AppSense Environment Manager

Enterprise Design Guide

AppSense®
ENVIRONMENT MANAGER

Contents

Introduction

The purpose of this document is to clarify the architectural components of an AppSense Environment Manager solution, and to provide guidance and reference examples for scalability, high availability, and disaster recovery scenarios. This document does not contain detailed instructions on every possible permutation of server and database setup; however, the guide does give reference examples for common scenarios which should allow enterprise deployments to be molded to fit any customer environment.

Document Purpose

Document IS	Document IS NOT
Suggested strategies for building scalable, highly-available environments	A replacement for the Administration Guides for each respective product
High-Level overview of each topic	Detailed, step-by-step install instructions
Inclusive of other technologies, such as SQL Replication	Exhaustive in its treatment of peripheral technologies, such as SQL Replication

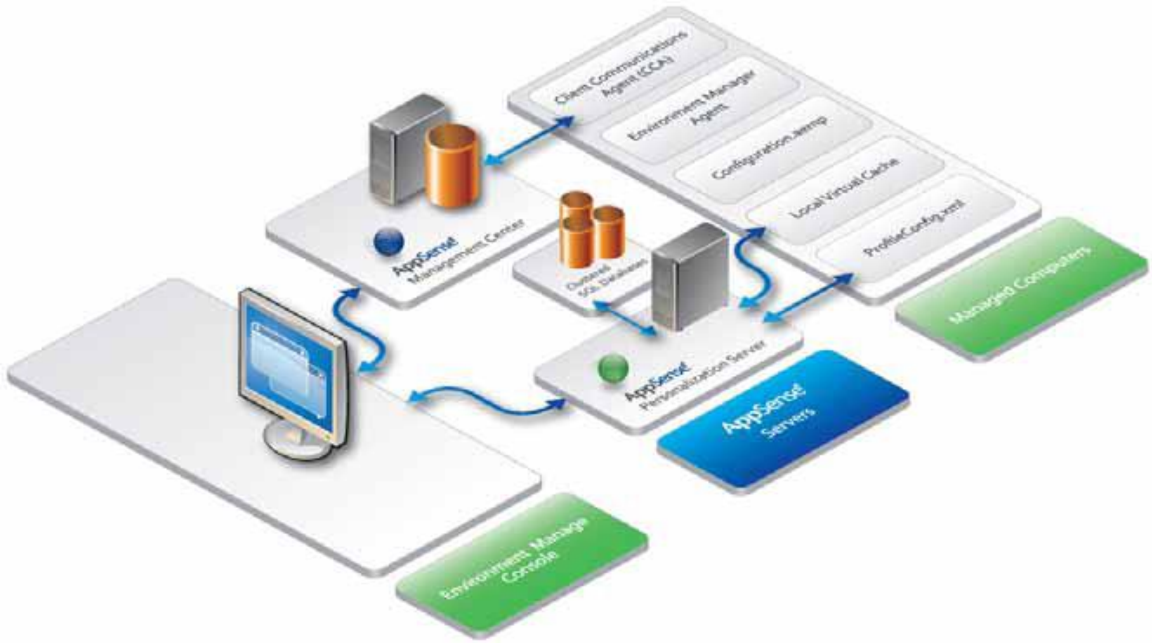
Basic Architecture

The fundamental requirements for installing and configuring AppSense Environment Manager, along with the AppSense Management Center, can be found in the AppSense Management Suite Installation Guide. This guide is available to evaluators and licensed customers at www.myappsense.com.

This guide will review the high-level basics of an architecture in order to build out scalability and high-availability scenarios in later sections. As detailed in Picture 1, a fully managed implementation requires two databases and two servers, which can be physical or virtual; scalability figures detailed in this document refer to a physical infrastructure. One server and database is dedicated to the Management Server and is responsible for storing and delivering AppSense agents and configurations to the managed endpoints. The Management Center is also responsible for centralized, enterprise auditing, alerting and reporting. The other server and database function as the Personalization Server, which together deliver application personalization to users on-demand on a per application, per user basis. This document refers to these components as the Management Server, the Management Database, the Personalization Server, and the Personalization Database.

Note

The Management Server is commonly used to deploy and manage the agents and configurations on the end point, however, 3rd party package delivery tools (ex. Microsoft SCCM) can be used for deployment if required. The agents and configurations can also be included as part of a gold build if deemed more suitable.


Client Communications Agent (CCA)
Environment Manager Agent
Configuration.aemp
Local Virtual Cache
ProfileConfig.xml
Managed Computers
AppSense Management Center
Clustered SQL Databases
AppSense Personalization Server
AppSense Servers
Environment Manage Console

Picture 1 – Basic Environment Manager Architecture

Common Components and Terminology

- **CCA** – **Client Communications Agent** – The CCA allows for communication between a managed endpoint device and the Management Server. The CCA ensures that the latest AppSense ™ Environment Manager Agent and Configuration are correctly deployed and installed on the endpoint. The CCA is also used to deploy other AppSense product agents and configurations (eg. Application Manager and Performance Manager, both of which are outside the scope of this document) The CCA communicates with the Management Server over either HTTP or HTTPS on standard defined ports. The CCA can be deployed in several ways; please consult the Administration guide for guidance on proper deployment of the CCA.

