Reverse Proxy Server

This section is for information purposes only.

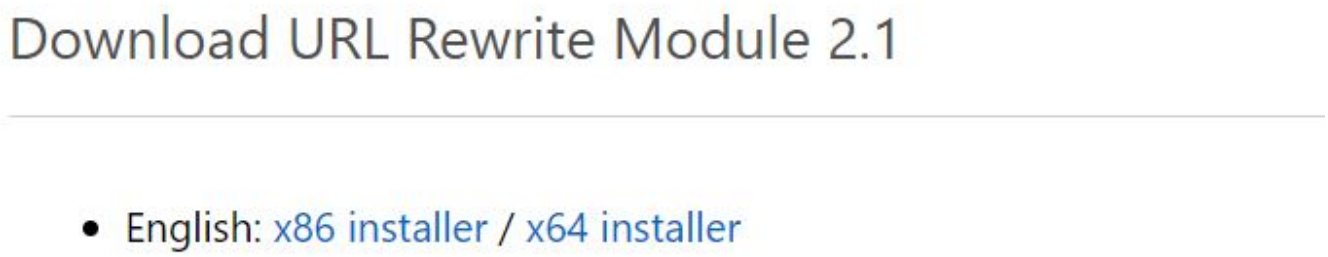
For Streaming Viewer, the Client-side 3D HOOPS Viewer will need to communicate with the Server-side Services using networking ports that are not typically open in customer firewalls. Reverse Proxies allow network traffic to use standard HTTP connections (80, 443) for such information flow.

The following steps outlines the process of installing and configuring the Reverse Proxy for Streaming 3D Viewer:

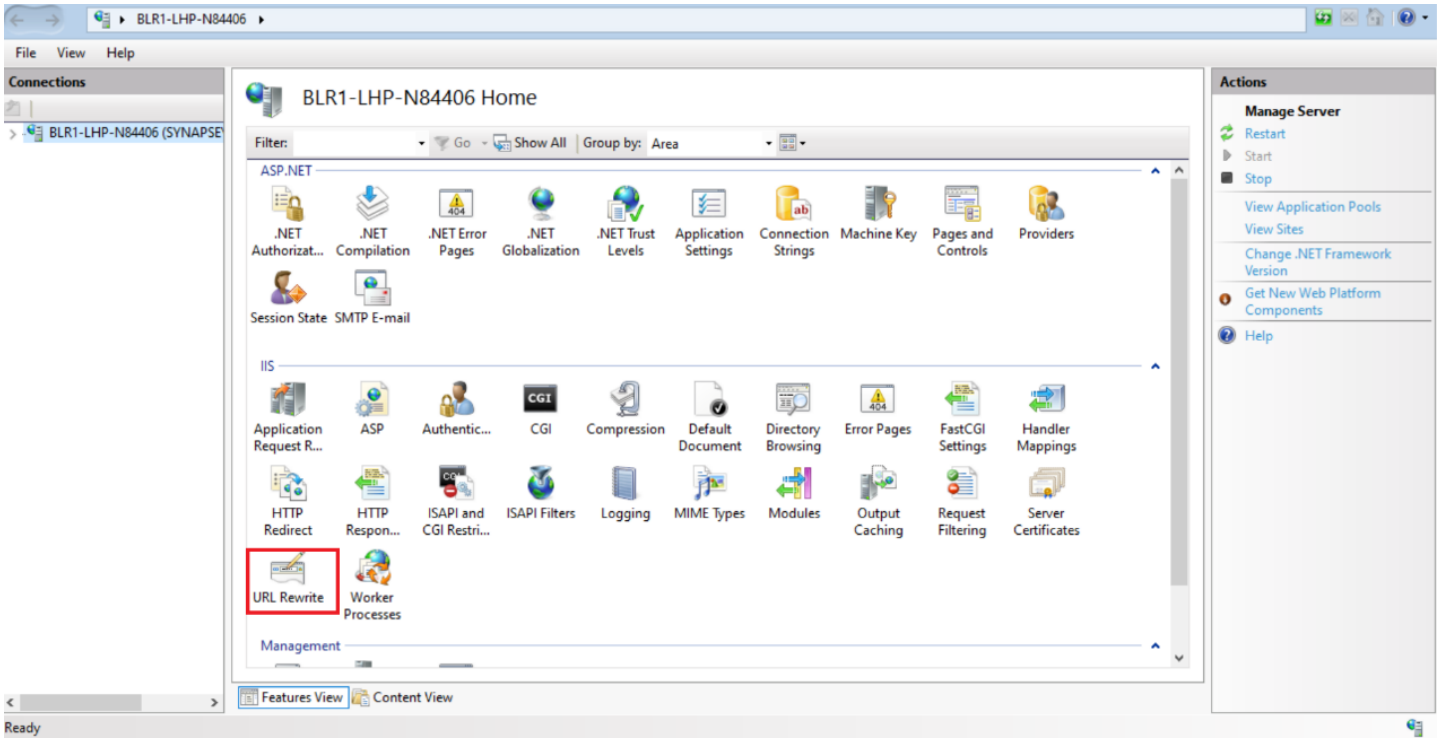
1. Install Application **Request Routing** (ARR) 3.0 from the following link: https://www.iis.net/downloads/microsoft/application-request-routing .

