webMethods Integration Server Clustering Guide

Version 10.15

October 2022

WEBMETHODS

Document ID: IS-CL-1015-20221015

Table of Contents

About this Guide

This guide describes how to install and configure the webMethods Integration Server Clustering feature. It contains information for administrators who configure and manage a webMethods Integration Server and for application developers who want to create services that interact directly with the Integration Server short-term store.

Note:
This guide describes features and functionality that may or may not be available with your licensed version of webMethods Integration Server. For information about the licensed components for your installation, see the **Server > Licensing** page in the Integration Server Administrator.

To use this guide effectively, you should understand the basic concepts described in the *webMethods Integration Server Administrator's Guide* and *webMethods Service Development Help*. You should also be familiar with all the servers you want to include in the cluster.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at https://documentation.softwareag.com.

In addition, you can also access the cloud product documentation via https://www.softwareag.cloud. Navigate to the desired product and then, depending on your solution, go to "Developer Center", "User Center" or "Documentation".

Product Training

You can find helpful product training material on our Learning Portal at https://knowledge.softwareag.com.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at https://techcommunity.softwareag.com. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at https://github.com/softwareag and https://hub.docker.com/publishers/softwareag and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at https://empower.softwareag.com. Many services on this portal require that you have an account. If you do not yet have one, you can request it at https://empower.softwareag.com/register. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

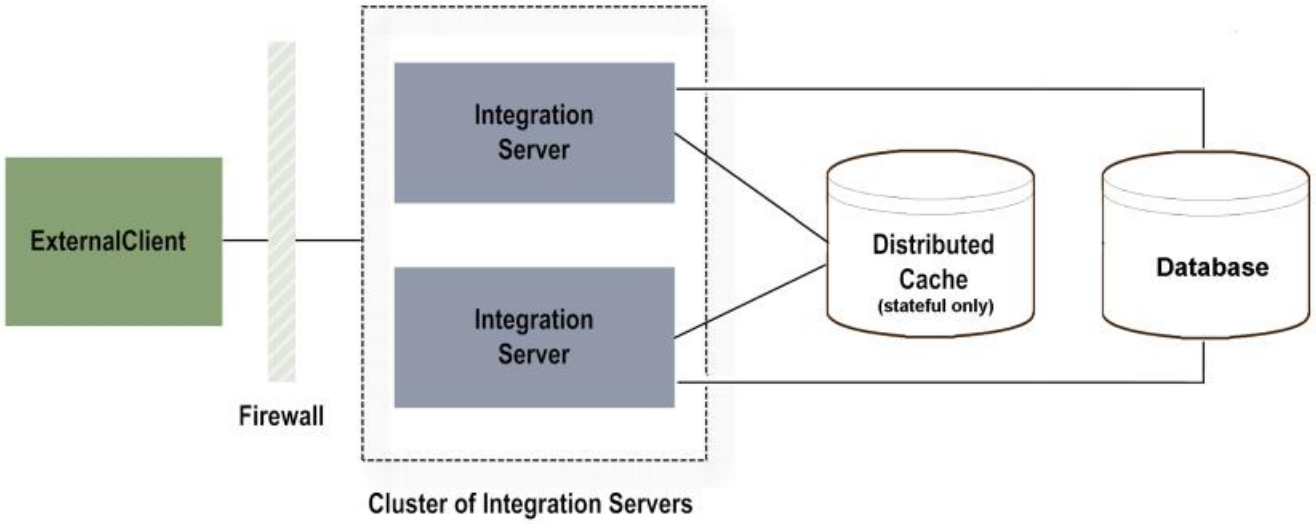
1 An Overview of Clustering

What Is Clustering?

Clustering is an advanced feature that substantially extends the reliability, availability, and scalability of Integration Server. Clustering accomplishes this by providing the infrastructure and tools to use multiple Integration Servers as if they were a single virtual server and to deliver applications that leverage that architecture.

- **Scalability**. Without clustering, only vertical scalability is possible. That is, increased capacity requirements can only be met by deploying on larger, more powerful machines, typically housing multiple CPUs. Clustering provides horizontal scalability, which allows virtually limitless expansion of capacity by simply adding more machines of the same or similar capacity.
- **Availability**. Without clustering, even with expensive fault-tolerant systems a failure of the system (hardware, Java runtime, or software) may result in unacceptable downtime. Clustering provides virtually uninterrupted availability by deploying applications on multiple Integration Servers; in the worst case, a server failure produces degraded but not disrupted service
- **Reliability**. Unlike a server farm (that is, an independent set of servers), clustering provides the reliability required for mission-critical applications. Distributed applications must address network, hardware, and software errors that might produce duplicate (or failed) transactions. Clustering makes it possible to deliver "exactly once" execution as well as checkpoint restart functionality for critical operations.

Clustering can be stateful or stateless, as described in later in this chapter. The diagram below shows stateful clustering in its simplest form.



Important:
Do not perform development work in a clustered environment. Basic namespace locking and local service development are not supported across clustered Integration Servers.

Other webMethods products support the use of clustering to synchronize requests, transactions, notifications, and so on across Integration Server instances in the cluster. For more information

about whether a particular product supports clustering and how to configure that product for a clustered environment, see the product documentation.

What Data Is Shared in a Cluster?

Data storage depends on whether the cluster is stateful or stateless.

In both types of clusters, Integration Server stores data from cluster nodes (for example, service results, scheduled tasks, audit data, and documents) in shared database components. For information about database components, see *Installing Software AG Products*.

In a stateful cluster, the session state for clients connected to cluster nodes is stored in a distributed cache in a Terracotta Server Array. The cache makes state available to all servers in the cluster, which enables multi-step transactions in a conversation to begin on one node in the cluster and continue on other nodes. For example, in the Cross-site Request Forgery (CSRF) Guard feature, a token is created locally on a server and saved in the distributed Terracotta Server Array cache, which makes the token available to all other cluster nodes.

In a stateless cluster, and the session state of a client is not stored in a distributed Terracotta Server Array cache; state is only available locally. Stateless clusters do not support all Integration Server-related features and components (see table below). Stateless clusters require less hardware and less software, and less work is needed for set up and maintenance. However, stateless cluster cannot support multi-step transactions on multiple cluster nodes because state is not available.

