Together Application Cluster

Together Application Cluster

Together Teamlösungen EDV-Dienstleistungen GmbH

Elmargasse 2-4
A-1190
Vienna
Austria
+43 (0) 5 04 04 - 122
+43 (0) 5 04 04 - 11 122
<office@together.at>
http://www.together.at/together/index.html

Table of Contents

Chapter 1. Introduction to Together Application Cluster (TAC)

Together Application Cluster is an open source (LGPL) Web server plugin that provides load-balancing, fail-over and session affinity capabilities on several web/application servers platforms.

Load balancing is a mechanism where the user requests are distributed to different nodes within the group of servers (cluster). Instead of execute an application on a single server, the system executes application code on a dynamically selected server. When a client requests a service, one of the clustered servers is chosen to execute the request.

Together Application Cluster simultaneously supports both methods of scaling: vertical and horizontal. Vertical scaling is achieved by increasing the number of application servers running on a single machine, and horizontal scaling is done by increasing the number of machines in the cluster. In both cases TAC act as single point of entry into the cluster and as traffic dispatcher to individual application servers. They also provides fail-over mechanism, when an application instance or application server becomes unavailable (or unreachable). In all this cases TAC takes care about session affinity, allowing every user request to be connected back to the same application server instance that started the user session

To avoid web server to become “single point of failure” TAC (v6.2.3 and above) supports parallel work (connections) of multiple web server (TAC) instances against the same instance of application server (or against cluster of applications servers). This feature in combination with hardware/software (web server) load balancing enables possibility for web server clustering, group of servers that transparently runs web application as if it were a single web server. In this case, session affinity and fail-over capabilities are preserved on application servers level (not web servers level).

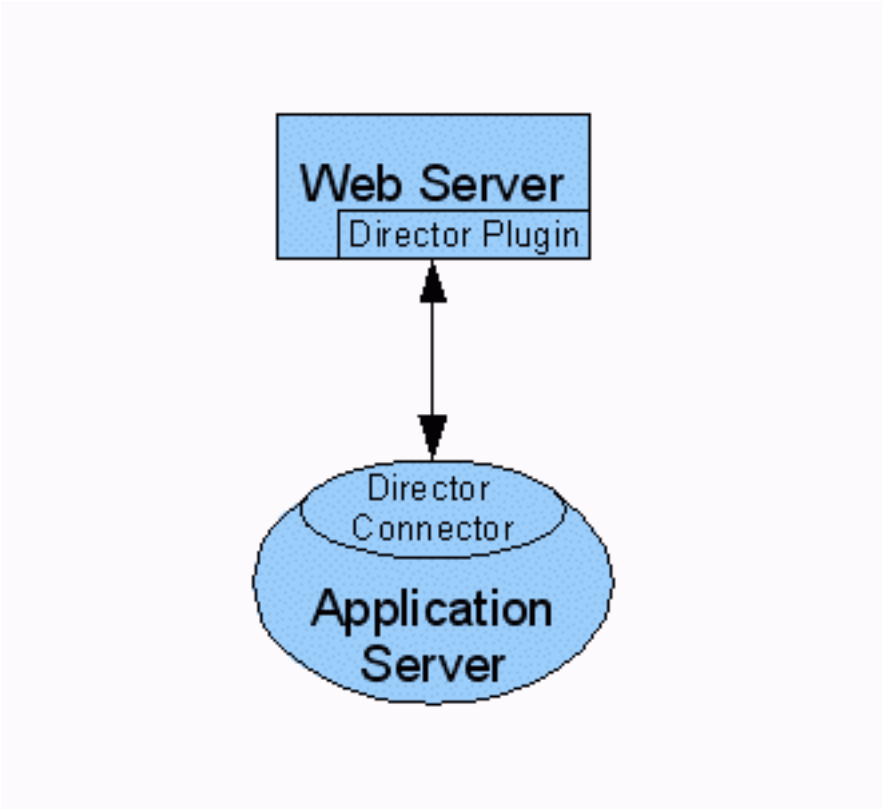
Features:

- Current list of supported features:
  - Application servers load-balancing
  - Application servers fail-over
  - Application servers Session affinity
  - Support for web servers clustering.
  - Administration web application.
- Supported web servers:
  - Apache (Windows/Linux/Debian)
  - IIS (Windows)
  - iPlanet (Windows)
- Supported application servers:

- Enhydra (5.x.xx and 6.x.xx)
- Tomcat (5.x.xx)
- Jetty (5.1.xx)
- Jonas (3.x.xx and 4.x.xx)

