IBM SDK for Node.js - z/OS, V8.0
Version 8 Release 0

User's Guide

SC27-9802-00 (2020-11-04)

Note

Before using this information and the product it supports, read the information in “Notices” on page 39.

This edition applies to Version 8 of IBM® SDK for Node.js - z/OS® (order number: SC27-9802-00) and to all subsequent releases and modifications until otherwise indicated in new editions.

It is our intention to update the product documentation for this release periodically, without updating the order number. If you need to uniquely refer to the version of your product documentation, refer to the order number with the date of update.

Last updated: 2020-11-04

Chapter 1. Articles, blogs, and videos

Business value

- Article: IBM SDK for Node.js – z/OS Offers Co-Location and Technology Advantages
- Blog: Enterprise Architects: an introduction to Node.js in CICS
- Blog: Node.js Developers: an introduction to Node.js in CICS
- Blog: System Programmers: An introduction to Node.js in CICS
- Blog: Reasons to Host your Node.js Applications on z/OS
- Blog: Node.js on z/OS Use Case: API Orchestration
- Blog: Node.js runs on everything...even mainframes!
- Video: IBM SDK for Node.js – z/OS (Overview)
- Blog: Rumor Confirmed: Node.js is now available on z/OS!

Getting the offering

- Blog: Paid S&S for IBM SDK for Node.js – z/OS
- Blog: How to obtain IBM SDK for Node.js – z/OS, at no charge
- Blog: Obtain the SMP/E edition and optional support of IBM SDK for Node.js – z/OS

How-to

- Blog: Accessing IBM Db2 with IBM SDK for Node.js – z/OS
- Blog: Getting started with Node.js in CICS
- Blog: Re-use code with npm community packages for Node.js
- Blog: HTTPS web services and clients using IBM SDK for Node.js – z/OS®
- Blog: Developing Node.js for z/OS with IBM Developer for z Systems

Other

- Blog: Test Node.js on z/OS at no cost with the cloud-based IBM Z Software Trial Program
- Blog: Support for Node.js in CICS with z/OS Provisioning Toolkit V1.1.5
- Blog: IBM Doc Buddy V2.2 – One-stop mobile IBM Z content portal
- Blog: Announcing support for Node.js in CICS with z/OS Provisioning Toolkit V1.1.2
- Blog: Node.js for z/OS – last update to v6 now available
- Blog: IBM SDK for Node.js – z/OS Version 8 now available!

Chapter 2. Overview of IBM SDK for Node.js - z/OS

The IBM SDK for Node.js - z/OS is an extended implementation of the Node.js runtime. Node.js extends the JavaScript programming language with a set of useful server-side APIs to provide a programming platform that allows efficient development of real-time, scalable server-side network applications.

Node.js is a runtime framework that enables you to write server applications that use the syntax and structures of JavaScript, a language familiar to 70% of developers who responded to the 2019 Stack Overflow survey. Node.js extends JavaScript by providing features such as:

- The creation, modification, and deletion of local (server-side) files
- Event-driven, non-blocking I/O model
- Scalability
- A very small "footprint" on the server
- API orchestration
- Web applications
- Full-stack development

The SDK is a z/OS port of the open-source community Node.js and provides the following extensions:

- Support for IBM platforms, including z/OS interoperability, diagnostic, and monitoring capabilities
- Additional Reliability, Availability, and Serviceability (RAS) features

Chapter 3. What's new in IBM SDK for Node.js - z/OS, V8.0

This section lists the new functions introduced in IBM SDK for Node.js - z/OS, Version 8:

- IBM SDK for Node.js - z/OS, Version 8 is based on the V8.16.0 release of Node.js from the open-source community.
- Newly created files are now tagged as ISO8859-1 by default.
- The experimental async/await, which is a modern approach, allows you to write asynchronous code that looks like synchronous code.
- New modules are added, including Async Hooks API and N-API API for native add-ons.
- The updated npm 6.4.1 (Node.js package manager) now includes standardized lockfiles for cross-package-manager compatibility and automatic saving of newly installed packages without explicitly adding the `--save` flag.
- N-API provides an ABI-stable abstraction over native JavaScript APIs, ensuring smooth migration of native add-ons modules across Node.js major releases.
- Previously the C++ compiler included with IBM SDK for Node.js - z/OS could only compile C++ add-ons in the EBCDIC format. New in Version 8, you can now compile the ASCII format which is the most popular format for external npm modules.
- Functional enhancements in DNS and OS modules are included.
- The buffer module is initialized to zeros providing better security against potential vulnerabilities.

Chapter 4. Understanding IBM SDK for Node.js - z/OS

The IBM SDK for Node.js - z/OS is a server-side JavaScript runtime environment. You can create applications and tools by using the JavaScript language and Node.js APIs, then use the SDK to deploy them onto your z/OS system.