Integration Server Component or Feature	Supported by Stateless Cluster?
Cross-site request forgery (CSRF) Guard	No
Exactly once for document history	Yes
Failover support	No
File polling	Yes
Guaranteed delivery	Yes
OAuth authorization server	Yes
OpenID Connect	No
Scheduler	Yes, target node of Any or a specific server; all option not supported
Service results caching	Yes
Business Rules	Yes
CloudStreams	Yes
Dynamic Business Orchestrator	No

Integration Server Component or Feature	Supported by Stateless Cluster?
Process Engine and Monitor operation	Yes
Interaction between Process Engine and Trading Networks	Yes
Trading Networks	Yes
Adapters, including notifications produced by Integration Server scheduler services	Yes
1SYNC, AS4, ebXML, EDI, EDIINT, SWIFT eStandards Modules	Yes
Chem, FIX, HIPAA, Papinet, Rosettanet eStandards Modules	No

Load Balancing in a Stateful or Stateless Cluster

Load balancing is an optimizing feature you use with clustered Integration Servers. Load balancing controls how requests are distributed to the servers in the cluster. A load balancer can be useful if you need load balancing for multiple types of servers, for example, web servers and application servers, in addition to Integration Servers. Load balancers also offer virtual IP support, but they are not "out of the box." Most load balancers perform load balancing in a round-robin manner or based on network level metrics such as TCP connections and network response time.

You can always use load balancing with stateful clusters. You can use load balancing with stateless clusters if you are not using any of the unsupported features or components listed in the table above..

Failover Support for Stateful Clusters

Failover support enables recovery from system failures that occur during processing, making your applications more robust. For example, by having more than one Integration Server, you protect your application from failure in case one of the servers becomes unavailable. If the Integration Server to which the client is connected fails, the client automatically reconnects to another Integration Server in the cluster.

Note: Integration Server clustering provides failover capabilities to clients that implement the webMethods Context and TContext classes. Integration Server does not provide failover capabilities when a generic HTTP client, such as a web browser, is used.

You can use failover support with stateful clusters.

Reliability

The guaranteed delivery and checkpoint restart features improve reliability.

Guaranteed Delivery for Stateful or Stateless Clusters

Guaranteed delivery ensures that a service executes once and only once. It is particularly useful when used with clustering to prevent a restarted service from running on more than one server. This feature is only for use with server-to-server communications. You can use guaranteed delivery with both stateful and stateless clusters.

Guaranteed delivery ensures one-time execution of services by guaranteeing the following:

- Requests from the client to execute services are delivered to the server.
- Services are executed once, and only once.
- Responses from the execution of services are delivered to the client.

When not clustering, guaranteed delivery makes sure a client resubmits a service request to an Integration Server until it succeeds and a response is returned, and makes sure the service executes only once. For example, if the network connection between the client and the Integration Server fails after execution but before the response is successfully redirected to the client, the service might be executed twice. Guaranteed delivery prevents this from happening.

With clustering, guaranteed delivery makes sure that if the server on which the service is running becomes unavailable, the client retries the service on another server in the cluster until it succeeds and a response is returned. As in an unclustered environment, guaranteed delivery prevents a service from executing more than once.

For more information about guaranteed delivery, see the section *Configuring the Server for Guaranteed Delivery* in the *webMethods Integration Server Administrator's Guide*. and the *Guaranteed Delivery Developer's Guide*.

Exactly Once for Document History

Exactly-once processing is a facility that ensures one-time processing of a guaranteed document by a webMethods messaging trigger; the trigger does not process duplicates of the document. You can use exactly-once document processing with both stateful and stateless clusters.

Checkpoint Restart for Stateful and Stateless Clusters

You can use the pub.storage or pub.cache service to code flow services to store state information and other pertinent information in the short-term data store. If a flow service fails because a server becomes unavailable, the flow service can be restarted from the last checkpoint rather than at the beginning. For more information about pub.storage and pub.cache services, see the *webMethods Integration Server Built-In Services Reference*. You can use both checkpoint restart services with stateful clusters. For stateless clusters, use the pub.storage service to use checkpoint restart.

Putting It All Together

This table summarizes how clustering, guaranteed delivery, and checkpoint restart work together to provide availability, failover, and reliability in different situations.

Checkpoint restart specified	Clustering specified	Guaranteed Delivery specified	If the server on which the service is running becomes unavailable	Point at which the service restarts
			After the server becomes available, you must manually restart the service.	At the beginning
		X	After the server becomes available, the client automatically restarts the service. The client keeps trying to run the service until it runs successfully.	At the beginning
	X		The client automatically retries the service on another server in the cluster. If that server fails, the client retries the service on the next server in the cluster, and so on. If all attempts fail, you must manually restart the service.	At the beginning
	X	X	The client retries the service on the next server in the cluster. If that server fails, the client retries the service on the next server in the cluster, and so on. The client continues to try running the service until it runs successfully.	At the beginning
X			When the server becomes available again, you must manually restart the service.	Flow services: At the specified checkpoint Other services: At the beginning
X		X	When the server becomes available again, the client automatically retries the service. The client continues trying to run the service until it completes successfully.	Flow services: At the specified checkpoint Other services: At the beginning
X	X		The client automatically retries the service on the next server in the cluster. If that server fails, the client retries the service on the next server in the cluster, and so on. If attempts on all servers fail,	Flow services: At the specified checkpoint Other services: At the beginning

Checkpoint restart specified	Clustering specified	Guaranteed Delivery specified	If the server on which the service is running becomes unavailable	Point at which the service restarts
			you must restart the service manually.	
X	X	X	The client automatically retries the service on the next server in the cluster. If that server fails, the client retries the service on the next server in the cluster, and so on. The client keeps trying until the service runs successfully.	Flow services: At the specified checkpoint Other services: At the beginning.

Integration Server Session Objects for Stateful Clusters

Session objects are maintained for stateful clusters.

Clustered Integration Servers create and maintain the session objects that are stored in the distributed Terracotta Server Array cache.

In a non-clustered environment, Integration Server maintains session information in its own local memory. In a cluster, however, Integration Server creates a session in the distributed (or shared) cache, so that when a load balancer redirects a client for failover, the new server can access the session information.

The Integration Server that initially receives a request from a client creates the session object in the cache. Other Integration Servers in the cluster can access the session object to access and update session information as necessary.

When you configure an Integration Server to use clustering, you specify a setting that indicates how long inactive session objects are maintained in the cache. Periodically, each Integration Server in the cluster checks the session objects in the cache to determine if any have expired, and if so, removes them.

Usage Notes for Session Storage

- Only objects that are serializable can be successfully stored in the session when Integration Server is running in a cluster. That is, those objects must implement the java.io.Serializable interface or one of the com.wm.util.coder interfaces such as com.wm.util.coder.IDataCodable. In a cluster, a session is serialized to the shared session store. When the session is restored from the shared store to an Integration Server, the complete state of the session, including any application data that had been saved to it, is available. If the session contains objects that are not serializable, those objects are converted into strings that hold the objects' class names. The actual state of those objects is lost.

  The requirement that these objects be serializable applies to the entire object graph, including the object placed into the session and every object it contains, no matter how deeply nested.

