IBM® DB2® Technical White Paper

DB2 pureScale enablement

Workload balancing, automatic client reroute, and client affinities

Charlie Wang
Advisory Software Engineer

Mike Springgay
Senior Development Manager

Aslam Nomani
Quality Assurance Architect

Frankie Sun
Advisory Software Engineer

CONTENTS

1 Executive summary

In today's highly competitive marketplace, it is important to deploy a data processing architecture that not only meets your immediate tactical needs but can also grow and change to adapt to your future strategic requirements. In December 2009, IBM introduced the DB2 pureScale Feature for Enterprise Server Edition software (DB2 pureScale Feature). The DB2 pureScale Feature builds on familiar and proven design features from the IBM DB2 for z/OS database software (DB2 for z/OS), bringing the industry-leading technology and reliability of DB2 for z/OS to open systems.

The DB2 pureScale Feature provides the following advantages:

- Practically unlimited capacity. The DB2 pureScale Feature provides practically unlimited capacity by allowing for the addition and removal of members on demand. The DB2 pureScale Feature can scale to 128 members by using a highly efficient centralized management facility. The DB2 pureScale Feature also uses a technology called Remote Direct Memory Access (RDMA). RDMA provides a highly efficient internode communication mechanism that also facilitates the scaling capabilities of the DB2 pureScale Feature.

- Application transparency. An application that runs in a DB2 pureScale environment does not need to have any knowledge of the different members in the cluster or have to be concerned about partitioning data. The DB2 client that is supported by the DB2 pureScale Feature automatically routes application requests to the most appropriate members. The DB2 pureScale Feature also provides native support for a great deal of syntax that is used by database vendors other than IBM. Therefore, applications from those vendors can run in a DB2 pureScale environment with minimal or no changes. In many cases, you can take advantage of the benefits of the DB2 pureScale Feature without having to modify your applications.

- Continuous availability. The DB2 pureScale Feature provides a fully active-active configuration such that if one member goes down, processing can continue on the remaining active members. During a failure, only data that is being modified on the failing member is unavailable, and only until database recovery is completed for that set of data, which is very quick.

- Reduced TCO. The DB2 pureScale Feature can help reduce total cost of ownership through its integrated and simplified deployment and maintenance capabilities. The DB2 pureScale Feature provides tools and utilities to handle the deployment and maintenance of components that are integrated within the DB2 pureScale Feature. These tools and utilities help reduce the steep learning curves that are typically associated with deploying such technologies.

Workload balancing (WLB) and client affinities are the critical DB2 client features in delivering the value propositions of the DB2 pureScale Feature. This paper describes the concept of DB2 workload balancing (WLB) and client affinities. This paper provides detailed examples to illustrate how to enable and use these features to access DB2 pureScale cluster from WebSphere Application Server and various stand-alone DB2 clients. This paper also demonstrates how automatic client reroute (ACR) routes application requests among pureScale members when an outage occurs. In addition, this paper provides tips on how to recover from a non-responsive TCP/IP layer along with how to provide alternate servers for the first connection to ensure successful access to the DB2 pureScale database.

2 Introduction

2.1 Workload balancing overview

Database applications running in a DB2 pureScale environment can use the DB2 transaction-level or connection-level workload balancing (WLB) functionality. WLB balances application requests among all members of the DB2 pureScale cluster. When WLB is enabled the DB2 clients distribute workload or application request based on the capacity (that is, the priority or weight) values in a server list that the DB2 pureScale server returns. These capacity values indicate the current load on a DB2 pureScale member. A member with a capacity value below that of the other members in the server list is busier than other members.

To avoid overloading particular members, when client processing begins, the client automatically and transparently sends the transaction to a member that has a higher capacity value. The following diagrams illustrate scenarios where the workload is not evenly balanced across the DB2 pureScale members and the workload is then balanced across the DB2 pureScale members by using the WLB feature.

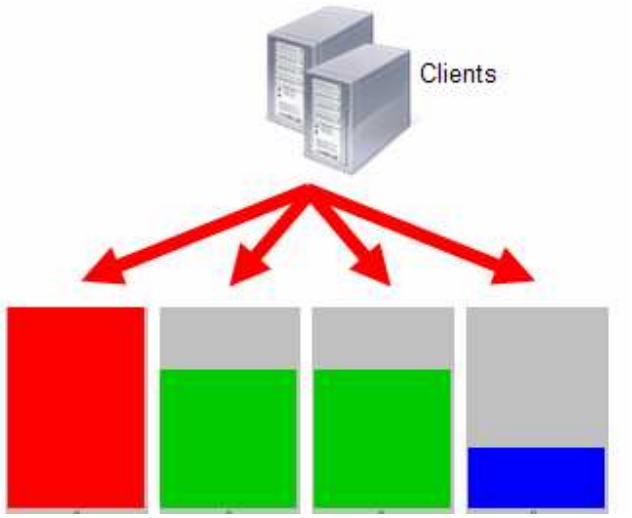


Figure 1. *Example of unbalanced workload across DB2 pureScale members*

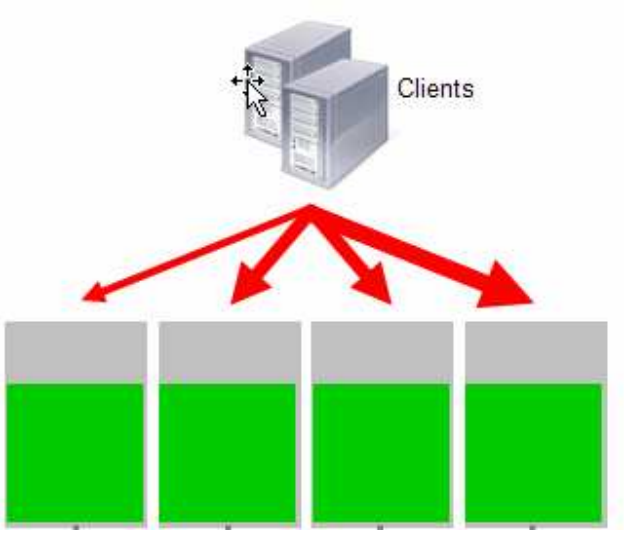


Figure 2. *Example of balanced workload across DB2 pureScale members*

For Java clients, WLB uses transaction-level balancing only. In this paper, WLB means transaction-level balancing unless explicitly indicated otherwise.

2.2 Client affinities overview

The client affinities feature is another DB2 client technology that you can use to explicitly control how database applications use your DB2 pureScale members. The client affinities feature provides a highly customizable way for you to use your DB2 pureScale members based on your specific business requirements.

The client affinities feature allows you to define an ordered list of DB2 pureScale members to which a DB2 client can connect; different clients can implement a different ordered list. In certain situations, you might want to direct application requests from a DB2 client to a particular DB2 pureScale member on the list. If that DB2 pureScale member goes down because of a planned or unplanned outage, the DB2 client can direct the client application requests to another DB2 pureScale member on the list. If that member is unavailable, the DB2 client can work through the list of all DB2 pureScale members to find an available member. This feature is typically used in an environment where the applications and data are inherently segregated and particular servers are targeted to service requests of particular applications.

With client affinities, you can also control whether application requests fail back to the failure primary server after it comes back online. The primary server is the DB2 pureScale member that the application originally connected to. If you set up the client in this manner, you can choose how often the DB2 client should check whether the primary server is back online.

The following diagram illustrates four application servers that are initially directed toward a specific DB2 pureScale member. If the DB2 pureScale member to which the application servers in Group C are connected fails, those applications are automatically routed to another DB2 pureScale member based on the specified ordered list.

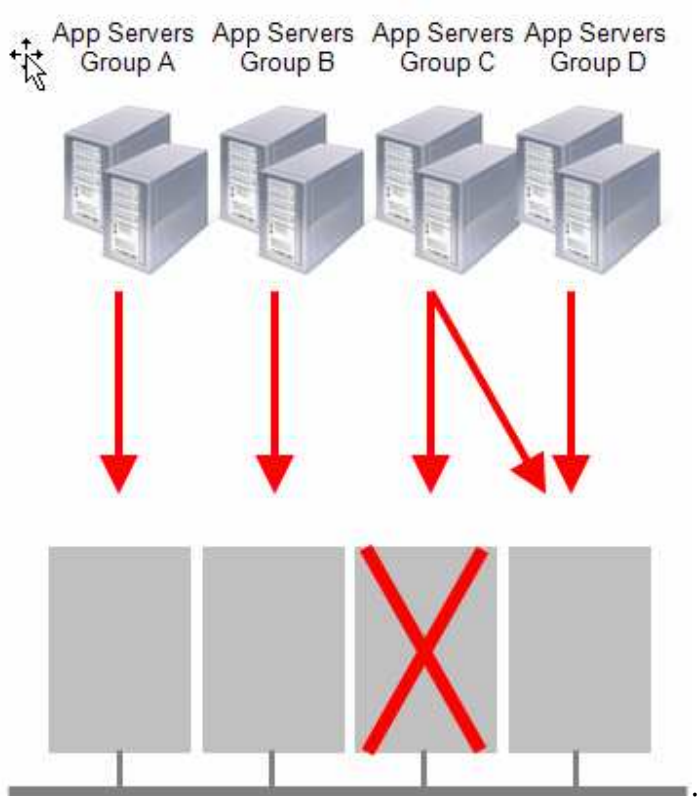


Figure 3. *Example of client affinities*

> **Important:** The client affinities feature is mutually exclusive with the WLB feature because the intended behavior of the application with these two methods is different.

