webMethods Integration Server Clustering Guide

Version 9.0 SP1

June 2013

Document ID: IS-CL-90SP1-20130618

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 Settings > License 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 storage locations for services on webMethods Integration Server, using the convention *folder.subfolder:service*.
UPPERCASE	Identifies keyboard keys. Keys you must press simultaneously are joined with a plus sign (+).
Italic	Identifies variables for which you must supply values specific to your own situation or environment. Identifies new terms the first time they occur in the text.
`Monospace font`	Identifies text you must type or messages displayed by the system.
{ }	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 (...).

Documentation Installation

You can download the product documentation using the Software AG Installer. Depending on the release of the webMethods product suite, the location of the downloaded documentation will be as shown in the table below.

For webMethods...	The documentation is downloaded to...
8.x and 9.x	A central directory named _documentation in the main installation directory (Software AG by default).
7.x	A central directory named _documentation in the main installation directory (webMethods by default).
6.x	The installation directory of each product.

Online Information

You can find additional information about Software AG products at the locations listed below.

If you want to...	Go to...
Access the latest version of product documentation.	Software AG Documentation website http://documentation.softwareag.com
Find information about product releases and tools that you can use to resolve problems. See the Knowledge Center to: ■ Read technical articles and papers. ■ Download fixes and service packs. ■ Learn about critical alerts. See the Products area to: ■ Download products. ■ Download certified samples. ■ Get information about product availability. ■ Access older versions of product documentation. ■ Submit feature/enhancement requests.	Empower Product Support website https://empower.softwareag.com

If you want to...	Go to...
■ Access additional articles, demos, and tutorials. ■ Obtain technical information, useful resources, and online discussion forums, moderated by Software AG professionals, to help you do more with Software AG technology. ■ Use the online discussion forums to exchange best practices and chat with other experts. ■ Expand your knowledge about product documentation, code samples, articles, online seminars, and tutorials. ■ Link to external websites that discuss open standards and many web technology topics. ■ See how other customers are streamlining their operations with technology from Software AG.	Software AG Developer Community for webMethods http://communities.softwareag.com/

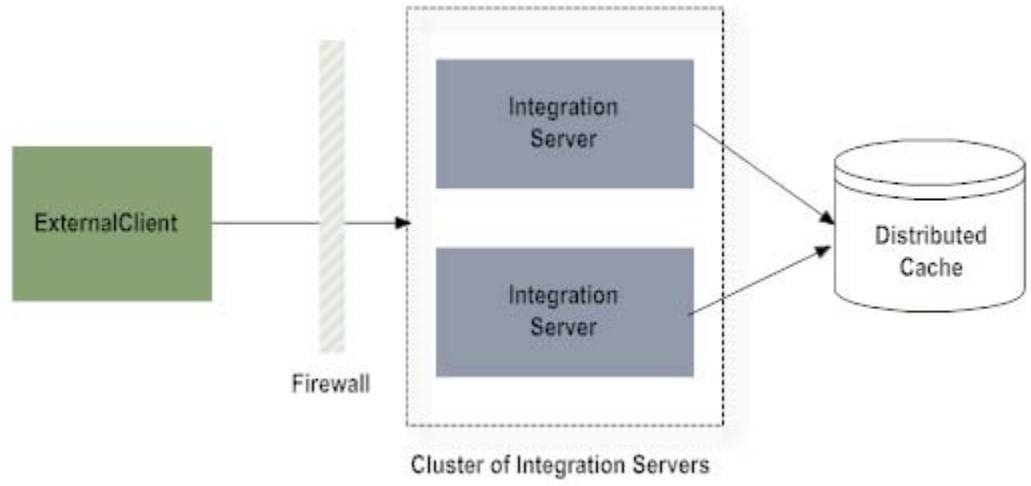
1 An Overview of Clustering

What Is Clustering?

Clustering is an advanced feature of the webMethods product suite that substantially extends the reliability, availability, and scalability of the webMethods Integration Server. Clustering accomplishes this by providing the infrastructure and tools to deploy 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 (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.

The following diagram shows clustering in its simplest form:



Important! Do not perform development work in a clustered environment. Basic namespace locking, the VCS Integration feature, and local service development are not supported across Integration Servers in a cluster.

