IBM SDK for Node.js
Version 1.1

# *User Guide for IBM SDK for Node.js*

IBM

IBM SDK for Node.js
Version 1.1

# *User Guide for IBM SDK for Node.js*

**Copyright information**

This edition of the user guide applies to the IBM SDK for Node.js™, Version 1.1, and to all subsequent releases, modifications, and Service Refreshes, until otherwise indicated in new editions.

# Contents

# Chapter 1. Overview of the IBM SDK for Node.js™

The main features and capabilities of the IBM® SDK for Node.js™, Version 1.1.

The SDK is an extended implementation of the Node.js™ version 0.10.21 scripting language. In turn, Node.js™ is an implementation of the JavaScript programming language, modified for use on a server rather than within a client web browser.

Node.js™ is a framework that enables you to write server applications using the familiar syntax and structures of JavaScript. Node.js™ extends JavaScript by providing features such as:

- The creation, modification, and deletion of local (server-side) files.
- Event-driven behavior.
- Concurrency.
- A very small "footprint" on the server.

The SDK extends Node.js™ in two ways:

- IBM extensions to JavaScript and Node.js™ implementations, supported on various platforms, including IBM System p® (32-bit only).
- Provision of Reliability, Availability, and Serviceability (RAS) features.

# Chapter 2. Understanding the IBM SDK for Node.js™

Telling you more about the IBM SDK for Node.js™, Version 1.1.

The SDK is a server-side scripting language. It enables you to create tools and applications that run on your server, by writing scripts in the JavaScript language. An advantage of using JavaScript is that it is familiar to web developers. In addition, the "small, but fast and powerful" characteristics of JavaScript provide extra benefits for server development.

The majority of the SDK applications work on a simple principle: wait for a request to come in to the server, process the request as quickly as possible, then return the resulting response back to the client.

An IBM SDK for Node.js™ implementation works by running JavaScript programs. The SDK itself is not written in JavaScript; it is a software component that executes JavaScript on the server. A simple SDK application is run by starting the SDK executable on the command line, and providing it with the name of an application to run:

```
$> node ExampleNodeApplication.js
```

One of the first thing most applications do is to wait for an incoming service request. The request is received through a specific port number, where the port number is provided in the SDK script.

Next, the request is examined to understand exactly what is required. The SDK passes the request to the corresponding request handler, or returns an error message if the request is not understood.

If your SDK application is running and listening on a port, no other application can be listening on the same port at the same time. This means that your application must be able to respond to more than one request at a time. If a request takes a long time to process - several seconds, for example - all other requests would have to wait until the original request completes. To prevent this problem, SDK applications use an event-driven model.

The event-driven model enables an SDK application to initiate a task in response to a request, then go and attend to something else such as a new request. When the initiated task completes, an event occurs and your application is interrupted from whatever it is doing, to continue processing the original request.

Resilience within the SDK is provided by an isolation of processes. The SDK is fundamentally single-threaded. This means that race conditions for access to resources cannot occur. It also means that the SDK has a simple robustness model, often referred to as Crash-only software. In this model, emphasis is placed on graceful failure and restart of the application, rather than trying to identify and handle all possible error conditions. The result is much greater overall stability, because other processes are much less likely to be affected by problems in the SDK system. It also means that if a problem does occur, an application restart is much easier to achieve.

# Chapter 3. Planning

Ensure that you meet the system requirements for installing and running the IBM SDK for Node.js™, Version 1.1.

The IBM SDK for Node.js™ runs on the following platforms:

- System p Linux
- System x® platforms (Linux and Windows)
- OS X

*Table 1. Environments tested*

| Platform | 32-bit | AMD64/EM64T 64-bit |
|---|---|---|
| RedHat Enterprise Linux 6.3 | GA (IA-32 and POWER) | GA |
| Ubuntu 10.04 LTS | GA (IA-32) | GA |
| Windows 7 | GA (IA-32) | GA |
| OS X 10.8 | GA (IA-32) | GA |

To use the InstallAnywhere package, you must have a Java™ technology runtime environment version 1.6 or higher already installed.

# Chapter 4. Installing the IBM SDK for Node.js™

You can install the SDK by downloading the correct package for your platform from IBM developerWorks.

The IBM SDK for Node.js™ is provided as an installation package downloaded from IBM developerWorks.

