IBM® Information Management Software

High Availability and Disaster Recovery for Temenos T24 with IBM DB2 and AIX

T24 R11 on AIX with IBM WebSphere Application Server and WebSphere MQ

High availability and disaster recovery implementation

T24 R11 with DB2 9 and DB2 pureScale

Mike Barrie
Moditha Sanjay De Costa
Gowrimani Gopalakrishnan
Murali Neralla

Redpaper

ibm.com/redbooks

International Technical Support Organization

High Availability and Disaster Recovery for Temenos T24 with IBM DB2 and AIX

October 2012

REDP-4794-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (October 2012)

This edition applies to IBM DB2 for Linux, UNIX, and Windows Version 9.7 and 9.8 and Temenos T24 R11 with TAFC R11.

Contents

Notices

Trademarks

Preface

The Temenos T24 core banking application is a business critical application for all the banks that use it and has a primary focus on providing an appropriate level of high availability and disaster recovery. The level of availability that can be achieved is determined largely by the configuration of the infrastructure that supports T24. This infrastructure is built on hardware, middleware, and networking, in addition to the operational procedures and practices that are used to operate T24.

The role of middleware is critical to the delivery of high availability and disaster recovery. The middleware enables the configuration as a whole to continue operation when individual components fail or even a whole site becomes unavailable. IBM® middleware provides the necessary means for T24 users to achieve high levels of availability and provides a choice of strategies for disaster recovery.

Many options are available for meeting a client's high availability and disaster recovery requirements. The solution chosen by a Temenos T24 user depends on many factors. These factors include a user's detailed availability and recovery requirements; their existing datacenter standards, practices, and processes; and the available network infrastructure. Therefore, the optimum solution must be determined on a case-by-case basis for each deployment.

This IBM Redpaper™ publication serves as a guide to help IT architects and other technical staff who are designing, configuring, and building the infrastructure to support Temenos T24. It shows how IBM software can deliver high availability and disaster recovery for Temenos T24 to meet a client's requirements. This software might run on IBM AIX®, IBM WebSphere® Application Server, WebSphere MQ Server, and IBM DB2®. These IBM software components are typically used for a Temenos T24 deployment on an IBM middleware stack to ensure a highly available infrastructure for T24.

The team who wrote this paper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO).

Mike Barrie is an IBM IT Architect who is working as a Technology Executive in the IBM ISV and Developer Relations organization in the United Kingdom (UK). For the past three years, he has supported Temenos product development and has worked with many other IBM ISV partners. Before his current role, Mike worked as an IT Architect in IBM Global Services and achieved certification as an IT Architect with IBM and with the Open Group. Mike holds a degree in Natural Sciences from Cambridge University in the UK.

Moditha Sanjay De Costa is an independent Enterprise Architect Consultant based in the UK, with 20 years experience of designing and engineering banking platforms. His areas of specialization include the design and implementation of highly available real-time banking, trading, and market data solutions. He has in-depth experience with the major vendor products in the client-server, service-oriented architecture (SOA), messaging, and middleware markets. He has held senior enterprise architecture roles for many organizations, such as Temenos, Thomson Reuters, Deutsche Bank, NatWest Markets, and Lloyds Bank. Moditha Sanjay graduated with a Bachelor of Engineering Honors degree in Computer Systems and Electronic Engineering from Kings College of the University of London.

Gowrimani Gopalakrishnan is an Enterprise Database Manager for Temenos UK Ltd. For the past 10 years, she has specialized in information management on IBM database platforms for IBM System z®, Linux, UNIX, and Windows. She has experience with information management of Temenos T24 on Java and C implementations, in addition to the client-server, messaging, and middleware products of IBM. She has held senior information management roles for many organizations such as Temenos, HCL, Birlasoft, and Inautix. Gowrimani graduated with Master of Computer Applications degree with first class honors from the College of Engineering at Anna University in India.

Murali Neralla is a Senior Software Engineer in the IBM ISV Enablement organization. Murali currently works with the Financial Services Sector solution providers. He specializes in solution architecture, workload optimization, porting, capacity planning, and performance benchmarking on IBM Power Systems™ running AIX.

Thanks to the following people for their contributions to this project:

- The IBM Innovation Centre in Hursley, UK, including Louise Cooper, Christian Barr, Daniel Martin, Tim Poultney, Richard Feltham, and Rachel Denbigh for their help in setting up and running the test environment.
- Technical specialists at Temenos UK Ltd., including Ted jolly, Alexander Demin, C.S. Santhosh, Charles Lee, Julie Bennett, Lucas Gut, and Sheeraz Junejo.
- Whei-Jen Chen, Project Leader, and Emma Jacob, Graphics Specialist of the IBM ITSO

Now you can become a published author, too!

Comments welcome

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  `http://www.facebook.com/IBMRedbooks`

- Follow us on Twitter:

  `http://twitter.com/ibmredbooks`

- Look for us on LinkedIn:

  `http://www.linkedin.com/groups?home=&gid=2130806`

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  `https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm`

- Stay current on recent Redbooks publications with RSS Feeds:

  `http://www.redbooks.ibm.com/rss.html`

1

Scope of the test project

This IBM Redpaper publication highlights the available options for T24 users and provides guidance about which approaches to use with T24. The approaches were validated in a joint testing project that Temenos and IBM jointly conducted at the IBM Innovation Centre in Hursley, UK. This paper explains the configurations that were used and the test results to help you recreate and extend them to your environment for high availability and disaster recovery.

The configurations described in this document were validated by testing Temenos T24 with high availability and disaster recovery technologies. This testing was carried out at the IBM Innovation Centre in Hursley, United Kingdom with the support of the IBM Innovation Center staff.

Understanding of availability requirements: A high availability strategy begins with an understanding of the availability requirements, such as how the middleware is configured. Without an understanding of the strategy, it is impossible to make the informed judgements and trade-offs that you will need to make. Other major factors that determine the level of availability that can be achieved include the physical data center infrastructure, the choices and configuration of servers, storage and networking, and the operational procedures and practices that are used to run the data center. Therefore, this paper is just one part of an overall strategy for delivering high availability.

This chapter includes the following sections:

- Software levels used for the project
- Scope of the project and aspects considered out of scope

1.1 Software levels used for the project

The project was run by using T24 R11 and TAFC R11. The configurations were validated with the following IBM software release levels:

- IBM WebSphere Application Server Network Deployment Version 7.0.0.17
- IBM WebSphere MQ Server Version 7.0.1.6
- IBM AIX Version 7.1
- IBM DB2 for Linux, UNIX, and Windows Enterprise Edition Version 9.7
- IBM DB2 pureScale® Feature Version 9.8
- IBM PowerHA® Version 6 SP5

1.2 Scope of the project and aspects considered out of scope

The project had the following scope:

- Test the high availability of the T24 application with IBM middleware products and the DB2 9.7 and DB2 pureScale database.
- Test methods for implementing disaster recovery by using the T24 application with IBM middleware products and DB2 9.7 and DB2 pureScale database.

High availability (HA) and disaster recovery (DR) are sometimes thought of as synonymous with each other, but they are different concepts.

A highly available infrastructure component or IT system is described as having the ability to fail over. Therefore, these tests aim at proving that T24 continues to perform with minimal impact on failover on all the different layers.

Disaster recovery can be defined as, "Resources and activities to re-establish information technology services at an alternate site after a disruption of services due to a disaster." This test aims at proving that T24 can be performed in an alternate site if a disaster occurs in the primary site. Depending on the implementation strategy, the recovery time and effort will vary.

The following aspects of configuring for high availability and disaster recovery were not considered as part of this study, but must be taken into account in every actual T24 implementation:

- Availability of administrative processes, such as WebSphere Cell Management, because the generally do not affect the availability of the system for production work.
- Performance. The testing did not include any performance testing. Configuring for high availability can affect application performance, and high workloads can affect recovery times.
- Configuration and recovery strategies to mitigate failures of individual hardware components such as power supplies, network cards, and individual disks. With appropriate system configuration, often a system can survive component failures without triggering a failover to a backup system. Protection against component failure must be considered as part of normal configuration planning, for example, by using duplicate network cards, uninterruptible power supplies, and RAID disks.
- High availability for components outboard of the Java Platform, Enterprise Edition (J2EE) server, which includes the network, HTTP servers, firewalls, and proxies.

2

Introduction to Temenos T24

Temenos T24 is one of the most popular and widely used banking applications in the world. This chapter introduces T24 from a technical and architectural viewpoint.

This chapter includes the following sections:

- T24 in context
- T24 layered application architecture
- Technical component view

2.1 T24 in context

Temenos T24 provides access by internal users in a bank through the T24 browser user interface. External users, such as customers, can access T24 through a range of channels that interface to T24. Temenos provides channels products, which include ARC Internet Banking and ARC Mobile Banking. Third-party channel products can also be used with T24, such as for access through ATM machines and for voice or call center access.

External IT systems can interface to Temenos T24 through transactional interfaces, such as using web services, or through batch interfaces.

The testing in this project focused on Temenos T24 in isolation from the systems to which it might have to interface. It involved using the T24 user interface (T24 Browser) for interactive testing and used a part of the daily close-of-business (COB) batch processing for the batch testing. In planning for a high availability implementation, consider how user channels and interfaces to related systems (and the related systems if T24 depends on them) can be made highly available.

Figure 2-1 summarizes the business functions of T24. For a description of these functions, see the Temenos website at:

http://www.temenos.com/products/t24/#

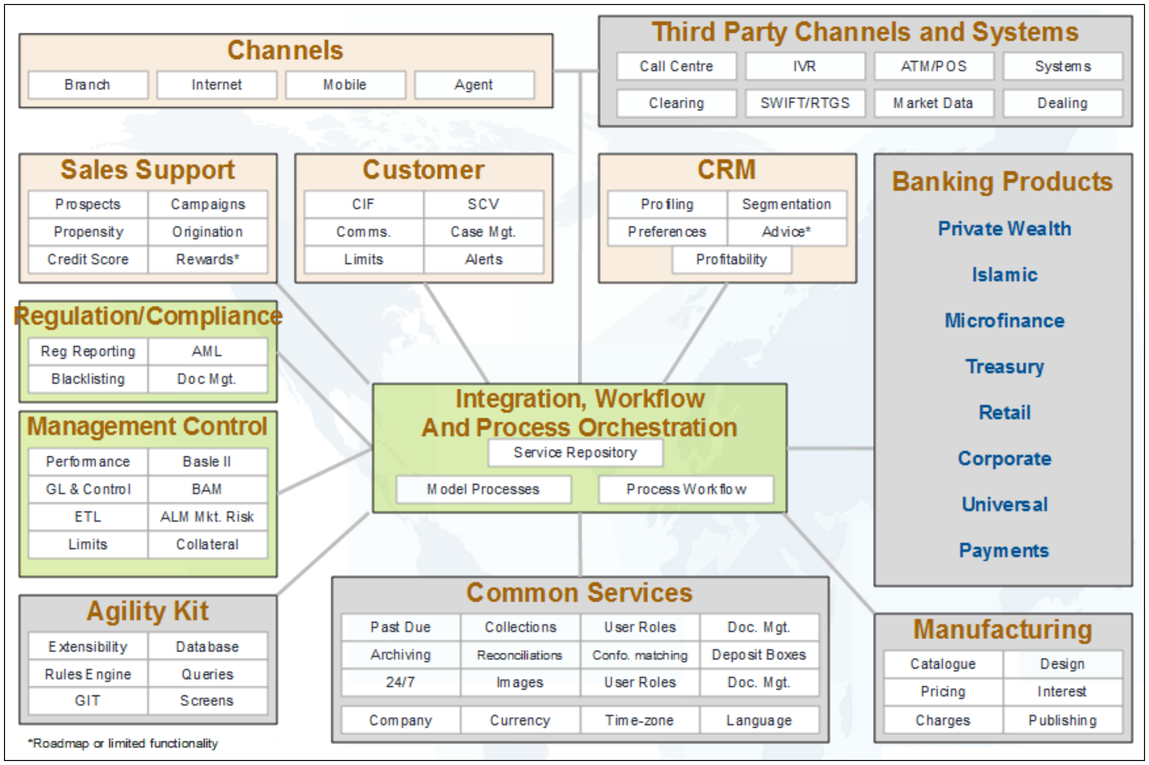


Figure 2-1 An outline of T24 business functions

Figure 2-2 shows the typical system context for a T24 implementation. The system context diagram shows the users and other IT systems that need to interface with T24 to deliver its functional capabilities.

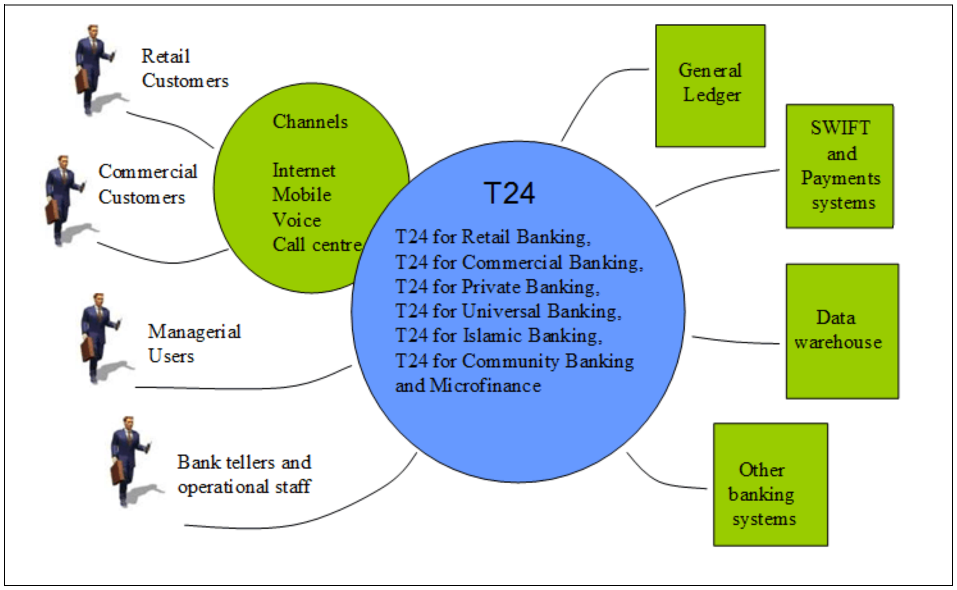


Figure 2-2 T24 typical user and systems interfaces

2.2 T24 layered application architecture

Figure 2-3 shows the logical architectural layers that are used in the T24 application architecture.

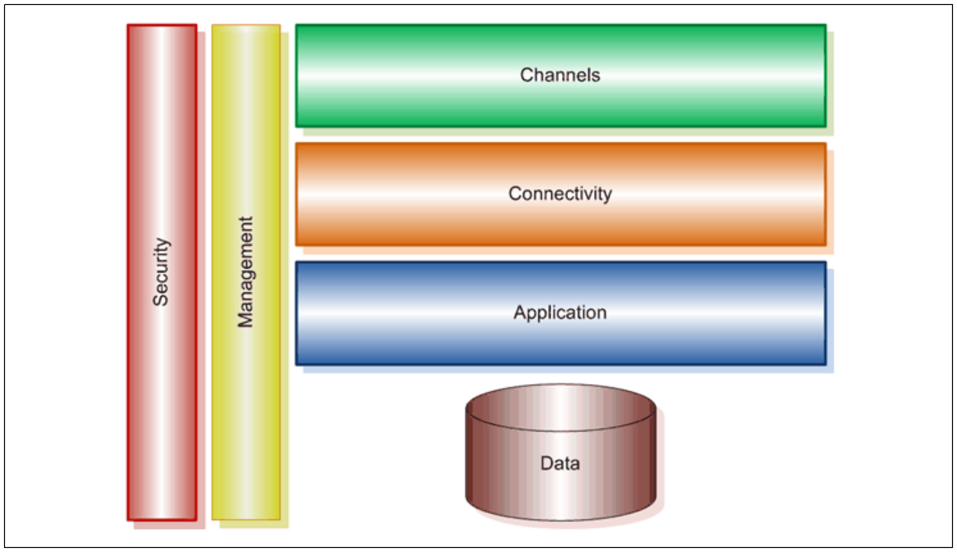


Figure 2-3 Layers of the T24 logical architecture

Each layer has a role:

Channels	Represents the delivery channels used to interface with users and external systems. It enables users to use data and services from the T24 application tier.
Connectivity	All channel consumers access T24 data and services through the Connectivity layer. This layer connects the channels to the application tier. It supports two modes of connection: *direct connect* and *indirect connect*: • Direct connect is used for simple single-server configurations. • Indirect connect is used where high availability and horizontal scalability are required. It routes all requests for T24 services through a message queue, so that multiple T24 application servers can service the same queue.
Application	Supports starting the T24 application, including all business logic and much of the presentation logic.

Data	Includes a data persistence layer and a database management system. The T24 application uses a persistence layer to remove the persistence mechanism that was used. The database used can be the Temenos database system (jBASE) or a relational database product such as IBM DB2. In the persistence layer, T24 can remain agnostic to the persistence technology that is used. The persistence layer that enables use of DB2 is called the *Direct Connect Driver for DB2*.
Security	Provides security services to the other tiers.
Management	Provides operational management and monitoring services to the other tiers. This layer includes the tools provided by Temenos and available vendor and platform tools.

2.3 Technical component view

Several technical components make up the system. These components are decomposed from the logical view and layers shown in Figure 2-3 on page 9. The technical components are application components that provide technical function, rather than business function. The technical components are deployed across the physical infrastructure and do not represent physical devices.

T24 can be implemented in the following modes:

Message-driven mode	T24 channels use message queue middleware to communicate with the T24 application. This mode can be further classified as the *Channels Flow* and *TOCF Plugin Interface Flow*.
Agent mode	Is used by the T24 channels to communicate with the T24 application when message queue middleware is not available.
Enterprise Edition mode	Can be implemented in message-driven mode and agent mode.

2.3.1 Typical message flows

Figure 2-4 explains the channel flows that use the message-driven mode. This figure also shows where the technical components are in the T24 layered architecture.

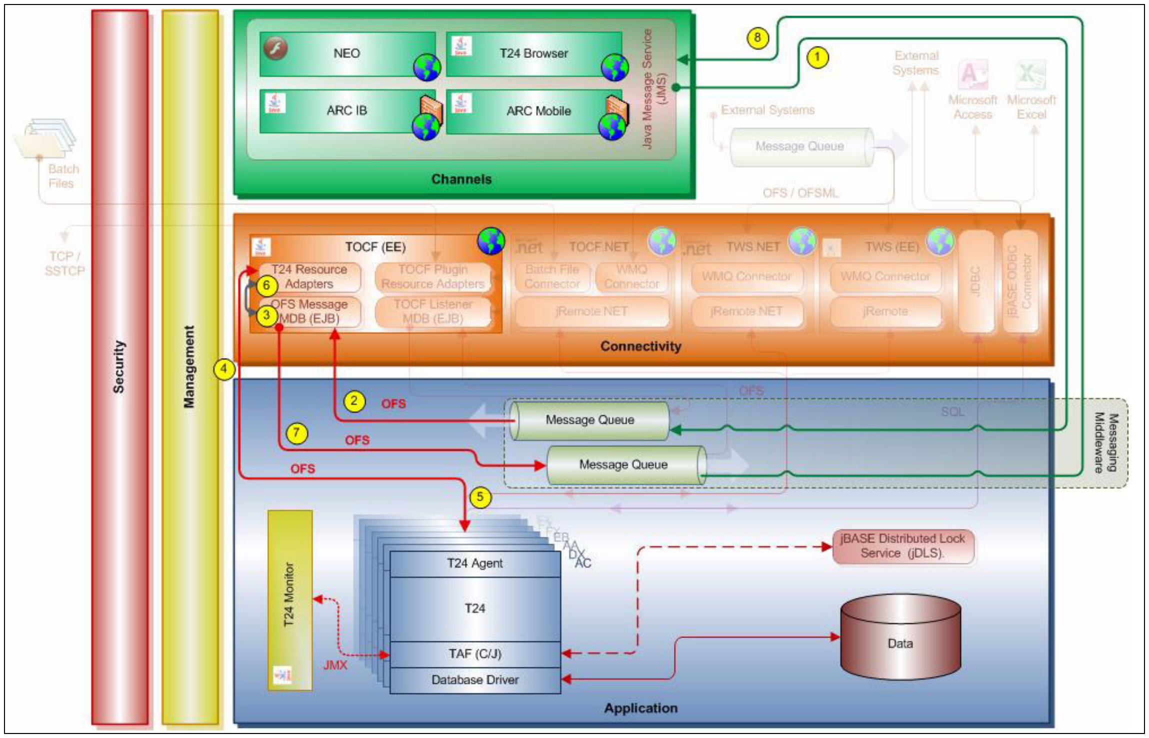


Figure 2-4 Technical components in the T24 architectural layers

The numbers in the following list correspond to the numbers in Figure 2-4 to describe the channel flow:

1. Channels write OFS, BrowserML, or both types of messages into the Message Queue middleware.
2. OFS Message message-driven bean (MDB) selects the messages from the Message Queue middleware.
3. OFS Message MDB calls the T24 Resource Adapter for each message that is read from the Message Queue middleware.
4. The T24 Resource Adapter sends each message to the T24 application and waits for the response.

5. The T24 application processes each message and response back to the T24 Resource Adapter.
6. The T24 Resource Adapter passes the response to the OFS Message MDB.
7. The OFS Message MDB writes the response to the Message Queue middleware.
8. The T24 channels read each message from the Message Queue middleware.

2.3.2 Component descriptions

Each layer in the T24 architecture has several technical components.

The *Channels layer* has the following components:

T24 Browser The T24 Browser is a servlet application that provides the HTML thin-client user interface for T24. JMeter is the test driver that is used in these tests. It simulates the T24 Browser in the Channels layer.

ARC Internet Banking (ARC IB)
The Internet Banking application for T24, where ARC stands for Acquire, Retain and Cross-sell. ARC IB includes additional business logic in the T24 Application layer and a separate user channel.

The ARC IB channel uses a specially configured version of the T24 Browser to provide enhanced security and additional user interface features. It is normally deployed on separate hardware on a DMZ network.

ARC Mobile ARC Mobile is the Mobile Banking application for T24 and is a part of the ARC suite of modules. ARC Mobile supports various mobile technologies. These technologies include SMS, XHTML, and device applications for iPhone, Android phones, and Windows Mobile 7 phones. ARC Mobile provides a single point of connectivity into T24 for all of these phones. Mobile menus and SMS keywords are mapped to standard T24 transactions.

ARC Mobile uses handset detection to identify a customer handset and provide the optimum interaction mechanism for that handset.

The *Connectivity layer* has the following components:

Temenos Open Connectivity Framework - Enterprise Edition (TOCF-EE)
A family of Java Platform, Enterprise Edition (J2EE) components that provide industry standard connectivity that is compliant with J2EE for T24. TOCF manages communication between channels and T24 and enables batch communication with T24.

The TOCF MDB reads messages from an input queue, presents them to T24 for processing, through a Java EE Connector Architecture (JCA) adapter (the T24 resource adapter). Then TOCF MDB receives responses from T24 and places them on an output queue. The Batch File plug-in reads batch transaction files and adds the individual transactions to the input queue. Then it builds an output file from the message responses.

Temenos Web Services Enterprise Edition (TWS-EE)
Enables T24 transactions to be used as web services. Temenos Web Services include a development environment (Temenos Web Services Composer) and a runtime environment. The runtime environment accepts incoming web service requests, places them on the T24 input queue, and builds a web service response from the T24 output message.

Message Queues (JMS or WMQ)
Message queuing is used to enable indirect connection between the Channels layer and the Application layer.

The *Application layer* has the following components:

T24 Agent The T24 can use the Temenos Application Framework for C (TAFC Agent) for C environments or the Temenos Application Framework for Java (TAFJ) for Java environments. TAFC was used in the environment for the testing documented in this paper.

The TAFC Agent is a socket server that listens on a user-defined TCP port. It accepts and processes incoming client requests and returning the results.

T24 The core of T24 that contains the banking business logic. T24 business logic is written in a high-level language called *jBC*, which is then compiled into C or C++ for the TAFC environment (or Java for the TAF or TAFJ environment).

Temenos Application Framework for C (TAFC)	A C library that provides runtime services for T24.
Temenos Application Framework for J (TAFJ)	Similar to the TAFC, but the Java version.
Database driver	Direct Connect Driver (DCD) is the T24 data abstraction layer that decouples T24 business logic from the underlining data storage or structure. This project uses the DB2 DCD as the database driver for the DB2 database. The jBASE Distributed Lock Service (JDLS) lock manager is a component of the DCD that manages locks across multiple T24 application servers.
T24 Monitor	T24 Monitor is a Java Management Extensions (JMX) and web-based online monitoring tool for T24. It offers real-time statistics and historical views of a particular T24 system.
Message Queue	An optional middleware infrastructure that allows T24 to use message-driven communication with the Channels layer. It is essential where high availability or horizontal scalability in the T24 Application layer are required. This project used WebSphere MQ Server as the messaging infrastructure. An alternate approach is possible by using WebSphere Application Server JMS queues.
Database	Can be any database management system that is supported with T24. Examples include DB2, Oracle, SQL Server, and jBASE (the Temenos proprietary database management system).

3

IBM high availability and disaster recovery options

Banking systems typically require a high quality of service. In particular, they also require a high level of availability because the cost and reputational damage caused by the system being unavailable is so high.

This chapter explores how you can use IBM middleware with T24 to deliver high availability. The methods described in this chapter provide for redundancy and resilience in each architectural layer in T24. You can use these methods to design a system that can deliver a specified level of availability. Calculating the level of availability depends on the specifics of an individual implementation of T24, but is beyond the scope of this paper.

T24 is a layered application. Therefore, each layer must support high availability if the application as a whole must be highly available. This chapter considers each layer and its corresponding IBM middleware. It shows how you can use the middleware to deliver the levels of availability that are typically demanded of banking applications.

This chapter contains the following sections:

- WebSphere Application Server Network Deployment
- WebSphere MQ server
- DB2 Enterprise Server 9.7
- DB2 pureScale 9.8 feature

3.1 WebSphere Application Server Network Deployment

The WebSphere Application Server for Network Deployment uses clustering to achieve horizontal scalability and high availability. WebSphere Application Server for Network Deployment has a built-in application server clustering function. It also has a high availability manager function to protect WebSphere Application Server instances within a cluster. Clustering application servers provide workload management and failover for applications that are on the application server cluster.

Administration component failure: Failures of administration components, such as the deployment manager and node agent, must be considered separately. Failure of a deployment manager has no impact on client processing. The reason is that the deployment manager is used to configure, administer, and monitor the WebSphere Application Server Network Deployment environment.

The failure impact is also mitigated because the configuration data is replicated to each node agent in a WebSphere Application Server Network Deployment environment. In this case, cell-wide administration tasks cannot take place. Node-level administration tasks are synchronized with the deployment manager when it becomes available again.

When a node agent fails, all administration activities for the WebSphere Application Server Network Deployment instances that are managed by it become unavailable. Administration activities that can become unavailable include application server start and stop commands. However, if an application server is already available, it is unaffected by node agent failures and can serve client requests as normal. Typically, both the deployment manager and node agent failures can be protected by using an IBM PowerHA cluster to restart the failed process.

3.1.1 Cluster configuration

This cluster configuration deploys a minimum of two WebSphere Application Server Network Deployment cluster members per physical server. This way, the physical server can continue processing workload when one WebSphere Application Server Network Deployment cluster member fails (for example, if a Java virtual machine (JVM) crashes). This configuration provides higher availability, but having multiple WebSphere Application Server Network Deployment cluster members per physical server often also increases throughput.

This configuration mitigates the impact of a software failure in an individual member. To mitigate the impact of hardware failure, you need at least two physical servers. Additional physical servers beyond the first two servers might be added to provide greater capacity, which will also provide a further marginal improvement in availability.

If you have a requirement for business continuity with no loss of performance, you need sufficient physical servers to meet performance objectives when one (or more) physical servers are unavailable. Physical servers can also be distributed across multiple sites to provide continuity if a site is unavailable. This setup usually requires a high-speed network between sites (typically a fiber optic link) because it replaces a LAN connection between two servers with a WAN connection.

Figure 3-1 shows the recommended cluster configuration that scales the cluster members vertically and horizontally across two physical servers.

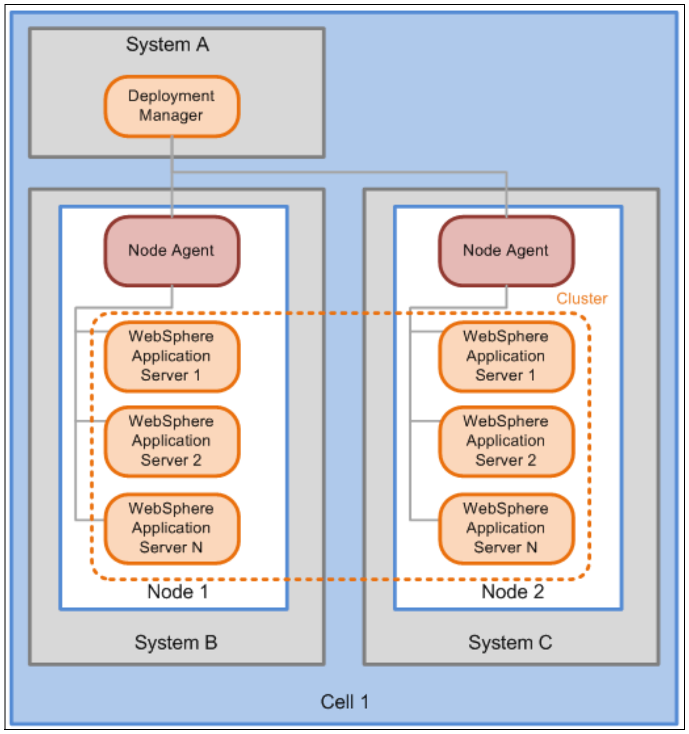

System A
Deployment Manager
Node Agent
Node Agent
Cluster
WebSphere Application Server 1
WebSphere Application Server 2
WebSphere Application Server N
WebSphere Application Server 1
WebSphere Application Server 2
WebSphere Application Server N
Node 1
Node 2
System B
System C
Cell 1

Figure 3-1 Vertically and horizontally scaled WebSphere Application Server Network Deployment cluster

3.1.2 Session management

When using multiple copies of an application server, multiple consecutive requests from clients can be serviced by different servers. If each client request is independent of every other client request, whether consecutive requests are processed on the same server does not matter.

The following types of requests are possible in terms of session management:

Stateless	The server processes requests based solely on information that is provided with each request and does not rely on information from earlier requests. Therefore, the server does not need to maintain state information between requests.
Stateful	The server processes requests based on both the information that is provided with each request and information that is stored from earlier requests. To achieve this processing, the server must access and maintain state information that is generated during the processing of an earlier request.

For stateless interactions, different requests can be processed by different servers.

For stateful interactions, the server that processes a request needs access to the state information necessary to run that request. The same server processes all requests that are associated with dedicated state information. Alternatively, the state information can be shared by all servers that require it. Data that is shared across multiple systems might have a performance impact.