Chapter 2. Together Application Cluster use cases

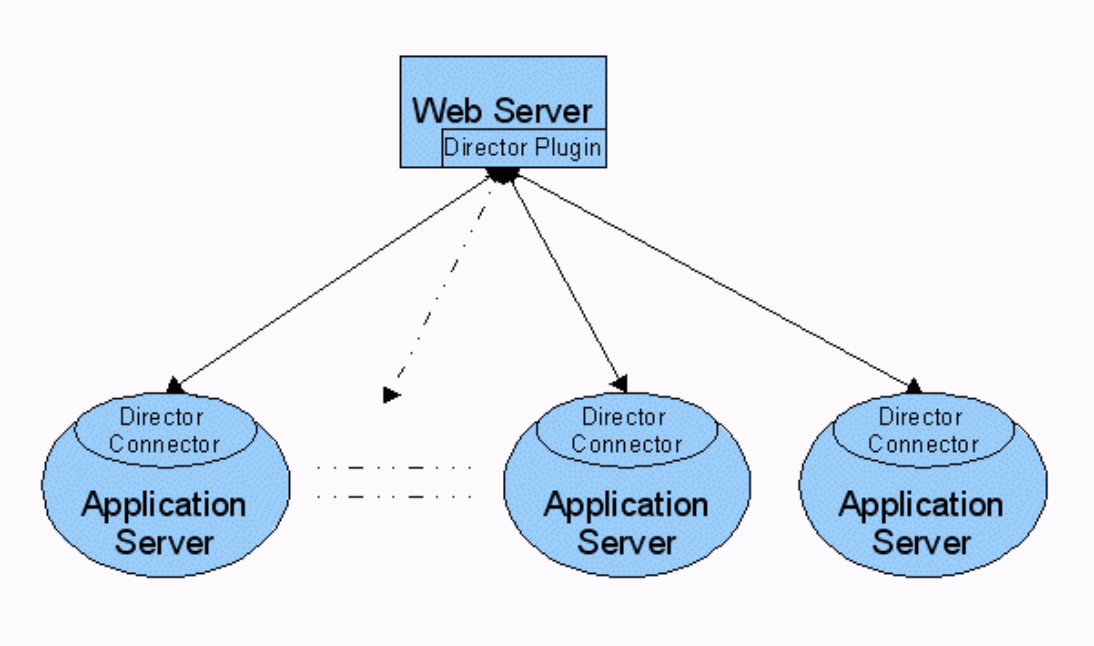
- Simplest use case includes one web server and one application server (on same host or on separated hosts).



  In this case TAC is simple used as connection provider between web server and application server. Both, web and application servers need to be separately configured to establish connection.

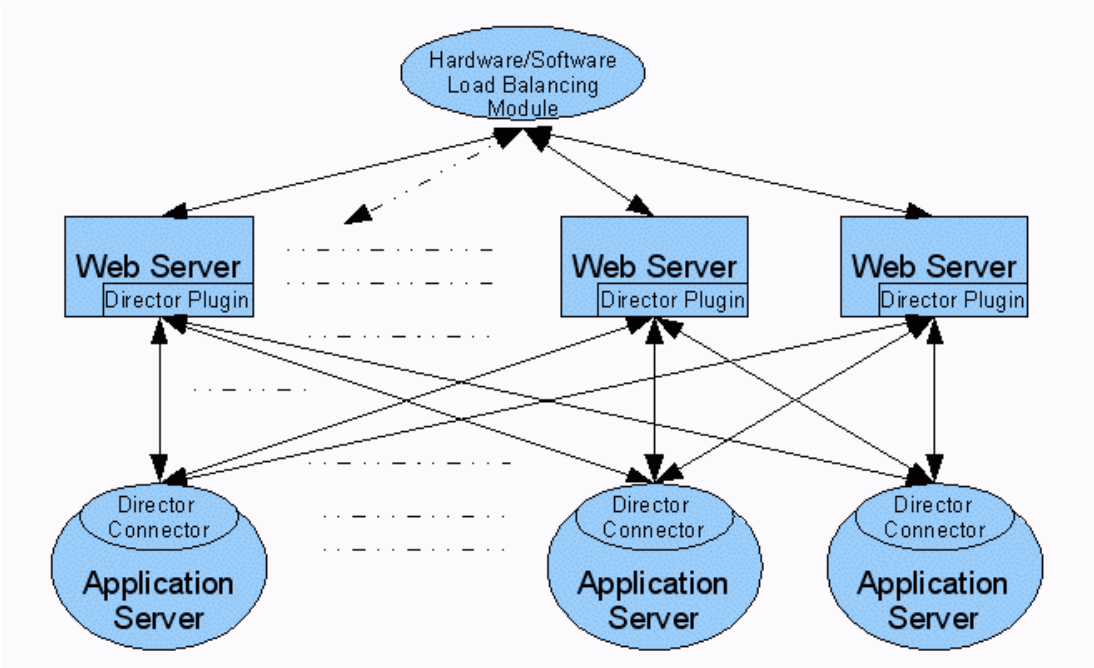
  Full list of supported applications and web servers is provided on the end of this chapter.

- Most common Enhydra Diretor use case is shown on next diagram.



In this case TAC provides load-balancing, fail-over and session affinity service for group of application servers (on same host or on separated hosts).

