Web Infrastructure Concepts for SAP Web Application Server

SAP AG
Neurottstraße 16
69190 Walldorf
Germany
T +49/18 05/34 34 24
F +49/18 05/34 34 20
www.sap.com

Copyright

Icons

The following icons are used in this document as visual aids.

Icon	Meaning
	Caution
	Example
	Note
	Recommendation

Disclaimer

SAP AG assumes no responsibility for any errors or omissions in these materials.

Contents

Preface

This document provides you with background information to help planning a technical infrastructure for SAP Web Application Server (SAP Web AS).

The purpose of this guide is to:

- Give you an understanding of the technical requirements of SAP Web Application Server.
- Explain design criteria and solutions for those requirements.
- Provide guidelines for planning your technical system landscape.
- Illustrate a range of technical solutions from small test and demo systems all the way up to fully scaled, highly available and secure setups.

Who Should Read This Document

Use this guide as a starting point for planning the technical infrastructure for your SAP Web AS. It is written for anyone interested in the technical implementation aspects and IT infrastructure for SAP Web AS. This includes:

- System architects
- IT managers responsible for implementing and operating applications based on SAP Web AS
- System integration consultants

Status and Version History

October 2002: First public release.

Source and Feedback

You can find this document and related ones on technical infrastructure topics in the SAP Service Marketplace at http://service.sap.com/ti. If you do not have access to that Web site send email to network@sap.com. Please use this address also for any kind of feedback regarding this document.

1 Network Requirements of Business Applications

SAP Web AS is a general purpose Web application server that can be used for a broad variety of applications. SAP uses SAP Web AS basis technology for all business applications. Therefore, most SAP products like CRM, BW and even R/3 are based on SAP Web AS or will be in the near future.

Any kind of Web application can be built using SAP Web AS. In this paper, we concentrate on business type applications, their requirements and how to support them with a suitable network infrastructure. When designing a technical infrastructure for a SAP Web AS application, you should consider these requirements.

The requirements of SAP Web AS are actually not different from the requirements of other business Web applications. Business applications have requirements that may differ in part from those of other Web applications. Let us have a look at those requirements and what the technical consequences are:

- **CPU intensive —** Each request for a Web page typically consumes quite a lot of CPU cycles. Conventional Web sites using static pages or even if they offer some personalization are much less CPU intensive. This causes a strong need for horizontal scalability in business Web application. To provide this scalability, a load **balancing mechanism** has to be used. Most of this paper deals with how to realize load balancing for SAP Web AS.
- **Dynamic content —** A large fraction of the content is created dynamically. The fraction of static content (like Pictures or static HTML pages) is low.
- **Transactional —** In many cases, business applications are more complex than your average informational Web site or even a Web shop. Many times, a session state must be held in the application server between successive requests. This imposes the requirement of **session persistence** for the supporting infrastructure.
- **Access control —** Fine granular access restrictions based on a variety of authentication methods are used.
- **Security of communication —** Network traffic **encryption** (using SSL) is usually necessary for business applications. This requirement imposes special requirements for load balancing that we will discuss in detail below.
- **Security of application servers —** Systems that process business data usually need to be protected very efficiently from network based attacks as well as attacks on application layer. Providing secure access is a great technological challenge. Security considerations have to be included into the technical architecture from the very start, because they influence just about any other design decision you make.

In the following sections, we will discuss the consequences that these requirements have on the network infrastructure and the options you have to fulfill these requirements.

2 Server Load Balancing

This section explains how the requirements of **load balancing, session persistence and encryption** using Secure Sockets Layer have to be reconciled for a load balancing design.

We start with an overview of load balancing mechanisms. Then we cover some other aspects like high availability and configuration that are also closely related to the load balancing mechanisms.

An SAP Web AS system consists of one or more servers, just like any other SAP system. Small systems can be run with a single application server instance only, in SAP terminology this is the so-called central instance. For a larger application, or one that needs to provide some level of high availability, **multiple application servers** are usually needed. Note that load sharing and high availability uses the same mechanisms, therefore these two aspects always have to be considered together.

Classical SAP applications are operated through the SAP GUI user interface. It has a built in mechanism for load balancing through the so-called SAP Message Server. For applications operated through the general purpose Web browser, another way to achieve **load balancing** (**and fail-over** in case of failure of an application server) has to be used. In the following sections, we present some load balancing mechanisms and discuss their features and how they can be used together with the SAP Web AS.

2.1 Technology Overview

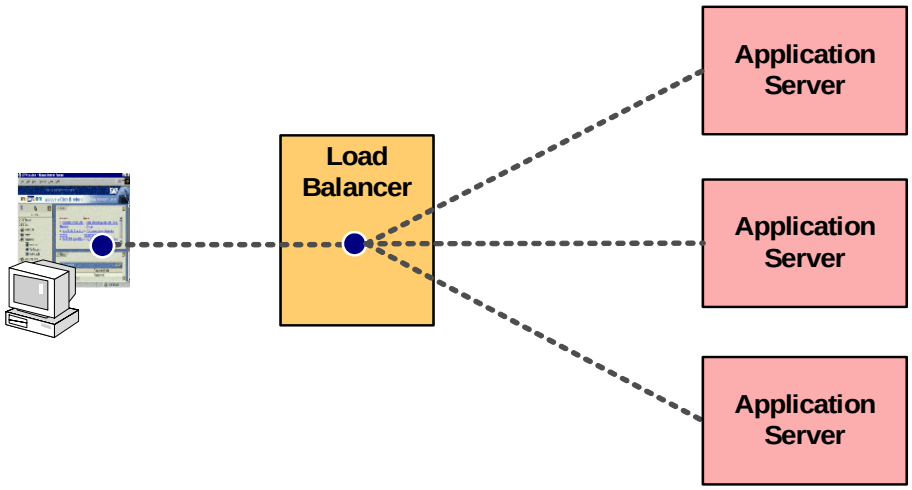
This section gives an overview of the different load balancing mechanisms available for SAP Web AS. We distinguish two general mechanisms for load balancing: client based and server based. SAP generally **recommends server-based load balancing** mechanisms.

