Use a Reverse Proxy

On this page:

- Reverse Proxy Architectures
- General Guidelines
- Using Nginx as a Simple HTTP Reverse Proxy
- Using Apache as a Reverse Proxy
- Configure SSL Termination at the Reverse Proxy
- Using SSL from the Reverse Proxy to the Controller

This topic describes how to use set up the Controller behind a reverse proxy.

Reverse Proxy Architectures

The AppDynamics Controller is often deployed behind a reverse proxy. The proxy presents a virtual IP address to external connections, such as agents and browser clients. The proxy often resides in the DMZ for the network, where it terminates SSL connections from external clients.

The proxy provides a security layer for the Controller, but it also lets you move a Controller to another machine or switch between high availability pairs without having to reconfigure agents.

The following diagram illustrates the scenario:
As shown, the reverse proxy listens for incoming requests on a given path, /controller in this case, on port 80. It forwards matching requests to the HTTP listening port of the primary Controller at apphost1:8090.

In terms of network impact in this scenario, switching active Controllers from the primary to the secondary in this scenario only requires the administrator to update the routing policy at the proxy so that traffic directed to the secondary instead of the primary.

These instructions describe how to set up the Controller with a reverse proxy. They also provide sample configurations for a few specific types of proxies, including NGINX and Apache Web Server.

This information is intended for illustration purposes only. The configuration requirements for your own deployment are likely to vary depending on the nature of your existing environment, the applications being monitored, and the policies of your organization.

While AppDynamics supports Controllers that are deployed with a reverse proxy, AppDynamics Support cannot guarantee help with specific setup questions and issues particular for your environment or the type of proxy you are using. For this type of information, please consult the documentation provided with your proxy technology. Alternatively, ask the AppDynamics community.

**General Guidelines**

The following describes general requirements, considerations, and features for deploying the AppDynamics Controller and App Agents with a reverse proxy.

- Set the deep link JVM option, `-Dappdynamics.controller.ui.deeplink.url`, to the value of the Controller URL. Use either the hostname or IP address of the Controller host (if directly accessible to clients) or to the VIP for the Controller as exposed at the proxy in the following format:

  ```bash
  modifyJvmOptions add Dappdynamics.controller.ui.deeplink.url=http[s]://controller.corp.example.com[:port]/controller
  ```

  Use the URI scheme (http or https), hostname and port number appropriate for your Controller. The Controller uses the deep link value to compose URLs it exposes in the UI.

- If terminating SSL at the proxy, also set the following JVM options:
You should use the Enterprise Console's Controller Settings page to edit the JVM options to retain your settings. See Update Platform Configurations for more information.

- If the proxy sits between monitored tiers in the application, make sure that the proxy passes through the custom header that AppDynamics adds for traffic correlation, singularityheader. Most proxies pass through custom headers by default.
- For App Agents, the Controller Host and Controller Port connection settings should point to the VIP or hostname and port exposed for the Controller at the reverse proxy. For details see Agent-to-Controller Connections.
- If using SSL from the agent to the proxy, ensure that the security protocols used between the App Agent and proxy are compatible. See the compatibility table for the SSL protocol used by each version of the agent.
- If the proxy (or another network device) needs to check for the availability of the Controller, it can use Controller REST resource at: http://<host>:<port>/controller/rest/serverstatus. If the Controller is active and if in high availability mode, is the primary, it returns an XML response similar to this one:

  ```xml
  <serverstatus vendorid="" version="1">
  <available>true</available>
  <serverid/>
  <serverinfo>
    <vendorname>AppDynamics</vendorname>
    <productname>AppDynamics Application Performance Management</productname>
    <serverversion>003-008-000-000</serverversion>
  </serverinfo>
  </serverstatus>
  ```

  If the Controller is in standby mode, this resource returns a 503 response.

The following sections provide notes and sample configurations for a few specific types of proxies, including Nginx and Apache Web Server.

### Using Nginx as a Simple HTTP Reverse Proxy

Nginx is a commonly used web server and reverse proxy available at http://nginx.org/.