- And at last diagram we can see use case where multiple instances of TAC works together to support web server clustering and in the same time provides application server clustering. In this scenario every instance works independently of each other and provides same services as in previous use case, but if one (or more then one) of servers fails down, rest of web servers in cluster can keep to provide services to the client requests. In this case every request that is passed to TAC will be dispatched to related application server according to session affinity, load-balancing and fail-over rules. So, web server can be avoided as “single point of failure”, but to provide full web server clustering TAC depends on load balancing an fail-over mechanism in front of web servers cluster.



Chapter 3. Together Application Cluster installation notes

Apache Module

The Apache implementation of Together Application Cluster is an Apache 1.3.x module that is compiled and installed directly into your Apache server. The 'mod_enhydra_director.so' file is the name of the compiled module. Also, due to the design of the Apache server, a daemon program runs and performs housekeeping on a shared memory scoreboard that is used by the httpd child processes to keep track of multiserver connection traffic for load balancing and fail-over purposes. This daemon program is called 'edir_daemon'. A rough and simple debugging utility, 'edir_status' also exists to check the state of the shared memory scoreboard. This utility exists only for Apache because of its shared memory design.

For more information on the Together Application Cluster module for Apache, see the Apache (html [./tac/apache.html], pdf [./tac/apache.pdf]) documentation.

Note

Please configure properly your "ServerName" in httpd.conf file (see note in httpd.conf) and avoid to use "127.0.0.1" as "ServerName".

NSAPI Module

The Netscape server implementation of Together Application Cluster is an NSAPI server extension. This extension is normally compiled and installed on the Netscape Web server following the usual procedures for an NSAPI plugin. Additionally, the standard releases of Enhydra contain pre-built 'DLL' binaries for installation on Microsoft Windows NT or Windows 2000 systems.

Because the Netscape server implements a multithreaded shared memory design, there is no shared memory functionality in the NSAPI plugin, and the 'edir_daemon' and 'edir_status' are not available.

For more information on the Together Application Cluster module for NSAPI, see the IPlanet (html [./tac/iplanet_ws.html], pdf [./tac/iplanet_ws.pdf]) documentation.

IIS Module

The Microsoft IIS implementation of Together Application Cluster consists of both an ISAPI filter DLL and an ISAPI extension DLL. The IIS module only operates on Windows NT or Windows 2000, since

those are the platforms supported by IIS. The DLLs can be built from source code using Visual C++ 6.0 or later, or pre-built binaries can be obtained from the Enhydra release.

Because the IIS server implements a multithreaded shared memory design, there is no shared memory functionality in the ISAPI plugins, and the 'edir_daemon' and 'edir_status' are not available.

For more information on the Together Application Cluster module for IIS, see the IIS (html [./tac/internet_is.html], pdf [./tac/internet_is.pdf]) documentation.

Chapter 4. Building Together Application Cluster

Necessary Third Party Materials

In order to build Together Application Cluster, you must have the following third party materials:

- Ant, Java based build tool. The Ant is freely available at the Apache Jakarta Project [http://ant.apache.org]. *NOTE*:In case of Linux, it is needed only to build documentation.
- Java Development Kit, version 1.4.x or greater - this release was built with the j2sdk1.4.2. The jdk is freely available at http://java.sun.com/ [http://java.sun.com].
- To build Together Application Cluster Tomcat-Connector ({TAC.src.dir}/tomcat) you must have Tomcat.5.0.xx installation (http://jakarta.apache.org/tomcat/index.html), Enhydra6.1, EnhydraEnterprise6.x or JOnAS (with Tomcat.5.0.xx) installation on disk.
- To build Together Application Cluster Tomcat5.5-Connector ({TAC.src.dir}/tomcat5.5) you must have Tomcat.5.5.xx installation. (http://jakarta.apache.org/tomcat/index.html) or Enhydra6.2 installation on disk.
- To build Together Application Cluster Jetty 5.1.xx -Connector ({TAC.src.dir}/jetty) you must have Jetty 5.1.xx installation. (http://jetty.mortbay.org/jetty/), EnhydraEnterprise (Jetty version) or JOnAS (Jetty version) installation on disk.

Configuration

Before building Together Application Cluster, you must edit the file "build.properties" in the TAC directory. Alternatively, you may specify command line options to the *configure* executable, which will edit the "build.properties" file.