2.3 Automatic client reroute with WLB and client affinities

DB2 automatic client reroute (ACR) complements a continuously available DB2 pureScale cluster that offers 24/7 availability for clients connecting to mission-critical production systems. ACR is a feature in DB2 clients that takes application requests that are directed toward an offline DB2 pureScale member and reroutes them to active DB2 pureScale members. ACR is automatically enabled with WLB or client affinities so no additional steps are required to specify which member the application should connect to upon encountering an outage.

In some cases, after an outage, clients are seamlessly routed to another DB2 pureScale member, and no error messages are returned to the application because the failure is transparent to the application. For more details on when failures are seamless, see the DB2 Information Center information about seamless client reroute. However, in some situations, the DB2 client cannot

replay the statement execution environment on the new connection after automatic client reroute occurs. In such a situation, the transaction is rolled back, and SQLCODE -30108 (or SQLCODE -4498 for Java applications) is returned to the application after the connection is rerouted from the failing member to a surviving member. If this occurs, applications must replay the statement execution environment and redo the statements, but applications do not have to explicitly reconnect to the database because ACR automatically reconnects the applications.

3 Enabling WLB and client affinities with WebSphere Application Server

DB2 Java applications with the supported IBM Data Server Driver for JDBC and SQLJ (JCC driver) running in an IBM WebSphere Application Server environment can use the WLB or client affinities features to access databases on a DB2 pureScale cluster. This section describes how to enable WLB and client affinities for DB2 Java applications in an IBM WebSphere Application Server environment. The IBM WebSphere Application Server level that supports the DB2 pureScale Feature is 7.0.0.9 or later. For information about the supported JCC driver versions, see Appendix A. Supported software levels.

3.1 Enabling WLB and seamless ACR for Java applications running on WebSphere Application Server

You must add JCC data source property enableSysplexWLB and set the value to true to enable WLB. Seamless ACR is implicitly enabled for JCC driver when you enable WLB. You must explicitly configure seamless ACR for WebSphere Application Server. You can add and configure these properties through the WebSphere Application Server administration console, as illustrated in the steps in this section.

It is assumed that you already added a JCC driver and created a JCC data source for connecting to a database on a DB2 pureScale cluster. For instructions on how to add a JCC driver and a data source through the WebSphere Application Server administration console, see the IBM WebSphere Information Center.

Here are the steps for enabling WLB and ACR through the WebSphere Application Server administration console:

1. Configure the driver type and the DB2 pureScale connect member properties:

    a. Log in to the WebSphere Application Server administration console.
    b. Browse to the JCC data source that you use to connect to the DB2 pureScale cluster by clicking the corresponding WebSphere Application Server administration console links.
    c. In the **Common and required data source properties** area, specify the following information:

    - From the **Driver type** list, select **4**. This is the value of the JCC driver type.
    - In the **Database name** field, enter the name of the database that you want the applications to connect to.

- In the **Server name** field, enter the full domain name (the full hostname) of the DB2 pureScale connect member (the DB2 pureScale member that you want the applications to connect to). This can be any DB2 pureScale member in the cluster.
- In the **Port number** field, enter the TCP/IP port number on which the DB2 pureScale connect member is listening for connect requests.

Figure 4 shows sample data. In the **Driver type** field, we selected 4. In the **Database name** field, we entered `eComHQ`. In the **Server name** field, we entered `coralpib19a.torolab.ibm.com`. In the **Port number** field, we entered `56733`.

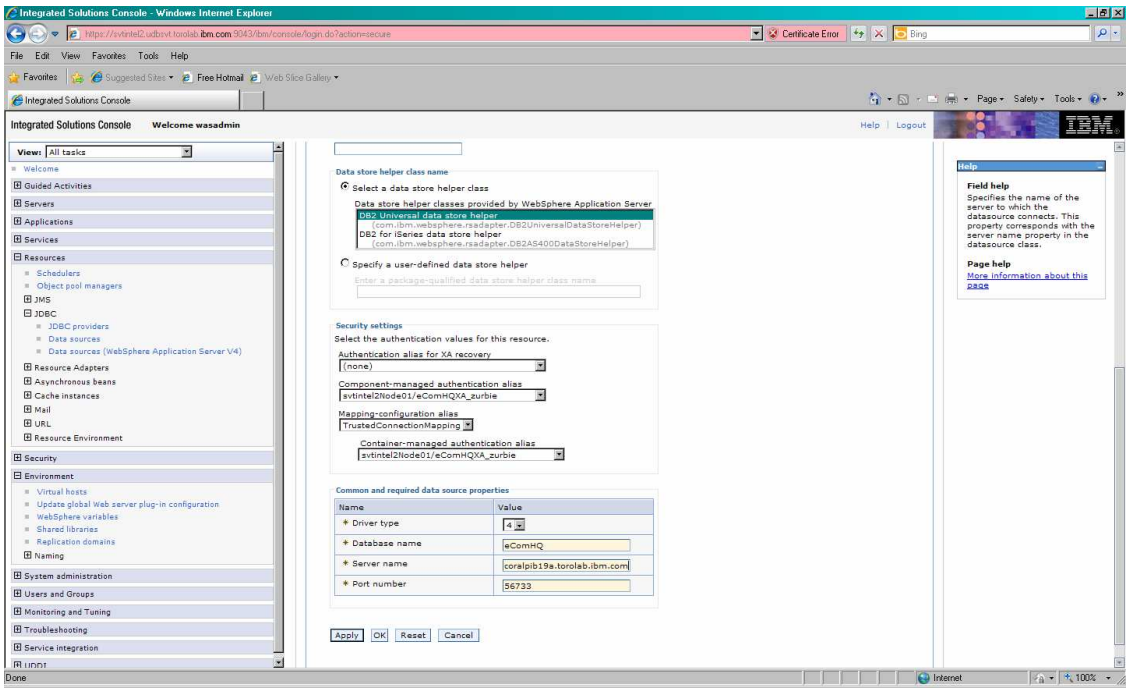


Figure 4. *Configure the driver type and the DB2 pureScale connect member properties*

d. Click **Apply** and save the changes.

2. Add and configure the `enableSysplexWLB` property:

a. Browse to your JCC data source.
b. Click **Custom properties** and then click **New**.
c. To enable WLB, in the **Name** field, enter `enableSysplexWLB`, and in the **Value** field, enter `true`, as illustrated in Figure 5.

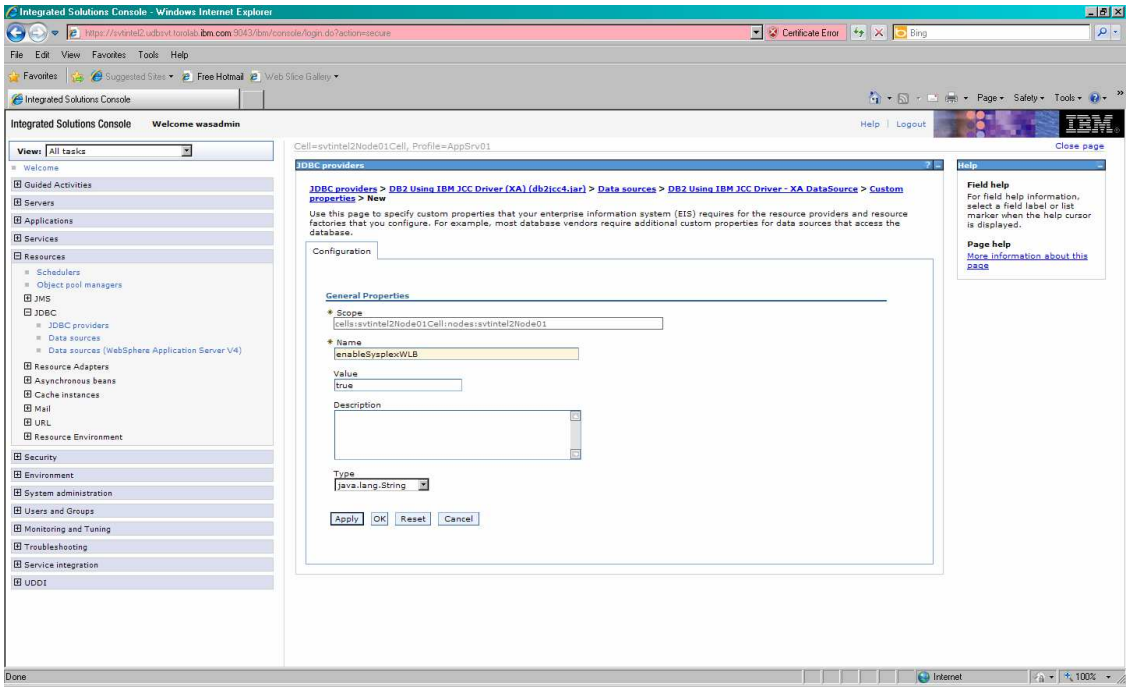


Figure 5. *Add and configure the enableSysplexWLB property*

d. Click **Apply** and save the changes.

