## 13.6 Integration Reverse Proxy

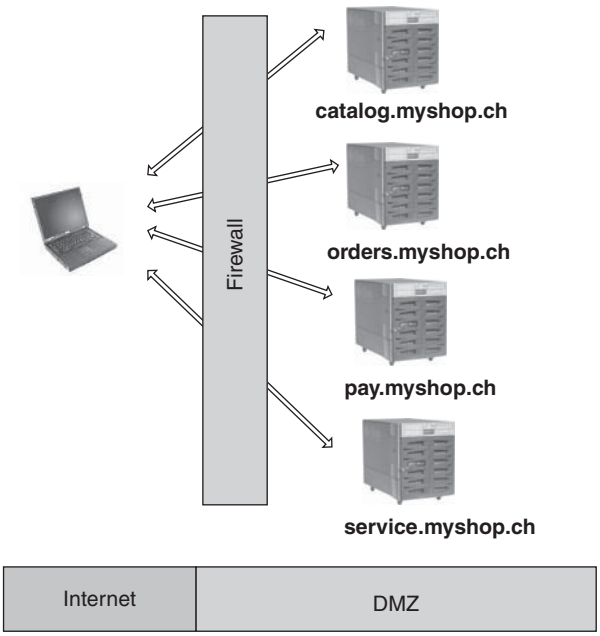
A Web site constructed from applications from different sources might require several different servers because of the heterogeneous operating requirement of the different applications. Because of the Internet addressing scheme, this distribution across several hosts is visible to the end user. Any change of the distribution or switch of parts of the site to a different host can invalidate URLs used so far, either cross-links to the Web site or bookmarks set up by users. An INTEGRATION REVERSE PROXY (465) alleviates this situation by providing a homogenous view of a collection of servers, without leaking the physical distribution of the individual machines to end users.

### *Example*

Consider a typical Web site of a company Myshop.com that sells goods and services. Their on-line presence was established with an interface to their support group, giving users access to static documentation such as a FAQ and a simple e-mail interface to contact support personnel. This Web server runs on a machine `support.myshop.com`. The marketing department the purchase an on-line catalog software vendor that displays their offerings on the server `catalog.myshop.com`.

Later on they implement a simple on-line ordering system with a small development company, because orders need to be routed to their home-grown ERP system automatically. Because they use a different platform for development for cost reasons, this order-taking system again needs to run on a separate server, `order.myshop.com`.

To avoid problems with late-paying customers and ease operation of their on-line business, they add credit card on-line payment software from yet another vendor. Again an additional machine is needed, `pay.myshop.com`. They end up with the structure shown in the diagram.




Multiple servers in a DMZ

The business flourishes and their original infrastructure hits some limits. However, their practice of having every server known on the Internet makes shifting applications to another server or running an application on two different systems hard. The complexity of the infrastructure and the cross-linking of the different application servers make every change a complex endeavor, with the risk of many broken links. How can the IT organization shield end users and servers from changes in the infrastructure? How can they extend functionality or processing power without breaking links or invalidating bookmarks of users?

## Context

A Web site consisting of several Web servers or Web applications.

## Problem

You want to implement your Web site using different servers, or use different vendors solutions for your Web site. How do you provide everything in a consistent Web application space without showing your server topology to users? How do you gain flexibility in network topology, for example by adding or removing servers without

surprising users? How do you provide fail-over switching or load balancing if an application server gets overloaded?

The solution to this problem must resolve the following forces:

- You cannot implement your complete Web site with a single server and platform because of complexity, performance, robustness, or reuse reasons.
- You want to hide network topology from your users, so that changes in machine configuration do not break their bookmarks or links to your Web site.
- In addition back-end cross-server links should continue to work regardless of network topology. This ensures that individual back-end applications continue to work unchanged, even when one back-end application is moved to some other machine.
- You want to be able to exchange parts of the Web site's implementation without breaking links.
- You want to add new elements and functionality to your Web site easily.
- You want to be able to switch a request for an application between hosts, either for fail-over or load balancing.
- You want only a single SSL certificate, because certificates are expensive, especially coordinating their renewal.

## Solution

Use a reverse proxy to integrate all your Web servers as back-end servers with a common host address (that of the reverse proxy).

