How to...

Configure the Reverse Proxy Filter for SAP Enterprise Portal 6.0 SP2

ASAP “How to…” Paper

SAP Enterprise Portal 6.0
April 2004
Contents

1 INTRODUCTION .......................................................................................................................... 2
  1.1 Additional Documentation ................................................................................................. 2

2 HOW THE REVERSE PROXY FILTER WORKS ................................................................. 3
  2.1 Checks and Distinguishes between HTTP requests ............................................................. 3
      2.1.1 URL Types .................................................................................................................. 3
  2.2 Replaces Scheme, Host and Port Number ........................................................................... 4
      2.2.1 Origins of HTTP requests: External and Internal Clients ........................................... 4
  2.3 Using Multiple Domain Names to Access the Portal ......................................................... 4

3 SETTING UP AND CONFIGURING THE REVERSE PROXY FILTER .............................. 5
  3.1 Setting Up the Reverse Proxy Filter .................................................................................... 5

4 CREATING A REVERSE PROXY FILTER .............................................................................. 9
  4.1 Sample Java Code for a Reverse Proxy Filter ................................................................. 9
      4.1.1 RequestWrapper and ResponseWrapper Implementation .......................................... 10
      4.1.2 Configuring the Portal for Your Reverse Proxy Filter Solution ................................ 10
1 Introduction

This document describes the reverse proxy filter mechanism in SAP Enterprise Portal 6.0 SP2. The reverse proxy filter enables the portal to do the following:

- Modify HTTP responses to requests transmitted via a proxy server, so that content in the portal that is referenced by absolute URLs becomes available to clients. For more information on the types of URL in the portal, see “URL Types” on page 3.
- Work with multiple domain names via multiple proxy servers.

In addition, this document explains how to set up the reverse proxy filter in the portal, and describes how to create your own filter mechanism for any proxy server, for example one that is home-grown. Code samples are provided. For additional information on code samples, see “Creating a Reverse Proxy Filter” on page 9.

1.1 Additional Documentation

The information in this document relates only to the reverse proxy filter mechanism in SAP Enterprise Portal 6.0. It does not include discussions and recommendations for setting up a proxy server for the portal.

Information on proxy servers in the portal landscape is available in SAP Enterprise Portal Technical Infrastructure Guide, which is part of the documentation package EP6.0 SP2 Install Guides (Portal, KM, Unif., Coll., Oracle) at: service.sap.com/ep60 > Documentation & More > Installation.

Check SAP note 480520 for additional information on configuring an Apache Web server for the portal environment. You can access SAP notes at: service.sap.com/notes
2 How the Reverse Proxy Filter Works

The filter enables the portal to do the following:

- Check all HTTP requests coming to the portal platform.
- Distinguish between the following HTTP requests to the portal: those coming from external sources, such as the Internet and outside the firewall of an enterprise, and those coming from internal sources, such as the intranet within the firewall.
- Replace the scheme, host, and port number in an HTTP request.
- Respond to more than one domain name using different proxy servers.

2.1 Checks and Distinguishes between HTTP requests

URLs in the portal identify the location of available portal content and resources. Content can fail to display in clients depending on several factors including the following: the type of URL, and the origin of the URL.

2.1.1 URL Types

The portal platform contains two kinds of URL: relative and absolute URLs.

- Relative URLs
  A relative URL contains access information for an object in abbreviated form. This allows objects in a portal to reference each other without requiring the complete reference information. In addition, the group of objects can be moved without changing any references.
  
  The majority of URLs in the portal platform are relative URLs that start with the forward slash “/” character. For example, /irj/portalapps/com.sap.portal.themes.lafservice

- Absolute URL
  An absolute URL contains the fully qualified access information for an object. This is made up of the scheme, hostname, and port number, such as:
  <scheme>://<hostname>.<domain>.<extension>:<port>/<path>/<object>
  
  For example:

Objects referenced by absolute URLs fail to display in the client browser when a proxy server transmits the responses from the portal. For example, some graphics in the header area of the portal fail to display in clients. This failure is due to the fact that the object or resource is available in the physical server on which the portal runs, but not in the proxy server.

The filter allows the portal to alter the HTTP response so that the client can successfully display all objects via a proxy server.
2.2 Replaces Scheme, Host and Port Number

The filter mechanism modifies an HTTP response based on the source of the request.

2.2.1 Origins of HTTP requests: External and Internal Clients

The portal can identify the source of any HTTP request, either external or internal. External HTTP requests are queries coming from clients outside the DMZ, and internal HTTP requests are queries that come from the clients within the firewall.