3. Configure seamless ACR for WebSphere Application Server:

a. Browse to your JCC data source.
b. Click the **Connection pool properties**.
c. To enable WebSphere Application Server to support seamless ACR, from the **Purge policy** list, select **FailingConnectionOnly**, as illustrated in Figure 6.

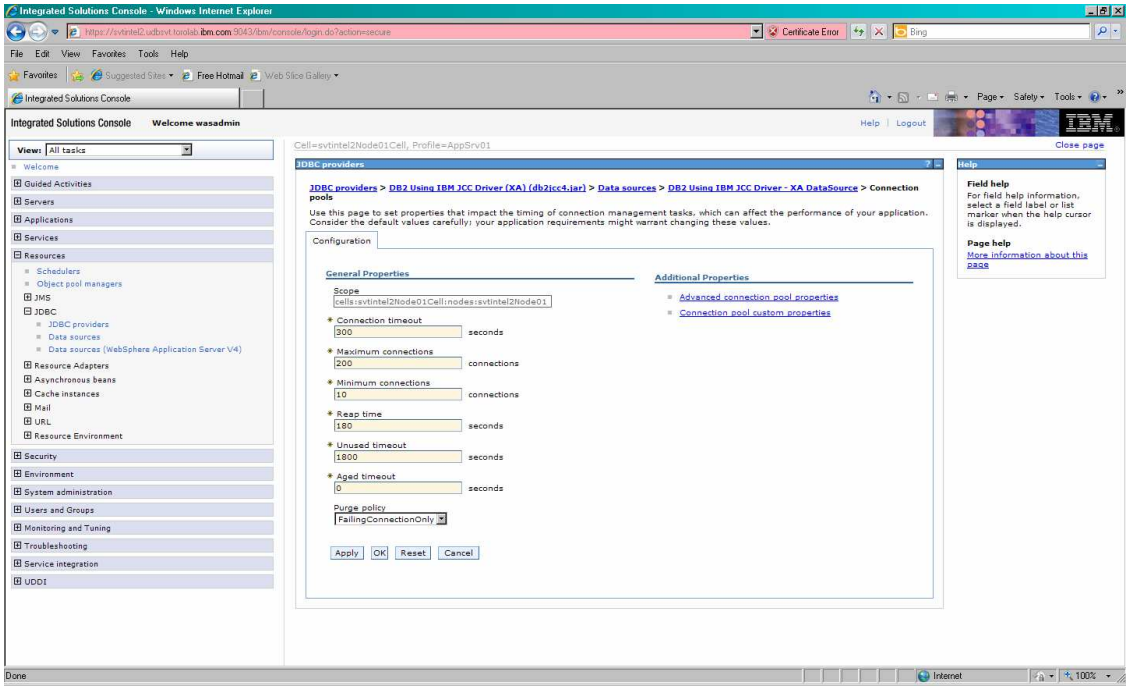


Figure 6. *Configure the Purge policy property*

d. Click **Apply** and save the change.

4. To have the property changes take effect, restart WebSphere Application Server.

Important: For WLB and ACR to work properly you must ensure you can ping the full domain names (i.e. the full hostnames) of all the DB2 pureScale members from the WebSphere Application Server.

3.2 Enabling client affinities for Java applications running on WebSphere Application Server

You must configure or both add and configure the following properties to enable client affinities for Java applications that connect to a DB2 pureScale cluster from WebSphere Application Server. To configure or add the properties, use the property panels and instructions in the previous section.

Custom properties for the data source:

- **Driver type**: Specify the JCC driver type. The value must be 4.
- **Database name**: Specify the destination database for the application connections.
- **Server name**: Leave this field blank. JCC does not use this server name when the client affinities feature is enabled.
- **Port number**: Leave this field blank. JCC does not use this port number when the client affinities feature is enabled.
- `enableSysplexWLB` property: Either delete the property from the **Custom properties** page panel or set its value to `false` to disable WLB. You must disable WLB for the client affinities feature to work.
- `enableClientAffinitiesList` property: Add the property to the **Custom properties** page panel and set the value to `1`.

- `clientRerouteAlternateServerName` property: The property is in the **Custom properties** page panel. Configure the property to provide a list of DB2 pureScale member full host names in a comma-separated list. The first member that you specify in this list is the primary server, which is the first server that applications try to connect to. If applications fail to connect to the primary server, they attempt to connect to the remaining servers in the list in sequence until they are able to connect to a server or the list is exhausted.
- `clientRerouteAlternatePortNumber` property: The property is in the **Custom properties** page panel. Configure the property to provide a list of TCP/IP port numbers in a comma-separated list for applications to use to connect to the servers that you specified for the `clientRerouteAlternateServerName` property.
- `affinityFailbackInterval` property: Add the property to the **Custom properties** page panel and specify, the length of time, in seconds, between attempts by the client to fail back to the primary server.
- `enableSeamlessFailover` property: Add the property to the **Custom properties** page panel and set the value to `1`.

Data source, connection pool properties:

- **Purge policy**: To enable WebSphere Application Server to support seamless ACR, set to `FailingConnectionOnly`.

To have the property changes take effect, restart WebSphere Application Server after you apply and save the changes.

The following figure shows examples of some of the client affinities feature properties that you can configure through the WebSphere Application Server administration console. In this example, the `clientRerouteAlternateServerName` property is set to `coralpib19a.torolab.ibm.com,coralpib19b.torolab.ibm.com`, and the `affinityFailbackInterval` property is set to `120` seconds. Using this configuration, all the application requests running on the WebSphere Application Server host go to the primary server (DB2 pureScale member coralpib19a.torolab.ibm.com), regardless of the value in the **Server name** field.

The alternate server (DB2 pureScale member coralpib19b.torolab.ibm.com) does not receive any requests when the primary server is operational. If the primary server goes down, all new and existing application requests are rerouted to the alternate server. When the primary server comes back online, connections fail back to the primary server if transaction boundaries (commit or rollback) are reached and the 120 seconds interval has expired. New application requests are sent to the primary server.

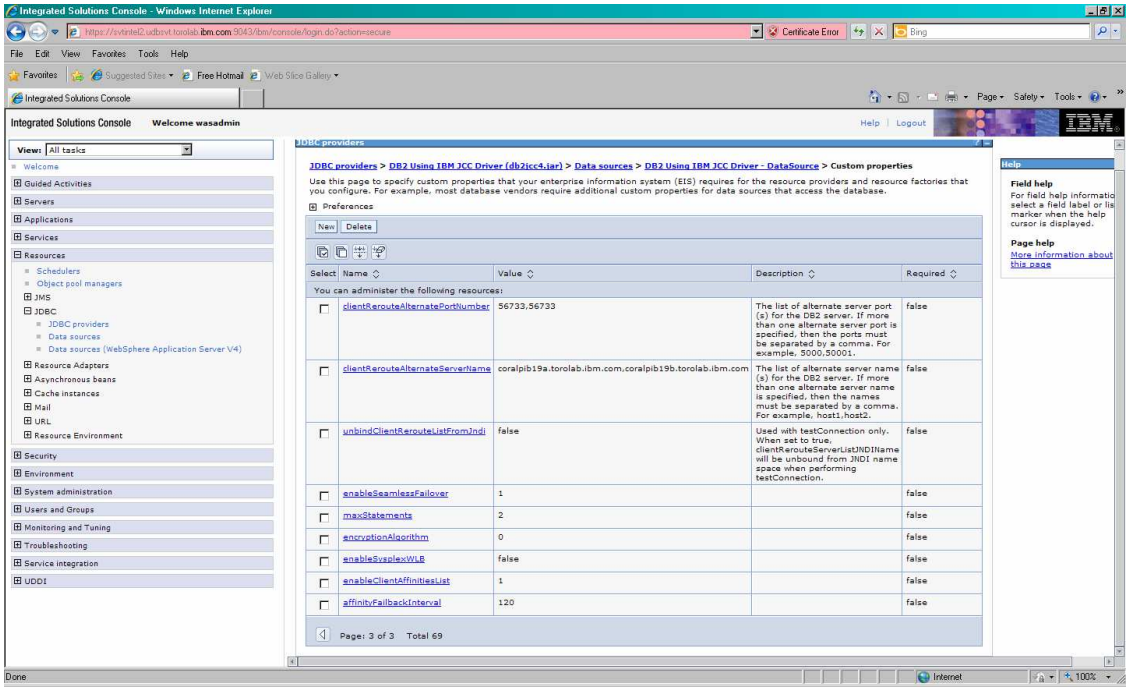


Figure 7. *Configure client affinities related properties*

There is a slight difference between Version 7 and Version 8 of the WebSphere Application Server administration console with respect to the configuration of the `clientRerouteAlternateServerName` and `clientRerouteAlternatePortNumber` properties.

