Application Note: Veritas High Availability Solution for TIBCO Enterprise Message Service

UNIX

VERITAS™

Veritas InfoScale™ Availability Agents

Last updated: 2018-06-14

Legal Notice

Veritas Technologies LLC
500 E Middlefield Road
Mountain View, CA 94043

http://www.veritas.com

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:

https://www.veritas.com/support

You can manage your Veritas account information at the following URL:

https://my.veritas.com

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)	CustomerCare@veritas.com
Japan	CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The latest documentation is available on the Veritas website:

https://sort.veritas.com/documents

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:

xyz@veritas.com

You can also see documentation information or ask a question on the Veritas community site:

http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:

https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Veritas High Availability Solution for TIBCO Enterprise Message Service

This document includes the following topics:

About the Veritas High Availability Solution

The Veritas High Availability Solution uses the following products: Veritas Storage Foundation, Veritas Cluster Server, and Cluster Server agents, which are designed specifically for TIBCO Enterprise Message Service. The result is a quickly deployable solution that protects critical TIBCO Enterprise Message Service servers from planned or unplanned downtime.

This document describes the overall Veritas High Availability Solution and the TIBCO Enterprise Message Service availability environment. It also offers a technical overview of local availability in the TIBCO Enterprise Message Service environment, and describes a few scenarios using the Veritas solution.

Veritas offers an end-to-end, fully integrated solution for ensuring highly available TIBCO Enterprise Message Service environments. The Veritas solution simplifies administration of complex environments with a single interface and supports local failover for business continuity purposes.

Components of the solution

The Veritas High Availability Solution integrates the following software products:

- Veritas Storage Foundation (SF)
  This product combines Veritas's industry-leading file system and volume management solutions to create a highly available, robust foundation for TIBCO Enterprise Message Service data. The journal file system restarts in seconds for fast failovers. Logical volumes support highly available, high performance storage configurations. Database-specific components such as direct I/O accelerate database read and write performance while simplifying the manageability of database data. Veritas Storage Foundation provides database-specific optimizations for Oracle, DB/2, Sybase, Windows, and Oracle RAC databases.
- Veritas Cluster Server (VCS)
  VCS eliminates planned and unplanned downtime by clustering critical applications and resources required by TIBCO Enterprise Message Service. All of the critical components of your TIBCO Enterprise Message Service environment are monitored and managed centrally to ensure maximum application availability.
- Veritas Cluster Server (VCS) agent for TibcoEMS Server

VCS agent for Tibco EMS server provides high availability for Tibco EMS server in a cluster. The agent can bring a specific Tibco EMS server online and monitor the state of the EMS server. The agent can also detect failures and shuts down the EMS server in case of a failure and cleans the environment in case of any issues.

For more details, refer to *Veritas High Availability Agent for TibcoEMS Server Installation and Configuration Guide*.

- Veritas Cluster File System (CFS)

  CFS allows clustered servers to mount and use a file system simultaneously as if all applications using the file system were running on the same server. This capability enables sharing of data stores between TIBCO Enterprise Message Service servers thereby ensuring the availability of a standby server for quick recovery in the event of a downtime. The Veritas Volume Manager cluster functionality (CVM) makes logical volumes and raw device applications accessible throughout a cluster.

Benefits of the solution

The combination of TIBCO Enterprise Message Service with Veritas Storage Foundation Cluster File System and Veritas Cluster Server delivers the following significant benefits:

- The cluster file system provides parallel access to files improving performance.
- The number of failovers are reduced because volumes and file systems do not need to be brought online after a recovery as they are already available to all nodes in the cluster.
- The availability of TIBCO Enterprise Message Service is improved in the following areas:

  CFS file lock management is significantly enhanced compared to NFS, which makes failover more reliable.

  VCS can detect faults outside of TIBCO Enterprise Message Service' awareness.

  Redundancy of the TIBCO solution is restored when a failover occurs.

About TIBCO Enterprise Message Service

TIBCO Enterprise Message Service is a leader in enterprise messaging platforms. TIBCO Enterprise Message Service allows for efficient system-to-system communications across different technologies. The communication may occur synchronously or asynchronously. Although the use cases vary, there is heavy usage in billing, financial analysis and alerting, and enterprise resource planning

(ERP) workloads. Common applications include trading software, shipping management, and internal supply-chain systems.

The TIBCO Enterprise Message Service customer base has a strong overlap with the Veritas customer base. Many TIBCO Enterprise Message Service customers also use Veritas Storage Foundation products. TIBCO Enterprise Message Service has tremendous penetration into the financial sector, airlines, telecommunications, technology, and shipping. Many of the published TIBCO Enterprise Message Service customers are currently customers of Veritas availability products.

While NFS can be used in conjunction with TIBCO's native fault-tolerance mechanisms, this configuration is discouraged. Customers have discovered that this solution is sub-optimal for meeting needs of application availability, throughput, and administration. Since then, there has been a strong and growing engagement from the field and corporate teams at TIBCO to utilize Veritas Storage Foundation Cluster File System (CFS) as the embedded solution for TIBCO.

Veritas offers a proven end-to-end, integrated solution for ensuring high availability for a wide variety of environments. By leveraging the industry leading Veritas Storage Foundation Cluster File System, Veritas Cluster Server, and the fault tolerant support in TIBCO Enterprise Message Service, the products can be combined to provide a high performance, highly available messaging solution.

Service level requirements for a typical TIBCO Enterprise Message Service environment are variable, but where performance and reliability are critical it is not uncommon to see requirements for 50,000 messages per second and 99.999% uptime. As customers have come to rely more and more on messaging services, the business drivers have come to expect these listed Service Level Agreements (SLA) or better. Customers seeking to obtain the highest level of availability and throughput can leverage the benefits of Veritas Storage Foundation and Veritas Cluster File System. The combined solution can eliminate the potential for data loss and guarantee maintenance of the data after messages are transmitted.

To provide the highest level of reliability and performance, shared storage must meet the criteria for fault tolerance required by TIBCO Enterprise Message Service, some of which are write-order fidelity, synchronous write persistence, and distributed locking.

When NAS hardware uses NFS as its file system, it is particularly difficult to determine whether the solution meets the criteria required for fault tolerant shared storage.