Go to the TAC directory and issue the following command:

```
configure
```

This command will use values from *build.properties*. In that file, you must specify the following variables:

```
version - version of distribution
```

```
release - release of distribution
```

```
jdk.dir - the path to your installation of the Java Development Kit
```

```
scope - only for Linux, what to build: all, apache or iplanet
```

For instance:

```
version=5.1
release=1
jdk.dir=C:/j2sdk1.4.2
```

Additionaly you may need to edit {TAC.src.dir}/<connectorName>/build.properties, to build Together Application Cluster Connector module(s), where <connectorName> can be: **''tomcat'', ''tomcat5.5'', or ''jetty''**

- To build Together Application Cluster Tomcat-Connector ({TAC.src.dir}/tomcat):
  - You must have Tomcat.5.0.xx installation (http://jakarta.apache.org/tomcat/index.html), Enhydra6.1, EnhydraEnterprise6.x or JOnAS (with Tomcat.5.0.xx) installation on disk.
  - Set "catalina.home" property in **{TAC.src.dir}/tomcat/build.properties** to proper value. In case of EnhydraLite or EnhydraEnterprise instalation set "catalina.home" to "{enhydra.install.dir}/multiserver" folder. In case of JOnAS(Tomcat.5.0.xx) installation set "catalina.home" to "{jonas.install.dir}".
- To build Together Application Cluster Tomcat5.5-Connector ({TAC.src.dir}/tomcat5.5):
  - You must have Tomcat.5.5.xx installation. (http://jakarta.apache.org/tomcat/index.html) or Enhydra6.2 installation on disk.
  - Set "catalina.home" property in **{TAC.src.dir}/tomcat5.5/build.properties** to proper value. In case of Enhydra or EnhydraEnterprise instalation set "catalina.home" to "{enhydra.install.dir}/multiserver" folder.
- To build Together Application Cluster Jetty 5.1.xx -Connector ({TAC.src.dir}/jetty):
  - You must have Jetty 5.1.xx installation. (http://jetty.mortbay.org/jetty/), EnhydraEnterprise (Jetty version) or JOnAS (Jetty version) installation on disk.
  - Set "jetty.home" property in **{TAC.src.dir}/jetty/build.properties** to proper value. In case of Enhydra or EnhydraEnterprise instalation set "jetty.home" to "{enhydra.install.dir}/multiserver" folder. In case of JOnAS(Jetty) installation set "jetty.home" to "{jonas.install.dir}".

Please note that Unix stile slashes (/) must always be used instead of Dos stile backslashes (\)

Configure options

You can give the following options to the configure command:

- Use all values from build.properties file:

```
configure
```

- Display option screen

```
configure -help
```

- Set variable values (individually or in group)

  Windows:

  ```
  configure [-version version_number] [-release release_number] [-jdkhome jdk_home_dir]
  ```

  Linux:

  ```
  configure [-version version_number] [-release release_number] [-jdkhome jdk_home_dir]
  [-scope all | apache | iplanet ]
  ```

  where scope defines what to build. In order to build apache, you must obtain an apache_1.3.XX.tar.gz source and put it into the 'install/Linux/apache' directory. The distributions will be built for all sources found there, but only the most suitable will be installed.

NOTE: If you have Apache 2 installed, you must disable it first, by temporary removing its httpd executable (found in /usr/sbin or similar folder) from your PATH, or by uninstalling it.

Building Together Application Cluster

Once you have edited the "build.properties" file discussed above, being still in the same directory issue the following command:

```
make
```

This command will produce a great deal of output, and it is recommended that you capture this output to a log file for later reference. The binary output of the build can be found in the following directories, depending on the OS:

- ./Distribution/Windows
- ./Distribution/Linux
- ./jetty/distribution
  - ./jetty/distribution/Windows
  - ./jetty/distribution/Linux
- ./tomcat5.5/distribution
  - ./tomcat5.5/distribution/Windows
  - ./tomcat5.5/distribution/Linux
- ./tomcat/distribution
  - ./tomcat/distribution/Windows
  - ./tomcat/distribution/Linux

Make Options

You can give one of the following options to the make command:

- *make*: builds default distribution.
- *make help*: display this screen.
- *make distributions*: builds distribution files and all Connectors.
- *make distributionsNoConnector*: Builds distribution without TAC Connectors (default).
- *make tomcatConnector*: Builds distributions of Tomcat TAC Connector.
- *make tomcatConnector5.5*: Builds distributions of Tomcat5.5 TAC Connector.
- *make jettyConnector*: Builds distributions of Jetty TAC Connector.
- *make clean*: removes the Distribution and doc folders.

Installing

Please issue make install from the TAC directory.

Chapter 5. Using Together Application Cluster

Introduction

This document describes the steps required to install and configure Together Application Cluster.

For the most up-to-date information on TAC, see the Together Application Cluster Working Group [http://director.enhydra.org/index.html] .

System Requirements

TAC runs on Windows (NT, 2000, XP) and Linux (ix86 and S390).

NOTE : Together Application Cluster requires TCP/IP networking support. If you have not already done so, configure TCP/IP on at least one of your system's network interfaces.