With WebSphere Application Server 7, you configure these two properties through the JCC data source **Custom properties** page, as shown in Figure 7. With WebSphere Application Server 8, you configure them in the **Alternate server names** field and the **Alternate port numbers** field on the **WebSphere Application Server data source properties** page. You access this page through the WebSphere Application Server 8 administration console by selecting the data source and then clicking **WebSphere Application Server data source properties**. See Figure 8:

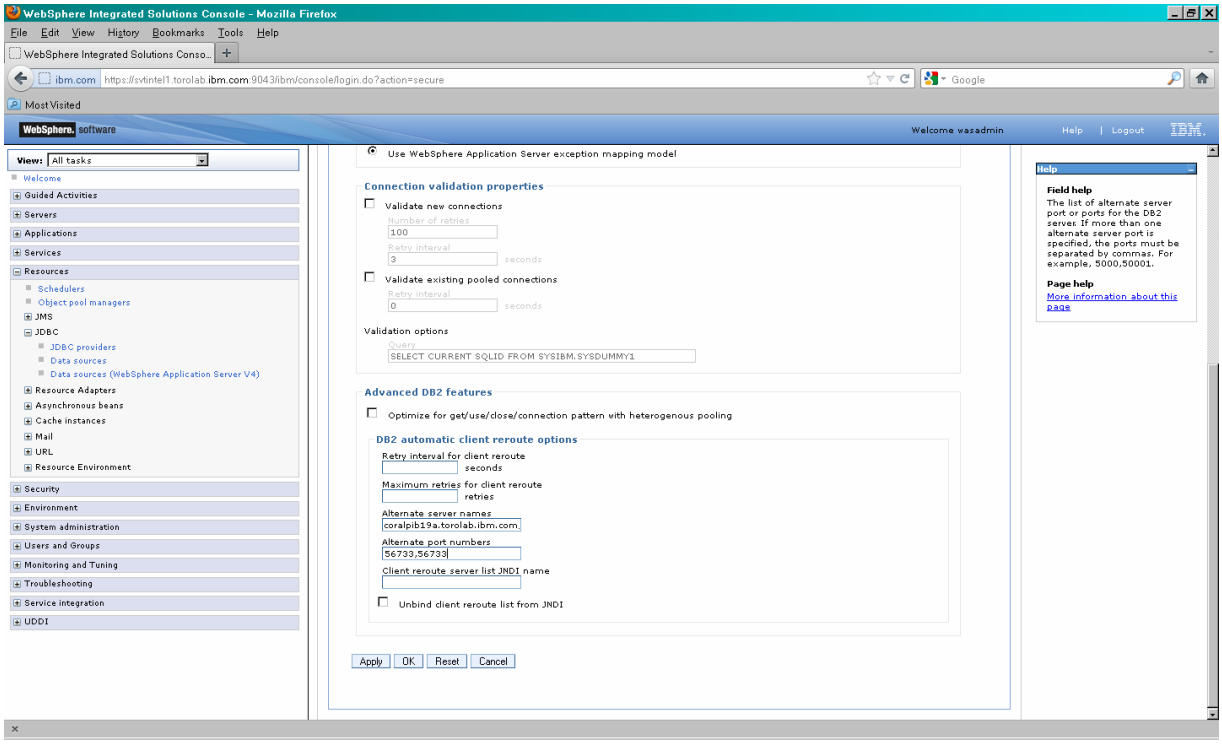


Figure 8. *Set alternate server names and alternate port numbers fields in WebSphere Application Server 8*

4 Enabling WLB and client affinities for stand-alone Java applications

DB2 stand-alone Java applications (applications that are invoked by the java command directly) with the supported IBM Data Server Driver for JDBC and SQLJ (JCC driver) can use the WLB or client affinities features to access databases on a DB2 pureScale cluster. This section describes how to enable WLB and client affinities for stand-alone Java applications.

4.1 Enabling WLB for stand-alone Java applications

Stand-alone Java applications must use a JCC Type 4 connectivity URL to connect to databases on DB2 pureScale. To enable WLB, set the `enableSysplexWLB` property either in the URL or in the application code. ACR is enabled implicitly if you enable WLB.

If the stand-alone Java application supports configuring a JCC Type 4 URL in an application property file or in command-line arguments, you can add `enableSysplexWLB=true` to the JCC Type 4 URL to enable WLB. The URL must end with a semicolon (;), see the following example. In this case, no application changes are required.

```
jdbc:db2://HostName:PortNumber/DatabaseName:existing-jcc-
properties;enableSysplexWLB=true;
```

- **HostName.** Specify the full hostname of a DB2 pureScale member; this can be any member in the DB2 pureScale cluster.
- **PortNumber.** Specify the TCP/IP port number that the DB2 pureScale member listens to for connection requests.
- **DatabaseName.** Specify the destination database for the application connections.
- **enableSysplexWLB.** Set to true.

For example, the following JCC Type 4 URL specifies that the Java application is to connect to database eComHQ on the DB2 pureScale member coralpib19a.torolab.ibm.com at TCP/IP port 56733 with WLB enabled.

```
jdbc:db2://coralpib19a.torolab.ibm.com:56733/eComHQ:enableSysplexWLB=true
;
```

If you cannot provide a JCC Type 4 URL in a property file or in command-line arguments, you can set the `enableSysplexWLB` property inside the application code, as shown in the following example:

```
…
String url = jdbc:db2://coralpib19a.torolab.ibm.com:56733/eComHQ;

Properties properties = new Properties();
properties.put("user", "yourID");
properties.put("password", "yourPassword");
properties.put("enableSysplexWLB", "true");

Connection con = DriverManager.getConnection( url, properties );
…
```

4.2 Enabling client affinities for stand-alone Java applications

The best way to enable the client affinities feature for stand-alone Java applications is to define the required JCC data source properties in the db2dsdriver.cfg file. Applications then access this file through a new JCC Type 4 connection URL through the `dsdriverConfigFile` property:

```
jdbc:db2:///DatabaseName:dsdriverConfigFile=value;other-jcc-properties;
```

Alternatively, you can enable the client affinities feature by setting the required JCC data source properties in the legacy JCC Type 4 connection URL, as shown in the following example:

```
jdbc:db2://HostName:PortNumber/DatabaseName:required-jcc-properties-for-
client-affinities;other-jcc-properties;
```

Neither method requires any Java application changes if the applications support customizing the JCC Type 4 connection URL inside a property file.

The advantage of using the db2dsdriver.cfg file is that the client affinities properties for all client machines are kept in a single file that you can package with the application. Therefore, different client machines running the same application package can use different alternate server lists to access the same DB2 pureScale cluster, depending on the definition in the db2dsdriver.cfg file.

By contrast, the alternate server list that you set in a legacy JCC Type 4 URL is used by all the client machines that run the same application with the same URL. This means that all the client machines use the same alternate server list to access the same DB2 pureScale cluster. If you want different client machines to use different alternate server lists, you must set different values for the `clientRerouteAlternateServerName` property in the JCC Type 4 URL, which requires the use of different application packages.