WebSphere Application Server Network Deployment has several techniques for maintaining state:

Session affinity	All client requests that are within the same session are directed to a particular server for the duration of the session.
Transaction affinity	All requests within the scope of a transaction are directed to a particular server.
Server affinity	All requests are directed to a particular server.

The workload management service of WebSphere Application Server Network Deployment takes server affinity and transaction affinity into account when directing client requests among cluster members.

The session manager of WebSphere Application Server Network Deployment stores client session information. It takes session affinity and server affinity into account when directing client requests to the cluster members of an application server. When the session manager of WebSphere Application Server Network Deployment stores client session information, it is made available to other cluster members. This way, if a particular WebSphere Application Server Network Deployment instance fails, another cluster member can take over without losing the session information.

The session manager of WebSphere Application Server Network Deployment has the following options for session persistence:

Local sessions The session information is stored in the local application server memory so that it is not available to other servers. This option is the fastest and simplest to configure. However, an application server failure causes the session information to be lost, causing the session to end. Therefore, do not use this option for failover of WebSphere Application Server instances in a cluster.

Replicated persistent sessions

A memory-to-memory replication that copies session data across application servers in a cluster. It stores the data in the memory of an application server and provides session persistence. Using this type of replication eliminates the effort of maintaining a production database and the read and write performance overhead of a database. It also eliminates the single point of failure (SPOF) that can occur with a database.

The amount of session information you can store is bound by the JVM heap size of your application servers. Regardless of how it is bound, the maximum application server heap size is much smaller than the amount of disk space available on a database server that is serving as a session store.

This option is the simplest to configure and is suitable for WebSphere Application Server instance failover within a cluster where large session objects are not required.

Database persistent sessions

The session information is stored in an external database. This option is typically the worst performing option and the most complex to configure and maintain. However, it is not limited to the size of the session object that is being persisted.

This option is suitable for WebSphere Application Server instance failover within a cluster where large session objects (beyond the JVM heap size) are a requirement.

T24 is stateless in the browser layer and is stateful at the presentation layer, T24 application server layer, and database layer. The ideal approach is to enable memory-to-memory replicated persistent sessions in a WebSphere Application Server Network Deployment cluster, which is what was used in the tests documented in this paper.

Sharing sessions across cells: Although possible, do not share sessions across multiple WebSphere Application Server cells. The reason is that session sharing requires a single database server (or database server and failover server) for session persistence. Maintenance on the database server of any type can result in an outage, adding complexity to the overall solution.

3.1.3 IBM HTTP Server

In clustered IBM WebSphere Application Server environments, the HTTP plug-in for IBM HTTP Server can provide failover if the HTTP plug-in can no longer send requests to a particular cluster member. By default, the following conditions exist in which the HTTP plug-in marks down a particular cluster member and fails over client requests to another cluster member that can receive connections:

- The HTTP plug-in is unable to establish a connection to a cluster member's application server transport.
- The HTTP plug-in detects a newly connected socket that was prematurely closed by a cluster member during an active read or write.

The IBM HTTP Server can be deployed as *Unmanaged*, so that it is not managed by using deployment management for WebSphere Application Server Network Deployment. The IBM HTTP Server can also be deployed as *Managed*, where it is managed by using a deployment manager for WebSphere Application Server Network Deployment. If the IBM HTTP Server is deployed within a DMZ, it might be deployed as unmanaged to prevent any deployment manager traffic from passing into the DMZ.

Clustering application servers that host web containers automatically enable plug-in workload management for the application servers and the servlets that they host. Routing of servlet requests occurs between the web server plug-in and the clustered application servers by using HTTP or HTTPS.

This routing is based on weights that are associated with the cluster members. If all cluster members have identical weights, the plug-in sends equal requests to all members of the cluster equally with no affinity to a particular cluster member. If the weights are used, the plug-in routes requests to those cluster members with the higher weight value more often.

Several configurable settings in the `plugin-cfg.xml` file can be tuned to affect how quickly the HTTP plug-in marks down a cluster member and fails over to another cluster member.

Use the default setting for a T24 deployment, except for the settings shown in Example 3-1 for each server node in the configuration file.

Example 3-1 T24 deployment settings

```
/usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml
<Server CloneID="168fk2ap5" ConnectTimeout="5"
ExtendedHandshake="false" LoadBalanceWeight="2" MaxConnections="-1"
Name="eg09ph02Node01_FailOverServer1" ServerIOTimeout="120"
WaitForContinue="false">
.....
<Server CloneID="168fn4cv0" ConnectTimeout="5"
ExtendedHandshake="false" LoadBalanceWeight="2" MaxConnections="-1"
Name="eg09ph03Node01_FailOverServer2" ServerIOTimeout="120"
WaitForContinue="false">
In addition, the following settings are relevant for the HTTP Server
itself to maximize its performance characteristics.
/usr/IBM/HTTPServer/conf/httpd.conf
<IfModule worker.c>
ThreadLimit        2000
ServerLimit         1
StartServers        1
MaxClients         2000
MinSpareThreads    2000
MaxSpareThreads    2000
ThreadsPerChild    2000
MaxRequestsPerChild  0
</IfModule>
```

3.2 WebSphere MQ server

Two primary technologies are used to provide high availability for WebSphere MQ on the AIX platform:

- Deploying WebSphere MQ with multi-instance queue managers
- Deploying WebSphere MQ into a cluster managed by PowerHA on AIX

3.2.1 Multi-instance queue managers

WebSphere MQ is deployed on two servers, where one server is a spare standby server and the other server is an active server. A queue manager is deployed on the active server, and its data and logs are on a shared storage device that is accessible by the standby server. A reference to the queue manager and logs is deployed to the standby server, which enables the queue manager to start from the active and standby servers.

Each server references the same queue manager data and logs on the shared storage. The running queue manager on the standby server remains ready to take over from the active server and does no processing until that point. When the active queue manager fails, the standby server assumes the active role and begins processing requests. When the failed active queue becomes available again, it assumes the standby role.

The active queue manager can fail for the following reasons:

- The server fails that hosts the active queue manager instance.
- Connectivity fails between the server that is hosting the active queue manager instance and the file system.
- WebSphere MQ detects the unresponsiveness of queue manager processes and then shuts down the queue manager.

A multi-instance queue manager deployment does not require additional software or licensing. However, it requires additional configuration at the WebSphere MQ channel and connection level that allows WebSphere MQ clients to reconnect the current active queue manager (Figure 3-2). This configuration is required because a multi-instance queue manager deployment does not fail over the IP address of the active queue manager.

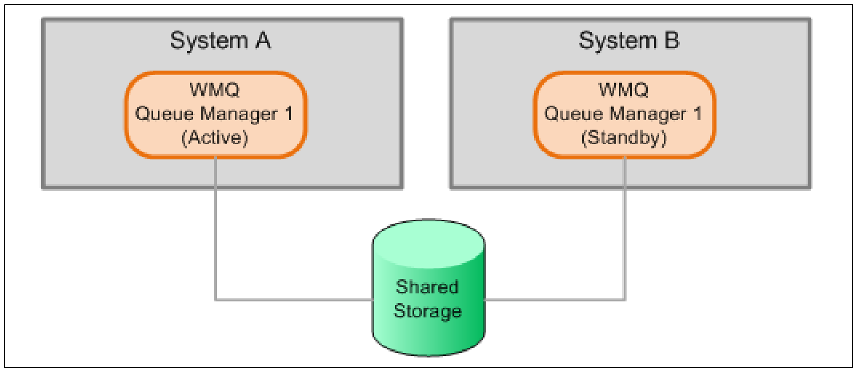


Figure 3-2 WebSphere MQ multi-instance queue manager

3.2.2 WebSphere MQ cluster managed by PowerHA

In the deployment of a WebSphere MQ cluster managed by PowerHA, a PowerHA resource group provides failover for the WebSphere MQ queue manager, its data and logs, and its IP address. Although you can include the listener process of the queue manager, instead include the start and stop of the queue manager listener with the start and stop of the queue manager. You create an explicit WebSphere MQ listener, which is under the control of the queue manager.

The queue manager to protect must have its data and logs on a shared storage device that is accessible by both nodes of the cluster. A PowerHA cluster ensures that only one node has write access to the disk.

PowerHA uses scripts to gracefully start and stop the protected resource. It also uses a script to monitor the availability of the protected resource. For WebSphere MQ, the default scripts are available for start, stop, and monitor. For more information, see the WebSphere MQ Version 7 Information Center at:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.amqzag.doc/fa70230_.htm

Figure 3-3 shows a PowerHA MQ active-passive cluster with two nodes. Only node A is running a WebSphere MQ queue manager. When PowerHA detects a failure, it fails over the resource group to node B.

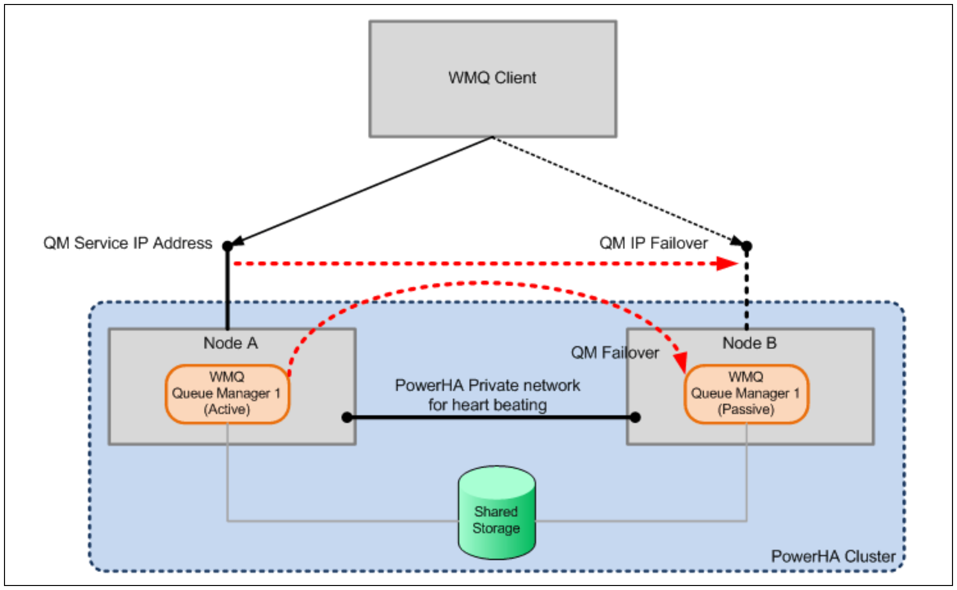


Figure 3-3 PowerHA WebSphere MQ Cluster

PowerHA runs as user *root* on AIX, and on WebSphere MQ, PowerHA runs as user *mqm*. User mqm is a member of the mqm group, which is granted permissions to administer WebSphere MQ. Both the mqm user and group must be present on both nodes of the cluster. In addition, the start, stop, and monitor scripts must switch to user mqm before running. You can use a wrapper script to achieve this switch before running the actual start, stop, and monitor scripts. WebSphere MQ must also be installed on both nodes.

The PowerHA configuration is used to determine the speed of failure detection and the rate at which the queue manager is monitored. A default "normal" detection speed can be used. Other speeds are "fast" or "slow." For a complete description of monitoring configuration, see the *PowerHA for AIX Cookbook*, SG24-7739.

Important: Careful configuration here can limit the failover duration and reduce the chance of client timeout.

With T24, deploy WebSphere MQ into a cluster that is managed by PowerHA on AIX, which is the configuration is used in the testing documented in this paper. The reason is that PowerHA does not require additional client-side configuration, and therefore, is transparent to WebSphere MQ clients.

3.3 DB2 Enterprise Server 9.7

DB2 has many features that optimize its deployment for scalability, high availability, and disaster recovery. In addition, DB2 can integrate with other IBM technologies to provide similar functionality. Consider the following DB2 features and other technologies:

- Hardware solutions
- Cluster manager integration with DB2 high availability feature
- PowerHA on AIX
- IBM Tivoli® System Automation for Multiplatforms Base Component
- High availability disaster recovery (HADR)
- Automatic client reroute (ACR)
- DB2 pureScale Version 9.8

3.3.1 Hardware solutions

Redundant Array of Independent Disks (RAID) and remote storage mirroring are two hardware solutions that you can use with DB2 to provide high availability solutions.

RAID

RAID technology consists of a group of individual disks that can be accessed as a whole to increase performance, fault tolerance, or both. Multiple disk drive components are combined into a logical unit, where data is distributed across the drives in one of several ways called *RAID levels*. Although RAID technologies can be implemented at the software level, such implementations usually do not have the level of security or performance that most database servers require.

RAID levels have a range of RAID 0 - RAID 10, with varying degrees of performance and fault tolerance. Typically, RAID 1 or 10 is considered better for DB2 databases that are running online transaction processing (OLTP) applications for performance and fault tolerance. However, with improvements in storage area network (SAN) technology, RAID 5 is also an available option. RAID 5 was previously considered a poor choice for databases because it requires additional processing for parity calculation and distribution. Modern SAN controllers can now process RAID 5 with little or no degradation in performance.

Typically, you might choose a RAID solution if fault tolerance, performance, or both are the most important factors. RAID in isolation has no provision for site failures. Typically, RAID is used for failure within a site, and site failures are covered by an alternative solution.

Remote storage mirroring

Remote storage mirroring technology consists of two or more disk arrays in storage enclosures in geographically remote locations. These enclosures are connected by using a high-speed interconnect (typically dark fiber based or similar technologies). They implement algorithms to replicate the data between the enclosures.

Storage mirroring replicates all disk writes, including the log information. In general, synchronous mode and asynchronous mode are two typical modes to update the remote disk. Most of the time, remote storage mirroring must be integrated with additional resources if automatic failover operations are required. IBM offers the following remote storage mirroring products, among other remote storage mirroring products:

- IBM Peer-to-Peer Remote Copy (PPRC) and IBM Peer-to-Peer Remote Copy over Extended Distances (PPRC-XD) on IBM System Storage® Enterprise Storage Server®
- IBM Metro Mirror and IBM Global Mirror on IBM System Storage Disk Storage and IBM System Storage SAN Volume Controller

Using this technology has the following advantages:

- Data is copied to geographically remote locations.
- Data is copied at the hardware level, which usually improves performance and reliability.

A typical use case for this technology is to provide a disaster recovery site for a DB2 database with no loss of data. Additional technology, such as clustering, is required to automate the process of starting DB2 at the new site.

3.3.2 Cluster management

The DB2 high availability feature enables integration between DB2 and cluster management software.

When a DB2 instance is in a clustered environment and changes its configuration (for example, from a running state to a stopped state), the cluster manager must be made aware of it. The DB2 high availability feature provides infrastructure for enabling a DB2 instance to communicate with a cluster manager when an instance configuration changes, such as stopping a DB2 instance.

In addition, within a clustered environment, some DB2 instance configuration and administration operations require related cluster configuration changes. The DB2 high availability feature enables the DB2 instance to automatically request cluster manager configuration changes whenever certain DB2 instance configuration and administration operations occur.

Two types of clustering solutions are possible, operating system dependent and operating system independent:

- DB2 supports the IBM PowerHA for AIX operating system for dependent cluster managers.
- DB2 supports the IBM Tivoli System Automation for Multiplatforms base component for independent cluster managers.

The cluster managers have the following primary technical differences:

- PowerHA is functionality offered by AIX.
- PowerHA is typically simpler to configure.
- Because PowerHA is tightly integrated with AIX, cluster member failure detection and failover can be faster.
- Tivoli System Automation is bundled with DB2 as the DB2 high availability feature.

- Tivoli System Automation is based around failure detection through the use of monitoring scripts against resources and, therefore, can be complex to configure.
- Tivoli System Automation is not limited to the features of the underlying operating system cluster manager and can often offer a richer set of options.

Implement DB2 with an operating system-dependent clustering solution such as PowerHA. Only use an operating system-independent cluster manager such as Tivoli System Automation in the following situations:

- A heterogeneous environment exists with different hardware providers.
- A platform change is a possibility.
- Integration is required with existing heterogeneous clusters.
- A particular cluster configuration is not possible by using the clustering solution of the operating system.

Typically, PowerHA or Tivoli System Automation is used with HADR and hardware solutions such as remote storage mirroring to provide automatic failover to a standby DB2 instance. The standby DB2 instance can be in the same data center to facilitate continuous operations, or at a remote location for disaster recovery.

3.3.3 DB2 high availability disaster recovery

By using the DB2 HADR feature in DB2, you can replicate any logged database activity to a local or remote location. DB2 HADR works in a similar fashion to the older log shipping solutions. However, all the work is made inside of the DB2 engine at the software level. Use HADR instead of custom-made log shipping mechanisms. The DB2 HADR configuration was used in the testing that is documented in this paper.

A DB2 HADR primary database uses internal processes to ship database log buffers to an HADR standby database. A process at the standby server then replays the log records directly to the standby database. The standby database takes on the role of the primary database server and is accessed by applications and users if the primary server fails. The standby takeover process is used when a planned takeover occurs (such as for maintenance) or an unplanned takeover occurs (such as for a failover). This process is manual and is normally initiated by the database administrator (DBA) from the standby database.

Heartbeats are sent from the primary server to the standby database server to confirm the health of both servers. The standby database sends acknowledgments to the primary server after it receives its heartbeat.

Using DB2 HADR includes the following advantages:

- Fast failover
- No committed transaction data loss in synchronous mode

  In synchronous mode, a log write is successful when both of the following situations are true:

  – The log buffers are written to the log files of the primary server.

  – An acknowledgement is received that it was also written to the log files of the standby server and they were applied to the standby database.

  In the default, near-synchronous mode, log write is successful only when both of the following situations are true:

  – The log buffer of the primary server is written to log files on the primary server.

  – An acknowledgement is received that the log buffer was received on the standby server.
- Negligible performance affect
- Simple to monitor
- Can have transparent failover and failback from an application perspective when using ACR
- Simple integration with DB2 high availability and PowerHA on AIX, for failover when physical failures occur and for takeover automation

DB2 HADR is used when fast failover and no committed transaction data loss are required. The ideal DB2 HADR deployment for failover consists of having the primary and standby database servers in the same site. If possible, they should be close to each other to mitigate the effects of network propagation delay for the heart beat communication. The effect can be minimized with appropriate configuration. However, this configuration has no provision for disaster recovery. If the network connection to the disaster recovery site is fast enough to prevent an unacceptable degradation in performance, you can use DB2 HADR to replicate the database to a separate disaster recovery site.

3.3.4 HADR usage with PowerHA

This section explains how HADR is used with PowerHA.

Failover

For the scenario environment, two physical nodes are defined within a PowerHA cluster. One node is active and receives all user requests. The primary HADR database runs on this node. The other node is the standby node on which the

HADR standby database resides. If a primary node failure occurs, PowerHA detects this outage and fails over to the standby node. The PowerHA failover process includes the HADR takeover commands, eliminating the need for manual intervention for HADR failover. PowerHA also fails over the service IP of the cluster to the standby node, which is optional if ACR is used. Figure 3-4 illustrates this scenario.

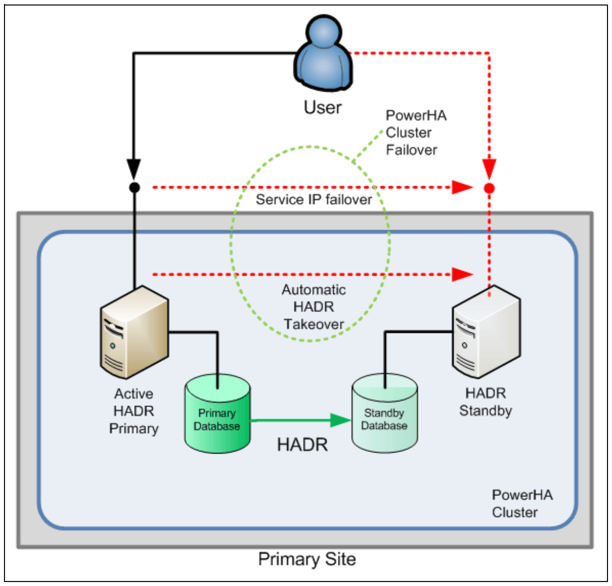


Figure 3-4 HADR and PowerHA for DB2 failover

Disaster recovery

In DB2 9.7, DB2 HADR supports log shipping to only one backup site. If DB2 HADR is used for disaster recovery, by shipping logs at a disaster recovery site, an alternative solution must be used. This solution must provide continuity of operations at the primary site when the primary DB2 instance fails, or the site must be taken down for operational reasons.

In DB2 10.1, DB2 HADR can ship logs to multiple backup sites. Therefore, it is possible to use DB2 HADR for both on-site failover and off-site disaster recovery at the same time.

In the scenario environment used, the DB2 database is held on a disk that is shared within a PowerHA cluster between an active server and a passive standby server. Only the active server in the cluster accesses the shared disk. The users are connected to the active server. HADR is configured between the active server and a standby server at a disaster recovery site. The passive standby server can become the active server in the cluster and assume the HADR primary role if the active server fails, which provides for local failover. If the primary site fails, the DBA performs an HADR takeover from the standby server in the disaster recovery site, which provides for disaster recovery. The users reconnect to the disaster recovery site manually or optionally through an ACR configuration.

Figure 3-5 illustrates this scenario. Use this deployment for T24, with a RAID shared storage device such as IBM System Storage DS8000®.

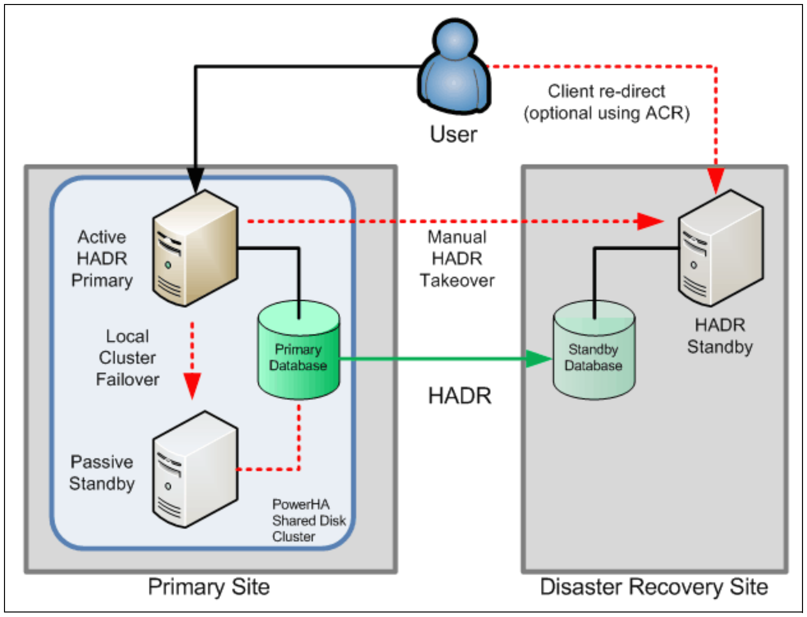


Figure 3-5 HADR and PowerHA for DB2 failover and disaster recovery

3.4 DB2 pureScale 9.8 feature

DB2 pureScale is an optional DB2 feature in DB2 9.8. You use this feature to scale out your database on a set of servers in an “active-active” configuration for a solution that is highly available and highly scalable. In this configuration, the copy of DB2 running on each host (or server) can simultaneously access the same data for both read and write operations.

A collection of one or more DB2 servers that share DB2 data is called a *data sharing group*. A DB2 server that belongs to a data sharing group is a member of that group. All members of a data sharing group share the database. Currently, the maximum number of members in a data sharing group is 128.

In addition to the DB2 members is a cluster caching facility (CCF). The CCF is a DB2 pureScale component that provides centralized lock management and a centralized global cache for data pages (known as the *group buffer pool*). This feature is provided by the AIX operating system. A DB2 pureScale cluster normally includes two CCFs for resilience.

Each member in the data sharing group can interact directly with the CCF, through an efficient InfiniBand network. In this network, each member has point-to-point connectivity to the centralized locking and caching facility.

A typical DB2 pureScale cluster consists of the following components, among other components:

- Two or more DB2 pureScale members
- Two cluster caching facilities
- SAN-attached cluster storage that runs IBM General Parallel File System (GPFS™)
- A high-speed, low-latency cluster interconnect such as InfiniBand

Figure 3-6 shows such a configuration, with four members and two CCFs, by using InfiniBand for low-latency communications. The DB2 pureScale feature is a shared-data architecture, in which all members operate on a single copy of the database. They communicate with each other by using the CCF to synchronize activities and to ingest, modify, and retrieve data as required by the application. Messages between the members and CCF use the remote direct memory access (RDMA) capability in the cluster interconnect, which provides low communication latencies and low CPU utilization per message.

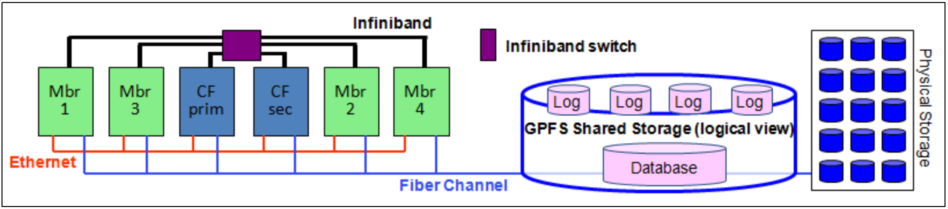


Figure 3-6 Architecture overview of the DB2 pureScale feature

DB2 pureScale uses RDMA technology so that the Cluster Caching Facility can eliminate communication between members for lock management and global caching services. Therefore, applications do not have to be cluster-aware. An application can make a data request to any cluster member, not just the member where the data resides.

When a failure occurs to a cluster member, the DB2 pureScale cluster services automatically monitor all necessary resources and restart them as needed.

Applications that are connected to a failing member are rerouted to an active member where the application can reissue any failed transactions. Applications that are connected to a nonfailing component are not affected.

One of the distinguishing factors of the DB2 pureScale feature is that no cluster-wide freeze occurs when recovering from a failure. Only data in the process of being updated on the failing member is temporarily unavailable until recovery is complete. Applications on active members that are trying to access the locked data on the failing member are briefly in a lock-wait state, and by default, they do not receive any errors.

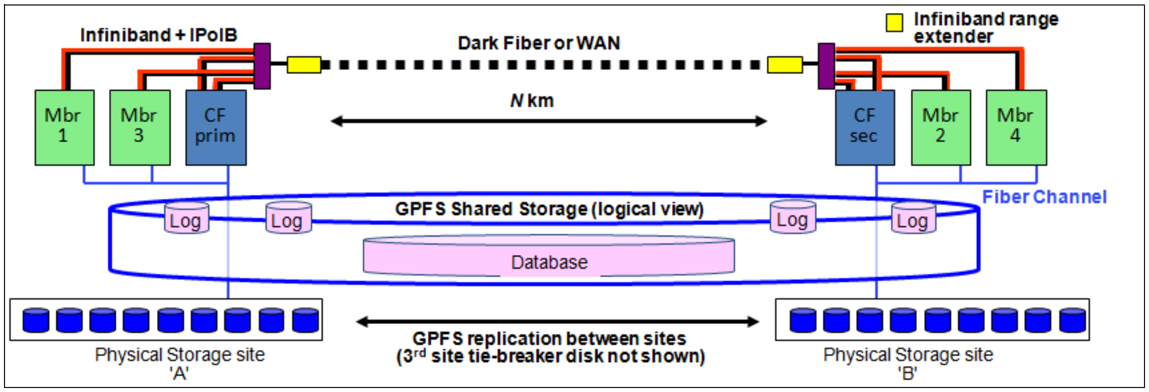
For a DB2 pureScale 9.8 implementation, two approaches are possible for disaster recovery. DB2 pureScale can be configured as a split configuration across two sites. This configuration is a logical extension to the DB2 pureScale architecture. It provides two active sites with either site continuing to operate if the other site fails. It requires a high performance fiber connection between sites. It also has a performance overhead that grows with the distance between sites and with the amount of write activity on the database.

The preferred option for disaster recovery with DB2 pureScale is to use a geographically dispersed cluster configuration. This configuration is a two-site configuration, where the underlying shared storage file system synchronizes itself automatically in both directions. If one site fails, the second site remains available.

For more information about a long distance DB2 pureScale cluster, see the IBM developerWorks® article “Configuring Geographically Dispersed DB2 pureScale Clusters” at:

http://www.ibm.com/developerworks/data/library/long/dm-1104purescalegdpc/index.html

Figure 3-7 shows a typical multisite Geographically Dispersed DB2 pureScale Cluster (GDPC) configuration. In this diagram, half of the members and CFs are on each side of the cluster at different sites. Each site has its own storage, which is kept in synch by using GPFS replication. Also a third site tie-breaker disk is needed (not shown) for half of the cluster to reach quorum if a site failure occurs.



Infiniband + IPoIB
Dark Fiber or WAN
Infiniband range extender
Mbr 1
Mbr 3
CF prim
N km
CF sec
Mbr 2
Mbr 4
Fiber Channel
GPFS Shared Storage (logical view)
Log
Log
Log
Log
Database
Physical Storage site 'A'
GPFS replication between sites (3rd site tie-breaker disk not shown)
Physical Storage site 'B'

Figure 3-7 Typical Geographically Dispersed DB2 pureScale Cluster configuration

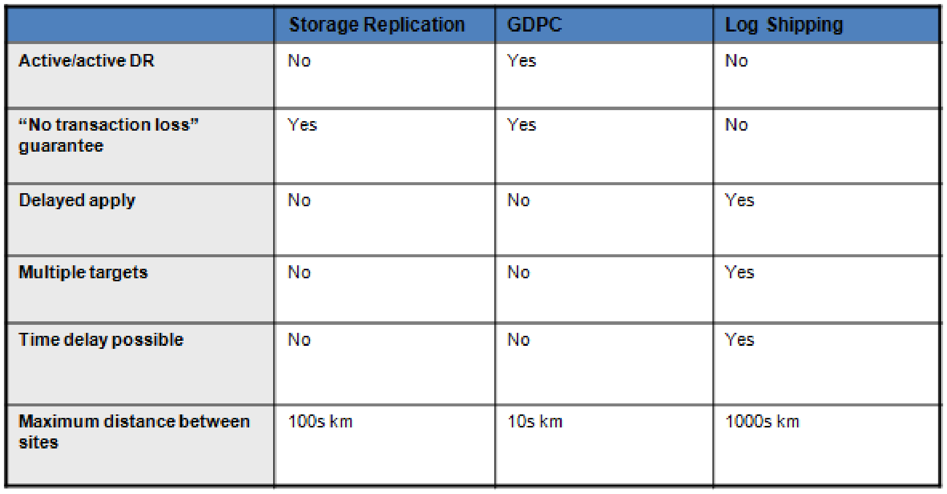
Splitting a DB2 pureScale cluster in half across two sites A and B implies that half of the member systems will physically be at site A and half will be at site B. A third site with minimal configuration is required for tie-breaking purposes, as explained later. One CCF must also be placed at each of the two main sites to avoid a SPOF. To maintain the excellent performance and scalability of the DB2 pureScale software, an RDMA-capable interconnect must be used between sites. This way, messages from a member at one site to the CCF at the other site are as fast and inexpensive as possible. The spanning distance of an InfiniBand network is typically measured in tens or maybe hundreds of meters. However, devices, such as the Obsidian Longbow InfiniBand extender, allow the reach of an InfiniBand network to span greater distances, over wide-area networks or dedicated fiber optic links.

