# Oracle® NoSQL Database

## Quick Start to KVLocal

Release 23.3
F51345-11
March 2024

Oracle NoSQL Database Quick Start to KVLocal, Release 23.3

F51345-11

Copyright © 2022, 2023, Oracle and/or its affiliates.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

# Contents

# Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

**Access to Oracle Support**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

# Introduction

KVLocal is an embedded Oracle NoSQL Database that can be embedded in data-rich applications to process and present live data from large datasets. KVLocal provides a single-node store that is not replicated.

It runs as a separate child process in the application JVM and requires minimal administration. KVLocal is a robust database and handles failures efficiently. You can start and stop KVLocal using APIs.

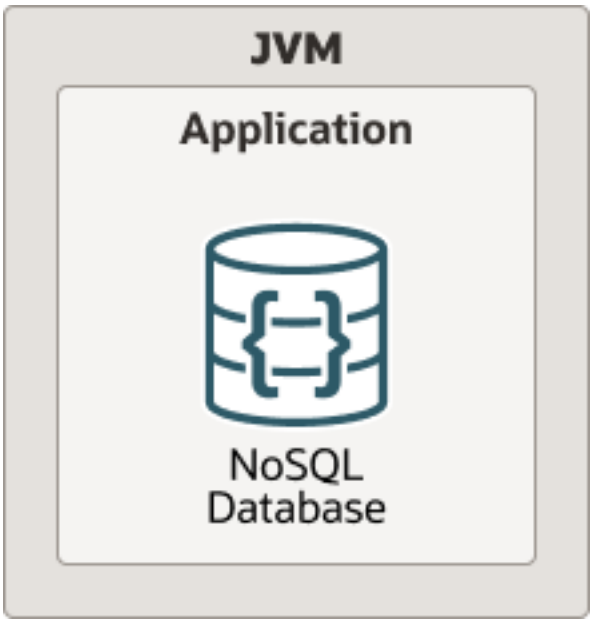
KVLocal provides the ability to run a single instance of Oracle NoSQL Database by including `kvstore.jar` in the application's classpath, starting a JVM, and calling an API to initialize the database. KVLocal is accessed using the Java Direct Driver API.

KVLocal to use either TCP/IP sockets or Unix domain sockets for communication between the client APIs and KVLocal. If you configure KVLocal to use TCP/IP sockets, it runs by default in secure mode, and you can configure it explicitly to run non-securely. If you configure KVLocal to use Unix domain sockets, it is inherently secure because it is not accessible over the network. The security depends on file protections on the socket files used for communication.

> **Note:**
>
> There is no separate support for secure or non-secure KVLocal when using Unix domain sockets.

KVLocal store has an Administrative (Admin) service that helps to configure, deploy, monitor, and change KVLocal store components.

> **Note:**
>
> One application (in other words one JVM) can run *only* one KVLocal store.




This Quick Start guide demonstrates how to perform the following tasks:

- Embedding KVLocal in your Java Application
- Starting Administration CLI
- Running SQL Shell with KVLocal
- Backing Up KVLocal Store

The following topics provide more detailed information:

- KVLocal Diagnostic Utility
- KVLocal FAQs

# Java Development Kit (JDK) Requirements

Unix domain sockets require Java 16 or later.

# Embedding KVLocal in your Java Application

Make sure that the `kvstore.jar` file is included in your application's CLASSPATH. The `kvstore.jar` file is available as part of Oracle NoSQL Database software. You can download it from Oracle Technology Network.

### Create `KVLocalConfig` Object

Create an object of the `KVLocalConfig` class. This class represents the configuration parameters used by KVLocal. `KVLocalConfig` class contains two builders: `InetBuilder` and `UnixDomainBuilder`. The `InetBuilder` is the builder to construct a `KVLocalConfig` instance using TCP/IP sockets, and the `UnixDomainBuilder` is the builder to construct a `KVLocalConfig` instance using Unix domain sockets.