2. Install **URL** Rewrite **2.1** from the following link:

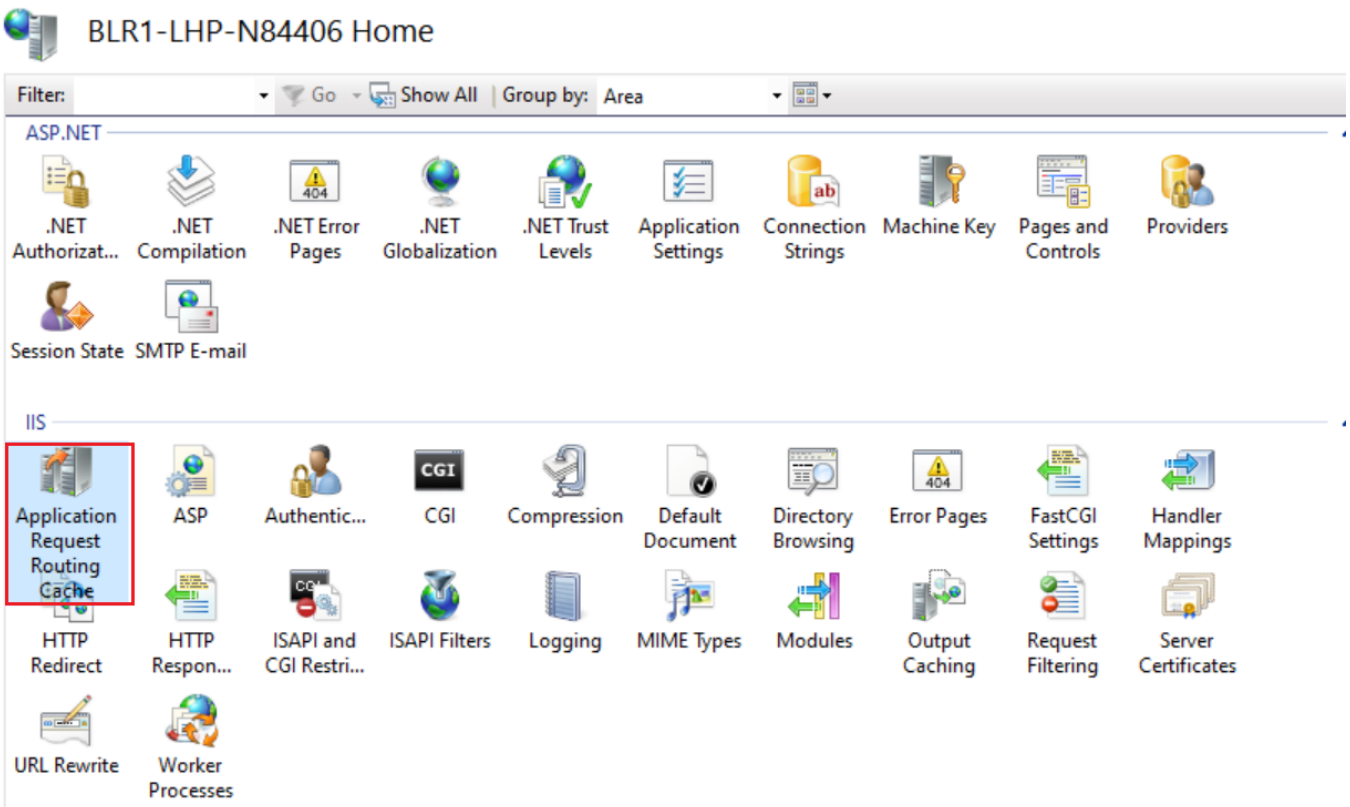
https://www.iis.net/downloads/microsoft/url-rewrite



