the logical pipe created for the connection before having to wait for an acknowledgement and window size update from the receiving host. For greatest efficiency, the window size should be an even multiple of the TCP Maximum Segment Size (MSS). The default initial window sizes are approximately:

  > Gigabit: 65535 bytes

  > 100 Mbps: 16384

  > All interfaces of lower speeds: 8192

For example, for a 65535 byte receive window you can only achieve an approximate throughput of 5.24 megabits per second (Mbps) on a path with a 100ms RTT, regardless of the transmission path's actual bandwidth. With today's high-BDP transmission paths, the originally designed TCP window size, even at its maximum value, becomes a throughput bottleneck.

To optimize TCP throughput (assuming a reasonably error-free path), the sender should send enough packets to fill the logical pipe between the sender and receiver. Here is a good way to estimate the appropriate window size for any given logical pipe:

```
Capacity (bits) = path bandwidth (bps) * RTT (seconds)
```

So, consider a Gigabit LAN with a 2ms RTT. A window size of 249856 Bytes would optimize the use this fat (high-bandwidth) and short (low RTT) BDP. However, considering a MSS of approximately 1460 Bytes, the window size should be rounded down to 249660 Bytes (even multiple of the TCP MSS) in order to try and avoid fragmentation.

Location:

<HKLM>\System\CurrentControlSet\Services\Tcpip\Parameters

- **Tcp1323Opts**

  The Window Scaling Parameter. On a link with high bandwidth-delay product (satellite links – fat and long BDP) there may be a need to increase the window size to greater than 64K as larger receive windows improve performance over high-delay (high RTT), high-bandwidth networks. For that, you need to enable TCP Options as specified in RFC 1323 by appropriately setting this entry. To enable window sizes greater than 65,535, this registry entry should be set to 1 (decimal). After this change has been made, the registry entry controlling TCPWindowSize can be set to values larger than 64K (up to 1GB).

  Location:

  <HKLM>\System\CurrentControlSet\Services\Tcpip\Parameters

Tuning Recommendations for UNIX Systems

Consider the following parameters for basic tuning on UNIX systems:

- Semaphores
- Shared memory
- Process attributes
- File attributes
- Network attributes

Refer to the documentation provided with the specific operating system (e.g., Sun Solaris, Red Hat Linux, etc.) for more information on additional tuning for Java applications.

Database Tuning Parameters

CA IT PAM uses external databases for storage and will perform better if those databases are tuned to handle BLOB (Binary Large Objects) data in an efficient way. Consult the vendor documentation provided with your particular database for the appropriate tuning instructions for improving the performance, data integrity, and transaction management for that database. In particular, consult the recommendations for managing BLOB data.

JVM Tuning Parameters

Since one of the potential bottlenecks in an IT PAM environment is the JVM it is important to ensure that your JVM is properly tuned to suit your particular processing environment. Although CA IT PAM optimizes a number of JVM settings by default, additional adjustments may be necessary. Following are some JVM settings to consider.

Note: All JVM parameters discussed below are set through the following file:

`<CA IT PAM Install Folder>\server\c2o\bin\c2osvcw.conf`

Most of these settings are configured through `wrapper.java.additional.X` parameters, where `X` indicates the additional parameter sequence. Additional parameters can be added simply by adding new rows and incrementing `X` by 1 for each new row.

Permanent Generation

This parameter controls the section of the heap reserved for the permanent generation, which holds all of the reflective data for the JVM. This size can be increased to optimize the performance of applications that dynamically load and unload a signficant number of classes, eliminating the overhead of increasing this part of the heap.

By default, the value of `-XX:MaxPermSize` is set to 256MB for CA IT PAM r2.x.

Garbage Collector Types

Generational collectors were implemented in the J2SE Platform to emphasize application throughput and low garbage collection pause times. Although JVM supports several different types of collectors, CA IT PAM 2.2 uses the incremental low pause collector (the `-Xincgc)` parameter. This collector minimizes the cost of a major collection by collecting only a portion of the tenured generation each time it does a minor collection. Use of alternative collector types is not recommended and may cause unpredictable results and negatively impact processing.