In addition to the dispersal of computing resources, such as members and CFs, a disaster recovery cluster configuration also requires storage to be replicated across sites. Building on the standard DB2 pureScale cluster design, the GDPC configuration uses GPFS synchronous replication between sites to keep all disk write activity up-to-date across the cluster, which includes table space writes and transaction log writes. At a high level, a GDPC design might look similar to the cluster illustrated in Figure 3-7 on page 33.

Client applications that connect to the DB2 pureScale cluster typically have workload balancing (WLB) enabled, which transparently routes work to the member with the most available capacity. WLB maintains optimal use of resources during normal operation and reroutes connections if member downtime (planned or unplanned) or a site failure occur. The client systems are often configured as application servers in a multitier environment. They are often configured with redundancy across sites, providing fault tolerance at the upper layers as well.

Where such a configuration is not practical, an alternative approach is to use disk-based replication, such as IBM Metro Mirror to a remote site where another DB2 pureScale 9.8 cluster can mount the disaster recovery storage. In this case, the DB2 instance of the disaster recovery site is passive, and manual tasks are required to bring it online if the primary site fails.

Figure 3-8 shows a comparison of disaster recovery options.



	Storage Replication	GDPC	Log Shipping
Active/active DR	No	Yes	No
"No transaction loss" guarantee	Yes	Yes	No
Delayed apply	No	No	Yes
Multiple targets	No	No	Yes
Time delay possible	No	No	Yes
Maximum distance between sites	100s km	10s km	1000s km

Figure 3-8 Disaster recovery options

Active-active disaster recovery means that you can do work against both the primary and the secondary disaster recovery site, which can be done by using GDPCs, Q Replication, and Change Data Capture (CDC).

A "No transaction loss" guarantee means that, if the primary site fails, the disaster recovery site is guaranteed to be up to date and all changes that are associated with committed transactions are applied. Storage replication supports this guarantee because all writes to the storage occur on the primary and secondary site. GDPCs support this guarantee through GPFS replication.

Logical replication solutions, such as Q Replication and CDC, cannot support this guarantee. The reason is that the act of reading the logs and applying the changes to the target is asynchronous and is done after the original work that was performed on the primary site. Therefore, transactions can do work and be committed on the primary site, but then the primary site must go down before the logs can be read and the changes can be applied. Log shipping typically involves only log files that are archived. Therefore, transactions that were logged to files that were not archived are not replayed.

Storage replication and GDPCs involve immediate and synchronous updates to the storage or file system. Therefore, no delayed apply is possible. However, due to the capture and apply nature of Q Replication and CDC, it is possible to have delays between the capture and apply processing. Likewise, with log shipping, you can choose how often to perform the roll-forward processing or how current the log files are on the standby.

Storage replication and GDPCs (using GPFS replication) support only a single target. Q Replication and CDC can support multiple targets. With log shipping, you can have multiple standbys and ship logs to all of those standbys.

Storage replication and GDPCs (by using GPFS replication) are synchronous solutions that are intended to keep two separate sites completely in sync with respect to the database's storage. Therefore, the distance between the sites is a significant factor in determining their suitability. The further the distance is between the sites, the greater the impact is on the I/O latency.

Also, for GDPCs, I/O and communication between members and CFs across those sites are factors. Message latency between members and CFs can greatly impact the performance and scalability of a DB2 pureScale cluster. Therefore, the distance between sites is limited. Q Replication, CDC, and log shipping are asynchronous in nature and, therefore, they can support much larger distances. For storage replication and GDPCs, architectures and solutions by IBM are available. For log shipping, customer-specific script development is required.

4

T24 reference configurations for high availability and disaster recovery

The testing that is documented in this paper verifies two reference configurations that T24 customers might use to deliver a highly available configuration at a primary site and a disaster recovery capability at a disaster recovery site. T24 users generally require the following capabilities:

- High availability by using PowerHA and DB2 9.7 with DB2 high availability disaster recovery (HADR) tools, which have been available and established for several years
- DB2 pureScale, which is a relatively new technology that was delivered in DB2 9.8

The configuration diagrams in this chapter show a standby disaster recovery environment with the following capabilities:

- Provides the same processing capacity as the primary site
- Meets the same performance criteria
- Provides high availability to the same standard as the primary site

However, this solution is costly. A common approach is to provide a disaster recovery site without the same high availability as the primary site, which removes the requirement for duplicating capacity within the disaster recovery site.

The test configuration and scenarios do not re-create the full environments. However, they aim to re-create enough of them as necessary to test the recovery technologies that are used and their interaction with T24.

4.1 Reference configuration by using DB2 9.7 with PowerHA and DB2 HADR

This reference configuration is based on the classic IBM WebSphere "Gold standard" configuration pattern, with the addition of components for the T24 server and for the MQ server. At the primary site, clustering technologies that are appropriate to each layer provide high availability to the Java 2 Platform, Enterprise Edition (J2EE) Server, MQ Server, and T24 Server layers. The database layer uses PowerHA to provide failover to a standby server by using a shared disk for the database.

The disk space might normally be protected by RAID technologies within the SAN infrastructure that are not shown here. DB2 HADR supports disaster recovery by log shipping from the primary database to a secondary database at the disaster recovery site. This approach ensures that a complete and current copy of the database is maintained at the disaster recovery site. If the service must move to the disaster recovery site, manual intervention might be required to affect failover. For example, the disaster recovery site is typically held in cold start mode apart from the database server, which is in warm-start mode, to take over database processing at any time.

The configuration diagram (Figure 4-1) shows a fully configured backup site that can provide the same capacity and resilience as the primary site. In practice, many companies configure a disaster recovery site to a lower level of resilience or capacity than the primary site. The tests that are described in this chapter used only a minimal configuration to represent the disaster recovery site to show that the disaster recovery database was maintained correctly by DB2 HADR.

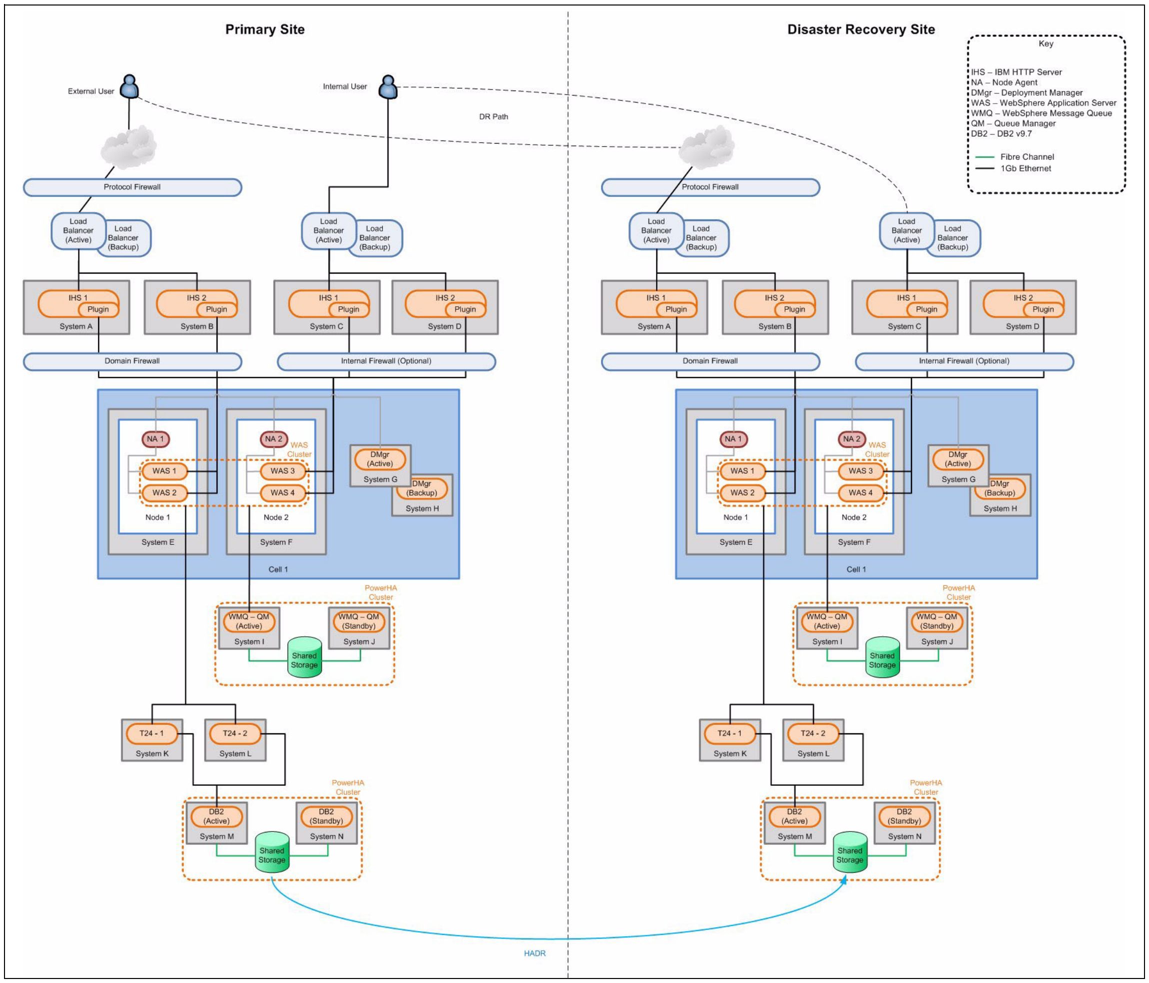


Figure 4-1 Reference configuration for T24 by using PowerHA for high availability and DB2 HADR for disaster recovery

4.2 Reference configuration with DB2 pureScale 9.8

In this configuration, DB2 pureScale provides high availability for the database layer. Other layers (WebSphere, MQ, and T24) are the same as in the first configuration. Figure 4-2 shows this configuration.

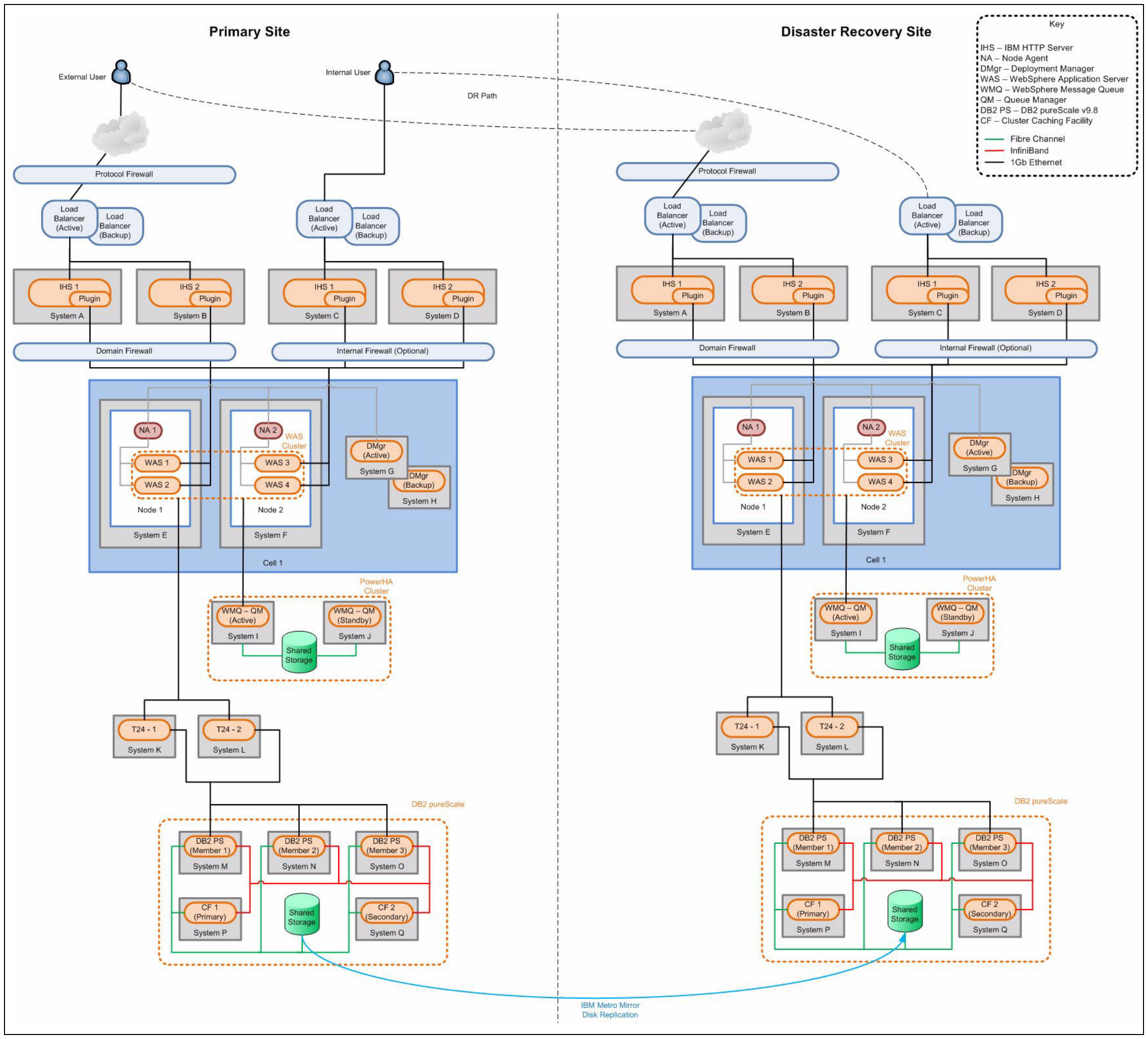


Figure 4-2 Reference configuration by using DB2 pureScale for high availability and Metro Mirror for disaster recovery

A disaster recovery capability can be provided for DB2 pureScale by using a split-site configuration as explained in “Configuring geographically dispersed DB2 pureScale clusters” at:

http://www.ibm.com/developerworks/data/library/long/dm-1104purescalegdpc/index.html?ca=drs

However, this capability was not included in the testing for this paper because a suitable network infrastructure was not supported at the test site. Therefore, only the high availability capability was tested with DB2 pureScale.

5

T24 high availability and disaster recovery scenarios

This chapter explores the scenarios for testing high availability and disaster recovery situations on DB2 9.7 and DB2 pureScale. The scenarios each underwent the following tests:

- Read test. The read query runs in two modes. An automatic refresh query is started from a T24 Browser session, or scripts are initiated from the JMeter to simulate Browser Reading of the customer details. Each server element is stopped and started during the process. This test verifies whether the read query continues to run throughout the process. Additionally this test validates whether the stack is configured correctly.
- Write test. The write test examines the data of 10,000 customers per 10 different users that is pushed to JMeter through the Browser channel. Each server element is stopped and restarted. Through each stage of failover, we expect the same number of customers to be added to the database. When the testing goes through the failover stages, the customer data is backed out and then rerun each time.

- Close-of-business (COB) test. The T24 batch COB process is run from the T24 application servers. No T24 Browser, WebSphere, or queues are involved in this test. Failover of the T24 servers and database instances are tested during the COB run. The COB was set up to run as a multiserver with a halt before and after the interest capitalization (IC) COB job. The failover tests were run during the IC COB job.

This chapter includes the following sections:

- Scenario 1: High availability test by using DB2 pureScale with an active-active configuration
- Scenario 2: High availability test with DB2 9.7 in active-passive mode
- Scenario 3: Disaster recovery test with DB2 9.7 with an active-active configuration
- Scenario 4: Disaster recovery test with DB2 pureScale

5.1 Scenario 1: High availability test by using DB2 pureScale with an active-active configuration

In this test case, the database is configured to use DB2 pureScale running a multinode active-active cluster. Figure 5-1 on page 45 shows a high-level view of the components that were used.

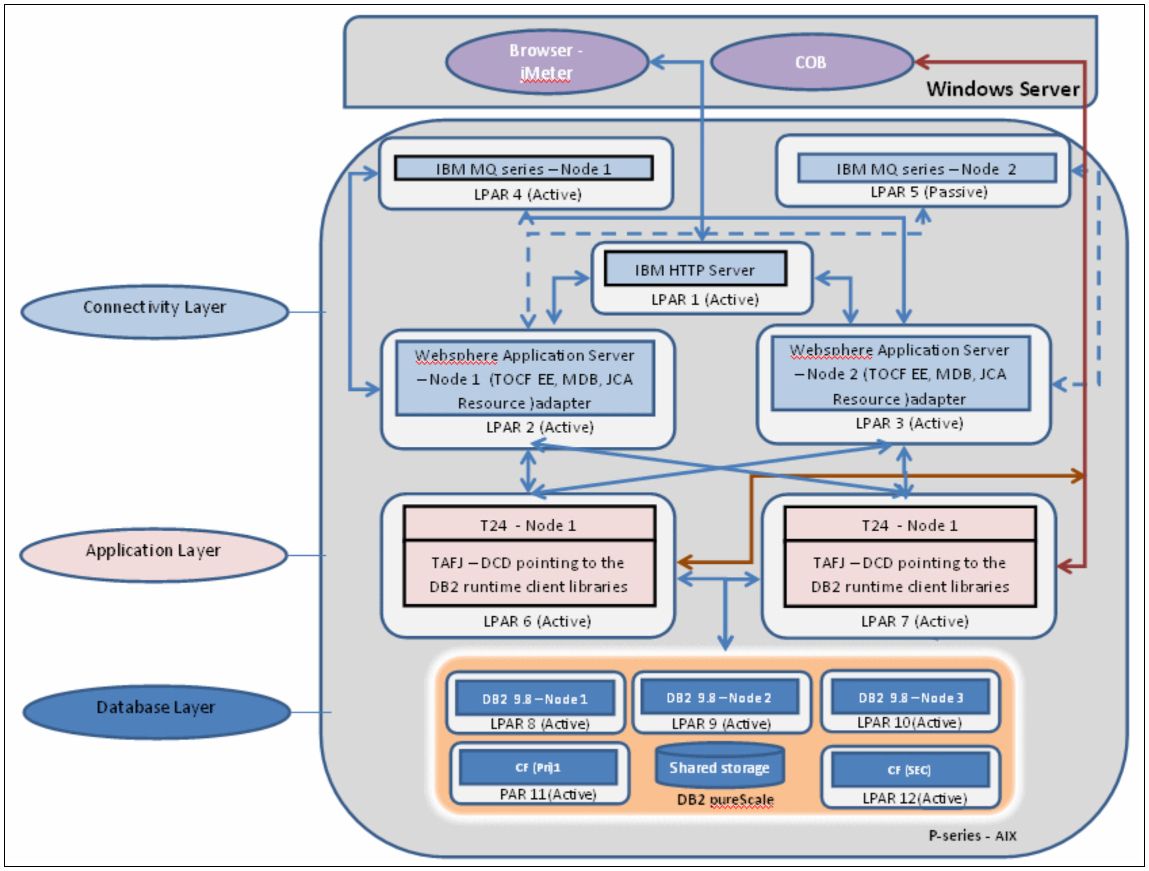


Figure 5-1 Scenario 1: Testing DB2 pureScale recovery on a database node failure

The tests in the following sections were performed on this configuration.

5.1.1 Read tests

To verify the failover of the layers during a read query, several read tests were run.

Test 1: Investigating the WebSphere node failover during a read query

To investigate the WebSphere node failover during the read query, we performed the following steps:

1. Run the read test by using the browser.
2. Stop WebSphere node 1 (transactions failover or pass to WebSphere node 2).

3. Restart WebSphere node 1.
4. Stop WebSphere node 2 (Transactions fail over or pass to WebSphere node 1.
5. Restart WebSphere node 2.
6. Check the read status.

Test 2: Investigating the MQ node failover during a read query

To investigate the MQ node failover during a read query, we performed the following steps:

1. Run the read test by using a browser.
2. Stop IBM MQ node 1. (Transactions fail over or pass to MQ node 2.
3. Check for the read status.

Test 3: Investigating the T24 node failover during a read query

To investigate the T24 node failover during a read query, we performed the following steps:

1. Run the read test by using a browser.
2. Stop jbase_agent T24 node 1. (Transactions fail over or pass to T24 node 2.)
3. Restart jbase_agent T24 node 1.
4. Stop jbase_agent T24 node 2. (Transactions fail over or pass to T24 node 1.)
5. Restart jbase_agent T24 node 2.
6. Check for the read status.

Test 4: Investigating the DB2 pureScale elements failover during a read query

To investigate the DB2 pureScale elements failover during a read query, we performed the following steps:

1. Stop DB2 pureScale member 1. (Transactions fail over or pass to member 2.)
2. Restart DB2 pureScale member 1. (Transactions rebalance over members 1 and 2.)
3. Stop DB2 pureScale member 2. (Transactions fail over or pass to member 1.)
4. Restart DB2 pureScale member 2. (Transactions rebalance over members 1 and 2.)
5. Add a DB2 pureScale member 3. (Transactions rebalance over all members.)
6. Stop CCF primary. (Failover goes to the CCF secondary, which becomes the new CCF primary.)
7. Restart CCF secondary, which was the previous CCF primary. (CCF secondary is used in catch-up state.)

5.1.2 Write tests

To verify the failover of different layers during a write query. We ran seven different write tests.

Test 1: Investigating the WebSphere node failover during the write query

To investigate the WebSphere node failover during the write query, we performed the following steps:

1. Run the write test.
2. Stop WebSphere Node 1 (transactions failover or pass to WebSphere Node 2).
3. Restart WebSphere Node 1.
4. Stop WebSphere Node 2 (transactions failover or pass to WebSphere Node 1).
5. Restart WebSphere Node 2.
6. Check the database for customers who were inserted.
7. Back out any inserted customer data.

Test 2: Investigating the MQ Node failover during the write query

To investigate the MQ node failover during the write query, we performed the following steps:

1. Run the write test.
2. Stop IBM MQ node 1. (Transactions fail over or pass to MQ node 2.)
3. Check the database for customers who were inserted.
4. Back out any inserted customer data.

Test 3: Investigating the T24 node failover during the write query

To investigate T24 node failover during the write query, we performed the following steps:

1. Run the write test.
2. Stop jbase_agent T24 node 1. (Transactions txns fail over or pass to T24 node 2.)
3. Restart the jbase_agent T24 node 1.
4. Stop jbase_agent T24 node 2. (Transactions txns fail over or pass to T24 node 1.)
5. Restart jbase_agent T24 node 2.
6. Check the database for customers who were inserted.
7. Back out any inserted customer data.

Test 4: Investigating the DB2 pureScale member failover during the write query

To investigate the DB2 pureScale member failover during the write query, we performed the following steps:

1. Run the write test.
2. Stop DB2 pureScale member 1. (Transactions fail over or pass to member 2.)
3. Restart DB2 pureScale member 1. (Transactions rebalance over members 1 and 2).
4. Check the database for customers who were inserted.
5. Back out any inserted customer data.

Test 5: Investigating the DB2 pureScale member failover during the write query

To investigate the DB2 pureScale member failover during the write query, we performed the following steps:

1. Run the write test.
2. Stop DB2 pureScale member 2. (Transactions fail over or pass to member 1.)
3. Restart DB2 pureScale member 2. (Transactions rebalance over members 1 and 2).
4. Check database for customers who were inserted.
5. Back out any inserted customer data.

Test 6: Investigating the DB2 pureScale member addition during the write query

To investigate the DB2 pureScale member addition during the write query, we performed the following steps:

1. Run the write test.
2. Add a DB2 pureScale member 3. (Transactions rebalance over all members.)
3. Check the database for customers who were inserted.
4. Back out any inserted customer data.

Test 7: Investigating the DB2 pureScale elements failover during the write query

To investigate the DB2 pureScale elements failover during the write query, we performed the following steps:

1. Run the write test.
2. Stop CCF primary. (Failover goes to the CCF secondary, which is the new CCF primary.)
3. Restart the CCF secondary, which was the previous CCF primary. (The CCF secondary is in a catch-up state.)
4. Check the database for any customers who were inserted.

5.1.3 COB tests

When running the IC COB process, we ran several tests as explained in the following sections.

Test 1: Investigating DB2 pureScale member 1 failover during the IC COB process

To investigate the DB2 pureScale member 1 failover during the IC COB process, we performed the following steps:

1. Run the COB test.
2. Stop DB2 pureScale member 1. (Transactions fail over or pass to Member 2.)
3. Restart DB2 pureScale member 1. (Transactions rebalance over members 1 and 2.)
4. Check for COB errors.
5. Back out any COB data.

Test 2: Investigating DB2 pureScale member 2 failover during the IC COB process

To investigate the DB2 pureScale member2 failover during the IC COB process, we performed the following steps:

1. Run the COB test.
2. Stop DB2 pureScale member 2. (Transactions fail over or pass to member 1.)
3. Restart DB2 pureScale member 2. (Transactions rebalance over members 1 and 2.)
4. Check for COB errors.
5. Back out any COB data.

Test 3: Investigating the DB2 pureScale member addition during the IC COB process

To investigate the DB2 pureScale member addition during the IC COB process, we performed the following steps:

1. Run the COB test.
2. Add a DB2 pureScale member 3. (Transactions rebalance over all members.)
3. Check for COB errors.
4. Back out any COB data.

Test 4: Investigating the DB2 pureScale elements failover during the IC COB process

To investigate the DB2 pureScale elements failover during the IC COB process, we performed the following steps:

1. Run the COB test.
2. Stop the CCF primary. (Failover goes to the CCF secondary, which is the new CCF primary.)
3. Restart the CCF secondary, which is the previous CCF primary. (The CCF secondary is in a catch-up state.)
4. Check for COB errors.

Test 5: Investigating the T24 node failover during the IC COB process

To investigate the T24 node failover during the IC COB process, we performed the following steps:

1. Run the COB test.
2. Stop T24 node 1. (Transactions fail over or pass to T24 node 2).
3. Restart T24 node 1.
4. Check for COB errors.
5. Back out any COB data.

Test 6: Investigating the T24 node failover during the IC COB process by using T24 Service Manager termination

To investigate the T24 node failover at T24 Service Manager termination during the IC COB process, we performed the following steps:

1. Run the COB test.
2. Stop T24 node 2. (Transactions fail over or pass to T24 node 1.)
3. Restart T24 node 2.
4. Check for COB errors.
5. Back out any COB data.

5.2 Scenario 2: High availability test with DB2 9.7 in active-passive mode

In this test case, the database is configured in a PowerHA cluster in active-passive mode. The rest of the stack is configured in active-active mode, except for WebSphere MQ, which is used in active-passive mode. Figure 5-2 on page 51 illustrates the setup for testing the different layers. It shows a high-level view of the components that are used in this test. For information about the configuration, see Chapter 6, “Test configuration setup” on page 55.

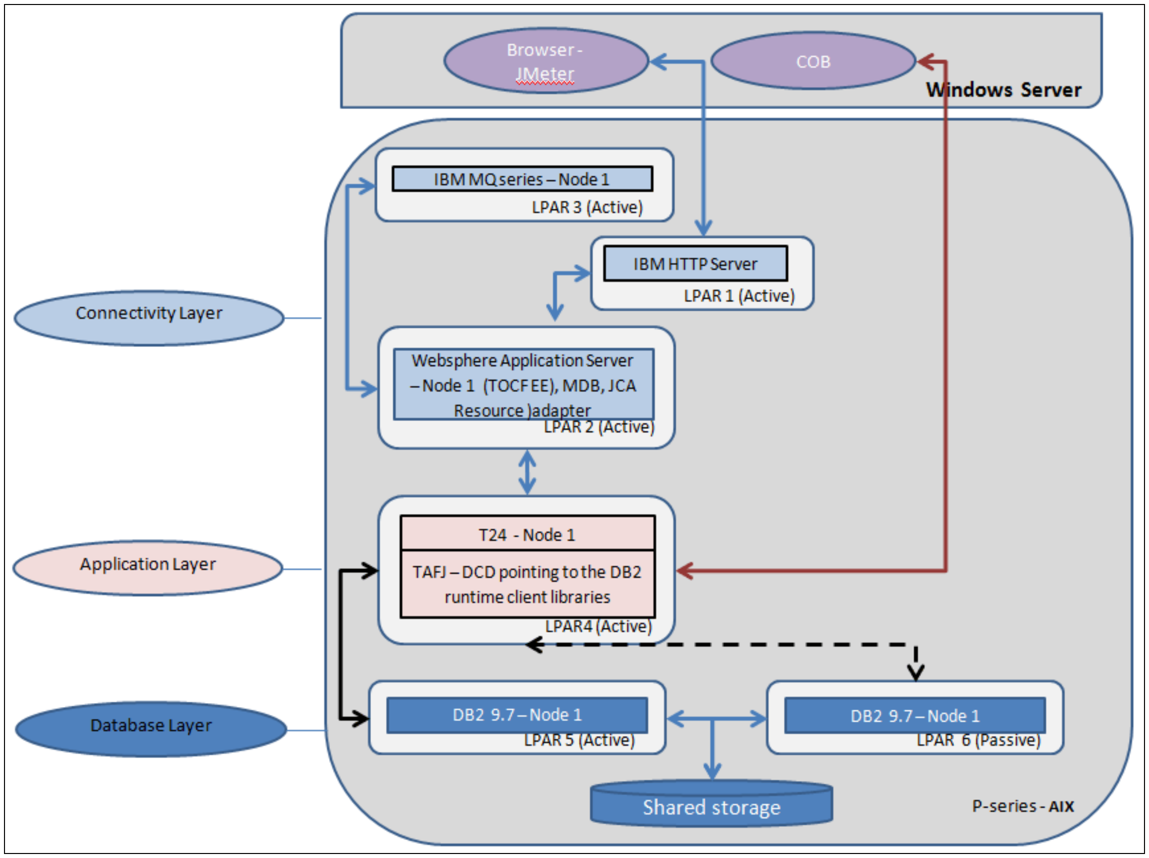


Figure 5-2 Scenario 2: DB2 9.7 configuration for the high availability test

Only the failover of the database is verified in this test. We did not test failures in the rest of the stack to avoid repeating the failures tested in 5.1, “Scenario 1: High availability test by using DB2 pureScale with an active-active configuration” on page 44. On failure of the active database node, PowerHA initiates an automatic takeover by the passive node.

The following sections show the tests that we performed for this configuration.

5.2.1 Read test

To test the T24 application failover during a database failure when running a read-only query, we performed the following steps:

1. Run the read test through browser.
2. Stop DB2 9.7 node 1. (Transactions fail over or pass to DB2 9.7 node 2.)
3. Check for the read status.