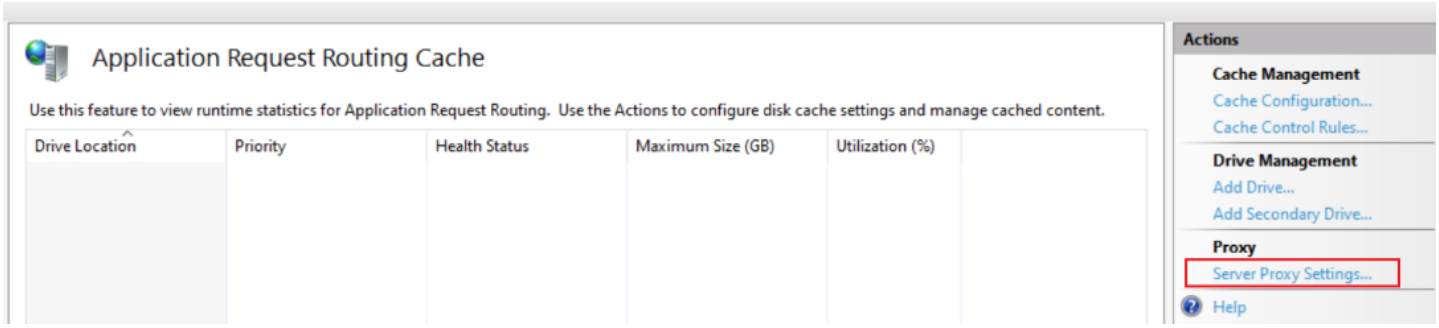
3. After installation is complete, open **IIS** from **Windows** browser. A new module appears in IIS Manager.



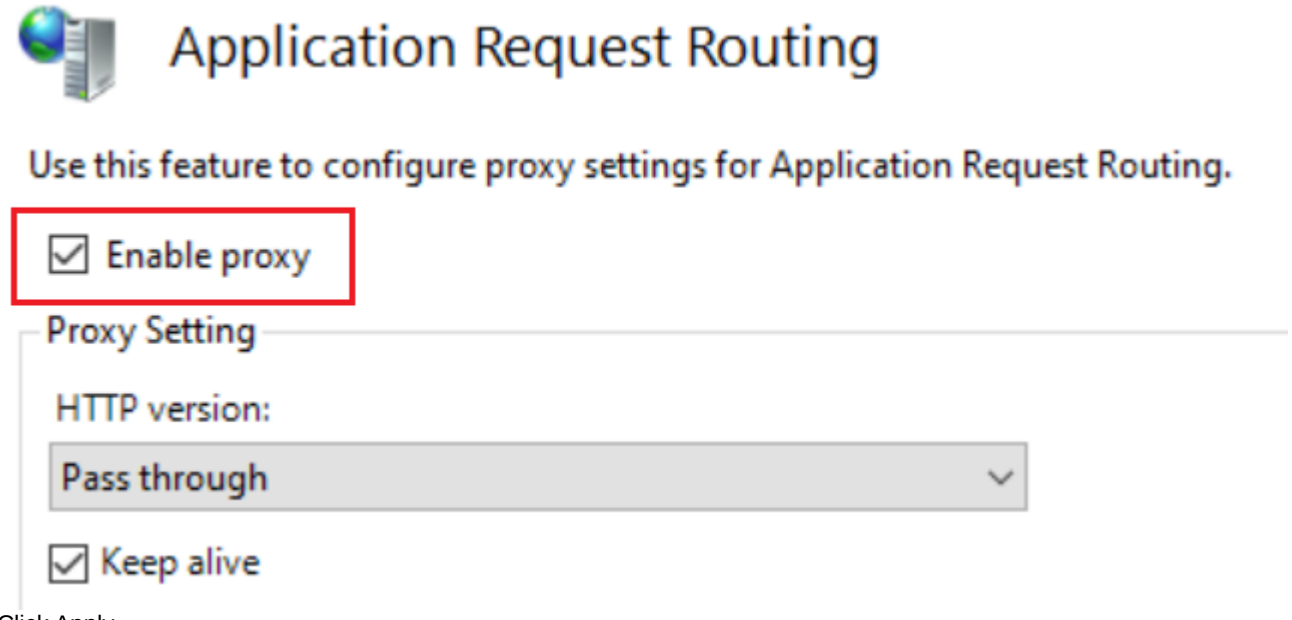
4. On IIS, click Application Request Routing Cache.