JavaScript is a pillar of web applications, and a familiar language to many web and front-end developers. Using JavaScript, you can leverage the same technology and pool of skills to build end-to-end applications (front-end and back-end) in the same language. JavaScript and SDK for Node.js - z/OS offer a versatile platform that typically improves speed of development, delivering scalable applications in fewer lines of code (as shown by PayPal in their migration from Java™ to Node.js).

A prominent use of IBM SDK for Node.js - z/OS is to develop network applications that provide a web portal or http endpoints, while orchestrating services and data in the backend. SDK for Node.js - z/OS provides a non-blocking, event-driven, single-threaded approach that exploits z/OS's asynchronous I/O capabilities to achieve scalability. Instead of having dedicated threads to handle server connections, a network request coming into the Node.js application will result in an asynchronous I/O event, which will then trigger the corresponding handler function within the Node.js application. As a result, Node.js applications can typically scale well to large number of connections, while maintaining low memory and CPU footprint.

The IBM SDK for Node.js - z/OS provides two main executables in the z/OS UNIX System Services (z/OS UNIX) environment - `node` and `npm`. A basic Node.js application provides an initial JavaScript file that serves as the entry point of the application, which is typically named `server.js` or `app.js`. To invoke the Node.js application, you need to execute the following command from z/OS UNIX:

```
$> node server.js
```

Node.js applications are typically designed to act as microservices, which are built on top of established node packages using the built-in `npm` tool. This style discourages monolithic applications, and encourages better encapsulation and reuse that is amenable to agile development, micro-services, and APIs. Module dependencies are tracked within a `package.json` file, which also contains other versioning and meta-data about the application. The `npm` tool will parse the `package.json` file and install the necessary dependencies to build the application.

IBM SDK for Node.js - z/OS also supports a native add-on feature, which allows C/C++ code to be bound as part of the JavaScript module. This feature is useful for invoking native drivers or existing assets that are written in C/C++. The SDK for Node.js - z/OS includes a 64-bit C/C++ compiler (`njsc`/`njsc++`) with C++11 language support to facilitate the compilation of native add-on modules.

For an example of how to build your own sample Node.js application, refer to Chapter 6, “Getting started with IBM SDK for Node.js - z/OS,” on page 15.

Chapter 5. Installing and configuring IBM SDK for Node.js - z/OS

IBM SDK for Node.js - z/OS is available in the following two editions:

- SMP/E edition, which can be used in a production environment
- PAX archive installable, which is offered for evaluation purpose only

Installing and configuring the SMP/E edition

The Program Directory for the product details specific installation requirements and instructions in Chapter 5 and Chapter 6. For information about the latest APAR fixes, see Fix list for IBM SDK for Node.js - z/OS.

The following checklist summarizes the key configuration steps for a successful installation.

Hardware prerequisites

- z15™
- z14®
- z13®/z13s®
- zEnterprise® EC12/BC12
- zEnterprise 196/114

Software prerequisites

- z/OS UNIX System Services enabled on any of following operating systems:
  - z/OS V2R4 with all the following PTFs installed:
    - UI64839
    - UI64940
    - UI64837
    - UI64830
  - z/OS V2R3 with all the following PTFs installed:
    - UI61308
    - UI61375
    - UI61747
  - z/OS V2R2 with all the following PTFs installed:
    - UI62788
    - UI46658
    - UI62416
    - UI62415
- Integrated Cryptographic Services Facility (ICSF) must be enabled on systems where IBM SDK for Node.js - z/OS is run. For details, refer to ICSF System Programmer's Guide (SC14-7507) and ICSF Administrator's Guide (SC14-7506).
- Perl 5.6.x or later, required for installation of the provided C/C++ runtime and compiler.
- Python 2.7.13 or later 2.x versions, required to compile native add-ons. Must also be available on your PATH. Note that Python 3.x is not compatible.

- Make 4.1 or later, required to compile native add-ons. Must also be available on your PATH.

Note: To download Perl, Python, or Make, go to the Rocket Software website, sign in or sign up, click Downloads, navigate to the z/OpenSource category, and use the Filter List search to find Perl (ID USSP-703), Python 2.7.13 (ID USSP-705), or Make 4.1 (ID USSP-722).

- If your shell is bash, make sure that the Rocket bash is installed because earlier versions of bash on z/OS are known to have autoconversion issues. The Rocket bash is available for download at "Bash for z/OS" on the Rocket website.

Configuration

IBM SDK for Node.js - z/OS is an OMVS-based application, which requires certain configuration on the z/OS UNIX System Services file system to ensure proper operation.