5.2.2 Write test

To test the T24 application failover during a database failure when running a write query, we performed the following steps:

1. Run the write test.
2. Stop DB2 9.7 node 1. (Transactions fail over or pass to DB2 9.7 node 2.)
3. Check the database for customers who were inserted.
4. Back out any inserted customer data.

5.2.3 COB test

Run the COB test as explained in Appendix A, “Configuration and testing details” on page 87. We performed the following steps:

1. Run the COB test.
2. Stop DB2 9.7 node 1. (Transactions fail over or pass to DB2 9.7 node 2.)
3. Check for COB errors.
4. Back out any COB data.

5.3 Scenario 3: Disaster recovery test with DB2 9.7 with an active-active configuration

A standby machine was used and configured with a standby database instance. This standby machine was populated by using high availability and disaster recovery from an active database instance to simulate a primary site. Upon failure of the active database, a manual high availability disaster recovery (HADR) take over occurs by the standby database.

The remainder of the stack at the primary can be used without high availability. The primary application stack is reconnected to the disaster recovery stack after the high availability and disaster recovery takeover is complete to simulate a disaster recovery application stack.

Figure 5-3 shows a high-level view of the components that are used in this scenario.

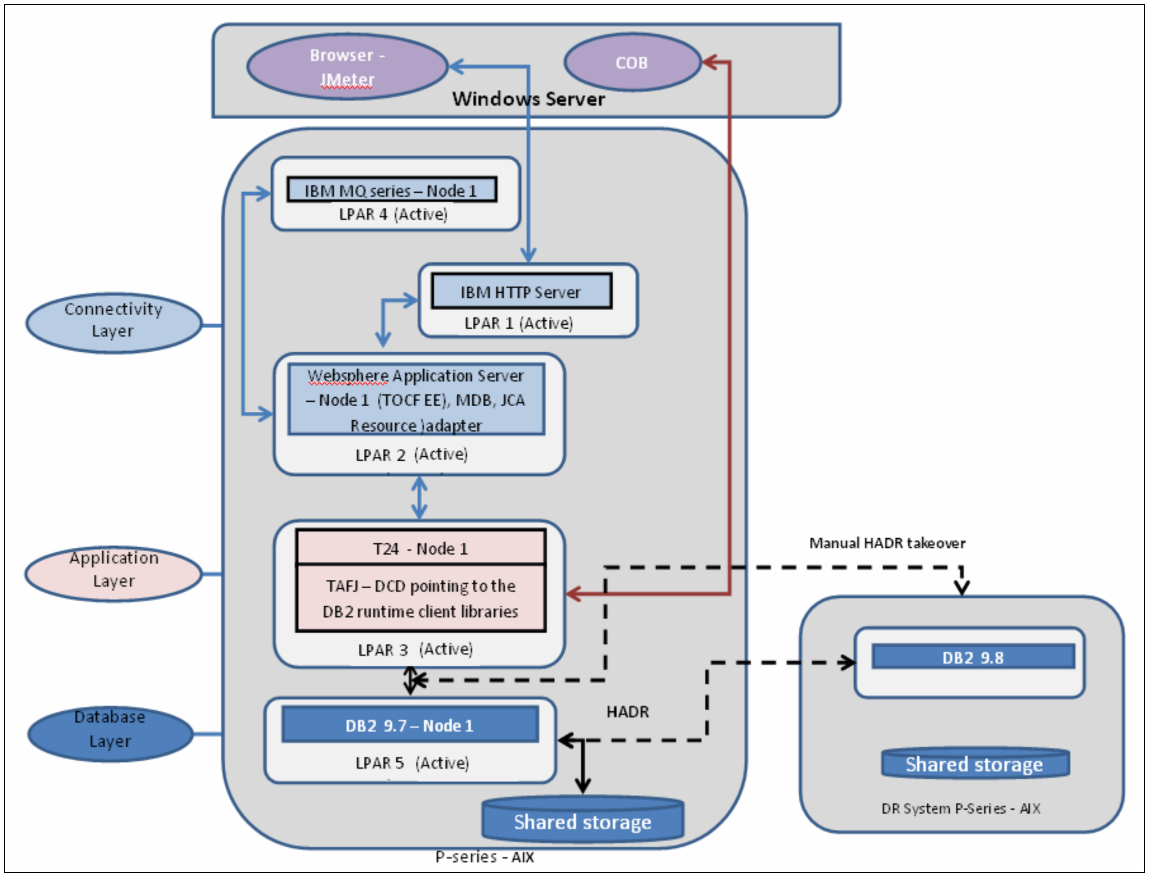


Figure 5-3 Scenario 3: DB2 HADR configuration for the disaster recovery test

Important: Only one side of the primary site stack must be used.

5.3.1 Read test

To test the T24 application failover during a database failure when running a read-only query, we performed the following steps:

1. Run the read test by using a browser.
2. Stop DB2 9.7 node 1. (Transactions fail over or pass to DB2 9.7 node 2.)
3. Check for the read status.

5.3.2 Write test

To test the T24 application failover during a database failure when running a write query, we performed the following steps:

1. Run the write test.
2. Stop DB2 9.7 node 1. (Transactions fail over or pass to DB2 9.7 node 2.)
3. Check the database for customers who were inserted.
4. Back out any inserted customer data.

5.3.3 COB test

Run the COB test as explained in Appendix A, "Configuration and testing details" on page 87. We performed the following steps:

1. Run the COB test.
2. Stop DB2 9.7 node 1. (Transactions fail over or pass to DB2 9.7 node 2.)
3. Check for COB errors.
4. Back out any COB data.

5.4 Scenario 4: Disaster recovery test with DB2 pureScale

The preferred option for disaster recovery with DB2 pureScale is to use a geographically dispersed cluster configuration. This option is explained in 3.4, "DB2 pureScale 9.8 feature" on page 30.

No testing: It was not possible to test this configuration during the test project because the necessary fiber network facilities were not available.

6

Test configuration setup

Temenos and IBM undertook investigated the high availability and disaster recovery options for T24. They set out to prove that T24 functions correctly when using the high availability and disaster recovery reference configurations and in the defined failover situations. The hardware and software configurations were set up at the IBM Innovation Centre at IBM Hursley, United Kingdom. IBM and Temenos experts collaborated on the implementation and testing processes.

This chapter includes the following sections:

- Hardware and network configuration of the test system
- Configuring and setting up the test systems

6.1 Hardware and network configuration of the test system

Figure 6-1 shows the configuration of the hardware partitions and network for the test system used in this project.

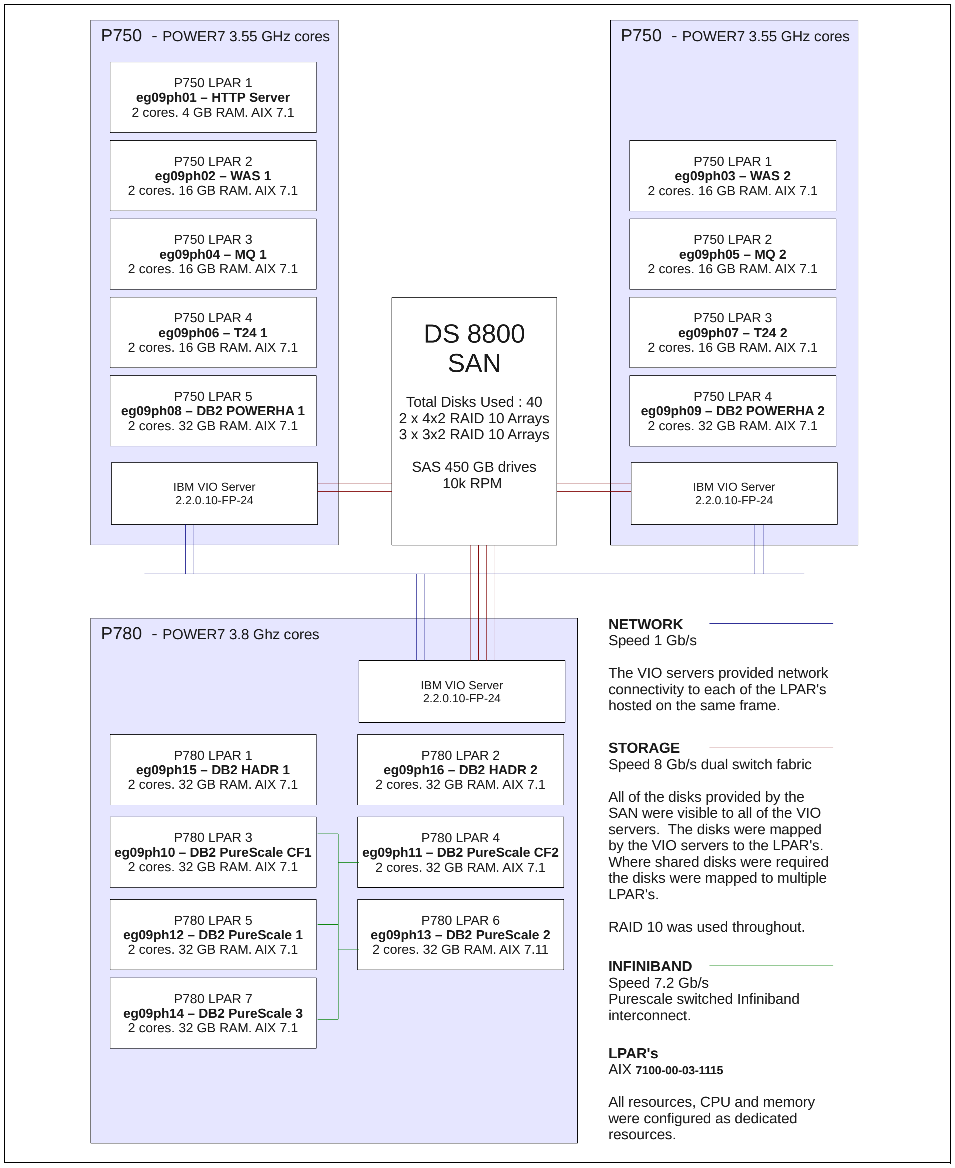

P750 - POWER7 3.55 GHz cores
P750 LPAR 1 **eg09ph01 – HTTP Server** 2 cores. 4 GB RAM. AIX 7.1
P750 LPAR 2 **eg09ph02 – WAS 1** 2 cores. 16 GB RAM. AIX 7.1
P750 LPAR 3 **eg09ph04 – MQ 1** 2 cores. 16 GB RAM. AIX 7.1
P750 LPAR 4 **eg09ph06 – T24 1** 2 cores. 16 GB RAM. AIX 7.1
P750 LPAR 5 **eg09ph08 – DB2 POWERHA 1** 2 cores. 32 GB RAM. AIX 7.1
IBM VIO Server 2.2.0.10-FP-24
P750 - POWER7 3.55 GHz cores
P750 LPAR 1 **eg09ph03 – WAS 2** 2 cores. 16 GB RAM. AIX 7.1
P750 LPAR 2 **eg09ph05 – MQ 2** 2 cores. 16 GB RAM. AIX 7.1
P750 LPAR 3 **eg09ph07 – T24 2** 2 cores. 16 GB RAM. AIX 7.1
P750 LPAR 4 **eg09ph09 – DB2 POWERHA 2** 2 cores. 32 GB RAM. AIX 7.1
IBM VIO Server 2.2.0.10-FP-24
DS 8800 SAN
Total Disks Used : 40
2 x 4x2 RAID 10 Arrays
3 x 3x2 RAID 10 Arrays
SAS 450 GB drives
10k RPM
P780 - POWER7 3.8 Ghz cores
IBM VIO Server 2.2.0.10-FP-24
P780 LPAR 1 **eg09ph15 – DB2 HADR 1** 2 cores. 32 GB RAM. AIX 7.1
P780 LPAR 2 **eg09ph16 – DB2 HADR 2** 2 cores. 32 GB RAM. AIX 7.1
P780 LPAR 3 **eg09ph10 – DB2 PureScale CF1** 2 cores. 32 GB RAM. AIX 7.1
P780 LPAR 4 **eg09ph11 – DB2 PureScale CF2** 2 cores. 32 GB RAM. AIX 7.1
P780 LPAR 5 **eg09ph12 – DB2 PureScale 1** 2 cores. 32 GB RAM. AIX 7.1
P780 LPAR 6 **eg09ph13 – DB2 PureScale 2** 2 cores. 32 GB RAM. AIX 7.11
P780 LPAR 7 **eg09ph14 – DB2 PureScale 3** 2 cores. 32 GB RAM. AIX 7.1
NETWORK
Speed 1 Gb/s
The VIO servers provided network connectivity to each of the LPAR's hosted on the same frame.
STORAGE
Speed 8 Gb/s dual switch fabric
All of the disks provided by the SAN were visible to all of the VIO servers. The disks were mapped by the VIO servers to the LPAR's. Where shared disks were required the disks were mapped to multiple LPAR's.
RAID 10 was used throughout.
INFINIBAND
Speed 7.2 Gb/s
Purescale switched Infiniband interconnect.
LPAR's
AIX 7100-00-03-1115
All resources, CPU and memory were configured as dedicated resources.

Figure 6-1 Test configuration for hardware partitions and networks

The three IBM Power Systems servers were configured with logical partitions (LPARS) to provide the necessary LPARs for testing as shown in Figure 6-1 on page 56. These LPARS provided the nodes that make up the connectivity, application, and database layers. By using this configuration, the first three test scenarios were created as explained in Chapter 5, “T24 high availability and disaster recovery scenarios” on page 43.

The testing environment included setting up a Windows server to run the JMeter injection scripts, browser, and close-of-business (COB) jobs.

6.2 Configuring and setting up the test systems

Tip: The configuration of the software components is intended as a guide for setting up a similar configuration.

WebSphere Application Server Network Deployment V7 was used for the Java 2 Platform, Enterprise Edition (J2EE) servers. Two application servers were configured in each of two separate LPARs with two individual node agents. The HTTP server was configured as an unmanaged server. The connectivity layer used Temenos Open Connectivity Framework - Enterprise Edition (TOCF-EE) and browser servlet components from T24 R11.

IBM WebSphere MQ Server V7 was configured as an active-passive cluster that used PowerHA clustering and a shared storage configuration.

For the high availability active-active setup that uses DB2 pureScale, DB2 9.8 was installed with three database members and two coupling facilities. Each of the members and coupling facilities are in different partitions. For more information, see 5.1, “Scenario 1: High availability test by using DB2 pureScale with an active-active configuration” on page 44.

For the high availability active-passive tests, DB2 9.7 was installed in two partitions with shared storage and failover was managed by PowerHA. For more information, see 5.2, “Scenario 2: High availability test with DB2 9.7 in active-passive mode” on page 51.

For the disaster recovery scenario, DB2 9.7 was installed in two partitions by using DB2 high availability disaster recovery (HADR). For more information, see 5.3, “Scenario 3: Disaster recovery test with DB2 9.7 with an active-active configuration” on page 52.

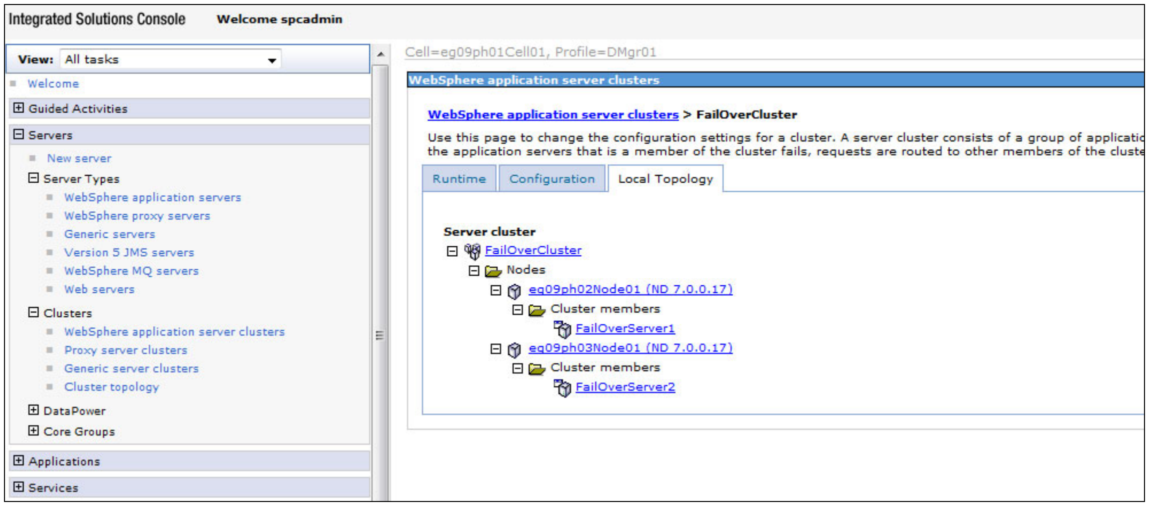
6.2.1 WebSphere Application Server

Install the WebSphere Application Server and the HTTP server basic version, by using the *WebSphere Application Server Network Deployment V7 Installation Guide* from IBM at:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.was_v7/was/7.0/InstallationAndMigration/WASv7_InstallationLab.pdf

Then, follow the guidance in *High Availability T24 Browser & JCA - IBM WebSphere Application Server V7.0 Network Deployment* to configure the WebSphere Application Server for T24. You can request this document from Temenos.

The cluster was configured with two individual node agents with one application server each, as shown in Figure 6-2.

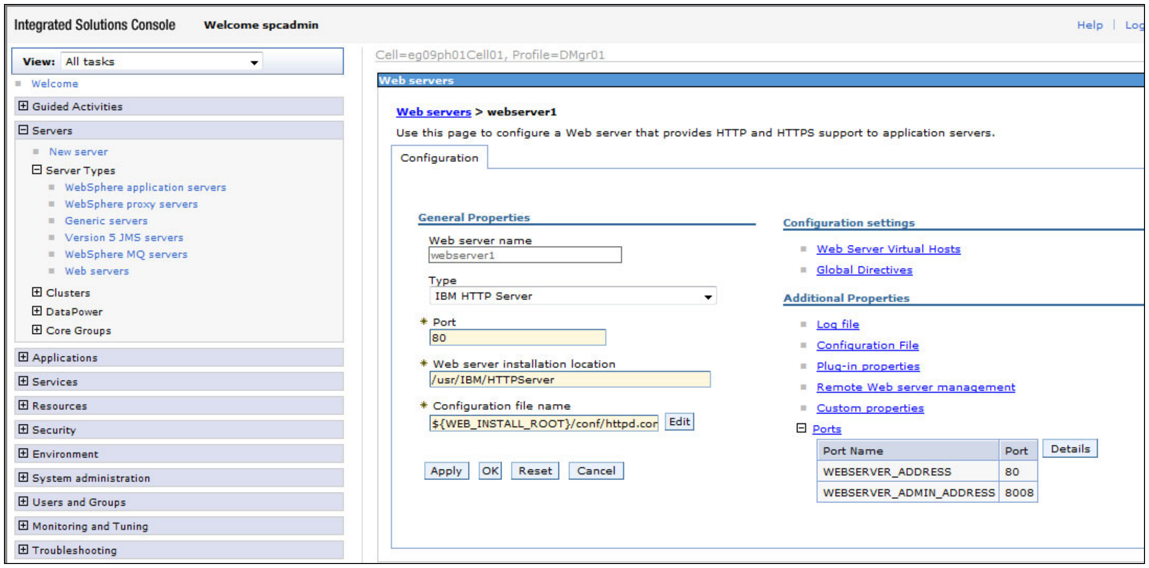


Use this page to change the configuration settings for a cluster. A server cluster consists of a group of applicati
the application servers that is a member of the cluster fails, requests are routed to other members of the cluste

Figure 6-2 WebSphere Application Server configuration

6.2.2 HTTP server

The HTTP server was configured as an unmanaged server in a stand-alone server and added to the WebSphere Application Server, as shown in Figure 6-3.



Cell=eg09ph01Cell01, Profile=DMgr01

Web servers

Web servers > webserver1

Use this page to configure a Web server that provides HTTP and HTTPS support to application servers.

Configuration

General Properties

Web server name: webserver1

Type: IBM HTTP Server

* Port: 80

* Web server installation location: /usr/IBM/HTTPServer

* Configuration file name: ${WEB_INSTALL_ROOT}/conf/httpd.cor

Configuration settings

- Web Server Virtual Hosts
- Global Directives

Additional Properties

- Log file
- Configuration File
- Plug-in properties
- Remote Web server management
- Custom properties

Ports

Port Name	Port
WEBSERVER_ADDRESS	80
WEBSERVER_ADMIN_ADDRESS	8008

Figure 6-3 HTTP Server configuration

Example 6-1 shows the main IBM HTTP server configuration file. It contains the configuration directives that give the server its instructions.

Example 6-1 HTTP plug-in configuration file

```
WebSpherePluginConfig
/usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-cfg.xml


...

   <ServerCluster CloneSeparatorChange="false" GetDWLMTable="true"
IgnoreAffinityRequests="true" LoadBalance="Round Robin" Name="FailOverCluster"
PostBufferSize="64" PostSizeLimit="-1" RemoveSpecialHeaders="true"
RetryInterval="60">
      <Server CloneID="168fk2ap5" ConnectTimeout="5" ExtendedHandshake="false"
LoadBalanceWeight="2" MaxConnections="-1" Name="eg09ph02Node01_FailOverServer1"
ServerIOTimeout="120" WaitForContinue="false">
         <Transport Hostname="eg09ph02" Port="9080" Protocol="http"/>
         <Transport Hostname="eg09ph02" Port="9443" Protocol="https">
            <Property Name="keyring"
Value="/usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-key.kdb"/>
```

```
            <Property Name="stashfile"
Value="/usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-key.sth"/>
         </Transport>
      </Server>
      <Server CloneID="168fn4cv0" ConnectTimeout="5" ExtendedHandshake="false"
LoadBalanceWeight="2" MaxConnections="-1" Name="eg09ph03Node01_FailOverServer2"
ServerIOTimeout="120" WaitForContinue="false">
         <Transport Hostname="eg09ph03" Port="9080" Protocol="http"/>
         <Transport Hostname="eg09ph03" Port="9443" Protocol="https">
            <Property Name="keyring"
Value="/usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-key.kdb"/>
            <Property Name="stashfile"
Value="/usr/IBM/HTTPServer/Plugins/config/webserver1/plugin-key.sth"/>
         </Transport>
      </Server>
      <PrimaryServers>
         <Server Name="eg09ph02Node01_FailOverServer1"/>
         <Server Name="eg09ph03Node01_FailOverServer2"/>
      </PrimaryServers>
   </ServerCluster>

...
```

Regenerating the plug-in

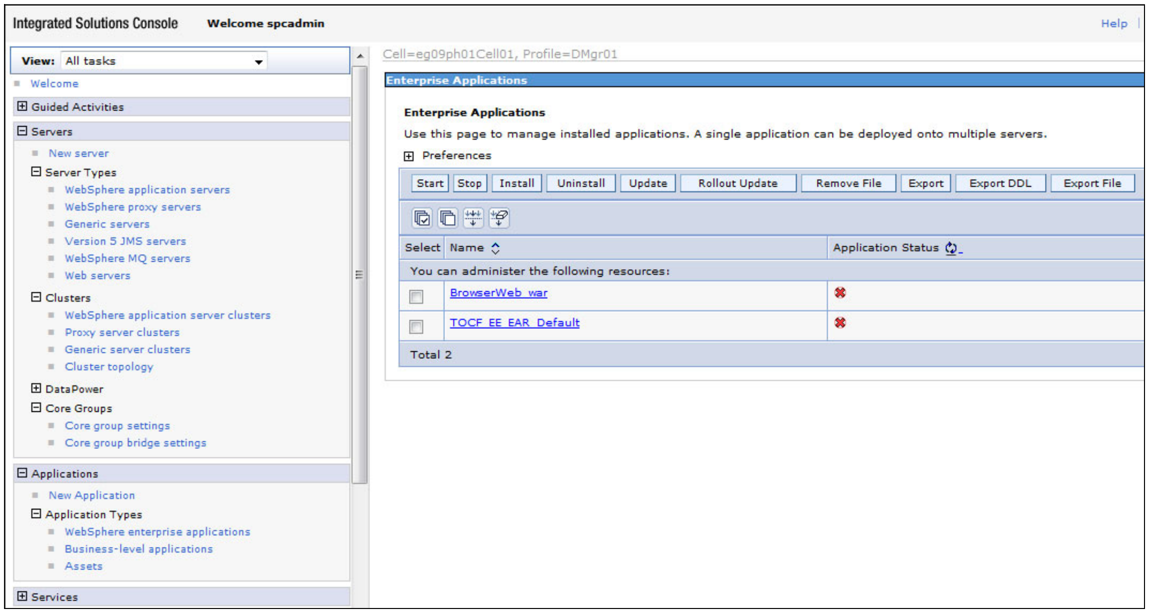
Because the HTTP server is unmanaged, the plug-in must be regenerated manually every time a change occurs. To regenerate the plug-in:

1. In the WebSphere Application Server administration console, expand **Environment**, and click **Update global Web server plug-in configuration**. The `/usr/IBM/websphere/appserver/profiles/dmgr01/config/cells` file is then generated in the deployment manager folder.
2. Copy the generated file to the `/usr/IBM/HTTPServer/plugins/config/webserver` folder.
3. In the `plugin_config.xml` file, change the following parameters:
   a. For Loglevel, enter `/usr/IBM/HTTPServer/plugins`.
   b. For `Pluginstallroot`, enter `=/usr/IBM/HTTPServer/plugins`.
   c. For `Getdwlntable=`, enter `True`.
   d. For `Ignoreaffinityrequest`, enter `False`.
   e. For `Loadbalanceweight`, enter `20` for server 1.
   f. For `Loadbalanceweight`, enter `19` for server 2.

6.2.3 Deployment manager and application server

Follow the *T24 guide for High Availability T24 Browser & JCA - IBM WebSphere Application Server v7.0 Network Deployment* to configure the WebSphere Application Server for T24. You can request this document from Temenos.

In the test configuration, the deployment manager is installed on the same server as the HTTP server. The required T24 applications are installed during this process. See Figure 6-4.



Cell=eg09ph01Cell01, Profile=DMgr01

Enterprise Applications

Use this page to manage installed applications. A single application can be deployed onto multiple servers.

Select	Name	Application Status
You can administer the following resources:		
☐	BrowserWeb_war	✖
☐	TOCF_EE_EAR_Default	✖
Total 2		

Figure 6-4 Deployment manager configuration

6.2.4 WebSphere MQ Server

WebSphere MQ Server (MQ) was set up in the active-passive mode by using the PowerHA cluster. The following sections outline the details of implementing PowerHA for MQ.

Configuring the PowerHA single NIC

In this example, we created the PowerHA cluster by using the smitty configuration assistant, by using only one network interface card (NIC) per server. In a production environment, you might need two NICs per server, within the PowerHA configuration, on the EtherChannel on the LPAR, or virtualized

EtherChannel on the Virtual I/O Server (VIOS). Where you need the NICs depends on your hardware and system constraints.

Defining the settings

Before you build your cluster, define the following settings:

1. Create your shared volume group or groups:
   a. Enter the following commands:
   ```
   smitty
   mkvg select VG type
   ```
   b. Enter the following information:
   ```
   VOLUME GROUP name                           []
    Physical partition SIZE in megabytes
   * PHYSICAL VOLUME names                     []
    Force the creation of a volume group?       no
    Activate volume group AUTOMATICALLY         yes
       at system restart?
    Volume Group MAJOR NUMBER                    []
     Create VG Concurrent Capable?              no
   ```
   c. Enter the details for the volume group (VG):
      i. For Activate VG Automatically at system restart, enter `NO`.
      ii. For Create VG Concurrent Capable, enter `enhanced concurrent`.
2. Populate the `/etc/hosts` and the `/usr/es/sbin/cluster/netmon.cf` files on both servers with the IP details that you will use for the cluster. The cluster will share the persistent addresses if they exist with the service address. The `netmon.cf` file is the same, except that you need only the IP address of the networks in use and the service address, for example:
   ```
   172.19.81.94
   172.19.81.95
   172.19.83.94
   ```
   These addresses contain the two boot adapters and the service address, and no other information or comments.
3. Create and test application stop and start scripts to ensure that they work. You do not need to populate the scripts to all the nodes in the cluster. Instead, populate them to the primary from which you are working. The creation populates the rest later. Example 6-2 shows the script for starting and stopping the application.

   Example 6-2 Script for starting and stopping the application
   ```
   #  vi /home/db2inst1/mqstart.sh
   #!/bin/ksh
   ```

```
# A simple wrapper script to switch to the mqm user.
su mqm -c /usr/local/bin/start.sh T24QM $*

#  vi /home/db2inst1/mqstop.sh
#!/bin/ksh
# A simple wrapper script to switch to the mqm user.
su mqm -c /usr/local/bin/stop.sh T24QM 10 $*
```

The setup is now completed.

Configuring the cluster

Now configure the cluster:

1. By using the two-node configuration assistant, configure the cluster as shown in Example 6-3.

Example 6-3 Cluster configuration

```
# smitty cl_smartassistant · Two-Node Cluster Configuration Assistant ·

* Primary / Local Node                                      eg09ph04
* Communication Path to Takeover Node                      []
* Application Server Name                                  [eg09ph04_app_01]
* Application Server Start Script                          []
* Application Server Stop Script                           []
* Service IP Label              []
  Netmask(IPv4)/Prefix Length(IPv6)                        []
* Resource Group Name                                      [eg09ph04_rg_01]
* Cluster Name                                             [eg09ph04_cluster]
```

Where:

– The primary node is the current node you are on.

– The communication path is the standby node. The details must be in the `/etc/hosts` directory and can be selected from a list (F4).

– The Application Server Name is the name you want to call your application, or use the default.

– Application server start and stop scripts must be populated with the scripts that you created in step 3 on page 62.

– The Service IP label is the service address in the `/etc/hosts` directory that must be failed between the nodes, so that it can be selected from the list (F4).

– Netmask can be left blank.

– For Resource Group Name, more can be added later if needed.

– Cluster name is the current name that is assigned to the cluster.

Press Enter and create the cluster.