Load Balancing Device

In general, it is preferable to have a **single point of access** to the system, even if the system consists of multiple application servers. This yields the following advantages:

- One official name for the service required, one IP address to reach all application servers
- Same URL every time the user works in the system
- One SSL server certificate sufficient for all servers
- All this results in reduced cost of operation and maintenance and greater flexibility.

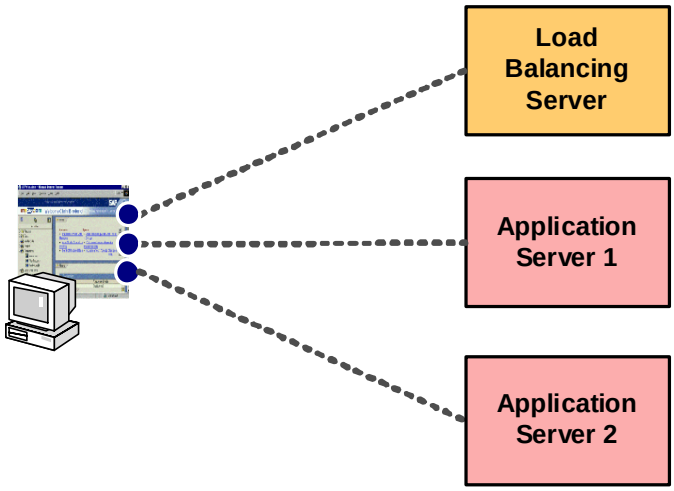
To grant a single point of access to the system you usually need an additional component called a "load balancer". This device accepts incoming requests and directs them to one of the application servers as illustrated by the following graphic:



Using a load balancing device is usually the **preferred method of load balancing**. We will discuss the general features of Web load balancing devices and requirements of SAP Web AS in more detail below. Specific solutions, like the SAP Web Dispatcher, will be described in another article.

Client-Based Load Balancing Technologies

Using a load balancer is not the only way to accomplish Web server load balancing. There are alternative mechanisms that may even be favorable under certain conditions, especially when ease of implementation is a high priority. We call these alterative mechanisms "client based load balancing", because they work in the following way: When making a connection to the Web application server, it first contacts some load balancing device that tells it, to which server it should connect. This is illustrated by the following graphic:

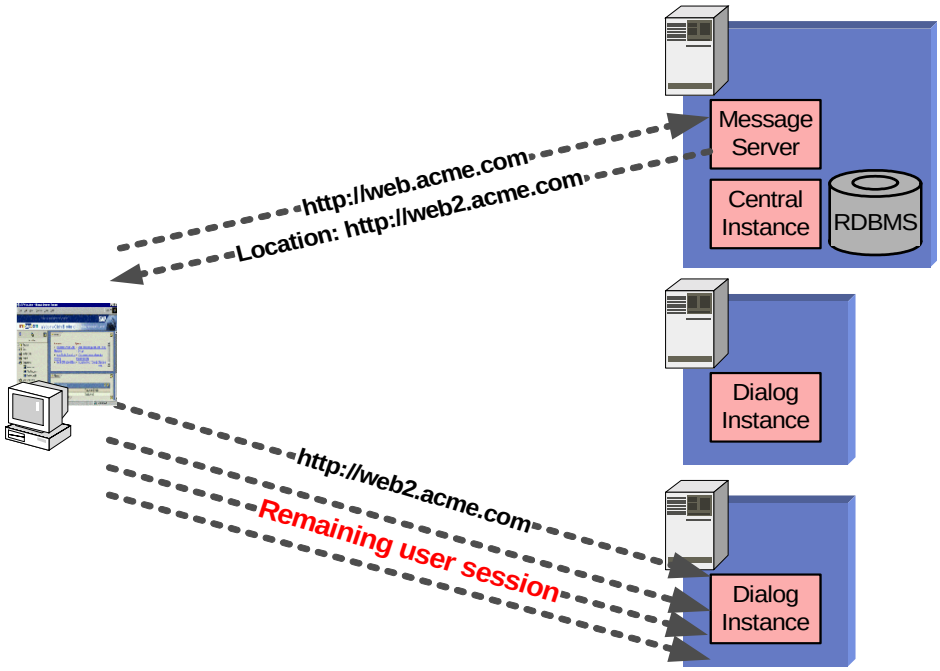


There are two client based load balancing mechanisms, which are explained in the following sections:

- Redirections use features of the application protocol HTTP.
- DNS based methods use features of the Internet name resolution protocol DNS.

Redirections

A simple redirection based load balancing mechanism is built into SAP Web AS. It closely resembles the SAP GUI mechanism. It works as shown in the following figure:



The mechanism functions as follows:

1. Browser sends request to Message Server
2. Message Server sends HTTP redirection to suitable Application Server
3. Browser sends request to this Application Server
4. User stays on this Application Server for remaining session

The SAP Message Server decides, which Application Server is most suited to handle the next user. The decision is based on the relative "strength" of each Application Server and the number of users already logged on to each Application Server. The strength is based on the number of work processes and other factors.

In addition, the Message Server takes into account whether the request is directed toward an application running in the ABAP or the J2EE environment. For the load balancing decision, it takes into account only appropriate servers.

A weighted round-robin mechanism is used for load distribution.

Redirections are not limited to the SAP message server. If you have an infrastructure that already supports load balancing using redirections, you can integrate SAP Web AS into this infrastructure as well.