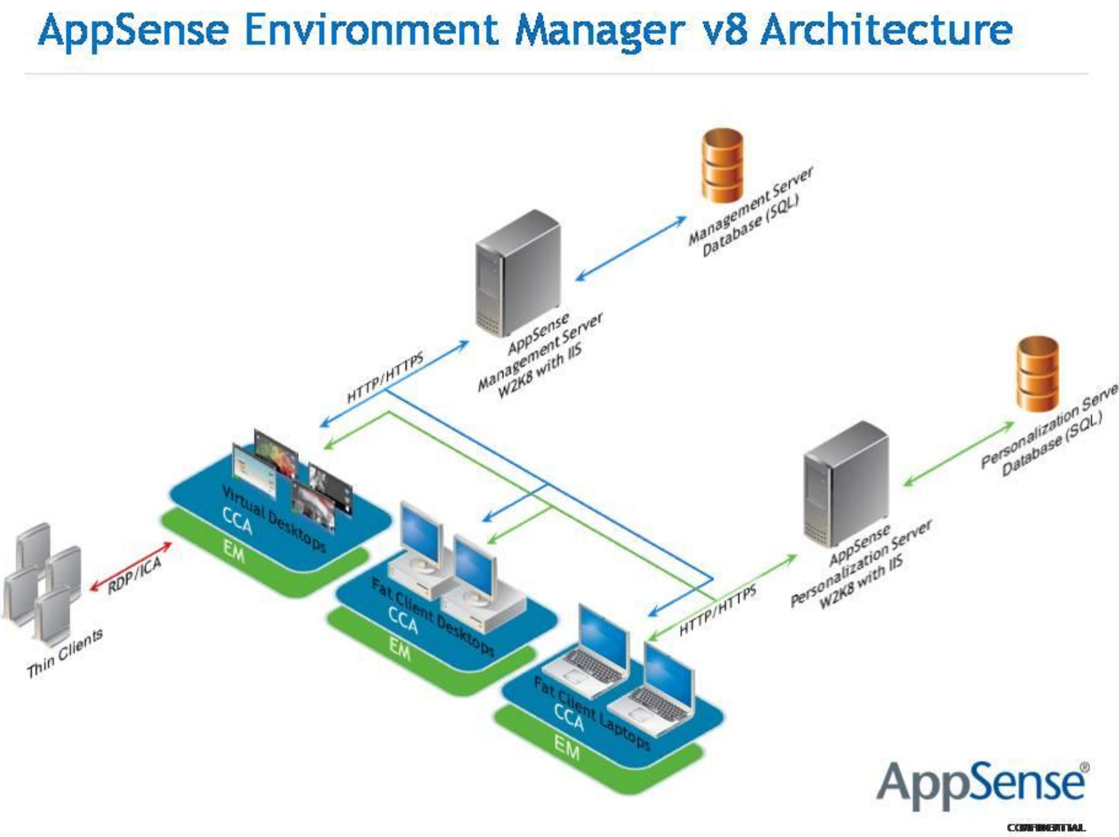
- **IIS** – **Internet Information Services** – IIS allows for the exchange of data between a managed endpoint device and the Management or Personalization server. IIS must be installed and enabled on the Management and Personalization servers prior to installation.

- **BITS** – **Background Intelligent Transfer Services** – The AppSense Management Server leverages BITS to perform the passing of data between the CCA and the Management Server. This allows the transfer of agents, configurations, and events to be bandwidth intelligent, although each agent is typically no more than 5 – 10MB and a configuration less than 1MB. BITS should be enabled on the Management Server prior to installation.

- **EM Agent** – **Environment Manager Agent** – The EM Agent is responsible for the delivery of both Policy and Personalization. Policy actions are stored in a local configuration (XML file) which is processed by the agent to carry out the necessary actions such as drive mappings or printer mappings. The EM agent is also responsible for delivering personalization when a user runs a managed application. EM Agents are stored in the Management Database so that they can be downloaded to managed end points via the Management Server or the EM agent can be pre installed in a gold build. There is a 32 bit and a 64 bit version of the EM agent and every managed endpoint needs to have the agent installed. The EM agent once installed runs amongst other processes, a primary service called EMAgent.exe

- **EM Policy Configuration** – The policy configuration is a package that is delivered down to a managed endpoint in the form of an MSI file. The CCA downloads this MSI configuration from the Management Server via BITS and installs it according to a defined schedule. The actual payload on the end point is an XML file containing all the rules and actions required for policy management, e.g. any policy settings for a given device and user.
- **EM Personalization Configuration** – this configuration helps the EM Agent manage the user's personality (profile) on a per application basis. This configuration is downloaded every time a user logs in, in the form of an XML file. It is not deployed or managed by the Management Center. The EM Personalization configuration is a copy of the configuration settings in the personalization database at the time the user logs on.