Although your production application can run in a cluster, you will be developing it on a stand-alone Integration Server (clustered development is not supported). It is important to be aware of the serializable requirement so that you do not encounter problems with your session data once you start to test in a cluster.

- The processing speed of a cluster is determined in large part by network I/O. Adding application data to the session state will increase the amount of I/O the cluster must perform, and make it operate more slowly. The addition of a single large or complex object to each session can have a noticeable effect on the overall throughput of a cluster.

  If you are concerned that saving application data to the session might impact the performance of your Integration Server cluster, consider other ways of saving this data. If it does not have to persist across server restarts and does not have to be shared throughout the cluster, a simple Java collection such as a Vector or HashMap might be appropriate. If the data needs to survive server restarts but does not have to be shared throughout the cluster, writing it to the local file system is an option. If the data needs to be shared throughout the cluster, consider saving it in a database.

- Even though sessions are created and maintained in a distributed cache, each Integration Server keeps a portion of the cache locally. The local cache stores information relating to the sessions that are active on Integration Server as well as a list of nodes in the cluster. If you anticipate that your Integration Server sessions will use a large portion of the cache, you should increase the **Maximum Elements In Memory** or **Maximum Off-Heap** settings for each Integration Server in the cluster. For information about changing these settings, see the section *Working with Caches* in the *webMethods Integration Server Administrator's Guide*.

Scheduled Jobs for Stateful or Stateless Clusters

In a stateful cluster, you can schedule jobs to run on one, any, or all Integration Servers in the cluster. For jobs to run in the cluster, the server must be enabled for clustering and existing jobs must be flagged to run in the cluster.

In a stateless cluster, you can schedule jobs to run on one or any Integration Servers in the cluster, but not all.

Integration Server stores information about scheduled jobs in the ISInternal database component.

For instructions, see the chapter about managing services in the section *About Services* in the *webMethods Integration Server Administrator's Guide*.

Client Applications

Server clustering is almost transparent to the client. A client can issue requests to a server that is in a clustered environment in the same way it issues a request to a server that is not in a clustered environment. Integration Server clustering provides failover capabilities only to HTTP-based webMethods clients, such as those clients built using the webMethods Context and TContext classes.

When a client connects to a server in a cluster by calling the Context or TContext class, the server returns information about the other servers in the cluster to the client. To improve failover capability, before your client calls Context or TContext, have your client issue the setRetryServer method

in that class to specify another server to try in case the first server the client tries to connect to is unavailable.

You can use setRetryServer in a stateful cluster, and in a stateless cluster if your application does not require features that are not supported by stateless clusters. If a request is not processed, the client can use this information to connect to another server in the cluster to have the request fulfilled. In a stateful cluster, the server returns a list of cluster members to the client, which can then automatically switch to other servers if the one it is connected to becomes unresponsive.

Note:
You can use the setAllowRedir method in the Context class on each client to specify whether the client should connect to other servers in the cluster after a connectivity failure.

No special processing is required in your clients.

Using remote Integration Servers in a Stateful or Stateless Cluster

An Integration Server can be configured to connect to a remote server for a number of reasons, including:

- Allow clients to run services on other Integration Servers using the pub.remote:invoke service and the pub.remote.gd:* services.
- Connect publisher and subscriber Integration Servers to each other for the purpose of package replication.
- Facilitate the process of presenting different certificates to different Integration Servers.

If you want to specify a backup server in case the remote server is not available, specify that backup server as the retry server on the remote server's alias definition.

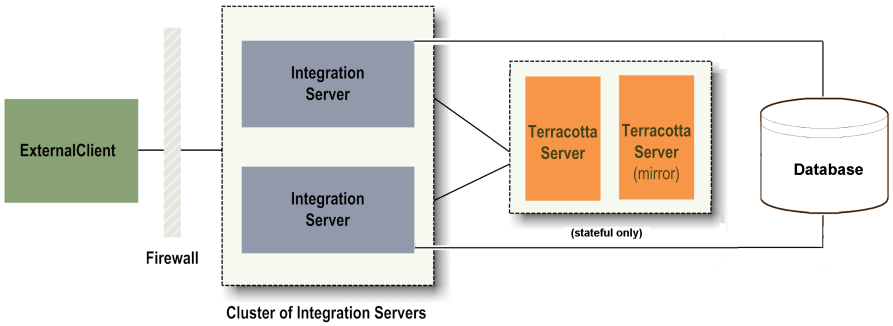
In a stateful cluster, use the `$clusterRetry` parameter that is passed to the pub.remote:invoke service to control what happens when an Integration Server tries to connect to the remote server in a stateful cluster. If you want the Integration Server to first try other servers in the cluster when the remote server is not available, and then go to the retry server if no cluster servers are available, set `$clusterRetry` to true. If you do not want the Integration Server to first try other servers in the cluster but rather to go directly to the retry server, set the parameter to false. For more information about remote servers, see the section *Setting Up Aliases for Remote Integration Servers* in the *webMethods Integration Server Administrator's Guide*. For more information about the pub.remote:invoke service, see the the section *pub.remote:invoke* in the *webMethods Integration Server Built-In Services Reference*.

Stateless clusters do not support the `$clusterRetry` parameter; in this scenario, the Integration Server will automatically connect to the retry server.

2 Installation and Setup

Using Terracotta as the Caching Software for a Stateful Cluster

The data that the Integration Servers share is stored in distributed caches on a Terracotta Server Array. A Terracotta Server Array is a group of one or more Terracotta Servers. The data associated with a cache is spread across the servers in the array (each server maintains a portion of the cache). You can mirror the servers in the array for high availability. For more information about Terracotta Server Arrays, see the Terracotta product documentation.



You must install and configure the Terracotta Server Array *before* you enable clustering on the Integration Servers. For the list of general steps that you must perform before you enable clustering, see "Setting up the Terracotta Server Array for Use by the Cluster of Integration Servers" on page 20.

Installing Terracotta Server Array for a Stateful Cluster

For a stateful cluster, install and configure the Terracotta Server Array **.** You must install and configure the Terracotta Server Array *before* you enable clustering on the Integration Servers. For more information about this step, see "Setting up the Terracotta Server Array for Use by the Cluster of Integration Servers" on page 20.

Setting up the Terracotta Server Array for Use by the Cluster of Integration Servers

Before you enable clustering using Terracotta, perform the following steps to ensure that the Terracotta Server Array is ready for use by the Integration Servers in the cluster.

