Sun Java System Web Server 6.1 SP6 Reverse Proxy Plug-in Release Notes

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–6510

060905@15490

Contents

Release Notes for Sun Java™ System Web Server 6.1 Reverse Proxy Plug-in

These release notes contain important information available at the time of release of Sun Java System Web Server 6.1 Reverse Proxy plug-in, including information about server application functions (SAFs), installation, configuration, technical notes, and pointers to additional resources. Review the release notes prior to installing and configuring your software, and then periodically thereafter for the most up-to-date information.

In addition to the 32–bit platform support, Sun Java System Web Server 6.1 SP6 also provides support to 64–bit Reverse Proxy.

The complete Sun Java System Web Server 6.1 documentation is available at `http://docs.sun.com/app/docs/coll/1308.2`.

These release notes contain the following sections:

Introduction

The Reverse Proxy plug-in is a NSAPI plug-in designed for use with the Sun Java System Web Server 6.1 SP3 and later Service Packs. This add-on allows the Web Server to act as a non-caching HTTP reverse proxy for specified URIs.

A reverse proxy is a proxy that appears to be a web server (origin server) to clients but in reality forwards the requests it receives to one or more origin servers. Because a reverse proxy presents itself as an origin server, clients do not need to be configured to use a reverse proxy. By configuring a given reverse proxy to forward requests to multiple similarly configured origin servers, a reverse proxy can operate as an application level software load balancer. In a typical deployment one or more reverse proxies will be deployed between the browsers and the origin servers.

The Reverse Proxy plug-in can be used in conjunction with the Sun Web Server features such as on-the-fly gzip compression, output filters, advanced Access Control Lists, and so on.

The Reverse Proxy plug-in includes support for the following features:

HTTP/1.0 and HTTP/1.1 Compliance

The Reverse Proxy plug-in issues HTTP/1.1 requests to origin servers and will accept HTTP/1.0 responses to requests. It will not upgrade incoming HTTP/1.0 requests to HTTP/1.1 in ways that are incompatible with HTTP/1.0 (for example, it will not add a Transfer-encoding: chunked header to a request).

Credential Pass-through

The Reverse Proxy plug-in passes through `basic-auth` and `digest-auth` credentials presented by the client. It encodes client certificates from the client and present them in proprietary headers that an appropriately coded application on the origin server can utilize.

Authentication to Origin Servers

The Reverse Proxy plug-in can be configured to present its own credentials to an origin server. The Reverse Proxy plug-in is capable of presenting `basic-auth` or utilizing a specified certificate nickname.

Data Encryption

The Reverse Proxy plug-in can utilize SSLv2, SSLv3 and TLS when making requests to origin servers.

Session Stickiness

The Reverse Proxy plug-in can be configured to recognize sticky cookies, and can have the name of the sticky cookies be configured.

Simple Load Balancing

The Reverse Proxy plug-in distributes load to several configured origin servers.

Granular Error Logging

The Reverse Proxy plug-in takes advantage of the Web Server's granular error logging capabilities such as config, failure, warning, fine, finer, and finest. See the Web Server documentation at for more detail.

Server Application Functions (SAFs)

The Reverse Proxy plug-in provides the following Server Application Functions (SAFs):

auth-passthrough

The `auth-passthrough` AuthTrans SAF inspects an incoming HTTP request for client information encoded by a service-passthrough function running on an intermediate server. The client information includes the following:

- The IP address from which the request originated, as encoded in the `Proxy-ip` header
- The SSL/TLS session ID of the originating connection, as encoded in the `Proxy-ssl-id` header
- The SSL/TLS cipher by the originating client, as encoded in the `Proxy-cipher`, `Proxy-keysize`, and `Proxy-secret-keysize` headers
- The SSL/TLS client certificate presented by the originating client, as encoded in the `Proxy-issuer-dn`, `Proxy-user-dn`, and `Proxy-auth-cert` headers.

When auth-passthrough detects encoded client information, it instructs the server to treat the request as if it had arrived directly from the originating client instead of through an intermediate server running service-passthrough.