4.2.1 Enabling client affinities for stand-alone Java applications by using the db2dsdriver.cfg file

To enable the client affinities feature for stand-alone Java applications by using the db2dsdriver.cfg file, add the following parameters or sections to the <ACR> section of the db2dsdriver.cfg file, and provide the file to the applications using the new JCC Type 4 URL:

- **enableACR.** Set to `true`.
- **enableSeamlessAcr.** Set to `true`.
- **enableAlternateServerListFirstConnect.** Set to `false`.
- **affinityFailbackInterval** (optional). Set the interval in seconds for retrying failback. The default value is -1 (no failback).
- **maxRetriesForClientReroute** (optional). Set the maximum ACR retry number. The default value is 3.
- **retryIntervalForClientReroute** (optional). Set the seconds to wait between ACR retries. The default is no wait.
- **<alternate_server_list>.** Define a list of server names and ports. The first server that you define in this section has index 0, the second server has index 1, and so on.
- **<affinity_list>.** Define the server lists that clients can use, where a server means a DB2 pureScale member.
- **<client_affinity_defined>.** Define clients that use server lists to access servers.
- **<client_affinity_round_robin>.** Define clients that access servers in a round-robin way. The first client that you define in this section has index 0, the second client has index 1, and so on. Assume that the number of servers that you define for the <alternate_server_list> section is *n*. Therefore, the first server to be accessed by a client with index *i* is the server that you defined with index (*i* mod *n*) for the <alternate_server_list> section. If the first server is inaccessible, the next server for client *i* to access is the one with index ((*i*+1) mod *n*). This behavior will be illustrated through an example later.

The client affinities feature is enabled if you define either the `<client_affinity_defined>` or the `<client_affinity_round_robin>` section. You can define both the `<client_affinity_defined>` and the `<client_affinity_round_robin>` sections in the same db2dsdriver.cfg file.

The following sample db2dsdriver.cfg file shows how to enable the client affinities feature for stand-alone Java applications:

```
<configuration>

   <dsncollection>
      <dsn alias="eComHQ" name="eComHQ"
host="coralpib19a.torolab.ibm.com" port="56733">
         <parameter name="Authentication" value="Server_Encrypt"/>
      </dsn>
   </dsncollection>

   <databases>
      <database name="eComHQ" host="coralpib19a.torolab.ibm.com"
port="56733">
         <ACR>
            <parameter name="enableACR" value="true"/>
            <parameter name="enableSeamlessAcr" value="true"/>
            <parameter name="enableAlternateServerListFirstConnect"
value="false"/>
            <parameter name="affinityFailbackInterval" value="120"/>
            <parameter name="maxRetriesForClientReroute" value="3"/>
            <parameter name="retryIntervalForClientReroute" value="2"/>
            <alternate_server_list>
               <server name="server1"
hostname="coralpib19a.torolab.ibm.com" port="56733"></server>
               <server name="server2"
hostname="coralpib19b.torolab.ibm.com" port="56733"></server>
               <server name="server3"
hostname="coralpib19c.torolab.ibm.com" port="56733"></server>
            </alternate_server_list>
            <affinity_list>
               <list name="list1"
serverorder="server2,server1,server3"></list>
               <list name="list2"
serverorder="server1,server2,server3"></list>
               <list name="list3"
serverorder="server3,server2,server1"></list>
            </affinity_list>
            <client_affinity_defined>
               <client name="client1"
hostname="svtintel2.torolab.ibm.com" listname="list1"></client>
               <client name="client2" hostname="apu.torolab.ibm.com"
listname="list2"></client>
               <client name="client3" hostname="swat.torolab.ibm.com"
listname="list3"></client>
            </client_affinity_defined>
            <client_affinity_round_robin>
               <client name="client4"
hostname="utl41.torolab.ibm.com"></client>
               <client name="client5"
hostname="carat.torolab.ibm.com"></client>
            </client_affinity_round_robin >
         </ACR>
      </database>
   </databases>
```

```
   <parameters>
      <parameter name="CommProtocol" value="TCPIP"/>
   </parameters>

</configuration>
```

If you place the configuration file in the C:\cfg directory, for example, the following new JCC Type 4 URL is passed to the Java applications. You must end the URL string with a semicolon.

```
jdbc:db2:///eComHQ:dsdriverConfigFile=C:\cfg\db2dsdriver.cfg;
```

Using the sample configuration, when Java applications running on client machine svtintel2.torolab.ibm.com (that is, client1 as defined in the <client_affinity_defined> section) attempt to access the DB2 pureScale database eComHQ, the client machine uses the list1 server list. This list defines the server order as server2, server1, server3. Therefore, the first server to be connected to is server2, which is coralpib19b.torolab.ibm.com, instead of the server (coralpib19a.torolab.ibm.com) that is specified in the <database> section. If server2 goes down because of a planned or unplanned outage, the application requests are rerouted to the second server (server1) on the list. When server2 comes online again, connections are failed back to server2 if a transaction boundary (a commit or rollback) is reached and the time interval of 120 seconds has expired.

According to the configuration file, Java applications running on client machine carat.torolab.ibm.com (that is, client5 in the <client_affinity_round_robin> section) can attempt to access the pureScale database eComHQ. When this happens, because client5 has index 1 in the <client_affinity_round_robin> section, the first server for client5 to connect to is the one with index 1 (*1* mod *3*), which is server2 in the <alternate_server_list> section.

Client affinities failback for JCC applies only to the primary server (that is, the first server in the server list for a client). Assume that the previously shown db2dsdriver.cfg file applies. The server list for client1 is server2, server1, server3. Therefore, server2 is the primary server for client1. Now, assume that both server2 and server1 are offline, so client1 connects to server3. When server1 comes back online, the JCC client does not fail back the client1 connections from server3 to server1. The failback happens only when server2 comes back online. Even though server1 is not qualified for failback, it is qualified for client reroute activities. Thus, if server1 is online, client connections can be rerouted to server1 if server3 goes down.

The `maxRetriesForClientReroute` and `retryIntervalForClientReroute` parameters are used in this example as follows. Suppose that a communication failure occurs during a connection attempt from client1 (svtintel2.torolab.ibm.com) to the primary server, which is server2 (coralpib19b.torolab.ibm.com). The following output shows the connection retry behavior when the `maxRetriesForClientReroute` parameter is set to 3 and the `retryIntervalForClientReroute` parameter is set to 2:

```
The JCC driver tries to connect to server2 (coralpib19b.torolab.ibm.com)
port 56733.
The connection to server2 (coralpib19b.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
The JCC driver tries to connect to server2 (coralpib19b.torolab.ibm.com)
port 56733.
The connection to server2 (coralpib19b.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
```

```
The JCC driver tries to connect to server2 (coralpib19b.torolab.ibm.com)
port 56733.
The connection to server2 (coralpib19b.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
The JCC driver tries to connect to server1 (coralpib19a.torolab.ibm.com)
port 56733.
The connection to server1 (coralpib19a.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
The JCC driver tries to connect to server1 (coralpib19a.torolab.ibm.com)
port 56733.
The connection to server1 (coralpib19a.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
The JCC driver tries to connect to server1 (coralpib19a.torolab.ibm.com)
port 56733.
The connection to server1 (coralpib19a.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
The JCC driver tries to connect to server3 (coralpib19c.torolab.ibm.com)
port 56733.
The connection to server3 (coralpib19c.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
The JCC driver tries to connect to server3 (coralpib19c.torolab.ibm.com)
port 56733.
The connection to server3 (coralpib19c.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
The JCC driver tries to connect to server3 (coralpib19c.torolab.ibm.com)
port 56733.
The connection to server3 (coralpib19c.torolab.ibm.com) port 56733 fails.
The JCC driver waits two seconds.
The JCC driver throws an SQLException with error code -4499.
```

4.2.2 Enabling client affinities for stand-alone Java applications in a JCC Type 4 connection URL or inside application code

To enable the client affinities feature for stand-alone Java applications without using a db2dsdriver.cfg file, set the following JCC data source properties in the JCC Type 4 connection URL or inside the application code:

- **enableClientAffinitiesList.** Set to 1.
- **clientRerouteAlternateServerName.** Specify a comma-separated list of server names.
- **clientRerouteAlternatePortNumber.** Specify a comma-separated list of port numbers.
- **affinityFailbackInterval.** Set the interval in seconds for retrying failback.
- **enableSeamlessFailover.** Set to 1.
- **maxRetriesForClientReroute** (optional). Set the maximum ACR retry number.
- **retryIntervalForClientReroute** (optional). Set the seconds to wait between ACR retries.