Step	Description
1	Install the Terracotta Server Array if you have not done so already. For installation instructions, see *Installing Software AG Products*. **Note:** Before you install your Terracotta Server Array, you need to decide how many Terracotta Servers will make up the array and whether or not you will mirror those servers. To guide your decision, review the sections on high availability and

Step	Description
	architecture in the product documentation for the Terracotta Server Array or consult your Software AG representative.
2	Configure the tc-config.xml file on the Terracotta Server Array as described in the section on configuring the tc-config.xml file in the Terracotta Ehcache chapter of *webMethods Integration Server Administrator's Guide*. This section describes the parameter settings that are required when using a Terracotta Server Array with Integration Server.
3	Install your Terracotta license key on each Integration Server in the cluster. For procedures, see the section on installing and changing a Terracotta license in the Terracotta Ehcache chapter of *webMethods Integration Server Administrator's Guide*.

Note:
The minimum offheap (or memory) required for clustering in webMethods hybrid mode is 2GB. However, for non-hybrid deployments, distributed caching and other applications, refer to the Terracotta documentation for recommended configuration

Installing and Setting Up the Integration Servers in a Stateful or Stateless Cluster

Server Environment

All Integration Servers that you want to include in a cluster must be of the same release and at the same patch level. However, the Integration Servers can be on different operating systems; for example, one server on Windows, one on UNIX, and one on Linux.

Software AG recommends that you maintain the same environment for all servers in a cluster. For server clustering to work effectively, you should keep the following the same on all servers in the cluster:

- **Each server should have the same set of licensed capabilities.**
- **All servers should have the same packages of services.** For a service to execute on any server in the cluster, that service must exist in the same package on every server in the cluster.

  Important:
  Identically named MQTT triggers and MQTT connection aliases should *not* exist on each node in the cluster. Furthermore, MQTT connection aliases in a cluster should all use unique connection client IDs. No duplicate connection client IDs should be used. The cluster limitations are because the MQTT specification does not support multiple clients sharing a subscription. Identical MQTT triggers share the same subscriptions and the same connection client ID. The connection client ID of the MQTT trigger is the connection client ID value of the MQTT connection alias plus the trigger name.

  Note: Software AG recommends that you use webMethods Deployer or the package replication functionality in the Integration Server Administrator to copy packages to other

servers in the cluster, instead of using Designer to copy them. For information about webMethods Deployer, see the *webMethods Deployer User's Guide*. For information about package replication, see *webMethods Integration Server Administrator's Guide*.

- **Each server should have the same user accounts.** The server uses user account information to authenticate clients. When a server redirects a request to an alternate server, the alternate server re-authenticates the user using the credentials supplied to the original server. If authentication fails, the request fails.
- **Access to services should be the same on all servers.** Integration Server uses group information and ACLs to determine whether a client has access to a service. All Integration Servers should have the same groups with the same group membership. Services should be protected by the same ACLs. The ACLs should identify the same Allow Groups and Deny Groups.

  If a request is redirected to a server that denies access to the requested service, the request will fail.
- **Each server should have the same public caches.** One of the purposes of clustering is to allow a session to be directed to any server within a cluster and for the session to be processed the same way. Some of that processing may involve caching, in which case the same caches, with identical configurations, need to be on each server in the cluster.

  Note: Software AG recommends that you use webMethods Deployer to copy cache managers and caches to other servers in the cluster.
- **Each server should have the same event subscriptions.** Integration Server saves information for event types and event subscriptions in the eventcfg.bin file. This file is generated the first time you start an Integration Server and is located in the following directory: *Integration Server_directory* /instances/*instance_name*/config. Copy this file from one server to another to duplicate event subscriptions on all servers in the cluster.
- **Clock time must be the same on all servers.** The clocks on all machines in the Integration Server cluster must be synchronized for scheduled jobs to work properly in a cluster.
- **Each Integration Server should connect to the same messaging providers.** All servers should use the same messaging connection aliases to connect to the same messaging providers (Broker and/or Universal Messaging). Each messaging connection alias with the same name must use the same client prefix. Furthermore, each Broker connection alias must use the same client group.
- **Each server should connect to the same set of databases.** For example, if you are using the audit database to store audit data, all servers must connect to the same audit database. If you are using a back-end database for application-related data, all servers must connect to the same back-end database.
- **Each server should have its own diagnostic port.** If you plan to troubleshoot using the diagnostic port, you must configure a diagnostic port for each server in the cluster. The diagnostic port can access only the Integration Server on which it is defined. For more information about the diagnostic port, see the section *Adding an HTTP Diagnostic Port* in the *webMethods Integration Server Administrator's Guide*.

- **Each server should have its own Tspace.** If you want the Integration Servers in a cluster to temporarily store large documents in a hard disk drive space rather than keep them in memory, you must define a different Tspace for each Integration Server in a cluster.
- In a stateful cluster, **every server must point to the same** Terracotta Server Array **.** The Integration Servers must use the same Terracotta Server Array URLs and the same cluster name. Session timeout and time-to-live should be the same on all servers. If it is not the same, session lifetime will be unpredictable. For more information about using Terracotta with the cluster, see "Using Terracotta as the Caching Software for a Stateful Cluster" on page 20.

Adding a Server with an Embedded Database to a Cluster

For Integration Servers to participate in a cluster, they must share database components in an external RDBMS. If you installed an Integration Server with the embedded database, but now want to add it to a cluster, you must switch the Integration Server to use the shared external RDBMS.

To switch an Integration Server from an embedded database to a shared external RDBMS

1. In Integration Server Administrator, go to **Security > Certificates > Configure Client Certificates** and note the certificate mappings.

   Create JDBC connection pools for the IS Internal and Cross Reference database components, and point the ISInternal and Xref functions at the shared IS Internal and Cross Reference database components. For instructions, see *Installing Software AG Products*.

2. Run the migration utility pub.scheduler:migrateTasksToJDBC to migrate your scheduled tasks from the embedded database to the external RDBMS. See the the section *pub.scheduler:migrateTasksToJDBC* in the *webMethods Integration Server Built-In Services Reference*. for instructions.

   Note:
   This service migrates scheduled tasks only. Client mappings and run-time data will not be migrated.

3. In Integration Server Administrator, go to **Security > Certificates > Configure Client Certificates** and re-specify your certificate mappings. For instructions on mapping a client certificate to a user, refer to *webMethods Integration Server Administrator's Guide*.

Enabling Clustering for a Stateful Cluster

After you ensure that Integration Server has the appropriate environment, you can add it to the cluster by configuring the server to use clustering.

Keep the following points in mind when enabling clustering for an Integration Server.

- You must complete the setup steps described in "Setting up the Terracotta Server Array for Use by the Cluster of Integration Servers" on page 20 before you enable clustering on the Integration Servers.
- To be in the same cluster, Integration Servers must use the same Terracotta Server Array URLs and the same cluster name.
- An enterprise can have more than one cluster. To isolate multiple clusters on the same network, each cluster must have a different cluster name or different Terracotta Server Array or both.
- You must have webMethods Integration Server administrator privileges to enable clustering. If you do not have administrator privileges, have your webMethods Integration Server administrator perform this procedure.