The overall process for installing the SDK is as follows:

- Check that your system meets or exceeds the pre-requisite requirements for the SDK.
- Locate the correct package for your system on the IBM developerWorks site.
- Accept the download license.
- Download the package to a convenient location on your system. The package is supplied as an archive file.
- Extract the contents of the archive file into your chosen installation directory.
- Perform a “first run” of the SDK. This confirms that your installation is working correctly. It also enables you to accept the executable license for the SDK.

## Installing from an InstallAnywhere package

These packages provide an interactive program that guides you through the installation options. You can run the program as a graphical user interface, or from a system console.

### Before you begin

Ensure that your system has the required pre-requisites installed. These are detailed in the Chapter 3, “Planning,” on page 5 information. If you do not have the pre-requisites installed, you might see error messages.

### About this task

The InstallAnywhere packages for the different platforms are recognizable by the file extension.

- Linux packages for all platforms have a `.bin` file extension.
- OS X packages for all platforms have a `.zip` file extension.
- Windows packages for all platforms have a `.exe` file extension.

### Procedure

- To install the package in an interactive way on a Linux or Windows platform, complete an attended installation.
- To install the package on a OS X platform, use an attended installation on OS X.
- To install the package on a Linux or Windows platform, without any additional user interaction, complete an unattended installation. You might choose this option if you want to install many systems.

### Results

The product is installed.

**Note:** Do not interrupt the installation process, for example by pressing Ctrl+C. If you interrupt the process, you might have to reinstall the product. For more information, see "Interrupted installation" on page 11.

## Completing an attended installation

Install the product from an InstallAnywhere package, in an interactive way.

### Before you begin

There are no prerequisite conditions before you begin the installation.

### Procedure

1. Download the installation package file to a temporary directory.
2. Change to the temporary directory.
3. Ensure that the installation package has permission to run. Do this by running the command:
   ```
   chmod +x ./package.bin
   ```
4. Start the installation process by typing `./`*`package`*`.bin` at a shell prompt, where *package* is the name of the package that you are installing.
5. The installation process begins with an introduction screen. Read the details, then press **Enter** to continue.
6. You must accept the International License Agreement to proceed with the installation. To accept the terms of the license, type **1** then **Enter**.
7. You are asked to choose the target folder for the installation. If you do not want to install into the default folder, enter the full path name for an alternative folder. After choosing the installation folder, type **Y** to confirm that the folder is correct.
8. Optional: You are invited to specify a link location. This location holds a link to the uninstall utility included with the IBM SDK for Node.js™. If you wish to create this link, specify a location that is included in your system **`PATH`**.
9. You are asked to review the choices that you made. If your choices are correct, type **Enter** to continue.
10. A final screen confirms that the installation process is ready to begin. Type **Enter** to install the SDK. While installation takes place, a simple "fuel gauge" shows you the estimated progress.
11. When the installation process is complete, a confirmation message appears. Type **Enter** to exit the installer.

## Completing an attended installation on OS X

Install the product from an InstallAnywhere package, in an interactive way.

### Before you begin

There are no prerequisite conditions before you begin the installation.

### Procedure

1. Download the installation `.zip` file to your desktop.
2. Double-click the installation `.zip` file. The result is a new OS X installation package on your desktop.
3. Double-click the new installation package.

4. The installation process begins with an introduction screen. Read the details, then press **Enter** to continue.
5. You must accept the International License Agreement to proceed with the installation. To accept the terms of the license, type **1** then **Enter**.
6. Continue through the installation screens, making selections appropriate for your configuration.
7. You are asked to review the choices that you made. If your choices are correct, type **Enter** to continue.
8. A final screen confirms that the installation process is ready to begin. Type **Enter** to install the SDK. While installation takes place, a simple "fuel gauge" shows you the estimated progress.
9. When the installation process is complete, a confirmation message appears. Type **Enter** to exit the installer.

## Completing an unattended installation

If you have more than one system to install, and you already know the installation options that you want to use, you might want to use the unattended installation process.

### Before you begin

To complete an unattended installation, you must have a response file that is created previously by an attended installation. Use the response file to complete further installations without any additional user interaction.

**Note:** It is not possible to create a response file without also performing an actual installation. The response file is effectively a record of the options used to perform the installation. The response file can be used for later, additional installations, using the same options.