Research by Veritas indicates the following conclusions:

- NFS v2 definitely does not satisfy the criteria.
- NFS v3 with UDP definitely does not satisfy the criteria.
- NFS v3 with TCP might satisfy the criteria.

Consult with the NAS vendor to verify that the NFS server (in NAS) satisfies the criteria. Consult with the operating system vendor to verify that the NFS client (in the operating system on the server host computer) satisfies the criteria. When both vendors certify that their components cooperate to guarantee the criteria, then the shared storage solution supports TIBCO Enterprise Message Service.

In contrast, the combination of Veritas Storage Foundation Cluster File System and Veritas Cluster Server provides a shared storage solution with the following benefits:

- Full compliance with the TIBCO shared storage criteria for fault tolerance.
- Persistent shared storage for message data (for queues and topics), client connections to the primary server, and metadata about message delivery.
- Reduced storage costs due to over provisioning.
- Simplified administration by providing a single namespace across all nodes.
- Assured lock management and data integrity in the event of a node failure within the cluster.
- Ability to restore redundancy in the event of node failure due to a hardware or software fault.
- Monitoring of node health and proactive failure detection.

TIBCO Enterprise Message Service model for fault tolerance

This section reviews the TIBCO Enterprise Message Service default model for fault tolerance and directly cites from the TIBCO Enterprise Message Service User Guide, version 4.4 and the TIBCO Enterprise Message Service Installation Guide, version 4.4 released in November 2006.

TIBCO Enterprise Message Service servers may be configured for fault-tolerant operation by configuring a pair of servers—one primary and one backup. The primary server accepts client connections and interacts with clients to deliver messages. If the primary server fails, the backup server resumes operation in its place.

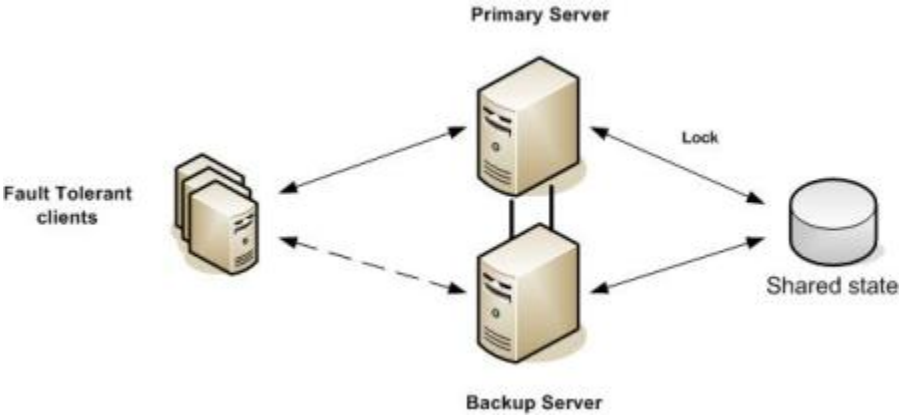
Note: TIBCO can load balance two servers in a fault-tolerant configuration for additional reliability. Multiple primary/secondary pairs can be utilized to increase the overall resilience. With the Storage Foundation Cluster File System and Veritas Cluster Server, it is possible to deploy additional failover servers into a cluster to further improve availability.

About shared state

A pair of fault-tolerant servers must have access to shared state, which consists of information about client connections and persistent messages. This information enables the backup server to properly assume responsibility for those connections and messages.

Figure 1-1 illustrates the fault-tolerant configuration of TIBCO Enterprise Message Service.

Figure 1-1 Fault-tolerant configuration of TIBCO Enterprise Message Service



About locking

To prevent the backup server from assuming the role of the primary server, the primary server locks the shared state during normal operation. If the primary server fails, the lock is released, and the backup server can obtain the lock.

Note: With Veritas Storage Foundation Cluster File System, this is handled through the distributed lock manager that prevents bottlenecks and allows for concurrency and more rapid failover without requiring a hand-off.

About configuration files

When a primary server fails, its backup server assumes the status of the primary server and resumes operation. Before becoming the new primary server, the backup server re-reads all of its configuration files. If the two servers share configuration files, then administrative changes to the old primary carry over to the new primary.

About failover

This section presents details of the failover sequence.

Detection

A backup server detects a failure of the primary in either of two ways:

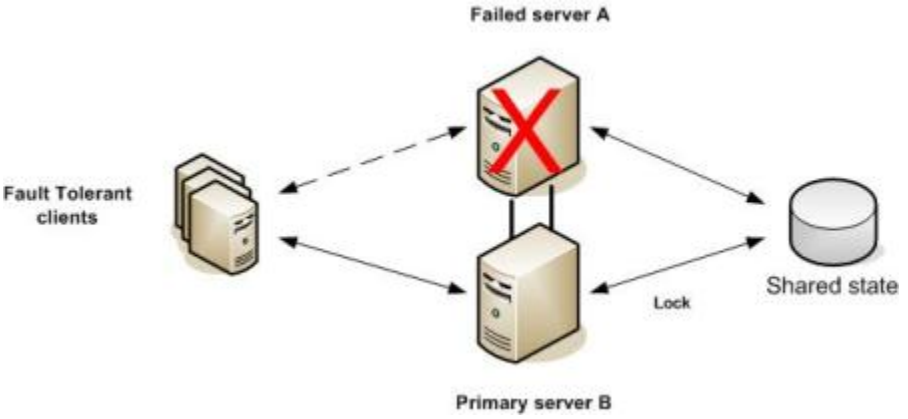
Heartbeat Failure	The primary server sends heartbeat messages to the backup server to indicate that it is still operating. When a network failure stops the servers from communicating with each other, the backup server detects the interruption in the steady stream of heartbeats. For details, see Heartbeat Parameters.
Connection Failure	The backup server can detect the failure of its TCP connection with the primary server. When the primary process terminates unexpectedly, the backup server detects the broken connection.

Response

When a backup server (B) detects the failure of the primary server (A), then the backup server attempts to assume the role of the primary server. First, the backup server obtains the lock on the current shared state. When it can access this information, it becomes the new primary server.

Figure 1-2 illustrates a failed primary server.