To enable clustering

1. In Integration Server Administrator, go to **Settings > Clustering > Edit Cluster Settings > Enable Cluster** and specify the following:

For this parameter	Specify
Cluster Name	Name of the cluster to which this Integration Server belongs. Keep the following in mind when specifying the cluster name: ■ The cluster name cannot include any periods ".". Integration Server converts any periods in the name to underscores when you save the cluster configuration. ■ The cluster name cannot exceed 32 characters. If it does, Integration Server uses the first 20 characters of the supplied name and then a hash for the remaining characters. If you specify a name that does not already exist, Terracotta Server Array creates a new cache manager for each system cache manager on Integration Server. Terracotta Server Array appends the cluster name to the name of the system cache manager. System cache managers begin with `SoftwareAG`. For example, if you name the cluster "myCluster" the system cache manager `SoftwareAG.IS.Core` becomes `SoftwareAG.IS.Core.myCluster`. If you specify the name of a cluster that already exists and specify the same Terracotta Server Array URL used by the existing cluster name, the caching software adds this Integration Server to that cluster.
Session Timeout	Number of minutes an inactive session will be retained in the clustered session store. The default is 60. Set the clustered session timeout value to be longer than the session timeout value, which governs how long a server keeps session

For this parameter	Specify
	information in its memory. (Integration Server Administrator displays the session timeout on the **Settings > Resources** page.)
Action on Startup Error	How Integration Server responds when an error at start up prevents Integration Server from joining the cluster. Select one of the following

Option	Description
Start as Stand-Alone Integration Server	Integration Server starts as a stand-alone, unclustered Integration Server if it encounters any errors that prevent it from joining the cluster at start up. Integration Server continues to receive requests on inbound ports and serve requests. This is the default.
Shut Down Integration Server	Integration Server shuts down if it encounters any errors that prevent it from joining the cluster at start up.
Enter Quiesce Mode on Stand-Alone Integration Server	Integration Server starts as a stand-alone, unclustered Integration Server in quiesce mode if it encounters any errors that prevent it from joining the cluster at start up. When Integration Server is in quiesce mode, only the diagnostic port and quiesce port are enabled.

For more information about the **Action on Startup Error** options, see "About Action on Startup Error Options" on page 25.

Terracotta Server Array URLs	A comma separated list of the URLs for the Terracotta Server Array to be used with the cluster to which this Integration Server belongs. The URLs must be in the following format: *host*:*port*

2. Click **Save Settings** and then restart the Integration Server.

3. In Integration Server Administrator, go to **Settings > Clustering** and verify that all nodes in the cluster are displayed under **Cluster Hosts**.

About Action on Startup Error Options

When Integration Server that is configured for clustering starts up, errors may prevent Integration Server from connecting to the Terracotta Server Array. These errors can be caused by some of the following: a configuration issue with the Terracotta license file, an unavailable or incorrectly configured Terracotta Server Array, or an unavailable database.

When you configure clustering for an Integration Server, you can determine how Integration Server responds when the server starts and cannot connect to the Terracotta Server Array. The action Integration Server takes affects whether Integration Server is available to process requests, which Integration Server features are available, and how you might correct the errors. The **Action on Startup Error** parameter determines how Integration Server reacts when the Terracotta Server Array cannot be reached at startup. You can specify one of the following options for the **Action on Startup Error** parameter.

- **Start as Stand-Alone Integration Server**. If Integration Server encounters any errors that prevent it from joining the cluster at start up, Integration Server starts as a stand-alone, unclustered Integration Server. Integration Server continues to receive requests on inbound ports and serve requests. Integration Server does not use any distributed caches and instead uses local caches. You can use Integration Server to make changes needed to resolve the startup errors. When you restart Integration Server, Integration Server, will rejoin the cluster if Integration Server does not encounter any start up errors that prevent the connection to the Terracotta Server Array.

  Start as Stand-Alone Integration Server is the default option for the **Action on Startup Error** parameter.

- **Shut Down Integration Server**. If Integration Server encounters any errors that prevent it from joining the cluster at start up, Integration Server shuts down immediately. To make any changes to the clustering configuration, you must edit the configuration while running in safe mode. If you need to change the Terracotta Server Array URLs, you must start Integration Server in safe mode, change the values, and restart the Integration Server.

  For more information about starting Integration Server in safe mode, see the section *Starting the Integration Server in Safe Mode* in the *webMethods Integration Server Administrator's Guide*.

- **Enter Quiesce Mode on Stand-Alone Integration Server**. If Integration Server encounters any errors that prevent it from joining the cluster at start up, Integration Server starts as a stand-alone, unclustered Integration Server in quiesce mode. When Integration Server is in quiesce mode, only the diagnostic port and quiesce port are enabled. Integration Server will not receive requests on inbound ports. However, you can use Integration Server and Integration Server Administratorto make changes needed to correct the startup errors. After correcting the errors, exit quiesce mode. When exiting quiesce mode, Integration Server starts all the cache managers and joins the cluster.

  If Integration Server encounters any errors that prevent it from joining the cluster when exiting quiesce mode, Integration Server does not re-enter quiesce mode. Instead, Integration Server runs as an unclustered, stand alone server. Integration Server captures any errors that occurred while attempting to join the cluster in a report displayed in Integration Server Administrator when the server exits quiesce mode. Use the report to help correct the underlying configuration errors. Then, to rejoin the cluster, either restart Integration Server or enter and then exit quiesce mode.

  To use the **Enter Quiesce Mode on Stand-Alone Integration Server** option, a quiesce port must be configured for Integration Server. If a quiesce port is not configured and the **Enter Quiesce Mode on Stand-Alone Integration Server** option is selected, when Integration Server encounters any errors that prevent it from joining the cluster at start up, Integration Server shuts down instead of entering quiesce mode.

For more information about quiesce mode, see the section *What Happens when Integration Server Enters Quiesce Mode* in the *webMethods Integration Server Administrator's Guide*.

What Happens When an Integration Server in a Stateful Cluster Cannot Connect to the Distributed Cache?

- If Integration Server cannot connect to the distributed cache at the time the Integration Server initializes, Integration Server logs an error to the server log and takes the action associated with the selected option for the **Action on Startup Error** parameter. To enable clustering, diagnose and correct the problem. For more information about the **Action on Startup Error** parameter, see "About Action on Startup Error Options" on page 25.
- If Integration Server becomes disconnected from the distributed cache after initialization completes, the rejoin behavior instructs the system cache managers on the Integration Server to reconnect to the Terracotta Server Array automatically. All system cache managers are configured to rejoin. For more information about the rejoin behavior for distributed cache managers, see *webMethods Integration Server Administrator's Guide*.