Because this cluster is a single network interface card cluster, you see warning messages that is related to this which can be ignored. Also, the cluster issues a warning message that relates to the application monitor not being configured. The application monitor configuration is not needed for this example, but can be tailored later by running the following command:

```
smitty cm_cfg_appmon
```

2. Confirm the status of the cluster (Example 6-4).

Example 6-4 Status of the cluster

```
# /usr/es/sbin/cluster/utilities/cltopinfo
Cluster Name: Temenos_db2cluster1
Cluster Connection Authentication Mode: Standard
Cluster Message Authentication Mode: None
Cluster Message Encryption: None
Use Persistent Labels for Communication: No
There are 2 node(s) and 2 network(s) defined
NODE eg09ph04:
        Network net_diskhb_01
                eg09ph04_hdisk1_01      /dev/hdisk1
        Network net_ether_01
                temdb2svc       172.19.83.94
                eg09ph04        172.19.81.94
NODE eg09ph05:
        Network net_diskhb_01
                eg09ph05_hdisk1_01      /dev/hdisk1
        Network net_ether_01
                temdb2svc       172.19.83.94
                eg09ph05        172.19.81.95

Resource Group Temenos_db2rg1
        Startup Policy   Online On Home Node Only
        Fallover Policy  Fallover To Next Priority Node In The List
        Fallback Policy  Never Fallback
        Participating Nodes      eg09ph04 eg09ph05
        Service IP Label                temdb2svc
```

From this example, you can see the cluster is called `Temenos_db2cluster1`. The disk-to-disk heartbeat network is called `net_diskhb_01` and uses hdisk1 on nodes `eg09ph04` and `eg09ph05`. To see more information about the configured application and topology of the cluster, enter the following command:

```
# /usr/es/sbin/cluster/utilities/cldisp
```

3. Create a logical volume for the cluster:
   a. Enter the following command:
      ```
      # smitty cl_lvm · Logical Volumes
      ```
   b. Add a Logical volume.
   c. Select the volume group.
   d. Select the disk if you need to ensure specific devices.
   e. Create the logical volume as required with, for example, name, FS type, and PP numbers, leaving some PP numbers available for lvlog auto creation.
4. Create the file systems:
   a. Enter the following command:
      ```
      # smitty cl_lvm
      ```
   b. Select **File Systems** → **Add a File System**.
   c. Select the volume group.
   d. Select the file system type.
   e. Select the previously defined logical volume. Give it a mount point, and create as many logical volumes as required.
5. Confirm the creation as shown in Example 6-5.

Example 6-5 Confirming the cluster creation

```
# /usr/es/sbin/cluster/sbin/cl_showfs2
File System    Volume Group      Resource Group      Node List
 /db2log        db2logvg        Temenos_db2rg1    eg09ph04,eg09ph05
 /db2           db2vg           Temenos_db2rg1    eg09ph04,eg09ph05
 /backup        db2vg           Temenos_db2rg1    eg09ph04,eg09ph05
```

Running the cluster

To run the cluster:

1. Start the Cluster Information Daemon by running the following command:
   ```
   # smitty clstart
   ```
   Run this command for each node in the cluster. Start one node in the cluster, wait for it to stabilize by using the **`clstat`** command, and then start any further stand-by nodes. Each time, make sure that the cluster is stable (see `SubState` in **`clstat`**) before moving onto the next node.

2. After the cluster is up, monitor the status as shown in Example 6-6.

Example 6-6 Cluster status

```
# /usr/es/sbin/cluster/utilities/clRGinfo
-----------------------------------------------------------------------------
Group Name     Group State                  Node
-----------------------------------------------------------------------------
Temenos_db2rg1 OFFLINE                      eg09ph04
               ONLINE                       eg09ph05
```

From here, you can see that the cluster resource is running on node `eg09ph04`. The output of the **`clstat`** command (Example 6-7) shows an interactive version of the cluster status, which is shown when the **`clinfoES`** daemon is running.

Example 6-7 Output of the clstate command

```
# /usr/es/sbin/cluster/clstat
               clstat - HACMP Cluster Status Monitor
                -------------------------------------

Cluster: Temenos_db2cluster1    (1089015638)
Wed 31 Aug 09:55:30 2011
                State: UP               Nodes: 2
                SubState: STABLE

        Node: eg09ph04          State: UP
           Interface: eg09ph04 (1)             Address: 172.19.81.94
                                               State:   UP
           Interface: eg09ph04_hdisk1_01 (0)           Address: 0.0.0.0
                                               State:   UP

        Node: eg09ph05          State: UP
           Interface: eg09ph05 (1)             Address: 172.19.81.95
                                               State:   UP
           Interface: eg09ph05_hdisk1_01 (0)           Address: 0.0.0.0
                                               State:   UP
           Interface: temdb2svc (1)            Address: 172.19.83.94
                                               State:   UP
           Resource Group: Temenos_db2rg1                State:  On line
```

Application monitoring

By default, PowerHA with Reliable Scalable Cluster Technology (RSCT) monitors the network infrastructure. Beyond the availability of the network that is used by clients to access the application, application health is not monitored. This fact makes configuring application monitors in PowerHA an important consideration.

For more information, see “Considerations for configuring application monitors” in *PowerHA for AIX Cookbook*, SG24-7739.

Configuring application monitors

Two types of application monitors can be configured with PowerHA:

Process monitors	Detect the termination of one or more processes of an application by using RSCT Resource Monitoring and Control (RMC). Any process that is displayed in the output of the **`ps -el`** command can be monitored by using a PowerHA process monitor.
Custom monitors	Check the health of an application with a user-written custom monitor method at user-specified polling intervals. The administrator can check for anything that can be defined as a determining factor in the health of an application. Such factors include the ability to log in to an application, to open a database, to write a dummy record, and to query the internal state of an application. Return code 0 from the user-written monitor indicates that the application is healthy and no further action is necessary. A non-zero return code indicates that the application is not healthy and recovery actions are required.

Application monitors can be configured to run in the following modes:

Long-running mode	Periodically checks whether the application is running.
Startup mode	Checks that the application successfully started.
Both modes	Performs the actions of the long-running mode and startup mode.

The SMIT fast path for application monitor configuration is **`smitty cm_cfg_appmon`** fast path.

Creating a monitor

Create a monitor by using process monitors and running smitty:

1. Enter the following command:

   ```
   smitty cm_cfg_appmon
   ```

2. Select **Configure Process Application Monitors** → **Add a Process Application Monitor**.

3. Enter the information as shown in Figure 6-5.

```
* Monitor Name                                    [Temenos_mqapp1]
* Application Server(s) to Monitor+
* Monitor Mode                                    [Long-running monitoring]+
* Processes to Monitor                            [strmqm]
* Process Owner                                   [mqm]
  Instance Count                                  [1]#
* Stabilization Interval                          [30]#
* Restart Count                                   [3]#
  Restart Interval                                [100]#
* Action on Application Failure                   [failover]+
  Notify Method                                   []
  Cleanup Method                                  [/home/db2inst1/mqstop.sh]
  Restart Method                                  [/home/db2inst1/mqstart.sh]
```

Figure 6-5 Monitor configuration

To view the details of the processes to monitor, use the **`ps -ef`** command. In this case, the **`strmqm`** process is monitored. There is more than one of **`strmqm`** process when you use the **`ps -ef`** command. However, we look only at owner ID 205, which is mqm, and for which only one instance of this process is available. Therefore, we set the “install stabilization count” parameter to 30 seconds, with three restarts at a restart interval of 100 seconds. If the application fails, it fails over.

This method is similar to custom application monitoring in that you supply a script that reports when the application is up or down.

6.2.5 T24 application

To install T24 and export the required data into the database, follow the T24 installation guide, which you can request from Temenos. Extract the appropriate T24 and TAFC driver files from the system.

Setting up locking

Use the jBASE distributed locking guide, which you can request from Temenos, to configure the locking for T24. The following example shows a locking parameter that needs to be added to the `.profile` file of the T24:

```
export JDLS=SERVER=172.19.81.96:50002,SERVER2=172.19.81.97:50002
```

To initiate the jDLS locking, enter the following command:

```
jDLS –ibs13000,50
```

Configuring the database to T24

To configure the driver to point to the DB2 server, install a DB2 client in the T24 application server location if it is situated in a separate location than the location of the DB2 server. When you alter the database, always change `config-XMLDB2` for the DB user and password initially. Then uncatalog and catalog the database in the required TCP/IP node.

Initiating the conversion program

Follow the database conversion process in the *Database Conversion User Guide*, which you can request from Temenos. Run the T24 **`rdbmsConversion`** command from the Java shell (jsh). At the end of the process, the data is converted to DB2. For the test, the already converted database was copied and extracted in the test site.

6.2.6 DB2

This section describes the DB2 pureScale and high availability configurations.

DB2 pureScale configuration

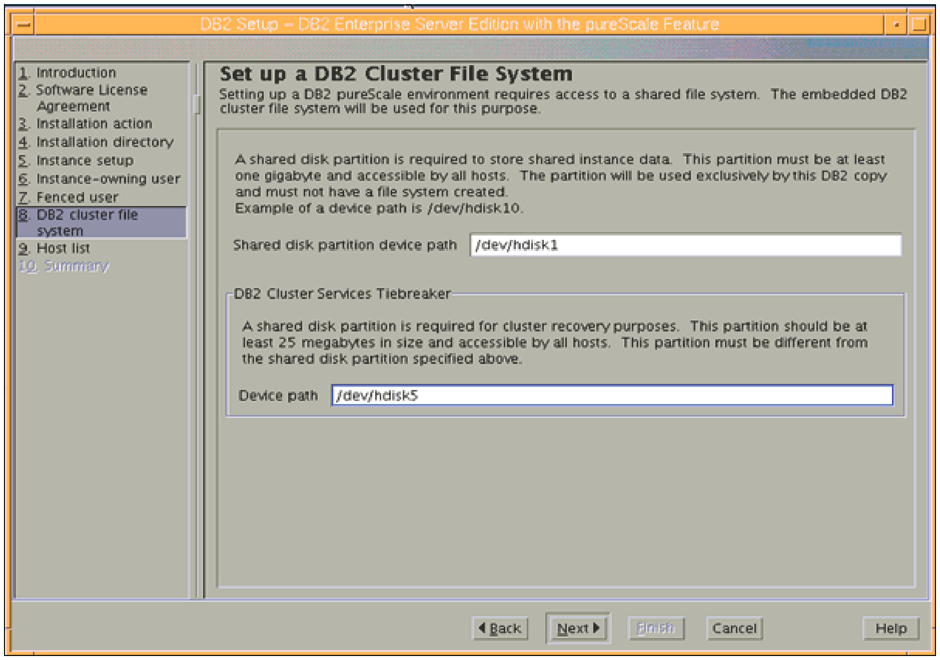
Verify whether the DB2 pureScale 9.8 AIX installation prerequisites are in place. For more information, see "Installation prerequisites for DB2 pureScale Feature (AIX)" at:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/index.jsp?topic=/com.ibm.db2.luw.sd.doc/doc/r0054850.html

Install the DB2 pureScale with three members and two control facilities. For more information, see "DB2 best practices: Upgrading to the DB2 pureScale Feature" at:

http://www.ibm.com/developerworks/data/bestpractices/purescaleupgrade/index.html

Figure 6-6 shows the DB2 pureScale configuration.



DB2 Setup - DB2 Enterprise Server Edition with the pureScale Feature

1. Introduction
2. Software License Agreement
3. Installation action
4. Installation directory
5. Instance setup
6. Instance-owning user
7. Fenced user
8. DB2 cluster file system
9. Host list
10. Summary

Set up a DB2 Cluster File System

Setting up a DB2 pureScale environment requires access to a shared file system. The embedded DB2 cluster file system will be used for this purpose.

A shared disk partition is required to store shared instance data. This partition must be at least one gigabyte and accessible by all hosts. The partition will be used exclusively by this DB2 copy and must not have a file system created.
Example of a device path is /dev/hdisk10.

Shared disk partition device path: /dev/hdisk1

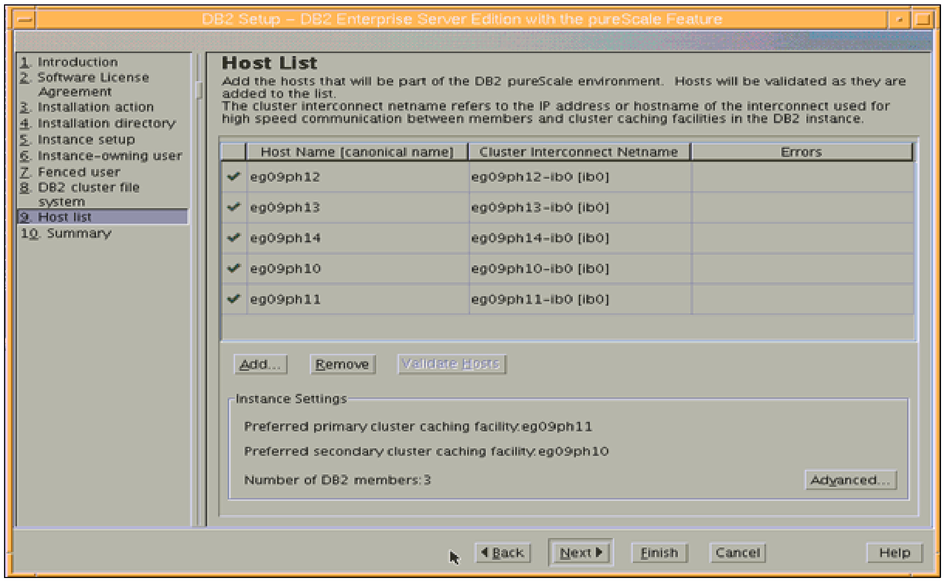
DB2 Cluster Services Tiebreaker

A shared disk partition is required for cluster recovery purposes. This partition should be at least 25 megabytes in size and accessible by all hosts. This partition must be different from the shared disk partition specified above.

Device path: /dev/hdisk5

Figure 6-6 DB2 pureScale configuration

Figure 6-7 shows the DB2 pureScale hosts list.


DB2 Setup – DB2 Enterprise Server Edition with the pureScale Feature

1. Introduction
2. Software License Agreement
3. Installation action
4. Installation directory
5. Instance setup
6. Instance-owning user
7. Fenced user
8. DB2 cluster file system
9. Host list
10. Summary

Host List

Add the hosts that will be part of the DB2 pureScale environment. Hosts will be validated as they are added to the list.
The cluster interconnect netname refers to the IP address or hostname of the interconnect used for high speed communication between members and cluster caching facilities in the DB2 instance.

	Host Name [canonical name]	Cluster Interconnect Netname	Errors
✓	eg09ph12	eg09ph12-ib0 [ib0]	
✓	eg09ph13	eg09ph13-ib0 [ib0]	
✓	eg09ph14	eg09ph14-ib0 [ib0]	
✓	eg09ph10	eg09ph10-ib0 [ib0]	
✓	eg09ph11	eg09ph11-ib0 [ib0]	

Instance Settings

Preferred primary cluster caching facility:eg09ph11

Preferred secondary cluster caching facility:eg09ph10

Number of DB2 members:3

Figure 6-7 DB2 pureScale hosts list

The following sections provide information about the DB2 pureScale configuration.

The db2nodes.cfg file

Example 6-8 shows the definitions of the database partition servers that participate in a DB2 instance.

Example 6-8 The db2nodes.cfg file

```
0 eg09ph12 0 eg09ph12-ib0 - MEMBER
1 eg09ph13 0 eg09ph13-ib0 - MEMBER
2 eg09ph14 0 eg09ph14-ib0 - MEMBER
128 eg09ph10 0 eg09ph10-ib0 - CF
129 eg09ph11 0 eg09ph11-ib0 – CF
```

db2set information

Example 6-9 shows the DB2 environment variables that are stored in the DB2 profile registry.

Example 6-9 DB2 environment variables

```
[e] DB2LIBPATH=/home/db2sdin1/sqllib/java/jdk64/jre/lib/ppc64
[i] DB2RSHCMD=/bin/ssh
[i] DB2COMM=TCPIP
```

```
[i] DB2AUTOSTART=NO
[g] DB2SYSTEM=eg09ph12
[g] DB2INSTDEF=db2sdin1
```

System database directory

Example 6-10 shows an example of a system database directory file that exists for each instance of the database manager. It contains one entry for each database that is cataloged for this instance.

Example 6-10 System database directory file

```
Number of entries in the directory = 1

Database 1 entry:

 Database alias                        = R11_HADR
 Database name                         = R11_HADR
 Local database directory              = /db2sd_20110823113125/db2sdin1
 Database release level                = e.00
 Comment                               =
 Directory entry type                  = Indirect
 Catalog database partition number     = 0
 Alternate server hostname             = eg09ph14
 Alternate server port number          = 50000

After the installation we can have a look at the pureScale configuration
details with the command,

DB2INSTANCE –LIST

ID  TYPE    STATE    HOME_HOST  CURRENT_HOST  PARTITION_NUMBER  LOGICAL_PORT NETNAME
--  ----    -----    ---------  ------------  ----------------  ------------ -------
0   MEMBER  STARTED  eg09ph12   eg09ph12        0                0         eg09ph12-ib0
1   MEMBER  STARTED  eg09ph13   eg09ph13        0                0         eg09ph13-ib0
2   MEMBER  STARTED  eg09ph14   eg09ph14        0                0         eg09ph14-ib0
128 CF      PRIMARY  eg09ph10   eg09ph10        -                0         eg09ph10-ib0
129 CF      PEER     eg09ph11   eg09ph11        -                0         eg09ph11-ib0

HOSTNAME      STATE           INSTANCE_STOPPED
--------     ------          --------------------
eg09ph11      ACTIVE          NO
eg09ph10      ACTIVE          NO
eg09ph14      ACTIVE          NO
eg09ph13      ACTIVE          NO
eg09ph12      ACTIVE          NO
```

DB2 9.7 high availability configuration

The PowerHA is set up for failover of the databases. The configuration is similar to the configuration for WebSphere MQ as explained in 6.2.4, “WebSphere MQ Server” on page 61.

Install DB2 9.7 as explained in *Installing DB2 Servers*, GC27-2455-02. The following sections highlight the configuration details of the test environment installation.

db2set

Example 6-11 shows the configuration for **db2set**.

Example 6-11 Configuration for db2set

```
[i] DB2COMM=TCPIP
[g] DB2FCMCOMM=TCPIP4
[g] DB2SYSTEM=eg09ph04
[g] DB2INSTDEF=db2inst1
```

System database directory

Example 6-12 shows a system database directory file that exists for each instance of the database manager. This file contains one entry for each database that was cataloged for this instance.

Example 6-12 System database directory configuration

```
Number of entries in the directory = 1

Database 1 entry:

 Database alias                       = R11_HADR
 Database name                        = R11_HADR
 Local database directory             = /db2
 Database release level               = d.00
 Comment                              =
 Directory entry type                 = Indirect
 Catalog database partition number    = 0
 Alternate server hostname            =
 Alternate server port number         =
```

DB2 9.7 disaster recovery configuration

Install the DB2 9.7 stand-alone version and the configuration details as shown in the following sections. The HADR option is activated between the disaster recovery sites.

db2set

Example 6-13 shows the configuration for db2set.

Example 6-13 Configuration for db2set

```
[i] DB2COMM=tcpip
[i] DB2AUTOSTART=YES
[g] DB2FCMCOMM=TCPIP4
[g] DB2SYSTEM=eg09ph16
[g] DB2INSTDEF=db2inst1
[g] DB2ADMINSERVER=dasusr1
```

db2pd

Example 6-14 shows the configuration for **db2pd**.

Example 6-14 Configuration for db2pd

```
Database Partition 0 -- Database R11_HADR -- Active -- Up 3 days 04:16:24 --
Date 10/03/2011 16:09:11

HADR Information:
Role    State                SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Primary Disconnected         Sync     0                  0

ConnectStatus ConnectTime                           Timeout
Disconnected  Fri Sep 30 12:23:38 2011 (1317381818) 120

PeerWindowEnd                         PeerWindow
Null (0)                              120

LocalHost                                LocalService
eg09ph16                                 65000

RemoteHost                               RemoteService      RemoteInstance
eg09ph15                                 65000              db2inst1

PrimaryFile  PrimaryPg  PrimaryLSN
S0000189.LOG 3617       0x00000003CA9494AC

StandByFile  StandByPg  StandByLSN
S0000000.LOG 0          0x0000000000000000
```

System database directory

Example 6-15 shows a system database directory file for each instance of the database manager. This file has one entry for each database that was cataloged for this instance.

Example 6-15 System database directory configuration

```
Number of entries in the directory = 2

Database 1 entry:

 Database alias                       = R11_HADR
 Database name                        = R11_HADR
 Local database directory             = /db2
 Database release level               = d.00
 Comment                              =
 Directory entry type                 = Indirect
 Catalog database partition number    = 0
 Alternate server hostname            = eg09ph15
 Alternate server port number         = 50000

Database 2 entry:

 Database alias                       = SAMPLE
 Database name                        = SAMPLE
 Local database directory             = /home/db2inst1
 Database release level               = d.00
 Comment                              =
 Directory entry type                 = Indirect
 Catalog database partition number    = 0
 Alternate server hostname            = eg09ph16
 Alternate server port number         = 50000
```

7

Testing processes and results

This chapter describes the test processes and results. In each test, a T24 test workload was run, which typically runs for a few minutes. A failover initiation was simulated during the run. The test was considered successful if the system recovered correctly from the failure.

The following workloads were used:

- *Read-only workload*. This workload consists of the inquiry transactions JMeter script to read the currency table that uses 10 parallel users. Also the browser was used to read the currency table by using the **`ENQ CURRENCY-LIST`** option.
- *Write workload*. This workload consists of transactions that update the database JMeter script to write customers by using 10 parallel users. Each user inserted 1,000 customers after the login parallel.
- A *batch workload*: The close-of-business (COB) process was run by using the batch job details that are stored in the T24 database.

This chapter contains the following sections:

- Creating a failover initiation mechanism
- Test results

7.1 Creating a failover initiation mechanism

We performed the following actions to create a failover initiation mechanism:

1. To achieve the sudden failover initiation of the WebSphere Application Server instances, stop these instances by using the terminate option in the WebSphere Application Server administration console.
2. Bring down the network adapter of the partition on which active WebSphere MQ is located to fail WebSphere MQ by using the following command:

   ```
   ifconfig ib0 inet down
   ```

   After the test, the storage administrator made the network adapter available to proceed with the subsequent tests.
3. On DB2, perform the following steps:

   a. After network adapter failover initiation, stop the DB2 pureScale member by using the following command:

   ```
   ifconfig ib0 inet down
   ```

   To make the member available, use the following command:

   ```
   ifconfig ib0 inet up
   ```

   b. For DB2 9.7 high availability, complete the following actions for the software and hardware failover initiation methods:

   - For the software failover initiation method, stop DB2 by using the **db2kill** command. The PowerHA server restarts the DB2 after verification that the current DB2 is not running in the same partition.
   - For the hardware failover initiation method, use the network failure method. Use this method when the current partition becomes unavailable and the PowerHA server switches from a failed server to a passive DB2, making it an active server.

   c. For DB2 9.7 disaster recovery, perform both the software and hardware failover initiation methods.
4. For T24 failover initiation methods:

   a. Set up the COB process to run as a multiserver with a halt before and after the interest capitalization COB (IC COB) job.

   b. Gather the base data to compare it with the failover tests. Due to the entry key generation that was used for net bulking, a varying number of entries in the STMT.ENTRY and CATEG.ENTRY tables is possible because the agent timing differs over the runs. Therefore, we use the RE.CONSOL.SPEC.ENTRY table to determine that all expected entries were processed on the failover.

Table 7-1 shows a comparison of the base data.

Table 7-1 T24 base data comparison

COB run	After restore	After first halt	After second halt
COUNT FBNK.STMT.ENTRY	113178	120603	121786
COUNT FBNK.CATEG.ENTRY	51460	62849	64174
COUNT FBNK.RE.CONSOL.SPEC.ENTRY	122101	129579	132360

c. Run the COB process in debug mode:

```
START.TSM -DEBUG
```

d. Start the individual COB agents (T24 Service Agent).

e. Stop the T24 Service Manager.

f. Stop the T24 Service Agent.

g. For T24 online failover initiation method, stop and restart jbase_agent.

7.2 Test results

This section highlights the test results for the tests that were performed.

7.2.1 High availability test with DB2 pureScale: Active-active mode

This section describes the test results for an active-active mode by using DB2 pureScale.

Read test

This section describes the results of the read tests. The read tests were run as explained in 5.1.1, “Read tests” on page 45.

Test 1: Investigating the WebSphere node failover during the read query

The WebSphere Application Server node was stopped by the process that is explained in 7.1, “Creating a failover initiation mechanism” on page 78.

The read test went through successfully. Upon failure of the WebSphere Application Server, automatic restart of the other WebSphere Application Server node occurred. In the JMeter script, the reads proceeded successfully during the failover process.

Table 7-2 shows the test result.

Table 7-2 JMeter summary report for read test 1

Transaction name	Number of records error	Error%	Average bytes
Login	10	0.00%	5701
Currency inquiry	287972	0.00%	28502
Total	287972	0.00%	28494

Test 2: Investigating MQ node failover during the read query

The MQ node was stopped by the process explained in 7.1, "Creating a failover initiation mechanism" on page 78. The failover of MQ to the passive node took approximately a few seconds. The maximum amount of the time was spent in the movement of storage from one partition to the other. During this time, few of the read transactions failed in JMeter. Upon completion of the MQ failover mechanism, the read query progressed successfully.

Table 7-3 shows the test result.

Table 7-3 JMeter summary report for read test 2

Transaction name	Number of records	Error %	Average bytes
Login	10	0.00%	5701
Currency inquiry	56155	0.20%	28459
Total	56165	0.20%	28455

Test 3: Investigating the T24 node failover during the read query

The T24 was stopped by the process that is explained in 7.1, "Creating a failover initiation mechanism" on page 78. The browser was run with the currency table read. T24 was stopped when the read was in progress, and the test completed successfully.

Test 4: Investigating the DB2 pureScale elements failover during the read query

The DB2 pureScale was stopped by the process that is explained in 7.1, "Creating a failover initiation mechanism" on page 78. During the database disconnect, the current active agent of T24 was terminated by the driver, and a new agent was initiated for the new workload. The read progressed with minimal failures.

Table 7-4 shows the test result.

Table 7-4 JMeter summary report for read test 4

Transaction name	Number of records	Error %	Average bytes
Login	20	0.00%	6158
Currency inquiry	20000	0.02%	28498
Logoff	10	0.00%	6325
Total	20030	0.01%	28465

Write test

This section describes the results of the write tests. The write test was performed as explained in 5.1.2, “Write tests” on page 47.

Test 1: Investigating the WebSphere node failover during the write query

The WebSphere Application Server was stopped by using the method that is explained in 7.1, “Creating a failover initiation mechanism” on page 78. After the WebSphere Application Server was bought down, the passive WebSphere Application Server node started instantaneously, which enabled smooth continuation of the write test.

Test 2: Investigating the MQ node failover during the write query

The MQ failover was performed by using the method that is explained in 7.1, “Creating a failover initiation mechanism” on page 78. A few of the transactions failed during MQ failover time. The storage copy from one server to the other server took some time. After this switch, the testing progressed successfully.

Table 7-5 shows the test result.

Table 7-5 JMeter summary report for write test 2

Transaction name	Number of records	Error %	Average bytes
Login	35	2.86%	5578
Customer commit	40155	0.70%	29946
Logoff	24	4.17%	6073
Total	40214	0.70%	29911

Test 3: Investigating the T24 node failover during the write query

The T24 failover was performed by using the method that is explained in 7.1, “Creating a failover initiation mechanism” on page 78. During the T24 failure and restart, the transactions processed successfully with few failures.

Table 7-6 shows the test result.

Table 7-6 JMeter summary report for write test 3

Transaction name	Number of records	Error %	Average bytes
Login	10	0.00%	5701
Customer commit	10000	0.10%	29098
Logoff	10	0.00%	6277
Total	40214	0.10%	29051

Tests 4 - 7: Investigating the DB2 pureScale member failover, member addition, and elements failover during the write query

The DB2 member failover was performed by using the method that is explained in 7.1, “Creating a failover initiation mechanism” on page 78. Tests 4 - 7 were performed in sequence. Upon failure of a member, the database driver terminated the agent that currently had an active connection to the database. Upon automatic remapping of the new DB2 pureScale member, the newly created agent ran successfully with minimal failure of the transaction.

COB test

This section describes the results of the COB tests. The COB test was run as explained in the 5.1.3, “COB tests” on page 49.

Tests 1 - 4: Investigating the DB2 pureScale member failover, member addition, and elements failover during the IC COB process

The DB2 failover was performed as explained in 7.1, “Creating a failover initiation mechanism” on page 78. Tests 1 - 4 were performed in a sequence during the COB process. When the DB2 pureScale member or cluster caching facility (CCF) went down, the XML driver terminated the active agent that was connected to the database. A new agent that was initiated continued with the COB process. The COB process ran successfully.

The success of the COB process was verified by using the method that is explained in 7.1, “Creating a failover initiation mechanism” on page 78. For the resulting record counts for this test, see “Results of the tests” on page 109.

Test 5: Investigating the T24 node failover during the IC COB process

The T24 failover was performed as explained in 7.1, “Creating a failover initiation mechanism” on page 78. This same section explains the method that was used to verify the success of the COB process. For the resulting record counts for this test, see “Results of the tests” on page 109.

Test 6: Investigating the T24 node failover during the IC COB process by using T24 Service Manager termination

The T24 failover was performed as explained in 7.1, “Creating a failover initiation mechanism” on page 78. This same section explains the method that was used to verify the success of the COB process. When the T24 Service Manager was stopped, the agents continued running. When the agents stopped, we restarted the T24 Service Manager and then restarted the agents. The COB ran successfully. For the resulting record counts for this test, see “Results of the tests” on page 109.

7.2.2 High availability test with DB2 9.7: Active-passive mode

Because the connectivity layer failover and application layer failover were already tested in 7.2.1, “High availability test with DB2 pureScale: Active-active mode” on page 79, they are not included again in this set of tests. This testing focuses solely on failover of the database DB2 9.7 for active-passive mode.

Read test

The read test was performed as explained in 5.2.1, “Read test” on page 52. The database was stopped by using the method that is explained in 7.1, “Creating a failover initiation mechanism” on page 78. The browser was used to verify the reads. A wait of a few seconds occurred when DB2 was remapped by PowerHA before the testing continued.

Write test

The write test was performed as explained in 5.2.2, “Write test” on page 52. The database was stopped by using the method that is explained in 7.1, “Creating a failover initiation mechanism” on page 781. A wait of a few seconds occurred when DB2 was remapped by PowerHA. During this wait time, few of the write transactions failed. The write went through successfully after remapping the database.

Table 7-7 shows the test result.

Table 7-7 JMeter summary report for the write test

Transaction name	Number of records	Error %	Average bytes
Login	10	0.00%	5701
Currency enquiry	10000	0.70%	28349
Logoff	10	0.00%	6325
Total	40214	0.70%	28305

COB test

The write test was performed as explained in 5.2.3, “COB test” on page 52. The database was stopped by using the method that is explained in 7.1, “Creating a failover initiation mechanism” on page 78. Ending both the software and hardware were tried, and in both scenarios the COB process completed successfully. For the resulting record counts for this test, see “Results of the tests” on page 109.

7.2.3 Disaster recovery test with DB2 9.7: Active-active mode

The high availability disaster recovery (HADR) function of the database was used to perform this test. Initially, the standby site was set up as a peer site. Upon failure, we did a manual takeover of the disaster recovery site by using the **`db2 takeover`** command. The failover situation is simulated by using the method that is described in 7.1, “Creating a failover initiation mechanism” on page 78. For details about the DB2 command and the DB2 status, see “Results of the tests” on page 109. After the takeover, the secondary site became the primary site.

Read test

The read test was performed by using the method that is explained in 5.3.1, “Read test” on page 54. Because the takeover was manual and took a few seconds, the test completed successfully with few failures.