- Validate that `/usr/bin/env` exists. If not configured, refer to the instructions in Chapter 11, “Verifying the env command path,” on page 25.
- Ensure that `/tmp` has at least 1 GB or more of disk space configured. To use an alternative file system, you can set the TMP environment variable to a directory that has sufficient space.

Environment variables

You need to set the following environment variables before using IBM SDK for Node.js - z/OS.

- Set the auto conversion environment variables:

```
export _BPXK_AUTOCVT=ON
export _CEE_RUNOPTS="FILETAG(AUTOCVT,AUTOTAG) POSIX(ON)"
```

- Configure the PATH environment variable to include the `bin` directories for IBM SDK for Node.js - z/OS, Python, and GNU Make tools with the following command:

```
export PATH=$PATH:$PathPrefix/usr/lpp/IBM/cnj/IBM
/node-latest-os390-s390x/bin:$PathPrefix/usr/lpp/IBM/cnj/njsc/bin:
/rsusr/rocket/bin
```

  Note: `$PathPrefix` can be undefined for default installation under `/usr/lpp/IBM/cnj/IBM`. If the product is not installed in the default installation location, contact your system administrator for the `$PathPrefix` value and ensure the installation was performed properly. For detailed information, see section 6.2.1 in Program Directory (GI13-4573-00).

- To build native add-ons, set the following environment variables:

```
export CXX="njsc++"
export CC="njsc"
export CXX_host=$CXX
export LINK=$CXX
export CFLAGS="-q64 -fexec-charset=ISO8859-1"
export CXXFLAGS="-q64 -fexec-charset=ISO8859-1"
export LDFLAGS="-q64"
```

- The C/C++ environment comes pre-configured when using **npm** to build native add-ons. However, if you invoke **node-gyp** directly to build native code, the following C/C++ compiler environment variables need to be set:

```
export _C89_CCMODE=1
export _CC_CCMODE=1
export _CXX_CCMODE=1
```

npm configuration

The **npm** utility is included in IBM SDK for Node.js - z/OS to install Node.js modules and packages. The **npm** utility performs checks to limit unsafe installation of modules by root / BPXROOT. To proceed to use BPXROOT id, you can take either of the following steps:

- Run with the **--unsafe-perm** npm option. For example:

```
npm install <npm_module> --unsafe-perm
```

  You can configure this option as default with:

```
npm config set unsafe-perm true
```

- Create a user id: nobody and ensure it is a member of a group. **npm** switches to this nobody uid/gid as necessary when running as BPXROOT.

Installing and configuring the PAX archive installable

Hardware prerequisites

- z15™
- z14®
- z13/z13s
- zEnterprise EC12/BC12
- zEnterprise 196/114

Software prerequisites

- z/OS UNIX System Services enabled on any of following operating systems:
  - z/OS V2R4 with all the following PTFs installed:
    - UI64839
    - UI64940
    - UI64837
    - UI64830
  - z/OS V2R3 with all the following PTFs installed:
    - UI61308
    - UI61375
    - UI61747
  - z/OS V2R2 with all the following PTFs installed:
    - UI62788
    - UI46658
    - UI62416
    - UI62415
- Integrated Cryptographic Services Facility (ICSF) must be enabled on systems where IBM SDK for Node.js - z/OS is run. For details, refer to ICSF System Programmer's Guide (SC14-7507) and ICSF Administrator's Guide (SC14-7506).
- Perl 5.6.x or later, required for installation of the provided C/C++ runtime and compiler.
- Python 2.7.13 or later 2.x versions, required to compile native add-ons. Must also be available on your PATH. Note that Python 3.x is not compatible.
- Make 4.1 or later, required to compile native add-ons. Must also be available on your PATH.

Note: To download Perl, Python, or Make, go to the Rocket Software website, sign in or sign up, click Downloads, navigate to the z/OpenSource category, and use the Filter List search to find Perl (ID USSP-703), Python 2.7.13 (ID USSP-705), or Make 4.1 (ID USSP-722).

- If your shell is bash, make sure that the Rocket bash is installed because earlier versions of bash on z/OS are known to have autoconversion issues. The Rocket bash is available for download at "Bash for z/OS" on the Rocket website.

Storage requirements

The PAX archive installable file contains an updated C/C++ runtime and compiler, which has the following storage requirements:

- 2 GB free space on HFS.
- 400 MB free space for the MVS™ dataset under the high-level qualifier (HLQ) where the compiler will be installed. The default HLQ will be your user ID.

Configuring

IBM SDK for Node.js - z/OS is an OMVS-based application, which requires certain configuration on the z/OS UNIX System Services file system to ensure proper operation.