Controlling Logging for Caching in a Stateful Cluster

Logging for cache activity on the Terracotta Server Array is controlled by Terracotta Ehcache. Terracotta Ehcache logging has a default configuration, but you can change it by modifying the .tc.dev.log4j.properties file. This file is located in the *Integration Server_directory* . Terracotta Ehcache creates additional log files on Integration Server when Integration Server connects to a Terracotta Server Array. You can specify where you want Terracotta Ehcache to put the log files by setting the watt.server.cachemanager.logsDirectory property on Integration Server. The default value for this property is *Integration Server_directory* /instances/*instance_name*/logs/tc-client-logs directory. For information about how Terracotta Ehcache logs cache manager activity when Integration Server connects to a Terracotta Server Array, see the section *Logging Cache Manager Activity in the Terracotta Server Array* in the *webMethods Integration Server Administrator's Guide*.

Scheduling Jobs to Run in a Stateful or Stateless Cluster

In a stateful cluster, you can schedule jobs to run on a target node of any server, a specific server, or all servers; for the jobs to run, the servers must be enabled for clustering.

In a stateless cluster, you can schedule jobs to run on a target node of any server or a specific server. Stateless clusters do not support running jobs on all servers.

For instructions about scheduling services, see the section *Scheduling a User Task* in the *webMethods Integration Server Administrator's Guide*.

Specifying Unique Logical Names for Integration Servers in a Stateful or Stateless Cluster

By default, Integration Server uses its host name to identify itself while scheduling tasks. However, when a cluster of Integration Servers are hosted on a single machine, the host name cannot uniquely

identify the individual Integration Servers. In such cases, use the watt.server.scheduler.logical.hostname property to specify a unique logical name for each Integration Server. The logical host name you specify cannot contain a semicolon (;). Set this property on each Integration Server instance in the cluster before scheduling any tasks.

To specify a unique name for an Integration Server in a cluster

1. In Integration Server Administrator, go to **Settings > Extended** and locate the watt.server.scheduler.logical.hostname parameter. If this parameter does not exist, add it. Set it as follows:

Set this parameter...	To specify...
`watt.server.scheduler. logical.hostname`	A unique logical name for Integration Server.

   For more information about using the **Extended** settings page to configure the Integration Server, see the section *Working with Extended Configuration Settings* in the *webMethods Integration Server Administrator's Guide*.

2. Click **Save Changes** and restart Integration Server.

Using Guaranteed Delivery with a Stateful or Stateless Cluster

When you use clustering, guaranteed delivery stores information about requests for all clustered servers in a shared, centrally located database. Because the information is stored centrally, guaranteed delivery uses a locking mechanism to synchronize updates to the database.

Attempts to update the database might time out. As a result, the server may require several attempts to complete a request. You can limit how long a cluster server sleeps between attempts to place an update lock on the database. In addition, you can set a maximum time limit after which the server overrides a lock on the database held by a non-responsive server in order to complete the update.

To configure cluster database locking

1. In Integration Server Administrator, go to **Settings > Extended** and locate the **watt.server.tx.cluster.lockTimeoutMillis** parameter. If this parameter does not exist, add it. Set it as follows:

Set this parameter:	To specify
`watt.server.tx.cluster. lockTimeoutMillis`	The number of milliseconds a cluster server waits between requests to place an update lock.

2. To set how long a cluster server will wait before overriding a lock, locate the **watt.server.tx.cluster.lockBreakSecs** parameter. If this parameter does not exist, add it. Set it as follows:

Set this parameter:	To specify
`watt.server.tx.cluster. lockBreakSecs`	The number of seconds a cluster server waits before overriding a lock.

For more information about using the **Extended** settings page to configure the Integration Server, see the section *Working with Extended Configuration Settings* in the *webMethods Integration Server Administrator's Guide*.

3. Click **Save Changes** and restart the Integration Server.

3 Managing Server Clustering

Overview

The sections in this chapter apply to stateful clusters only unless otherwise indicated.

Viewing the Servers in a Stateful Cluster

When you have server clustering enabled for a stateful cluster, you can display a list of all the servers in the cluster. The server learns of other servers in the cluster by retrieving information from the distributed Terracotta Server Array cache.

If you notice that a server is missing from the list, it might be for one of these reasons:

- The server is not running.
- The server cannot connect to the Terracotta Server Array.
- The clock time on the servers does not match.

To view a list of all clustered servers, in Integration Server Administrator, go to Settings > Clustering.

Removing a Server from a Stateful Cluster

If you no longer want a server to be part of a cluster, disable clustering on that server.

To remove a server from a stateful cluster

1. If you are using guaranteed delivery, take the server offline before removing it from the cluster. For instructions, see "Taking a Server Offline from a Stateful or Stateless Cluster" on page 33.

   Important:
   If you do not take the server offline before removing it from the cluster, the server will continue to receive transactions, but the transactions will not be integrated into the database and may be lost if the server is restored to the cluster.

2. In Integration Server Administrator, go to **Settings > Clustering > Edit Cluster Settings** and click **Disable Cluster**.

3. Click **Save Settings** and then restart the server.

   Note:
   Until you restart the server, it will remain in the list of cluster hosts on other Integration Servers in the cluster.

Adding a Server Back into a Stateful Cluster

You can add a server back into a cluster by re-enabling clustering on the server. When clustering was previously enabled, the server stored the configured settings in the server.cnf file in the *Integration Server_directory* /instances/*instance_name*/config directory.

Distributed caching configuration information is stored on the Terracotta Server Array. If you re-enable clustering and specify the same Terracotta Server Array URLS and the same cluster name, Integration Server retrieves the cache configuration information from the Terracotta Server Array.

When you re-enable clustering, ensure that the server has the same licensed capabilities as the rest of the servers in the cluster.

To add a server back into the cluster

1. In Integration Server Administrator, go to **Settings > Clustering > Edit Cluster Settings** and click **Enable Cluster**.
2. Change the settings to enable clustering.
3. Click **Save Settings** and then restart the server.

   Note:
   Until you restart the server, it will not appear in the list of cluster hosts on other Integration Servers in the cluster.

4. If you are using guaranteed delivery, the server you are returning to the cluster might be offline. Before you can use the server, bring it back online. For instructions, see "Bringing a Server Back Online to a Stateful or Stateless Cluster" on page 34.

Taking a Server Offline from a Stateful or Stateless Cluster

You can hide a server from the load balancer by assigning the server an unpublished port.

To take a server offline

1. In Integration Server Administrator, go to **Server > Ports > Change Primary Port**.
2. Under **Select New Primary Port**, select an unpublished port. If an unpublished port does not exist, create one.
3. Click **Update**.
4. On your browser, update the URL for the Integration Server Administrator to use the unpublished port you just selected.