The following example string shows how to pass these properties to a JCC Type 4 connection URL:

```
jdbc:db2://coralpib19a.torolab.ibm.com:56733/eComHQ:enableClientAffinitiesList=1;affinityFailbackInterval=120;clientRerouteAlternateServerName=coralpib19b.torolab.ibm.com,coralpib19a.torolab.ibm.com;clientRerouteAlternatePortNumber=56733,56733;enableSeamlessFailover=1;maxRetriesForClientReroute=3;retryIntervalForClientReroute=2;
```

When a stand-alone Java application uses this JCC Type 4 URL to connect to the sample eComHQ database on a DB2 pureScale cluster, the client connects to the first server that is specified for the `clientRerouteAlternateServerName` property, That is, the client connects to coralpib19b.torolab.ibm.com instead of coralpib19a.torolab.ibm.com that is specified as the connect member in the URL. After the client is connected, all the application requests go to coralpib19b.torolab.ibm.com. If coralpib19b.torolab.ibm.com goes offline because of a planned or unplanned outage, the application requests are rerouted to the second server on the list, coralpib19a.torolab.ibm.com. When the primary server, coralpib19b.torolab.ibm.com, comes back online, connections are failed back to the primary server if a transaction boundary (a commit or rollback) is reached and the time interval of 120 seconds has expired.

5 Enabling WLB and client affinities for non-Java applications

This section describes how to enable WLB and client affinities for non-Java applications. In this paper, non-Java applications are the applications that access DB2 pureScale databases using DB2 ODBC, CLI, .NET, OLE DB, embedded SQL, and command line processor (CLP) interfaces.

5.1 Enabling WLB for DB2 ODBC, CLI, .NET, and OLE DB applications

DB2 ODBC, CLI, .NET, and OLE DB applications running on the following DB2 client products (Version 9.7 Fix Pack 1 or later) can use WLB feature to access DB2 pureScale databases:

- The IBM® Data Server Driver Package product (DB2 DSDRIVER)
- The IBM Data Server Driver for ODBC and CLI product (DB2 DSDRIVER)
- The IBM Data Server Client product (an instance-based DB2 client)
- The IBM Data Server Runtime Client product (an instance-based DB2 client)

To enable WLB, you must set the enableWLB parameter to true in the db2dsdriver.cfg file on the DB2 client. Seamless ACR is enabled implicitly if you enable WLB, so this section shows only how to enable WLB for the clients.

5.1.1 Enabling WLB on DB2 DSDRIVER clients

To enable WLB on DB2 DSDRIVER clients (that is, for the IBM Data Server Driver Package or the IBM Data Server Driver for ODBC and CLI product), set the `enableWLB` parameter to true in the `db2dsdriver.cfg` file, and place the `db2dsdriver.cfg` file in the correct directory. For more

information about the db2dsdriver configuration file, see the additional resources listed in Appendix B. Because DB2 DSDRIVER clients do not have catalog command support, you can also specify the database connection information in the <dsncollection> section of the db2dsdriver.cfg file. Following is a sample db2dsdriver.cfg file that shows how to specify the database connection information and enable WLB. The db2dsdriver.cfg file is located in the C:\Documents and Settings\All Users\Application Data\IBM\DB2\IBMDBCL1\cfg directory on a Microsoft Windows system if the default DB2 installation options are used.

```
<configuration>
  <dsncollection>
    <dsn alias="BComHQ" name="BComHQ" host="coralpib19a" port="56733">
      <parameter name="Authentication" value="Server_Encrypt"/>
    </dsn>
  </dsncollection>
  <databases>
    <database name="BComHQ" host="coralpib19a" port="56733">
      <WLB>
        <parameter name="enableWLB" value="true"/>
      </WLB>
    </database>
  </databases>
  <parameters>
    <parameter name="CommProtocol" value="TCPIP"/>
  </parameters>
</configuration>
```

5.1.2 Enabling WLB on instance-based DB2 clients

To enable WLB on instance-based DB2 clients (that is, on an IBM Data Server Client or an IBM Data Server Runtime Client), set the enableWLB property to true in the db2dsdriver.cfg file, and place the db2dsdriver.cfg file in the correct directory. For more information regarding the db2dsdriver configuration file, see the additional resources in Appendix B. Because the instance-based DB2 clients have catalog command support, you can catalog the databases on the DB2 clients directly, instead of specifying the database connection information in the <dsncollection> section of the db2dsdriver.cfg file.

Following is a simple two-step example of how to enable WLB on an instance-based DB2 client. In this example, it is assumed that you installed the DB2 client product on a Windows operating system as DB2COPY1 using the default DB2 installation options.

1. Catalog the DB2 pureScale server and database on the DB2 client as shown:

```
db2 catalog tcpip node npib19a remote coralpib19a server 56733
db2 catalog db BComHQ as BComHQ at node npib19a authentication server_encrypt
db2 terminate
```

2. Create a file named db2dsdriver.cfg with the following content and place it in the correct directory. On a typical Windows client, the path to use is C:\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY1\DB2\cfg. The catalog connectivity information that you provided in Step 1 and the connectivity information for the database entry in the following db2dsdriver.cfg file must match.

```
<configuration>
```

```
  <databases>
    <database name="BComHQ" host="coralpib19a" port="56733">
      <WLB>
        <parameter name="enableWLB" value="true"/>
      </WLB>
    </database>
  </databases>
  <parameters>
    <parameter name="CommProtocol" value="TCPIP"/>
  </parameters>
</configuration>
```

5.2 Enabling WLB for DB2 embedded SQL and command line processor applications

Starting with DB2 Version 9.8 Fix Pack 2, DB2 embedded SQL and command line processor (CLP) applications that run on the IBM Data Server Client or IBM Data Server Runtime Client product with Version 9.7 Fix Pack 1 or later can use transaction-level WLB if they meet certain criteria, see the additional resources listed in Appendix B. To enable WLB for these applications, in addition to setting the `enableWLB` parameter to `true` in the db2dsdriver.cfg file, you must rebind the applications or packages with the `KEEPDYNAMIC NO` parameter. For instructions on how to set the `enableWLB` property in the db2dsdriver.cfg file, see the previous section.

The following example shows how to rebind the DB2 CLP packages with the `KEEPDYNAMIC NO` parameter. You must run the bind commands after connecting to the DB2 pureScale database from the IBM Data Server Client or IBM Data Server Runtime Client product (with Version 9.7 Fix Pack 1 or later).

```
C:
cd C:\Program Files\IBM\SQLLIB\bnd
db2 connect to <pureScale database> user <UID> using <PWD>
db2 "bind db2clpcs.bnd SQLERROR CONTINUE blocking all action replace
KEEPDYNAMIC NO"
db2 "bind db2clprr.bnd SQLERROR CONTINUE blocking all action replace
KEEPDYNAMIC NO"
db2 "bind db2clpur.bnd SQLERROR CONTINUE blocking all action replace
KEEPDYNAMIC NO"
db2 "bind db2clprs.bnd SQLERROR CONTINUE blocking all action replace
KEEPDYNAMIC NO"
db2 "bind db2clpnc.bnd SQLERROR CONTINUE blocking all action replace
KEEPDYNAMIC NO"
db2 terminate
```

5.3 Enabling client affinities for non-Java applications

You can enable client affinities for non-Java applications only through the db2dsdriver.cfg file. The parameters and sections that you must define in the db2dsdriver.cfg file are the same as those for the stand-alone Java applications, except a few parameters have different names:

- enableACR is changed to enableAcr.
- maxRetriesForClientReroute is changed to maxAcrRetries.
- retryIntervalForClientReroute is changed to acrRetryInterval.

DB2 non-Java clients and the JCC driver deal with client affinities failback slightly differently. In the JCC case, failback happens only to the primary server (see the section “Enabling client affinities for stand-alone Java applications by using the db2dsdriver.cfg file”). However, for a non-Java client, failback can happen to the primary server and alternate servers.