Windows

- Hardware:
  - Intel 80386 or later CPU. (Pentium 200Mhz or faster recommended)
  - Sparc-compatible system higher 64 MB RAM (128 MB Recommended)
- Operating System: Windows NT 4.0, Service Pack 6 or later, Windows 2000, Service Pack 3 or later.
- Web servers:
  - IPlanet/Netscape Enterprise Server version 4.0 or higher
  - Microsoft Internet Information Server, version 4.0 or later
  - Apache Web Server version 1.3.9 or later (Apache2.x is currently unsupported)

Linux

- Hardware:
  - Intel 80386 or later CPU. (Pentium 200Mhz or faster recommended)
  - Sparc-compatible system higher 64 MB RAM (128 MB Recommended)
- Operating System: Version 2.2 kernel or later with support for recent versions of the glibc C library. (Redhat 6.1or later is recommended)
- Web servers:
  - IPlanet/Netscape Web Server version 4.0 or higher
  - Apache Web Server version 1.3.9 or later (Apache2.x is currently unsupported)

Preparation

Before installing Together Application Cluster, determine the following:

- The machines on which you will be running Aplication Server (Enhydra, Tomcat, JOnAS or Jetty).
- The applications you will be running on the Aplication Servers.
- For each application:
  - The name of the machine and the port number on which each instance of the application will run. The port number is the 'port' used when configuring a connection of type 'TAC' .
  - The URL prefix used to identify the application. (For example '/demoApp')
  - Whether the application requires session affinity. That is, whether requests for the same session to an application should always be routed to the application server instance that created the session. Unless your application supports distribution of its sessions among other instances, you should assume that session affinity is needed. If your application does not use session information at all, you do NOT need session affinity.
- The name of the web server instance on which you wish to configure Together Application Cluster.
- The TCP port of the web server instance on which you intend to configure Together Application Cluster. By default, this is port 80.
- For Netscape, determine:
  - The location of the obj.conf file for your web server instances. On Windows NT, this is typically:

    ```
    C:\Netscape\Server4\<instance>\config\obj.conf
    ```

    On Unix this is often

```
/usr/netscape/server4/<instance>/config/obj.conf
```

- Where to install the TAC files. On Windows NT, we recommend:

  ```
  C:\Netscape\TAC
  ```

  On Unix, we recommend:

  ```
  /usr/netscape/TAC
  ```

- For Apache, determine:
  - The location of the configuration files for Apache, typically

    ```
    /usr/local/apache/conf
    ```

  - The location of the executables directory for your Apache server, typically

    ```
    /usr/local/apache/bin
    ```

Configuring Your Application

Together Application Cluster connectors are set of application servers plugins that provides the connection between Together Application Cluster and Application Server. There are two kinds of connectors - that that implement an HTTP stack of their owner (called HTTP connectors) and those (called web server connectors) that tie Application server to an external web server like Apache (or IIS) or, in this case, to Together Application Cluster (web server plugin).

In Enhydra5.1.xx and prior version, every Enhydra Multiserver application (configured to handle request from TAC) has separate connection (on different port) to Together Application Cluster.

Unlike Enhydra5.1.xx, Jetty and Tomcat connectors, for Together Application Cluster use only one connection (port) for all applications hosted on that instance of server.

Together Application Cluster connector for Enhydra 5.x.xx and prior releases are part of Enhydra Server. Connectors for Tomcat and Jetty are distributed as separate binary modules inside TAC distribution.

Together Application Cluster connectors source are placed in separate modules:

- {$TAC.src}/tomcat - Connector for Tomcat versions prior of 5.5
- {$TAC.src}/tomcat5.5 - Connector for Tomcat5.5.xx and later versions.
- {$TAC.src}/jetty - Connector for Jetty.

Configuring Your Application In Enhydra5.x.xx

The following procedure assumes that you are using the Multiserver Administration console to add the connections for the application.

- Create Together Application Cluster connections for your application. - Bring up the MultiServer Administration page for your Enhydra server instance.
- Select your application in the application list.
- Click on the 'Connections' tab to manage connections for your application instance.
- Within the 'Connections' tab, click 'Create'. A 'new connection' dialog box will appear.
- In the dialog box, check 'TAC' as the connection type.
- Fill in the URL prefix and port. Also, if session affinity is NOT to be enabled, change the session affinity box from 'true' to 'false'.
- Click 'OK' to accept the new connection. In the main MultiServer admin page, click the 'Save State' floppy-disk icon to save your changes to the MultiServer configuration file. Click 'OK' to accept the rewrite of the configuration file.

Configuring Your Application In Tomcat 5.x.xx

Configuring Tomcat.5.x.xx to use with Together Application Cluster (versions up to 5.5.)

To use Together Application Cluster with Tomcat 5.0.xx You first need to put "tomcat-director.jar" on Tomcat classpath:

```
$CATALINA_HOME/server/lib
```

and then specify and configure connector (tomcat-director connector) to use with Together Application Cluster.