The redirection-based mechanism is easy to understand and set up, comes with SAP Web AS out of the box and it has no problems with persistent sessions. However, the mechanism has many disadvantages:

- **Confusing —** The user may be confused, because the redirection is usually shown in the URL field of the browser. This raises questions like: Can I trust this other server? Has my session been hijacked and now I am giving my personal information into the system of some hacker?
- **Inconvenient —** Using the standard bookmarking feature of the browser may lead to undesirable results, because the bookmark points to the specific server that the user happens to be redirected to. The bookmark may not work later, because the server may be down. Even if the bookmark does work, it circumvents the load balancing mechanism.
- **Many network addresses —** Every server needs an official external DNS names and IP address

- **Expensive** — The Server certificate must match the name of the server. Therefore, you need one certificate for every server. Server certificates are quite expensive and have to be renewed every year. Administration of the server certificates and IP addresses generates a high administrative overhead.
- **Firewall-unfriendly** — Redirections do not work well with firewalls, because they point to the internal names of the application servers. There are ways to circumvent this problem by re-writing the redirection responses in the firewall or reverse proxy. However, it is not the most elegant solution.
- **Authentication problems —** Superfluous authentications arise when starting new application on a different application server (after fresh load balancing decision). If basic authentication is used, the authentication information is not sent to the new server, because it is sent only to the exact same server that originally requested the authentication. This can be avoided by enabling cookie-based authentication. Cookies are sent to all servers within the same DNS domain.

To wrap it up into a single statement, load balancing with **redirections is just not "Web-like"**. In general, we recommend against this mechanism for production Web sites. This is true especially for Internet applications.

For intranet applications, the issues of trust, IP addresses and costly certificates are usually not a problem. On the cost side, the redirection mechanism comes with the system and works out-of-the-box. There is no need for extra hardware or configuration. There is also no need to read most of the rest of this paper, because it mainly deals with the issues associated with load balancing devices. Therefore, in spite of some drawbacks, redirection-based load balancing may be your acceptable for your intranet application.

Round-Robin DNS

Round Robin DNS is a simple load balancing mechanisms for any kind of network services. The Domain Name System (DNS) is used to map the name of a server to an IP address. Round Robin DNS maps DNS requests to a set of IP addresses belonging to the available Web servers in a round robin fashion. This way, different clients will use different IP addresses for reaching the same network name, therefore being distributed amongst different hosts.

Round Robin DNS can be used with the SAP Web AS. However, it is usually **not the preferred method**, because in the event of server failure, the client is not switched to a new server, because the DNS lookup information is usually performed only once during a browser session. The user has to close his or her browser to establish a new connection. Also you have to implement some method to do health check of the servers and remove non operational servers from the DNS. There are additional problems with session persistence when users come in through proxy servers similar to the problems related to IP address based persistence discussed below.

We will not discuss Round Robin DNS in more detail here.

Global Server Load Balancing

The concept of Round Robin DNS has been refined by some network equipment vendors. The name "Global Server Load Balancing" has become quite common for this new technology. Such devices can perform health checks of the servers and make a decision based on the geographical position of the client and the server. The same mechanisms are used for ensuring availability of the system in case of disasters or major network interruptions.

Usually, global server load balancing does not replace load balancing in each local data center. Here, we concentrate on local load balancing. Global server load balancing is **beyond the scope** of this paper.

2.2 Stateful Web Applications

In section *Network Requirements* we stated that applications running on SAP Web AS are often transactional in nature. In this section we explain how this fact influences the load balancing mechanism and discuss the different solutions with their pros and cons. This subject is a bit complicated, but it is the **fundamental basis** for building a good solution for your project.

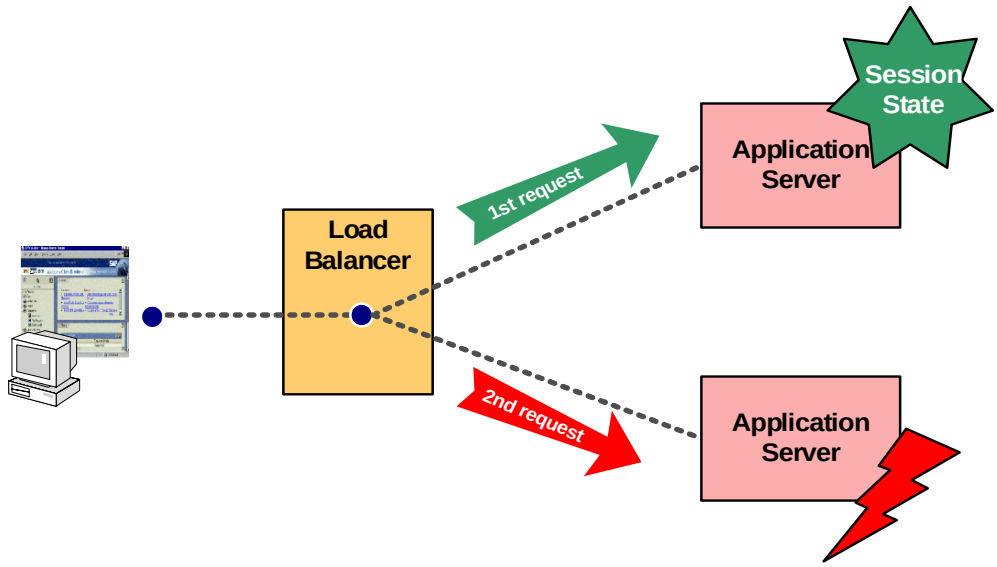
The programming model used for developing the Web application has an important impact on the required features of the load balancer and, in effect, the entire Web architecture. The two different programming models are **stateless** or **stateful**.

A stateless programming model can be used for simple applications. Here, every request to the SAP Web AS is independent of any previous request. For example, browsing a catalog and selecting a single item from the catalog can usually be implemented using a stateless model, because every request specifies exactly what the user wants to see.

Applications that are more complex often require that information about the state of the user session be kept in the application server. The most common example is the shopping basket of a shop application. In business applications, the user session can also hold locks on database objects, buffer the result of complex queries for future use etc. In the SAP Web AS the user session is realized as a so-called "roll area" – just the same way as in traditional SAP GUI applications. Since the stateful programming model is well supported by the SAP Web AS, many applications are realized this way.

In consequence, a **load balancing mechanism must support stateful applications**. Only under the unlikely condition that you know that you will never carry out any stateful applications in your system can you chose a load balancing mechanism that does not support stateful applications. We do not recommend this, because if requirements change, the architecture of your system has to be changed.