**Note:** More information about installer command line options is available here.

### Procedure

1. Create a response file by first completing an attended installation.

   Use the command line and append the `-r` option to the attended installation command, specifying the full path to the response file. For example:

   ```
   ./package -r /path/installer.properties
   ```

   Example response file contents:

   ```
   INSTALLER_UI=silent
   USER_INSTALL_DIR=/my_directory
   ```

   In this example, `/my_directory` is the target installation directory that you chose for the IBM SDK for Node.js™.
2. Edit the response file to include a statement that confirms you accept the license terms. Do this by adding the following line to the response file:

   ```
   LICENSE_ACCEPTED=true
   ```
3. Optional: Edit the response file to change installation options.

   **Note:** If you want different installation options, create more than one response file, each with a unique name in the format: `myfile.properties`

4. Optional: Generate a log file. An unattended installation produces no status messages at the end of the installation process. To generate a log file that contains the status of the installation, complete the following steps:
   a. Set the system properties by using the following command.
      ```
      export _JAVA_OPTIONS="-Dlax.debug.level=3 -Dlax.debug.all=true"
      ```
   b. Set the following environment variable to send the log output to the console.
      ```
      export LAX_DEBUG=1
      ```
5. Start an unattended installation by running the package installer with the **-i** silent option, and the **-f** option to specify the response file. For example:
   ```
   ./package -i silent -f /path/installer.properties 1>console.txt 2>&1
   ```

   Or:
   ```
   ./package -i silent -f /path/myfile.properties 1>console.txt 2>&1
   ```

   You can use a fully qualified path or relative path to the properties file. In these examples, the string `1>console.txt 2>&1` redirects installation process information from the `stderr` and `stdout` streams to the `console.txt` log file in the current directory. Review this log file if you think there was a problem with the installation.

   **Note:** If your installation directory contains multiple response files, the default response file, `installer.properties` is used.

## Sample response file for installation on Linux platforms

An example response file for an unattended installation on Linux platforms.

```
Tue Nov 26 20:47:31 GMT 2013
# Replay feature output
# ---------------------
# This file was built by the Replay feature of InstallAnywhere.
# It contains variables that were set by Panels, Consoles or Custom Code.



#Indicate whether the license agreement been accepted
#----------------------------------------------------
LICENSE_ACCEPTED=TRUE

#Choose Install Folder
#---------------------
USER_INSTALL_DIR=/home/auser/node

#Choose Link Location
#--------------------
USER_SHORTCUTS=/home/auser

#Install
#-------
-fileOverwrite_/home/auser/node/_node_installation/Change node Installation.lax=Yes
```

## Sample response file for installation on Windows platforms

An example response file for an unattended installation on Windows platforms.

```
# Tue Nov 26 16:11:32 EST 2013
# Replay feature output
# ---------------------
# This file was built by the Replay feature of InstallAnywhere.
# It contains variables that were set by Panels, Consoles or Custom Code.
```

```
#Indicate whether the license agreement been accepted
#----------------------------------------------------
LICENSE_ACCEPTED=TRUE

#Choose Install Folder
#---------------------
USER_INSTALL_DIR=C:\\IBM\\node

#Choose Shortcut Folder
#----------------------
USER_SHORTCUTS=Do Not Install

#Install
#-------
-fileOverwrite_C\:\\IBM\\node\\_node_installation\\Change\ node\ Installation.lax=Yes
-fileOverwrite_C\:\\IBM\\node\\_node_installation\\resource\\iawin64_x64.dll=Yes
-fileOverwrite_C\:\\IBM\\node\\_node_installation\\resource\\iawin32.dll=Yes
-fileOverwrite_C\:\\IBM\\node\\_node_installation\\resource\\remove.exe=Yes
```

## Interrupted installation

If the package installer is unexpectedly stopped during installation, for example if you press **`Ctrl+C`**, the installation is corrupted and you cannot uninstall or reinstall the product. If you try to uninstall or reinstall you might see the message `Fatal Application Error` .

### About this task

To solve this problem, delete files and reinstall, as described in the following steps.

### Procedure

1. Delete the `.com.zerog.registry.xml` registry file. The file location is dependent on your platform:

| Platform | Registry file location |
|---|---|
| **Linux** | `$HOME/.com.zerog.registry.xml` |
| **Windows** | `C:\Program Files\Zero G Registry\.com.zerog.registry.xml` |
| **OS X** | `/Library/Preferences/.com.zerog.registry.xml` |
| **Other UNIX platforms** | `/var/.com.zerog.registry.xml` |

2. Delete the directory containing the SDK installation, if it was created.
3. Run the installation program again.

## Verifying your installation

How to check that your installation of the IBM SDK for Node.js™, Version 1.1 works correctly.