Class Garbage Collection

By default, when there are no live instances of a class left, the JVM unloads that class from memory and this can degrade performance. CA IT PAM 2.2 turns class garbage collection off by setting the `-Xnoclassgc` parameter. Doing this eliminates the overhead caused by loading and unloading the same class multiple times. If a class is no longer needed, the space that it occupies on the heap is normally used for the creation of new objects, but if an application handles requests by creating a new instance of a class and if requests for that application come in at random times, it is possible that when the previous requester is finished, the normal class garbage collection will clean up this class by freeing the heap space it occupied, only to have to re-instantiate the class when the next request comes along.

Heap Size

Server applications usually experience problems with the default heap size, which is too small for most of those applications. To counteract this it is a good idea to grant as much memory as possible to the JVM.

Setting initial and maximum heap size parameters to the same value is good practice with server applications because it increases predictability by not letting the virtual machine make important sizing decisions. When the initial size is set to a value smaller than the value of the maximum size, not all of the reserved space is committed to the virtual machine immediately. As a result, the heap will shrink or grow to the limit of the uncommitted space (virtual) as needed.

By default CA IT PAM 2.2 sets the values of `wrapper.java.initmemory` and `wrapper.java.maxmemory` to 128MB and 1024MB respectively, but it is recommended that you further tune by using a fixed size 1GB heap. In other words, both, `wrapper.java.initmemory` and `wrapper.java.maxmemory` should be set to 1024.

```
wrapper.java.initmemory=1024

wrapper.java.maxmemory=1024
```

A few additonal comments related to the Heap Size are listed below.

- Heap size does not determine the amount of memory that the JVM will use. The amount of virtual memory used will exceed the amount specified for the maximum heap size because Java allocates memory for other things, such as the permanent generation, which is sized independently.
- More JVMs, i.e., heaps, outperform fewer JVMs with very large heaps. So instead of huge heaps, cluster CA IT PAM Orchestrators and balance work between them whenever possible.

Young Generation

The portion of heap memory dedicated to the young generation is very important. A large young generation means less minor garbage collection overhead.

The `-Xmn` parameter specifies how much space the young generation is allowed to consume on the heap and fixes it. Properly tuning this parameter can reduce the overhead of garbage collection, improving server response time and throughput. By default, this value is typically set too low, resulting in a high number of minor garbage collections. It is good practice to let the young generation consume a quarter of the total heap size. For example, for a 1 Gig heap this value should be set to 256MB (`-Xmn=256m`). A very large young generation would cause only major collections to occur.

To implement this, add `-Xmn=256m` to the `c2osvcw.conf` file as follows:

```
wrapper.java.additional.11=-Xmn=256m
```

As previously noted, "`11`" indicates the additional parameter sequence, which must be respected (your sequence number may vary).

Survivor Ratio

The Java heap is divided into a generation for old (long lived) objects and a generation for young objects, which is further subdivided into eden and survivor spaces.

Live objects are copied from one part of the young generation (eden space) to another part (survivor space). However, it is not guaranteed that all live objects will fit into the survivor space.

Therefore, enough memory must be reserved in the tenured generation to accommodate all live objects. Otherwise, the more expensive major collection will occur.

The `-XX:SurvivorRatio` parameter is used to specify the ratio between each survivor space and eden. Tuning the size of survivor spaces by increasing this ratio optimizes the JVM for applications with high object creation and low object preservation (more eden space). On the other hand, lowering it optimizes the JVM for applications that create more medium and long lived objects (more survivor space).

By default, `SurvivorRatio` is set to 32 but we have found that CA IT PAM performs better when a larger size is specified. For example if `-XX:SurvivorRatio=16` is used, each survivor space will be 1/18 of the young generation (there are two survivor spaces), indicating that objects tend to survive their first collection.

To implement this add `-XX:SurvivorRatio=16` to the `c2osvcw.conf` file as follows:

```
wrapper.java.additional.12=-Xmn=256m
```

Keep in mind that the "12" in the above example indicates the additional parameter sequence, which must be respected (your sequence may vary).

Distributed Garbage Collection

It is necessary to ensure that unreachable remote objects are unexported and that garbage is collected in a timely fashion.

The value of the Sun RMI `sun.rmi.dgc.server.gcInterval` property represents the maximum interval (in milliseconds) that the RMI runtime will allow between garbage collections of the local heap. The default interval of 60000 milliseconds (60 seconds) is too short as this is an expensive algorithm to run every minute.