The auth-passthrough SAF is optional. When used, auth-passthrough is used on the server instance that receives the request forwarded by service-passthrough.

Because auth-passthrough makes it possible to override information that may be used for authentication (for example, the IP address of the original request), it is important that only trusted clients and servers be allowed to connect to a server running auth-passthrough. As a minimal precaution, only servers behind a corporate firewall should run auth-passthrough; no internet-accessible server should run auth-passthrough. Further, if information about the originating client is not required, auth-passthrough should not be used.

The following obj.conf code demonstrates the use of auth-passthrough (note that these lines are not indented in a real obj.conf):

```
<Object name="default">
AuthTrans fn="auth-passthrough"
...
</Object>
```

check-passthrough

The `check-passthrough` ObjectType SAF checks to see if the requested resource (for example, the HTML document or GIF image) is available on the local server. If the requested resource does not exist locally, check-passthrough sets the type to indicate that the request should be passed to another server for processing by service-passthrough.

The `check-passthrough` SAF accepts the following parameters:

- **type** - (Optional) The type to use for files that do not exist locally. If not specified, type defaults to magnus-internal/passthrough.

service-passthrough

The `service-passthrough` Service SAF forwards a request to another server for processing.

The `service-passthrough` SAF accepts the following parameters:

- **servers** - A quoted, space-delimited list of the servers that receive the forwarded requests. Individual server names may optionally be prefixed with http:// or https:// to indicate the protocol and suffixed with a colon and integer to indicate the port.
- **sticky-cookie** - (Optional) The name of a cookie that will cause requests from a given client to stick to a particular server. Once a request containing a cookie with this name is forwarded to a given server, service-passthrough attempts to forward subsequent requests from that client to the same server by sending a JROUTE header back to the client. If not specified, `sticky-cookie` defaults to JSESSIONID.
- **user** - (Optional) The username that `service-passthrough` uses to authenticate to the remote server through Basic-Auth.

  Note – ‘user&rsquo’ requires that ‘password’ also be specified.

- **password** - (Optional) The password that service-passthrough uses to authenticate to the remote server via Basic-Auth.

Note – 'password' requires that 'user' also be specified.

- **client-cert-nickname** - (Optional) Nickname of the client certificate that `service-passthrough` uses to authenticate to the remote server.
- **validate-server-cert** - (Optional) Indicates whether `service-passthrough` should validate the certificate presented by the remote server. If not specified, `validate-server-cert` defaults to true.
- **rewrite-host** - (Optional) Indicates whether `service-passthrough` should rewrite the Host header sent to remote servers, replacing the local server's hostname with the remote server's hostname. If not specified, `rewrite-host` defaults to false.
- **rewrite-location** - (Optional) Indicates whether `service-passthrough` should rewrite the Location headers returned by a remote server, replacing the remote server's scheme and hostname with the local server's scheme and hostname. If not specified, `rewrite-location` defaults to true.
- **ip-header** - (Optional) Name of the header that contains the client's IP address, or "" if the IP address should not be forwarded. If not specified, `ip-header` defaults to Proxy-ip.
- **cipher-header** - (Optional) Name of the header that contains the symmetric cipher used to communicate with the client (when SSL/TLS is used), or "" if the symmetric cipher name should not be forwarded. If not specified, `cipher-header` defaults to `Proxy-cipher`.
- **keysize-header** - (Optional) Name of the header that contains the symmetric key size used to communicate with the client (when SSL/TLS is used), or "" if the symmetric key size name should not be forwarded. If not specified, `keysize-header` defaults to `Proxy-keysize`.
- **secret-keysize-header** - (Optional) Name of the header that contains the effective symmetric key size used to communicate with the client (when SSL/TLS is used), or "" if the effective symmetric key size name should not be forwarded. If not specified, `secret-keysize-header` defaults to `Proxy-secret-keysize`.
- **ssl-id-header** - (Optional) Name of the header that contains the client's SSL/TLS session ID (when SSL/TLS is used), or "" if the SSL/TLS session ID should not be forwarded. If not specified, `ssl-id-header` defaults to `Proxy-ssl-id`.
- **issuer-dn-header** - (Optional) Name of the header that contains the client certificate issuer DN (when SSL/TLS is used), or "" if the client certificate issuer DN should not be forwarded. If not specified, `issuer-dn-header` defaults to `Proxy-issuer-dn`.
- **user-dn-header** - (Optional) Name of the header that contains the client certificate user DN (when SSL/TLS is used), or "" if the client certificate user DN should not be forwarded. If not specified, `user-dn-header` defaults to `Proxy-user-dn`.
- **auth-cert-header** - (Optional) Name of the header that contains the DER-encoded client certificate in Base64 encoding (when SSL/TLS is used), or "" if the client certificate should not be forwarded. If not specified, `auth-cert-header` defaults to `Proxy-auth-cert`.