Figure 1-2 Failed primary server

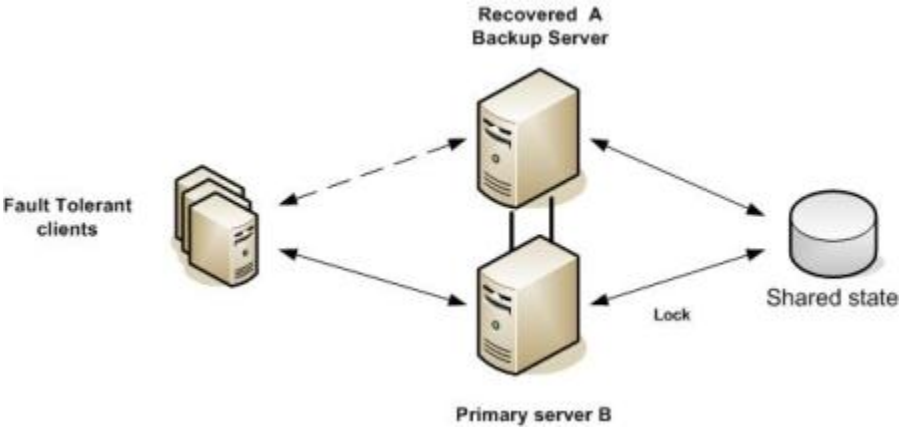


Role reversal

When the backup server becomes the new primary server, the old primary server (A) can restart as a backup server, so that the two servers exchange roles.

Figure 1-3 illustrates the change in role of a server from a recovered server to a backup server.

Figure 1-3 Recovered server becomes backup server



Client transfer

Clients of the primary server (A) that are configured to fail over to the backup server (B) automatically transfer to the backup server (B) when it becomes the new primary server. The backup server (B) reads the client's current state from the shared storage to deliver any persistent messages to the client.

Shared state

The primary server and the backup server must share the same state.

The server state includes the following categories of information:

- Persistent message data (for queues and topics)
- Client connections of the primary server
- Metadata about message delivery

During a failover, the backup server re-reads all shared state information.

Implementing shared state

We recommend that you implement shared state using shared storage devices. The shared state must be accessible to both the primary and backup servers.

TIBCO's support criteria

Several options are available for implementing shared storage using a combination of hardware and software.

Warning: Always consult your shared storage vendor and your operating system vendor to ascertain that the storage solution you select satisfies all the criteria.

Table 1-1 describes the criteria that needs to be fulfilled by your storage solution.

Table 1-1 Shared storage criteria for fault tolerance

Criterion	Description
Write Order	The storage solution must write data blocks to shared storage in the same order as they occur in the data buffer. Solutions that write data blocks in any other order (for example, to enhance disk efficiency) do not satisfy this requirement.
Synchronous Write Persistence	Upon return from a synchronous write call, the storage solution guarantees that all the data have been written to durable, persistent storage.
Distributed File Locking	The TIBCO Enterprise Message Service servers must be able to request and obtain an exclusive lock on the shared storage. The storage solution must not assign the locks to two servers simultaneously. See “Software options” on page 13. TIBCO Enterprise Message Service servers use this lock to determine the primary server.
Unique Write Ownership	The TIBCO Enterprise Message Service server process that has the file lock must be the only server process that can write to the file. Once the system transfers the lock to another server, pending writes queued by the previous owner must fail.

TIBCO's hardware options

Some of the commonly-sold hardware options for shared storage are dual-port SCSI devices, Storage Area Network (SAN), and Network Attached Storage (NAS).

About SCSI and SAN options

Dual-port SCSI and SAN solutions generally satisfy the "Write Order" and "Synchronous Write Persistence" criteria. The clustering software must satisfy the remaining two criteria. As always, you must confirm all the listed requirements with your vendors.

About NAS option

NAS solutions require a Cluster Server (rather than a CFS) to satisfy the "Distributed File Locking" criterion. Some NAS solutions satisfy the criteria, and some do not; you must confirm all the listed requirements with your vendors.

NAS with NFS

When NAS hardware uses NFS as its file system, it is particularly difficult to determine whether the solution meets the criteria.

Our research indicates the following conclusions:

- NFS v2 definitely does not satisfy the criteria.
- NFS v3 with UDP definitely does not satisfy the criteria.
- NFS v3 with TCP might satisfy the criteria.
  Consult with the NAS vendor to verify that the NFS server (in the NAS) satisfies the criteria. Consult with the operating system vendor to verify that the NFS client (in the operating system on the server host computer) satisfies the criteria. When both vendors certify that their components cooperate to guarantee the criteria, then the shared storage solution supports TIBCO Enterprise Message Service.

For more information on how the TIBCO Enterprise Message Service locks shared store files, see the section "How Enterprise Message Service Manages Access to Shared Store Files".

Software options

Consider the following examples of commonly-sold software options:

Cluster Server (CS)	A cluster server monitors the TIBCO Enterprise Message Service server processes and their host computers, and ensures that exactly one server process is running at all times. If the primary server fails, the CS restarts it; if it fails to restart the primary, it starts the backup server instead.
Clustered File System (CFS)	A clustered file system lets the two TIBCO Enterprise Message Service server processes run simultaneously. It even lets both servers mount the shared file system simultaneously. However, the CFS assigns the lock to only one server process at a time. The CFS also manages operating system caching of file data, so the backup server has an up-to-date view of the file system (instead of a stale cache).

With dual-port SCSI or SAN hardware, either a CS or a CFS software might satisfy the "Distributed File Locking" criterion. With NAS hardware, only a CS software can satisfy this criterion (CFS software generally does not). Of course, you must confirm all the listed requirements with your vendors.

Note: Influencing this decision will be critical to determining whether CFS can win or whether a Celerra/NetApp solution will be the vehicle.

About messages stored in shared state

Messages with "persistent" delivery mode are stored and are available in the event of primary server failure. Messages with "non_persistent" delivery mode are not available if the primary server fails. For more information about recovery of messages during failover, see the section "Message Redelivery".

About shared state files

Table 1-2 lists the files created by the tibemsd server to store shared state.

Table 1-2 Shared state files

File name	Description
meta.db	This file records durable subscribers, fault-tolerant connections, and other metadata.
sync-msgs.db	When a queue or topic definition (in a configuration file) specifies that the destination is failsafe, then the server stores its messages in this file (using synchronous I/O calls).