Load Balancing in a Cluster

Load balancing is an optimizing feature you use with clustered webMethods Integration Servers. Load balancing controls how requests are distributed to the servers in the cluster. You must use a third-party load balancer to perform load balancing. A third-party 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 webMethods Integration Servers. Third-party load balancers also offer virtual IP support, but they are not "out of the box." Most third-party load balancers perform load balancing in a round-robin manner or based on network level metrics such as TCP connections and network response time.

What Data Is Shared in a Cluster?

In a clustered configuration, the session state for clients connected to all servers in the cluster is stored in the distributed cache created by the caching software. Integration Server uses Terracotta Server Array as the caching software. The cache allows transactions in a conversation to be continued on other servers in the cluster. In addition, Trading Networks query results and ART polling notifications are stored in the distributed cache.

Information about scheduled jobs, jobs tracked by guaranteed delivery, and XREF data is stored in an external database, identified by the ISInternal functional alias. In addition, JOIN documents (ANDs and XORs only) and the processing state of activations that are participating in a JOIN are also stored in the database. For more information about functional aliases, refer to *Installing webMethods Products*.

Overview of Failover Support

Failover support allows you to recover 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 webMethods Integration Server to which the client is connected fails, the client automatically reconnects to another webMethods Integration Server in the cluster.

Note: webMethods 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 achieve failover support a number of ways:

- webMethods Integration Server clustering. With multiple webMethods Integration Servers, if one Integration Server fails, requests can be automatically redirected to another server, thereby avoiding a single point of failure.

- Redundant hardware configurations. By specifying hardware configurations that are redundant, for example multiple machines, multiple file servers, and so on, you further reduce the chance of a single point of failure.

Overview of Reliability

Several features of the webMethods Integration Server improve reliability in a clustered or unclustered environment.

- Guaranteed delivery. This feature 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.
- Checkpoint restart services. The pub.storage services allow you to code your flow service to store state information and other pertinent information in the short-term data store. If the 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 services, see the *webMethods Integration Server Built-In Services Reference*.

Guaranteed Delivery and Clustering

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.

Guaranteed delivery is useful with or without clustering. If you are not using clustering, guaranteed delivery makes sure a client resubmits a service request until it succeeds and a response is returned. Guaranteed delivery also makes sure the service executes only once. For example, if the network connection between the client and the webMethods 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, if the server on which the service is running becomes unavailable, the client retries the service on another server in the cluster. If the request fails against all servers in the cluster, you must use guaranteed delivery to guarantee execution. As in an unclustered environment, you must use guaranteed delivery to prevent a service from executing more than once.

For more information about guaranteed delivery, refer to *webMethods Integration Server Administrator's Guide* and the *Guaranteed Delivery Developer's Guide*.

Checkpoint Restart

webMethods Integration Server provides a number of built-in services (in the pub.cache folder) that you can use to make your flows more robust. With these services, you can write state information and other pertinent data to a data store in the Integration Server short-term data store. If the webMethods Integration Server on which your flow is executing becomes unavailable, when your flow restarts it can check the state information in the data store and begin processing at the point where the flow was interrupted. For more information about pub.cache services, see the *webMethods Integration Server Built-In Services Reference*.

Putting It All Together

This table summarizes how checkpoint restart, clustering, and guaranteed delivery 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

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
	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, you must restart the service manually.	Flow services: At the specified checkpoint Other services: At the beginning
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.

webMethods Integration Server Session Objects

Clustered Integration Servers create and maintain the session objects that are stored in a 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.

Scheduled Jobs

You can schedule jobs to run on one, any, or all Integration Servers in the cluster. Integration Server stores information about scheduled jobs in the external database identified by the ISInternal functional alias. 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. For instructions, see the chapter about managing services in *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.

When a client connects to a server that is in a clustered environment (using the Context class), the server returns information about the other servers in the cluster to the client. In the event that a request is not processed, the client can use this information to connect to another server in the cluster to have the request fulfilled.

Whether you are using the Context or TContext class to communicate with the webMethods Integration Server, the redirection of failed requests is transparent to the client. The logic to handle redirection is in the webMethods Integration Server Context or TContext class.

