Abstract

This Release Note contains important information related to Node.js 16 LTS.
# Table of Contents

**PREFACE** .................................................................................................................. 3  
**CHAPTER 1. REQUIRED INFRASTRUCTURE COMPONENT VERSIONS** ............................. 4  
**CHAPTER 2. FEATURES** .............................................................................................. 5  
  2.1. NEW AND CHANGED FEATURES ............................................................................ 5  
    2.1.1. Support for Node.js Runtime on FIPS-enabled Red Hat Enterprise Linux (RHEL) system 5  
    2.1.2. V8 JavaScript engine upgraded from v8.4 to v9.4 5  
    2.1.3. Timers Promises API 5  
    2.1.4. Updated Node.js metering labels for OpenShift 6  
  2.2. DEPRECATED FEATURES ....................................................................................... 6  
    2.2.1. Runtime deprecation of access to process.binding() 7  
    2.2.2. Old Node.js metering labels for OpenShift 7  
  2.3. TECHNOLOGY PREVIEW FEATURES ................................................................... 7  
    2.3.1. Web Crypto API 7  
    2.3.2. Web Streams API 7  
    2.3.3. ESM Loader Hooks API 8  
    2.3.4. New class for sharing raw data across worker threads 8  
  2.4. SUPPORTED ARCHITECTURES ............................................................................. 8  
**CHAPTER 3. RELEASE COMPONENTS** ........................................................................ 9  
**CHAPTER 4. FIXED ISSUES** ....................................................................................... 10  
**CHAPTER 5. KNOWN ISSUES** .................................................................................. 11  
**CHAPTER 6. KNOWN ISSUES AFFECTING REQUIRED INFRASTRUCTURE COMPONENTS** 12  
**CHAPTER 7. ADVISORIES RELATED TO THIS RELEASE** ........................................... 13
PREFACE

Date of release: 2021-12-21
CHAPTER 1. REQUIRED INFRASTRUCTURE COMPONENT VERSIONS

The following infrastructure components are required when using Red Hat build of Node.js. Except for components that are explicitly designated as supported, Red Hat does not provide support for these components.

<table>
<thead>
<tr>
<th>Component name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodeshift</td>
<td>2.1.1</td>
</tr>
<tr>
<td>npm 8</td>
<td>8.1.2</td>
</tr>
<tr>
<td>OpenShift Container Platform (OCP)[a]</td>
<td>3.11, 4.5</td>
</tr>
<tr>
<td>git</td>
<td>2.0 or later</td>
</tr>
<tr>
<td>oc command line tool</td>
<td>3.11 or later[b]</td>
</tr>
</tbody>
</table>

[a] OCP is supported by Red Hat

[b] The version of the oc CLI tool should correspond to the version of OCP that you are using.
CHAPTER 2. FEATURES

This section contains information about feature changes introduced in the Red Hat build of Node.js 16 release.

2.1. NEW AND CHANGED FEATURES

Node.js 16 LTS has the following new features and enhancements that Red Hat build of Node.js supports.

For detailed changes in Node.js 16 LTS, see the [upstream release notes](#) and [upstream documentation](#).

2.1.1. Support for Node.js Runtime on FIPS-enabled Red Hat Enterprise Linux (RHEL) system

Red Hat build of Node.js runs on a FIPS-enabled RHEL system and uses FIPS-certified libraries that are provided by RHEL.

2.1.2. V8 JavaScript engine upgraded from v8.4 to v9.4

The V8 JavaScript engine has been upgraded to v9.4.

The engine contains new features such as the ECMAScript RegExp Match Indices, which provides the start and end indices of captured strings in regular expression matches. The indices array is accessible by using the `.indices` property on match objects when the regular expression contains the `/d` flag. For example:

```javascript
> const matchObj = /(Java)(Script)/d.exec('JavaScript');
undefined
> matchObj.indices
[[ 0, 10 ], [ 0, 4 ], [ 4, 10 ], groups: undefined ]
> matchObj.indices[0];  // Match
[ 0, 10 ]
> matchObj.indices[1];  // First capture group
[ 0, 4 ]
> matchObj.indices[2];  // Second capture group
[ 4, 10 ]
```

The upgraded V8 JavaScript engine also provides the following JavaScript language features:

- `Array.prototype.at`
- Errors with cause
- `Object.hasOwn`

For more information about the changes that are available in the V8 JavaScript Engine, see the [V8 blog](#).

2.1.3. Timers Promises API

The Timers Promises API is a new stable feature in Node.js 16 LTS.

The Timers Promises API provides an alternative set of timer functions that return Promise objects. This API removes the need to use `util.promisify()`. For example:

```javascript
```
2.1.4. Updated Node.js metering labels for OpenShift

You can add metering labels to your Node.js pods and check Red Hat subscription details with the OpenShift Metering Operator.