Table 1-2 Shared state files *(continued)*

File name	Description
async-msgs.db	When a queue or topic definition (in a configuration file) does not specify that the destination is failsafe, then the server stores its messages in this file (using asynchronous I/O calls).

Using CFS and VCS to create a fault tolerant TIBCO setup

The key to minimal queue downtime in a TIBCO Enterprise Message Service environment is to have the data store available on the standby node as soon as possible. This allows the standby node to recover the messages more quickly. This can be done either through network or shared storage, but network-attached storage options such as NFS are not always an option since they do not conform to the data integrity demands of TIBCO Enterprise Message Service.

Research concludes that NFSv2, NFSv3 (UDP) is not an option; NFSv3 with TCP might be an option, but not guaranteed; however, CFS fully complies with the data integrity demands of TIBCO Enterprise Message Service.

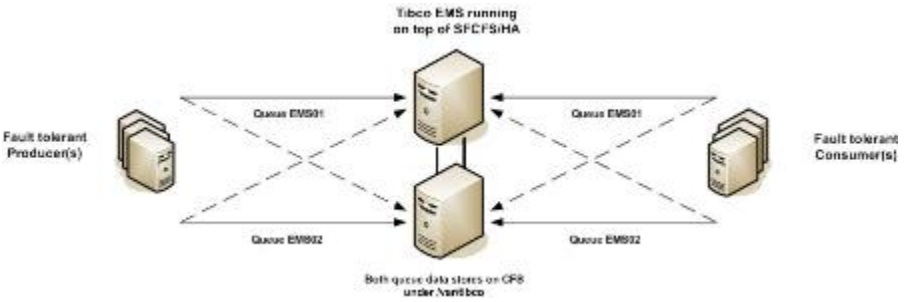
Using a clustered file system to share the data store between the TIBCO Enterprise Message Service servers, the standby server allows for a recovery operation as soon as it detects that the primary server is offline and not responding to heartbeats (and it can acquire the file locks). The heartbeat interval is configurable. By default, TIBCO sends a heartbeat every third second and the standby will initiate the recovery process when two heartbeats have gone un-acknowledged.

With default parameters for Storage Foundation Cluster File System and TIBCO Enterprise Message Service, the recovery operation is initiated in less than 10 seconds. Compared with traditional active/passive environments, where the failover time is counted in minutes, the value of Storage Foundation Cluster File System is clear.

If the primary server failure is recoverable (if it is a software error or a transient error), VCS will automatically restart the TIBCO Enterprise Message Service server to restore a fault tolerant state as soon as possible.

Figure 1-4 shows a two-node configuration with one queue, but this can be expanded into multiple servers and queues by adding resources.

Figure 1-4 CFS fault-tolerant configuration



Creating cluster file systems

To support the TIBCO Enterprise Message Service installation, the following cluster file systems are used:

/opt/tibco	Used to host the TIBCO Enterprise Message Service binaries.
/var/tibco	Used to host all data stores and configuration files.

Since the availability of the configuration depends on these file systems, it is important to ensure that they are mirrored, preferably, between different storage arrays to ensure maximum availability.

Installing TIBCO Enterprise Message Service

The TIBCO Enterprise Message Service binaries can be stored on the local disk of every node or on a shared cluster file system. Each method has its advantages.

Using local storage allows the administrator to easily upgrade one node at a time, but increases the administrative burden by demanding that multiple binary trees are kept in sync.

See “Installing TIBCO Enterprise Message Service on local storage” on page 17.

Using a clustered file system allows the administrator to have a single copy of the binaries and simplifies day-to-day administration at the expense of a slightly more complex process for upgrading TIBCO.

See “Installing TIBCO Enterprise Message Service on a shared file system” on page 17.

Installing TIBCO Enterprise Message Service on local storage

To install TIBCO Enterprise Message Service on local storage, the administrator should follow the instructions in the TIBCO installation and user's guide. No additional steps need to be taken.

Installing TIBCO Enterprise Message Service on a shared file system

To install a single, shared TIBCO Enterprise Message Service binary tree, the administrator must first create a cluster file system and ensure that it is mounted in the correct place. The default location is /opt/tibco.

See "Creating cluster file systems" on page 16.

It is also suggested that the administrator take time to manually specify where the installation stores the TIBCO installation properties and history files. These files must preferrably be placed on the shared file system together with the binaries.

Configuring TIBCO Enterprise Message Service for fault tolerance

The various "ft_*" parameters in the tibemsd.conf file dictate how fault tolerance is set up. To enable fault tolerance, at a minimum, the server, store, and ft_activate parameters need to be set. The server parameter is set to an arbitrary name, the name of the TIBCO Enterprise Message Service server, and should be identical in the configuration file for both the primary server and the standby server. The shared data store is designated by the store argument and needs to point to a shared or clustered file system. The ft_activate parameter should be set to point to the "other" server in a primary/standby pair, such that on the primary it points to the standby and on the standby it points to the primary.

The servers then communicate with each other on startup and agree on who is the current primary and who is the current standby server. Typically, the first node that is started is the primary server.

In our example configuration the relevant sections look like this:

```
tibemsd_tibems01.conf:
  server                  = EMS01
  store                   = /var/tibco/ems01/datastore
  listen                  = tcp://tibemsd01.veritas.com:55200
  ft_active               = tcp://tibems02.veritas.com.com:55200

tibemsd_tibems02.conf:
  server                  = EMS01
```

```
store                    = /var/tibco/ems01/datastore
listen                   = tcp://tibemsd02.veritas.com:55200
ft_active                = tcp://tibems01.veritas.com.com:55200
```

For an in-depth explanation of the remaining TIBCO Enterprise Message Service fault tolerance parameters, it is recommended that the administrator read the TIBCO Enterprise Message Service User's Guide, Chapter 15: Configuring Fault-Tolerant Servers.

Configuring TIBCO Enterprise Message Service producers and consumers for fault tolerance

To allow TIBCO clients (either commercial or home-grown applications) to benefit from the fault tolerant configuration, they must be configured to recognize the standby server as well as the primary server. This is done by specifying multiple server URLs when the client/application is started.

If the original server definition was "tcp://tibems01:55200", the corresponding fault tolerant server definition is "tcp://tibems01:55200, tcp://tibems02:55200". This allows the client to automatically recover and reconnect once it detects a server failure.