To improve the failover capability, before your client calls Context or TContext to connect to an Integration Server in the cluster, have your client issue the setRetryServer method in that class to specify another server to try if the first server the client tries to connect to is unavailable.

No special processing is required in your clients.

Note: webMethods Integration Server clustering provides failover capabilities to HTTP-based webMethods clients, such as those clients built using the webMethods Context and TContext classes.

Remote Integration Servers in a 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.

In general, if a remote server is not available when a request is made, and the remote server is not part of a cluster, Integration Server will use the retry server specified in the alias definition for the remote server.

If the requested remote server is part of a cluster, Integration Server will attempt to use the next Integration Server in the cluster until one is found. If no available Integration Server is found in the cluster, Integration Server will use the retry server specified in the alias definition for the remote server.

In cases where a client calls the pub.remote:invoke to run a service on a remote server that is part of a cluster, it is possible to modify the default behavior by using the *$clusterRetry*.

This parameter controls whether the service will try other Integration Servers in the cluster. When this parameter is set to true, the service will try other Integration Servers in the cluster. If none are found, the service will try the retry server specified in the remote server alias definition. When this parameter is set to false, the service does not try other Integration Servers in the cluster. Instead, it will try the retry server specified in the alias definition.

For more information about remote servers, see *webMethods Integration Server Administrator's Guide*.

For more information about the pub.remote:invoke service, see the *webMethods Integration Server Built-In Services Reference*.

You can also use the Java API to control the retry behavior. Specifically, if you are using the Context or TContext class, you can control whether Integration Server looks for a retry server by calling the BaseContext.setAllowRedir method. In addition, you can specify which retry server to use by calling the BaseContext.setRetryServer method. For more information about the Context and TContext classes, see the *webMethods Integration Server Java API Reference*.

2 Installation and Setup

Overview

This section summarizes the steps you must take to set up server clustering:

- Install the Integration Servers. All Integration Servers that you want to include in the cluster should be running the same version of Integration Server software and be at the same patch level.
- 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 23.
- Configure the Integration Server for clustering. Instructions are provided in "Setting Up the webMethods Integration Servers in the Cluster" on page 20.

Considerations for Using webMethods Trading Networks in a Clustered Environment

For Trading Networks to run in a clustered environment, you must do the following:

- Configure Trading Networks so that user accounts, data cached in memory, and Integration Server properties are synchronized on all servers in the cluster. You control these synchronizations by setting Trading Networks system properties.
- Ensure that all Trading Networks instances use the same database.

For more information, see *webMethods Trading Networks Administrator's Guide*.

Considerations for Using webMethods Adapters in a Clustered Environment

For an adapter to run in a clustered environment, a duplicate version of it must exist on each machine in the cluster. Maintaining duplicate versions of an adapter requires an administrator to perform manual tasks. For current cluster configuration information for adapters, see the documentation provided with each adapter.

Setting Up the webMethods Integration Servers in the Cluster

It is recommended that you maintain the same environment for all servers in a cluster, for example, the same packages of services and ACLs that protect services. For information on the server environment variables that you should maintain, refer to "Server Environment" on page 21 below.

You can have a variety of Integration Servers in your cluster, for example one server on Windows, one on UNIX, and one on Linux.

Server Environment

Although not required, it is recommended that the servers in a cluster each have the same server environment. 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.

  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 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*\config. Copy this file from one server to another to duplicate event subscriptions on all servers in the cluster.
- 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. For information about webMethods Deployer, see the *webMethods Deployer User's Guide*.

- 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 webMethods Broker. All servers must connect to the same Broker and present the same client prefix for the Broker.
- 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 *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.
- Each server must have the same cluster name.
- 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. Additionally, the Terracotta Server Array that you want to use to store the system caches must already be configured. 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" on page 23.

Switching from the Embedded Database to an External RDBMS

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 to a shared external RDBMS

1 In Integration Server Administrator, navigate to the Security > Certificates > Client Certificates screen, and make a note of the certificate mappings.

If the cluster already exists, 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. See *Installing webMethods Products* for instructions.

If those database components do not yet exist, create them and connect all Integration Servers in the cluster to them.