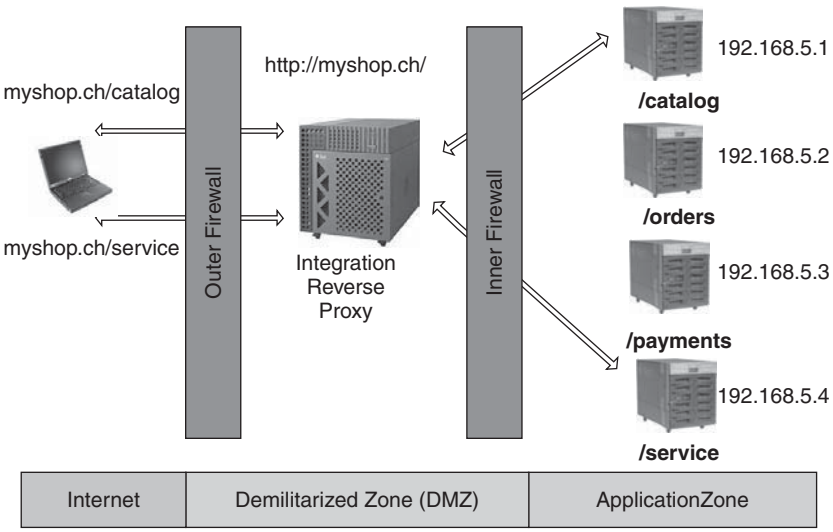
Map URL paths below the common host address to individual back-end server functions, so that any modification of the association of a function to a specific back-end host can easily be changed at the reverse proxy. Optionally provide your INTEGRATION REVERSE PROXY (465) with an SSL certificate for your Web site domain.

## Structure

You end up with the following structure when you place the INTEGRATION REVERSE PROXY (465) machine in the DEMILITARIZED ZONE (449). See figure on page 468.

## Dynamics

The dynamics of INTEGRATION REVERSE PROXY (465) are very similar to those of PROTECTION REVERSE PROXY (457) or PROXY-BASED FIREWALL (411). In addition to checking the permission, the INTEGRATION REVERSE PROXY (465) also calculates the network address and URL of the back-end server it should use as request target.




Protection using an INTEGRATION REVERSE PROXY

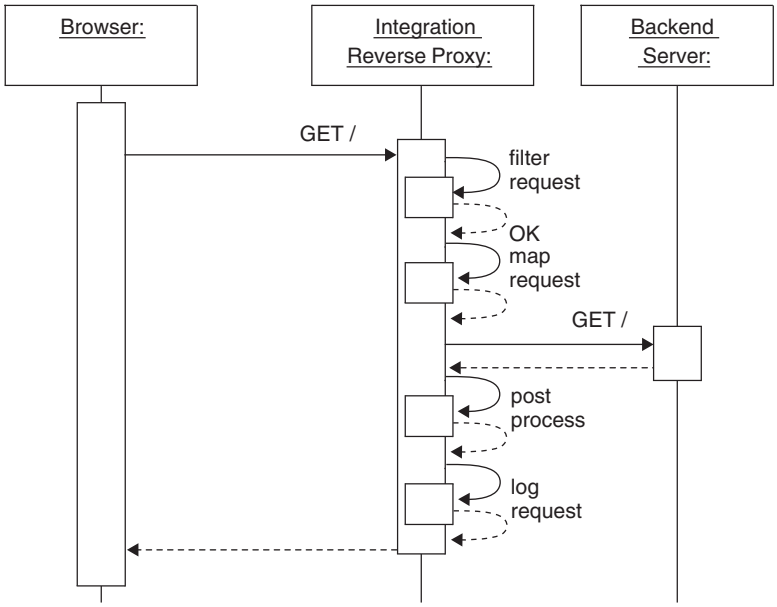
The parameters of this mapping are the original request URL and the application zone's network topology and addresses. Optionally you might chose to implement load balancing or a back-end failover strategy at the integration reverse proxy. See figure on page 469.

## Implementation

The implementation of an INTEGRATION REVERSE PROXY (465) follows most of the steps explained for PROTECTION REVERSE PROXY (457). Additional steps to be considered are:

1. *Design your Web site's name space*. This is a step that requires some planning, to allow for future extensibility. In our example, a path prefix maps to a specific server implementing the functionality. Several prefixes might also map to the same machine. Nevertheless, try to keep the mapping simple. There is one special case regarding the entry point '/': one back-end server can handle this, or the reverse proxy itself can show a navigation page to the user consisting of a menu of configured back-end services. This can change automatically with changes in configuration of the reverse proxy.

   An alternative to the path prefix mapping is to use virtual hosts for the reverse proxy, where a host name still designates a back-end service. This allows Myshop.com to continue to provide their original host names, even after they switched to a reverse proxy architecture.




Class diagram for INTEGRATION REVERSE PROXY

A combination of prefixes and virtual hosts allows a service provider to host similar functionality for several clients without the need to duplicate all infrastructure, and with easy extension of infrastructure if the need for it arises. Our example company can use this combination to provide a shop service (catalog, order, payment) for resellers under their reseller's domain address with Myshop.com's servers.