For more information on how to configure and control the client behaviors, read the TIBCO Enterprise Message Service User's Guide.

Configuring VCS resources and service groups

To cluster TIBCO Enterprise Message Service, it is recommended that two different service groups be used for the infrastructure resources along with one additional service group for each TIBCO Enterprise Message Service server instance (tibemsd daemon).

The sample configuration in this document uses the following service group setup:

cvm	This service group is a standard service group that controls the Veritas Cluster Volume Manager and Cluster File System shared resources. This group is automatically created during the configuration phase of Storage Foundation Cluster File System installation.
tibco_cfs	This service group controls access to the clustered TIBCO file systems. It depends on the cvm group.
tibco_ems01	This service group contains the resource that controls the tibemsd daemon for the TIBCO Enterprise Message Service instance "EMS01". It depends on the availability of the tibco_cfs group and the file systems within.

The service groups are all configured as parallel service groups (running on more than one node at a time).

The cvm and tibco_cfs groups run on all nodes within the cluster and the tibco_ems01 group runs on the primary or standby server dedicated to the TIBCO Enterprise Message Service instance it controls.

Figure 1-5 shows the three service groups online.

Figure 1-5 Service groups online

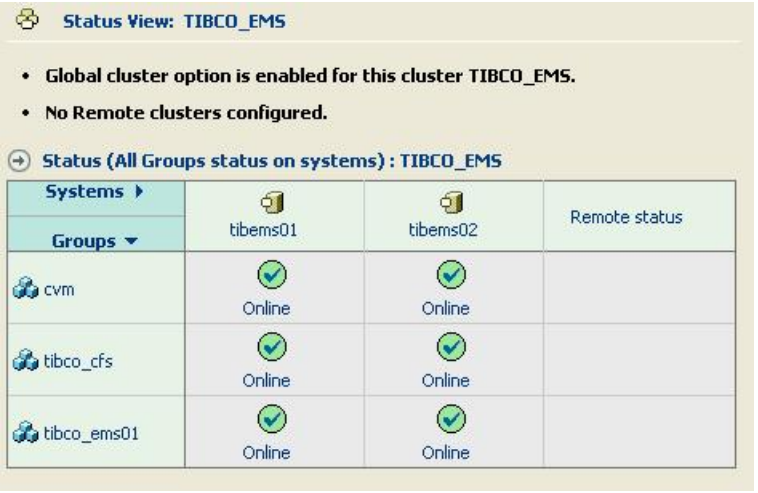


Figure 1-6 shows the service group dependencies.

Figure 1-6 Service group dependencies

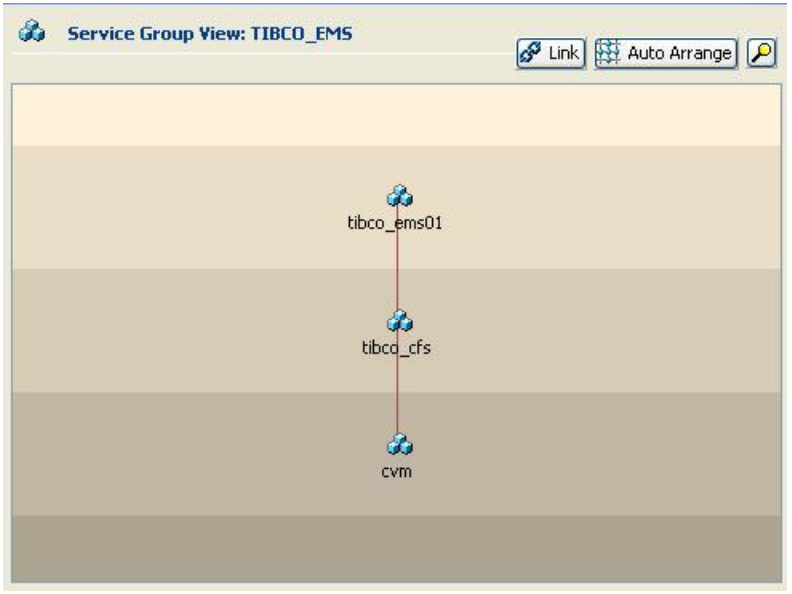


Figure 1-7 shows the default setup of the CVM service group.

Figure 1-7 CVM service group (default setup)

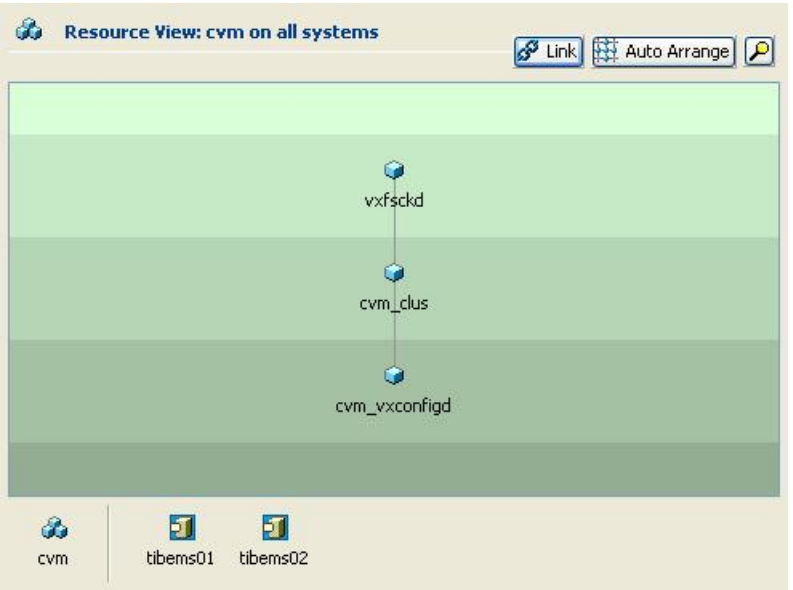
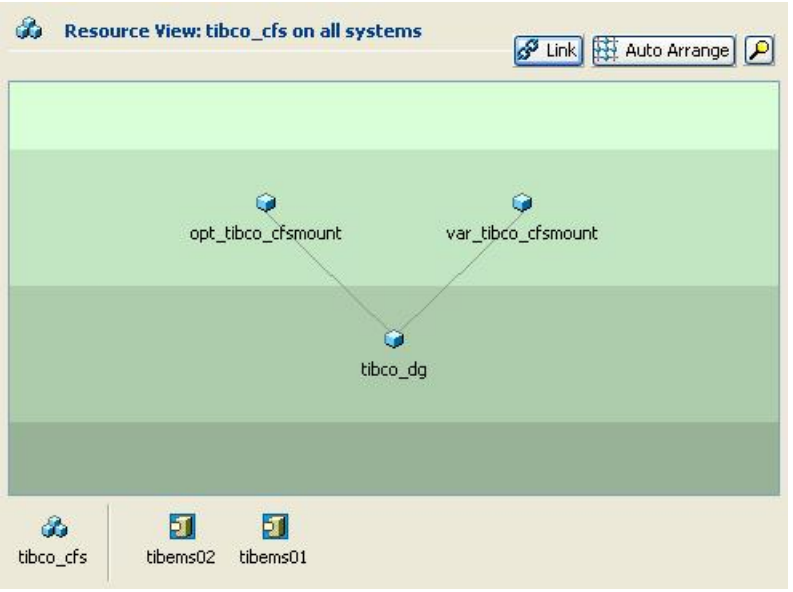