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

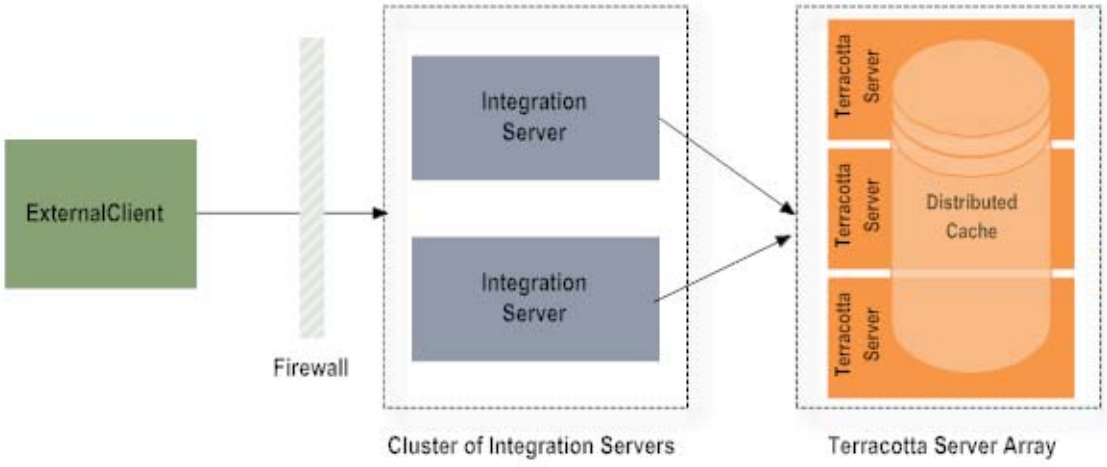
Note: This service migrates scheduled tasks only; client mappings and run-time data will not be migrated.

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

4 Configure Integration Server as described below.

Using Terracotta as the Caching Software

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 23.

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 webMethods 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 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 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 Ehcache chapter of *webMethods Integration Server Administrator's Guide*.

After you complete these steps, you can enable clustering as described in "Enabling Clustering for an Integration Server" on page 24.

Enabling Clustering for an Integration Server

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 23 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 the Settings menu of the navigation area, click Clustering.

2 Click Edit Cluster Settings.

3 Click Enable Cluster.

4 Specify the following information:

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.
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 information in its memory. (Integration Server Administrator displays the session timeout on the Settings > Resources screen.)
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*

5 Click Save Settings.

6 Restart the Integration Server.

7 Using Integration Server Administrator, navigate to Settings > Cluster and verify that all nodes in the cluster are displayed under Cluster Hosts.

Notes:

■ 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.

What Happens When Integration Server Cannot Connect to the Distributed Cache?

When Integration Server cannot connect to the distributed cache, one of the following occurs:

- 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 error log and terminates immediately. To enable clustering, diagnose and correct the problem. 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 *webMethods Integration Server Administrator's Guide*.
- 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

Logging for cache activity on the Terracotta Server Array is controlled by Ehcache. 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*. Ehcache creates additional log files on Integration Server when Integration Server connects to a Terracotta Server Array. You can specify where you want 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*/logs/tc-client-logs directory. For information about how Ehcache logs cache manager activity when Integration Server connects to a Terracotta Server Array, see *webMethods Integration Server Administrator's Guide*.

Scheduling Jobs to Run in the 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. For instructions, see the chapter about scheduling services in *webMethods Integration Server Administrator's Guide*.

Specifying Unique Logical Names for Integration Servers in a Cluster

By default, Integration Server uses the host name to identify itself while scheduling tasks. However, when a cluster of Integration Servers are hosted on a single machine, the host name itself cannot uniquely identify the individual Integration Server instances. In such cases, use the watt.server.scheduler.logical.hostname property to specify a unique logical name to identify individual Integration Server instances.

The default value for this parameter is the host name.

Keep the following points in mind when setting the watt.server.scheduler.logical.hostname property:

- Set this property only if you are running multiple Integration Server in a cluster on a single machine.
- Set this property on each Integration Server instance in the cluster before scheduling any tasks.
- The logical host name you specify must be unique.
- The logical host name you specify cannot contain a semicolon (;).