### Before you begin

This task assumes that the IBM SDK for Node.js™ is installed, and that you can run the package from a command line interface on your system.

### Procedure

1. Run the command `node --version` as described in the Chapter 9, “Support,” on page 23 topic to confirm that correct version of the IBM SDK for Node.js™ is installed.
2. Run the command `npm --version` as described in the Chapter 9, “Support,” on page 23 topic to confirm that correct version of the IBM SDK for Node.js™ Package Manager utility is installed.
3. Create a text file that is called `example.js`, containing the following SDK application code:
```
var http = require("http");

http.createServer(function(request, response) {
    response.writeHead(200, {"Content-Type": "text/plain"});
    response.write("Hello from the IBM SDK for Node.js");
    response.end();
}).listen(8080);
```
4. Start the SDK application by running the following command:
```
node example.js
```
5. Open a web browser and go to the URL: `http://localhost:8080/`

### Results

Your web browser presents a simple web page, with the following content:

```
Hello from the IBM SDK for Node.js
```

### What to do next

If you do not get the expected results, follow the instructions that are provided in “Problem determination” on page 21.

## Uninstalling the IBM SDK for Node.js™

Follow these steps to remove the IBM SDK for Node.js™.

### About this task

You uninstall the product by using a command, or by running the installation program again, as described in the following steps.

**Note:** If you installed IBM SDK for Node.js™ using silent mode, an uninstallation also proceeds in silent mode. For more information, see “Completing an unattended installation” on page 9.

**Note:** On the Linux platform, do not attempt to uninstall the product by using the `rpm -e` command, because the end result is a corruption of the installed package.

### Procedure

- Uninstall manually by using the **`uninstall`** command.
  1. Change to the directory that contains the IBM SDK for Node.js™ installation.
  2. Start the uninstall process by entering the following command:
     ```
     ./_uninstall/uninstall
     ```

- If you cannot locate the uninstall program easily, as an alternative you can run another attended installation. The installation program detects that the product is already installed, then gives you the opportunity to uninstall the previous installation.

## Results

The product is uninstalled.

**Note:** On the Linux platform, if you installed the product on a system that does not have the rpm-build tool, files such as `<package>.spec` remain after uninstallation. The rpm-build tool is a requirement for successful installation. For more information, see "Installing from an InstallAnywhere package" on page 7.

# Chapter 5. Configuring

Modifying the behavior of your IBM SDK for Node.js™, Version 1.1 installation.

## About this task

The SDK does not have any specific configuration options that apply to all running applications. To modify the behavior of a specific application, you should use a corresponding dedicated configuration file for that application.

For example, if you have an SDK application called `MyExample.js`, you might create a configuration file called `MyExample.config`. The file contains a series of properties that can be inspected by the main `MyExample.js`. Each property might modify the behavior of the application in some way, such as controlling which port is used to listen for incoming requests.

# Chapter 6. Debugging

Debugging your IBM SDK for Node.js™ applications.

In general, the process for debugging IBM SDK for Node.js™ applications is much like that for debugging applications written in other server scripting languages, for example using 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, and so on.

Most general purpose tools for debugging Node.js™ applications should work correctly with your SDK applications.

# Chapter 7. Running

This information tells you how to run an IBM SDK for Node.js™ application.

All SDK applications are run in the same way: invoke the SDK runtime binary and supply the name of the required application. For example:

```
node myapplication.js
```

If you want the SDK runtime binary to be invoked when your system starts, the easiest way to do this is by creating a startup (shell) script that is called during system startup.

Additional arguments can be supplied to your application by appending them to the command line invocation. For example:

```
node applicationexample.js parameter1 2 value3
```

Each of the arguments is available to your application using the `process.argv` array. The first element in the array (`process.argv[0]`) is the name of the SDK runtime binary: `node`. The second element in the array (`process.argv[1]`) is the name of the SDK application script. In this example, `process.argv[1]` has the value `applicationexample.js`. Any remaining elements in the array correspond to other arguments supplied on the command line.

# Chapter 8. Troubleshooting

Use the information in this section to help you determine the cause of problems, and run diagnostic tools.