- Validate that `/usr/bin/env` exists. If not configured, refer to the instructions in Verifying the env command path.
- Ensure that `/tmp` has at least 1 GB or more of disk space configured. To use an alternative file system, you can set the TMP environment variable to a directory that has sufficient space.

Installing

Follow these steps to install IBM SDK for Node.js - z/OS:

1. Download the PAX archive installable file to a z/OS machine.
2. Extract the PAX archive installable file inside an installation directory of your choice. For example:

   ```
   pax -rzf <path_to_pax_archive_installable_file> -p p
   ```

   Note: Step 3 and 4 are required if you intend to install or build Node native C/C++ add-ons. If you are unsure, it is recommended that you take step 3 and 4 to install the C/C++ compiler.
3. Export the following environment variables before you install the C/C++ compiler.

   COMPILER_INSTALL_HELD_MSGCLASS
   The compiler installation tool requires an appropriate MSGCLASS parameter that logs the output of a dependent JCL job to the HELD OUTPUT QUEUE. The default MSGCLASS value is s. You can override this value by using the COMPILER_INSTALL_HELD_MSGCLASS environment variable:

   ```
   export COMPILER_INSTALL_HELD_MSGCLASS=<custom MSGCLASS>
   ```

   Failure to set an appropriate MSGCLASS can result in XMIT errors, as described in Chapter 14, “Troubleshooting,” on page 31.

   COMPILER_INSTALL_HLQ
   By default, the compiler installation script creates datasets under your user ID’s HLQ. You can specify a custom HLQ for the datasets by setting the COMPILER_INSTALL_HLQ environment variable:

   ```
   export COMPILER_INSTALL_HLQ=<custom_HLQ>
   ```

   LE_INSTALL_HLQ
   By default, the compiler installation script assumes the Language Environment® (LE) datasets are installed under the HLQ CEE. You can specify a custom HLQ for LE by setting the LE_INSTALL_HLQ environment variable:

   ```
   export LE_INSTALL_HLQ=<custom_HLQ>
   ```

RTE_INSTALL_HLQ

By default, the compiler installation script assumes the Runtime Library Extension (RTE) datasets are installed under the HLQ CBC. You can specify a custom HLQ for RTE by setting the RTE_INSTALL_HLQ environment variable:

```
export RTE_INSTALL_HLQ=<custom_HLQ>
```

4. Run the following command to install the C/C++ compiler:

   ```
   ./setup.pl
   ```

   The script unpacks the compiler into the current working directory then loads the compiler datasets under your user ID's HLQ or the custom HLQ that you specified earlier. The script then runs a "Hello world" application to verify the installation.

5. Prior to running node, set the required environment variables as follows.

   Set the auto conversion environment variables:

   ```
   export _BPXK_AUTOCVT=ON
   export _CEE_RUNOPTS="FILETAG(AUTOCVT,AUTOTAG) POSIX(ON)"
   ```

   Add the full path of your installation directory to your PATH environment variable:

   ```
   export PATH=<installation_directory>/bin/:$PATH
   ```

   Optional: Set the environment variables for building native add-ons:

   ```
   export CXX="njsc++"
   export CC="njsc"
   export CXX_host=$CXX
   export LINK=$CXX
   export CFLAGS="-q64 -fexec-charset=ISO8859-1"
   export CXXFLAGS="-q64 -fexec-charset=ISO8859-1"
   export LDFLAGS="-q64"
   ```

6. Run the node and npm commands from the command line.

Uninstalling

Follow these steps to uninstall:

1. Uninstall the C/C++ compiler by running the following command from the installation directory:

   ```
   ./setup.pl -uninstall
   ```

2. Delete the installation directory.

Related reference

Node.js v8.16.0 API reference documentation

Chapter 6. Getting started with IBM SDK for Node.js - z/OS

To get started with IBM SDK for Node.js - z/OS, you can work with the Node edition of a "Hello World" program as an example. You can set up a simple web-server that responds to a GET request with the `Hello World` message through the following steps:

1. Under z/OS UNIX System Services (z/OS UNIX), create a separate HFS directory for the example "Hello World" program.

   ```
   $>mkdir myapp
   $>cd myapp
   ```

2. Set up the project by using the `npm` executable and filling out the requested details to populate a `package.json` file.

   ```
   $>npm init
   ```

3. Node.js boasts an impressive 1 million (approximately) open-source add-on modules, which can be used to develop applications. You can use the Express® web framework for the web-server example. To install the Express web framework, you can use the `npm` executable.

   ```
   $>npm install express --save
   ```

4. After the Express web framework is installed, a `node_modules` directory is added under the `myapp` directory.
5. Write the following sample code into a `server.js` file.