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

1 Open the Integration Server Administrator if it is not already open.

2 In the Settings menu of the Navigation panel, click Extended.

Integration Server displays a screen that lists the configuration properties.

3 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 screen to configure the Integration Server, see *webMethods Integration Server Administrator's Guide*.

4 Click Save Changes.

5 Restart Integration Server for the changes to take effect.

Using Guaranteed Delivery with Clustering

The guaranteed delivery capabilities of the webMethods Integration Server ensure guaranteed one-time execution of services. Guaranteed delivery ensures 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.

In a clustered environment, if the Integration Server on which the service is running becomes unavailable, the client retries the service on another Integration Server in the cluster. However, to make sure a request will run if it fails on every server in the cluster, you must configure your Integration Servers to use guaranteed delivery. In addition, as in an unclustered environment, you must use guaranteed delivery to prevent a service from executing more than once. For more information about guaranteed delivery, refer to *webMethods Integration Server Administrator's Guide*. For more information about developing services that use guaranteed delivery, refer to the *Guaranteed Delivery Developer's Guide*.

When you use webMethods Integration Server clustering, guaranteed delivery stores information about requests for *all* the servers in the cluster 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 Open the Integration Server Administrator if it is not already open.

2 In the Settings menu of the Navigation panel, click Extended.

The server displays a screen that lists configuration properties specified in the server.cnf file.

3 To set how long a cluster server waits between attempts to place an update lock, 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.

4 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 screen to configure the Integration Server, see *webMethods Integration Server Administrator's Guide*.

5 Click Save Changes.

6 Restart the server to put your changes into effect.

3 Managing Server Clustering

Viewing the Servers in the Cluster

When you have server clustering enabled, 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 (or shared) cache.

To view a list of all clustered servers

1 Start the Integration Server Administrator. See *webMethods Integration Server Administrator's Guide* if you need help with this step.

2 In the Settings menu of the navigation area, click Cluster.

The server displays the list of servers in the cluster.

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

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

Note: You can use the wm.server.cluster:clearClusterMemberList service to refresh the list of clustered servers.

Removing a Server from the Cluster

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

To remove a server from the cluster

1 If you are using guaranteed delivery, you must take the server offline before removing it from the cluster. If you do not, 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. See "Taking a Server Offline" on page 34 for instructions on taking a server offline.

2 If it is not already running, start the Integration Server Administrator. See *webMethods Integration Server Administrator's Guide* if you need help with this step.

3 In the Settings menu of the navigation area, click Clustering.

4 Click Edit Cluster Settings.

5 Click Disable Cluster to turn clustering off for this server.

6 Click Save Settings.

7 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 the Cluster

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

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 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 If it is not already running, start the Integration Server Administrator of the server you want to add back into the cluster. See *webMethods Integration Server Administrator's Guide* if you need help with this step.

2 In the Settings menu of the navigation area, click Clustering.

3 Click Edit Cluster Settings.

4 Click Enable Cluster to turn clustering on for this server.

5 Make any changes you want to the configuration.

6 Click Save Settings.

7 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.

Important! If you are using guaranteed delivery, the server you are returning to the cluster might be offline. When a server is offline, it can only be accessed through a single, unpublished port. Before you can use the server, you must first bring it back online. See "Bringing a Server Back Online" on page 34 for instructions.

Taking a Server Offline

There may be times when you need to take a server offline. When you take a server offline, you are limiting access to it through a single, unpublished port.

Before taking a server offline, make sure that the Integration Server has an existing, unpublished port. If one does not exist, create it.

Note: If you have guaranteed delivery, you must take a server offline before removing it from the cluster. If you do not, 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.

To take a server offline

1 Start the Integration Server Administrator. See *webMethods Integration Server Administrator's Guide* if you need help with this step.

2 In the Security menu of the navigation area, click Ports.

3 Click Change Primary Port.

4 In the Select New Primary Port area of the screen, select the port that you want to make the primary port.

5 Click Update.

6 On your browser, update the URL for the Integration Server Administrator to use the new port.

7 Disable all other listening ports.

Bringing a Server Back Online

There may be times when you need to bring a server back online. For example, you might have taken a server offline because you run guaranteed delivery and you needed to remove the server from the cluster. (When a server is offline, it can only be accessed through a single, unpublished port.) After you add the server back to the cluster, you must bring it back online.