Picture 2, below, demonstrates the architecture discussed.


AppSense Environment Manager v8 Architecture
Management Server Database (SQL)
AppSense Management Server W2K8 with IIS
HTTP/HTTPS
Personalization Server Database (SQL)
AppSense Personalization Server W2K8 with IIS
HTTP/HTTPS
Virtual Desktops
CCA
EM
Fat Client Desktops
CCA
EM
Fat Client Laptops
CCA
EM
RDP/ICA
Thin Clients
AppSense
CONFIDENTIAL

Picture 2- AppSense Environment Manager Architecture

AppSense Environment Manager Best Practices

There are several design considerations that any AppSense Environment Manager architect should keep in mind when designing a solution. For the purposes of this document, High Availability is defined as the ability of a solution to continue to service its customers in the face of a component failure; scalability is defined as the per-user limits of individual components, and how to overcome those limits. As with any solution, these two factors often go hand-in-hand. This paper is broken into separate sections for HA and Scalability, but please be aware that any final design will almost certainly incorporate pieces of both, as well as elements of a Disaster Recovery strategy.

AppSense makes several recommendations for architectural best practices. These are general guidelines built on experience from existing enterprise customers and internal testing which result in the best user experience. The subsequent sections of this document provide more detail, while installation instructions can be found in the AppSense documentation Suite. These best practices include:

- **N+1 design for Personalization Servers** – Care should be taken to ensure that managed endpoints, and the users that log in to them, always have access to a functioning Personalization Server with an accessible backend database. This ensures the most reliable customer experience.
- **N or N+1 designs for Management Servers** As this server is not required for the continued operation of each endpoint or for Personalization, SLAs for the Management Server are often less restrictive than for the Personalization Server. Administrators will not be able to deploy new configurations while the Management Server is out of service, nor will auditing information be uploaded. Existing configurations installed on the end points will continue to function. Additionally, a failed Management Server has no impact on Personalization.
- **Crisp definition of user mobility requirements -** Database replication technologies can allow users to roam between two disparate geographical sites, even if those sites are serviced by separate Personalization Server architectures. However, if mobility requirements are relatively small, multi-site architectures can be simplified.

Scalability Designs

Both the Management Server and the Personalization Server have been designed to handle loads placed on them from enterprise conditions. Architectural decisions regarding the number of Management and Personalization servers needed to handle the expected load within a corporate environment should be made considering several factors, including number of users, geographic requirements, and high-availability requirements.

Management Server Scalability

Load testing performed by AppSense has proven that one Management Server can reliably manage up to 5000 endpoint devices. During tests run on a quad core Xeon processor with 4GB RAM, the Management Server was able to handle requests from approximately 140 clients per second. This would mean 40,000 clients would take just short of 5 minutes to process from a single server. If the polling period for updates to the managed endpoint was configured to be longer than 5 minutes the server would experience no issues. Event auditing information slightly slower to upload - uploading 100,000 events from a VM to the same quad core server mentioned above took approximately 3 minutes. A large volume of auditing information will also reduce the number of endpoint devices per server. 40,000 clients is a theoretical maximum number; AppSense best practice recommends roughly 5000 devices per server for reliability and to allow for auditing information to be collected from the environment.

If the number of endpoint devices is to exceed this number, additional Management Servers can be added into the architecture. The amount of supported devices can significantly be increased if the management server is only being used for deployment and not auditing, but the numbers quoted are based on a typical amount of audit information being collected.

Additional Management Servers can be configured by installing the Management Server software onto a new server. These management servers can be load balanced if desired, providing an increased number of supported devices and resiliency. Load balancing the Management server can be achieved using Microsoft Load Balancing discussed in the next section. Management servers can also be configured in a fail over scenario. Groups of managed endpoints are grouped into deployment groups. These deployment groups are based on machine name or OU membership. Each deployment group can be configured to communicate with a primary management server, which may be different to that of another group. Failover management servers can also be specified on a per deployment group basis.

In this example, a portion of endpoint devices (ex. All Windows XP machines) can be configured to rely on Management Server 1, and another portion of endpoint devices (ex. All Windows 7 machines) will rely on Management Server 2. This primary and failover management server architecture may also be used where multi-site implementations are required.

