

Running Node.js with IBM HTTP Server for i

Using JavaScript programs for server-side web development

Xu Meng
Zheng Chang Qing

September 14, 2015
(First published September 14, 2015)

Node.js, the server-side JavaScript runtime platform, is now supported on IBM i. The [event-driven, non-blocking](#) I/O model supported by node.js is well suited to web application development. IBM® HTTP Server for i is a powerful Apache-based web server on IBM i. This article explains how to associate IBM HTTP Server for i with Node.js through a Node.js FastCGI add-on to create a powerful and secure web environment.

Introduction

This article describes how to associate IBM HTTP Server for i with Node.js, leveraging the Node.js FastCGI add-on. Both local and remote associations are introduced in this article. Additionally, a simple Node.js example in this article explains how to use the HTTP Server authentication mechanism to secure and protect an application. This article provides step-by-step guidance to set up the environment.

Node.js is an open source project based on Google Chrome V8 engine. It provides a platform for server-side JavaScript applications running without the need for a browser. Node.js is now supported on the IBM i platform. To get started with Node.js on IBM i, it is recommended to read the article, [Native JavaScript applications on IBM i with Node.js](#).

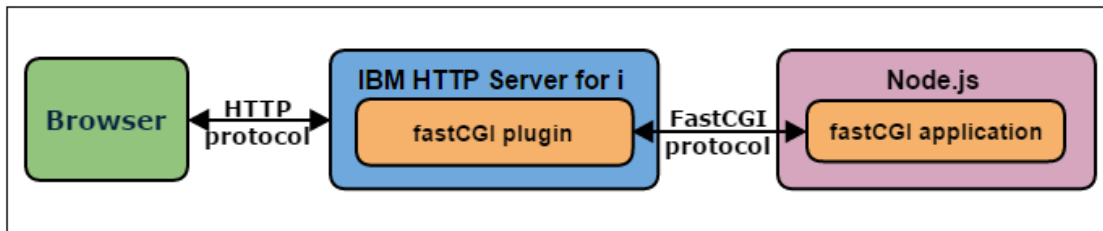
Note: Node.js is an official trademark of Joyent.

FastCGI is an open standard, language-independent and Common Gateway Interface (CGI) like protocol for interfacing interactive programs with a web server. The open source Node.js FastCGI add-on has been enabled on IBM i. This gives Node.js on IBM i the ability to communicate with other servers through the FastCGI protocol. We have supported this for the Hypertext Preprocessor (PHP) ecosystem for many years. For more details about the FastCGI on IBM i, refer to [FastCGI](#).

IBM HTTP Server (powered by Apache) for i is a complete web server product, which offers several components and features to assist in your website configuration and development. With

the IBM HTTP Server (powered by Apache) for i, you have everything you need to quickly and easily establish a web presence to get started working on the web for business.

Figure 1. Node.js FastCGI application flow



Software prerequisites

Node.js for IBM i is packaged as part of 5733OPS, which is supported in IBM i 7.1 and later versions. Node.js is delivered in option 1 of this new Licensed Program Offering (LPO). In addition to installing the new LPO, some additional software programs are required in order to use Node.js and the FastCGI add-on on IBM i. The FastCGI add-on, `node-fastcgi`, is included in the following program temporary fix (PTF) groups.

- Required licensed programs:
 - 5733OPS, option 1 – IBM i open source solutions
 - 5770SS1, option 33 – Portable app solutions environment
 - 5733SC1, option 1 – OpenSSH, OpenSSL, zlib
 - 5770DG1,*BASE – IBM HTTP Server for i
- Required Group PTF
 - IBM i 7.1 PTF Group SF99368 - level 34 (or higher)
 - IBM i 7.2 PTF Group SF99713 - level 8 (or higher)

After successfully installing the prerequisites, you need to install the FastCGI add-on, **node-fastcgi**, in the `/QOpenSys/QIBM/ProdData/Node/os400/node-fastcgi` directory.