The following db2dsdriver.cfg file has the same client affinities definition that is described in the section “Enabling client affinities for stand-alone Java applications by using the db2dsdriver.cfg file”. You must disable transaction-level WLB for the client affinities feature to work. Therefore, either remove the <WLB> entry or set the enableWLB property to false.

```
<configuration>

   <dsncollection>
      <dsn alias="eComHQ" name="eComHQ"
host="coralpib19a.torolab.ibm.com" port="56733">
         <parameter name="Authentication" value="Server_Encrypt"/>
      </dsn>
   </dsncollection>

   <databases>
      <database name="eComHQ" host="coralpib19a.torolab.ibm.com"
port="56733">
         <ACR>
            <parameter name="enableAcr" value="true"/>
            <parameter name="enableSeamlessAcr" value="true"/>
            <parameter name="enableAlternateServerListFirstConnect"
value="false"/>
            <parameter name="affinityFailbackInterval" value="120"/>
            <parameter name="maxAcrRetries" value="3"/>
            <parameter name="acrRetryInterval" value="2"/>
            <alternate_server_list>
               <server name="server1"
hostname="coralpib19a.torolab.ibm.com" port="56733"></server>
               <server name="server2"
hostname="coralpib19b.torolab.ibm.com" port="56733"></server>
               <server name="server3"
hostname="coralpib19c.torolab.ibm.com" port="56733"></server>
            </alternate_server_list>
            <affinity_list>
               <list name="list1"
serverorder="server2,server1,server3"></list>
               <list name="list2"
serverorder="server1,server2,server3"></list>
               <list name="list3"
serverorder="server3,server2,server1"></list>
            </affinity_list>
            <client_affinity_defined>
               <client name="client1"
hostname="svtintel2.torolab.ibm.com" listname="list1"></client>
```

```
               <client name="client2" hostname="apu.torolab.ibm.com"
listname="list2"></client>
               <client name="client3" hostname="swat.torolab.ibm.com"
listname="list3"></client>
            </client_affinity_defined>
            <client_affinity_round_robin>
               <client name="client4"
hostname="utl41.torolab.ibm.com"></client>
               <client name="client5"
hostname="carat.torolab.ibm.com"></client>
            </client_affinity_round_robin >
         </ACR>
      </database>
   </databases>

   <parameters>
      <parameter name="CommProtocol" value="TCPIP"/>
   </parameters>

</configuration>
```

6 Configuring alternate servers for the first connection

You normally specify the connection information (for example, the database name, the server name, and the TCP/IP port number) of the primary server for the first connection in one of the following places:

- An application connection string
- A JCC Type 4 URL or properties string
- The catalog entries
- The <dsn> section of the db2dsdriver.cfg file

When that first connection is made to the primary server, the DB2 pureScale cluster returns a list of all members along with their priorities to the DB2 client in a server list. The DB2 client relies on this server list to distribute or balance application requests among the available DB2 pureScale members based on the priorities of the members.

However, if applications fail to make the first connection to the primary DB2 pureScale member because that member that is specified in the connection information is offline, the DB2 client does not receive a server list. Thus, the DB2 client has no way to know what other DB2 pureScale members are available. Therefore, making a successful first connection to a DB2 pureScale member is very important for WLB to work in a DB2 pureScale environment. DB2 provides a mechanism to allow DB2 clients to retry the first connection to the specified alternate servers if the specified primary server is unreachable. In DB2 Version 9.7 Fix Pack 1 or later, you can use the enableAlternateServerListFirstConnect feature on both DB2 Java clients and the DB2 non-Java clients for this purpose. On DB2 Java clients, you can also specify the alternate servers through the legacy JCC clientRerouteAlternateServerName and clientRerouteAlternatePortNumber properties.

You can use either the enableAlternateServerListFirstConnect feature or the `clientRerouteAlternateServerName` and `clientRerouteAlternatePortNumber` properties to configure the alternate servers before you start the applications. The DB2 client loads this preconfigured alternate server list into memory when the application is started and tries to open the first connection. The list is used only for this particular application process if the client fails to make the first connection to the specified primary server. This list requires manual maintenance to ensure that it contains the current members of your DB2 pureScale cluster.

To free you from manually maintaining the alternate server list, a feature called server list cache was introduced for DB2 non-Java clients in Version 9.7 Fix Pack 4. A similar feature was introduced for DB2 Java clients in Version 9.7 Fix Pack 3, but as of Version 9.7 Fix Pack 5, this feature does not support WLB.

With the server list cache feature, a DB2 client can maintain an up-to-date server list based on the information that the DB2 pureScale cluster returns. This information allows the DB2 client to rebuild an alternate server list based on the last known DB2 pureScale cluster configuration. Multiple application processes can share the alternate server list if the applications are running on the same DB2 client and are connecting to the same database on the same DB2 pureScale cluster. For example, consider an application A in such an environment that makes the first connection to the DB2 pureScale cluster and creates a cached server list. The applications that are subsequently started, such as B and C, share the server list cache file that application A created and do not have to provide their own alternate server lists.

6.1 Enabling the enableAlternateServerListFirstConnect feature

To enable the enableAlternateServerListFirstConnect feature, you must set the `enableAlternateServerListFirstConnect` parameter to true and define the alternate server list in the db2dsdriver.cfg file. The following example shows how to do this on DB2 non-Java clients:

```
<configuration>

   <dsncollection>
      <dsn alias="eComHQ" name="eComHQ"
host="coralpib19a.torolab.ibm.com" port="56733">
         <parameter name="Authentication" value="Server_Encrypt"/>
      </dsn>
   </dsncollection>

   <databases>
      <database name="eComHQ" host="coralpib19a.torolab.ibm.com"
port="56733">
         <WLB>
            <parameter name="enableWLB" value="true"/>
         </WLB>
         <ACR>
            <parameter name="enableAcr" value="true"/>
            <parameter name="enableSeamlessAcr" value="true"/>
            <parameter name="enableAlternateServerListFirstConnect"
value="true"/>
            <parameter name="maxAcrRetries" value="3"/>
            <parameter name="acrRetryInterval" value="2"/>
            <alternate_server_list>
               <server name="alternateserver1"
hostname="coralpib19b.torolab.ibm.com" port="56733"></server>
```

```
               <server name="alternateserver2"
hostname="coralpib19c.torolab.ibm.com" port="56733"></server>
            </alternate_server_list>
         </ACR>
      </database>
   </databases>

   <parameters>
      <parameter name="CommProtocol" value="TCPIP"/>
   </parameters>
</configuration>
```

In this sample db2dsdriver.cfg file, the alternate servers are defined in the <alternate_server_list> section. Member coralpib19a.torolab.ibm.com is specified as the initial connect member (that is, the primary server), and there are two alternate servers (coralpib19b.torolab.ibm.com and coralpib19c.torolab.ibm.com). If the application fails to make the first connection to coralpib19a.torolab.ibm.com, the DB2 client tries to connect to coralpib19b.torolab.ibm.com because that is the first alternate server in the <alternate_server_list> section. If the connection succeeds, the DB2 client gets the server list returned by the DB2 pureScale cluster and distributes application requests based on that information. Any subsequent client reroute operations are based on the available members in the server list that the server returns. If the application fails to connect to the first alternate server, the DB2 client tries to connect to the second alternate server (coralpib19c.torolab.ibm.com) in the <alternate_server_list> section. If the client cannot connect to the second alternate server, the client continues to work through the list of possible alternate servers until the list is exhausted.

You can use this same configuration for DB2 Java clients, except that a few parameters have different names:

- enableAcr is changed to enableACR.
- maxAcrRetries is changed to maxRetriesForClientReroute.
- acrRetryInterval is changed to retryIntervalForClientReroute.

6.2 Enabling the server list cache feature

For DB2 non-Java clients, the server list cache feature is enabled if you do not define the alternate server list in the db2dsdriver.cfg file when the enableAlternateServerListFirstConnect parameter is set to true. You can use the db2dsdriver.cfg file that is shown in the prior example, but remove the <alternate_server_list> section.

For DB2 Java clients, the server list feature is enabled when the JCC driver can find a directory location into which to write the jccServerListCache.bin cache file. You can set this directory location through the db2.jcc.outputDirectory JCC property or through the java.io.tmpdir Java system property.

You can set the db2.jcc.outputDirectory property as follows:

- In the DB2JccConfiguration.properties JCC properties file, as shown in the following example:

```
db2.jcc.outputDirectory=/home/tmp
```

- As a Java system property, by specifying –D*property=value* when executing the Java command, as shown in the following example:

```
java -Ddb2.jcc.outputDirectory=/home/tmp Test
```

7 Enabling fast detection for unresponsive TCP/IP connections

Automatic client rerouting happens when DB2 clients detect communication failures. If communication failures are caused by a network failure, a hardware failure, or a power outage, the TCP/IP layer might not return an error to the DB2 client until the TCP/IP timeout occurs. The default timeout on many operating systems is 2 hours. You can reconfigure the system-wide default TCP/IP setting on DB2 client machines so that unresponsive TCP/IP socket connections are detected sooner, but this change affects all TCP/IP traffic on the DB2 clients. To get the same effect without affecting other TCP/IP traffic, you can use the DB2 `keepAliveTimeout` configuration parameter on DB2 non-Java clients or use the JCC `keepAliveTimeOut` or `blockingReadConnectionTimeout` property on DB2 Java clients. Using one of these DB2 client-side configuration techniques, you can configure how quickly you want DB2 to detect unresponsive TCP/IP connections and initiate ACR. This is vital if you want your application requests rerouted to the online DB2 pureScale members as quickly as possible if the current connected DB2 pureScale member goes offline.