In larger deployments where the number of end points exceeds 5000 machines or where multi-site management is required, it is regarded good practice to load balance management servers at each site in addition to utilizing different management servers per site. Eg: New York machines will communicate with load balanced management servers for NY, where as the New Jersey machines will communicate with a number of load balanced management servers for NJ

Management Database Scalability

The amount of data tracked by common enterprise deployments – agents for each AppSense product, plus configurations for each – is a relatively small amount. Internal testing has shown that a single database can reliably serve more than 50,000 endpoint devices. If the number of endpoint devices is expected to exceed this number, standard SQL practices can be employed to increase capacity. Common examples are adding disks to the supporting disk infrastructure to increase IOPS load, and adding clustering technology between databases for load balancing.

Personalization Server Scalability

Load testing performed by AppSense has proven that one Personalization Server can reliably manage up to 7200 concurrent user connections on a single quad-processor server. This scalability factor is determined by the number of users active before the response time of a Personalization Server increases to more than one second. If the number of concurrent users is to exceed this number, additional Personalization Servers can be added into the architecture.

There are several considerations when adding multiple Personalization Servers into an environment. If the total expected user count is below 7200 and the second Personalization Server will be providing N+1 reliability, please see Section 4.3 for more details. When adding additional Personalization Servers to scale an environment beyond the 7200 user limit, some form of load balancing solution should be implemented, as AppSense Environment Manager does not provide for native load balancing. For single-site operation, each of these added Personalization Servers can address a single database; up to prescribed database limits (see Section 3.4 for details).

AppSense supports Windows Network Load Balancing (NLB) to help balance traffic and load between multiple IIS Servers. AppSense provides a detailed NLB setup guide at www.myappsense.com. The guide is available from the Downloads Section, in the AppSense Management Suite group. Also, administrators should note that all traffic to and from a Personalization Server is standard IIS traffic. Therefore, most industry hardware load balancing solutions should work with no issues. As AppSense validates additional solutions, it will post statements of support at www.myappsense.com.

Personalization Database Scalability

Scalability of the Personalization Database has been shown to be related to the number of users, the average profile size of users, and the frequency at which users open and close managed applications. Therefore, database performance will be unique to each environment. Load testing performed internal to AppSense has shown that more than 20,000 users can be supported on a single SQL 2005 database under the following assumptions:

1. Average profile size of 812k
2. 26 Logons per second
3. 252 Application starts per second
   (source -internal testing)

Performance numbers vary greatly with storage capacity and layout (as with any database). Additionally, AppSense recommends standard SQL tuning practices, including the separation of log and data files and SAN storage arrays for increased numbers of disks that can be applied to data files.

There are several methods to estimate final database size. Administrators can examine the size of any current roaming profiles that are in place. Since Environment Manager stores much more granular information than a traditional roaming profile, administrators can assume that a user's footprint in the database will be less than the size of the roaming profile. Empirical data suggests that using 50% of a roaming profile as a per-user footprint is a reasonable estimate. This number should then be multiplied by the number of snapshots that are required to be kept. By default, AppSense Environment Manager will keep 5 snapshots per user. This estimated footprint can then be multiplied by the expected number of users in an environment, to gain a reasonable estimate for overall database size. Table 1 below gives example numbers.

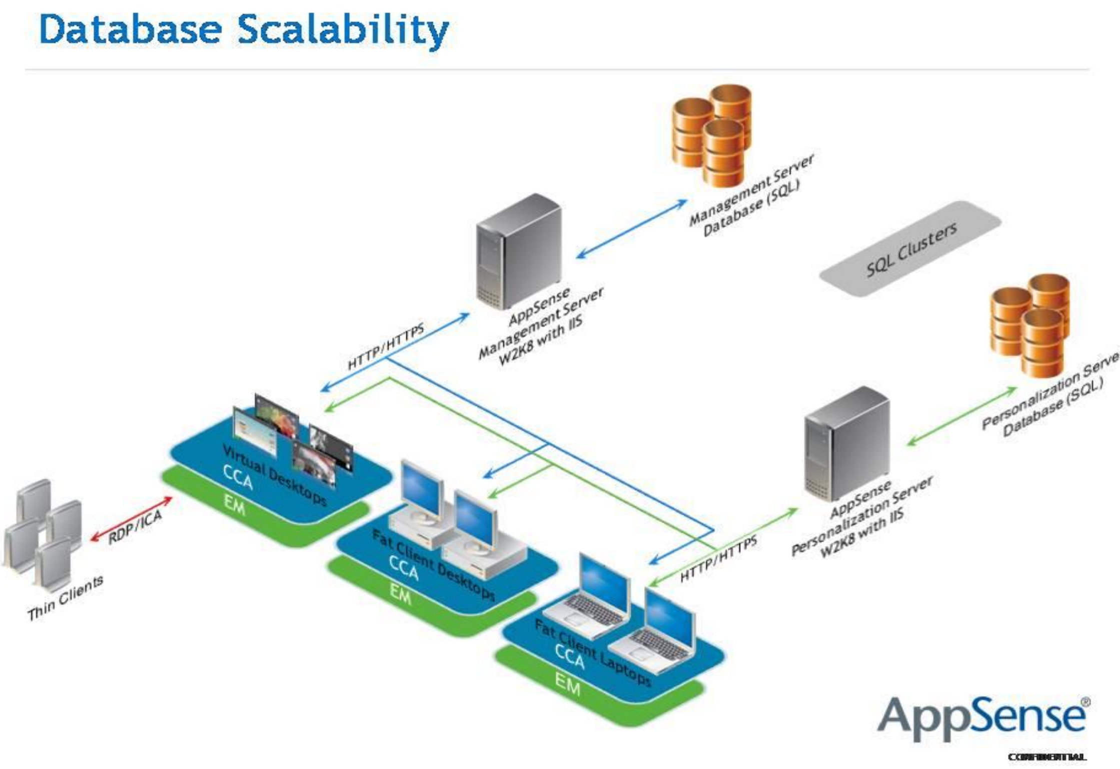
Component	Estimated Database Footprint
Roaming Profile	15MB
AppSense Footprint Size	50% of roaming
Per-user Footprint	7.5MB
Number of Snapshots	5
Total Per-User Footprint	37.5MB
Number of Users	5000
Total Estimated Database Size	187.5 GB