In the following sections, we create a HTTP Server instance and associate it with a Node.js application using FastCGI protocol. And finally, secure the FastCGI application with IBM i profile validation.

HTTP Server instance creation and configuration

This section describes the configuration of the HTTP Server. First, we need to have a HTTP instance and configure it to load the IBM i FastCGI plug-in so that it can communicate with the backend FastCGI application.

Step 1. Create a HTTP Server instance.

IBM Web Administration for i (<http://<your.server.name>:2001/HTTPAdmin>) can be used easily to create an instance of IBM HTTP Server for i. Please refer to [IBM i Knowledge Center](#) for details of how to create an instance of HTTP Server for i.

This article uses an HTTP instance named **HTTPSERV** as example. It listens to port **10036** and the root directory is **/www/htpserv**.

Step 2. Update the HTTP Server instance configuration file with FastCGI setting.

When you create the instance through IBM Web Administration for i, click **Edit Configuration File** in the left panel of IBM Web Administration for i.

In the right text box, add following directive to the top of the configuration file to load the HTTP Server FastCGI plug-in.

```
LoadModule zend_enabler_module /QSYS.LIB/QHTTSPVR.LIB/QZFAST.SRVPGM
```

Add the following two directives to add a new handler '**FastCGI-script**' for requests to files with the .jsx file extension.

```
AddType application/x-httpd-javascript .jsx
AddHandler fastcgi-script .jsx
```

The full configuration file httpd.conf looks as shown below.

```
LoadModule zend_enabler_module /QSYS.LIB/QHTTSPVR.LIB/QZFAST.SRVPGM
Listen *:10036
DocumentRoot /www/htpserv/htdocs
HotBackup Off

AddType application/x-httpd-javascript .jsx
AddHandler fastcgi-script .jsx
```

Figure 2. HTTP configuration file for FastCGI setting

The screenshot shows the IBM Web Administration for i interface. The top navigation bar includes 'Setup', 'Manage', 'Advanced', and 'Related Links'. Below this, there are tabs for 'All Servers', 'HTTP Servers', 'Application Servers', and 'Installations'. The 'HTTP Servers' tab is active, showing a 'Stopped' status and a dropdown for 'Server: HTTPSERV - Apache' and 'Server area: Global configuration'. The left sidebar contains a tree view with 'Tools' expanded, where 'Edit Configuration File' is selected. The main content area is titled 'HTTPSERV > Edit Configuration File' and shows the configuration file content for 'Selected file: /www/htpserv/conf/httpd.conf'. The following directives are highlighted with blue boxes:

```
LoadModule zend_enabler_module /QSYS.LIB/QHTTSPVR.LIB/QZFAST.SRVPGM
Listen *:10036
DocumentRoot /www/htpserv/htdocs
HotBackup Off

AddType application/x-httpd-javascript .jsx
AddHandler fastcgi-script .jsx
```

Step 3. Update the home page of the instance.

Edit the index.html file located in `/www/httpserv/htdocs` as below. This simple page shows a text box that accepts IBM i system value name input and a button that runs the query.

```
<!DOCTYPE HTML>
<html lang="en-US">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>FastCGI Example</title>
</head>
<style>
input {
height:30px;
width:200px;
text-align:center;
border:#ccc solid 1px
}
</style>
<body>
<form name="input" action="dspsysval.jsx" method="get">
<input type="text" name="key" placeholder="QCCSID"/>
<input type="submit" value="Display System Value"/>
</form>
</body>
</html>
```

Step 4. Verify that the HTTP Server instance is workable.


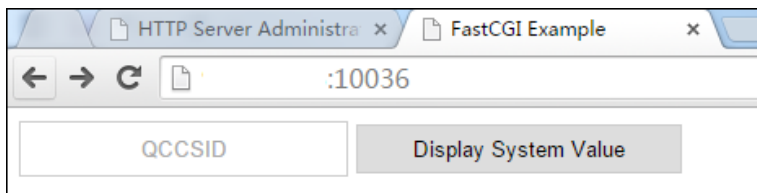
Start the instance by clicking the  icon at the top of panel. Access `http://your.server.name:10036` to check whether the instance is working. If everything is fine, the following web page will be displayed. But, it cannot serve any FastCGI request yet as we have not configured the corresponding backend part. We will set up and associate a FastCGI application in next section to handle a FastCGI requests.