KVLocal supports the following configuration parameters.

| Parameter | Description |
|---|---|
| `StoreName`<br>If not set, defaults to `kvstore`.<br>Example:<br>• `InetBuilder.setStoreName("nosql")`<br>• `UnixDomainBuilder.setStoreName("nosql")` | Name of the KVLocal instance. |
| `HostName`<br>(For `InetBuilder`) If not set, defaults to `localhost`.<br>(For `UnixDomainBuilder`) Defaults to `unix_domain:<KVROOT>/sockets/sock` and is non-modifiable.<br>Example:<br>`InetBuilder.setHostName("nosql.app.com");` | The network address to use to reach the host when using `InetBuilder` for KVLocal instance. You cannot set this parameter when using `UnixDomainBuilder`. |
| `Port`<br>If not set, defaults to `5000`.<br>Example: `InetBuilder.setPort(5500);` | The TCP/IP port on which the client APIs communicate with the KVLocal instance. You can change this parameter only if you are using `InetBuilder`. When using Unix domain sockets, it does not represent a TCP/IP port. However, the default value of `5000` must be specified with the `-port` flag when connecting to the Admin CLI. |
| `enableSecurity`<br>(For `InetBuilder`) If not set, defaults to `true`.<br>(For `UnixDomainBuilder`) Defaults to `false` and is non-modifiable.<br>Example:<br>`InetBuilder.enableSecurity(false);` | Determines whether it is a secure or non-secure KVLocal store. For `UnixDomainBuilder`, the value of this parameter is always false, and you cannot modify it. The reason being the Unix domain sockets configurations are inherently secure for communication. |

| Parameter | Description |
| --- | --- |
| `StorageSize`<br>If not set, defaults to `10`.<br>Example: To set the storage size to `80` GB<br>• `InetBuilder.setStorageSize(80);`<br>• `UnixDomainBuilder.setStorageSize(80);` | The maximum amount of available disk space in GB for a KVLocal database. If the KVLocal database exceeds its disk usage threshold value (excluding 5 GB of free space), KVLocal suspends all write operations on the host system until you remove sufficient data to satisfy the threshold requirement. If you set storage directory size to 0, KVLocal opportunistically uses all the available space, excluding 5 GB free disk space. |
| `MemoryMB`<br>If not set, defaults to `8192`. The minimum value for this parameter is `48`.<br>Example: To set the memory size to `85` MB<br>• `InetBuilder.setMemoryMB(85);`<br>• `UnixDomainBuilder.setMemoryMB(85);` | AAmount of memory size in MB of the Java heap used to run the embedded KVLocal database. The Java heap size here refers to the Java heap size of the child process and not of the JVM process that is running the application.<br>**Note:**<br>Ensure that the host machine has enough memory available to create the JVM with the requested heap size. |


- KVLocalConfig instance using TCP/IP sockets
- KVLocalConfig instance using Unix domain sockets


## KVLocalConfig instance using TCP/IP sockets

- For a secure KVLocal

```
import oracle.kv.KVLocalConfig;
/* Create a KVLocalConfig object with TCP sockets */
KVLocalConfig config = new KVLocalConfig.InetBuilder("rootDir")
                         .build();
```

- For a non-secure KVLocal

```
import oracle.kv.KVLocalConfig;
/* Create a KVLocalConfig object with port number 6000 and security not
enabled */
KVLocalConfig config = new kvlocalConfig.InetBuilder("rootDir")
                         .setPort(6000)
```

```
                        .enableSecurity(false)
                        .build();
```

*rootDir* is where the `kvroot` directory should reside. It refers to the absolute path to the directory where Oracle NoSQL Database data is stored, for example, `/home/kvstore`.

If you enable security for KVLocal, a security file (`user.security`) gets generated under the `kvroot` directory when you start KVLocal. If a kvstore already exists (in other words, the `kvroot` directory already exists), KVLocal uses the security file present in the existing `kvroot` directory to secure the kvstore.

## KVLocalConfig instance using Unix domain sockets

```
import oracle.kv.KVLocalConfig;
/* Create a KVLocalConfig object with Unix domain sockets */
KVLocalConfig config = new KVLocalConfig.UnixDomainBuilder("rootDir")
                        .build();
```

*rootDir* is where the `kvroot` directory should reside. It refers to the absolute path to the directory where Oracle NoSQL Database data is stored, for example, `/home/kvstore`.

When you use Unix domain sockets, socket files get created under the kvroot directory. These socket files are used for communication between the server (KVLocal) and the client (user application). For example, if your kvroot directory is /home/kvroot, the full path to one such Unix domain socket file is: /home/kvroot/sockets/sock-5000

### Invoke KVLocal Start and Stop APIs

The KVLocal class provides APIs to start or stop the embedded NoSQL database instance. In your application, invoke the start or stop APIs and get the handle to kvstore.