Table 1: Example ***Estimated*** Personalization Server Size Calculation

High-Availability Designs

Database High Availability

AppSense recommends employing standard SQL clusters for database high availability. Technology included in SQL Server 2005 and SQL Server 2008 provides for very resilient architectures, which can help ensure the constant availability of the Management and Personalization Databases. In many cases, an enterprise will already have a SQL infrastructure with rigorous HA requirements. Picture 3 below shows a highly-available AppSense architecture featuring SQL Clustering.



Picture 3 – SQL Clustering added to an AppSense Architecture

Management Server High Availability

Failover Server Configuration for HA

The AppSense Management Center provides the ability to directly configure a set of failover servers for Management. In the event of a primary Management Server failure, the Client Communications Agent will attempt to access a secondary or failover server.

Administrators can configure a second Management Server which can reference a single backend database, shared among both Management Servers. The **Failover Servers** node allows you to maintain a list of failover servers which can take over the role of the Management Server in the event of a connection, hardware or environment failure, when decommissioning a Management Server, conducting an update or overhauling a Management Server. Failover servers can also be configured on a per deployment group basis.

The general process for setting up a highly-available Management Server architecture is the following:

1. Install the Management Server onto the first server
2. Allow this initial installation to create and populate the Management Database
3. Install Management Server onto each additional server
4. During installation of secondary servers, choose connect to a remote database instance rather than create a new database instance

The Client Communications Agent (CCA) on managed computers downloads the list of servers and maintains the list as a reference. If a Management Server is unavailable, the managed endpoint attempts to register with the next available server in the list. The list of servers consists of one or more URLs. Each URL can specify a server using the server NetBIOS name, the fully qualified domain name or the IP address.

The failover servers can be maintained in the default list which applies to all deployment groups and in local lists for each deployment group. Local deployment group lists override the default settings. The failover settings are maintained in the following locations of the Management Console:

- Home > Failover Servers
- Deployment Groups > ... > Settings > Failover Servers

Details for configuring multiple Management Servers are given in the *AppSense Management Center Administration Guide*. The administration guide is available from www.myappsense.com.

Network Load Balancing for HA

Windows Network Load Balancing technologies provide for High Availability in the case of a server failure, by detecting failed nodes in the NLB cluster, and routing traffic to other, available nodes. AppSense provides a detailed NLB setup guide at www.myappsense.com. The guide is available from the Downloads Section, in the AppSense Management Suite group. Also, administrators should note that all traffic to and from a Personalization Server is standard IIS traffic. Therefore, most industry hardware load balancing solutions should work with no issues. As AppSense validates additional solutions, it will post statements of support at www.myappsense.com.

Personalization Server High Availability

Failover Server Configuration for HA

The AppSense Environment Manager Personalization Server contains built-in High Availability functionality. As per AppSense best practice, all Environment Manager architectures should have at least two Personalization Servers to ensure users and devices never lose connectivity to a Personalization Server. Administrators can configure a second Personalization Server which can reference a single backend database. By adding this additional Personalization Server to the Sites area of Environment Manager, the secondary Personalization Server can take over if the primary Personalization Server fails.