Figure 3. Default HTTP Server web page

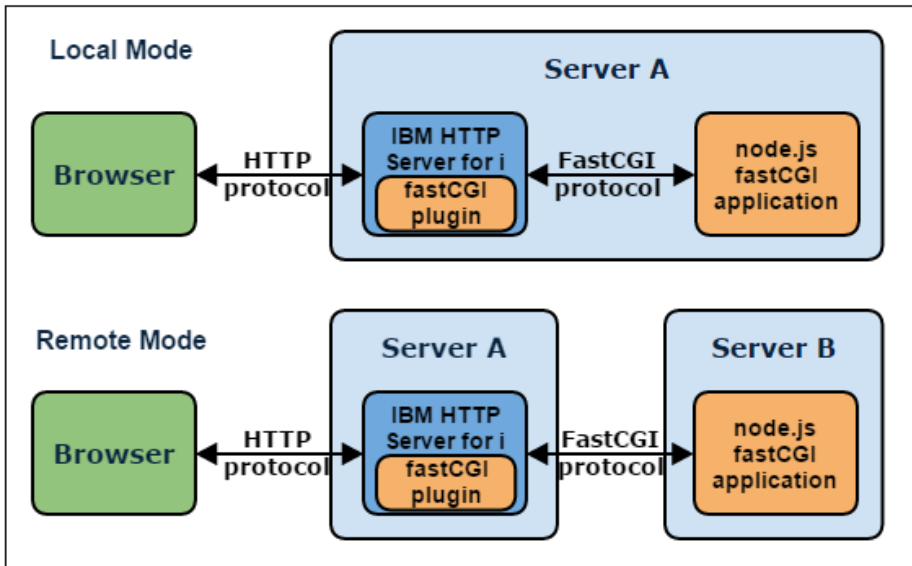


Node.js FastCGI application creation

The open source FastCGI add-on, node-FastCGI, has been enabled in the latest node.js PTF. If this add-on is installed successfully, the `/QOpenSys/QIBM/ProdData/Node/os400/node-fastcgi` directory should exist and contain two template files.

According to the FastCGI protocol, the web server and FastCGI application can be on the same system or on different systems. There are two modes for the association, (local or remote) as shown in Figure 4. For a simple configuration, the local mode enables the FastCGI application that is running on the same server of the frontend HTTP Server. But in some complex network environments (for example, the frontend HTTP Server just works as a proxy or a balancer and there are multiple backend application servers), remote mode is the best choice. In the remaining sections, both the modes are introduced.

Figure 4. The two working mode of FastCGI server



There are two key components needed to configure the association. One is the configuration file for FastCGI plug-in used by HTTP Server to communicate with the backend FastCGI application. The other is the Node.js FastCGI application itself. Currently, Node.js for i provides templates for **fastcgi.conf** and **fcgi.js**, which are located in /QOpenSys/QIBM/ProdData/Node/os400/node-fastcgi. Users need to update the content of these templates for practical requirements. The following steps demonstrate how to build an example FastCGI application from these templates.

Step1. Copy template files to HTTP Server configuration directory.

Copy the template files, fastcgi.conf and fcgi.js, to the configuration directory of the HTTPSERV HTTP instance.

```

cp -p /QOpenSys/QIBM/ProdData/Node/os400/node-fastcgi/fastcgi.conf /www/httpserve/conf/
cp -p /QOpenSys/QIBM/ProdData/Node/os400/node-fastcgi/fcgi.js /www/httpserve/conf/
  
```

Note: It is not recommended to modify the original template files **directly**. You can modify the copies of them and make sure that the HTTP Server runtime profile, QTMHHTTP, has *RX authorities to them. You can copy fcgi.js to any other directory. But, if you are configuring in remote mode, you should copy fcgi.js to a directory of the system running the Node.js FastCGI application.