Table 7-8 shows the test results.

Table 7-8 JMeter summary report for read test

Transaction name	Number of records	Error %	Average bytes
Login	30	0.00%	5701
Currency enquiry	30000	0.53%	28385

Transaction name	Number of records	Error %	Average bytes
Logoff	30	0.00%	6325
Total	30000	0.53%	28341

Write test

The write test was performed by using the method that is explained in 5.3.2, “Write test” on page 54. Because the takeover was manual and took a few seconds, the test completed successfully with few failures.

Table 7-9 shows the test results.

Table 7-9 JMeter summary report for the write test

Transaction name	Number of records	Error %	Average bytes
Login	30	0.00%	5701
Currency inquiry	30000	0.33%	28923
Logoff	30	0.00%	6277
Total	30000	0.33%	28877

COB test

The COB test was performed by using the method that is explained in 5.3.3, “COB test” on page 54. The test completed successfully. For the resulting record counts for this test, see “Results of the tests” on page 109.

7.3 Summary

To summarize, the high availability configurations and testing for T24 with DB2 pureScale and for T24 with DB2 9.7 were successful. Also, successful was the disaster recovery configuration and testing with DB2 9.7. These tests verified the recommended configurations for the high availability and disaster recovery solution for T24 with the IBM platform. These tests proved that the high availability solution of T24 with the high availability features from DB2, WebSphere Application Server, WebSphere MQ, and AIX are functional.

A

Configuration and testing details

This appendix explains various situations and their resolutions that occurred during the testing and the database configurations.

Issues and resolutions

This section explains in detail the key issues faced during the testing and the resolutions that were provided.

Situation 1

During the initial setup, the DB2 client did not recognize the DB2 pureScale member.

To resolve this issue, the `db2dsdriver.cfg` configuration file was changed as shown in Example A-1 and as indicated in the DB2 pureScale 9.8 client configuration.

Example A-1 Changed db2dsdriver.cfg file

```
<databases>
    <database name="R11_HADR" host="eg09ph12" port="50000">
        <parameter name="keepAliveTimeout" value="20"/>
        <parameter name="connectionLevelLoadBalancing" value="true" />
        <WLB>
           <parameter name="enableWLB" value="true"/>
           <parameter name="maxTransports" value="50"/>
           <parameter name="maxTransportIdleTime" value="600"/>
           <parameter name="maxTransportWaitTime" value="-1"/>
           <parameter name="maxRefreshInterval" value="30"/>
        </WLB>
        <ACR>
           <parameter name="enableACR" value="true"/>
           <parameter name="enableSeamlessACR" value="true"/>
           <parameter name="maxAcrRetries" value="2"/>
           <parameter name="acrRetryInterval" value="5"/>
           <parameter name="enableAlternateServerListFirstConnect"
value="true"/>
           <alternateserverlist>
             <server name="R11_HADR" hostname="eg09ph13" port="50000">
                </server>
             <server name="R11_HADR" hostname="eg09ph14" port="50000">
                </server>
           </alternateserverlist>
        </ACR>
      </database>
    </databases>
</configuration>
```

The node directory was changed to have a server name instead of an IP address, as shown in Example A-2.

Example A-2 Changed node directory

```
[eg09ph06:db2inst1] /cfg $ db2 list node directory

 Node Directory

 Number of entries in the directory = 1

Node 1 entry:

 Node name                      = R11_HADR
 Comment                        =
 Directory entry type           = LOCAL
 Protocol                       = TCPIP
 Hostname                       = eg09ph12
 Service name                   = 50000
```

Situation 2

The T24 database configuration failed to connect to the database. This issue resulted because the new `JEDI` file was not created. To correct the problem, the non-IP based `JEDI` file was added to the TAFC.

Situation 3

WebSphere Application Server failover indicated a problem with the IBM HTTP Servers plug-in and the way it was load balancing and detecting failover. When an WebSphere Application Server node failed and returned to the cluster, the load was not rebalanced across both active nodes. Also, when the remaining active node failed, the IBM HTTP Server assumed that the entire cluster failed and no further online requests were possible. This behavior was unexpected. The IBM HTTP Server was not detecting the first WebSphere Application Server failure and returned to the cluster cycle.

To correct the problem, first, a PMR 30222, 7TD, 000 was raised. Then the following fixes were applied, in the order shown, to help the downed node start:

1. Remove Fix Pack 17.
2. Install Fix Pack 15.
3. Install Fix Pack PM30968.

These fixes applied to the plug-in only. The rest of the environment stayed at Fix Pack 17.

In addition, the following TCP tuning parameters were applied to AIX, which are recommended for performance purposes where many connections are in use:

```
tcp_timewait =1
tcp_keepidle=600
tcp_keepintvl=10
tcp_keepinit=40
```

You an learn more about these parameters in the “Tuning AIX systems” topic in the WebSphere Application Server, Network Deployment, Version 7.0 Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tprf_tuneaix.html

Situation 4

A multiserver close of business (COB) test failed with fatal errors due to locking concerns. To correct this problem, jBASE distributed locking was implemented by using the jBASE distributed locking guide. To access a copy of this document, see “Online resources” on page 125.

Database manager and database configuration

This section shows the DB2 database manager configuration and the database configuration used in the high availability and disaster recovery tests.

DB2 pureScale high availability

This section highlights the DB2 database manager configuration and the database configuration in the DB2 pureScale high availability test.

Database manager configuration

Example A-3 shows the detailed database manager configuration for the DB2 pureScale high availability test.

Example A-3 Database manager configuration for DB2 pureScale high availability

```
Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0e00

 CPU speed (millisec/instruction)             (CPUSPEED) = 3.778754e-07
 Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) = 1.000000e+02
 Max number of concurrently active databases     (NUMDB) = 32
 Federated Database System Support           (FEDERATED) = NO
 Transaction processor monitor name        (TP_MON_NAME) =

 Default charge-back account           (DFT_ACCOUNT_STR) =

 Java Development Kit installation path       (JDK_PATH) =
/home/db2sdin1/sqllib/java/jdk64

 Diagnostic error capture level              (DIAGLEVEL) = 3
 Notify Level                              (NOTIFYLEVEL) = 3
 Diagnostic data directory path               (DIAGPATH) =
/home/db2sdin1/sqllib/db2dump
 Size of rotating db2diag & notify logs (MB)  (DIAGSIZE) = 0

 Default database monitor switches
   Buffer pool                         (DFT_MON_BUFPOOL) = ON
   Lock                                   (DFT_MON_LOCK) = ON
   Sort                                   (DFT_MON_SORT) = ON
   Statement                              (DFT_MON_STMT) = ON
   Table                                 (DFT_MON_TABLE) = ON
   Timestamp                         (DFT_MON_TIMESTAMP) = ON
   Unit of work                            (DFT_MON_UOW) = ON
 Monitor health of instance and databases   (HEALTH_MON) = OFF

 SYSADM group name                        (SYSADM_GROUP) = DB2IADM1
 SYSCTRL group name                      (SYSCTRL_GROUP) =
 SYSMAINT group name                    (SYSMAINT_GROUP) =
 SYSMON group name                        (SYSMON_GROUP) =

 Client Userid-Password Plugin          (CLNT_PW_PLUGIN) =
 Client Kerberos Plugin                (CLNT_KRB_PLUGIN) =
 Group Plugin                             (GROUP_PLUGIN) =
 GSS Plugin for Local Authorization    (LOCAL_GSSPLUGIN) =
 Server Plugin Mode                    (SRV_PLUGIN_MODE) = UNFENCED
 Server List of GSS Plugins      (SRVCON_GSSPLUGIN_LIST) =
 Server Userid-Password Plugin        (SRVCON_PW_PLUGIN) =
 Server Connection Authentication          (SRVCON_AUTH) = NOT_SPECIFIED
 Cluster manager                                         = TSA

 Database manager authentication        (AUTHENTICATION) = SERVER
 Alternate authentication           (ALTERNATE_AUTH_ENC) = NOT_SPECIFIED
 Cataloging allowed without authority   (CATALOG_NOAUTH) = NO
 Trust all clients                      (TRUST_ALLCLNTS) = YES
 Trusted client authentication          (TRUST_CLNTAUTH) = CLIENT
 Bypass federated authentication            (FED_NOAUTH) = NO
 Default database path                       (DFTDBPATH) =
/db2sd_20110823113125/db2sdin1

 Database monitor heap size (4KB)          (MON_HEAP_SZ) = AUTOMATIC(90)
 Java Virtual Machine heap size (4KB)     (JAVA_HEAP_SZ) = 2048
 Audit buffer size (4KB)                  (AUDIT_BUF_SZ) = 0
 Size of instance shared memory (4KB)  (INSTANCE_MEMORY) = AUTOMATIC(7391029)
 Instance memory for restart light (%) (RSTRT_LIGHT_MEM) = AUTOMATIC(7)
 Backup buffer default size (4KB)            (BACKBUFSZ) = 1024
 Restore buffer default size (4KB)           (RESTBUFSZ) = 1024

 Agent stack size                       (AGENT_STACK_SZ) = 1024
 Sort heap threshold (4KB)                  (SHEAPTHRES) = 0

 Directory cache support                     (DIR_CACHE) = YES

 Application support layer heap size (4KB)   (ASLHEAPSZ) = 15
 Max requester I/O block size (bytes)         (RQRIOBLK) = 32767
 Query heap size (4KB)                   (QUERY_HEAP_SZ) = 1000

 Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

 Priority of agents                           (AGENTPRI) = SYSTEM
 Agent pool size                        (NUM_POOLAGENTS) = AUTOMATIC(100)
 Initial number of agents in pool       (NUM_INITAGENTS) = 0
 Max number of coordinating agents     (MAX_COORDAGENTS) = AUTOMATIC(200)
 Max number of client connections      (MAX_CONNECTIONS) =
AUTOMATIC(MAX_COORDAGENTS)

 Keep fenced process                        (KEEPFENCED) = YES
 Number of pooled fenced processes         (FENCED_POOL) =
AUTOMATIC(MAX_COORDAGENTS)
 Initial number of fenced processes     (NUM_INITFENCED) = 0

 Index re-creation time and redo index build  (INDEXREC) = RESTART

 Transaction manager database name         (TM_DATABASE) = 1ST_CONN
 Transaction resync interval (sec)     (RESYNC_INTERVAL) = 180

 SPM name                                     (SPM_NAME) = eg09ph12
 SPM log size                          (SPM_LOG_FILE_SZ) = 256
 SPM resync agent limit                 (SPM_MAX_RESYNC) = 20
 SPM log path                             (SPM_LOG_PATH) =

 TCP/IP Service name                          (SVCENAME) = db2c_db2sdin1
 Discovery mode                               (DISCOVER) = SEARCH
 Discover server instance                (DISCOVER_INST) = ENABLE

 SSL server keydb file                   (SSL_SVR_KEYDB) =
 SSL server stash file                   (SSL_SVR_STASH) =
 SSL server certificate label            (SSL_SVR_LABEL) =
 SSL service name                         (SSL_SVCENAME) =
 SSL cipher specs                       (SSL_CIPHERSPECS) =
 SSL versions                             (SSL_VERSIONS) =
 SSL client keydb file                   (SSL_CLNT_KEYDB) =
 SSL client stash file                   (SSL_CLNT_STASH) =

 Maximum query degree of parallelism   (MAX_QUERYDEGREE) = ANY
 Enable intra-partition parallelism     (INTRA_PARALLEL) = NO

 Maximum Asynchronous TQs per query    (FEDERATED_ASYNC) = 0

 No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = AUTOMATIC(4096)
 No. of int. communication channels   (FCM_NUM_CHANNELS) = AUTOMATIC(2048)
 Node connection elapse time (sec)         (CONN_ELAPSE) = 10
 Max number of node connection retries (MAX_CONNRETRIES) = 5
 Max time difference between nodes (min) (MAX_TIME_DIFF) = 1

 db2start/db2stop timeout (min)        (START_STOP_TIME) = 10

 CF Server Configuration:
   Memory size (4KB)                          (CF_MEM_SZ) = AUTOMATIC(4787968)
   Number of worker threads              (CF_NUM_WORKERS) = AUTOMATIC(7)
   Number of connections                   (CF_NUM_CONNS) = AUTOMATIC(16)
   Diagnostic error capture level          (CF_DIAGLEVEL) = 2
   Diagnostic data directory path           (CF_DIAGPATH) =
```

Database configuration

Example A-4 shows the database configuration parameters for the DB2 pureScale high availability test.

Example A-4 Database configuration parameters for DB2 pureScale high availability

```
Database configuration release level                    = 0x0e00
 Database release level                                  = 0x0e00

 Database territory                                      = US
 Database code page                                      = 1208
 Database code set                                       = UTF-8
 Database country/region code                            = 1
 Database collating sequence                             = IDENTITY
 Alternate collating sequence              (ALT_COLLATE) =
 Number compatibility                                    = OFF
 Varchar2 compatibility                                  = OFF
 Date compatibility                                      = OFF
 Database page size                                      = 4096
```

```
Dynamic SQL Query management           (DYN_QUERY_MGMT) = DISABLE

Statement concentrator                      (STMT_CONC) = OFF

Discovery support for this database       (DISCOVER_DB) = ENABLE

Restrict access                                         = NO
Default query optimization class         (DFT_QUERYOPT) = 5
Degree of parallelism                      (DFT_DEGREE) = 1
Continue upon arithmetic exceptions   (DFT_SQLMATHWARN) = NO
Default refresh age                   (DFT_REFRESH_AGE) = 0
Default maintained table types for opt (DFT_MTTB_TYPES) = SYSTEM
Number of frequent values retained     (NUM_FREQVALUES) = 10
Number of quantiles retained            (NUM_QUANTILES) = 20

Decimal floating point rounding mode  (DECFLT_ROUNDING) = ROUND_HALF_EVEN

Backup pending                                          = NO

All committed transactions have been written to disk    = NO
Rollforward pending                                     = NO
Restore pending                                         = NO

Multi-page file allocation enabled                      = YES

Log retain for recovery status                          = NO
User exit for logging status                            = NO

Self tuning memory                    (SELF_TUNING_MEM) = ON
Size of database shared memory (4KB)  (DATABASE_MEMORY) = AUTOMATIC(1560832)
Database memory threshold               (DB_MEM_THRESH) = 10
Max storage for lock list (4KB)              (LOCKLIST) = 6200
Percent. of lock lists per application       (MAXLOCKS) = 60
Package cache size (4KB)                   (PCKCACHESZ) = AUTOMATIC(65514)
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = AUTOMATIC(7090)
Sort list heap (4KB)                         (SORTHEAP) = AUTOMATIC(1418)

Database heap (4KB)                            (DBHEAP) = AUTOMATIC(3637)
Catalog cache size (4KB)              (CATALOGCACHE_SZ) = 300
Log buffer size (4KB)                        (LOGBUFSZ) = 256
Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 430653
Buffer pool size (pages)                     (BUFFPAGE) = 1000
SQL statement heap (4KB)                     (STMTHEAP) = 4096
Default application heap (4KB)             (APPLHEAPSZ) = AUTOMATIC(256)
Application Memory Size (4KB)             (APPL_MEMORY) = AUTOMATIC(40000)
Statistics heap size (4KB)               (STAT_HEAP_SZ) = AUTOMATIC(4384)

Interval for checking deadlock (ms)         (DLCHKTIME) = 10000
```

```
 Lock timeout (sec)                        (LOCKTIMEOUT) = -1

 Changed pages threshold                (CHNGPGS_THRESH) = 80
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = AUTOMATIC(127)
 Number of I/O servers                   (NUM_IOSERVERS) = AUTOMATIC(6)
 Index sort flag                             (INDEXSORT) = YES
 Sequential detect flag                      (SEQDETECT) = YES
 Default prefetch size (pages)         (DFT_PREFETCH_SZ) = AUTOMATIC

 Track modified pages                         (TRACKMOD) = NO

 Default number of containers                            = 1
 Default tablespace extentsize (pages)   (DFT_EXTENT_SZ) = 32

 Max number of active applications            (MAXAPPLS) = AUTOMATIC(254)
 Average number of active applications       (AVG_APPLS) = 1
 Max DB files open per application            (MAXFILOP) = 61440

 Log file size (4KB)                         (LOGFILSIZ) = 10240
 Number of primary log files                (LOGPRIMARY) = 20
 Number of secondary log files               (LOGSECOND) = 20
 Changed path to log files                  (NEWLOGPATH) =
 Path to log files                                       =
/db2sd_20110823113125/db2sdin1/db2sdin1/R11_HADR/DBPARTITION0000/LOGSTREAM0000/
 Overflow log path                     (OVERFLOWLOGPATH) =
 Mirror log path                         (MIRRORLOGPATH) =
 First active log file                                   =
 Block log on disk full                (BLK_LOG_DSK_FUL) = NO
 Block non logged operations            (BLOCKNONLOGGED) = NO
 Percent max primary log space by transaction  (MAX_LOG) = 0
 Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0

 Group commit count                          (MINCOMMIT) = 1
 Percent log file reclaimed before soft chckpt (SOFTMAX) = 520
 Log retain for recovery enabled             (LOGRETAIN) = OFF
 User exit for logging enabled                (USEREXIT) = OFF

 HADR database role                                      = STANDARD
 HADR local host name                  (HADR_LOCAL_HOST) =
 HADR local service name                (HADR_LOCAL_SVC) =
 HADR remote host name                (HADR_REMOTE_HOST) =
 HADR remote service name              (HADR_REMOTE_SVC) =
 HADR instance name of remote server  (HADR_REMOTE_INST) =
 HADR timeout value                       (HADR_TIMEOUT) = 120
 HADR log write synchronization mode     (HADR_SYNCMODE) = NEARSYNC
 HADR peer window duration (seconds)  (HADR_PEER_WINDOW) = 0

 First log archive method                 (LOGARCHMETH1) = OFF
 Options for logarchmeth1                  (LOGARCHOPT1) =
 Second log archive method               (LOGARCHMETH2) = OFF
 Options for logarchmeth2                 (LOGARCHOPT2) =
 Failover log archive path               (FAILARCHPATH) =
 Number of log archive retries on error   (NUMARCHRETRY) = 5
 Log archive retry Delay (secs)         (ARCHRETRYDELAY) = 20
 Vendor options                             (VENDOROPT) =

 Auto restart enabled                     (AUTORESTART) = ON
 Index re-creation time and redo index build  (INDEXREC) = SYSTEM (RESTART)
 Log pages during index build           (LOGINDEXBUILD) = OFF
 Default number of loadrec sessions    (DFT_LOADREC_SES) = 1
 Number of database backups to retain   (NUM_DB_BACKUPS) = 12
 Recovery history retention (days)     (REC_HIS_RETENTN) = 366
 Auto deletion of recovery objects    (AUTO_DEL_REC_OBJ) = OFF

 TSM management class                   (TSM_MGMTCLASS) =
 TSM node name                           (TSM_NODENAME) =
 TSM owner                                  (TSM_OWNER) =
 TSM password                            (TSM_PASSWORD) =

 Automatic maintenance                     (AUTO_MAINT) = ON
   Automatic database backup            (AUTO_DB_BACKUP) = OFF
   Automatic table maintenance          (AUTO_TBL_MAINT) = ON
     Automatic runstats                  (AUTO_RUNSTATS) = ON
       Automatic statement statistics  (AUTO_STMT_STATS) = ON
     Automatic statistics profiling    (AUTO_STATS_PROF) = OFF
       Automatic profile updates         (AUTO_PROF_UPD) = OFF
     Automatic reorganization              (AUTO_REORG) = OFF

 Auto-Revalidation                         (AUTO_REVAL) = DEFERRED
 CF Resource Configuration:
     CF database memory size (4KB)       (CF_DB_MEM_SZ) = AUTOMATIC(4774656)
 Group buffer pool size (4KB)               (CF_GBP_SZ) = AUTOMATIC(3800064)
 Global lock memory size (4KB)             (CF_LOCK_SZ) = AUTOMATIC(716288)
 Shared communication area size (4KB)       (CF_SCA_SZ) = AUTOMATIC(256768)

 Catchup target for secondary CF (mins)(CF_CATCHUP_TRGT) = AUTOMATIC(15)

 Currently Committed                       (CUR_COMMIT) = ON
 CHAR output with DECIMAL input        (DEC_TO_CHAR_FMT) = NEW
 Enable XML Character operations        (ENABLE_XMLCHAR) = YES
 WLM Collection Interval (minutes)     (WLM_COLLECT_INT) = 0
 Monitor Collect Settings
 Request metrics                       (MON_REQ_METRICS) = BASE
 Activity metrics                      (MON_ACT_METRICS) = BASE
 Object metrics                        (MON_OBJ_METRICS) = BASE
 Unit of work events                      (MON_UOW_DATA) = NONE
 Lock timeout events                   (MON_LOCKTIMEOUT) = NONE
 Deadlock events                          (MON_DEADLOCK) = WITHOUT_HIST
```

```
 Lock wait events                           (MON_LOCKWAIT) = NONE
 Lock wait event threshold                 (MON_LW_THRESH) = 5000000
 Number of package list entries           (MON_PKGLIST_SZ) = 32
 Lock event notification level           (MON_LCK_MSG_LVL) = 1

 SMTP Server                                 (SMTP_SERVER) =
 SQL conditional compilation flags            (SQL_CCFLAGS) =
 Section actuals setting                 (SECTION_ACTUALS) = NONE

 Database is in write suspend state                        = NO
```

DB2 9.7 high availability

This section shows the DB2 database manager configuration and the database configuration in the DB2 9.7 high availability test.

Database manager configuration

Example A-5 shows the database manager configuration for the DB2 9.7 high availability test.

Example A-5 Database Manager configuration for DB2 9.7 high availability

```
Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0d00

 CPU speed (millisec/instruction)             (CPUSPEED) = 3.778754e-07
 Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) = 1.000000e+02

 Max number of concurrently active databases     (NUMDB) = 8
 Federated Database System Support           (FEDERATED) = NO
 Transaction processor monitor name        (TP_MON_NAME) =

 Default charge-back account           (DFT_ACCOUNT_STR) =

 Java Development Kit installation path       (JDK_PATH) =
/home/db2inst1/sqllib/java/jdk64

 Diagnostic error capture level              (DIAGLEVEL) = 3
 Notify Level                              (NOTIFYLEVEL) = 3
 Diagnostic data directory path               (DIAGPATH) =
/home/db2inst1/sqllib/db2dump/
 Alternate diagnostic data directory path (ALT_DIAGPATH) =
 Size of rotating db2diag & notify logs (MB)  (DIAGSIZE) = 0
```

```
Default database monitor switches
  Buffer pool                         (DFT_MON_BUFPOOL) = ON
  Lock                                   (DFT_MON_LOCK) = ON
  Sort                                   (DFT_MON_SORT) = ON
  Statement                              (DFT_MON_STMT) = ON
  Table                                 (DFT_MON_TABLE) = ON
  Timestamp                         (DFT_MON_TIMESTAMP) = ON
  Unit of work                            (DFT_MON_UOW) = ON
Monitor health of instance and databases   (HEALTH_MON) = ON

SYSADM group name                        (SYSADM_GROUP) = DB2GRP1
SYSCTRL group name                      (SYSCTRL_GROUP) =
SYSMAINT group name                    (SYSMAINT_GROUP) =
SYSMON group name                        (SYSMON_GROUP) =

Client Userid-Password Plugin          (CLNT_PW_PLUGIN) =
Client Kerberos Plugin                (CLNT_KRB_PLUGIN) =
Group Plugin                             (GROUP_PLUGIN) =
GSS Plugin for Local Authorization    (LOCAL_GSSPLUGIN) =
Server Plugin Mode                    (SRV_PLUGIN_MODE) = UNFENCED
Server List of GSS Plugins      (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin        (SRVCON_PW_PLUGIN) =
Server Connection Authentication          (SRVCON_AUTH) = NOT_SPECIFIED
Cluster manager                           (CLUSTER_MGR) =

Database manager authentication        (AUTHENTICATION) = SERVER
Alternate authentication           (ALTERNATE_AUTH_ENC) = NOT_SPECIFIED
Cataloging allowed without authority   (CATALOG_NOAUTH) = NO
Trust all clients                      (TRUST_ALLCLNTS) = YES
Trusted client authentication          (TRUST_CLNTAUTH) = CLIENT
Bypass federated authentication            (FED_NOAUTH) = NO

Default database path                       (DFTDBPATH) = /db2

Database monitor heap size (4KB)          (MON_HEAP_SZ) = AUTOMATIC(90)
Java Virtual Machine heap size (4KB)     (JAVA_HEAP_SZ) = 2048
Audit buffer size (4KB)                  (AUDIT_BUF_SZ) = 0
Size of instance shared memory (4KB)  (INSTANCE_MEMORY) = AUTOMATIC(7391029)
Agent stack size                       (AGENT_STACK_SZ) = 1024
Sort heap threshold (4KB)                  (SHEAPTHRES) = 0

Directory cache support                     (DIR_CACHE) = YES

Application support layer heap size (4KB)   (ASLHEAPSZ) = 15
Max requester I/O block size (bytes)         (RQRIOBLK) = 32767
Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

Priority of agents                           (AGENTPRI) = SYSTEM
Agent pool size                        (NUM_POOLAGENTS) = AUTOMATIC(100)
```

```
 Initial number of agents in pool       (NUM_INITAGENTS) = 0
 Max number of coordinating agents     (MAX_COORDAGENTS) = AUTOMATIC(200)
 Max number of client connections      (MAX_CONNECTIONS) =
AUTOMATIC(MAX_COORDAGENTS)

 Keep fenced process                        (KEEPFENCED) = YES
 Number of pooled fenced processes         (FENCED_POOL) =
AUTOMATIC(MAX_COORDAGENTS)
 Initial number of fenced processes     (NUM_INITFENCED) = 0

 Index re-creation time and redo index build  (INDEXREC) = RESTART

 Transaction manager database name         (TM_DATABASE) = 1ST_CONN
 Transaction resync interval (sec)     (RESYNC_INTERVAL) = 180

 SPM name                                     (SPM_NAME) = eg09ph04
 SPM log size                          (SPM_LOG_FILE_SZ) = 256
 SPM resync agent limit                 (SPM_MAX_RESYNC) = 20
 SPM log path                             (SPM_LOG_PATH) =

 TCP/IP Service name                          (SVCENAME) = db2c_db2inst1
 Discovery mode                               (DISCOVER) = SEARCH
 Discover server instance                (DISCOVER_INST) = ENABLE

 SSL server keydb file                   (SSL_SVR_KEYDB) =
 SSL server stash file                   (SSL_SVR_STASH) =
 SSL server certificate label            (SSL_SVR_LABEL) =
 SSL service name                         (SSL_SVCENAME) =
 SSL cipher specs                      (SSL_CIPHERSPECS) =
 SSL versions                             (SSL_VERSIONS) =
 SSL client keydb file                  (SSL_CLNT_KEYDB) =
 SSL client stash file                  (SSL_CLNT_STASH) =

 Maximum query degree of parallelism   (MAX_QUERYDEGREE) = ANY
 Enable intra-partition parallelism     (INTRA_PARALLEL) = NO

 Maximum Asynchronous TQs per query    (FEDERATED_ASYNC) = 0

 No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = AUTOMATIC(4096)
 No. of int. communication channels   (FCM_NUM_CHANNELS) = AUTOMATIC(2048)
 Node connection elapse time (sec)         (CONN_ELAPSE) = 10
 Max number of node connection retries (MAX_CONNRETRIES) = 5
 Max time difference between nodes (min) (MAX_TIME_DIFF) = 60

 db2start/db2stop timeout (min)        (START_STOP_TIME) = 10
```

Database configuration

Example A-6 shows the database configuration parameters for the DB2 9.7 high availability test.