The general process for setting up a highly-available Personalization Server architecture is the following:

1. Install the Personalization Server onto the first server
2. Allow this initial installation to create and populate the Personalization Database
3. Install Personalization Server onto each additional server
4. During installation of secondary servers, choose connect to a remote database instance rather than create a new database instance
5. During Policy Configuration, add all Personalization Servers to the server list
6. Once you have enabled Personalization for a configuration, ensure that all Personalization servers appear under the Sites work area

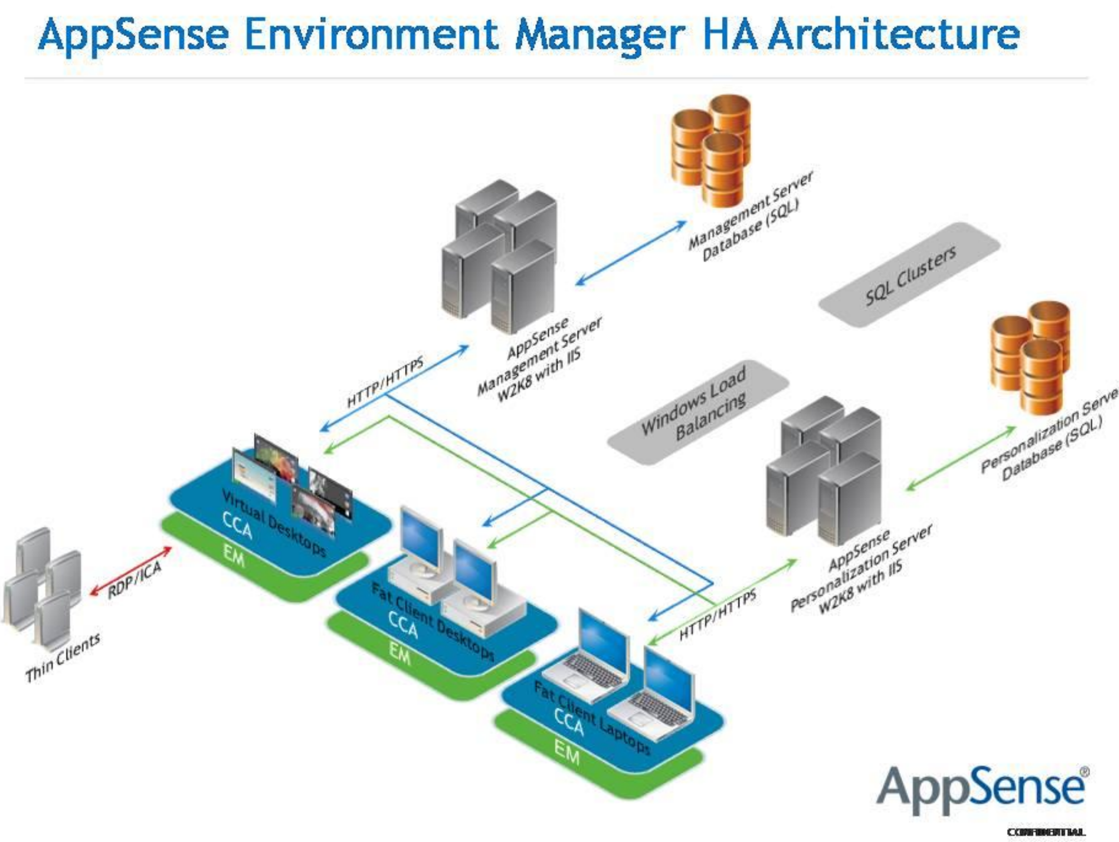
When creating a highly-available Personalization Server architecture, make sure to add both of the available Personalization Servers to the wizard when the *Enable Personalization* button is pressed. This action ensures that multiple Personalization Servers are defined in the configuration file that is deployed on each managed endpoint. Upon boot, each managed endpoint will use this fixed list of Personalization Servers to pull down any updated information that is needed. Once this boot activity is complete, the Environment Manager agent uses the list of Personalization Servers defined in the Sites work area to determine which server to query for personalization information when a user runs an application. However, administrators need to ensure that the servers listed in the configuration file are valid for proper configuration information to be downloaded at boot time.

Details for configuring multiple Personalization Servers are given in the *AppSense Environment Manager Administration Guide*. The administration guide is available from www.myappsense.com.

Network Load Balancing for HA

For larger environments where the user count is planned to exceed 7200, Network Load Balancing technologies provide for High Availability. In the case of a server failure, NLB can detect failed nodes in an NLB cluster, and route traffic to other, available nodes. While NLB is more commonly associated with balancing traffic to and from available servers, it also offers HA functionality, and can simplify large deployments where multiple Personalization Servers are needed for performance, rather than solely for high availability. An NLB solution is recommended for both HA and load balancing in >7200 user environments, as per AppSense best practice. AppSense provides a detailed NLB setup guide at www.myappsense.com. The guide is available from the Downloads Section, in the AppSense Management Suite group. Also, administrators should note that all traffic to and from a Personalization Server is standard IIS traffic. Therefore, most industry hardware load balancing solutions should work with no issues. As AppSense validates additional solutions, it will post statements of support at www.myappsense.com.