Figure 1-8 shows the tibco_cfs service group, which hosts the TIBCO file systems.

Figure 1-8 tibco_cfs service group (which hosts the TIBCO file systems)



Configuring VCS to start, stop, or monitor a TIBCO Enterprise Message Service server

VCS can be used to automatically start, stop, and monitor the TIBCO Enterprise Message Service servers within the cluster. To do this, a service group for each TIBCO Enterprise Message Service server instance is created (in the example, EMS01) and a single TibcoEMS agent is used to monitor the TIBCO Enterprise Message Service daemon (tibemsd).

The service group is configured as a parallel group on the primary and the standby server for the instance.

Table 1-3 lists the arguments and their description.

Table 1-3 Attributes and their description

Argument	Value	Description
User	root	UNIX user name that the TibcoEMS agent uses to execute the programs for managing a Tibco Enterprise Message Server.
EMSHomeDir	/var/tibco/ems01/bin/	Full path of directory in which the tibemsd binary file and tibemsadmin utility is located.
ConfigFile	@tibems01= "/var/tibco/ems01/bin/tibemsd_tibems01.conf" @tibems02= "/var/tibco/ems01/bin/tibemsd_tibems02.conf"	Full path and file name of the main configuration file *tibemsd.conf* for the Tibco Enterprise Message Server.
TibEmsServerUrl	@tibems01= "tcp://tibems01.veritas.com:55200" @tibems02= "tcp://tibems02.veritas.com:55200"	Tibco EMS server URL. During offline and second-level monitoring, this URL is used to specify the -server parameter for tibemsadmin utility.
SecondLevelMonitor	1	Used to enable second-level monitoring and specify the frequency. Second-level monitoring is a deeper, more thorough state check of the configured TibcoEMS instance.
TibUser	admin	Tibco user name to connect to server. During offline and second-level monitoring, this TibUser is used to specify the -user parameter for tibemsadmin utility.
TibPassword	*hvnTkvK*	Password of the Tibco operator, connecting to EMS server. During offline and second-level monitoring, this Password is used to specify the -password parameter for tibemsadmin utility.

Table 1-3 Attributes and their description *(continued)*

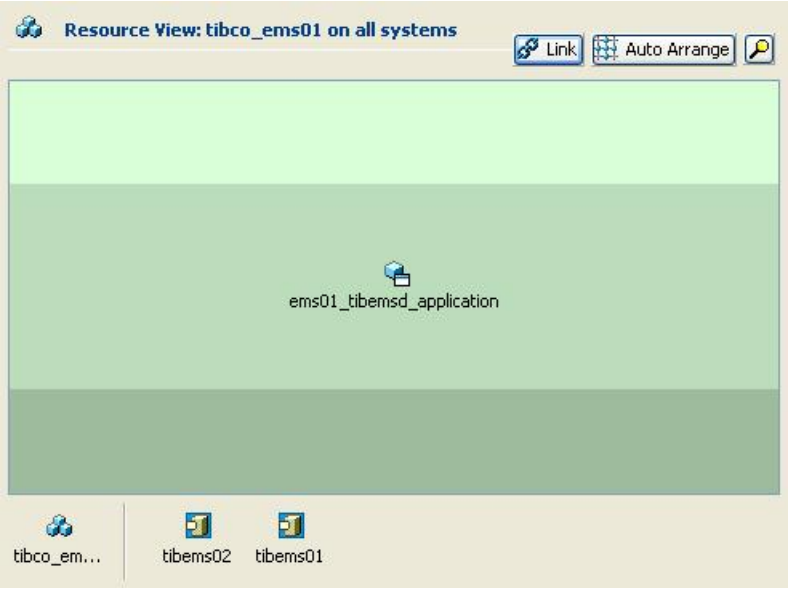
Argument	Value	Description
StartOptions	@tibems01= "-ft_active tcp://tibems02.veritas.com.com:55200" @tibems02= "-ft_active tcp://tibems01.veritas.com.com:55200"	List of tibemsd startup parameters. During startup of Tibco Enterprise Message Server, these startup parameters are passed to tibemsd command.

It is also recommended that the RestartLimit attribute of the TibcoEMS agent be changed to 1 (from default 0). This allows VCS to enable the TIBCO Enterprise Message Service server to restart if it crashes.

For more details on TibcoEMS agent attributes, refer to *Veritas High Availability Agent for Tibco EMS Server Installation and Configuration Guide*.

Figure 1-9 shows the TIBCO application group for EMS01.