Stateful applications impose special requirements on the load balancing device. The reason is that **HTTP is a stateless protocol**, because the network connection does not last for the duration of a user session. The protocol itself offers no provisions to return a subsequent request to an already established session. There is a conflict between the application (stateful) and the protocol (stateless). The problem is illustrated by the following figure:



The first request from user establishes session in one of the application servers. If subsequent request would be directed to a different server, session context is not known there so context would be lost. Even worse, if the first context holds any locks, second session would not be able to access locked items.

2.3 Persistence Mechanisms

The load balancing device must ensure that all requests belonging to an application session are directed always to the same application server. In this section you will learn how this can be achieved and what problems arise in connection with SSL encrypted HTTP communication.

Cookies and/or URL Rewriting

The stateful application uses content of a so-called "cookie" that identifies the session ("session cookie") to identify the application session associated with a request. A cookie is a small string that the server sends to the browser and that is returned by the browser with every subsequent request. As an alternative to cookies, which some users chose to disable in their browsers, applications can put a session identifier into each request URL that is part of the HTML age returned to the browser. Both methods are supported by the SAP Web AS.

Since the application server uses the session cookie or the URL session identifier to identify the session, the same can be done by the load balancing device. The **load balancing device uses the same session identifier** in the HTTP header and create a mapping between cookie content and application server. In the case of SAP Web AS it is even simpler, because the hostname of the applications server is part of the session identifier. See the specification of SAP Web AS session identifier for details.

This means that the load balancer has to have some knowledge of the structure of SAP's session identifier in order to be able to assure session persistence. This configuration of how to find the session cookie has to be done manually (for non-SAP load balancers) or can be pre-configured in the load balancing device. A pre-configured device simplifies implementation.

An alternative is offered by many Web load balancing devices: They can insert a separate session cookie into the HTTP data stream by themselves. This method is independent of SAP's persistence mechanism. This method works only with cookies, but not with rewriting parts of the URL. Therefore it fails if the user has chosen to disable cookies in his or her browser.

There is one problem in the entire method of using cookies or URL session identifiers: It can only be used **only for clear text HTTP**, if SSL is used the entire data stream (including the HTTP request header) is encrypted.

To make things worse, SAP strongly recommends to use applications that require **user authentication only with SSL**. The reason is the following: Usually, the session cookie or URL session identifier or some other part of the request is also used to authenticate successive requests after logon has been completed. Therefore, if the cookie could be "stolen", it would enable an intruder to act on behalf of the original cookie owner. If clear text HTTP is used for communication, this could be accomplished, e.g. by capturing network packets between browser and server.

To use SSL in conjunction with cookies or URL session identifiers for session persistence, we have to **terminate the SSL encryption in the load balancer**. Whether this is acceptable is a very severe design decision, therefore we will discuss the pro's and con's of this in a dedicated chapter below. Once the SSL session is terminated, the load balancer can examine the request and therefore the session cookie or URL session identifier can be used for ensuring session persistence.

Client IP Address

The client's IP address can be used to identify a user session. Actually this description is not accurate: The client's IP address can be used to identify the host that issues the requests. This is identical with the user session only if there is only one user working on the host and the user has only one concurrent session.

In spite of this limitation, the method of using the client IP address for session persistence works well for most applications. However, there are some potential **problems** with this method when users access the system through Web proxy servers:

Users from same IP address are associated to same session and thus routed to the same server. This is the case for all **users coming in through the same proxy server**. The server can distinguish between the sessions, so session integrity is OK, but load balancing is not optimal. In the intranet, an extreme case could be that a large group of users (for example, users in a subsidiary) or even all users access the system through a single proxy. Here, the method fails completely. A similar scenario would be a business-to-business application with a partner company whose users access your system through a Web proxy.

Some Internet Service Providers use **proxy load balancing** among different proxy servers resulting in different IP addresses for the same user and possibly within an application session. Usually, these IP addresses differ only in the last byte (are in the same class C subnet). Thus the problem can mostly be circumvented by configuring the load balancing device to treat all clients in the same class C subnet as equal. However, the quality of load balancing may suffer even more from this.

Another weakness of this method is that the device **can not determine when** an application **session is terminated** and so the load balancing decision lasts for a longer time than necessary. With the cookie method, the cookie can be deleted when the user logs off the system explicitly. Restarting the browser also forces a new load balancing decision to be made, because then the session cookie is forgotten. This also has no effect if the client IP address is used for persistence.

If you use both stateful and stateless applications on the same system, the load balancer **has to treat all requests as stateful**, because it has no way to check whether they really are.

Redirections

Redirections have no trouble with session persistence. With redirections, the browser is redirected to another server directly during the load balancing process. The user stays on this server for the remaining user session, unless the application sends him or her explicitly back to the message server to enforce a new load balancing decision.

Things That Do Not Work

The **SSL session ID** was once deemed a way to get session persistence without the weaknesses of the IP address method while preserving the end-to-end SSL session. Unfortunately, it has turned out that the SSL session ID is not suitable to ensure the integrity of application sessions, because the lifetime of SSL session does not correspond to the lifetime of a user session. The browser decides after a fixed period of time to end the SSL session and start a new one, regardless of user activity at that time. On some browsers this time is only 2 minutes, but even if it is longer, the end of the SSL session is never synchronized with the end of a user session.

It is a frequently asked question, why it is not possible to use the **TCP port** of the client instead of the IP address. The reason is again that HTTP is a stateless protocol. The TCP session may be terminated at any point of time. The subsequent HTTP request just opens a new TCP session with a new port. This is even the case if the HTTP 1.1 keep-alive mechanism is used. Connection keep-alive, like the SSL session ID, can only serve as a performance enhancements, not for session persistence.

2.4 Load Distribution Schemes

The load balancer uses an algorithm to determine for every incoming request which application server to direct the request to. Of course, it has to make this decision only for request without valid session information, because requests that belong to a persistent session have to be routed to the application server that holds the user session.