When multiple remote servers are configured, `service-passthrough` chooses a single remote server from the list on a request-by-request basis. If a remote server cannot be contacted or returns an invalid response, `service-passthrough` sets the status code to 502 Bad Gateway and returns

REQ_ABORTED. This will return an error to the browser. This error can be customized in the Web Server by configuring a customized response for the 502 error code.

When user and password are specified, `service-passthrough` will use these credentials to authenticate to the remote server using HTTP basic authentication. When one or more of the servers in the servers parameter are configured with a https:// prefix, `client-cert-nickname` specifies the nickname of the client certificate `service-passthrough` will use to authenticate to the remote server.

Note – `service-passthrough` generally uses HTTP/1.1 and persistent connections for outbound requests with the following exceptions:

- When forwarding a request with a Range header that arrived through HTTP/1.0, `service-passthrough` issues an HTTP/1.0 request. This is done because the experimental Range semantics expected by Netscape HTTP/1.0 clients differ from the Range semantics defined by the HTTP/1.1 specification.
- When forwarding a request with a request body (For example POST request), `service-passthrough` will not reuse an existing persistent connection. This is done because the remote server is free to close a persistent connection at any time, and `service-passthrough` will not retry requests with a request body.

In addition, `service-passthrough` encodes information about the originating client in the headers named by the `ip-header`, `cipher-header`, `keysize-header`, `secret-keysize-header`, `ssl-id-header`, `issuer-dn-header`, `user-dn-header`, and `auth-cert-header`parameters (removing any client-supplied headers with the same name) before forwarding the request. Applications running on the remote server may examine these headers to extract information about the originating client.

Installing the Reverse Proxy plug-in

The Reverse Proxy plug-in is available for use with the Sun Java System Web Server 6.1 SP3 or later Service Packs.

This section includes the following topics:

Package Contents

The contents of the platform specific packages are:

Solaris, Linux, AIX:

- README.txt
- libpassthrough.so (the NSAPI shared object, or "plug-in")

HP-UX:

- README.txt
- libpassthrough.sl (the NSAPI shared object, or "plug-in")

Windows:

- README.txt
- passthrough.dll (the NSAPI shared object, or "plug-in")

Installing on Solaris, Linux, HP-UX and AIX Using tar Packaging

```
$ gzip -d sun-webserver61-passthrough-{sol|lin|hpux|aix}.tar.gz

;; Uncompress the tar archive,
;; where {sol|lin|hpux|aix} reflects
;; the operating system
;; environment the library will
;; be used

$ tar xvf sun-webserver61-passthrough-{sol|lin|hpux|aix}.tar

;; Extract the tar archive
```

Installing on Windows

```
$ unzip sun-webserver61-passthrough-win.zip

;; Uncompress the ZIP archive
```

Installing on Solaris Using SVr4 Packaging

```
$ su

;; root access is required to install
;; SVr4 packages
```

```
$ cd <path/to/package>

;; Change directory to where the package
;; is located

# pkgadd -d .

;; Install SUNWwbsvr-passthrough.pkg package
```

Note – This installation places the shared object and README in: `/opt/SUNWwbsvr/plug-ins/passthrough`