   ```
   /*Load express and instantiate, get the port number from command line*/
   var express = require("express");
   var server_instance = express();
   var port = process.argv[2];

   /*Respond to get requests to root dir*/
   server_instance.get("/", function (req, res){
       res.send("Hello World\n");
   })

   /*Start webserver*/
   server_instance.listen(port,function() {
       console.log("Listening on: " + port);
   })
   ```

6. Start the web-server application from the z/OS UNIX prompt.

   ```
   $>node server.js 1339
   Listening on: 1339
   ```

7. Open a web browser or use the `wget` command to access the server through HTTP on port 1339. The `Hello World` message is returned.

   ```
   $>wget https://<server address>:1339
   cat index.html
   Hello World
   ```

Through the previous steps, a simple web server is up and running.

Chapter 7. Migrating to IBM SDK for Node.js - z/OS, V8.0

Community changes

- Updated Node.js to 8.17.0 community release
- Updated Node package manager (npm) to 6.13.4
- Updated Libuv to 1.23.2
- Upgraded OpenSSL sources to 1.0.2s
- N-API: add API for synchronous functions

For more information on these changes, see Node.js 8 ChangeLog.

z/OS specific changes

You might need to take actions for the following change:

- When reading an untagged file, Node.js will now auto-convert data to CCSID 819 only if the file contains data in CCSID 1047. If you want to revert to the previous behavior, which always converts to CCSID 819 regardless of the CCSID of the data, you can take any one of the following settings:
  - Set the environment variable __UNTAGGED_READ_MODE to V6.
  - Set __UNTAGGED_READ_MODE to STRICT, which disables conversion of data.
  - Set __UNTAGGED_READ_MODE to WARN for the default auto-conversion behavior, but issue a warning if conversion occurs.

You don't need to take actions for the following changes:

- Fixed ipv6 bind.
- Enabled environment variables passed to a child process to override the z/OS environment variables of the parent process.
- Removed zoslib from Node.js and v8z; zoslib is now used as a dependency.
- Output to STDOUT/STDERR is now in EBCDIC.
- Allow DNSIPV4 environment variable to force IPV4 DNS resolution.
- Retrieve DNS server list and construct query packets correctly
- Substitute socketpair with a call to pipe for STDERR.
- Add _ALL_SOURCE/MAP_FAILED/_UNIX03_SOURCE macros for native npm ad dons.
- Backtrace of the stack displayed now stops at the starting address and uses a larger alternate signal stack.

Chapter 8. Building native add-ons with IBM SDK for Node.js - z/OS

IBM SDK for Node.js - z/OS supports native add-ons, which effectively allow you to load dynamically-linked shared objects written in C/C++ into your Node.js applications. With this ability, you are no longer confined to what you can write with JavaScript and Node.js APIs, but can extend your applications to tap existing native drivers, libraries, and services, etc.

The SDK for Node.js - z/OS includes a 64-bit C/C++ compiler (`njsc`/`njsc++`) with C++11 language support to facilitate the compilation of native add-on modules.

Before you build native add-ons, you must install all the following applications:

- Python 2.7.13 or later
- Make 4.1 or later
- The C/C++ compiler that you downloaded with IBM SDK for Node.js - z/OS

For installation instructions, see Chapter 5, “Installing and configuring IBM SDK for Node.js - z/OS,” on page 9.

You can build or install native add-on packages with the following commands:

```
npm install .                # To build in package directory
npm install <package name>   # To install the specified package from npmjs
```

Related information

Node.js v8.16.0 Documentation

Chapter 9. Debugging

You can debug your IBM SDK for Node.js - z/OS applications by using standard debugging techniques. In general, the process for debugging IBM SDK for Node.js - z/OS applications is much like that for debugging applications written in other server-side scripting languages. For example, you can use a debug mode on your application to inspect status and variable content, interrupt execution at specific breakpoint or when defined conditions or exceptions are encountered.

Node.js has a built-in debugger client that allows you to set breakpoints, step through code, and perform other debug operations. For more information about the client, refer to the Node.js documentation.

Chapter 10. Running

After you install, configure, and set the required environment variables for an IBM SDK for Node.js - z/OS application, you can run the application.

All IBM SDK for Node.js - z/OS applications are run in the same way:

1. Invoke the IBM SDK for Node.js - z/OS runtime binary file.
2. Supply the name of the required application

For example, you can use the following command to run an IBM SDK for Node.js - z/OS application:

```
node myapplication.js
```

If you want the IBM SDK for Node.js - z/OS runtime binary file to be invoked when your system starts, you can create a startup (shell) script that is called during system startup.

More arguments can be supplied to your application by appending them to the invocation command. For example,

```
node applicationexample.js args1 args2 args3
```

Each of the arguments is available to your application by using the `process.argv` array.