The load balancing algorithm can be based on one of the following mechanisms:

- Round robin — Distribute requests evenly between application servers.
- Weighted round-robin — Like round robin, but give strong servers more requests than weak servers. The weighting factors can be determined by
    - CPU power of the application server or
    - Number of work processes configured in the application server.
- System load — This usually requires installing a software agent on the server hosts for load measurement.
- Application load — SAP currently has not implemented an interface for using application load for Web load balancing.
- Number of open connections or number of requests per second — This also requires maintaining weights for individual application servers.
- Response time — The load balancer measures the time it takes each application server to return a response. It uses this time to determine which application server is most suitable to handle the next request.

Weighted round robin is usually sufficient. More elaborate mechanisms may gain a few percent of performance, but usually do not make a very large difference.

2.5 Configuration

The load balancer has to know about the configuration of the SAP Web AS system. The minimum knowledge it needs is the names or IP addresses of the application servers. Here is a list of information that could be useful for the load balancing device:

- Names and/or IP addresses of the application servers — mandatory
- Session ID used for persistence — The load balancer should know the session identifiers used by SAP Web AS. This is not required if IP address based persistence is used or if the load balancer inserts its own session ID into the HTTP data stream (cookie insertion).
- Processing power of the application servers
- Health checks — The load balancer must know how to perform health checks on the application servers.
- Logon groups and the corresponding application servers — Logon groups can be encoded into the URL or mapped onto special URL prefixes.
- Different services offered by the application servers — J2EE only, ABAP only, ABAP and J2EE
- Mapping of URL path prefixes to service type (J2EE, ABAP)

The specification of SAP Web AS's session identifier can be obtained from SAP. To request it, send an email to network@sap.com.

2.6 High Availability

The second major reason for using load balancing is to ensure high availability of the SAP Web AS. The load balancer has to be able to **determine failure of an application server**. Then it directs requests only to the application servers that are still alive. It should also be able to detect when a failed application server comes **back online**.

The load balancer itself should not become a single point of failure, either. Usually, a redundant configuration with at least two devices and a suitable fail-over mechanism is used to ensure availability of the load balancer.

All network components have to be included in a high availability concept. This includes wide area connections to Internet service providers. Using connections with two separate Internet providers is also used frequently to reduce probability of downtime.

The SAP Web AS system itself should be configured for high availability. This typically involves fail-over mechanisms for the database and central instance.

For more information on high availability, see http://service.sap.com/ha.

2.7 Summary

For optimal usability and maintainability of the SAP Web AS system we recommend providing a single point of access to the system using a **load balancing device**. You can decide whether to use a load balancing software like a reverse proxy or a hardware device like a Web switch. We will discuss some possible solution in another paper.

The must ensure **session persistence**. If SSL is used, you can choose between two ways to ensure session persistence. The persistence mechanism is related to the question whether to terminate SSL in the load balancer in the following way:

- **Terminate SSL** in the load balancer and use the **session ID** for persistence.
- **End-to-end SSL** from browser to application server and use **client IP address** for persistence.

We discuss the pros and cons of terminating SSL more detailed in section *Security* to give you a better basis for this important decision.

Another factor that influences the choice of load balancing device is the method of selecting the most favorable application server. Usually, a **weighted round robin** scheme is sufficient. However, if the device offers other – perhaps more elaborate – mechanisms, these can also be used with SAP Web AS.

The load balancing device also has to be integrated with the **high availability** strategy for the entire system.

Ease of configuration is a very important aspect. First, you should decide whether you will need complicated load balancing rules like SAP's logon groups, or whether simple load balancing among a static group of application servers is sufficient. Then chose the device that makes it most easy to set up and administer your system. If you have expertise for a certain device already, it is preferable to use it also for your SAP Web AS system.

3 Security

In this section we discuss security aspects of the load balancing and network infrastructure for SAP Web AS. Unfortunately, load balancing and security are so closely related to one another that it is impossible to discuss one without the other. Therefore, this section and the preceding one belong very closely together.

Here, we focus on the aspects of security that are related to load balancing and the overall network architecture of the system. Special aspects like intrusion detection, operating system security, firewall technologies are beyond the scope of this paper.

3.1 SSL Encryption

As we stated above, business type applications usually involve the transmission of data that is not publicly available. Therefore, you need to **protect confidential data**. However, protection of data is not the most important point. What is more crucial is the fact that users usually have to authenticate themselves when connection to a system with business applications.

Authentication is done through a variety of mechanisms, for example, by providing a user name and password. In SAP applications, authentication of subsequent requests is done by cookies that provide single sign-on (SSO) capabilities. If an eavesdropper would listen in to the connection and read passwords or cookies off the wire, he or she could then use this information to log on to the system and most likely a number of other systems in the name of the user that the data was stolen from. This **risk of stolen authentication** is usually far larger that of stolen content.

Authentication can also be accomplished by using client side X.509 certificates. This form of authentication relies on a secret key, which never leaves the client computer. It is therefore safe in the way that an eavesdropper is not able to fake authentication. It is used only in conjunction with SSL encrypted connections.

Therefore, we can postulate the following general rule:

All applications the require user authentication need to use SSL encryption.

This applies to most applications that run on SAP Web AS, with very few exceptions (like public catalogs or information retrieval Web sites). Therefore, a typical **infrastructure for SAP Web AS must support SSL**.

3.2 Terminating SSL in the Load Balancer

This section gives you a comprehensive overview of the advantages and disadvantaged of terminating the SSL session in the load balancing device as opposed to routing the SSL session directly through to the application server. The latter is also called end-to-end SSL.

This discussion affects the security of the solution as much as the quality of the load balancing mechanism. The two aspects just cannot be separated from each other.

We start with a list of the pros and cons of SSL termination in the load balancing device and discuss them in depth in the following sections:

Advantages:

- SSL offloading (off the SAP Web AS)
- Persistence based on Cookie or URL
- Possibility to inspect and filter requests

- Early authentication possible

Disadvantages:

- Potentially reduces data security
- Client authentication with X.509 certificates not possible

SSL Offloading

Handling **SSL is expensive** in terms of CPU usage. Especially the handshake, during which asymmetric cryptographic methods are used to exchange a symmetric key for bulk data exchange, is very CPU-intensive. The computational effort is also strongly influenced by the type of browser that connects to the system. Older browser versions tend to consume (by large factors) more CPU cycles than current browsers.