All external HTTP requests go to the proxy server, and not directly to the portal. The proxy server obtains the responses for such requests and provides them to the client as if it were the source of the responses.

When it receives a request, the filter first determines the source of that request, external or internal. When the source is external, the filter provides a different implementation of the response. The regenerated implementation provides the scheme, host, and port of the reverse proxy server.

When the source is internal, the filter forwards the HTTP request without further processing. To enable the filter to distinguish between external and internal HTTP requests, do the following:

- Configure your proxy server to add HTTP header data to every request
- Configure the filter to obtain the HTTP header data in a request.

For additional information on how to configure the header data in the filter for all proxy servers, see “Setting Up the Reverse Proxy Filter” on page 5.

The HTTP header data consists of a header name, and a header value. For example, the header name can be “Via,” and a header value that identifies a specific proxy server.

You can implement different header data for different proxy servers in the portal landscape, and configure the filter to identify the different header data in all requests from the proxy servers.

2.3 Using Multiple Domain Names to Access the Portal

To access the portal through several different domain names, you must have several proxy servers, each with a different domain name. Then you configure and activate the filter mechanism to work with the proxy servers. For detailed information on how to set up and configure the filter for proxy servers, see “Setting Up the Reverse Proxy Filter” on page 5.

For each proxy server, repeat both the <filter> and <filter-mapping> declarations, and then modify the values for the repeated elements to make the portal aware of the proxy servers.

Make sure that the declared <filter> element for each proxy server does the following:

- Specifies the Java class file. The deployment descriptor has to use the same Java class file to activate the filter mechanism for each proxy server.
- Defines different “<filter-name>” values for use with each proxy server.
  For each proxy server, declare the same <filter-name> value in the <filter-mapping> element as that declared in the <filter> declaration.
- Defines different <filter-header-name> and its corresponding <param-value> values for use with each proxy server.

For each <filter> element declared in the deployment descriptor for a proxy server, define both the name of the specific proxy server machine and its domain name.
3 Setting Up and Configuring the Reverse Proxy Filter

You can activate the reverse proxy filter under the following conditions:

- You have deployed SAP Enterprise Portal 6.0 SP2 or higher in a secure network environment.
- You have successfully deployed and configured a proxy server that serves portal content to clients.

Note:

If your proxy server has been configured to work with a mechanism that speeds the authentication process, such as an SSL accelerator, then configure the mechanism to add an extra header that will overwrite the request scheme, and provide the correct scheme, namely HTTPS. For example, <clientProtocol> = HTTPS

If the authentication processing mechanism, such as an SSL accelerator, is not configured, then you must define the HTTPS value in the relevant XML code for the reverse proxy filter, so that the response scheme is always HTTPS.

The Java class, servlets and files that implement the filter mechanism are deployed in the portal landscape.

3.1 Setting Up the Reverse Proxy Filter

The reverse proxy filter is installed with the portal platform. You can configure the portal to use the filter by editing the file, web.xml, from the folder:

..<SAP system>/j2ee/j2ee_<instance_number>/cluster/server/services/servlet_jsp/work/jspTemp/irj/root/WEB-INF/

- In the web.xml file, add the following XML code before the last line: </web-app>

The values specified in the XML code below are examples.

Note:

If you intend to copy this example XML code into your Web.xml file, make sure you remove all comments. The portal will not run if the comments are in the text. Failing to remove comments returns an error.

<filter>
  <!-- Reverse Proxy Filter: If you intend to work with more than one proxy server, you must repeat the entire XML code below for each server, and then modify the repeated code to make the portal aware of each proxy server. -->
  <filter-name>
    <!-- Specifies the Java class file and filter name. Define different <filter-name> values for each server, when working with more than one proxy server. The different <filter-name> values can use the same Java class file. -->
    ReverseProxyFilter
  </filter-name>
  <filter-class>
    com.sapportals.portal.crosstopics.reverseproxyfilter.ReverseProxyFilter
  </filter-class>
  <load-on-startup>1</load-on-startup>
</filter>
<init-param>
   <!-- Limits the requests to a specified scheme, such as HTTP, HTTPS. If the <param-value> is empty, the filter can read the value from the request.
   The value specified in the <param-value> element is an example. -->
   <param-name> scheme </param-name>
</init-param>

<init-param>
   <!-- Specifies the name of the proxy server machine and its domain name. The value specified in the <param-value> element is an example. -->
   <param-name> proxy-host-name </param-name>
   <param-value>
      lego.tlvp.sap.corp
   </param-value>
</init-param>

<init-param>
   <!-- Port for HTTP requests: specifies the port number of the proxy Web server.
   The value specified in the <param-value> element is an example. -->
   <param-name> proxy-port-http </param-name>
   <param-value>
      80
   </param-value>
</init-param>