By default, the content of the fastcgi.conf file is as given below:

```

; Local Mode --the HTTP server should run at the same IP address with the FastCGI server.
Server type="application/x-httpd-javascript"
  CommandLine="/QOpenSys/QIBM/ProdData/Node/os400/node-fastcgi/fcgi.js" StartProcesses="1"

; Remote Mode --the FastCGI server can run at different IP addresses. Set 'Binding=' to the IP/port
  that the FastCGI server listens to.
; ExternalServer type="application/x-httpd-javascript" Binding="127.0.0.1:8080" ConnectionTimeout="300"
  RequestTimeout="300"

; Where to place socket files
IpcDir /www/fastcgi/logs
  
```

Note: In this configuration file, the semi-colon, ';', is used to comment out directives. You can refer to the detailed [configuration guide](#) for more information.

Local

There are three directives in `fastcgi.conf` (one is commented out by default). Directive `Server` allows the FastCGI application to run in the local mode. It means that the HTTP Server and the FastCGI application must exist on the same system. When you start the associated HTTP Server, the HTTP Server will automatically start the FastCGI application defined by the `CommandLine` property.

Remote

The second directive that is commented out is for the remote mode. In this mode, the HTTP Server and the FastCGI application can run in different systems. The attribute, `Binding`, defines the IP address and port that the FastCGI application listens to. If you want to enable this mode, comment out the first directive to disable the local mode first and uncomment the second directive. They cannot co-exist.

The last directive, `IpCDir`, defines where to place the socket files of the FastCGI application. We need to change it to an existing directory.

Step 2. Update `fastcgi.conf`.

To configure the **local** mode, update `fastcgi.conf` with the following content.

```
Server type="application/x-httpd-javascript" CommandLine="/www/httpserv/conf/fcgi.js"
    StartProcesses="1"
; ExternalServer type="application/x-httpd-javascript" Binding="127.0.0.1:8080 "
IpCDir /www/httpserv/logs
```

To configure the **remote** mode, update `fastcgi.conf` with the following content.

```
; Server type="application/x-httpd-javascript" CommandLine="/www/httpserv/conf/fcgi.js"
    StartProcesses="1"
ExternalServer type="application/x-httpd-javascript" Binding="127.0.0.1:8080"
IpCDir /www/httpserv/logs
```

Now, we have completed the FastCGI plug-in configuration for IBM HTTP Server for i.

Step 3. Update `fcgi.js` to handle FastCGI requests.

The default content of `/www/httpserv/conf/fcgi.js`. It only responds to the URL for FastCGI request from HTTP Server side.

```
#!/QOpenSys/QIBM/ProdData/Node/bin/node
var fcgi = require('/QOpenSys/QIBM/ProdData/Node/os400/node-fastcgi');
var fcgiServer = fcgi.createServer(function(req, res) {
  if (req.method === 'GET') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('GET: ' + req.url);
  } else {
    res.writeHead(501);
    res.end();
  }
});
fcgiServer.listen(); //Local mode
// fcgiServer.listen(8080); //Remote mode
```

Make a note of the last two lines. If the FastCGI application does not listen to a valid port, then it works in the local mode. Starting the HTTP Server will run this FastCGI application automatically. If the FastCGI application listens to a valid port, then it will work in the remote mode. Change the line according to your real requirement.