To use Nginx as a reverse proxy for the Controller, simply include the Controller as the upstream server in the Nginx configuration. If deploying two Controllers in a high availability pair arrangement, include the addresses of both the primary and secondary Controllers in the upstream server definition.

The following steps walk you through the set up at a high level. It assumes you have already installed the Controller and have an Nginx instance, and you only need to modify the existing configuration to have Nginx route traffic to the Controller.

**To route Controller traffic through an Nginx reverse proxy**

1. Add a JVM option named -Dappdynamics.controller.ui.deeplink.url. Set its value to the URL for the Controller, as described in the guidelines above.
2. Shut down the Controller.
3. If terminating SSL at the proxy, also set the -Dappdynamics.controller.services.hostName and -Dappdynamics.controller.services.port JVM options to the external DNS hostname for the Controller and the external port number, typically 443.
4. In the Nginx home directory on the reverse proxy machine, open the conf/nginx.conf file for editing.
5. In the configuration file, add a cluster definition the specifies each Controller as an upstream server. For example:
upstream appdcontroller {
    server 127.0.15.11:8090 fail_timeout=0;
}

server {
    listen 80;
    server_name appdcontroller.example.com;
    expires 0;
    add_header Cache-Control private;
    location / {
        proxy_set_header    Host $host;
        proxy_set_header    X-Real-IP $remote_addr;
        proxy_set_header    X-Forwarded-Proto https;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_pass          http://appdcontroller;
    }
}

In the sample, the Controller resides on 127.0.15.11 and has the fully qualified domain name appdcontroller.example.com.

6. Restart the Nginx server to have the change take effect.
7. Restart the Controller.

After the Controller starts, it should be able to receive traffic through Nginx. As an initial test of the connection, try opening the Controller UI via the proxy, that is, in a browser, go to http://<virtualip>:80/controller. For the App Agents, you'll need to configure their proxy host and port settings as described in the general guidelines above.

Using Apache as a Reverse Proxy

To use Apache as a reverse proxy, you need to make sure the appropriate Apache module is installed and enabled in your Apache instance. For HTTP proxying, this is typically mod_proxy_http. The mod_proxy_http module supports proxied connections that use HTTP or HTTPS.

To configure Apache with mod_proxy_http

1. Add a JVM option named -Dappdynamics.controller.ui.deeplink.url. Set its value to the URL for the Controller, as described in the guidelines above.
2. Shut down the Controller.
3. If terminating SSL at the proxy, also set the -Dappdynamics.controller.services.hostname and -Dappdynamics.controller.services.port JVM options to the external DNS hostname for the Controller and the external port number, typically 443.
4. On the machine that runs Apache, check whether the required modules are already loaded by your Apache instance by running this command:

   ```bash
   apache2ctl -M
   ```

   In the output, look for proxy modules as follows:

   ```
   proxy_module (shared)
   proxy_http_module (shared)
   ```

   The proxy_module is a dependency for proxy_module_http.
5. If they are not loaded, enable the Apache module as appropriate for your distribution of Apache. For example, on Debian/Ubuntu:

   a. Type the following:
b. Restart Apache:

```
sudo service apache2 restart
```

6. Add the proxy configuration to Apache. For example, a configuration that directs clients requests to the standard web port 80 at the proxy host to the Controller could look similar to this:

```
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>
ProxyRequests Off
ProxyPreserveHost On
ProxyPass /controller http://controller.example.com:8090/controller
ProxyPassReverse /controller  http://controller.example.com:8090/controller
```

7. Apply your configuration changes by reloading Apache modules. For example, enter:

```
sudo service apache2 reload
```

8. Start the Controller.

After the Controller starts, test the connection by opening a browser to the Controller UI as exposed by the proxy. To enable AppDynamics App Agents to connect through the proxy, be sure to set the proxy host and port settings in the proxy, as described in the general guidelines above. Also, be sure to apply any of the other general guidelines described in the general guidelines above.