Example A-6 Database configuration parameters for DB2 9.7 high availability

```
Database configuration release level                    = 0x0d00
 Database release level                                  = 0x0d00

 Database territory                                      = US
 Database code page                                      = 1208
 Database code set                                       = UTF-8
 Database country/region code                            = 1
 Database collating sequence                             = IDENTITY
 Alternate collating sequence              (ALT_COLLATE) =
 Number compatibility                                    = OFF
 Varchar2 compatibility                                  = OFF
 Date compatibility                                      = OFF
 Database page size                                      = 4096

 Dynamic SQL Query management           (DYN_QUERY_MGMT) = DISABLE

 Statement concentrator                      (STMT_CONC) = OFF

 Discovery support for this database       (DISCOVER_DB) = ENABLE

 Restrict access                                         = NO
 Default query optimization class         (DFT_QUERYOPT) = 5
 Degree of parallelism                      (DFT_DEGREE) = 1
 Continue upon arithmetic exceptions   (DFT_SQLMATHWARN) = NO
 Default refresh age                   (DFT_REFRESH_AGE) = 0
 Default maintained table types for opt (DFT_MTTB_TYPES) = SYSTEM
 Number of frequent values retained     (NUM_FREQVALUES) = 10
 Number of quantiles retained            (NUM_QUANTILES) = 20

 Decimal floating point rounding mode  (DECFLT_ROUNDING) = ROUND_HALF_EVEN

 Backup pending                                          = NO

 All committed transactions have been written to disk    = NO
 Rollforward pending                                     = NO
 Restore pending                                         = NO

 Multi-page file allocation enabled                      = YES

 Log retain for recovery status                          = NO
 User exit for logging status                            = NO

 Self tuning memory                    (SELF_TUNING_MEM) = ON
 Size of database shared memory (4KB)  (DATABASE_MEMORY) = AUTOMATIC(475056)
 Database memory threshold               (DB_MEM_THRESH) = 10
 Max storage for lock list (4KB)              (LOCKLIST) = 6200
 Percent. of lock lists per application       (MAXLOCKS) = 60
 Package cache size (4KB)                   (PCKCACHESZ) = AUTOMATIC(7566)
 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = AUTOMATIC(268)
 Sort list heap (4KB)                         (SORTHEAP) = AUTOMATIC(53)

 Database heap (4KB)                            (DBHEAP) = AUTOMATIC(3233)
 Catalog cache size (4KB)              (CATALOGCACHE_SZ) = 300
 Log buffer size (4KB)                        (LOGBUFSZ) = 256
 Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 286142
 Buffer pool size (pages)                     (BUFFPAGE) = 1000
 SQL statement heap (4KB)                     (STMTHEAP) = 4096
 Default application heap (4KB)             (APPLHEAPSZ) = AUTOMATIC(256)
 Application Memory Size (4KB)             (APPL_MEMORY) = AUTOMATIC(40000)
 Statistics heap size (4KB)               (STAT_HEAP_SZ) = AUTOMATIC(4384)

 Interval for checking deadlock (ms)         (DLCHKTIME) = 10000
 Lock timeout (sec)                        (LOCKTIMEOUT) = -1

 Changed pages threshold                (CHNGPGS_THRESH) = 80
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = AUTOMATIC(7)
 Number of I/O servers                   (NUM_IOSERVERS) = AUTOMATIC(3)
 Index sort flag                             (INDEXSORT) = YES
 Sequential detect flag                      (SEQDETECT) = YES
 Default prefetch size (pages)         (DFT_PREFETCH_SZ) = AUTOMATIC

 Track modified pages                         (TRACKMOD) = OFF

 Default number of containers                            = 1
 Default tablespace extentsize (pages)   (DFT_EXTENT_SZ) = 2

 Max number of active applications            (MAXAPPLS) = AUTOMATIC(40)
 Average number of active applications       (AVG_APPLS) = 1
 Max DB files open per application            (MAXFILOP) = 61440

 Log file size (4KB)                         (LOGFILSIZ) = 10240
 Number of primary log files                (LOGPRIMARY) = 20
 Number of secondary log files               (LOGSECOND) = 20
 Changed path to log files                  (NEWLOGPATH) =
 Path to log files                                       =
/db2log/r11_hadr/NODE0000/
 Overflow log path                     (OVERFLOWLOGPATH) =
 Mirror log path                         (MIRRORLOGPATH) =
 First active log file                                   =
 Block log on disk full                (BLK_LOG_DSK_FUL) = NO
 Block non logged operations            (BLOCKNONLOGGED) = NO
 Percent max primary log space by transaction  (MAX_LOG) = 0
```

```
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0

Group commit count                         (MINCOMMIT) = 1
Percent log file reclaimed before soft chckpt (SOFTMAX) = 520
Log retain for recovery enabled            (LOGRETAIN) = OFF
User exit for logging enabled               (USEREXIT) = OFF

HADR database role                                     = STANDARD
HADR local host name                 (HADR_LOCAL_HOST) =
HADR local service name               (HADR_LOCAL_SVC) =
HADR remote host name               (HADR_REMOTE_HOST) =
HADR remote service name             (HADR_REMOTE_SVC) =
HADR instance name of remote server (HADR_REMOTE_INST) =
HADR timeout value                      (HADR_TIMEOUT) = 120
HADR log write synchronization mode    (HADR_SYNCMODE) = NEARSYNC
HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0

First log archive method                (LOGARCHMETH1) = OFF
Options for logarchmeth1                 (LOGARCHOPT1) =
Second log archive method               (LOGARCHMETH2) = OFF
Options for logarchmeth2                 (LOGARCHOPT2) =
Failover log archive path               (FAILARCHPATH) =
Number of log archive retries on error  (NUMARCHRETRY) = 5
Log archive retry Delay (secs)        (ARCHRETRYDELAY) = 20
Vendor options                             (VENDOROPT) =

Auto restart enabled                     (AUTORESTART) = ON
Index re-creation time and redo index build  (INDEXREC) = SYSTEM (RESTART)
Log pages during index build           (LOGINDEXBUILD) = OFF
Default number of loadrec sessions    (DFT_LOADREC_SES) = 1
Number of database backups to retain   (NUM_DB_BACKUPS) = 12
Recovery history retention (days)     (REC_HIS_RETENTN) = 366
Auto deletion of recovery objects    (AUTO_DEL_REC_OBJ) = OFF

TSM management class                    (TSM_MGMTCLASS) =
TSM node name                            (TSM_NODENAME) =
TSM owner                                   (TSM_OWNER) =
TSM password                             (TSM_PASSWORD) =

Automatic maintenance                      (AUTO_MAINT) = ON
  Automatic database backup             (AUTO_DB_BACKUP) = OFF
  Automatic table maintenance           (AUTO_TBL_MAINT) = ON
    Automatic runstats                   (AUTO_RUNSTATS) = ON
      Automatic statement statistics   (AUTO_STMT_STATS) = ON
    Automatic statistics profiling     (AUTO_STATS_PROF) = OFF
      Automatic profile updates          (AUTO_PROF_UPD) = OFF
    Automatic reorganization                (AUTO_REORG) = OFF

Auto-Revalidation                          (AUTO_REVAL) = DEFERRED
```

```
 Currently Committed                          (CUR_COMMIT) = ON
 CHAR output with DECIMAL input          (DEC_TO_CHAR_FMT) = NEW
 Enable XML Character operations          (ENABLE_XMLCHAR) = YES
 WLM Collection Interval (minutes)       (WLM_COLLECT_INT) = 0
 Monitor Collect Settings
 Request metrics                         (MON_REQ_METRICS) = BASE
 Activity metrics                        (MON_ACT_METRICS) = BASE
 Object metrics                          (MON_OBJ_METRICS) = BASE
 Unit of work events                        (MON_UOW_DATA) = NONE
 Lock timeout events                     (MON_LOCKTIMEOUT) = NONE
 Deadlock events                            (MON_DEADLOCK) = WITHOUT_HIST
 Lock wait events                           (MON_LOCKWAIT) = NONE
 Lock wait event threshold                 (MON_LW_THRESH) = 5000000
 Number of package list entries           (MON_PKGLIST_SZ) = 32
 Lock event notification level           (MON_LCK_MSG_LVL) = 1

 SMTP Server                                 (SMTP_SERVER) =
 SQL conditional compilation flags            (SQL_CCFLAGS) =
 Section actuals setting                 (SECTION_ACTUALS) = NONE
 Connect procedure                          (CONNECT_PROC) =
```

DB2 9.7 disaster recovery

This section shows the DB2 database manager configuration and the database configuration in the DB2 9.7 disaster recovery test.

Database manager configuration

Example A-7 shows the database manager configuration for the DB2 9.7 disaster recovery test.

Example A-7 Database manager configuration for DB2 9.7 disaster recovery

```
Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0d00

 CPU speed (millisec/instruction)             (CPUSPEED) = 2.361721e-07
 Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) = 1.000000e+02

 Max number of concurrently active databases     (NUMDB) = 8
 Federated Database System Support           (FEDERATED) = NO
 Transaction processor monitor name        (TP_MON_NAME) =

 Default charge-back account           (DFT_ACCOUNT_STR) =
 Java Development Kit installation path       (JDK_PATH) =
/home/db2inst1/sqllib/java/jdk64

 Diagnostic error capture level              (DIAGLEVEL) = 3
 Notify Level                              (NOTIFYLEVEL) = 3
 Diagnostic data directory path               (DIAGPATH) =
/home/db2inst1/sqllib/db2dump/
 Alternate diagnostic data directory path (ALT_DIAGPATH) =
 Size of rotating db2diag & notify logs (MB)  (DIAGSIZE) = 0

 Default database monitor switches
   Buffer pool                         (DFT_MON_BUFPOOL) = OFF
   Lock                                   (DFT_MON_LOCK) = OFF
   Sort                                   (DFT_MON_SORT) = OFF
   Statement                              (DFT_MON_STMT) = OFF
   Table                                 (DFT_MON_TABLE) = OFF
   Timestamp                         (DFT_MON_TIMESTAMP) = ON
   Unit of work                            (DFT_MON_UOW) = OFF
 Monitor health of instance and databases   (HEALTH_MON) = ON

 SYSADM group name                        (SYSADM_GROUP) = DB2GRP1
 SYSCTRL group name                      (SYSCTRL_GROUP) =
 SYSMAINT group name                    (SYSMAINT_GROUP) =
 SYSMON group name                        (SYSMON_GROUP) =

 Client Userid-Password Plugin          (CLNT_PW_PLUGIN) =
 Client Kerberos Plugin                (CLNT_KRB_PLUGIN) =
 Group Plugin                             (GROUP_PLUGIN) =
 GSS Plugin for Local Authorization    (LOCAL_GSSPLUGIN) =
 Server Plugin Mode                    (SRV_PLUGIN_MODE) = UNFENCED
 Server List of GSS Plugins      (SRVCON_GSSPLUGIN_LIST) =
 Server Userid-Password Plugin        (SRVCON_PW_PLUGIN) =
 Server Connection Authentication          (SRVCON_AUTH) = NOT_SPECIFIED
 Cluster manager                           (CLUSTER_MGR) =

 Database manager authentication        (AUTHENTICATION) = SERVER
 Alternate authentication           (ALTERNATE_AUTH_ENC) = NOT_SPECIFIED
 Cataloging allowed without authority   (CATALOG_NOAUTH) = NO
 Trust all clients                      (TRUST_ALLCLNTS) = YES
 Trusted client authentication          (TRUST_CLNTAUTH) = CLIENT
 Bypass federated authentication            (FED_NOAUTH) = NO

 Default database path                       (DFTDBPATH) = /home/db2inst1

 Database monitor heap size (4KB)          (MON_HEAP_SZ) = AUTOMATIC(90)
 Java Virtual Machine heap size (4KB)     (JAVA_HEAP_SZ) = 2048
 Audit buffer size (4KB)                  (AUDIT_BUF_SZ) = 0
 Size of instance shared memory (4KB)  (INSTANCE_MEMORY) = AUTOMATIC(7391029)
 Agent stack size                       (AGENT_STACK_SZ) = 1024
 Sort heap threshold (4KB)                  (SHEAPTHRES) = 0

 Directory cache support                     (DIR_CACHE) = YES

 Application support layer heap size (4KB)   (ASLHEAPSZ) = 15
 Max requester I/O block size (bytes)         (RQRIOBLK) = 32767
 Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

 Priority of agents                           (AGENTPRI) = SYSTEM
 Agent pool size                        (NUM_POOLAGENTS) = AUTOMATIC(100)
 Initial number of agents in pool       (NUM_INITAGENTS) = 0
 Max number of coordinating agents     (MAX_COORDAGENTS) = AUTOMATIC(200)
 Max number of client connections     (MAX_CONNECTIONS) = 
AUTOMATIC(MAX_COORDAGENTS)

 Keep fenced process                        (KEEPFENCED) = YES
 Number of pooled fenced processes         (FENCED_POOL) = 
AUTOMATIC(MAX_COORDAGENTS)
 Initial number of fenced processes     (NUM_INITFENCED) = 0

 Index re-creation time and redo index build  (INDEXREC) = RESTART

 Transaction manager database name         (TM_DATABASE) = 1ST_CONN
 Transaction resync interval (sec)     (RESYNC_INTERVAL) = 180

 SPM name                                     (SPM_NAME) = eg09ph16
 SPM log size                          (SPM_LOG_FILE_SZ) = 256
 SPM resync agent limit                 (SPM_MAX_RESYNC) = 20
 SPM log path                             (SPM_LOG_PATH) = 

 TCP/IP Service name                          (SVCENAME) = 50000
 Discovery mode                               (DISCOVER) = SEARCH
 Discover server instance                (DISCOVER_INST) = ENABLE

 SSL server keydb file                   (SSL_SVR_KEYDB) = 
 SSL server stash file                   (SSL_SVR_STASH) = 
 SSL server certificate label            (SSL_SVR_LABEL) = 
 SSL service name                         (SSL_SVCENAME) = 
 SSL cipher specs                      (SSL_CIPHERSPECS) = 
 SSL versions                             (SSL_VERSIONS) = 
 SSL client keydb file                  (SSL_CLNT_KEYDB) = 
 SSL client stash file                  (SSL_CLNT_STASH) = 

 Maximum query degree of parallelism   (MAX_QUERYDEGREE) = ANY
 Enable intra-partition parallelism     (INTRA_PARALLEL) = NO

 Maximum Asynchronous TQs per query    (FEDERATED_ASYNC) = 0

 No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = AUTOMATIC(4096)
 No. of int. communication channels   (FCM_NUM_CHANNELS) = AUTOMATIC(2048)
 Node connection elapse time (sec)         (CONN_ELAPSE) = 10
 Max number of node connection retries (MAX_CONNRETRIES) = 5
 Max time difference between nodes (min) (MAX_TIME_DIFF) = 60

 db2start/db2stop timeout (min)        (START_STOP_TIME) = 10
```

Database configuration for database

Example A-8 shows the database configuration parameters for the DB2 9.7 disaster recovery test.

Example A-8 Database configuration parameters for DB2 9.7 disaster recovery

```
Database configuration release level                    = 0x0d00
 Database release level                                  = 0x0d00

 Database territory                                      = US
 Database code page                                      = 1208
 Database code set                                       = UTF-8
 Database country/region code                            = 1
 Database collating sequence                             = IDENTITY
 Alternate collating sequence              (ALT_COLLATE) =
 Number compatibility                                    = OFF
 Varchar2 compatibility                                  = OFF
 Date compatibility                                      = OFF
 Database page size                                      = 4096

 Dynamic SQL Query management           (DYN_QUERY_MGMT) = DISABLE

 Statement concentrator                      (STMT_CONC) = OFF

 Discovery support for this database       (DISCOVER_DB) = ENABLE

 Restrict access                                         = NO
 Default query optimization class         (DFT_QUERYOPT) = 5
 Degree of parallelism                      (DFT_DEGREE) = 1
 Continue upon arithmetic exceptions   (DFT_SQLMATHWARN) = NO
 Default refresh age                   (DFT_REFRESH_AGE) = 0
 Default maintained table types for opt (DFT_MTTB_TYPES) = SYSTEM
 Number of frequent values retained     (NUM_FREQVALUES) = 10
 Number of quantiles retained            (NUM_QUANTILES) = 20

 Decimal floating point rounding mode  (DECFLT_ROUNDING) = ROUND_HALF_EVEN

 Backup pending                                          = NO

 All committed transactions have been written to disk    = NO
 Rollforward pending                                     = NO
```

```
Restore pending                                         = NO

Multi-page file allocation enabled                      = YES

Log retain for recovery status                          = NO
User exit for logging status                            = YES

Self tuning memory                    (SELF_TUNING_MEM) = ON
Size of database shared memory (4KB)  (DATABASE_MEMORY) = AUTOMATIC(406920)
Database memory threshold               (DB_MEM_THRESH) = 10
Max storage for lock list (4KB)              (LOCKLIST) = 6200
Percent. of lock lists per application       (MAXLOCKS) = 60
Package cache size (4KB)                   (PCKCACHESZ) = AUTOMATIC(2095)
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = AUTOMATIC(268)
Sort list heap (4KB)                         (SORTHEAP) = AUTOMATIC(53)

Database heap (4KB)                            (DBHEAP) = AUTOMATIC(3233)
Catalog cache size (4KB)              (CATALOGCACHE_SZ) = 300
Log buffer size (4KB)                        (LOGBUFSZ) = 256
Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 286142
Buffer pool size (pages)                     (BUFFPAGE) = 1000
SQL statement heap (4KB)                     (STMTHEAP) = 4096
Default application heap (4KB)             (APPLHEAPSZ) = AUTOMATIC(256)
Application Memory Size (4KB)             (APPL_MEMORY) = AUTOMATIC(40000)
Statistics heap size (4KB)               (STAT_HEAP_SZ) = AUTOMATIC(4384)

Interval for checking deadlock (ms)         (DLCHKTIME) = 10000
Lock timeout (sec)                        (LOCKTIMEOUT) = -1

Changed pages threshold                (CHNGPGS_THRESH) = 80
Number of asynchronous page cleaners   (NUM_IOCLEANERS) = AUTOMATIC(7)
Number of I/O servers                   (NUM_IOSERVERS) = AUTOMATIC(3)
Index sort flag                             (INDEXSORT) = YES
Sequential detect flag                      (SEQDETECT) = YES
Default prefetch size (pages)         (DFT_PREFETCH_SZ) = AUTOMATIC

Track modified pages                         (TRACKMOD) = OFF

Default number of containers                            = 1
Default tablespace extentsize (pages)   (DFT_EXTENT_SZ) = 2

Max number of active applications            (MAXAPPLS) = AUTOMATIC(40)
Average number of active applications       (AVG_APPLS) = 1
Max DB files open per application            (MAXFILOP) = 61440

Log file size (4KB)                         (LOGFILSIZ) = 10240
Number of primary log files                (LOGPRIMARY) = 20
Number of secondary log files               (LOGSECOND) = 20
Changed path to log files                  (NEWLOGPATH) =
 Path to log files                                       =
/db2/db2inst1/NODE0000/SQL00001/SQLOGDIR/
 Overflow log path                     (OVERFLOWLOGPATH) =
 Mirror log path                         (MIRRORLOGPATH) =
 First active log file                                   = S0000189.LOG
 Block log on disk full                (BLK_LOG_DSK_FUL) = NO
 Block non logged operations            (BLOCKNONLOGGED) = NO
 Percent max primary log space by transaction  (MAX_LOG) = 0
 Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0

 Group commit count                          (MINCOMMIT) = 1
 Percent log file reclaimed before soft chckpt (SOFTMAX) = 520
 Log retain for recovery enabled             (LOGRETAIN) = OFF
 User exit for logging enabled                (USEREXIT) = OFF

 HADR database role                                      = PRIMARY
 HADR local host name                  (HADR_LOCAL_HOST) = eg09ph16
 HADR local service name                (HADR_LOCAL_SVC) = 65000
 HADR remote host name                (HADR_REMOTE_HOST) = eg09ph15
 HADR remote service name              (HADR_REMOTE_SVC) = 65000
 HADR instance name of remote server  (HADR_REMOTE_INST) = db2inst1
 HADR timeout value                       (HADR_TIMEOUT) = 120
 HADR log write synchronization mode     (HADR_SYNCMODE) = SYNC
 HADR peer window duration (seconds)  (HADR_PEER_WINDOW) = 120

 First log archive method                 (LOGARCHMETH1) =
DISK:/backup/r11_hadr/archivelogs/
 Options for logarchmeth1                  (LOGARCHOPT1) =
 Second log archive method                (LOGARCHMETH2) = OFF
 Options for logarchmeth2                  (LOGARCHOPT2) =
 Failover log archive path                (FAILARCHPATH) =
 Number of log archive retries on error   (NUMARCHRETRY) = 5
 Log archive retry Delay (secs)         (ARCHRETRYDELAY) = 20
 Vendor options                              (VENDOROPT) =

 Auto restart enabled                      (AUTORESTART) = ON
 Index re-creation time and redo index build  (INDEXREC) = SYSTEM (RESTART)
 Log pages during index build            (LOGINDEXBUILD) = ON
 Default number of loadrec sessions    (DFT_LOADREC_SES) = 1
 Number of database backups to retain   (NUM_DB_BACKUPS) = 12
 Recovery history retention (days)     (REC_HIS_RETENTN) = 366
 Auto deletion of recovery objects    (AUTO_DEL_REC_OBJ) = OFF

 TSM management class                    (TSM_MGMTCLASS) =
 TSM node name                            (TSM_NODENAME) =
 TSM owner                                   (TSM_OWNER) =
 TSM password                             (TSM_PASSWORD) =

 Automatic maintenance                      (AUTO_MAINT) = ON
```

```
  Automatic database backup            (AUTO_DB_BACKUP) = OFF
  Automatic table maintenance          (AUTO_TBL_MAINT) = ON
    Automatic runstats                  (AUTO_RUNSTATS) = ON
      Automatic statement statistics  (AUTO_STMT_STATS) = ON
    Automatic statistics profiling    (AUTO_STATS_PROF) = OFF
      Automatic profile updates         (AUTO_PROF_UPD) = OFF
    Automatic reorganization               (AUTO_REORG) = OFF

Auto-Revalidation                          (AUTO_REVAL) = DEFERRED
Currently Committed                        (CUR_COMMIT) = ON
CHAR output with DECIMAL input        (DEC_TO_CHAR_FMT) = NEW
Enable XML Character operations        (ENABLE_XMLCHAR) = YES
WLM Collection Interval (minutes)     (WLM_COLLECT_INT) = 0
Monitor Collect Settings
Request metrics                       (MON_REQ_METRICS) = BASE
Activity metrics                      (MON_ACT_METRICS) = BASE
Object metrics                        (MON_OBJ_METRICS) = BASE
Unit of work events                      (MON_UOW_DATA) = NONE
Lock timeout events                   (MON_LOCKTIMEOUT) = NONE
Deadlock events                          (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events                         (MON_LOCKWAIT) = NONE
Lock wait event threshold               (MON_LW_THRESH) = 5000000
Number of package list entries         (MON_PKGLIST_SZ) = 32
Lock event notification level         (MON_LCK_MSG_LVL) = 1

SMTP Server                               (SMTP_SERVER) =
SQL conditional compilation flags         (SQL_CCFLAGS) =
Section actuals setting               (SECTION_ACTUALS) = NONE
Connect procedure                        (CONNECT_PROC) =
```

Results of the tests

This section provides the results for the following tests:

- High availability test with DB2 pureScale in the active-active mode
- High availability test with DB2 9.7 using the active-passive database
- Disaster recovery test with DB2 9.7 in active-active mode

High availability test with DB2 pureScale in the active-active mode

This section highlights the test results for active-active mode by using DB2 pureScale.

Read test

This section provides information about the read failover tests.

Set 1: WebSphere node failover during read query

This read test completed successfully. Upon failure of the WebSphere Application Server, the message in Example A-9 was displayed in the WebSphere Application Server log file.

Example A-9 Message in the WebSphere Application Server log file at failure

```
[08/09/11 14:40:03:742 BST] 00000018 RoleMember    I   DCSV8052I: DCS Stack
DefaultCoreGroup.FailOverCluster at Member
eg09ph01Cell01\eg09ph03Node01\FailOverServer2: Core group membership set
changed. Removed: [eg09ph01Cell01\eg09ph02Node01\FailOverServer1].
[08/09/11 14:40:03:771 BST] 00000035 WASSession    I MTMCacheMsgListener event
Received event REPLICATION_DOWN.
```

After a successful restart, the message in Example A-10 was displayed in the WebSphere Application Server log file.

Example A-10 Message in the WebSphere Application Server log file at restart

```
[08/09/11 14:42:05:572 BST] 00000006 ViewReceiver  I   DCSV1033I: DCS Stack
DefaultCoreGroup.FailOverCluster at Member
eg09ph01Cell01\eg09ph03Node01\FailOverServer2: Confirmed all new view members
in view identifier (4:0.eg09ph01Cell01\eg09ph02Node01\FailOverServer1). View
channel type is View|Ptp.
[08/09/11 14:42:06:102 BST] 0000003f WASSession    I MTMCacheMsgListener event
Received event REPLICATION_UP.
```

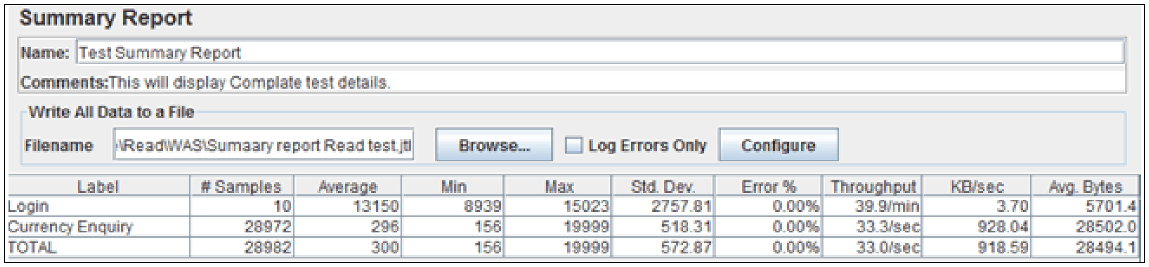
Example A-11 shows the failure and restart messages of WebSphere Application Server 2.

Example A-11 Failure and restart messages for WebSphere Application Server 2

```
[08/09/11 14:45:56:428 BST] 00000018 RoleMember    I   DCSV8052I: DCS Stack
DefaultCoreGroup.FailOverCluster at Member
eg09ph01Cell01\eg09ph02Node01\FailOverServer1: Core group membership set
changed. Removed: [eg09ph01Cell01\eg09ph03Node01\FailOverServer2].
[08/09/11 14:45:56:446 BST] 0000003a WASSession    I MTMCacheMsgListener event
Received event REPLICATION_DOWN.

[08/09/11 14:48:50:269 BST] 00000005 ViewReceiver  I   DCSV1033I: DCS Stack
DefaultCoreGroup.FailOverCluster at Member
eg09ph01Cell01\eg09ph02Node01\FailOverServer1: Confirmed all new view members
in view identifier (6:0.eg09ph01Cell01\eg09ph02Node01\FailOverServer1). View
channel type is View|Ptp.
```

```
[08/09/11 14:48:50:783 BST] 00000038 WASSession    I MTMCacheMsgListener event
Received event REPLICATION_UP.
```

In the JMeter script, the reads proceeded successfully during the process. Figure A-1 shows the Summary Report.


Summary Report

Name: Test Summary Report

Comments:This will display Complate test details.

Write All Data to a File

Filename \ReadWAS\Sumaary report Read test.jtl

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	10	13150	8939	15023	2757.81	0.00%	39.9/min	3.70	5701.4
Currency Enquiry	28972	296	156	19999	518.31	0.00%	33.3/sec	928.04	28502.0
TOTAL	28982	300	156	19999	572.87	0.00%	33.0/sec	918.59	28494.1

Figure A-1 Summary Report

Set 2: MQ node failover during read query

During the MQ node failure, the messages shown in Example A-12 were in the MQ log file.

Example A-12 MQ messages

```
09/02/11 18:27:06 - Process(17170628.5) User(mqm) Program(amqrmppa)
                    Host(eg09ph05)
AMQ9209: Connection to host 'eg09ph02 (172.19.81.92)' closed.
EXPLANATION:
An error occurred receiving data from 'eg09ph02 (172.19.81.92)' over TCP/IP.
The connection to the remote host has unexpectedly terminated.
ACTION:
Tell the systems administrator.
----- amqccita.c : 3472 -------------------------------------------------------
09/02/11 18:27:06 - Process(17170628.5) User(mqm) Program(amqrmppa)
                    Host(eg09ph05)
AMQ9999: Channel program ended abnormally.

EXPLANATION:
Channel program 'SYSTEM.DEF.SVRCONN' ended abnormally.
ACTION:
Look at previous error messages for channel program 'SYSTEM.DEF.SVRCONN' in the
error files to determine the cause of the failure.
----- amqrmrsa.c : 533 --------------------------------------------------------
09/02/11 18:27:06 - Process(19333170.159) User(mqm) Program(amqrmppa)
                    Host(eg09ph05)
AMQ9209: Connection to host 'eg09ph02 (172.19.81.92)' closed.

EXPLANATION:
An error occurred receiving data from 'eg09ph02 (172.19.81.92)' over TCP/IP.
```

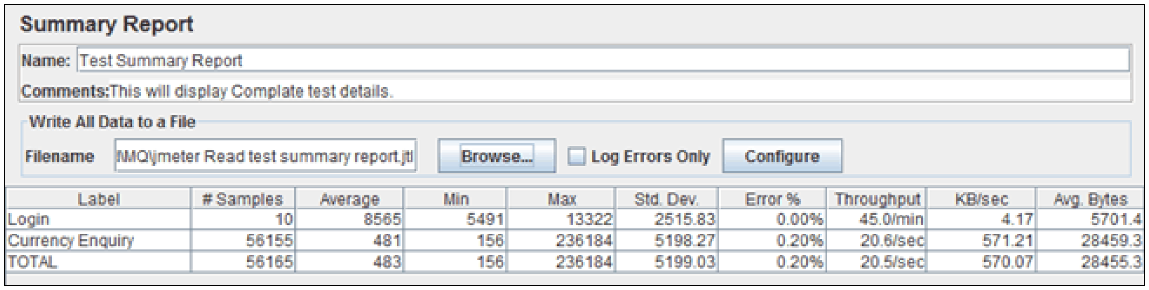
```
The connection to the remote host has unexpectedly terminated.
ACTION:
Tell the systems administrator.
```

The failover of the MQ node to passive mode took approximately a few minutes, where most of the time was spent in storage movement from one partition to the other partition. During this time, a few of the read transactions failed in the JMeter script with the message shown in Example A-13.

Example A-13 Failed message

```
09/02/11 18:31:43 - Process(19136578.1) User(mqm) Program(amqzmgr0)
                    Host(eg09ph04)
AMQ5026: The Listener 'T24Listener' has started. ProcessId(20709546).

EXPLANATION:
The Listener process has started.
ACTION:
None.
-------------------------------------------------------------------------------
09/02/11 18:31:43 - Process(8650850.1) User(mqm) Program(runmqchi)
                    Host(eg09ph04)
AMQ8024: WebSphere MQ channel initiator started.

EXPLANATION:
The channel initiator for queue  has been started.
ACTION:
None.
```

Figure A-2 shows the Summary Report.



Name: Test Summary Report

Comments: This will display Complate test details.

Filename: NMQ\jmeter Read test summary report.jtl

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	10	8565	5491	13322	2515.83	0.00%	45.0/min	4.17	5701.4
Currency Enquiry	56155	481	156	236184	5198.27	0.20%	20.6/sec	571.21	28459.3
TOTAL	56165	483	156	236184	5199.03	0.20%	20.5/sec	570.07	28455.3

Figure A-2 Summary Report

Set 3: T24 node failover during the read query

The browser ran with a currency table read. T24 was stopped while the read was progressing, and the test completed successfully. See the message in Example A-14.

Example A-14 Testing message

```
AIX-/usr/Temenos/T24/R11.testbase/TestBase.run: jbase_agent
(Tue Sep 13 2011 11:31:26.642921|10420242|1) NOTICE starting up jAgent, Process Per Connection
mode, listening on port 20002, SocketAcceptor.h +111
(Tue Sep 13 2011 11:32:20.497560|14156002|1) NOTICE RequestHandlerService::open: connected with
eg09ph02, RequestHandlerService.cpp +221
................................
(Tue Sep 13 2011 11:35:00.070092|8192146|1) NOTICE RequestHandlerService::open: connected with
eg09ph02, RequestHandlerService.cpp +221
 (Tue Sep 13 2011 11:35:17.577154|10420242|1) NOTICE (%P|%t) shutting down SocketAcceptor,
SocketAcceptor.h +120
AIX-/usr/Temenos/T24/R11.testbase/TestBase.run: jbase_agent
(Tue Sep 13 2011 11:36:53.687644|16908400|1) NOTICE starting up jAgent, Process Per Connection
mode, listening on port 20002, SocketAcceptor.h +111
(Tue Sep 13 2011 11:37:12.994750|16384196|1) NOTICE RequestHandlerService::open: connected with
eg09ph02, RequestHandlerService.cpp +221
```

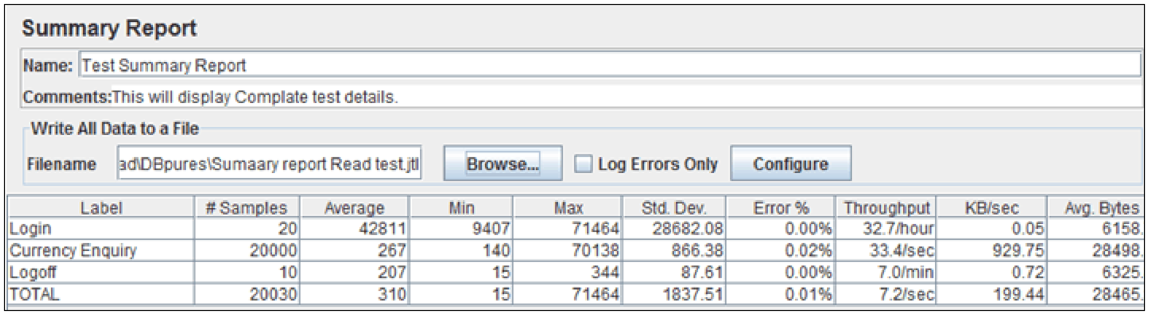
Set 4: The DB2 pureScale elements failover

During the database disconnect, the message shown in Example A-15 was displayed in the jAgent.