Likewise, it is necessary to ensure that DGC clean calls for unreachable remote references are delivered in a timely fashion.

The value of the Sun RMI `sun.rmi.dgc.client.gcInterval` property sets the maximum interval (in milliseconds) that the RMI runtime will allow between garbage collections of the local heap. The default value is also 60000 milliseconds (60 seconds).

In lab tests we have noted better performance with less frequently distributed garbage collections by setting both Sun RMI properties to 30 (1800000 milliseconds).

To implement this add both Sun RMI properties to the `c2osvcw.conf` file as follows:

```
wrapper.additional.13=-Dsun.rmi.dgc.client.gcInterval=1800000

wrapper.additional.14=-Dsun.rmi.dgc.client.gcInterval=1800000
```

As previously noted, "`13`" and "`14" in the examples above` indicate additional parameter sequences, which must be respected (your sequences may vary).

Note: These properties only exist in certain implementations of RMI from Sun Microsystems. They are not part of the RMI public API.

JBOSS Tuning

The following JBOSS parameters should be considered for tuning:

- JDBC connections may also have to be increased when the Pool Size is increased
- Turn off JSP Compilation in Production
- If 70-80% of the Maximum Threads are being used for Incoming Requests, the Maximum Threads may have to be either increased, or a newer JBOSS node may have to be configured to load-balance the requests. In this case, an external load-balancer is required to distribute the incoming requests

Web Services and UI Tuning Parameters

As the incoming JBOSS requests increase, the JBOSS thread pool may have to be increased. The other option is to build a JBOSS cluster to load-balance incoming requests. This can be achieved by using an external load-balancer, like Apache Web Server, which can also provide additional parameters for load-balancing, prioritizing cluster nodes, and other tasks.

If incoming requests are close to the maximum queue size (within 70-80%), then clustering is a very good option.

On some multi-processor systems, it is possible to tie the JVM to specific processors for performance and serviceability.

The three important parameters for incoming web service requests are:

- maxThreads

  The maximum number of threads to be allocated for handling client HTTP requests.

- acceptCount

  The number of request threads that are put in request queue when all available threads are used. When this number is exceeded, the clients get a request timeout response.

- compression

  If this is set to "force", then the content will be compressed by JBOSS and sent to the browser. The browser will extract and display content. This will reduce bandwidth substantially.

IT PAM Orchestrators and JBoss Pool Size Tuning

In the previous sections we discussed tuning recommendations for the entire underlying infrastructure including OS, Network performance, Database, JVM and JBoss. In addition to these you should consider tuning the IT PAM Orchestrators, keeping in mind that all components are related to each other. A few important facts to consider include:

- The JMS pool size indicates the number of concurrent messages that can be processed simultaneously.
- Each Message Bean is associated with a single JMS Pool Object.
- Each Message Bean typically consumes 2-4 Database Pool Objects to handle a single message.

- The UI clients and other Orchestrator Threads (Reporter, Archiver, Scheduler, and others) consume at least 1 Database Pool Object each.
- It is recommended that the Database Pool Size be at least:
  - 2.5 times the size of the JMS Pool Size
  - Plus 20 for the Orchestrator Threads
  - Plus the number of UI clients expected to connect to the orchestrator.

The default configuration is set to process 25 concurrent messages while having 5 UI clients connected. For larger environment the JMS and Database pool size need to be tuned using the formula above and the described configuration details below.

Following are a few common parameters that you always should ensure are properly tuned:

- **Set Default Log Level to ERROR**

  In most cases a log level setting of "ERROR" is sufficient unless you are currently troubleshooting. This setting can be configured by using your favorite text editor (for example, Notepad) to modify the /server/c2o/conf/log4j.xml file.

  Note: Before modifying any configuration files ensure that you have a recent backup copy of those files.

  To modify the default log level set the "priority" value in the <root> category to the lowest level. For example, :

```
<root>
    <priority value="ERROR" />
    <appender-ref ref="CONSOLE"/>
    <appender-ref ref="C2OFILE"/>
</root>
```

  If you need to change the log level for a specific category for the purposes of troubleshooting ensure that:

  - Base thresholds for all referenced categories (appender) are set to "ERROR" unless you currently are performing any troubleshooting that require additional logging for this category. For troubleshooting purpose you can temporarily change these thresholds to WARN or INFO.
  - The "root" category includes the categories you are interested in (such as "C2OFILE" and "CONSOLE") and the "root" log level is set to the lowest level specified in any of these categories.