- The first element in the array (`process.argv[0]`) is the name of the IBM SDK for Node.js - z/OS runtime binary file: `node`.
- The second element in the array (`process.argv[1]`) is the name of the IBM SDK for Node.js - z/OS application script. In this example, `process.argv[1]` has the value `applicationexample.js`.
- Any remaining elements in the array correspond to other arguments that are supplied on the command line.

Chapter 11. Verifying the env command path

The shell scripts for IBM SDK for Node.js - z/OS require `/usr/bin/env`, but your system might have only `/bin/env`. You can take the following steps to verify the path for the **env** command.

1. Ensure that `/usr/bin/env` exists and provides a correct listing of the environment. In an SSH or Telnet shell environment, run the following command to verify the location and contents of **env**. The command returns a list of name and value pairs for the environment in your shell.

   ```
   /usr/bin/env
   ```

   If `/usr/bin/env` does not exist, complete the following steps to set it up:

   a. Locate the **env** program on your system. A potential location is `/bin/env`.

   b. Create a symbolic link (symlink) so that `/usr/bin/env` resolves to the true location of **env**. For example:

      ```
      ln -s /bin/env /usr/bin/env
      ```

   c. In an SSH or Telnet shell environment, run the following command to verify that the symlink works. The command returns a list of name and value pairs for the environment in your shell.

      ```
      /usr/bin/env
      ```

2. Verify that the symbolic link for the **env** command persists across system IPLs.

   Depending on how `/usr/bin/` is configured on your system, the symbolic link for `/usr/bin/env` might not persist across an IPL without additional setup. Ensure that your IPL setup includes creation of this symbolic link, if necessary.

Chapter 12. Re-using code with npm community packages for Node.js

One of the key advantages of Node.js is its vast collection of open-source community packages. The popular npmjs.com registry contains approximately one million packages, and each day, the community adds 400+ new packages; 3x more than the second most active community (Java).

Download and integrate these packages to jump-start your applications, achieving faster time-to-implementation, and supporting your agile development practices. This development style discourages monolithic applications and encourages better encapsulation.

The Node.js framework is made freely available to z/OS clients via IBM SDK for Node.js – z/OS, which allows you to use the community npm packages on z/OS. Furthermore, the SDK includes the npm client so you can install packages right from the command line. View the Getting Started guide for more information.

Here are some popular packages that you should try on z/OS:

- Express* – fast and minimal web framework
- vsam.js* – read and modify VSAM datasets on z/OS
- ibm-cics-api* – call service-enabled IBM CICS® applications
- zosconnect-node* – discover and access z/OS Connect service-enabled IBM Z® resources
- ibm_db2* – interface with IBM Db2® and IBM Informix®
- zos-node-accessor* – interact with z/OS easily (list, download, and upload datasets or files, submit JCL and track status, access SYSOUT dataset, etc)
- Other popular packages – lodash, underscore, uuid, body-parser, glob, yargs, minimist, through2

***IBM-contributed**

Explore more packages on the npmjs.org registry website, where you'll find exclusive-to-z/OS packages tagged with z/OS. Assume that packages without a z/OS tag will work across all platforms, including z/OS.

Chapter 13. Tagging files

The IBM SDK for Node.js - z/OS runtime is a native z/OS application that, by default, treats any files on the file system as an EBCDIC text file, unless otherwise tagged. Following the specification of JavaScript (ECMAScript), any text is converted to an UTF-8/UTF-16 internal representation within the JavaScript engine. If a Node.js application is hosting a web server (that is, http or Express-based application), text data streamed to client connections is in UTF.

For input files that might not be EBCDIC text (that is, ASCII text files or binary files), you can use the chtag utility to properly tag such files. For example, if the Node.js application uses a JPEG file, it can be tagged as binary, or if the application consumes JavaScript files that are encoded in the ASCII codepage, you can tag them as ASCII. The IBM SDK for Node.js - z/OS runtime queries any file tags and respects the encoding that is specified. Failure to provide proper tagging may result in corrupted input data that undergoes an unnecessary EBCDIC to ASCII conversion.

Binary files support

To tag a file as binary, use the following command:

```
chtag -b <path/to/binary/file>
```

To verify that the file has the binary tag, use the following command:

```
ls -T <path/to/binary/file>
```

You will get the following output:

```
b binary   T=off  path/to/binary/file
```

Enhanced ASCII support

Some applications take advantage of Enhanced ASCII support, which requires ASCII encoded text files to be tagged as ASCII text files. The IBM SDK for Node.js - z/OS runtime also supports reading files that are tagged as ASCII text files.