> **Note:**
>
> One application (in other words, one JVM) can manage *only* one KVLocal store. The second KVLocal instantiation within the same JVM throws the exception "The KVLocal has already been initialized.".

The KVLocal class provides following APIs to start or stop KVLocal instance and get the handle to the kvstore.

| Method Name | Description |
|---|---|
| `start(KVLocalConfig config)` | Starts a KVLocal instance. |
| `startExistingStore(String rootDir)` | Starts KVLocal instance from the existing root directory. |
| `stop()` | Stops the running instance of KVLocal. |

| Method Name | Description |
| --- | --- |
| `KVStore getStore()` | Gets a store handle to a running KVLocal instance.<br>A new store handle gets created as needed on the first call to this method. All subsequent calls return the existing store handle. If the existing store handle is cleaned up by invocation of the `KVLocal.closeStore()` method, the next call to this method creates a new store handle again. |

**Note:**

The application must invoke `KVLocal.closeStore()` method when it is done accessing the store to free up resources associated with the store handle. DO NOT invoke `KVStore.close()` method, because it does not free up all the resources associated with the store handle and makes the store handle non-functional.

- Using TCP/IP sockets
- Using Unix domain sockets

## Using TCP/IP sockets

```
import oracle.kv.KVLocalConfig;
import oracle.kv.KVLocal;
/* Create a KVLocal object and pass the KVLocal configuration parameters to
the object */
KVLocalConfig config = new KVLocalConfig.InetBuilder("rootDir")
                       .build();
/* Start KVLocal*/
KVLocal local = KVLocal.start(config);
/* Get a handle to kvstore */
```

```
KVStore storeHandle = local.getStore();
/* Use existing key/value APIs to write to kvstore */
storeHandle.put(Key,Value);
ValueVersion valueVersion = storeHandle.get(Key.createKey(key));
/* Close kvstore */
KVLocal.closeStore();
/* Stop kvstore */
local.stop();
```

*rootDir* is where the kvroot directory should reside. It refers to the absolute path to the directory where Oracle NoSQL Database data is stored, for example, `/home/kvstore`.

## Using Unix domain sockets

```
import oracle.kv.KVLocalConfig;
import oracle.kv.KVLocal;
/* Create a KVLocal object and pass the KVLocal configuration
parameters to the object */
KVLocalConfig config = new KVLocalConfig.UnixDomainBuilder("rootDir")
                        .build();
/* Start KVLocal*/
KVLocal local = KVLocal.start(config);
/* Get a handle to kvstore */
KVStore storeHandle = local.getStore();
/* Use existing key/value APIs to write to kvstore */
storeHandle.put(Key,Value);
ValueVersion valueVersion = storeHandle.get(Key.createKey(key));
/* Close kvstore */
KVLocal.closeStore();
/* Stop kvstore */
local.stop();
```

*rootDir* is where the kvroot directory should reside. It refers to the absolute path to the directory where Oracle NoSQL Database data is stored, for example, `/home/kvstore`.

# Starting Administration CLI

The `runadmin` utility provides the Admin command-line interface (CLI). When a KVLocal instance is running, an admin client can connect to the KVLocal store using the following command:

```
java -jar <KVHOME>/lib/kvstore.jar runadmin -port 5000 -host localhost
```

Where, *<KVHOME>* refers to the directory where Oracle NoSQL Database package files reside.

> **Note:**
>
> When using Unix domain socket to connect to KVLocal, specify the hostname in this form: `unix_domain:KVROOT/sockets/sock`. For example, if the KVROOT directory is `/disk1/kvroot`, run the following command to start the Administration CLI:
>
> ```
> java -jar <KVHOME> /lib/kvstore.jar runadmin -port 5000 -host
> unix_domain:/disk1/kvroot/sockets/sock
> ```

### Setting JE Parameters for KVLocal

JE is the storage engine for KVLocal. KVLocal is a wrapper around JE and emulates a single replication node from the NoSQL Database Cluster. Hence, one interacts with KVLocal as though it were a single node NoSQL Database.

You can set the JE parameters for a KVLocal instance using the Replication Node (RN) parameter, `configProperties=<String>`. It contains property settings for the underlying BDB JE subsystem. Its format is `property=value;property=value...`.