Example A-15 Message in the jAgent

```
[12/09/11 11:00:56:749 BST] 0000003f JRemoteSocket W   Failed to receive message
[12/09/11 11:00:56:751 BST] 0000003f T24Connection E   T24 transaction failed. Error reference:
[97a75c6e-fdc8-4a32-a95a-db84ead90475] - Message [Failed to receive message - Server
disconnected]
```

Figure A-3 shows the Summary Report.



Summary Report

Name: Test Summary Report

Comments: This will display Complate test details.

Write All Data to a File

Filename: ad\DBpures\Sumaary report Read test.jtl

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	20	42811	9407	71464	28682.08	0.00%	32.7/hour	0.05	6158.
Currency Enquiry	20000	267	140	70138	866.38	0.02%	33.4/sec	929.75	28498.
Logoff	10	207	15	344	87.61	0.00%	7.0/min	0.72	6325.
TOTAL	20030	310	15	71464	1837.51	0.01%	7.2/sec	199.44	28465.

Figure A-3 Summary Report

Write test

This section provides information about the write failover tests.

Set 1: WebSphere node failover during write query

When the WebSphere Application Server was bought down, the message shown in Example A-16 was in the `/usr/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/FailOverServer1` WebSphere log file.

Example A-16 Test message

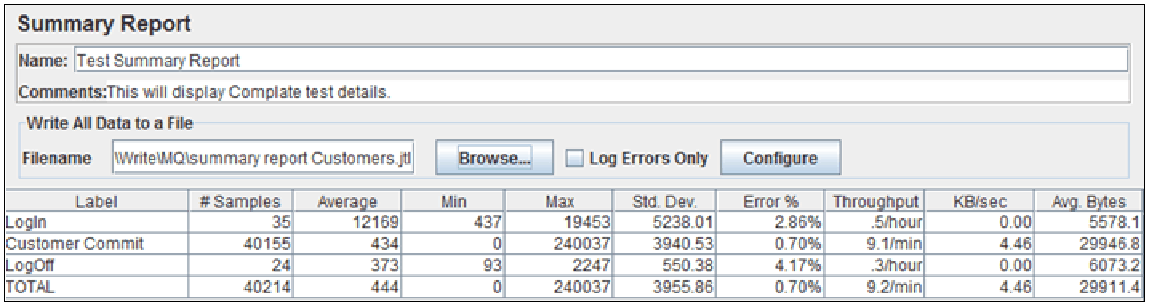
```
[08/09/11 16:01:36:772 BST] 0000001c RoleMember    I   DCSV8052I: DCS Stack
DefaultCoreGroup.FailOverCluster at Member eg09ph01Cell01\eg09ph02Node01\FailOverServer1: Core
group membership set changed. Removed: [eg09ph01Cell01\eg09ph03Node01\FailOverServer2].
[08/09/11 16:01:36:773 BST] 0000003c WASSession    I MTMCacheMsgListener event Received event
REPLICATION_DOWN.
While the initial server was bought up, the below message was seen in the WebSphere  log,
[08/09/11 16:05:59:130 BST] 00000004 ViewReceiver  I   DCSV1033I: DCS Stack
DefaultCoreGroup.FailOverCluster at Member eg09ph01Cell01\eg09ph02Node01\FailOverServer1:
Confirmed all new view members in view identifier
(6:0.eg09ph01Cell01\eg09ph02Node01\FailOverServer1). View channel type is View|Ptp.
[08/09/11 16:05:59:661 BST] 0000003d WASSession    I MTMCacheMsgListener event Received event
REPLICATION_UP.
```

A similar set of messages exists for the other WebSphere Application Server stops and starts is in the `/usr/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/FailOverServer2` log file.

The write tests completed successfully.

Set 2: MQ node failover during write query

The MQ failover was performed similar to the read testing. A few of the transactions failed during MQ failover time, and the testing progressed successfully. Figure A-4 shows the Summary Report.



Summary Report

Name: Test Summary Report

Comments: This will display Complate test details.

Write All Data to a File

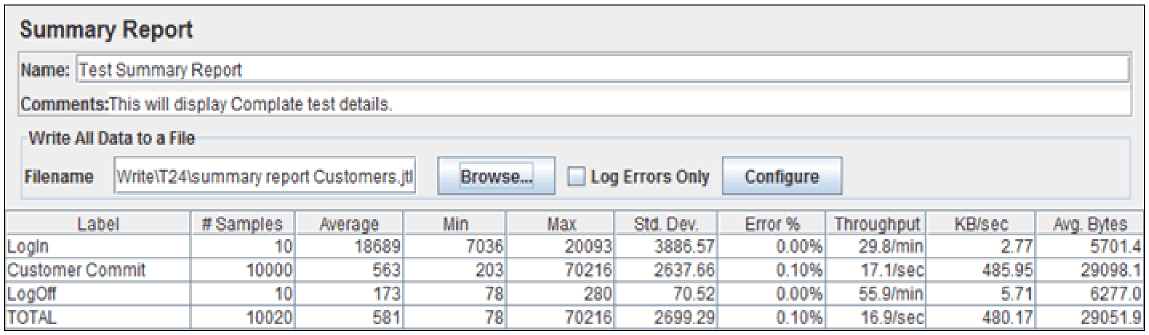
Filename: \Write\MQ\summary report Customers.jtl

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
LogIn	35	12169	437	19453	5238.01	2.86%	.5/hour	0.00	5578.1
Customer Commit	40155	434	0	240037	3940.53	0.70%	9.1/min	4.46	29946.8
LogOff	24	373	93	2247	550.38	4.17%	.3/hour	0.00	6073.2
TOTAL	40214	444	0	240037	3955.86	0.70%	9.2/min	4.46	29911.4

Figure A-4 Summary Report

Set 3: T24 node failover during write query

During the T24 failure and restart, the transactions completed successfully, with few failures. Figure A-5 shows the Summary Report.



Summary Report

Name: Test Summary Report

Comments: This will display Complate test details.

Write All Data to a File

Filename: Write\T24\summary report Customers.jtl

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
LogIn	10	18689	7036	20093	3886.57	0.00%	29.8/min	2.77	5701.4
Customer Commit	10000	563	203	70216	2637.66	0.10%	17.1/sec	485.95	29098.1
LogOff	10	173	78	280	70.52	0.00%	55.9/min	5.71	6277.0
TOTAL	10020	581	78	70216	2699.29	0.10%	16.9/sec	480.17	29051.9

Figure A-5 Summary Report

Sets 4 - 7: DB2 pureScale member failover, member addition, and elements failover during write query

The testing for sets 4 - 7 was performed in a sequence and progressed successfully.

COB test

This section provides details about the COB tests.

Sets 1 - 4: DB2 pureScale member failover, member addition, and elements failover during IC COB

Set 1 through 4 was performed. After a member of the DB2 pureScale is stopped, it can be verified by using the following command:

```
[eg09ph12:db2sdin1] /db2sdin1 $ /db2instance -list
```

Example A-17 shows the resulting instance list.

Example A-17 DB2 instance list

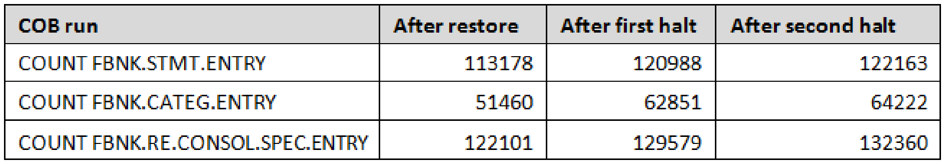
```
ID       TYPE             STATE        HOME_HOST     CURRENT_HOST     ALERT   PARTITION_NUMBER    LOGICAL_PORT    NETNAME
--       ----             -----        ---------     ------------     -----   ----------------    ------------    -------
0       MEMBER  WAITING_FOR_FAILBACK   eg09ph12      eg09ph14         YES          0                     1        eg09ph14-ib0
1       MEMBER          STARTED        eg09ph13      eg09ph13         YES          0                     0        eg09ph13-ib0
2       MEMBER          STARTED        eg09ph14      eg09ph14         NO           0                     0        eg09ph14-ib0
128     CF              PRIMARY        eg09ph10      eg09ph10         NO           -                     0        eg09ph10-ib0
129     CF              CATCHUP        eg09ph11      eg09ph11         NO           -                     0        eg09ph11-ib0

HOSTNAME                STATE                  INSTANCE_STOPPED       ALERT
--------                -----                  ----------------       -----
eg09ph11                ACTIVE                               NO          NO
eg09ph10                ACTIVE                               NO          NO
eg09ph14                ACTIVE                               NO         YES
eg09ph13                ACTIVE                               NO         YES
eg09ph12                ACTIVE                               NO         YES
```

When the DB2 pureScale member of CF went down, the information in Example A-18 was shown in the XML driver.

Example A-18 Information in the XML driver

```
R11.95548 - 9633832 - (jmainfunction.b,0) - Fri Sep 23 15:43:17 - F.LOCKING - F_LOCKING -
SQLSTATE: 40003  ErrorCode: -30081
Message:[IBM][CLI Driver] SQL30081N  A communication error has been detected. Communication
protocol being used: "TCP/IP".  Communication API being used: "SOCKETS".  Location where the
error was detected: "172.19.82.94".  Communication function detecting the error: "recv".
Protocol specific error code(s): "*", "*", "0".  SQLSTATE=08001
```

Figure A-6 shows record counts of the tables during the interest capitalization close-of-business (IC COB) failover test.

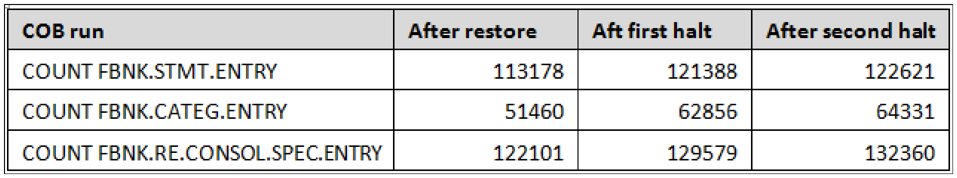


COB run	After restore	After first halt	After second halt
COUNT FBNK.STMT.ENTRY	113178	120988	122163
COUNT FBNK.CATEG.ENTRY	51460	62851	64222
COUNT FBNK.RE.CONSOL.SPEC.ENTRY	122101	129579	132360

Figure A-6 Record counts

Set 5: T24 node failover during IC COB

The running jbase_agent was stopped, and the COB table count was verified. Figure A-7 shows the record counts.

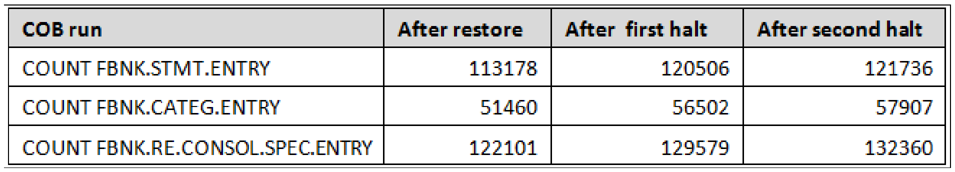


COB run	After restore	Aft first halt	After second halt
COUNT FBNK.STMT.ENTRY	113178	121388	122621
COUNT FBNK.CATEG.ENTRY	51460	62856	64331
COUNT FBNK.RE.CONSOL.SPEC.ENTRY	122101	129579	132360

Figure A-7 Record counts of the tables during the IC COB failover test

Set 6: T24 node failover during IC COB

When the T24 Service Manager stopped, the agents continued running. When they stopped, we restarted Tivoli Storage Manager and then restarted the agents to complete the process of failover. Figure A-8 shows the record counts of the tables during the IC COB failover test.



COB run	After restore	After first halt	After second halt
COUNT FBNK.STMT.ENTRY	113178	120506	121736
COUNT FBNK.CATEG.ENTRY	51460	56502	57907
COUNT FBNK.RE.CONSOL.SPEC.ENTRY	122101	129579	132360

Figure A-8 Record counts of the tables during the IC COB failover test

High availability test with DB2 9.7 using the active-passive database

This section explains the test results of the high availability testing for DB2 9.7.

Read test

The message in Example A-19 was displayed in the `XMLdriver.log` file.

Example A-19 Message in XMLdriver.log file

```
R11.95548 - 7012488 - (jmainfunction.b,0) - Tue Sep 27 15:49:32 -  -  - SQLSTATE: 08001
ErrorCode: -30081
Message:[IBM][CLI Driver] SQL30081N  A communication error has been detected. Communication
protocol being used: "TCP/IP".  Communication API being used: "SOCKETS".  Location where the
error was detected: "172.19.82.220".  Communication function detecting the error: "connect".
Protocol specific error code(s): "79", "*", "*".  SQLSTATE=08001
```

In the jbase_agent, you can see the error message (Example A-20) and that the agent continuing to run as intended.

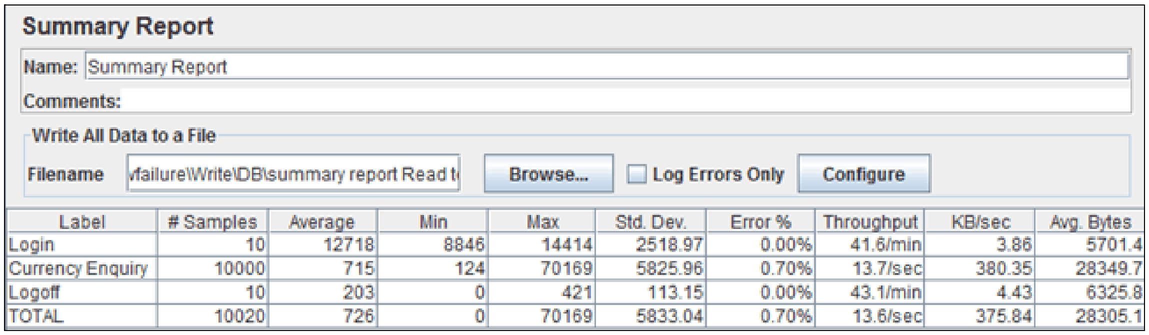
Example A-20 Error message in the jbase_agent

```
drvOpen: *** ERROR *** Unable to initialise DriverData Structure, check ./XMLdriver.log for
more information.
READ: I/O error
(Wed Sep 28 2011 06:33:20.635115|7799018|515) ERROR CALL: Routine invoked debugger or INPUT
with input disabled, StoredProcedure.cpp +146
(Wed Sep 28 2011 06:33:20.671971|12910798|1) NOTICE RequestHandlerService::open: connected with
eg09ph03, RequestHandlerService.cpp +221
```

The browser was used to verify the reads. A wait occurred for a few seconds when the DB2 was remapped by PowerHA, and then the testing continued.

Write test

Similar error messages are in the `XMLdriver.log` file and the jabase_agent panel. Figure A-9 shows the Summary Report of the write test.



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	10	12718	8846	14414	2518.97	0.00%	41.6/min	3.86	5701.4
Currency Enquiry	10000	715	124	70169	5825.96	0.70%	13.7/sec	380.35	28349.7
Logoff	10	203	0	421	113.15	0.00%	43.1/min	4.43	6325.8
TOTAL	10020	726	0	70169	5833.04	0.70%	13.6/sec	375.84	28305.1

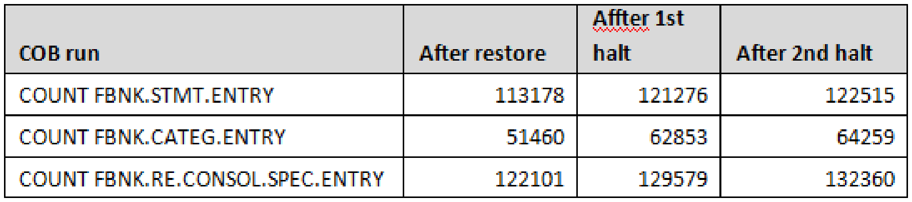
Figure A-9 Summary Report of the write test

COB test

The COB run had the following record counts:

- Software kill

  Figure A-10 shows the record counts of the table for the software skill failover test.

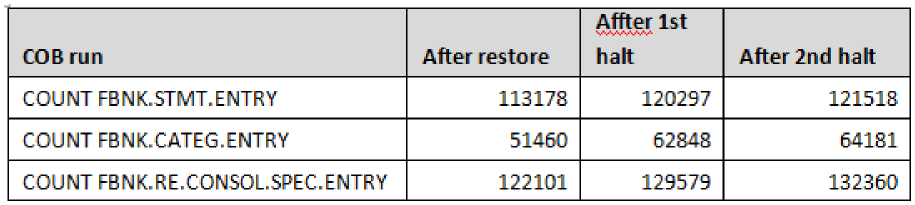


COB run	After restore	Affter 1st halt	After 2nd halt
COUNT FBNK.STMT.ENTRY	113178	121276	122515
COUNT FBNK.CATEG.ENTRY	51460	62853	64259
COUNT FBNK.RE.CONSOL.SPEC.ENTRY	122101	129579	132360

Figure A-10 Record counts of the tables during the IC COB failover test

- Hardware kill

  Figure A-11 shows the record counts of the table for the hardware kill failover test.



COB run	After restore	Affter 1st halt	After 2nd halt
COUNT FBNK.STMT.ENTRY	113178	120297	121518
COUNT FBNK.CATEG.ENTRY	51460	62848	64181
COUNT FBNK.RE.CONSOL.SPEC.ENTRY	122101	129579	132360

Figure A-11 Record counts of the tables during the IC COB failover test

Disaster recovery test with DB2 9.7 in active-active mode

Example A-21 shows the initial standby site before the takeover.

Example A-21 Standby site before takeover

```
[eg09ph16:db2inst1] /hadr $ ./monitor.sh

Database Partition 0 -- Database R11_HADR -- Standby -- Up 0 days 00:00:55 --
Date 09/29/2011 15:58:49

HADR Information:
Role    State                SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Standby Peer                 Sync     0                  0

ConnectStatus ConnectTime                           Timeout
Connected     Thu Sep 29 15:58:32 2011 (1317308312) 120

PeerWindowEnd                         PeerWindow
Thu Sep 29 16:00:37 2011 (1317308437) 120

LocalHost                                LocalService
eg09ph16                                 65000

RemoteHost                               RemoteService      RemoteInstance
eg09ph15                                 65000              db2inst1

PrimaryFile  PrimaryPg  PrimaryLSN
S0000117.LOG 0          0x0000000315B28010

StandByFile  StandByPg  StandByLSN         StandByRcvBufUsed
S0000117.LOG 0          0x0000000315B28010 0%
```

Upon failure, manual takeover of the disaster recovery site occurs. Example A-22 shows the **`takeover`** command.

Example A-22 The takeover command and status after the takeover

```
[eg09ph16:db2inst1] /hadr $ db2 takeover hadr on db r11_hadr by force
DB20000I  The TAKEOVER HADR ON DATABASE command completed successfully.

[eg09ph16:db2inst1] /hadr $ db2pd -db r11_hadr -hadr

Database Partition 0 -- Database R11_HADR -- Active -- Up 0 days 00:22:53 --
Date 09/29/2011 16:20:46

HADR Information:
Role    State                SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Primary Disconnected         Sync     0                  0
```

```
ConnectStatus ConnectTime                          Timeout
Disconnected  Thu Sep 29 16:17:16 2011 (1317309436) 120

PeerWindowEnd                          PeerWindow
Null (0)                               120

LocalHost                                 LocalService
eg09ph16                                  65000

RemoteHost                                RemoteService      RemoteInstance
eg09ph15                                  65000              db2inst1

PrimaryFile  PrimaryPg  PrimaryLSN
S0000117.LOG 2803       0x000000031661BEC1

StandByFile  StandByPg  StandByLSN
S0000000.LOG 0          0x0000000000000000
```

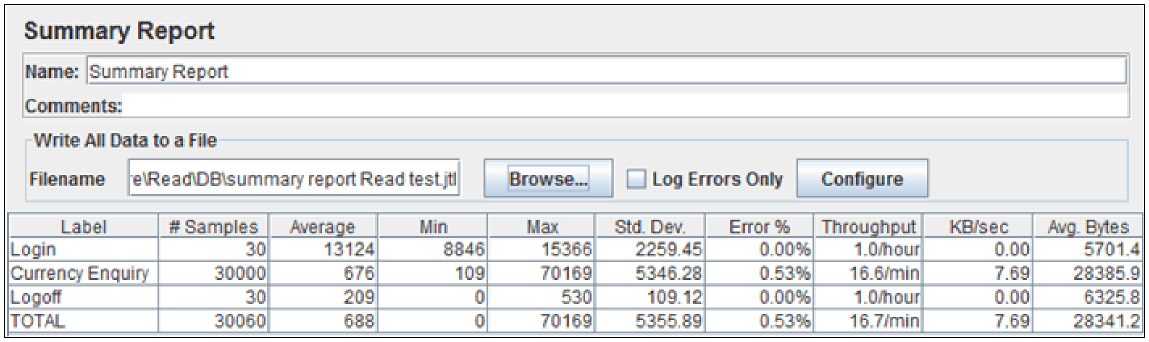
Example A-23 shows the error message in the `XMLdriver.log` file.

Example A-23 Error message in the XMLdriver.log file

```
R11.95548 - 14680178 - (jmainfunction.b,0) - Thu Sep 29 16:17:47 - F.OS.TOKEN - F_OS_TOKEN -
SQLSTATE: 40003  ErrorCode: -30081
Message:[IBM][CLI Driver] SQL30081N  A communication error has been detected. Communication
protocol being used: "TCP/IP".  Communication API being used: "SOCKETS".  Location where the
error was detected: "172.19.82.95".  Communication function detecting the error: "recv".
Protocol specific error code(s): "*", "*", "0".  SQLSTATE=08001
Messages found in jbase_agent screen,
(Thu Sep 29 2011 16:17:47.446155|14680178|515) ERROR CALL: Routine invoked debugger or INPUT
with input disabled, StoredProcedure.cpp +146
READ: I/O error
(Thu Sep 29 2011 16:17:47.459911|7929858|515) ERROR CALL: Routine invoked debugger or INPUT
with input disabled, StoredProcedure.cpp +146
(Thu Sep 29 2011 16:22:12.012140|10813496|1) NOTICE RequestHandlerService::open: connected with
eg09ph03, RequestHandlerService.cpp +221
```

Read test

Figure A-12 shows the JMeter Summary Report for the read test.


Summary Report

Name: Summary Report

Comments:

Write All Data to a File

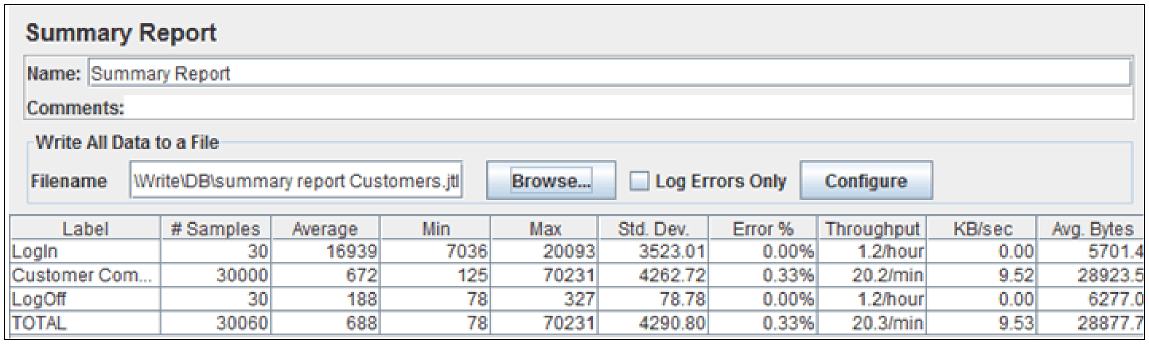
Filename: e\Read\DB\summary report Read test.jtl

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Login	30	13124	8846	15366	2259.45	0.00%	1.0/hour	0.00	5701.4
Currency Enquiry	30000	676	109	70169	5346.28	0.53%	16.6/min	7.69	28385.9
Logoff	30	209	0	530	109.12	0.00%	1.0/hour	0.00	6325.8
TOTAL	30060	688	0	70169	5355.89	0.53%	16.7/min	7.69	28341.2

Figure A-12 JMeter Summary Report for the read test

Write test

Figure A-13 shows the JMeter Summary Report for the write test.


Summary Report

Name: Summary Report

Comments:

Write All Data to a File

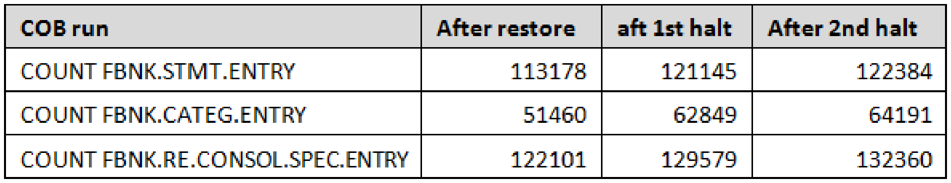
Filename: \Write\DB\summary report Customers.jtl

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
LogIn	30	16939	7036	20093	3523.01	0.00%	1.2/hour	0.00	5701.4
Customer Com...	30000	672	125	70231	4262.72	0.33%	20.2/min	9.52	28923.5
LogOff	30	188	78	327	78.78	0.00%	1.2/hour	0.00	6277.0
TOTAL	30060	688	78	70231	4290.80	0.33%	20.3/min	9.53	28877.7

Figure A-13 JMeter Summary Report for write test

COB test

Figure A-14 shows the COB test results.



COB run	After restore	aft 1st halt	After 2nd halt
COUNT FBNK.STMT.ENTRY	113178	121145	122384
COUNT FBNK.CATEG.ENTRY	51460	62849	64191
COUNT FBNK.RE.CONSOL.SPEC.ENTRY	122101	129579	132360

Figure A-14 Record counts of the tables during the IC COB failover test

Abbreviations and acronyms

ACR	automatic client reroute
CCF	cluster caching facility
CDC	Change Data Capture
CF	Coupling facility
COB	close of business
DBA	database administrator
DCD	Direct Connect Driver
EE	Enterprise Edition
GDPC	Geographically Dispersed DB2 pureScale Cluster
GPFS	General Parallel File System
HA	high availability
HADR	high availability disaster recovery
IBM	International Business Machines Corporation
IC COB	interest capitalization close of business
ITSO	International Technical Support Organization
J2EE	Java 2 Platform, Enterprise Edition
JCA	Java EE Connector Architecture
JDLS	jBASE Distributed Lock Service
JMX	Java Management Extensions
JVM	Java virtual machine
LPAR	logical partition
NIC	network interface card
OLTP	online transaction processing
PPRC	Peer-to-Peer Remote Copy
QM	queue manager
RAID	Redundant Array of Independent Disks
RDMA	remote direct memory access
RMC	Resource Monitoring and Control
RSCT	Reliable Scalable Cluster Technology
SAN	storage area network
SOA	service-oriented architecture
SPOF	single point of failure
TAFC	Temenos Application Framework for C
TAFJ	Temenos Application Framework for Java
TOCF-EE	Temenos Open connectivity framework - Enterprise Edition
TSA	T24 Service Agent
TSM	T24 Service Manager
TWS	Temenos Web Services
UK	United Kingdom
VG	volume group
VIOS	Virtual I/O Server
WLB	workload balancing

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *PowerHA for AIX Cookbook*, SG24-7739
- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources that were used for testing:

- The following documents in IBM developerWorks:
  - Configuring DB2 pureScale for disaster recovery using DS8700 Metro Mirror

    http://www.ibm.com/developerworks/data/library/techarticle/dm-1005purescalemetromirror/index.html
  - Configuring geographically dispersed DB2 pureScale clusters

    http://www.ibm.com/developerworks/data/library/long/dm-1104purescalegdpc/index.html?ca=drs-

  - DB2 best practices: SAP applications with the DB2 pureScale Feature on SUSE Linux Enterprise Server and IBM System x

    http://www.ibm.com/developerworks/data/bestpractices/purescalesapsuselinux/index.html?ca=drs-
  - Deploying the DB2 pureScale Feature

    http://www.ibm.com/developerworks/data/library/techarticle/dm-1005purescalefeature/
  - Performance tuning WebSphere Application Server 7 on AIX 6.1

    https://www.ibm.com/developerworks/mydeveloperworks/blogs/timdp/entry/performance_tuning_websphere_application_server_7_on_aix_6_1179?lang=en
- DB2 pureScale 9.8 AIX Installation prerequisites

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/index.jsp?topic=/com.ibm.db2.luw.sd.doc/doc/r0054850.html
- Network Deployment (Distributed operating systems), Version 7.0: Tuning AIX systems

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tprf_tuneaix.html
- Pre-installation checklist for DB2 pureScale Feature (AIX)

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/index.jsp?topic=/com.ibm.db2.luw.sd.doc/doc/r0056077.html
- *WebSphere Application Server Network Deployment V7 Installation Guide*

  http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.was_v7/was/7.0/InstallationAndMigration/WASv7_InstallationLab.pdf
- Using WebSphere MQ with a high availability cluster on UNIX

  http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.amqzag.doc/fa70230_.htm
- IBM WebSphere Application Server v7.0 Performance Optimization Scripts

  ftp://public.dhe.ibm.com/software/webservers/appserv/WASv7_Tuning_Script_Templates_v1_1.pdf
- Contact Temenos to obtain the following publications:
  - *jBASE Distributed Locking Guide*
  - *T24 DB2 Database Conversion Guide*
  - *Temenos/IBM High Availability and Disaster Recovery test plan*
  - *WebSphere_UG_JCA_HIGH_AVAILABILITY*

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

High Availability and Disaster Recovery for Temenos T24 with IBM DB2 and AIX

Redpaper™

T24 R11 on AIX with IBM WebSphere Application Server and WebSphere MQ

High availability and disaster recovery implementation

T24 R11 with DB2 9 and DB2 pureScale

The Temenos T24 core banking application is a critical application for the banks that use it and has a primary focus on providing an appropriate level of high availability and disaster recovery. The level of availability is determined largely by the configuration of the infrastructure that supports T24. This infrastructure is built on hardware, middleware, and networking, in addition to the operational procedures and practices that are used to operate T24.

Many options are available for meeting a client's high availability and disaster recovery requirements. The solution chosen by a Temenos T24 user depends on many factors. These factors include a user's detailed availability and recovery requirements; their existing datacenter standards, practices, and processes; and the available network infrastructure. Therefore, the optimum solution must be determined on a case-by-case basis for each deployment.

This IBM Redpaper publication serves as a guide to help IT architects and other technical staff who are designing, configuring, and building the infrastructure to support Temenos T24. It shows how IBM software can deliver high availability and disaster recovery for Temenos T24 to meet a client's requirements. This software might run on IBM AIX, IBM WebSphere Application Server, WebSphere MQ Server, and IBM DB2. These IBM software components are typically used for a Temenos T24 deployment on an IBM middleware stack to ensure a highly available infrastructure for T24.

REDP-4794-00