NOTE

Do not add metering labels to any pods that an operator deploys and manages.

Since their introduction in the Node.js 14 LTS release, the Node.js metering labels for OpenShift have been modified.

Node.js 16 LTS uses the following updated metering labels:

- `com.company: Red_Hat`
- `rht.prod_name: "Red_Hat_Runtimes"`
- `rht.prod_ver: 2021-Q4`
- `rht.comp: Node.js`
- `rht.comp_ver: 16.x.x`
- `rht.subcomp:`
- `rht.subcomp_t: application`

See Metering documentation for more information.

For more information on labels, see Understanding how to update labels on nodes.

NOTE

- Do not add metering labels to any pods that an operator or a template deploys and manages.
- Replace the x.x in `rht.comp_ver` with the product version of Node.js that you are using in the deployment. For example, if the Node.js product version is 16.13.1, specify 16.13.1 in the label.
- Ensure that the `rht.subcomp` label is assigned a blank value for Node.js.

2.2. DEPRECATED FEATURES

The following features are deprecated in Red Hat build of Node.js 16.
2.2.1. Runtime deprecation of access to process.binding()

For a number of core modules, this release includes the runtime deprecation of access to the process.binding() function. For example: `process.binding('http_parser')`

2.2.2. Old Node.js metering labels for OpenShift

The Node.js metering labels for OpenShift, which were originally introduced in the Node.js 14 LTS release, are now superseded by a newer set of metering labels.

The following Node.js metering labels are deprecated in this release:

- `com.redhat.component-name: Node.js`
- `com.redhat.component-type: application`
- `com.redhat.component-version: 16.x.x`
- `com.redhat.product-name: "Red_Hat_Runtimes"`
- `com.redhat.product-version: 2021-Q4`

2.3. TECHNOLOGY PREVIEW FEATURES

The following features are available as Technology Preview features in the Node.js 16 LTS release.

2.3.1. Web Crypto API

The Web Crypto API is an implementation of the standard W3C Web Cryptography API. The Web Crypto API is accessible by using the `require('crypto').webcrypto` statement. For example:

```javascript
const { subtle } = require('crypto').webcrypto;

(async function() {
    const key = await subtle.generateKey(
        { name: 'HMAC',
        hash: 'SHA-256',
        length: 256
    }, true, ['sign', 'verify']);

    const digest = await subtle.sign(
        { name: 'HMAC' }, key, 'I love cupcakes');

})();
```

2.3.2. Web Streams API
The Web Streams API is an implementation of the WHATWG Streams Standard for handling streaming data. Because it is experimental, the Web Streams API is not exposed on the global object in this release. The Web Streams API is only accessible by using the new `stream/web` core module. For example:

```javascript
import { ReadableStream, WritableStream } from 'stream/web';
// Or from 'node:stream/web'
```

### 2.3.3. ESM Loader Hooks API

The ESM Loader Hooks API consolidates ECMAScript modules (ESM) loader hooks to represent the steps that are needed to facilitate future loader chaining:

1. **resolve**: `resolve (+ getFormat)`
2. **load**: `getFormat + getSource + transformSource`

For consistency, the `getGlobalPreloadCode` loader hook has been renamed `globalPreload`.

Any loader that exports one or more obsolete hooks will trigger a single deprecation warning that lists the errant hooks.

### 2.3.4. New class for sharing raw data across worker threads

The `Blob` class encapsulates immutable, raw data that can be safely shared across multiple worker threads.

`Blob` is a subclass of the `Buffer` class.

### 2.4. SUPPORTED ARCHITECTURES

Node.js builder images and RPM packages are available and supported for use with the following CPU architectures:

- AMD x86_64
- ARM64
- IBM Z (s390x) in the OpenShift environment
- IBM Power Systems (ppc64le) in the OpenShift environment
CHAPTER 3. RELEASE COMPONENTS

- Node.js 16 Builder Image for RHEL 8
- Node.js 16 Universal Base Image 8
- Node.js 16 Minimal Stand-alone Image for RHEL 8
- Node.js 16 Minimal Universal Base Image 8
CHAPTER 4. FIXED ISSUES

This release incorporates all of the fixed issues in the community release of Node.js 16 LTS.
CHAPTER 5. KNOWN ISSUES

There are no known issues affecting this release.
CHAPTER 6. KNOWN ISSUES AFFECTING REQUIRED INFRASTRUCTURE COMPONENTS

There are no known issues affecting infrastructure components required by this release.
CHAPTER 7. ADVISORIES RELATED TO THIS RELEASE

The following advisories have been issued to document enhancements, bugfixes, and CVE fixes included in this release.

- RHBA-2021:5260