Configuration parameters of Together Application Cluster Connector are placed in Tomcats server.xml file:

```
$CATALINA_HOME/conf/server.xml)
```

in <service> section:

```
<Server ...>
      ...
```

```
     <Service name="Catalina">
          ...
          <Connector
className="org.enhydra.servlet.connectionMethods.EnhydraDirector.EnhydraDirectorConnectionMethod"
                    port="9000"
                    threadTimeout = "300"
                    clientTimeout = "30"
                    sessionAffinity = "true"
                    queueSize = "400"
                    numThreads = "200"
                    bindAddress = "(All Interfaces)"
                    authKey = "(Unauthenticated)"
          />
          ...
     </Service>
    ...
</Server>
```

Configuration parameters are:

1.Class that implements Together Application Cluster Connector.

```
className="org.enhydra.servlet.connectionMethods.EnhydraDirector.EnhydraDirectorConnectionMethod"
```

2.This instructs the Together Application Cluster Connector to directly listen for requests (Together Application Cluster Protocol) on the specified port.

```
port = <port>
```

3.The number of handler threads my be specified with:

```
numThreads = <num>   (optional)
```

4.The number of requests to queue (after accept, before processing) may be specified with:

```
queueSize = <num>    (optional)
```

5.The idle timeout period for a client connection, in seconds. This is the amount of time to block without activity.

```
clientTimeout = <num>   (optional)
```

6.The idle timeout period for a handler thread, in seconds. Shorter timeouts minimize the number of threads (memory) while slowing response time for bursts of activity.

```
threadTimeout = <num>   (optional)
```

7.IP address of Together Application Cluster host that is allowed to bind to this server instance.

```
bindAddress = <ipAdress> | "(All Interfaces)"  (optional)
```

8.The authKey .

```
authKey = <authKey> | "(Unauthenticated)"   (optional)
```

Configuring Enhydra 6.1.x to use with Together Application Cluster.

To use Enhydra 6.1.x with Together Application Cluster put "tomcat-director.jar" to:

```
{Enhydra-Lite-Install-Dir}/multiserver/server/lib
```

Since Enhydra v6.1.x. distributions uses Tomcat5.0.xx as servlet container, all details related to connector configurations in server.xml:

```
{Enhydra-Lite-Install-Dir}/multiserver/conf/server.xml
```

file are same as those described in previous section (Configuring Tomcat.5.0.xx to use with Together Application Cluster).

Configuring EnhydraEnterprise 6.2.x (Tomcat ver.) to use with Together Application Cluster.

To use EnhydraEnterprise with Together Application Cluster put "tomcat-director.jar" to:

```
{Enhydra-Enterprise-Install-Dir}/multiserver/lib/catalina/server/lib
```

Since EnhydraEnterprise v6.2.x distributions (Tomcat ver.) uses Tomcat5.0.xx as servlet container, all details related to connector configurations in server.xml:

```
{Enhydra-Enterprise-Install-Dir}/multiserver/conf/server.xml
```

file are same as those described in previous section (Configuring Tomcat.5.0.xx to use with Together Application Cluster).

Configuring Your Application with Tomcat 5.5.xx Connector

Configuring Tomcat.5.5.xx to use with Together Application Cluster

To use Together Application Cluster with Tomcat 5.5.xx You first need to put "tomcat5.5-director.jar" on Tomcat classpath:

```
$CATALINA_HOME/server/lib
```

and then specify and configure connector (tomcat5.5-director connector) to use with Together Application Cluster.

Configuration parameters of Together Application Cluster Connector are placed in Tomcats server.xml file:

```
$CATALINA_HOME/conf/server.xml)
```

in <service> section:

```
<Server ...>
     ...
     <Service name="Catalina">
          ...
          <Connector
protocol="org.enhydra.servlet.connectionMethods.EnhydraDirector.DirectorProtocol"
                     port="9003"
                     numThreads = "200"
                     queueSize = "400"
                     clientTimeout = "30"
                     threadTimeout = "300"
                     bindAddress = "(All Interfaces)"
                     authKey = "(Unauthenticated)"
                     sessionAffinity = "true"
                     scheme="http"
                     acceptCount = "50"
                     tcpNoDelay = "true"
                     enableLookups = "false"
                     redirectPort = "8443"
                     URIEncoding = "ISO-8859-1"
                     useBodyEncodingForURI = "true"
                     rebalanceIfSessionExpired="false"
          />
          ...
     </Service>
    ...
</Server>
```

Configuration parameters are:

1.Class that implements TAC Protocol.

```
protocol="org.enhydra.servlet.connectionMethods.EnhydraDirector.DirectorProtocol"
```

2.This instructs the Together Application Cluster Connector to directly listen for requests (Together Application Cluster Protocol) on the specified port.

```
port = <port>
```

3.The number of handler threads my be specified with:

```
numThreads = <num>   (optional)
```

4.The number of requests to queue (after accept, before processing) may be specified with:

```
queueSize = <num>    (optional)
```

5.The idle timeout period for a client connection, in seconds. This is the amount of time to block without activity.

```
clientTimeout = <num>   (optional)
```

6.The idle timeout period for a handler thread, in seconds. Shorter timeouts minimize the number of

threads (memory) while slowing response time for bursts of activity.

```
threadTimeout = <num>   (optional)
```