2. *Configure back-end Web servers*. In addition to the issues mentioned in PROTECTION REVERSE PROXY (457), you want links from one back-end service to another. For example, the `/catalog` back-end server will want to link to `/orders` and vice versa. Following your name space schema, adapt your Web pages and applications to create correct links without referring to the internal host addresses.
3. *Implement back-end server fail-over.* If your Web site must remain operational in the face of hardware or software failures, or when a new version of some back-end needs to be installed, you can provide a fail-over switch to a different back-end server machine implementing the same functionality. Such a switch can be automatic, when the reverse proxy cannot connect to the primary back-end server, or manually configured by operating personnel.
4. *Implement back-end server load balancing*. Similarly to fail-over, you can also implement some load balancing for back-ends if you need it. Several strategies

are possible. The simplest one is passing requests in a round robin fashion among several back-end servers that implement identical functionality. More sophisticated strategies can make use of statistics collected at the reverse proxy, such as response times of back-ends or special queries to the back-ends while collecting their respective loads.

Load balancing becomes more complicated in the case of Web applications on the back-ends that carry a user's session context, because a session's request need to be passed to the same back-end when more than one is available ('session stickiness'). See FRONT DOOR (473) for ideas on how to resolve these issues.

## Variants

*Integration Protection Reverse Proxy.* It is easy (and wise) to combine the INTEGRATION REVERSE PROXY (465) with the PROTECTION REVERSE PROXY (457) and gain the benefits of both.

You can also use INTEGRATION REVERSE PROXY (465) in an intranet integration scenario. Simple intranet applications (for example, those using PHP or Perl) can be deployed quickly behind the reverse proxy without the need to publicize the servers' address explicitly to all users. In addition, external Web applications can be similarly integrated into the workspace without users recognizing the external nature. Combined with a menu of available back-end services generated on the reverse proxy, deployment of tools can thus be instantaneous. This style of integration, relying only on HTTP, is much easier than using a fully-fledged portal server platform. In addition you can gain security, because the back-end servers can operate in a separated network zone that is not accessible directly from the potentially hostile corporate intranet (think about e-mail worms).

A combination of two INTEGRATION REVERSE PROXY (465), one facing the Internet and one for the intranet sharing of the back-end servers, is also possible. This reduces cost if the same functionality must be available on both networks.

## Known Uses

Pound (`http://www.apsis.ch/pound/index.html`) is an INTEGRATION REVERSE PROXY (465) that provides SSL wrapping and load balancing with a simple form of session stickiness.

## Consequences

The following benefits may be expected from applying this pattern:

- Only one externally-known host: only one name and one IP address for the reverse proxy need be known and accessible outside, except when virtual hosts

are used. You also increase security, because fewer machines need to operate in the DEMILITARIZED ZONE (449).

- The network topology of back-end servers is hidden. You can move back-end Web servers from one machine to another without invalidating external URLs or cross-application links.
- Ease of integration and extension. Mix and match of Web applications and technology becomes feasible for back-end Web servers and is transparent to end users.
- Bookmarks and cross-back-end links continue to work, even when a back-end is moved to another host.
- Load balancing of back-end servers by the reverse proxy is possible. However, if your back-end servers carry session, the reverse proxy must take stickiness into account. See FRONT DOOR (473) for more optional features.
- Centralized logging. The INTEGRATION REVERSE PROXY (465) provides a good hook on which to implement access and error logging. Ideally back-end servers no longer need to perform logging. A single log is easier to evaluate—for example, a user's navigation path can be followed easily even if more than one back-end server is used.
- You can save money and effort on SSL certificates and maybe also on IP addresses or host names, because only one host is connected to the Internet. Virtual host-to-service mapping is infeasible with a single SSL certificate. You then need to configure multiple IP addresses for the reverse proxy to make it possible to use valid SSL certificates, or you need to use an expensive 'wildcard' SSL certificate.

However, INTEGRATION REVERSE PROXY (465) also has its **liabilities**. It shares the last three liabilities with PROTECTION REVERSE PROXY (457): latency, some loss of transparency for back-ends, and introducing an additional point of failure.

- It is a potential single point of failure. If everything runs through your reverse proxy, this becomes a single point of failure. Additional redundancy is required for risk minimization. Without the reverse proxy, a single server outage can reduce available functionality, but might not bring everything down completely. Using a redundant hardware load-balancing switch and a redundant reverse proxy configuration can alleviate this problem.
- The number of concurrent connections is limited. IP imposes a hard limit on the number of usable ports and thus the number of concurrent connections that is possible. On really heavy loaded sites with relatively slow back-ends this might imply that you need additional means such as multiple reverse proxies with DNS round robin to stretch these limits.
- Complexity. There may be simpler means to gain one or the other benefit. For example, you can use a hardware load-balancing switch.

- Session stickiness with load balancing can be problematic when back-end servers rely on sessions. See FRONT DOOR (473) for more details and resolutions of this problem.
- Testing individual applications can be harder. You may need to set up a 'dummy' INTEGRATION REVERSE PROXY (465) to be able to test new applications. There can even be the need for a complete testing environment consisting of all back-ends to validate all possible cross-links.

## *See Also*

FRONT DOOR (473) is an even more sophisticated application of INTEGRATION REVERSE PROXY (465) using user authentication, authorization and session management.