5. From the Actions column, click Server **Proxy Settings**.



6. Select **Enable Proxy** checkbox.



7. Click Apply.

Next step is to Configure URL Rewrites rules. This can be done in two ways:

- Option 1: Configure URL Rewrite rules using **IIS**
- Option 2: Configure URL Rewrite rules using **web.config** file.

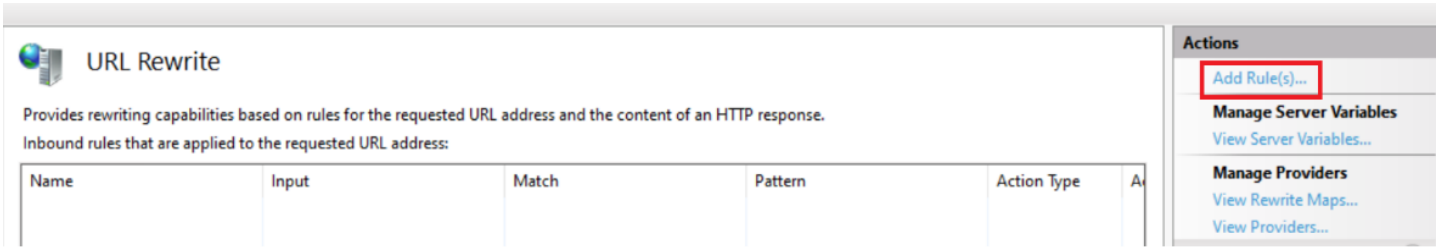
Option 1: Configure URL Rewrite rules using IIS:

The following two rules must be set up in URL Rewrite:

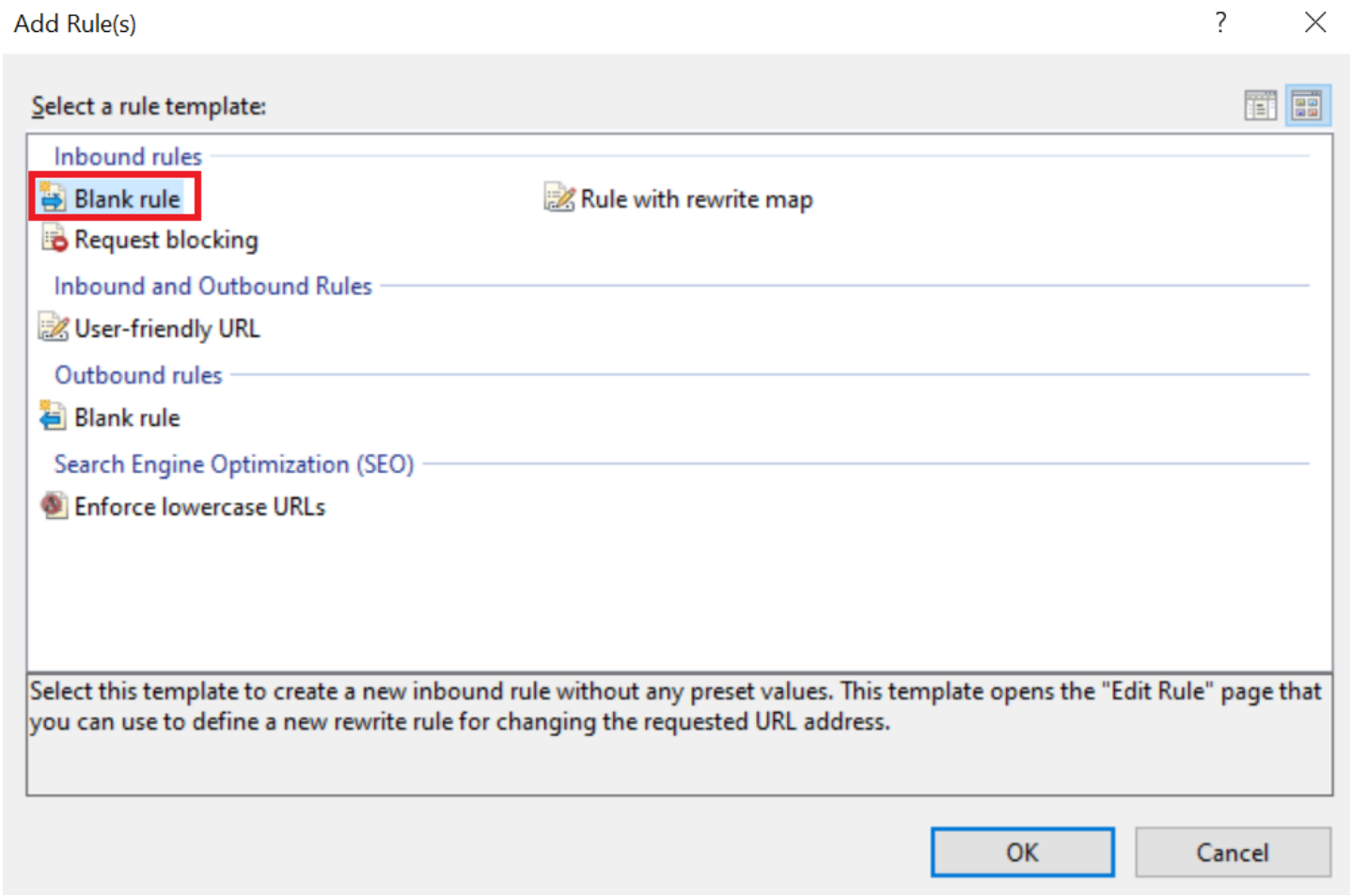
Entry point with link to Hoops Server

Web Socket connections