A fully highly-available, single-site implementation of AppSense Environment Manager with the Management Server installed looks like Picture 4, below.


AppSense Environment Manager HA Architecture
Management Server Database (SQL)
SQL Clusters
AppSense Management Server W2K8 with IIS
HTTP/HTTPS
Windows Load Balancing
Personalization Server Database (SQL)
AppSense Personalization Server W2K8 with IIS
HTTP/HTTPS
Virtual Desktops
CCA
EM
Fat Client Desktops
CCA
EM
Fat Client Laptops
CCA
EM
RDP/ICA
Thin Clients
AppSense
CONFIDENTIAL

Picture 4 – Highly-available AppSense Environment Manager architecture

Multi-Site Designs

A multi-site design can take into account users in remote or branch offices, worldwide employees, as well as help ensure a consistent experience for mobile employees who may need to log in to workstations in different geographies.

Key Considerations

There are several different considerations that will drive a multi-site architecture. Architects should investigate each requirement with their customer in order to properly design the architecture.

Geographic Locations

How many different sites will have managed endpoint devices, and how many users will occupy each site? Depending on the bandwidth between locations, a large branch office could require a dedicated Management Server and Personalization Server.

User Mobility

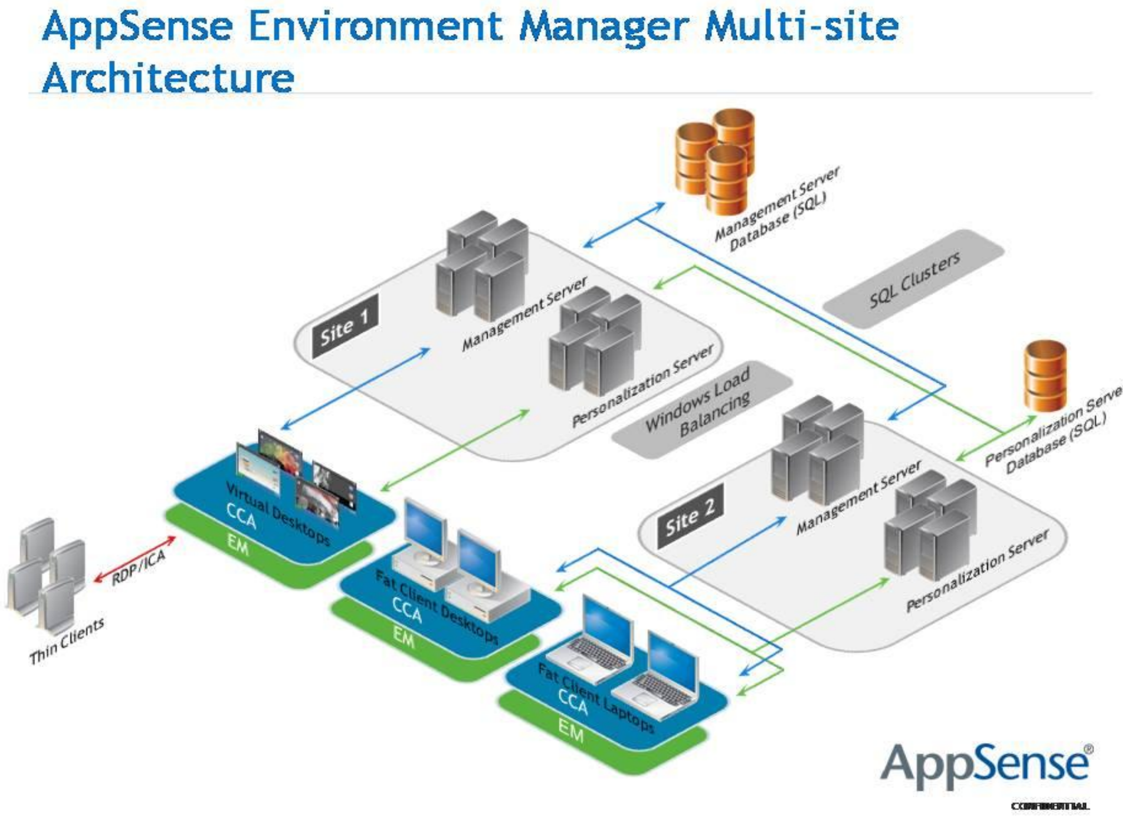
Are there users who will need to log in to a managed endpoint device in multiple locations? Or are all users in a given site only expected to consume local resources? If all users are local, then many companies will set up individual local Environment Manager architectures. However, if there are user mobility requirements, then Personalization Servers at each site will need to have access to consistent information. This can be accomplished with sufficient bandwidth back to a centralized database server, or by utilizing database replication between individual sites.