To bring a server back online

1 Start the Integration Server Administrator using the new listening port that you designated as the primary port when you took the server offline. See *webMethods Integration Server Administrator's Guide* if you need help with this step.

2 In the Security menu of the navigation area, click Ports.

3 Re-enable the port that you disabled when you took the server offline.

4 Click Change Primary Port.

5 Change the primary listening port back to the published primary port for the server.

6 Click Update.

7 On your browser, update the URL for the Integration Server Administrator to the published primary port. This is the original primary port used by the server before you took the server offline.

8 Enable all disabled listening ports.

How Integration Server Cluster and 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.

What Happens when Integration Server Cannot Connect to the Terracotta Server Array?

How 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 *Getting Started with the webMethods Product Suite and Terracotta* and *webMethods Integration Server Administrator's Guide*.

Integration Server Cannot Download the tc-config.xml File

If Integration Server is unable to download the tc-config.xml file, the startup sequence pauses while Integration Server attempts 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. 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 throws an exception and shuts down immediately.

For information about setting the watt.server.cachemanager.connectTimeout parameter, see *webMethods Integration Server Administrator's Guide*.

Solution

When Integration Server cannot connect to the Terracotta Server Array, do the following:

- Start the Terracotta Server Array if it is not already started.
- Make sure the machine on which Integration Server is running can reach the Terracotta Server Array host specified in the tc-config.xml file.
- Test the connection by pinging the servers in the Terracotta Server Array. The servers are listed in the Terracotta Server Array URLs list on the Settings > Caching > Add Cache Manager screen.
- 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*/logs directory. If the Terracotta software version numbers are not the same, the following error will be logged:

  ```
  2013-04-09 11:02:20,102 ERROR - Client/Server Version Mismatch Error: Client
  Version: 3.7.3, Server Version: 3.7.4.  Terminating client now.
  ```

- Make sure the Integration Server has a valid Terracotta license key. If the Integration Server does not have a valid license key, you can enable clustering on the Integration Server; however, it will not be able to connect to the Terracotta Server Array. When the Integration Server starts up, it will write an error to the server log.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 Settings > License > License Details > Edit 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 *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.

Solution

You can configure Integration Server to shut down after waiting for a specified amount of time to connect to the Terracotta Server Array. You configure Integration Server to do this by adding the following Java system properties to the setenv.bat/sh file, which is located the *Integration Server_directory*\bin directory:

On Windows systems

```
set JAVA_CUSTOM_OPTS="-Dcom.tc.l1.max.connect.retries=retryAttempts -
Dcom.tc.l1.socket.reconnect.waitInterval=waitInterval_ms"
```

On Unix systems

```
JAVA_CUSTOM_OPTS="-Dcom.tc.l1.max.connect.retries=retryAttempts -
Dcom.tc.l1.socket.reconnect.waitInterval=waitInterval_ms"
```

Where:

- *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.

For example:

```
JAVA_CUSTOM_OPTS="-Dcom.tc.l1.max.connect.retries=100 -
Dcom.tc.l1.socket.reconnect.waitInterval=1000"
```

Restart Integration Server for the changes to take effect. If Integration Server runs as a Windows service, you must update the service before restarting Integration Server. For more information about passing Java system properties to Integration Server and updating the service, see *webMethods Integration Server Administrator's Guide*.

What Happens when Integration Server Is Disconnected from the Terracotta Server Array?

How Integration Server behaves 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 *webMethods Integration Server Administrator's Guide*.

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

- Integration Server will accept new connections; however, the 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.
- Existing clients continue to process requests, if all of the following are true:
  - The Timeout Behavior parameter for each distributed cache is set to localReads. The existing clients execute only stateless services.
  - The services do not attempt to write any data to a distributed cache.
- The list of cluster hosts on the Settings > Cluster page will not contain any cluster members.
- Once the connection to the Terracotta Server Array is restored, new clients can establish sessions on the Integration Server.
- The Integration Server writes entries to the tc-client-logs and ehcache log files when it disconnects and reconnects to the Terracotta Server Array.

Solution

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 *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.

Solution

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.

Index