7.1 Configuring the keepAliveTimeout parameter on DB2 non-Java clients

You can configure the `keepAliveTimeout` parameter in the db2dsdriver.cfg file on DB2 non-Java clients. Setting the `keepAliveTimeout` parameter in the db2dsdriver.cfg file is recommended because you can use this configuration for both instance-based clients (that is, the IBM Data Server Client or IBM Data Server Runtime Client product) and DB2 DSDRIVER clients (that is, the IBM Data Server Driver Package or IBM Data Server Driver for ODBC and CLI product). In addition, by using the `db2dsdriver.cfg` file, each database can have a different `keepAliveTimeout` setting.

The value of 15 seconds in the following sample db2dsdriver.cfg file is the suggested value and the default value for DB2 Version 10.1 or later, but you should set this value based on your specific network and server capabilities. The `keepAliveTimeout` parameter line must be within the `<database>` section but outside the `<acr>` section.

```
    <configuration>
        <dsncollection>
           <dsn alias="eComHQ" name="eComHQ"
host="coralpib19a.torolab.ibm.com" port="56733" />
        </dsncollection>
        <databases>
           <database name="eComHQ" host="coralpib19a.torolab.ibm.com"
port="56733">
```

```
            <parameter name="keepAliveTimeout" value="15"/>
    <wlb>
            <parameter name="enableWLB" value="true"/>
          </wlb>
          <acr>
             <parameter name="enableAcr" value="true"/>
             <parameter name="enableSeamlessAcr" value="true"/>
          </acr>
       </database>

     </databases>
   ...
   </configuration>
```

The keepAliveTimeout parameter applies only when an application already has a TCP/IP connection. To quickly detect TCP/IP socket failures when attempting to open a TCP/IP connection, applications running on DB2 non-Java clients can use the tcpipConnectTimeout parameter to specify the number of seconds before an attempt to open a TCP/IP socket fails. You can configure the tcpipConnectTimeout parameter in the same way as you configure the keepAliveTimeout parameter. The recommended tcpipConnectTimeout parameter value is 1 - 5 seconds.

7.2 Configuring the keepAliveTimeOut and blockingReadConnectionTimeout properties on DB2 Java clients

Both keepAliveTimeOut and blockingReadConnectionTimeout are JCC DataSource properties and apply to JCC Type 4 connections only. The keepAliveTimeOut property has been available since DB2 Version 9.7 Fix Pack 5. In general, the property requires IBM JDK 6.0.0 SR10 or later. However, if you are using the WebSphere Application Server product with IBM JDK 6.0.0 SR9 FP1, you can also use the feature after applying a WebSphere Application Server iFix through WebSphere Application Server APAR PM41525. The blockingReadConnectionTimeout property is available for the DB2 client levels (Version 9.7 Fix Pack 1 or later) that are recommended for the DB2 pureScale server.

The difference between the keepAliveTimeOut and blockingReadConnectionTimeout properties is that the keepAliveTimeOut property detects TCP/IP communication failures at layers lower than the JCC driver. It does this by sending TCP/IP keep-alive probes. The keepAliveTimeOut property is more granular than the blockingReadConnectionTimeout property, for which you specify a socket read timeout setting at the JCC driver layer. To detect unresponsive TCP/IP connections, using the keepAliveTimeOut property is recommended if your JCC driver and JDK levels support it.

You can set the properties through a JCC Type 4 connection URL or inside the Java application code, as shown in the following examples. Set the values based on your environment.

- The following example shows how to set the keepAliveTimeOut property through a URL:

```
jdbc:db2://coralpib19a.torolab.ibm.com:56733/eComHQ:enableSysplexWLB=true
;keepAliveTimeOut=15;
```

- The following example shows how to set the `blockingReadConnectionTimeout` property through a URL:

```
jdbc:db2://coralpib19a.torolab.ibm.com:56733/eComHQ:enableSysplexWLB=true
;blockingReadConnectionTimeout=60;
```

- The following examples show how to set the properties in Java application code:

```
…
String  url = jdbc:db2://coralpib19a.torolab.ibm.com:56733/eComHQ;

Properties properties = new Properties();
properties.put("user", "yourID");
properties.put("password", "yourPassword");
properties.put("enableSysplexWLB", "true");
properties.put("keepAliveTimeOut", "15");


Connection con = DriverManager.getConnection( url, properties );
…
```

```
…
String url = jdbc:db2://coralpib19a.torolab.ibm.com:56733/eComHQ;

Properties properties = new Properties();
properties.put("user", "yourID");
properties.put("password", "yourPassword");
properties.put("enableSysplexWLB", "true");
properties.put("blockingReadConnectionTimeout", "60");


Connection con = DriverManager.getConnection( url, properties );
…
```

The `keepAliveTimeOut` and `blockingReadConnectionTimeout` properties apply only when an application already has a TCP/IP connection.

DB2 Java clients do not support the `tcpipConnectTimeout` property as of Version 9.7 Fix Pack 5. Applications running on DB2 Java clients can use the `loginTimeout` JCC connection or DataSource property to detect communication failures at the JCC layer when attempting to open a TCP/IP connection against a data source. This property specifies the maximum time in seconds to wait for a new connection to a data source. If this maximum time elapses, the JCC driver throws a connection exception to the application. The default value for the `loginTimeout` property is `0`, which means that the timeout value is the default system timeout value. The recommended value is 5 seconds. You can configure the `loginTimeout` property in the same way as you configure the `keepAliveTimeOut` and `blockingReadConnectionTimeout` properties.

8 Conclusion

The DB2 pureScale Feature for Enterprise Server Edition provides a database solution that meets the needs of the most demanding customers. It is designed to scale effectively to meet the growing and dynamic needs of different organizations. You can start additional members in the DB2 pureScale environment without any impact to existing applications to meet the demands of peak processing times. The DB2 pureScale Feature automatically balances the workload across all DB2 members in the cluster without any changes at the application side, taking full advantage of the additional processing capacity. Alternatively, you can direct applications to specific DB2 pureScale members based on business requirements. If a DB2 member fails, applications are automatically routed among the other active members. When the failed member comes back online, applications are transparently routed to the restarted member.

APPENDIX

Appendix A. Supported software levels

To use the WLB, ACR, and client affinities features to access DB2 pureScale, you require a supported DB2 client from DB2 Version 9.7 Fix Pack 1 or later. For more information, see the links to the IBM DB2 Information Center in Appendix B.

The version of the IBM Data Server Driver for JDBC and SQLJ that was included with DB2 Version 9.7 Fix Pack 1 is 3.58 for JDBC3 and 4.8 for JDBC4.

You should use the latest Version 9.7 fix pack version of the DB2 clients to access the DB2 pureScale servers.

DB2 ODBC, CLI, .NET, OLE DB, embedded SQL, and command line processor (CLP) applications running on the following DB2 client products can use the WLB, ACR, and client affinities features to access a DB2 pureScale cluster:

- The IBM Data Server Driver Package product (DB2 DSDRIVER)
- The IBM Data Server Driver for ODBC and CLI product (DB2 DSDRIVER)
- The IBM Data Server Client product (an instance-based DB2 client)
- The IBM Data Server Runtime Client product (an instance-based DB2 client)

WLB is allowed for DB2 embedded SQL and CLP applications if they meet certain criteria. For more information, see the links to the IBM DB2 Information Center in Appendix B.

Java applications must use JCC Type 4 connectivity for WLB and client affinities to work.

Appendix B. Additional resources

- For information about Java client support of high availability (WLB, ACR, and client affinities) connections to DB2 for Linux, UNIX, and Windows servers, see the DB2 for Linux, UNIX, and Windows Information Center.
- For information about non-Java client support of high availability (WLB, ACR, and client affinities) connections to DB2 for Linux, UNIX, and Windows servers, see the DB2 for Linux, UNIX, and Windows Information Center.
- For information about DB2 client considerations with the DB2 pureScale Feature, see the DB2 for Linux, UNIX, and Windows Information Center
- For information about the db2dsdriver configuration file, see the DB2 for Linux, UNIX, and Windows Information Center
- For information about WLB support for embedded SQL and DB2 CLP applications that are bound with the `KEEPDYNAMIC NO` parameter, see the DB2 for Linux, UNIX, and Windows Information Center.

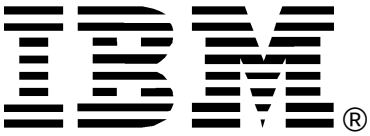
Contributors

Jason Woods
Staff Software Developer
Information Management

Jyh-Chen Fang
Software Developer
Information Management

Manish Sehgal
Software Developer
Information Management

Sherry Guo
Software Developer
Information Management



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 10504-1785
U.S.A.

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 PAPER “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 may be made periodically to the information herein; these changes may be incorporated in subsequent versions of the paper. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this paper at any time without notice.

Any references in this document 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 have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
4205 South Miami Boulevard
Research Triangle Park, NC 27709 U.S.A.

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

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

Trademarks