Specialized network devices (usually called "SSL Accelerators") usually do a much better job handling SSL than the general-purpose application server CPU. Therefore, using such devices may significantly **increase performance and decrease the total cost** of the system.

Cookie based persistence

With end-to-end SSL, all HTTP data is invisible to the load balancer. This includes the URL and all HTTP headers, therefore also the session cookie. Therefore, only the client IP address can be used to ensure session persistence. As we have explained in detail above, using the client IP address to ensure session persistence has many problems. Especially in the Internet, it does not work well. Therefore, for Internet access it is preferable to terminate the SSL session in the load balancer so the **session cookie** can be used **for session persistence**.

Data Security

Clear text communication between load balancer and SAP Web AS is potentially unsafe. An network intruder could potentially listen in to the conversation. He or she could then break into the existing user session or, potentially even more dangerous, steal the authentication token and use it to impersonate the user in the system or possibly in another system where the user name and password is also valid.

Therefore, if clear text communication between load balancer and the SAP Web AS is to be used, use **network security mechanisms** to prevent this kind of attacks. These mechanisms may include the use of modern switched network architecture or sub-divided demilitarized zones combined with intrusion detection systems and efficient security management.

As an alternative, use a load balancer that is able to **re-encrypt the communication** using SSL. This way the communication between load balancer and SAP Web AS can be protected.

Note that re-encryption retains the advantage of SSL offloading. The expensive part of SSL is the handshake that has to be carried out with every new client. For subsequent connections, the handshake does not have to be performed, because the keys used by the SSL session can be re-used by the communication partners. Negotiation of existing keys is done using the so-called "SSL session ID". However, there is a small performance penalty caused by the bulk data encryption.

Terminating the SSL encryption potentially enables **man-in-the-middle attacks**, even if the communication is re-encrypted. However, by appropriate configuration of the SAP Web AS it is possible use an SSL client certificate to authenticate the load balancing device at the SAP Web AS. This means that an attacker would have to be in control of the very host on which the load balancer is located in order to interfere with the communication.

Load Balancer Trusted Component

A load balancer with end-to-end SSL that does not break the SSL session between browser and SAP Web AS has no way of interception or modifying the data. Therefore, it does not need to be trusted into to a high degree. On the other hand, if the load balancer does break the SSL session and possibly implements other functions that the SAP Web AS relies on, it has to be much more trustworthy. Therefore, great care has to be taken to make the load balancer by itself a trustworthy component and to make the configuration robust against attacks.

Authentication with X.509 Client Certificates

Client authentication using X.509 client certificates currently **requires that the SSL session is terminated by the SAP Web AS**. There is currently no way to terminate the SSL session elsewhere and pass just the certificate information to the SAP Web AS for authentication.

Note: Such a mechanism does exist for SAP Internet Transaction Server, but not for SAP Web AS directly.

Therefore, if you want to use cryptographic client authentication using X.509 certificates in the SAP Web AS you must use end-to-end SSL.

Early Authentication

User authentication can be done only after the SSL session is terminated. This is true for client certificates as well as for user name/password, cookies or any other authentication scheme. Therefore, if you require the user **authentication before the request reaches the SAP Web AS**, the SSL session must be decrypted before reaching the SAP Web AS.

For example, the load balancer could authenticate users using data from a corporate LDAP directory.

Request Filtering

When using an end-to-end SSL connection, there is no way to **examine the communication** between the client and the server from the outside. It is not possible to filter requests, examine payload for viruses, detect attacks against the application. It is not even possible to ensure that the HTTP protocol is used.

If you want to filter requests before the request is passed to the SAP Web AS, you have to terminate the SSL session. Most load balancers offer some methods of filtering. The following are examples of filter rules that could be used to secure the system:

- **Mal-formed URLs —** Reject requests with URLs that contain characters outside the usual range allowed by URL-encoding.
- **Over-sized requests —** If you know that your application never uses requests that exceed a certain size, reject all requests that exceed this maximum allowed size.
- **Request method —** If your application uses only certain methods (typically GET and POST), allow only such requests.
- **Application filtering —** This is the most interesting filter. An SAP Web AS system usually contains many applications. The mapping of URL paths to the applications is done using transaction SICF. The settings made here are valid for every application server belonging to the SAP Web AS system. Consider a system that is both used from the Internet and the corporate intranet; however, only one application should be accessible from the Internet. Even an attacker that possesses the authentication of a highly privileged user in the system should not be able to use the other applications when logging on over the Internet.

The only way to achieve this is to allow only requests for the URL paths corresponding to the allowed applications. This can be done only after the SSL has been terminated.

- **Enforce user authorizations —** After successful authentication of the user, the load balancer can employ user-based filtering rules to enforce, that the user can call only applications for which he or she has the proper authorization. Of course, the users and their respective authorizations have to be made known to the load balancer I some way. This is typically done using directories.