### Configure SSL Termination at the Reverse Proxy

This section describes how to set up security when the client-side connection to the proxy uses SSL that's terminated at the proxy. This assumes that the proxy and Controller are in a secured data center and the App Agents or UI browser client connections are from a potentially insecure network.

Terminating SSL at a proxy offloads the burden of SSL processing from the Controller to the proxy. This configuration is strongly recommended when deploying the Controller to large scale, high workload environments. Terminating SSL at a proxy also provides the benefit of having a central point in the data center for the security certificate and for key management.

This section provides a sample configuration for Nginx, but the concepts translate to other types of reverse proxies as well.
Configure the Proxy for SSL Termination

To perform SSL termination at the reverse proxy, you need to:

- Ensure that the App Agents can establish a secure connection with the proxy. See Agent and Controller Compatibility for SSL settings for various versions of the agent. Ensure that the proxy includes a server certificate signed by an authority that is trusted by the agent. Otherwise, you will need to install the proxy machine's server key.
- If using .NET App Agents in your environment, verify that the reverse proxy server uses a server certificate signed by a certificate authority (CA). The .NET App Agent does not permit SSL connections based on a self-signed server certificate.
- Configure the proxy to forward traffic between it and the Controller to a secure port between it and the client.
- The client App Agents and browser clients under this configuration must use the secure port to communicate with the Controller (i.e., the proxy). Configuring a mixed channel on the Controller as described here causes the agents to perform as if they were using a secure port. Therefore, you need to ensure that they use a secure port only.

A complete example configuration with Nginx performing SSL termination for the Controller would look something like this:
upstream appdcontroller {
    server 127.0.15.11:8191 fail_timeout=0;
}

server {
    listen 80;
    server_name appdcontroller.example.com;
    return 301 https://$host$request_uri;
}

server {
    listen 443;
    server_name appdcontroller.example.com;
    
    ssl on;
    ssl_certificate /etc/nginx/server.crt;
    ssl_certificate_key /etc/nginx/server.key;

    ssl_session_timeout 5m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;

    expires 0;
    add_header Cache-Control private;

    location / {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_redirect http:// https://;
        proxy_pass http://appdcontroller;
    }
}

This example builds on the configuration shown in the simple passthrough example. In this one, any request received on the non-SSL port 80 is routed to port 443. The server for port 443 contains the settings for SSL termination. The ssl_certificate_key and ssl_certificate directives should identify the location of the server security certificate and key for the proxy.

The configuration also indicates the SSL protocols and ciphers accepted for connections. The security settings need to be compatible with the AppDynamics App Agent security capabilities, as described on the Agent and Controller Compatibility page.

Cookie Security

The Controller sets the secure flag on the X-CSRF-TOKEN cookie sent over HTTPS. The secure flag ensures that clients only transmit the cookies on secure connections.

If you are terminating HTTPS at a reverse proxy in front of the Controller, the Controller will not set cookie security by default, since connections to the Controller would occur over HTTP in this case.

To ensure cookie security, you should configure the reverse proxy to rewrite the set cookie statement so that the secure statement is included. The method for adding the secure flag varies depending on the type of the reverse proxy. The following illustrates a regular expression string replacement rule that could be used to accomplish this on HAProxy, as one example:

TLSv1 should only be enabled if absolutely necessary.

If you are terminating HTTPS at a reverse proxy in front of the Controller, the Controller will not set cookie security by default, since connections to the Controller would occur over HTTP in this case.

To ensure cookie security, you should configure the reverse proxy to rewrite the set cookie statement so that the secure statement is included. The method for adding the secure flag varies depending on the type of the reverse proxy. The following illustrates a regular expression string replacement rule that could be used to accomplish this on HAProxy, as one example:
Using SSL from the Reverse Proxy to the Controller

Have the proxy connect to the Controller with SSL requires a minor modification to the proxy configuration. Simply specify the use of HTTPS as the protocol to connect to the backend or upstream server. In other words, for the Nginx configuration, this simply requires you to modify the `proxy_pass` value as follows:

```
proxy_pass          https://appdcontroller;
```

To complete the configuration, make sure you have configured SSL on the Controller as described in Controller SSL and Certificates.