Virtual DataPort Connectivity

Revision 20191025

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 OVERVIEW

The Denodo Platform provides different connectivity options that client applications can choose when connecting to a Denodo Virtual DataPort server. This document describes the different possibilities and the implications of each of them.

2 CONNECTION APIS

2.1 RMI

RMI stands for Remote Method Invocation and is the Java API that Denodo uses internally in the applications that work as clients of the Denodo servers, such as the Virtual DataPort Administration Tool, the Data Catalog or the Denodo JDBC Driver. It is important to be aware of this protocol in order to diagnose connectivity problems between the Administration Tool and the Virtual DataPort server, but please, notice that connecting directly via RMI requires a custom development in Java using Denodo libraries and the resulting code will not be as portable as using other connection protocols and so, is not recommended. In general, the default method for connecting to a Denodo server from a non-Denodo client will be JDBC or ODBC.

RMI uses two ports for the connection, the "*RMI registry port*" and the "*RMI factory port*". The "*RMI registry hostname"* and the "*RMI factory hostname"* are typically the same. Only in some clustering scenarios, those values may differ.

We will provide detailed information on the RMI protocol in the sections below.

2.2 JDBC

One of the best options for connecting to Denodo from third party tools is by using the JDBC (Java Database Connectivity) API. Virtual DataPort distributes a JDBC Driver that allows Denodo clients to connect with Denodo servers through this API. The JDBC Driver is using the RMI API internally, so the JDBC connection string that the driver requires must include the *RMI registry hostname* and the *RMI registry port* of the Denodo server. However, notice that we do not need to include the *RMI factory port* in the Denodo connection string as the port and the *RMI registry hostname* are automatically transmitted by the RMI protocol.

2.3 ODBC

ODBC (Open DataBase Connectivity) is the default API for Microsoft products when working with data stores. Virtual DataPort provides an ODBC driver, and as with JDBC, it will be necessary to establish the Virtual DataPort port and hostname to use in the ODBC connection configuration. In this case, the ODBC driver will not use RMI, so it means that Denodo JDBC and ODBC port will be different.

2.4 WEB SERVICES

Denodo eases the consumption of data with the publication of REST and SOAP Web Services. In this case, the client applications are not connecting to the Denodo Virtual DataPort server, but to the web container that hosts the Web Services using the HTTP or HTTPS protocols. The web services will do the connection to the Virtual DataPort server to retrieve data via RMI. This information is relevant if you intend to use an external web container instead of the one bundled with the Denodo Platform.

3 THE RMI PROTOCOL

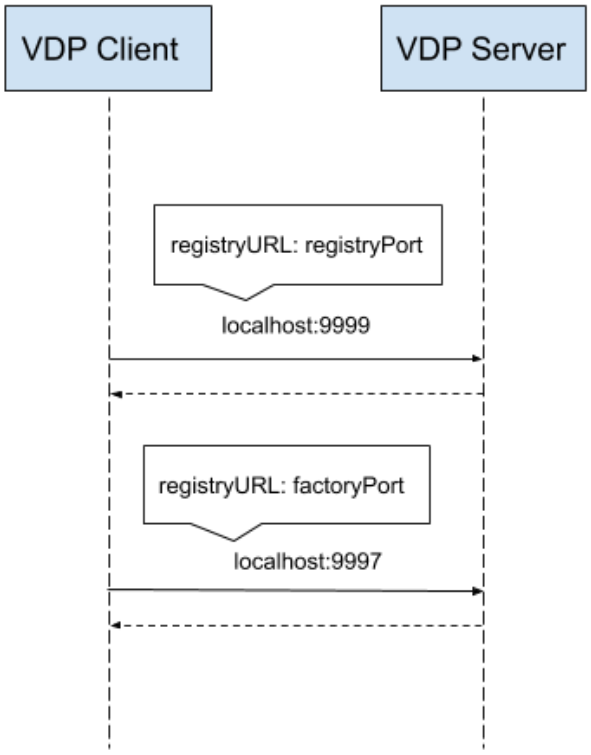
Virtual DataPort, the main module of the Denodo Platform, follows a client-server architecture. This way, developers and business users working with Denodo, do not need to worry about the performance of their computers since the data integration operations taking place will be computed on the server machine.

The protocol that supports this communication between the Virtual DataPort clients and servers is Java RMI (Remote Method Invocation). In this protocol, the RMI Applications define remote objects that can be referenced and invoked by the clients. One of the key elements of the protocol is the RMI registry, which allows us to find the remote “factory” of objects and to reference them.

The server-side in an RMI architecture is in charge of creating the RMI registry and RMI factory at some particular address and port. Default values in Virtual DataPort are localhost for registry and factory hostnames, and the port 9999 for the registry and 9997 for the factory.

The VDP server listens in the registry port for incoming requests, so once a request comes in, the factory address and port is returned, and that port will be used for all query and data communications.

The *registry host* and the *factory host* are typically the same and can be configured to have a value different than the default by editing the “registryURL” key in the configuration files.



Simplification of the requests involved in a new connection in Virtual DataPort

Using two different ports for connecting to Denodo leads frequently to some misunderstandings and network configuration issues, as Denodo users expect to be working only with the registry port (9999) for every connection to Denodo. However, as we have seen in the previous paragraph, there are two different ports involved in every connection to Virtual DataPort.

Sometimes, we may find other names that make reference to registry and factory ports. For instance, the “server port” or “RMI port” names all refer to the “registry port” and the “auxiliary port” refers to the “factory port”.