8. From Actions column, click **Add Rules**.



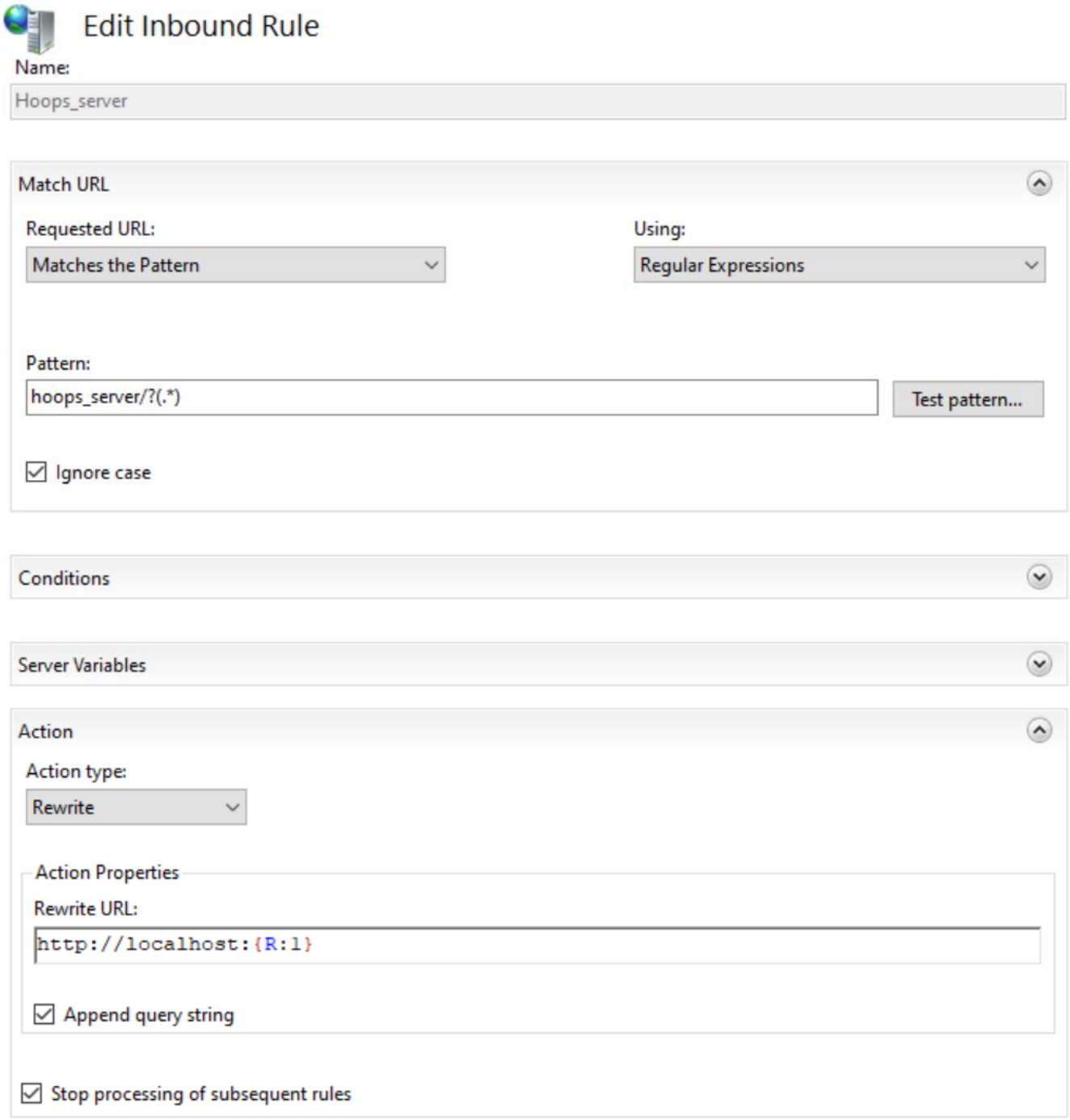
9. Click Blank Rule.



