Red Hat Data Grid 8.4

Hot Rod .NET Client Guide

Configure and use Hot Rod .NET/C# clients

Last Updated: 2024-03-26

Red Hat Data Grid 8.4 Hot Rod .NET Client Guide

Configure and use Hot Rod .NET/C# clients

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Hot Rod .NET/C# clients allow C# runtime applications to connect and interact with remote Data Grid clusters.

Table of Contents

RED HAT DATA GRID 3

DATA GRID DOCUMENTATION 4

DATA GRID DOWNLOADS 5

MAKING OPEN SOURCE MORE INCLUSIVE 6

CHAPTER 1. INSTALLING AND CONFIGURING THE HOT ROD .NET/C# CLIENT 7
1.1. INSTALLING HOT ROD .NET/C# CLIENTS 7
1.2. CONFIGURATION AND REMOTE CACHE MANAGER APIS 7

RED HAT DATA GRID

Data Grid is a high-performance, distributed in-memory data store.

Schemaless data structure

Flexibility to store different objects as key-value pairs.

Grid-based data storage

Designed to distribute and replicate data across clusters.

Elastic scaling

Dynamically adjust the number of nodes to meet demand without service disruption.

Data interoperability

Store, retrieve, and query data in the grid from different endpoints.

DATA GRID DOCUMENTATION

Documentation for Data Grid is available on the Red Hat customer portal.

- Data Grid 8.4 Documentation
- Data Grid 8.4 Component Details
- Supported Configurations for Data Grid 8.4
- Data Grid 8 Feature Support
- Data Grid Deprecated Features and Functionality

DATA GRID DOWNLOADS

Access the Data Grid Software Downloads on the Red Hat customer portal.

NOTE

You must have a Red Hat account to access and download Data Grid software.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. INSTALLING AND CONFIGURING THE HOT ROD .NET/C# CLIENT

Install the Hot Rod .NET/C# client on Microsoft Windows systems where you use .NET Framework to interact with Data Grid clusters via the **RemoteCache** API.

1.1. INSTALLING HOT ROD .NET/C# CLIENTS

Data Grid provides an installation package to install the Hot Rod .NET/C# client on Windows.

Prerequisites

- Any operating system on which Microsoft supports the .NET Framework
- .NET Framework 4.6.2 or later
- Windows Visual Studio 2015 or later

Procedure

1. Download **redhat-datagrid-<version>-hotrod-dotnet-client.msi** from the Data Grid Software Downloads.
2. Launch the MSI installer for the Hot Rod .NET/C# client and follow the interactive wizard through the installation process.

1.2. CONFIGURATION AND REMOTE CACHE MANAGER APIS

Use the **ConfigurationBuilder** API to configure Hot Rod .NET/C# client connections and the **RemoteCacheManager** API to obtain and configure remote caches.

Basic configuration

```
using Infinispan.HotRod;
using Infinispan.HotRod.Config;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace simpleapp
{
   class Program
   {
      static void Main(string[] args)
      {
         ConfigurationBuilder builder = new ConfigurationBuilder();
         // Connect to a server at localhost with the default port.
         builder.AddServer()
            .Host(args.Length > 1 ? args[0] : "127.0.0.1")
            .Port(args.Length > 2 ? int.Parse(args[1]) : 11222);
         Configuration config = builder.Build();
         // Create and start a RemoteCacheManager to interact with caches.
         RemoteCacheManager remoteManager = new RemoteCacheManager(conf);
```

```
            remoteManager.Start();
            IRemoteCache<string,string> cache=remoteManager.GetCache<string, string>();
            cache.Put("key", "value");
            Console.WriteLine("key = {0}", cache.Get("key"));
            remoteManager.Stop();
        }
    }
}
```

Authentication

```
ConfigurationBuilder builder = new ConfigurationBuilder();
// Add a server with specific connection timeouts
builder.AddServer().Host("127.0.0.1").Port(11222).ConnectionTimeout(90000).SocketTimeout(900);
// ConfigurationBuilder has fluent interface, options can be appended in chain.
// Enabling authentication with server name "node0",
// sasl mech "PLAIN", user "supervisor", password "aPassword", security realm "aRealm"
builder.Security().Authentication().Enable().ServerFQDN("node0")
    .SaslMechanism("PLAIN").SetupCallback("supervisor", "aPassword", "aRealm");
Configuration c = conf.Build();
```

Encryption

```
ConfigurationBuilder builder = new ConfigurationBuilder();
builder.AddServer().Host("127.0.0.1").Port(11222);
// Get configuration builder for encryption
SslConfigurationBuilder sslBuilder = conf.Ssl();
// Enable encryption and provide client certificate
sslBuilder.Enable().ClientCertificateFile("clientCertFilename");
// Provide server cert if server needs to be verified
sslBuilder.ServerCAFile("serverCertFilename");
Configuration c = conf.Build();
```

Cross-site failover

```
ConfigurationBuilder builder = new ConfigurationBuilder();
  builder.AddServer().Host("127.0.0.1").Port(11222);
  // Configure a remote cluster and node when using cross-site failover.
  builder.AddCluster("nyc").AddClusterNode("192.0.2.0", 11322);
```

Near caching

```
ConfigurationBuilder builder = new ConfigurationBuilder();
  builder.AddServer().Host("127.0.0.1").Port(11222);
  // Enable near-caching for the client.
  builder.NearCache().Mode(NearCacheMode.INVALIDATED).MaxEntries(10);
```