Installing on Linux using RPM Packaging

```
$ su

;; root access is required to
;; install RPM packages

$ cd <path/to/package>

;; Change directory to where
;; the package is located

# rpm -iUvh sun-webserver-passthrough.rpm

;; Install
;; sun-webserver-passthrough.rpm
;; package
```

Note – This installation places the shared object and README in `/opt/sun/webserver/plug-ins/passthrough`

Configuring the Reverse Proxy plug-in

The Reverse Proxy plug-in should be initialized in the Sun Java System Web Server magnus.conf file and configured in the corresponding obj.conf file.

This section includes the following topics:

- "Example 2" on page 14

magnus.conf

</path/to/sharedobject> is the path where the shared object is installed, including the shared object itself.

Note that the path elements are "/" regardless of the operating system.

```
Init fn="load-modules" shlib="</path/to/sharedobject>
```

obj.conf

Configuration of the `obj.conf` will vary depending on the intended use. See the Sun Java System Web Server documentation for use and syntax of the `obj.conf`.

Example 1

This configuration will proxy the URI "/example" if it does not exist locally. A local copy of "/example" is preferred to a remote copy:

```
<Object name="default">
# Assign the URI "/example" (and any more specific URIs;
# /example/foo.html, /example/qwe.jsp, etc) the object name
# "server.example.com"
NameTrans fn="assign-name"

          from="/example(|/*)"
          name="server.example.com"

...
</Object>

# Execute these instructions for any resource with the assigned name
# "server.example.com"
<Object name="server.example.com">


# Check to see if a local copy of the requested resource exists. Only
# proxy the request if there is not a local copy.
ObjectType fn="check-passthrough"

           type="magnus-internal/passthrough"
```

```
# Proxy the requested resource to the URL
# "http://server.example.com:8080" only if the "type" has been set to
# "magnus-internal-passthrough"
Service type="magnus-internal/passthrough"

        fn="service-passthrough"
        servers="http://server.example.com:8080"

</Object>
```

Example 2

This configuration will proxy all requests for the URI "/app" without first checking for a local version. The Reverse Proxy plug-in provides its own credentials through Basic-Auth to the origin server.

```
<Object name="default">
# Assign the URI "/app" (and any more specific URIs;
# /app/foo.html, /app/qwe.jsp, etc) the object name
# "server.example.com"
NameTrans fn="assign-name"

          from="/app(|/*)"
          name="server.example.com"

...
</Object>

# Execute these instructions for any resource with the assigned name
# "server.example.com"
<Object name="server.example.com">

# Proxy the requested resource to the URL
# "http://server.example.com:8080"
Service fn="service-passthrough"

        servers="http://server.example.com:8080"
        user="blues"
        password="j4ke&elwOOd
"

</Object>
```

How to Report Problems and Provide Feedback

If you have problems with Sun Java System Web Server 6.1 Reverse Proxy plug-in, contact Sun customer support using one of the following mechanisms:

- Sun Software Support services online at

  `http://www.sun.com/service/support/software/`
- The telephone dispatch number associated with your maintenance contract

So that we can best assist you in resolving problems, please have the following information available when you contact support:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. Send your comments to Sun using the "Send comments" link at: `http://docs.sun.com/`

Please include identifying information with your comments, such as the book's part number and title.

Additional Sun Resources

Useful Sun Java System information can be found at the following locations:

- Documentation for Sun Java System Web Server 6.1 and Service Packs

  `http://docs.sun.com/app/docs/coll/1308.2`
- Sun Java System Software Products and Service

  `http://www.sun.com/software`
- Sun Java System Developer Information

  `http://developers.sun.com/`
- Sun Developer Support Services

  `http://developers.sun.com/services/index.jsp`
- Sun Java System Software Support Services

  `http://www.sun.com/service/support/software/`

- Sun Support and Training Services

  http://www.sun.com/training/
- Sun Java System Consulting and Professional Services

  http://www.sun.com/service/sunjavasystem/sjsservicessuite.html