Bandwidth

What is the available bandwidth between sites? For corporations with sufficient bandwidth between remote sites and central IT, a single, centralized database farm can provide the needed backend infrastructure for all remote sites. Otherwise, some form of SQL Replication should be enabled to allow for local-site databases, while maintaining architecture-wide consistency.

Additionally, available bandwidth and network latency (round-trip time) will determine if local Personalization and Management Servers are necessary.

Picture 5 below shows one possible multi-site configuration. In this case, Site1 and Site2 both serve enough users to warrant local, dedicated Management and Personalization Servers. These local servers are utilizing Windows Network Load Balancing for both load balancing and failover protection. The local devices and users connect to those servers for information as needed. However, the servers at each of these two sites have sufficient bandwidth to connect back to a single, dedicated SQL Server farm, which is providing two highly-available databases, one each for Management and Personality. In this example, replication is not needed since the databases are not disparate. Such a configuration is likely for configurations in different buildings that are still proximal to each other. Such a configuration would not be appropriate for sites on separate continents, for example.



Picture 5 – Highly available multi-site AppSense Environment Manager architecture

Personalization Server Replication

AppSense Environment Manager provides tools to enable SQL Replication between multiple disparate SQL Databases. Using these tools, Administrators can ensure that Personalization information is consistent between multiple sites. Consider a configuration with locations in Los Angeles and London. Each of these locations will have local, dedicated Management and Personalization Servers, with proper HA designs in place as per AppSense best practice. It is not practical for a user in Los Angeles to leverage a database in London for personalization (nor the other way around). Thus, each location will need a dedicated, highly-available database solution. To allow for users in London to have their personalization information available in Los Angeles, AppSense Environment Manager provides a set of SQL scripts to enable database replication. AppSense utilizes a **merge replication** solution, which ensures that both databases are in sync. Replication schedules can be customized by administrators. Additionally, AppSense Environment Manager supports multi-node replication.

Setting up SQL Replication for Personalization is detailed in the Environment Manager Administration Guide, in the Appendix titled *Personalization Server Replication*. Your AppSense sales representative can provide this documentation, and it is also available from www.myappsense.com.

The above example will want to take advantage of the Sites work area inside of Environment Manager, to ensure users always access the local database. Sites contain a list of Personalization Servers that should be used by managed endpoints that match a *membership rule*. Membership rules are specific to a site, and define which end point devices fall into a given site. AppSense recommends having managed endpoint devices in separate Active Directory Computer OUs. In this situation, a site can check against OU membership to define what personalization server a given device should connect to. Using the example from above, PS1 is in Los Angeles, and PS2 is in London. Additionally, all managed endpoint devices in Los Angeles should be in an LA-specific OU, and all managed endpoint devices in London should be in a London-specific OU. From your Environment Manager console in the Personalization area, an administrator would build two new Sites. Site 1 should include a membership rule that checks for inclusion in Computer OU Los Angeles, and Site 2 should include a membership rule that checks for inclusion in Computer OU London. Site 1 should then list the Los Angeles personalization servers as resources, and Site 2 should list the London personalization servers as resources. Thus, if a user logs in to a workstation in London, that device will match the Site 2 membership rule, and the Environment Manager agent will be directed to use the London personalization servers. Care should be taken that EM Policy conditions are valid in both locations if users are planning to roam in this manner (ex. Active Directory user authentication and group membership should be the same in London as it is in Los Angeles).

Disaster Recovery Considerations

By choosing to standardize on common Windows IIS components, along with Microsoft SQL Server as a backend, the AppSense product suite gains many benefits for disaster recovery. A simple database restore from a backup provides a functioning Management or Personalization server with the backend needed to resume functionality. Long-distance SQL clusters, or log-shipping performed to an off-site SQL host can provide near-term recovery from a major event. Similarly, the Management and Personalization servers can be recovered in an off-site location, and re-configured to utilize the recovered databases to provide for rapid recovery. The recommendation of this guide is to integrate the Management and Personalization servers into an already-existing Disaster Recovery plan for Application Servers, and to integrate the Management and Personalization Databases into a Disaster Recovery plan for Database Servers.

A combination of load-balancing and failover techniques can serve as a basic disaster recovery plan. If locations are close enough such that databases can be accessed remotely with acceptable degradation in performance, administrators can configure each site to be a failover Personalization Server for the other. Picture 5 above can be extended to include Disaster Recovery capabilities by implementing local databases at each site, and then replicating between them. Note that each site already includes a set of NLB Personalization Clusters handling personalization and management configuration duties at each site. By making location 2 the secondary Personalization Server for Site 1, and location 1 the secondary Personalization Server for Site 2, each are protected. Should site 1 go down, a managed endpoint would fail to locate NLB cluster #1. By failing over to NLB cluster #2, operations can resume (likely in a degraded-performance state). However, with connectivity, business can continue until repairs are made.