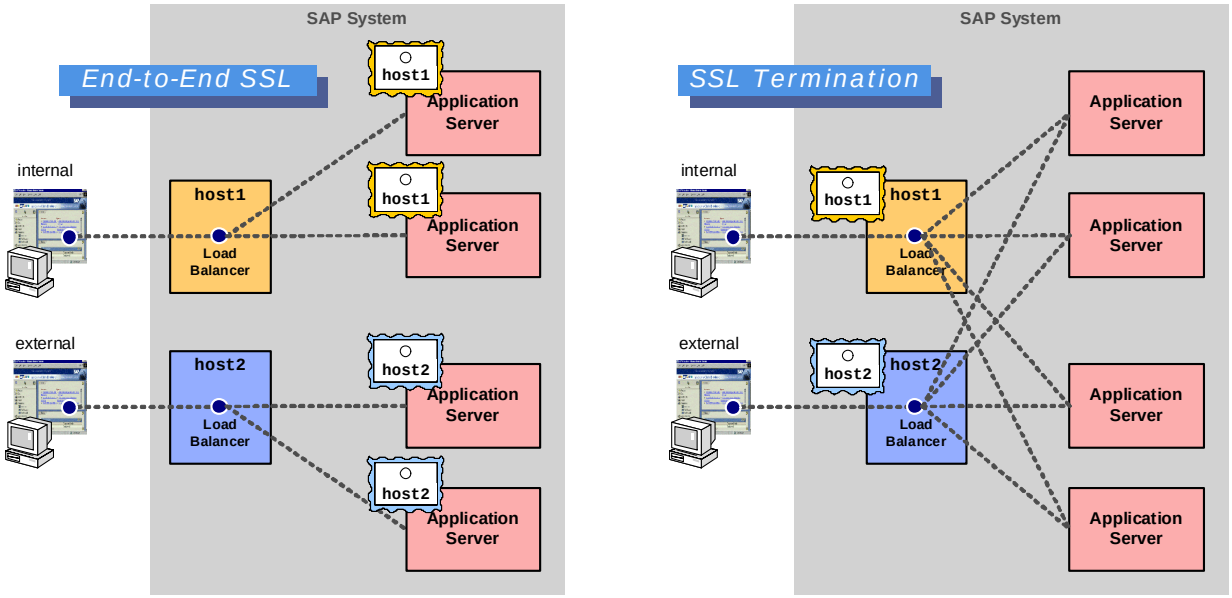
Multiple Names For One System

The **server certificate of a Web server must be issued to the host name in the URL** that the client uses to access the server. When using end-to-end SSL, all servers accessed through one load balancer are addressed by the same name. Therefore, they can share the same server certificate. This is good, because you need fewer certificates if every user accesses the system using the same name.

However, things get more complicated if you need to access the system using **different names**. For example, you may access a system using the URL https://shop.acme.com from the Internet and https://shop.internal-domain.com from the corporate intranet. Another example is **application hosting**, where one system in the service provider network has to be reachable from multiple customer networks.

Now the question arises: which certificate does the server present to the client? The answer is that every application server instance of the SAP Web AS system have only one server certificate. In consequence, for each name a **dedicated group of application server instances** must serve only users accessing the system with this name. All servers in this group must be configured with a certificate issued to this name.

The figure on the left illustrates this fact:



The figure on the right shows the same situation with **SSL terminated in the load balancer**. Here, the server certificate is located on the load balancer only. Therefore, every entry point can send dispatch request to every application server instance.

Summary

The decision whether to terminate SSL in the load balancer is **crucial for the architecture** of the system. Unfortunately, we cannot say that either one is better or more secure than the other. It really depends on many different factors that are determined by application requirements and/or security policies and requirements.

End-to-end SSL could be considered more secure against network based attacks. It is a requirement if you want to use X.509 client authentication. On the other hand, it makes effective load balancing impossible.

Terminating SSL allows or much better load balancing, especially in the Internet. In addition, you may consider it more secure against attacks on application layer because it enables request filtering in front of the SAP Web AS. However, it requires great care to secure the network communication between the SSL end-point and the SAP Web AS.

3.3 Network Security Zones

This section describes where the system components should be located in a firewall scenario.

Firewalls are used to prevent network attacks against the system. A typical firewall system consists of the following components:

- An Internet **access router** with simple filter rules
- A so-called "**demilitarized zone**" (DMZ) or "secure server network". Servers that are directly accessed from the Internet are located here. Typically, Web servers, mail servers and application proxies for other Web related applications that reside in internal networks.
- Another **filtering system** that allows tightly controlled access from servers in the DMZ to internal networks and/or vice versa. This is often done by specialized firewall products (like Checkpoint Firewall-1, Cisco PIX or Raptor). Often these products combine the function of access router and inner firewall into one product with three (or more) network interfaces: One external interface, one DMZ interface and one internal interface. The administrator can configure the filtering rules between these interfaces.

Read more about security of SAP applications at http://service.sap.com/securiy.

For the SAP Web AS you can chose between two basic architectures that we discuss in the following sections.

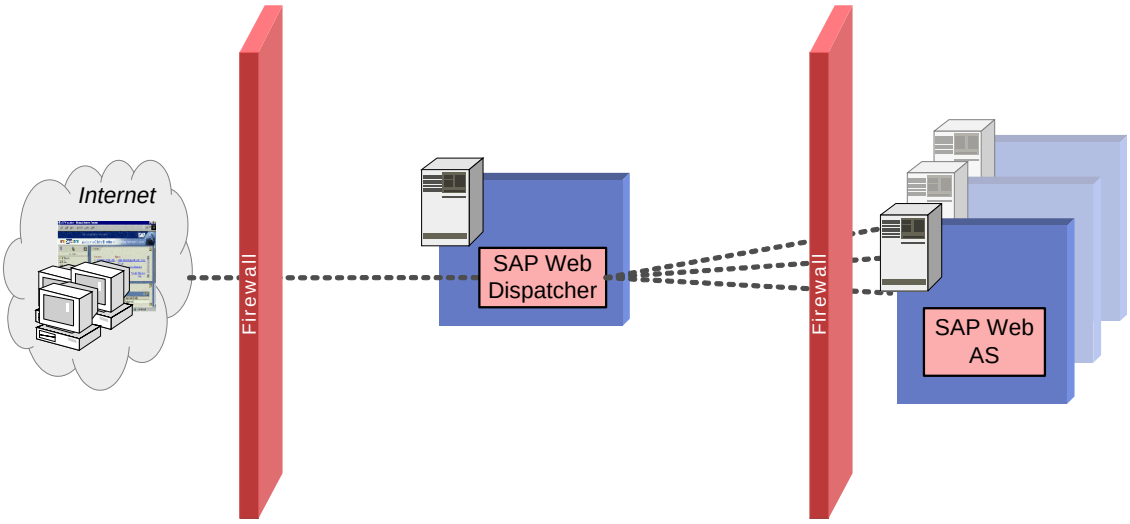
Client Access to the Business System

In general, we recommend placing the **SAP Web AS in the internal network** and not in the DMZ. Requests from the Internet reach the Web AS through an **"application layer gateway" located in the DMZ**. The load balancing device can play the role of this gateway, be it a network device like a Web switch or a program running on a general purpose computer like SAP Web Dispatcher (see below for details).