To process different types of requests, we can update this file to dynamically load different node.js applications according to the requested .jsx file name.

```
#!/QOpenSys/QIBM/ProdData/Node/bin/node
var fcgi = require('/QOpenSys/QIBM/ProdData/Node/os400/node-fastcgi');
var url = require('url');
var fs = require('fs');

var fcgiServer = fcgi.createServer(function(req, res) {
  var app = __dirname + url.parse(req.url).pathname.slice(0, -1);
  //Replace .jsx with .js to get the real path.
  fs.exists(app, function(exists){
    if(exists){
      var handler = require(app); // Load the javascript module according the URL requested
      handler.process(req, res);
    } else {
      res.writeHead(404);
      res.end();
    }
  });
});
fcgiServer.listen(); //Local mode
// fcgiServer.listen(8080); //Remote mode
```

Now we have the simple framework of the FastCGI application, allowing it to invoke Node.js programs as FastCGI applications.

Step 4. Add a system value query handler.



The previous steps configured the HTTP Server to run the Node.js program. We now create a Node.js program that is used to display system values. In the same directory of **fcgi.js**, create a new JavaScript file named **dspsysval.js** with the following content. This is a handler for system value query.

```
var url = require('url');
var db = require('/QOpenSys/QIBM/ProdData/Node/os400/db2i/lib/db2');
var xt = require('/QOpenSys/QIBM/ProdData/Node/os400/xstoolkit/lib/itoolkit');
var wk = require('/QOpenSys/QIBM/ProdData/Node/os400/xstoolkit/lib/iwork');

exports.process = function(req, res) { // Implement the interface 'process()'
  var key = url.parse(req.url, true).query.key.toUpperCase(); // Get the query key from the URL.
  var conn = new xt.iConn('*LOCAL');
  var work = new wk.iWork(conn);
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end(key + ' = ' + work.getSysValue(key)); // Write the system value to the HTTP response.
};
```

This program reads the key name from the URL and queries the corresponding value of local IBM i system using the Node.js toolkit application programming interface (API), `getSysValue()`. Then, it writes the system value to the HTTP response.

Step 5. Start HTTP Server and the Node.js FastCGI application.

Open the IBM Web Administration for i (<http://your.server.name:2001/HTTPAdmin>) and click the  button to start the HTTPSERV instance. If the instance has been running already, click the  button to restart it. We can also use the control language (CL) command to start or restart the HTTP instance.

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(HTTPSERV)
STRTCPSVR SERVER(*HTTP) RESTART(*HTTP) HTTPSVR(HTTPSERV)
```

When HTTP Server is started, we can issue the following CL command to check its status.

```
WRKACTJOB SBS(QHTTPSVR)
```

Figure 5. FastCGI application running status

Subsystem/Job	User	Type	CPU %	Function	Status
HTTPSERV	QTMHHTTP	BCI	.0	PGM-QZSRLOG	SIGW
HTTPSERV	QTMHHTTP	BCI	.0	PGM-QZSRHTTP	SIGW
HTTPSERV	QTMHHTTP	BCI	.0	PGM-zfcgi	SELW
HTTPSERV	QTMHHTTP	BCI	.0	PGM-node	SELW

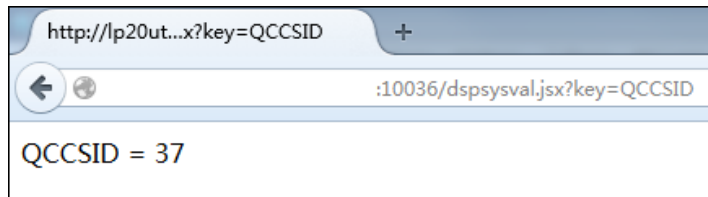
Job PGM-zfcgi and PGM-node exist only for the local mode. In the remote mode, when the HTTP Server is started, you have to manually start the `fcgi.js` program in PASE. Remember to keep the port number same in both `fcgi.js` and `FastCGI.conf`.

```
node /path/to/fcgi.js
```

Step 6. Access and verify the association between HTTP Server and Node.js.

We can browse the home page <http://your.server.name10036> and enter any system value name you want to query to verify the association. If the QCCSID system value is displayed as shown in Figure 6, then it means that the FastCGI application is working.

Figure 6. The FastCGI response



Secure your FastCGI application with IBM i profile validation

This section describes how to use the IBM HTTP Server for i server authentication function to protect the application. Only visitors with IBM i profiles are allowed to access the application.

Step 1. Update HTTP Server configuration with authentication setting.

Click the 'Edit Configuration File' button in the IBM Web Administration for i to configure the HTTPSERV instance. Add the following directives (in bold font) at the end.