5. Disable all other listening ports.

Bringing a Server Back Online to a Stateful or Stateless Cluster

To bring a server back online

1. In Integration Server Administrator, go to **Server > Ports > Change Primary Port**.
2. Under **Select New Primary Port**, select the published port that the load balancer is aware of for the server.
3. Click **Update**.
4. On your browser, update the URL for the Integration Server Administrator to use the published port you just selected.
5. Enable all disabled listening ports.

Increasing the Cache Size for a Stateful Cluster

If you think your Integration Server sessions will use a large portion of the distributed Terracotta Server Array cache, increase the cache size on each server in the cluster. You can increase the on-heap cache size by modifying the **Maximum Elements on Disk** setting (see "Increasing the Maximum Elements on Disk Size" on page 34). Or, if you are using BigMemory, you can modify the **Maximum Off-Heap** setting (see the Terracotta Ehcache chapter of the *webMethods Integration Server Administrator's Guide*).

Increasing the Maximum Elements on Disk Size

In a clustered environment, the **Maximum Elements on Disk** cache setting is non-dynamic. The procedure you use to modify the value depends on whether your Terracotta Server Array is configured to use a persistence mode of permanent-store or temporary-swap-only (the default). The following sections describe how to increase the **Maximum Elements on Disk** size based on the persistence mode. For information about how to determine the persistence mode of the Terracotta Server Array, see the Ehcache product documentation for 2.8 at http://ehcache.org/documentation.

Increasing the Maximum Elements on Disk Size When the Terracotta Server Array Uses Permanent-Store Persistence Mode

To increase the Maximum Elements on Disk size when the Terracotta Server Array uses permanent-store persistence mode

1. Shut down all Integration Servers that are connected to the Terracotta Server Array.

2. On each Integration Server, go to the *Integration Server_directory* /instances/*instance_name*/config/Caching directory and open the SoftwareAG-IS-Core.xml file in a text editor.

3. Increase the value of the `maxEntriesLocalDisk` parameter for the cache. The value for `maxEntriesLocalDisk` must be the same for every Integration Server in the cluster. The default value is 1000.

4. Save the files.

5. Shut down all servers in the Terracotta Server Array.

6. On each Terracotta Server Array server, go to the *TerracottaHome*\bin directory and open the tc-config.xml file in a text editor.

7. Locate the `<servers><server><data>` element in the tc-config.xml file and delete it. This element indicates the directory where the cache configuration and cached data reside.

8. Save the files.

9. Start all the servers in the Terracotta Server Array.

10. Start one Integration Server with the new configuration, go to **Settings** > **Caching** in Integration Server Administrator, and verify that the change has taken effect.

11. Start the other Integration Servers.

Increasing the Maximum Elements on Disk Size When the Terracotta Server Array Uses Temporary-Swap-Only Persistence Mode

To increase the Maximum Elements on Disk size when the Terracotta Server Array uses temporary-swap-only persistence mode

1. Shut down all Integration Servers that are connected to the Terracotta Server Array.

2. On each Integration Server, go to the *Integration Server_directory* /instances/*instance_name*/config/Caching directory and open the SoftwareAG-IS-Core.xml file in a text editor.

3. Increase the value of the `maxEntriesLocalDisk` parameter for the cache. The value for `maxEntriesLocalDisk` must be the same for every Integration Server in the cluster. The default value is 1000.

4. Save the files.

5. Shut down and restart all servers in the Terracotta Server Array.

6. Start one Integration Server with the new configuration, go to **Settings** > **Caching** in Integration Server Administrator, and verify that the change has taken effect.

7. Start the other Integration Servers.

How a Stateful Integration Server Cluster and a Terracotta Server Array Handle Failures

This section contains information for the server administrator who configures and troubleshoots the Integration Server cluster. Consider the failure modes and information described in this section when creating your start-up scripts and procedures. Your scripts and procedures should take into account how Integration Server responds if a failure occurs.

In Terracotta, the default failover behavior is availability, but you can set it to consistency instead. For information, see the section on failover tuning for guaranteed consistency in the *BigMemory Max Administrator Guide*.

What Happens when an Integration Server in a Stateful Cluster Cannot Connect to the Terracotta Server Array?

The way Integration Server behaves when it cannot connect to the Terracotta Server Array depends on whether Integration Server has made the initial connection to the Terracotta Server Array and has obtained the tc-config.xml file. When Integration Server starts up, it connects to the first specified Terracotta Server Array server in the **Terracotta Server Array URLs** list and downloads the tc-config.xml file. Integration Server then uses the settings in the tc-config.xml file to make connections to the servers in the Terracotta Server Array.

For more information about the tc-config.xml file, see *Using BigMemory with webMethods Products* and *webMethods Integration Server Administrator's Guide*.

Integration Server Cannot Download the tc-config.xml File

If Integration Server cannot download the tc-config.xml file, the startup sequence pauses while Integration Server tries to connect to the first Terracotta Server Array server specified in the **Terracotta Server Array URLs** list. The length of the pause is determined by the `watt.server.cachemanager.connectTimeout` parameter (see *webMethods Integration Server Administrator's Guide*). If a connection is not established to any of the servers in the Terracotta Server Array within the time specified by this parameter, Integration Server logs an error and takes the action associated with the selected option for the **Action on Startup Error** parameter (see"About Action on Startup Error Options" on page 25).

Do the following:

1. Start the Terracotta Server Array.

2. Make sure the machine on which Integration Server is running can reach the Terracotta Server Array host specified in the tc-config.xml file.

3. Test the connection by pinging the servers in the Terracotta Server Array. You can find a list of the servers are listed in the **Terracotta Server Array URLs** list on the **Settings > Caching > Add Cache Manager** page in Integration Server Administrator.

4. Make sure the same version of Terracotta software is running on the client and the Terracotta Server Array. Check the ehcache.log file located in the *Integration Server_directory* /instances/*instance_name*/logs directory. If the Terracotta software version numbers are not the same, a version mismatch error will be logged and Integration Server shuts down.

5. Make sure the Integration Server has a valid Terracotta license key. If not, you can enable clustering on the Integration Server, but it will not be able to connect to the Terracotta Server Array. When the Integration Server starts, it will write an error to the server log and then take the action associated with the selected option for the **Action on Startup Error** parameter. You can add a Terracotta license key by placing the Terracotta license file into the *SoftwareAG_directory*\common\conf directory of the Integration Server. You must restart the Integration Server after adding a Terracotta license. Alternatively, you can use the **Server > Licensing > Licensing Details > Edit Licensing Details** page in Integration Server Administrator to point to a Terracotta license file at a different location. For more information about adding the Terracotta license file, see the section *Adding a Terracotta License* in the *webMethods Integration Server Administrator's Guide*.