- **Tune "Engine Message Management Pool Size"**

  The "Engine Message Driven Bean" can be tuned by setting the "MaximumSize" property for the "container-pool" to an appropriate value. The default value of 30 is typically appropriate for most environments, but this could be tuned for larger environment.

  Note: Ensure that you have a recent backup copy of any files that you are planning to modify.

  This "MaximumSize"property can be found in the section outlined below in the file "/server/c2o/conf/standardjboss.xml".

```
<container-configuration>
  <container-name>Engine Message Driven Bean</container-name>
  <call-logging>false</call-logging>
  <invoker-proxy-binding-name>message-driven-bean</invoker-proxy-binding-name>
  <container-interceptors>
    <interceptor>org.jboss.ejb.plugins.ProxyFactoryFinderInterceptor</interceptor>
    <interceptor>org.jboss.ejb.plugins.LogInterceptor</interceptor>
    ...
    ...
    ...
  </container-interceptors>
  <instance-pool>org.jboss.ejb.plugins.MessageDrivenInstancePool</instance-pool>
  <instance-cache></instance-cache>
  <persistence-manager></persistence-manager>
  <container-pool-conf>
    <MaximumSize>30</MaximumSize>
  </container-pool-conf>
</container-configuration>
```

- **Tune "Database Pool Size"**

  The Database Pool Size can be tuned by setting the "max-pool-size" property for the "local-tx-datasource" section to an appropriate value. The default value of 100 is typically appropriate for most environments, but this could be tuned for larger environment.

  Note: Ensure that you have a recent backup copy of any files that you are planning to modify.

  This "max-pool-size" property can be found in the section outlined below in the file "/server/c2o/ext-deploy/oasis-ds.xml".

```
<local-tx-datasource>
    <jndi-name>OptinuityDS</jndi-name>
    <connection-url>... /snip/ ...</connection-url>
    <driver-class>${oasis.database.driver}</driver-class>
    <user-name>${oasis.database.username}</user-name>
    <password>${oasis.database.password}</password>
    <max-pool-size>100</max-pool-size>
    <exception-sorter-class-name>... /snip/ ...</exception-sorter-class-name>
    <check-valid-connection-sql>... /snip/ ...</check-valid-connection-sql>
    <metadata>
            <type-mapping>${oasis.database.typemapping}</type-mapping>
    </metadata>
</local-tx-datasource>
```

Process Flow Tuning

By default, an unclustered Orchestrator, with no Agents or Touchpoints configured, and with all ITPAM data (ITPAM library, queues and reporting) consolidated into a single database is configured with a database pool size of 100 and an Engine Message Bean Pool size of 30. In this configuration, the Orchestrator typically instantiates between 75-100 Process Workflows per minute. The actual number will depend on the concurrency of instantiation and on the number of ongoing transitions in the system.

IT PAM measures its performance in "Transitions per Minute". In the IT PAM process workflow engine a "transition" is defined as a single unit of work that is executed by the Engine to progress or complete a process flow. For example:

- Receiving an incoming message for an operations step
- Updating appropriate data results of the operation step
- Saving data results in the database for the operations step
- Generating the required outbound messages to progress or complete the process flow.

Each transition translates to the following database operations:

- Read from JMS Queue
- Update Operations Step Data (inbound)
- One or more (parallelism of branches) of the following:
    - > Read operation step data for the operation to be initiated next
    - > Calculate parameters and update operations step data
    - > Write to JMS Queue to progress or complete the process flow

Other active operations (in other words, those not in a state of transition) do not impact the performance of the CA IT PAM Process workflow engine.

Currently, the default Orchestrator configuration is set to approximately 150 "transitions" per minute. Transitions are similar units of work regardless of the service module operations carried out by the Orchestrator. Start and Stop operations take more time to complete their transitions due to the setup and teardown infrastructure for the process flows.

Based on this interpretation, increasing performance would generally require an increase in the number of transitions per minute. This can be achieved through the following means:

- Improved hardware. For example, using 64-bit hardware, faster or additional CPU, additional RAM, etc.
- JVM tuning
- Database tuning
- clustering