7.IP address of Together Application Cluster host that is alowed to bind to this server instance.

```
bindAddress = <ipAdress> | "(All Interfaces)"  (optional)
```

8.The authKey .

```
authKey = <authKey> | "(Unauthenticated)"   (optional)
```

9.The Request Scheme is parameter name of the protocol you wish to have returned by calls to request.getScheme().

```
scheme = <scheme> (optional)
```

10.The maximum queue length for incoming connection requests when all possible request processing threads are in use.

```
acceptCount  = <num> (optional)
```

11.If set to true, the tcpNoDelay option will be set on the server socket, which improves performance under most circumstances.

```
tcpNoDelay  = <true> (optional)
```

12.Set to true if you want calls to request.getRemoteHost() to perform DNS lookups in order to return the actual host name of the remote client. Set to false to skip the DNS lookup and return the IP address in String form instead (thereby improving performance). By default, DNS lookups are disabled.

```
enableLookups  = <false> (optional)
```

13.If this Connector is supporting non-SSL requests, and a request is received for which a matching <security-constraint> requires SSL transport, Catalina will automatically redirect the request to the port number specified here.

```
redirectPort  = <num> (optional)
```

14.This specifies the character encoding used to decode the URI bytes.If not specified, ISO-8859-1 will be used.

```
URIEncoding  = <URIEncoding> (optional)
```

15.This specifies if the encoding specified in contentType should be used for URI query parameters, instead of using the URIEncoding.

```
useBodyEncodingForURI  = <false> (optional)
```

16.This parameter specifies when the session expired, TAC will rebalance client to another server ac-

cording to round robin algorithm. If set to false (default value) TAC will send client to the same server.

```
rebalanceIfSessionExpired = <false> (optional)
```

Configuring Enhydra 6.2.x/6.3.x/6.4.x/6.5.x to use with Together Application Cluster.

To use Enhydra with Together Application Cluster put "tomcat5.5-director.jar" to:

```
{Enhydra-Install-Dir}/multiserver/server/lib
```

Since Enhydra v6.2.x distributions uses Tomcat5.5.x as servlet container, all details related to connector configurations in server.xml:

```
{Enhydra-Install-Dir}/multiserver/conf/server.xml
```

file are same as those described in previous section (Configuring Tomcat.5.5.x to use with Together Application Cluster).

Configuring Enhydra Enterprise 6.5.x (Tomcat ver.) to use with Together Application Cluster.

To use Enhydra with Together Application Cluster put "tomcat5.5-director.jar" to:

```
{Enhydra-Install-Dir}/multiserver/lib/catalina/server/lib
```

Since Enhydra Enterprise v6.5.x distributions uses Tomcat5.5.x as servlet container, all details related to connector configurations in server.xml:

```
{Enhydra-Install-Dir}/multiserver/conf/server.xml
```

file are same as those described in previous section (Configuring Tomcat.5.5.x to use with Together Application Cluster).

Configuring Your Application with Jetty Connector

Configuring EnhydraEnterprise 6.2.x/6.3.x/6.5x (Jetty ver.) to use with Together Application Cluster.

To use EnhydraEnterprise with Together Application Cluster put "jetty-director.jar" to:

```
{Enhydra-Enterprise-Install-Dir}/multiserver/lib/jetty/lib
```

To avoid "null path redirection", and enable Together Application Cluster to properly performe request dispatching we need to set "RedirectNullPath" parameter to "false". This parameter need to be placed in **"web-jetty.xml"** This file must be in org.mortbay.xml.XmlConfiguration format and if found in the WEB-INFdirectory of a web application, it is applied to the WebApplicationContext instance. It is typically used to change non standard configuration. If this file don't exists we need to create it.

eg.

```
<!DOCTYPE Configure PUBLIC
                           "-//Mort Bay Consulting//DTD Configure 1.2//EN"
                           "http://jetty.mortbay.org/configure_1_2.dtd">
<Configure class="org.mortbay.jetty.servlet.WebApplicationContext">
  <Set name="RedirectNullPath" type="boolean">false</Set>
</Configure>
```

EnhydraEnterprise v6.2.2/v6.3.x distributions (Jetty ver.) uses Jetty5.1.1 as servlet container, all details related to connector configurations is in jetty5.xml:

```
{Enhydra-Enterprise-Install-Dir}/multiserver/conf/jetty5.xml
```

in <Configure> section:

```
<Call name="addListener">
    <Arg>
      <New class="org.enhydra.servlet.connectionMethods.EnhydraDirector.EnhydraListener">
      <Set name="Port">9003</Set>
      <Set name="MinThreads">10</Set>
      <Set name="MaxThreads">100</Set>
      <Set name="SessionAffinity">true</Set>
      <Set name="ClientTimeout">300</Set>
      <Set name="BindAddress">(All Interfaces)</Set>
      <Set name="AuthKey">(Unauthenticated)</Set>
      <Set name="acceptCount">50</Set>
      <Set name="tcpNoDelay">true</Set>
      <Set name="enableLookups">true</Set>
      <Set name="threadTimeout">30</Set>
      <Set name="ConfidentialPort">SSLPORT</Set>
      <Set name="URIEncoding">ISO-8859-1</Set>
      <Set name="useBodyEncodingForURI">true</Set>
     </New>
    </Arg>
  </Call>
```