To tag a file as an ASCII text file, use the following command:

```
chtag -tc ISO8859-1 <path/to/ascii/text/file>
```

To verify that a file is tagged as an ASCII text file, use the following command:

```
ls -T <path/to/ascii/text/file>
```

You will get the following output:

```
t ISO8859-1   T=on  path/to/ascii/text/file
```

Usage

If you have some source files on an ASCII platform and you want to use them on z/OS, you can tag those files with the following steps:

1. Create a zip file of your source files on the ASCII platform.
2. Unzip the zip file on z/OS.
3. Tag all text files using the following command:

   ```
   chtag -tc ISO8859-1
   ```

4. Tag all binary files using the following command:

```
chtag -b
```

To copy files remotely from an ASCII platform to z/OS, you can use the **sftp** command, which converts every file from ASCII to EBCDIC as it copies. In this case, tagging is not necessary.

Related information

Using git for z/OS with GitHub

Chapter 14. Troubleshooting

This chapter describes some common issues that you might encounter with the IBM SDK for Node.js - z/OS runtime:

Integrated Cryptographic Service Facility not started - hang

If the node process hangs without any output check, you can run the following command to confirm whether `/dev/random` is functional:

```
cat /dev/random | od | head
```

If `/dev/random` is not available, ensure that Integrated Cryptographic Service Facility (ICSF) is started.

Installation error

During the compiler installation, you might see an error message that is similar to the following example:

```
An error occurred while receiving temporary dataset ... to dataset ... at ./unpack.pl line 147.
cannot receive ... to ... at ./unpack.pl line 61.
```

To resolve this problem, you should set the MSGCLASS parameter correctly. The default MSGCLASS parameter is “S”. To change this default, you need to set the COMPILER_INSTALL_HELD_MSGCLASS environment variable.

Suppose you have a "Hello world" program to run. You install the datasets under a HLQ “NODEUSR” and your user profile lists "H" as the default MSGCLASS. Then you can proceed to install and run Node.js with the following set of commands:

```
pax -rf <path_to_pax_archive_installable_file> -x pax
cd <path_to_pax_archive_installable_file>
export COMPILER_INSTALL_HLQ=NODEUSR
export COMPILER_INSTALL_HELD_MSGCLASS=H    #To set the COMPILER_INSTALL_HELD_MSGCLASS
environment variable
./unpack.pl
export PATH=`pwd`/bin:$PATH
./node
>console.log("Hello World")
```

If the installation completes successfully, you will see a “Hello World” pop up on your screen. If an error occurs during the installation, it is typically because the MSGCLASS parameter is not set correctly. You can query the MSGCLASS parameter on TSO with the following command:

```
LISTUSER NODEUSR TSO NORACF
```

If your user ID does not have the appropriate permission to view this information, contact your system administrator.

For more information about exporting the COMPILER_INSTALL_HELD_MSGCLASS environment variable, see Chapter 5, “Installing and configuring IBM SDK for Node.js - z/OS,” on page 9.

Forceful termination

If the node process needs to be killed forcefully, you can try sending an abort signal by using the following command:

```
kill -6 <process_id_of_node_process>
```

The process ID can be obtained using the **ps** command.

njsc++: The specified option "-qxplink" is not supported.

This indicates that you have built the npm with an older version of Node and the compiler options were cached into either the `$HOME/.npm` directory or the `$HOME/.node-gyp` directory. To resolve this issue, remove the cached directories as follows:

```
rm -rf $HOME/.npm
rm -rf $HOME/.node-gyp
```

Module CXXRT64 was not found.

To resolve this issue, install the LE PTFs as described in Chapter 5, “Installing and configuring IBM SDK for Node.js - z/OS,” on page 9.

unpack.pl expects SDSF

unpack.pl installs the C/C++ runtime and compiler that come with IBM SDK for Node.js - z/OS. The current unpack.pl installation requires the availability of SDSF on the system and proper access to use SDSF. The script uses SDSF to submit JES2 jobs and check status of jobs. To work around this requirement, try the SMP/E edition, which uses the SMP/E process for installation.

use: ./unpack.pl 3: FSUM7351 not found

The whole error message looks like this:

```
use: ./unpack.pl 3: FSUM7351 not found
use: ./unpack.pl 4: FSUM7351 not found
./unpack.pl 5: FSUM7332 syntax error: got (, expecting Newline_
```

The cause might be that the path `/usr/bin/env` is missing on your system. To set up this path for the **env** command, see Chapter 11, “Verifying the env command path,” on page 25.

Installing npm on a GitHub repository is not successful.