Before setting the JE parameters, you first need to determine the current settings in the KVLocal instance. To determine the current settings of `configProperties`, enter the Admin CLI `show parameters -service name` command as follows:

```
kv-> show parameters -service rg1-rn1;
...
...
configProperties=je.cleaner.threads 1;
je.rep.insufficientReplicasTimeout 100 ms;
je.env.runEraser true;
je.erase.deletedDatabases true;
je.erase.extinctRecords true;
je.erase.period 6 days;
je.env.runBackup false;
je.backup.schedule 0 8 * * *;
je.backup.copyClass
oracle.nosql.objectstorage.backup.BackupObjectStorageCopy;
je.backup.copyConfig /var/lib/andc/config/params/backup.copy.properties;
je.backup.locationClass
```

```
oracle.nosql.objectstorage.backup.BackupObjectStorageLocation;
je.backup.locationConfig /var/lib/andc/config/params/
backup.location.properties;
je.rep.electionsOpenTimeout=2 s;
je.rep.electionsReadTimeout=2 s;
je.rep.feederTimeout=3 s;
je.rep.heartbeatInterval=500;
je.rep.replicaTimeout=3 s;
je.rep.repstreamOpenTimeout=2 s;...
...
```

To set or modify a JE parameter, enter the Admin CLI `plan change-parameters -service <id>` command.

For example, if your `configProperties` for Replication Node is set to:

```
"configProperties=je.cleaner.minUtilization=40;">
```

And you want to add new settings for `configProperties`, you need to issue the following command:

```
kv-> plan change-parameters -all-rns -params \
     "configProperties=je.cleaner.minUtilization=50;\
     je.env.runVerifier=false;">
```

### Changing KVLocal JVM Memory Parameters

You can change the JVM memory settings for KVLocal using `javaRnParamsOverride=<String>` parameter. It accepts a string that is added to the command line when the child process is started. This parameter is intended for specifying miscellaneous JVM properties that cannot be specified using other RN parameters. If the string is not a valid sequence of tokens for the JVM command line, the Replication Node process fails to start.

To determine the current settings of `javaRnParamsOverride` and `configProperties` parameters, enter the Admin CLI `show parameters -service name` command as follows:

```
kv-> show parameters -service rg1-rn1;
```

For example, if you want to increase the JVM memory, use the `plan change-parameters` command from the Admin CLI, as follows:

```
kv-> plan change-parameters -wait -all-admins  -params \
javaRnParamsOverride="-Xms2048m -Xmx2048m-XX:ParallelGCThreads=4"
```

# Running SQL Shell with KVLocal

To connect SQL shell to a KVLocal instance, run the following command:

```
java -jar <KVHOME>/lib/sql.jar -helper-hosts <host:port> -store <storeName>
```

Where, <*KVHOME*> refers to the directory where Oracle NoSQL Database package files reside.

> **Note:**
>
> When using Unix domain socket to connect to KVLocal, specify the hostname in this form: `unix_domain:KVROOT/sockets/sock`. For example, if the KVROOT directory is `/disk1/kvroot`, run the following command to start the Administration CLI:
>
> ```
> java -jar <KVHOME>/lib/kvstore.jar -helper-hosts unix_domain:/disk1/
> kvroot/sockets/sock:5000 -store kvstore
> ```

For a complete list of utility commands accessed through "`java -jar <KVHOME>/lib/sql.jar <command>`", see Shell Utility Commands.

# Backing Up KVLocal Store

Using snapshots you can make backups of your KVLocal store to copy its data and configurations. Later, you can restore KVLocal store data and configurations from a snapshot.

An application can invoke the following KVLocal APIs to create a snapshot, or restore from it, or remove a snapshot.

| Method Name | Description | Parameter(s) | Returns |
| --- | --- | --- | --- |
| `String createSnapshot(String name)` | Creates a new snapshot using the specified name as a suffix. This method backs up the data files and configuration files of the KVLocal store, including files required for restore activities. The snapshot data is stored in a directory inside the kvroot directory.<br><br>**Note:** When you create a snapshot, it is store | `name` - specifies the suffix to use for the snapshot name. | The generated snapshot name. The generated snapshot name has a date-time prefix. The date-time prefix consists of a 6-digit, year, month, day value in YYMMDD format, and a 6-digit hour, minute, seconds timestamp as HHMMSS. The date and time values are separated from each other with a dash (-) and include a dash (-) suffix before the input snapshot name. Example: If you specify `Thursday` as value for the name parameter, the generated snapshot name is `110915-153514-Thursday`. |