Figure 1-9 TIBCO application group for EMS01



Starting the TIBCO Enterprise Message Service server using StartScript

The StartScript is used to start the TIBCO Enterprise Message Service server through VCS.

```
#!/bin/sh
cd /var/tibco/ems01/config
/opt/tibco/ems/bin/tibemsd -config tibemsd_`uname -n`.conf &
```

Sample VCS configuration file

Below is a sample VCS configuration file for a TIBCO Enterprise Message Service configuration with two nodes.

```
include "types.cf"
include "CFSTypes.cf"
include "CVMTypes.cf"

cluster TIBCO_EMS (
 UserNames = { admin = ElmElgLimHmmKumGlj }
 ClusterAddress = "127.0.0.1"
 Administrators = { admin }
 )

system tibems01 (
 )

system tibems02 (
 )

group cvm (
 SystemList = { tibems02 = 0, tibems01 = 1 }
 AutoFailOver = 0
 Parallel = 1
 AutoStartList = { tibems02, tibems01 }
 )

 CFSfsckd vxfsckd (
  )

 CVMCluster cvm_clus (
  CVMClustName = TIBCO_EMS
```

```
 CVMNodeId = { tibems01 = 0, tibems02 = 1 }
 CVMTransport = gab
 CVMTimeout = 200
 )

 CVMVxconfigd cvm_vxconfigd (
 Critical = 0
 CVMVxconfigdArgs = { syslog }
 )

 cvm_clus requires cvm_vxconfigd
 vxfsckd requires cvm_clus


 // resource dependency tree
 //
 // group cvm
 // {
 // CFSfsckd vxfsckd
 //     {
 //     CVMCluster cvm_clus
 //         {
 //         CVMVxconfigd cvm_vxconfigd
 //         }
 //     }
 // }


group tibco_cfs (
 SystemList = { tibems01 = 0, tibems02 = 1 }
 AutoFailOver = 0
 Parallel = 1
 AutoStartList = { tibems01, tibems02 }
 )

 CFSMount opt_tibco_cfsmount (
  MountPoint = "/opt/tibco"
  BlockDevice = "/dev/vx/dsk/tibcodg/opt_tibco"
  )

 CFSMount var_tibco_cfsmount (
  MountPoint = "/var/tibco"
  BlockDevice = "/dev/vx/dsk/tibcodg/var_tibco"
```

```
 )

 CVMVolDg tibco_dg (
  CVMDiskGroup = tibcodg
  CVMVolume = { var_tibco, opt_tibco }
  CVMActivation = sw
  )

 requires group cvm online local firm
 opt_tibco_cfsmount requires tibco_dg
 var_tibco_cfsmount requires tibco_dg


 // resource dependency tree
 //
 // group tibco_cfs
 // {
 // CFSMount opt_tibco_cfsmount
 //     {
 //     CVMVolDg tibco_dg
 //     }
 // CFSMount var_tibco_cfsmount
 //     {
 //     CVMVolDg tibco_dg
 //     }
 // }


group tibco_ems01 (
 SystemList = { tibems01 = 0, tibems02 = 1 }
 AutoFailOver = 0
 Parallel = 1
 AutoStartList = { tibems01, tibems02 }
 )

TibcoEMS ems01_tibemsd_application (
  User = root
  EMSHomeDir = "/var/tibco/ems01/bin/"
  ConfigFile @tibems01 = "/var/tibco/ems01/bin/tibemsd_tibems01.conf"
  ConfigFile @tibems02 = "/var/tibco/ems01/bin/tibemsd_tibems02.conf"
  TibEmsServerUrl @tibems01 = "tcp://tibemsd01.veritas.com:55200"
  TibEmsServerUrl @tibems02 = "tcp://tibemsd02.veritas.com:55200"
  SecondLevelMonitor = 1
```

```
 TibUser = admin
 TibPassword = hvnTkvK
 StartOptions @tibems01 = "-ft_active tcp://
 tibems02.veritas.com.com:55200"
 StartOptions @tibems02 = "-ft_active tcp://
 tibems01.veritas.com.com:55200"
 )

requires group tibco_cfs online local firm


// resource dependency tree
//
// group tibco_ems01
// {
// TibcoEMS ems01_tibemsd_application
// }
```