Use the following syntax to install npm and make sure you have Git version 2.14.4_zos_b09 installed:

```
npm install <github_repo_url>#branch
```

Error when running node --version

When you run `node --version`, you might see an error message that is similar to the following example:

```
msgget: EDC5133I No space left on device. (errno2=0x07050305)
CEE5207E The signal SIGABRT was received.
Ý1¨ + Done(131) node --version
 83951929      Abort   /ZOWE/node/node-v8.16.0-os390-s390x/bin/node
```

This indicates that you hit the message queue limit, and you need to set the __IPC_CLEANUP=1 environment variable for now to remove the existing message queues and run `node --version` again.

Message queues are left when the previous node processes do not end normally. If the node processes end normally, the message queues are cleaned and won't be left.

Diagnostic report for Node.js

The node-report npm module is available for the IBM SDK for Node.js - z/OS. It provides human-readable diagnostic data that is written to a file. A report is created automatically when an unhandled exception or fatal error event (such as an out of memory error) occurs. You can also trigger a report, for example by sending a USR2 signal to a Node.js process, or by an API call from a JavaScript application.

For more information, see the github repository for the node-report module. Further information and examples are available on the Monitoring & Diagnostics page and the Blogs page of the Node.js on IBM Developer.

Chapter 15. Known issues and limitations

IBM SDK for Node.js - z/OS has the following known issues or limitations:

- The current release requires an address space of at least 700 MB. Use the **`ulimit -A`** command to set or display the maximum size of the address space.
- The `Node.js fs.watch` API on subdirectories is not supported on z/OS.
- You cannot use IBM Monitoring and Diagnostic Tools – Interactive Diagnostic Data Explorer.
- Some C++11 and C11 features are not supported or have limited support:
  - Thread local storage - not supported
  - Thread specific locales - not supported
  - Atomics - limited support (non-lock-free atomics are not supported)

Chapter 16. Support

To find help about the IBM SDK for Node.js - z/OS, it is important to collect as much information as possible about your installation configuration.

To establish what version of the IBM SDK for Node.js - z/OS is in use, run the following command:

```
node --version
```

The version of Node.js is displayed.

To get more details about the exact build of the IBM SDK for Node.js - z/OS, run the following command:

```
node --help
```

The IBM SDK for Node.js - z/OS includes the Node.js `npm` utility for working with modules. To establish what version of `npm` is in use, run the following command:

```
npm --version
```

The version of the Node.js `npm` utility is displayed.

IBM support channels

IBM SDK for Node.js - z/OS offers the following methods of support:

- “Online self-help” on page 37
- “Getting IBM experts to solve your problem by opening a case” on page 37
- “Asking questions or raising comments to IBM on GitHub” on page 37
- “Asking questions or raising comments to IBM on Slack” on page 38

Paid world-class IBM support is available by opening a case. GitHub and Slack support is on a best-efforts basis.

Note: If you have a request for new features, instead of using one of the support methods above, submit an enhancement request in the Request for Enhancement (RFE) community.

Online self-help

Online documentation is available on this Knowledge Center. You can also download the PDF format documentation for offline use.

Getting IBM experts to solve your problem by opening a case

Paid IBM Subscription & Support (S&S) entitles you to world-class IBM support for IBM SDK for Node.js - z/OS. Get IBM support by opening a case. First obtain the SMP/E edition and purchase S&S; learn more at Paid S&S for IBM SDK for Node.js – z/OS. Once you have purchased S&S, open a case after logging in with your IBM customer ID to request support from IBM. If you need help on opening a case, see this IBM Support page.

Asking questions or raising comments to IBM on GitHub

GitHub support for IBM SDK for Node.js - z/OS is on a best-efforts basis. See the “Getting IBM experts to solve your problem by opening a case” on page 37 section to learn about purchasing world-class IBM support. IBM welcomes your feedback. You can ask questions or raise comments in GitHub.

Asking questions or raising comments to IBM on Slack

Slack support for IBM SDK for Node.js - z/OS is on a best-efforts basis. See the “Getting IBM experts to solve your problem by opening a case” on page 37 section to learn about purchasing world-class IBM support. IBM welcomes your feedback. You can ask questions or raise comments in Slack.

Join the public Slack Workspace by requesting an invitation on the Public Slack invitation request page.

Afterwards, you may join and post on the #nodejs-at-ibm channel.

Privacy Notice: This is a public Slack workspace. Personal information you provide to engage in the service, such as your name, email address, profile photo and messages you post may be visible to all members of the community, and so are effectively public. Any information only accessible to admins will be handled according to IBM’s Privacy Statement. By submitting the invitation request via the Public Slack invitation request page, you acknowledge that you have read and understand the above statement and the IBM Privacy Statement.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Corporation
J74/G4
555 Bailey Avenue
San Jose, CA 95141-1099
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES
THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND,
EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF NON-INFRINGEMENT,
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors.

Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this publication to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Product Number: 5655-DKN

SC27-9802-00 (2020-11-04)