| Method Name | Description | Parameter(s) | Returns |
| --- | --- | --- | --- |
| | | directory of the host machine. But these snapshots don't | |

| Method Name | Description | Parameter(s) | Returns |
| --- | --- | --- | --- |
| | | become persistent backups unless they are copied to separate tape to | |

| Method Name | Description | Parameter(s) | Returns |
| --- | --- | --- | --- |
| | | range. It is your responsibility to copy each of the snapshots to an | |

| Method Name | Description | Parameter(s) | Returns |
| --- | --- | --- | --- |
| | | other location, preferably on a different machine, for data safe | |

| Method Name | Description | Parameter(s) | Returns |
| --- | --- | --- | --- |
| | | t<br>y<br>. | |
| `KVLocal restoreFromSnapshot(String rootDir, String name)` | Restores the store from a snapshot.<br>This method replaces the data in the `kvroot` directory with the data specified in the snapshot, then starts an embedded NoSQL database instance. The KVLocal instance obtained using this API must be explicitly stopped using the KVLocal stop API. | `rootDir` - specifies the root directory of the store.<br>`name` - specifies the name of the snapshot, including the date and time that was generated by `createSnapshot` API. | An instance of KVLocal. |

| Method Name | Description | Parameter(s) | Returns |
| --- | --- | --- | --- |
| `void removeSnapshot(String name)` | Removes a snapshot. | `name` - specifies the full name of the snapshot, including date and time, that was generated by the `createSnapshot` API.<br>**Tip:** To preserve storage, you should periodically remove o | |

| Method Name | Description | Parameter(s) | Returns |
|---|---|---|---|
| | | bsoletesnapshots. | |
| `String[] listSnapshots()` | Lists the names of all the snapshots. | | An array of names of snapshots. |

- Using TCP/IP sockets
- Using Unix domain sockets

## Using TCP/IP sockets

```
import oracle.kv.KVLocalConfig;
import oracle.kv.KVLocal;

/* Start the KVLocal */
KVLocalConfig config = new KVLocalConfig.InetBuilder("/home/
kvstore").build();
KVLocal local = KVLocal.start(config);

/* Create a Snapshot */
String snapshotName = local.createSnapshot("sp1");

/* List all Snapshots */
String snapshotName = local.listSnapshots();

/* Stop the KVLocal */
local.stop();

/* Restore from a Snapshot */
local = KVLocal.restoreFromSnapshot(rootDir,snapshotName);
```

## Using Unix domain sockets

```
import oracle.kv.KVLocalConfig;
import oracle.kv.KVLocal;

/* Start the KVLocal */
KVLocalConfig config = new KVLocalConfig.UnixDomainBuilder("/home/
kvstore").build();
KVLocal local = KVLocal.start(config);

/* Create a Snapshot */
String snapshotName = local.createSnapshot("sp1");

/* List all Snapshots */
String snapshotName = local.listSnapshots();

/* Stop the KVLocal */
local.stop();

/* Restore from a Snapshot */
local = KVLocal.restoreFromSnapshot(rootDir,snapshotName);
```

# KVLocal Diagnostic Utility

An application can invoke the following KVLocal APIs to catch KVLocal configuration errors. These APIs return important and meaningful information in JSON format, which you can use to identify or diagnose the problem.

| Method Name | Description | Parameter | Returns |
|---|---|---|---|
| `String verifyConfiguration(boolean verbose)` | Verifies the store configuration by iterating over components and checking their state against the information maintained by the Admin service.<br>This method checks if an embedded NoSQL database instance is running in a stable and healthy state. If there are any configuration errors, violations or warnings are generated in the output. Violations are issues that can cause problems and should be investigated. | `verbose` - specifies whether or not the output contains the verbose output. If false, the output contains violations and warnings *only*. | The verified configuration results in JSON format. |

| Method Name | Description | Parameter | Returns |
|---|---|---|---|
| `String verifyData()` | Verifies store data integrity. This method is relatively time-consuming since it verifies the Log record integrity on disk and B-tree integrity in memory.<br>If a KVLocal instance has a persistent B-tree or log corruption, the service shuts down and the JE environment is invalidated. JE then creates a file called 7fffffff.jdb, placing it wherever other .jdb files exist in your environment. Manual administration intervention is required to recover from persistent data corruption. For more information on see Recovering from Data Corruption.<br>If a KVLocal instance has transient corruption, the service automatically exits. Transient corruption can occur due to memory corruption. Restarting an embedded NoSQL database instance is required to recover from transient corruption. | | The verified data result in JSON format. If no transient corruption is found, the result shows "No Btree Corruptions" and "No Log File Corruptions". |