Sample TIBCO Enterprise Message Service configuration file

Below is a sample TIBCO Enterprise Message Service configuration file. The file is used on tibems01; the corresponding file for tibems02 has different values where the hostname is used.

```
#########################################################################
#       Copyright (c) 2001-2003 TIBCO Software Inc.
#       All Rights Reserved.
#       For more information, please contact:
#       TIBCO Software Inc.
#       Palo Alto, California, USA
#
#       Server Configuration Properties
#########################################################################

#########################################################################
# All parameters follow 'name = value' syntax. Each parameter must
# be in one line, line breaks not supported. If value is empty then
# parameter is ignored. It is a good idea to keep all parameters listed
# using empty value when no value is required.
# Lines starting with # or ; as well as empty lines are ignored.
```

```
#########################################################################

#########################################################################
# Server Identification Information.
#
# server:    unique server name
# password:  password used to login into other routed server
#########################################################################

server                  = EMS01
password                =

#########################################################################
# Configuration files.
#########################################################################

users                   = /var/tibco/ems01/config/users.conf
groups                  = /var/tibco/ems01/config/groups.conf
topics                  = /var/tibco/ems01/config/topics.conf
queues                  = /var/tibco/ems01/config/queues.conf
acl_list                = /var/tibco/ems01/config/acl.conf
factories               = /var/tibco/ems01/config/factories.conf
routes                  = /var/tibco/ems01/config/routes.conf
bridges                 = /var/tibco/ems01/config/bridges.conf
transports              = /var/tibco/ems01/config/transports.conf
tibrvcm                 = /var/tibco/ems01/config/tibrvcm.conf
durables                = /var/tibco/ems01/config/durables.conf

#########################################################################
# Persistent Storage.
#
# store:          directory to store persistent messages.
# store_minimum:  pre-allocated space, the store file will not go below.
# store_crc:      verify crc checksum data when reading.
# store_truncate: should the server attempt to truncate the store files.
# store_minimum      = 500MB


#########################################################################

store                   = /var/tibco/ems01/datastore
store_minimum_sync      = 10GB
store_minimum_async     = 500MB
```

```
store_crc               = disabled
store_truncate          = disabled

################################################################################
# Maximum number of connections, 0 to set no limit
################################################################################

max_connections         = 0

################################################################################
# Maximum message memory, 0 to set no limit. Use KB, MB or GB form.
# msg_swapping enables or disables swapping messages on disk. Message
# swapping is similar to virtual memory, the server swaps out unneeded
# messages on disk to reduce memory footprint. However on machines
# with large amount of RAM it may be desirable to disable message
# swapping. You need extreme caution when changing it, in most practical
# cases it is required to be enabled and it is usually much faster than
# rely on the operating system memory swapping.
################################################################################

max_msg_memory   = 1080MB
msg_pool_size           = 5000000
msg_swapping            = enabled

################################################################################
# EMS 4.0.1 V8 HotFix Release Parameters 04-09-2004
################################################################################

#route_recover_interval = 10800
use_fsync_minimize_disk = 10000

################################################################################
# Listen ports. May be tcp or ssl, can specify any number.
# Form is tcp://hostname:port. If the hostname is not present then
# the default host and interface will be used.
################################################################################

listen                  = tcp://tibems01.veritas.com:55200

################################################################################
# Authorization. Disabled by default. Enable to verify user credentials
# and permissions on secure destinations.
################################################################################
```

```
authorization           = disabled

################################################################################
# Routing. Routes configuration is in 'routes.conf'. This enables or
# disables routing functionality for this server.
################################################################################

routing                 = disabled

################################################################################
# Producer flow control. Disabled by default. Set to "enabled"
# to enable for those destinations which have "flowControl"
# property defined.
################################################################################

flow_control            = enabled

################################################################################
# Enable Rendezvous transports defined in transports.conf.
# By default all transports are disabled.
# Set value to "enabled" to enable all transports.
################################################################################

tibrv_transports =

################################################################################
# Log file name and tracing parameters.
# By default the log file is disabled.
#
# Possible values of trace items are:
# INFO, WARNING, ACL, LIMITS, SSL, SSL_DEBUG, ROUTE, ROUTE_DEBUG,
# ADMIN, RVADV, CONNECT, CONNECT_ERROR, PRODCONS, DEST, AUTH, MSG,
# " FLOW, LDAP_DEBUG.
# Special name 'DEFAULT' means default set of trace items
# which consists of:
# INFO, WARNING, ACL, LIMITS, ROUTE, ADMIN, RVADV, CONNECT_ERROR.
# Example:
#    console_trace = DEFAULT,-LIMITS,-ACL,+SSL,+ROUTE,+ROUTE_DEBUG
# This will set default, remove "LIMITS" and "ACL" and add SSL, ROUTE,
# and ROUTE_DEBUG.
# logfile_max_size specifies maximum size of the log file before
# it is rotated.
```

```
#########################################################################

logfile                 = /var/tibco/tibems01.log
logfile_max_size        = 10MB
log_trace               = DEFAULT
console_trace           = WARNING

#########################################################################
# Statistics:
#
# server_rate_interval is in seconds and can not be 0.
# statistics is disabled by default, set to 'enabled' to enable.
# rate_interval is in seconds, 0 to disable.
# detailed_statistics can be empty or NONE, or any combination of
# PRODUCER, CONSUMER and ROUTES without spaces between the keywords.
# statistics_cleanup_interval is in seconds, 0 to disable.
# max_stat_memory is in form nnKB, nnMB or nnGB, 0 to set no limit.
#########################################################################
server_rate_interval        = 3

statistics                  = enabled
rate_interval               = 3
detailed_statistics         = NONE
statistics_cleanup_interval = 3
max_stat_memory             = 64MB

#########################################################################
# Message tracking by message ID and correlation ID.
# Both are disabled by default and should be enabled if required.
#########################################################################

track_message_ids       =
track_correlation_ids   =

#########################################################################
# Fault-tolerant setup.
#########################################################################

ft_active               =  tcp://tibems02.veritas.com.com:55200
#ft_heartbeat            =
#ft_activation           =
#ft_reconnect_timeout    =
```

```
# SSL setup for connection to another FT server. Only required if
# the FT connection has to be SSL.

#ft_ssl_identity         =
#ft_ssl_issuer           =
#ft_ssl_private_key      =
#ft_ssl_password         =
#ft_ssl_trusted          =
#ft_ssl_verify_host      =
#ft_ssl_verify_hostname  =
#ft_ssl_expected_hostname=
#ft_ssl_ciphers          =

################################################################################
# SSL Server Setup Information.
#
# These parameters define server-side certificate, private key, issuers
# of client certificates and miscellaneous parameters used by this EMS
# server when handling SSL connections from the clients and other EMS
# servers.
################################################################################

# specify Diffie-Hellman key size, valid values are 512, 768, 1024, 2048.
# Default is 1024. Not used for export grade cipher suites.

ssl_dh_size             =

# can be used to disable specific ciphers or change the
# priority order. This is used in the format accepted
# by OpenSSL, refer to OpenSSL documentation for more info.
# Example: ssl_ciphers = +RC4-MD5:+RC4-SHA

ssl_server_ciphers      =

# SSL renegotiation can be set for long-lived connections by specifying
# the maximum size of transferred bytes or as an interval. Can specify both.

# renegotiation size in bytes, minimum is 64Kb, set to 0 to disable.
# Can use KB, MB and GB notation.

ssl_renegotiate_size    =

# renegotiation interval in seconds minumum is 15, set to 0 to disable.
```

```
ssl_renegotiate_interval=

# The following is set if all clients must preset the certificate.
# If disabled the client may or may not have the certificate.

ssl_require_client_cert =

# This enforces the policy when the connection username is always
# extracted from the certificate, if the certificate was presented
# by the client. This does not affect users who have no certificate.

ssl_use_cert_username   =

# This specifies a special username, when presented as connection user
# name, the actual username is extracted from the client certificate
# if client has the certificate. This does not affect users who have
# no certificate.

ssl_cert_user_specname  = CERTIFICATE_USER

# Server certificate, key and private key password. If password not
# specified it is prompted for at start up time. The key and server
# certificate issuers may be included into specified PKCS12 file.
# Supports PEM, DER and PKCS12.

ssl_server_identity     =
ssl_server_key          =
ssl_password            =

# Server Issuer certificate(s).
# Supports PEM, DER and PKCS#12.
# This may be a part of PKCS12 specified by ssl_server_identity

ssl_server_issuer       =

# Trusted issuers of client certificates. Supports PEM, DER and PKCS7.

ssl_server_trusted      =

# Path to installed Enthropy Gathering Daemon

ssl_rand_egd            =
```

```
# File containing random data. If specified, used by the server
# at start up time.

ssl_rand_file           =
```