4 UPDATING THE RMI CONFIGURATION OF THE DENODO PLATFORM

In non-cluster scenarios, it is only necessary to configure the RMI host when the server machine provides multiple network interfaces. In these cases, you can replace the default *localhost* value in the RMI host with the interface visible to the clients, which can be a reachable IP address or hostname.

The registry and factory ports can also be changed if these ports are already been used by other services running in the machine.

4.1 UPDATING THE RMI CONFIGURATION GRAPHICALLY

The RMI host for each Denodo Platform server can be configured by navigating to the Denodo Control Center `Configure` > `JVM Options` > `RMI host` tab.

On the `RMI host` tab, next to each field, there is a refresh button that returns the field to the default value in case that you need to restore the parameter. The OK button saves the configuration changes. These changes are applied in the next startup of the affected programs. In most cases, you will only need to modify this value for the `Virtual DataPort Server/ ITPilot Wrapper Server` option.

On the other hand, you can open the dialog *Administration* > *Server Configuration* > *Server connectivity* from the Virtual DataPort Administration Tool to modify the registry and factory ports. The field name *Server Port Number* refers to the registry port while the *Auxiliary Port Number* refers to the factory.

4.2 UPDATING THE RMI CONFIGURATION FROM COMMAND LINE

On an environment without graphical support or without access to the Denodo Control Center the same configuration can be changed by following these steps:

1. Open the file `<DENODO_HOME>/conf/vdp/VDBConfiguration.properties` and set the property: `com.denodo.vdb.vdbinterface.server.VDBManagerImpl.registryURL` to the appropriate RMI host or IP address value.
2. Optionally, update the properties `com.denodo.vdb.vdbinterface.server.VDBManagerImpl.registryPort` and `com.denodo.vdb.vdbinterface.server.VDBManagerImpl.factoryPort` to use RMI customized ports.
3. Once the configuration file has been saved, you need to execute the script `<DENODO_HOME>/bin/regenerateFiles{.sh|.bat}` to propagate the changes to the startup scripts.
4. Finally, restart the Virtual DataPort server for the changes to take effect.

5 RMI TROUBLESHOOTING

Any time a connection error is received when attempting to log into the VDP server from a remote client, check that the RMI configuration is set up appropriately

Also, consider adding the RMI debugging properties to the Java configuration of the applications, they are very useful to print the RMI related information in both server and client sides.

For Denodo Platform 7.0 installations with update 20190903 or newer:

- Add the following property to the log4j configuration of the server in `<DENODO_HOME>/conf/vdp/log4j2.xml`:
  `<Logger name="sun.rmi.server.call" level="trace" />`

- Add the following property to the Log4j configuration of the Virtual DataPort Administration Tool in `<DENODO_HOME>/conf/vdp-admin/log4j2.xml`:
  `<Logger name="sun.rmi.client.call" level="trace" />`

For Denodo 7.0 installations with older updates than 20190903:

- On the server-side, set the following property in the JVM configuration of the server:
  `-Djava.rmi.server.logCalls=true`

  Then, start the Virtual DataPort server as:
  `<DENODO_HOME>/bin/vqlserver_startup.sh > server_out.txt 2>&1`

- On the client-side, set the following property in the JVM configuration of the client:
  `-Dsun.rmi.client.logCalls=true`

  Then, start the client redirecting the output to an auxiliary file:
  `<DENODO_HOME>/bin/vdpadmin.bat > client_output.txt 2>&1`

6 HIGH AVAILABILITY AND CONTAINERS

It is also important to take into account the particularities of RMI when working with load balancers or in cloud environments. For instance, given that RMI uses two different ports (by default 9999 and 9997) for establishing a standard connection, it will not be enough just opening one of those in the corporate firewall, we need both to be open.

6.1 LOAD BALANCERS

In a High Availability scenario, there are several Denodo servers placed behind a Load

Balancer, which will evenly distribute the requests. The configuration of the different pieces will depend on the Load Balancer chosen but, in general, we must take care of the RMI configuration of the system so the communication can be established from the client-side to the server-side back and forth.

The protocol used by the connection to the registry does not require specific settings in the load balancer as it is only used to locate the “factory”, but the protocol used by each connection to the factory is stateful, so you have to either configure the load balancer to have persistence/affinity settings to ensure that a client is always connected to the same factory node or use different factory ports for each node.

The Knowledge Base article Denodo Platform Cluster Architecture explains how to set up this kind of configurations.

6.2 DOCKER

Denodo also distributes a Docker container in order to deploy Virtual DataPort in a containerized architecture. Again, we need to bear in mind how Denodo and RMI works to ensure that our Denodo clients can connect to our containerized services. By default, Docker does not expose any port outside, so we will need to include the RMI ports in the Docker configuration.

The Denodo Platform Container QuickStart Guide explains the different options and parameters that we have to use to run the Denodo container.

6.3 KUBERNETES

In complex container architectures, it will be useful to use container orchestration systems such as Kubernetes, this will ease the configuration of replica sets and many other features.

As in Docker, Kubernetes requires to define what ports do the service needs to operate. Hence, here as well the RMI ports will be included in the design of the whole system configuration.

The Knowledge Base article Deploying Denodo in Kubernetes lists the steps for running a successful Denodo deployment in Kubernetes.

7 REFERENCES

Denodo Platform Installation Guide: Denodo Platform Configuration
Solution Manager Installation Guide: Virtual Machine and Web Container Configuration
The Java Tutorials: An Overview of RMI Applications
Debugging RMI: Java RMI Implementation Logging
Lesson: JDBC Introduction: JDBC Architecture