# Sample JSON output from Diagnostic Utility

**Sample verbose JSON output from `verifyConfiguration` API**


```
{
  "topology": {
    "storeName": "kvstore",
    "sequenceNumber": 5,
    "numPartitions": 1,
    "numStorageNodes": 1,
    "time": 1629830958933,
    "version": "21.3.0"
```

```
  },
  "storewideLogName": "localhost:/home/hongyang/projects/kvlocal/
kvstore/build/kvsandbox/diagnosticTool/kvstore/log/kvstore_{0..N}.log",
  "shardStatus": {
    "healthy": 1,
    "writable-degraded": 0,
    "read-only": 0,
    "offline": 0,
    "total": 1
  },
  "adminStatus": "healthy",
  "zoneStatus": [
    {
      "resourceId": "zn1",
      "name": "KVLite",
      "type": "PRIMARY",
      "allowArbiters": false,
      "masterAffinity": false,
      "rnSummaryStatus": {
        "online": 1,
        "offline": 0,
        "read-only": 0,
        "hasReplicas": false
      }
    }
  ],
  "snStatus": [
    {
      "resourceId": "sn1",
      "hostname": "localhost",
      "registryPort": 5000,
      "zone": {
        "resourceId": "zn1",
        "name": "KVLite",
        "type": "PRIMARY",
        "allowArbiters": false,
        "masterAffinity": false
      },
      "serviceStatus": "RUNNING",
      "version": "21.3.0 2021-08-24 18:48:25 UTC  Build id:
008f726d548c+ Edition: Enterprise",
      "isMasterBalanced": true,
      "serviceStartTime": "2021-08-24 18:49:09 UTC",
      "adminStatus": {
        "resourceId": "admin1",
        "status": "RUNNING",
        "state": "MASTER",
        "authoritativeMaster": true,
        "serviceStartTime": "2021-08-24 18:49:11 UTC",
        "stateChangeTime": "2021-08-24 18:49:11 UTC",
        "availableStorageSize": "2 GB"
      },
      "rnStatus": [
        {
          "resourceId": "rg1-rn1",
```

```
            "status": "RUNNING",
            "requestsEnabled": "ALL",
            "state": "MASTER",
            "authoritativeMaster": true,
            "expectedStatus": "RUNNING",
            "sequenceNumber": 36,
            "haPort": 5003,
            "storageType": "HD",
            "availableStorageSize": "9 GB",
            "serviceStartTime": "2021-08-24 18:49:12 UTC",
            "stateChangeTime": "2021-08-24 18:49:12 UTC"
          }
        ],
        "anStatus": []
      }
  ],
  "violations": [],
  "warnings": [],
  "operation": "verify configuration -json -silent",
  "return_code": 5000,
  "description": "Operation ends successfully"
}
```


### Sample JSON output from `verifyData` API


```
{
  "Verify Report": {
            "rg1-rn1": {
             "Btree Verify":"No Btree Corruptions",
             "Log File Verify":"No Log File Corruptions"
                        }
                   }
}
```

# KVLocal FAQs

**What if the existing store's configuration parameters are different from the KVLocal start() parameters?**

If the existing store's configuration parameters are different from the KVLocalConfig parameters, the `KVLocal.start()` throws an exception, "parameter not consistent.". You can either update the KVLocalConfig parameters to match the ones recorded in the directory or call the `startExistingStore()` API.

**Does the KVLocal RepNode service automatically restart after it has stopped?**

It depends on whether or not the service shutdown was requested by the user.

The RepNode service automatically gets restarted if it crashes. However, a lot of crashes in a short period prevent automatic restarts. In that case, you should diagnose the problem and restart the service manually.

When the RepNode service stops because the user called the `KVLocal.stop()` API or exit the application, then the RepNode service needs to be restarted manually by calling the `KVLocal.start()` API.

**Is the existing Java Direct Driver API supported?**

KVLocal supports the Java Direct Driver API.