```
LoadModule zend_enabler_module /QSYS.LIB/QHTTPSVR.LIB/QZFAST.SRVPGM
Listen *:10036
DocumentRoot /www/httpd/htdocs
HotBackup Off

AddType application/x-httpd-javascript .jsx
AddHandler fastcgi-script .jsx

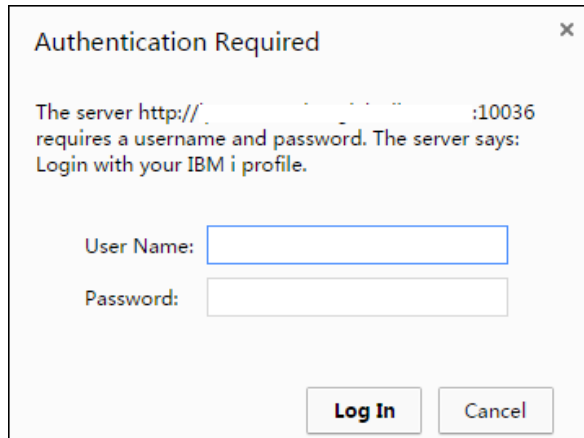
<Location />
  AuthType Basic
  AuthName "Login with your IBM i profile"
  Require valid-user
  PasswdFile %%SYSTEM%%
</Location>
```

These directives enable authentication for all .jsx file requests. The complete configuration file, httpd.conf, is shown above.

Step 2. Restart HTTP Server to make the authentication setting take effect.

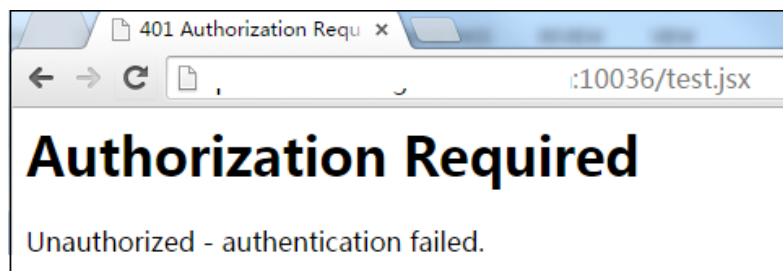
Restart the HTTPSERV instance and access <http://your.server.name:10036> to query a system value. A dialog box prompts you to log in with your IBM i profile.

Figure 7. Authentication dialog



If you passed the authentication, then you can access the query page and the backend FastCGI application. Otherwise, the HTTP Server will block the access and respond with the following error message.

Figure 8. Blocking unauthorized access



Summary

The event-driven, non-blocking I/O feature makes Node.js primarily used as a lightweight web server. There are more and more third-party add-ons being developed by open source communities that can greatly extend the function and value of Node.js. Additionally, the IBM HTTP Server for i is a highly customizable, robust, and proven web server. It uses IBM i specific features such as High Availability (HA), Fast Response Cache Accelerator (FRCA), profile authentication to achieve higher security and better performance.

Now, we can connect these IBM i specific features with the FastCGI protocol and complement each other and take advantage of the strengths of each. It is even possible to deploy these servers in a distributed system. The flexible combination of modes are applicable to different scenarios.

Resources

- [Node.js](#)
- [FastCGI on IBM i](#)
- [Native JavaScript applications on IBM i with Node.js](#)

- [Getting started with the IBM Web Administration for i interface](#)
- [IBM i Technology Updates](#)
- [Rev up your Tomcat server on IBM i](#)
- [IBM i Open Source](#)

© Copyright IBM Corporation 2015

(www.ibm.com/legal/copytrade.shtml)

[Trademarks](#)

(www.ibm.com/developerworks/ibm/trademarks/)