Red Hat Data Grid 8.4

Hot Rod C++ Client Guide

Configure and use Hot Rod C++ clients

Last Updated: 2024-03-26

Red Hat Data Grid 8.4 Hot Rod C++ Client Guide

Configure and use Hot Rod C++ clients

Legal Notice

Abstract

Hot Rod 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 THE HOT ROD C++ CLIENT 7
1.1. C++ COMPILER REQUIREMENTS 7
1.2. INSTALLING HOT ROD C++ CLIENTS ON RED HAT ENTERPRISE LINUX (RHEL) 7
1.3. INSTALLING HOT ROD C++ CLIENTS ON MICROSOFT WINDOWS 7

CHAPTER 2. COMPILING PROTOBUF SCHEMA 9
2.1. COMPILING PROTOBUF SCHEMA ON RED HAT ENTERPRISE LINUX (RHEL) 9
2.2. COMPILING PROTOBUF SCHEMA ON MICROSOFT WINDOWS 9

CHAPTER 3. CONFIGURING THE HOT ROD C++ CLIENT 11
3.1. CONFIGURATION AND REMOTE CACHE MANAGER APIS 11

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 THE HOT ROD C++ CLIENT

Install the Hot Rod C++ client on your host system as a dynamic library.

1.1. C++ COMPILER REQUIREMENTS

Operating system	Required compiler
Red Hat Enterprise Linux (RHEL) 8	C++ 11 compiler (GCC 8.5.0)
RHEL 9	C++ 11 compiler (GCC 11.3.1)
Microsoft Windows 7 x64	C++ 11 compiler (Visual Studio 14 2015 Win64, Microsoft Visual C++ 2013 Redistributable Package for the x64 platform)

1.2. INSTALLING HOT ROD C++ CLIENTS ON RED HAT ENTERPRISE LINUX (RHEL)

Data Grid provides an RPM distribution of the Hot Rod C++ client for RHEL.

Procedure

1. Enable the repository for the Hot Rod C++ client on RHEL.

RHEL version	Repository
RHEL 8	**jb-datagrid-8.4-for-rhel-8-x86_64-rpms**
RHEL 9	**jb-datagrid-8.4-for-rhel-9-x86_64-rpms**

2. Install the Hot Rod C++ client.

```
# yum install jdg-cpp-client
```

Additional resources

- Enabling or disabling a repository using Red Hat Subscription Management (Red Hat Knowledgebase)
- Red Hat Package Browser

1.3. INSTALLING HOT ROD C++ CLIENTS ON MICROSOFT WINDOWS

Data Grid provides an archived version of the Hot Rod C++ client for installation on Windows.

Procedure

1. Download the ZIP archive for the Hot Rod C++ client from the Data Grid Software Downloads.

2. Extract the ZIP archive to your file system.

CHAPTER 2. COMPILING PROTOBUF SCHEMA

Data Grid uses the ProtoStream API to store data as Protobuf-encoded entries.

Protobuf is a language-neutral format that allows clients to create and retrieve entries in remote caches using both Hot Rod and REST endpoints.

2.1. COMPILING PROTOBUF SCHEMA ON RED HAT ENTERPRISE LINUX (RHEL)

Compile Protobuf schema, **.proto** files, into C++ header and source files to describe your data to Data Grid.

Prerequisites

- Install the Protobuf library and **protobuf-devel** package.

```
# yum install protobuf
# yum install protobuf-devel
```

Procedure

1. Set the **LD_LIBRARY_PATH** environment variable, if it is not already set.

```
# export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/lib64
```

2. Compile Protobuf schema for the Hot Rod C++ client as required.

```
# /bin/protoc --cpp_out dllexport_decl=HR_PROTO_EXPORT:/path/to/output/ $FILE
```

HR_PROTO_EXPORT is a macro that the Hot Rod C++ client expands when it compiles the Protobuf schema.

3. Register your Protobuf schema with Data Grid if you plan to use queries.

Additional resources

- Registering Protobuf Schemas

2.2. COMPILING PROTOBUF SCHEMA ON MICROSOFT WINDOWS

Compile Protobuf schema, **.proto** files, into C++ header and source files to describe your data to Data Grid.

Procedure

1. Open a command prompt to the installation directory for the Hot Rod C++ client.
2. Compile Protobuf schema for the Hot Rod C++ client as required.

```
bin\protoc --cpp_out dllexport_decl=HR_PROTO_EXPORT:path\to\output\ $FILE
```

HR_PROTO_EXPORT is a macro that the Hot Rod C++ client expands when it compiles the Protobuf schema.

3. Register your Protobuf schema with Data Grid if you plan to use queries.

Additional resources

- Registering Protobuf Schemas

CHAPTER 3. CONFIGURING THE HOT ROD C++ CLIENT

Hot Rod C++ clients interact with remote Data Grid clusters via the **RemoteCache** API.

3.1. CONFIGURATION AND REMOTE CACHE MANAGER APIS

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

Configuration builder

```
#include "infinispan/hotrod/ConfigurationBuilder.h"
#include "infinispan/hotrod/RemoteCacheManager.h"
#include <infinispan/hotrod/RemoteCache.h>
#include <iostream>
int main () {
  ConfigurationBuilder builder;
    // Configure a cache manager to connect with Hot Rod version 2.8
    builder.protocolVersion(Configuration::PROTOCOL_VERSION_28);
    // Connect to a server at localhost with the default port.
    builder.addServer().host("127.0.0.1").port(11222);
    // Create and start a RemoteCacheManager to interact with caches.
    RemoteCacheManager cacheManager(builder.build(), false);
    cacheManager.start();
    ...
}
```

Cross-site replication

```
ConfigurationBuilder builder;
  builder.addServer().host("127.0.0.1").port(11222);
  // Configure a remote cluster and node when using cross-site replication.
  builder.addCluster("NYC").addClusterNode("192.0.2.0", 11322);
```

Near caching

```
ConfigurationBuilder builder;
  builder.addServer().host("127.0.0.1").port(11222);
  // Enable near-caching for the client.
  builder.nearCache().mode(NearCacheMode::INVALIDATED).maxEntries(4);
```

Additional resources

- Hot Rod C++ client API