A good starting point for *general* Node.js™ troubleshooting help is Joyent's Node.js™ GitHub wiki page. For more specific help about IBM SDK for Node.js™, post questions in the IBM developerWorks community.

## Problem determination

What to do if your IBM SDK for Node.js™, Version 1.1 installation does not work correctly.

### About this task

If you think the SDK is not working correctly, there are several actions you can take to determine the cause or location of the problem. After each action, try running your application again, or running a simpler test application, to test whether the problem has been fixed. If none of the suggested actions solve the problem, refer to the Support information for further assistance.

### Procedure

- Check if there are any missing dependencies. If your application requires any third party Node.js™ modules, ensure they are installed.
- Check that the system `PATH` variable includes the directory location of the SDK executable. If the `PATH` variable does not contain the installation directory, your system cannot find the executable, and your application cannot run.
- Look for and respond to error messages when running your application.
- (System p Linux, System x Linux and OS X only.) Try debugging while running your application in "background mode". Do this by running your application with a command line similar to:

  ```
  node myapplication.js &
  ```

  This command puts the application into background mode. To run the application in "foreground" mode, omit the "`&`". Any error messages or warnings can be seen clearly while your application is running.

## Diagnostic tools

Using the IBM Monitoring and Diagnostic Tools for Java - Interactive Diagnostic Data Explorer tool, with Node.js™ extension, to diagnose problems with the IBM SDK for Node.js™, Version 1.1.

The IBM Monitoring and Diagnostic Tools for Java - Interactive Diagnostic Data Explorer (IDDE) has the ability to analyze core files generated by your IBM SDK for Node.js™ application.

## Generating core files on System p Linux, System x Linux, and OS X

A core file is normally generated when a running SDK application receives a signal from the operating system, or when a problem condition arises that prevents the application from continuing to run.

A core dump file is only created if the system permits it. To check if core file creation is permitted, enter the command:

```
ulimit -c
```

If the value reported is zero (0), then no core files can be created because the maximum size permitted is 0 bytes. To enable core file creation, set a larger value for the file size limit. For example, to allow files up to 10,000 blocks in size, with each block holding 1,024 bytes, use the command:

```
ulimit -c 10000
```

You can also specify an unlimited size for the core file, by using the command:

```
ulimit -c unlimited
```

To force the generation of a core file from a running application, you send a suitable signal to the corresponding process. The most appropriate signals you might send include:

- SIGABRT
- SIGSEGV

The `SIGABRT` signal can be generated from the keyboard by typing the sequence `CTRL+\`. It can also be sent directly to the running process using the command:

```
kill -ABRT <process_id>
```

The `SIGSEGV` signal can be sent directly to the running process using the command:

```
kill -SEGV <process_id>
```

## Using the core file with IBM Monitoring and Diagnostic Tools for Java - Interactive Diagnostic Data Explorer

When you have obtained the core file, you should use IDDE with the Node.js™ extension, to analyze the information it contains.

# Chapter 9. Support

To find help about the IBM SDK for Node.js™, Version 1.1, it is important to collect as much information as possible about your installation configuration.

To establish what version of the SDK is in use, run the command:

```
node --version
```

The result is similar to:

```
v0.10.21
```

To get more details about the exact build of the SDK, run the command:

```
node --help
```

The result is similar to:

```
v0.10.21-IBMBuild-201311141356
(c) Copyright IBM Corp. 2013 All Rights Reserved
(c) Copyright Joyent, Inc. and other Node contributors. 2013 All Rights Reserved
```

The IBM SDK for Node.js™ includes the Node.js™ **npm** utility for working with modules. To establish what version of **npm** is in use, run the command:

```
npm --version
```

The result is similar to:

```
1.3.11
```

# 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 grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1758
U.S.A.

For license inquiries regarding double-byte character set (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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 information 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.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

JIMMAIL@uk.ibm.com
[Hursley Java Technology Center (JTC) contact]

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided “AS IS”, without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## Privacy Policy Considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see: (i) IBM’s Privacy Policy at http://www.ibm.com/privacy ; (ii) IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details (in particular, the section entitled “Cookies, Web Beacons and Other Technologies”); and (iii) the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol ($^{®}$ or $^{™}$), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Apple, Macintosh, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Node.js is an official trademark of Joyent. The IBM SDK for Node.js is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

Other company, product and service names may be trademarks or service marks of others.

Printed in USA