<init-param>
   <!-- Port for HTTPS requests: specifies the port number of the secured proxy Web server.
   The value specified in the <param-value> element is an example. -->
   <param-name> proxy-port-https </param-name>
   <param-value>
      443
   </param-value>
</init-param>
<init-param>
  <!-- The <param-value> of the <filter-header-name> is the HTTP header to search for in the request. Define different values for each <filter-header-name> and its corresponding <param-value>, when working with more than one proxy server. The value specified in the <param-value> element is an example. -->
  <param-name>filter-header-name</param-name>
  Via
  <param-value>
</init-param>

<init-param>
  <!-- The specified HTTP header value in the proxy server. Note that if not defined, the header in the request can be used to determine the external source. The value specified in the <param-value> element is an example. -->
  <param-name>filter-header-value</param-name>
  lego.tlvp.sap.corp
  <param-value>
</init-param>

<init-param>
  <!-- Specifies true or false. The default value is false. True enables the filter to print debugging messages to the server’s console log. False disables printing debugging messages. The value specified in the <param-value> element is an example. -->
  <param-name>debug</param-name>
  false
  <param-value>
</init-param>

<filter-mapping>
  <!-- The <filter-mapping> element defines the servlet for the filter. This bonds the filter to any activated PRT servlet. Make sure that you define the same <filter-name> value as specified in the <filter> element above. -->
  <filter-name>ReverseProxyFilter</filter-name>
  <servlet-name>prt</servlet-name>
</filter-mapping>
<filter-mapping>
  <!-- Captures the default servlet activator. In a multiple proxy server environment, make sure that the <filter-name> value is the same as the value defined in the <filter-name> tag at the beginning of the code. -->
  <filter-name>
    ReverseProxyFilter
  </filter-name>
  <url-pattern>
    /servlet/*
  </url-pattern>
</filter-mapping>

<filter-mapping>
  <!-- Captures all activated JSP files. In a multiple proxy server environment, make sure that the <filter-name> value is the same as the value defined in the <filter-name> tag at the beginning of the code. -->
  <filter-name>
    ReverseProxyFilter
  </filter-name>
  <url-pattern>
    *.jsp
  </url-pattern>
</filter-mapping>

<!-- Ends Reverse Proxy Filter -->
4 Creating a Reverse Proxy Filter

You can create a reverse proxy filter mechanism to use with a proxy server, for example a home-grown one. If you choose to create and implement your own reverse proxy filter solution for the portal environment, do the following:

1. Write a Java class for implementing the filter.
2. Implement your own request and response wrapper mechanisms.
3. Configure the settings for your implementation in the file, `web.xml`.
4. Create a JAR file that consists of your Java class, the request and response wrapper mechanisms, and then add the new JAR file to the folder:
   ```
   ...\cluster\server\services\servlet_jsp\work\jspTemp\irj\root\WEB-INF\lib
   ```

4.1 Sample Java Code for a Reverse Proxy Filter

Here is sample Java code for developing your own reverse proxy filter solution:

```java
import javax.servlet.*;
import javax.servlet.http.*;

public class ReverseProxySupport implements Filter
{
    public void init(FilterConfig arg0) throws ServletException {}
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException
    {
        boolean active = false;
        /* TODO: Enter the code that determines whether the filter should be activated
         * or not according to request */
        if (active)
        {
            ServletRequest sr = new RequestWrapper((HttpServletRequest)request);
            ServletResponse sResponse = new ResponseWrapper((HttpServletRequest)sr,
            (HttpServletResponse)response);

            chain.doFilter(sr, sResponse);
        }
        //Original request without change (filter not activated)
        else
        {
            chain.doFilter(request, response);
        }
    }
    public void destroy() {}
}
```

©2004 SAP AG
4.1.1 RequestWrapper and ResponseWrapper Implementation

Here is sample Java code for developing your own RequestWrapper and ResponseWrapper mechanisms:

```java
public class RequestWrapper extends HttpServletRequestWrapper {
    public String getServerName() {…}
    public int getServerPort() {…}
    public String getScheme() {…}
}
```

```java
public class ResponseWrapper extends HttpServletResponseWrapper {
    public void sendRedirect(String redirectURL) throws IOException {…}
}
```

4.1.2 Configuring the Portal for Your Reverse Proxy Filter Solution

Configure the portal by adding the XML code that declares and implements the filter to the file, `web.xml`. For more information of the XML declarations to use, see “Setting Up and Configuring the Reverse Proxy Filter” on page 5.