Integration Server Cannot Connect to the Servers in the tc-config.xml File

If Integration Server has made the initial connection to the Terracotta Server Array and has obtained the tc-config.xml file, but the server cannot connect to any other servers in the Terracotta Server Array, Integration Server will wait indefinitely. Integration Server writes entries to the tc-client-logs and ehcache log files.

You can configure Integration Server to wait for a specified amount of time to connect to the Terracotta Server Array. After the wait time elapses, Integration Server takes the action associated with the selected option for the **Action on Startup Error** parameter. You configure the wait time for Integration Server by adding Java system properties to the custom_wrapper.conf file.

1. Go to the *Software AG_directory* /profiles/IS_*instance_name*/configuration directory.

2. In a text editor, open the custom_wrapper.conf file.

3. Add two wrapper.java.additional.*n* parameters as follows:

```
wrapper.java.additional.n=-Dcom.tc.l1.max.connect.retries=retryAttempts
wrapper.java.additional.n=-Dcom.tc.l1.socket.reconnect.waitInterval
=waitInterval_ms
```

   Where:

   - *n* is the next unused sequential number.
   - *retryAttempts* is the number of attempts Integration Server is to make to connect to the Terracotta Server Array.

- *waitInterval_ms* is the number of milliseconds Integration Server is to wait before each attempt to retry the connection.

4. Save and close the file.

5. Restart Integration Server to make the changes take effect. For more information about passing Java system properties to Integration Server, see the section *Passing Java System Properties to Integration Server* in the *webMethods Integration Server Administrator's Guide*.

Note:
For Microservices Runtime, and the properties to JAVA_CUSTOM_OPTS in *Integration Server_directory* /bin/server.bat(sh). For example, the property might look like the following: `set JAVA_CUSTOM_OPTS="Dcom.tc.l1.max.connect.retries=retryAttempts -Dcom.tc.l1.socket.reconnect.waitInterval= waitInterval_ms"`

What Happens When Integration Server in a Stateful Cluster Is Disconnected from the Terracotta Server Array?

When an Integration Server in a stateful cluster is disconnected from the Terracotta Server Array, the following occurs:

- Integration Server will accept new connections.
- Stateless sessions, new and existing, will continue to be fully functional. Note that stateless session information is not written to Terracotta Server Array.
- Stateful sessions, new and existing, will execute but will limited functionality because stateless session information will not be replicated to other Integration Servers in the cluster as long as the Integration Server remains disconnected from the Terracotta Server Array. The Integration Server maintains its own session state information and does not have access to state information from other cluster members while disconnected from the Terracotta Server Array.

  When executing a top-level stateful service, Integration Server attempts to write to the Terracotta Server Array after service invocation completes. If the **Timeout Behavior** is set to localReads or exception, the attempt to write to the Terracotta Server Array returns an exception. Integration Server writes this exception, often a NonStopCacheException, to the server log as part of the following error message: `[ISS.0033.0154E] Could not save session sessionID to the session cache.` *exceptionMessage*

  At the time Integration Server reconnects to the Terracotta Server Array, session state information becomes available across the cluster. However, all of the Integration Servers in the cluster might have different session state information. The first Integration Server to process a request updates the Terracotta Server Array with the new session state information. The updated session information becomes the session state for all of the Integration Servers in the cluster going forward. When another Integration Server begins to process a request for the session, that Integration Server retrieves the session state from the Terracotta Server Array. Consequently, session information on other Integration Servers in the cluster might be lost.
- How distributed caches in Integration Server behave during a loss of connection is controlled by the **Timeout**, **Immediate Timeout When Disconnected**, and **Timeout Behavior** settings

for each distributed cache. For information about these settings, see the section Configuring a Distributed Cache in the *webMethods Integration Server Administrator's Guide*.

Clients for a distributed cache continue to process requests, if all of the following are true:

- The **Timeout Behavior** parameter for each distributed cache is set to localReads or noop.
- The services do not attempt to write any data to a distributed cache for which the **Timeout Behavior** is localReads or exception.

- The list of cluster hosts on the **Settings > Clustering** page will not contain any cluster members.
- Integration Server writes entries to the tc-client-logs and ehcache log files when it disconnects and reconnects to the Terracotta Server Array.

The way in which Integration Server rejoins the Terracotta Server Array depends on how long the connection is lost.

If...	Then...
The connection is momentarily interrupted	If the Terracotta Server Array is configured to automatically reconnect, the Integration Server identity and state are retained on the Terracotta Server Array so that Integration Server can readily reconnect when the connection is restored. To learn more about the reconnect behavior of a Terracotta Server Array, see the information about automatic client reconnect in the *Configuring Terracotta Clusters For High Availability* documentation on the Terracotta website. If the Terracotta Server Array is not configured to automatically reconnect, the Integration Server will attempt to rejoin the Terracotta Server Array.
The connection is lost for a long period of time	Integration Server attempts to rejoin the Terracotta Server Array. To prevent inconsistencies in your distributed cache, Terracotta automatically acquires a new cache state from the Terracotta server after rejoining the Terracotta Server Array. For more information about the rejoin and nonstop behavior of Integration Server, see the section *The Rejoin Behavior of a Distributed Cache* in the *webMethods Integration Server Administrator's Guide*. .

What Happens When Terracotta Servers in the Terracotta Server Array Are Disconnected?

If the network connection is interrupted between the active server and one or more of the standby servers in the Terracotta Server Array, a situation might arise where one of the standby servers becomes an active server. This situation could result in a split brain scenario. In a split brain scenario, the connection between the servers is restored but two or more Terracotta servers assume

the role of the active server. When this happens, the data shared between the servers can become lost or corrupted.

To prevent a split brain scenario, Terracotta determines which server should become the new active server based on how many clients were connected to each server.

How Do Integration Servers Respond when Terracotta Servers in the Terracotta Server Array Are Disconnected?

If the network connection is interrupted between the active server and one or more of the standby servers in the Terracotta Server Array, the Integration Servers that were connected to the active server will continue to use the same active server. However, if a new Integration Server joins the cluster, it will attempt to join the first Terracotta Server defined in the **Terracotta Server Array URLs** list. With this behavior, it is possible that new Integration Servers joining the cluster will connect to a new active server. This situation can lead to data loss when the servers reconnect. Additionally, while both servers are split, the Integration Server cluster might behave unpredictably because the Integration Servers in the cluster would have access to different cached data.

To avoid this situation, Software AG recommends that you shut down the new active server until the connection between the Terracotta servers is restored.

To prevent these issues, make sure the network connection between active server and standby servers in the Terracotta Server Array is reliable. For more information about preventing a split brain scenario, see "Split Brain Scenario" in the Terracotta Server Arrays Architecture documentation on the Terracotta website.