10. Add the following details to the rule:

- Name: Hoops_server
- Pattern: hoops_server/?(.*)

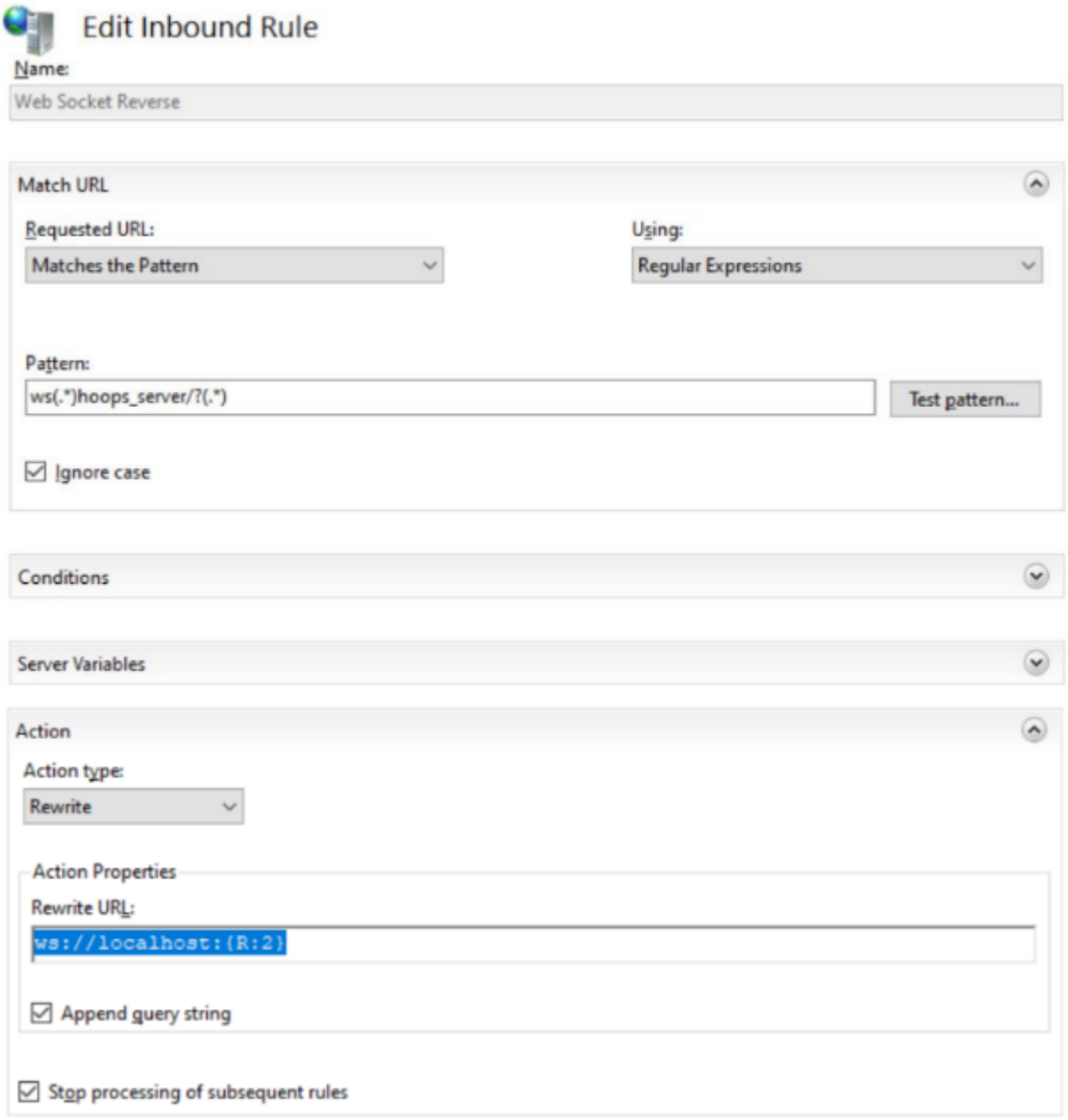
- Rewrite URL: http://localhost:{R:1}



11. Click **Apply**.
12. Add another Blank Rule with the following details:

- Name: Web Socket Reverse
- Pattern: ws(.*)hoops_server/?(.*)

- Rewrite URL: ws://localhost:{R:2}



Option 2: Configure URL Rewrite rules using web.config file:

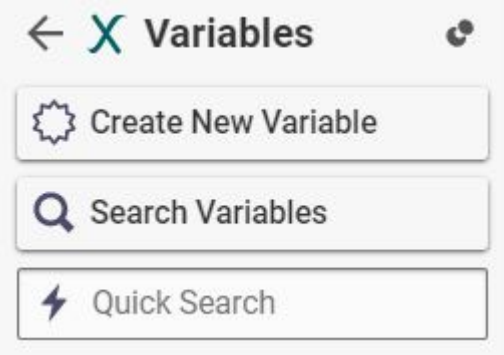
13. Open web.config file from Aras Innovator code tree: “C:\Aras\InnovatorServer\Innovator\web.config file”.
14. In the < `system.webserver` > tag, add the following code as shown in the screenshot below:

```
<system.webServer>
    <rewrite>
        <rules>
            <rule name="Web Socket Reverse" enabled="true" stopProcessing="true">
                <match url="ws(.*)hoops_server/?(.*)" />
                <action type="Rewrite" url="ws://localhost:{R:2}" />
            </rule>
            <rule name="hoops_server" enabled="true" stopProcessing="true">
                <match url="hoops_server/?(.*)" />
                <action type="Rewrite" url="http://localhost:{R:1}" />
            </rule>
        </rules>
    </rewrite>
</system.webServer>
```

- Hoops_server in match url can be different and this will be accounted as “Hoops Server Url”.
- Action url should point to Instance of the Hoops Server.

Next step is to adjust **HOOPS_ServerUrl**.

15. In the Aras Innovator Instance, from the **Table of Contents**, expand **Administration** and select **Variables**.
16. Click Search Variables.



17. Click **Search** and select **HOOPS_ServerUrl**. The HOOPS_ServerURL form appears.

Change HOOPS_ServerUrl to point to Hoops Server through configured proxy.

18. Click Edit.
19. In the **Value** field, add `http://localhost/InnovatorServer/hoops_server/11182`

    where,

    - Localhost – IP Address of the machine
    - InnovatorServer- Innovator instance of the user
    - 11182- Hoops Server port number

20. Click Done to save the changes.
21. Go to “HOOPS Server\server\node\Config.js”.
22. For publicHostname parameter, make the following change:

publicHostname: “localhost/Your innovator instance/hoops_server”,

```
publicHostname: "localhost/InnovatorServer/hoops_server",
```

23. Go to “HOOPS Server\server\node\lib\SpawnerExtension.js”.
24. For getEndpoint(`req, res`)method,

Change:

"endpoint":`${Utils_1.getWsProtocol(this.extConfig.sslEnableScServer)}://${hostname}:${this.extConfig.spaw

To:

"endpoint":`${Utils_1.getWsProtocol(this.extConfig.sslEnableScServer)}://${hostname}/${this.extConfig.spaw

```
    getEndpoint(req, res) {
        const hostname = Utils_1.getPublicHostname(this.extConfig.publicHostname, this.extConfig.ipVersion);
        const json = {
            "endpoint": `${Utils_1.getWsProtocol(this.extConfig.sslEnableScServer)}://${hostname}/${this.extConfig.spawnServerPort}/`,
        };
        Utils_1.sendJsonResponse(req, res, 200, json);
    }
}
exports.SpawnerExtension = SpawnerExtension;
//# sourceMappingURL=SpawnerExtension.js.map
```