Filtering of requests on the application layer (if applicable) can be carried out both by the load balancer and/or by the inner firewall that routes requests into the internal network. This serves as protection against attacks on the application layer. Protection against denial-of-service (**DoS**) attacks can be achieved by limiting the maximum number of requests that are routed through to the system.

Filtering of network packets using appropriate criteria has to be carried out both by the access router and by the inner firewall.

The following graphic illustrates this setup showing the Web Dispatcher as an example for an application layer gateway:



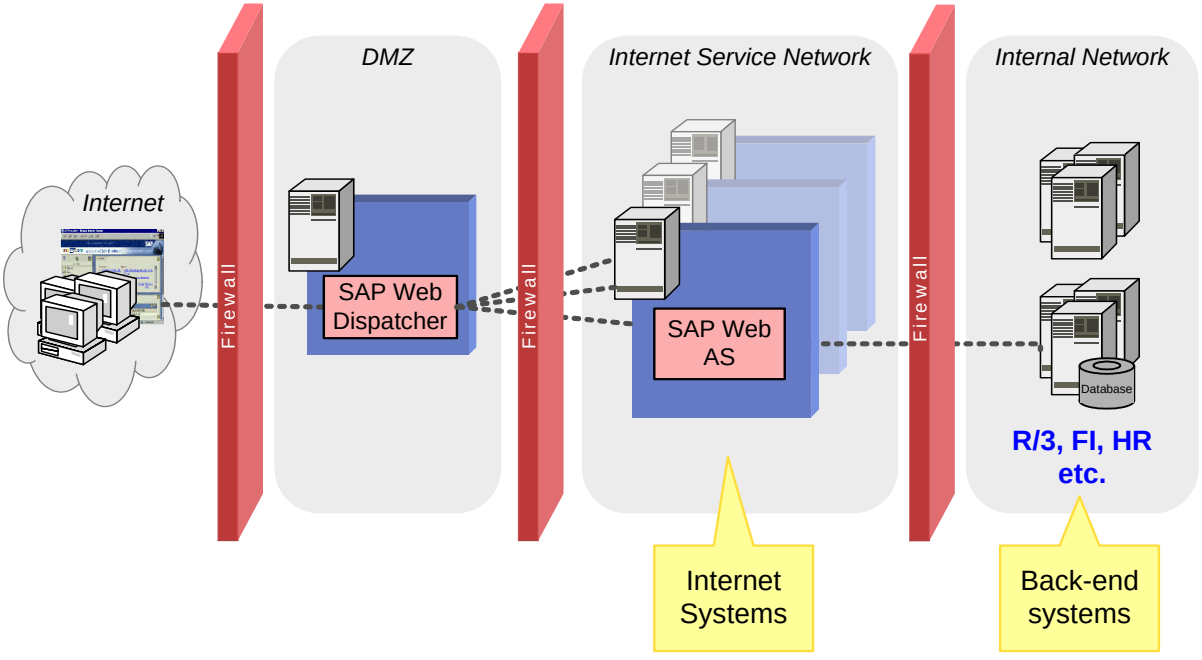
Note that security cannot be achieved by firewall technology alone. It requires management. We strongly recommend that you use **intrusion detection systems combined with efficient security management** to protect the firewall itself and the filtering mechanisms.

We do not recommend to position application servers in the DMZ and the database in the internal network, because a firewall between application server and database server may seriously degrade system performance. On the other side, the security benefit is small, because the application server needs full access to the database anyway.

Internal clients can usually have direct access to SAP Web AS in internal server network via corporate backbone network. However, note the possible problems with Server certificates as mentioned above when using end-to-end SSL.

Layered Security Zones

Additional protection for internal networks and highly sensitive system can be achieved using a layered firewall approach. This setup is illustrated by the following figure:



All systems accessible over the Internet are located in a second DMZ, the "**Internet Service Network**". Note that this may include central business systems, like the BW, if this system is accessed over the Internet.

The **applications that do not allow direct Web access** are located in a separate high security network. This typically includes sensitive applications, like FI or HR. A firewall controls all access to this network (and possibly access from the corporate intranet).

The SAP Web AS that is accessible from the Internet (and thus potentially "unsafe") can access internal systems through the firewall using RFC or HTTP, for example. Here, care has to be taken to **define exact rules** for accepted access to data or functions of the protected application and to **enforce these rules** using access control and proper authorizations.

As a further measure, the internal firewall may prohibit opening a connection from the external SAP Web AS to protected applications all together. Only systems in the high security network can open connection to SAP Web AS and send or retrieve data to or from it. This way, the protected applications and internal networks are completely **decoupled** from Internet access.

SAP generally recommends a setup with layered security zones as described here.

4 Web Infrastructure Design

You have to consider the following aspects:

- **Load balancing mechanism —** SAP generally recommends using some sort of load balancing device that provides a single entry into the system. See detailed discussion in the corresponding chapter above.
- **Security —** Depending on the features you require, you have to decide whether end-to-end SSL or SSL termination is the right choice for you. In addition, you have to decide on firewall architecture. See detailed discussion in the corresponding chapter above.
- **Cost of load balancing device —** Consider costs of hardware, software, maintenance etc.
- **Performance —** If you require maximum performance for a high volume Web site, combined with a heavy load of SSL transactions when terminating SSL in the load balancer, a dedicated hardware device is probably your best choice.
- **Robustness and high availability —** Especially for consumer interaction, availability is of great importance. Look for appropriate features in the device.
- **Ease of configuration and administration —** Some hardware load balancing devices require great skills to configure and operate. Be sure you either have access to an expert for the device, or pick a device that is easy to configure.
- **Integration into existing infrastructure —** make sure the load mechanisms fits in with your existing infrastructure as well as security policies.

Specific Web infrastructure solutions for SAP Web AS are discussed in another paper (currently under development). It will be available at http://service.sap.com/ti shortly.