Configuration parameters are:

1.Class that implements Together Application Cluster Connector.

```
class="org.enhydra.servlet.connectionMethods.EnhydraDirector.EnhydraListener"
```

2.This instructs the Together Application Cluster Connector to directly listen for requests on the specified port.

```
port = <port>
```

3. Minimum threads in threadpool for listener.

```
MinThreads = <MinThreads>
```

4. Maximum threads in threadpool for listener.

```
MaxThreads = <MaxThreads>
```

5.The idle timeout period for a client connection, in seconds. This is the amount of time to block without activity.

```
clientTimeout = <num>   (optional)
```

6.IP address of Together Application Cluster host that is alowed to bind to this server instance.

```
bindAddress = <ipAdress> | "(All Interfaces)"  (optional)
```

7.The authKey .

```
authKey = <authKey> | "(Unauthenticated)"   (optional)
```

8.The maximum queue length for incoming connection requests when all possible request processing threads are in use.

```
acceptCount  = <num> (optional)
```

9.If set to true, the tcpNoDelay option will be set on the server socket, which improves performance under most circumstances.

```
tcpNoDelay  = <true> (optional)
```

10.Set to true if you want calls to request.getRemoteHost() to perform DNS lookups in order to return the actual host name of the remote client. Set to false to skip the DNS lookup and return the IP address in String form instead (thereby improving performance). By default, DNS lookups are disabled.

```
enableLookups  = <false> (optional)
```

11.The idle timeout period for a handler thread, in seconds. Shorter timeouts minimize the number of threads (memory) while slowing response time for bursts of activity.

```
threadTimeout = <num>   (optional)
```

12.If this Connector is supporting non-SSL requests, and a request is received for which a matching <security-constraint> requires SSL transport, Jetty will automatically redirect the request to the port number specified here.

```
ConfidentialPort  = <num> (optional)
```

13.This specifies the character encoding used to decode the URI bytes.If not specified, ISO-8859-1 will be used.

```
URIEncoding  = <URIEncoding> (optional)
```

14.This specifies if the encoding specified in contentType should be used for URI query parameters, instead of using the URIEncoding.

```
useBodyEncodingForURI  = <false> (optional)
```

Together Application Cluster Status Page

To see Together Application Cluster Status Page you need to proper set "Status" element in **"enhydra_director.conf"** file.

```
<Status prefix="/status">
    <Restrict server="127.0.0.1"/>
    <Restrict client="127.0.0.1"/>
</Status>
```

The "Status" element tells TAC to create a "status" prefix and to respond to requests for that prefix with a dump of the load balancing scoreboard and/or other stats. Since v5.1.18 this feature are available for both Windiws/IIS and Linux/Apache version of Together Application Cluster.

Since v6.3.1 release rebalance algorithm is changed to support web server (TAC) clustering. Implication of new rebalance algorithm is that 'rebalance' operation from status page can be performed only when TAC is connected to Enhydra Server v5.3.x/v6.3.x or newer versions, or to Connector(Tomcat, Tomcat5.5, Jetty) v6.3.x or newer versions.

Actions and information's from status page are placed in several groups and levels:

- **Actions and information's related to status page and TAC global settings.**

  - **Refresh**

    "Refresh" button sets autorefresh rate for "StatusPage" to value from input field placed in same row (value represents seconds).

    Click on "Set Refresh" button if input field is empty will disable refreshing of status page (same if value in input field is less or equal 0 - zero)

  - **Reset Weigh**

    "Reset Weight" button sets weights off all servers to same value (default is 1), this will efectivly

disable "weight" based balancing in Together Application Cluster.

- **Clear History**

  "Clear History" button action will clear "Session history" on status page for all servers.

- **Disable/Enable All Servers**

  "Disable" button will disable redirection from TAC to all applications that are hosted on all servers. If servers are alredy disabled then action have no effects.

  "Enable" button will enable redirection from TAC to all applications that are hosted on all server. If servers are alredy enabled then action have no effects.

- **Application server controller actions and information's**

  Application server controller section is optional part of TAC status page. This section is part of status page only if in **"enhydra_director.conf"** file is defined application with predefined name called "ApplicationServerController".

```
<Application prefix="/ApplicationServerController">

    <AppServer host="192.168.0.16"    port="9101"  alias="BiliTheKid"/>
    <AppServer host="192.168.0.32"    port="9003"  alias="TalicniTom"/>

    <Restrict server="127.0.0.1"/>
    <Restrict client="127.0.0.1"/>

</Application>
```

  Only those applications servers that are defined inside "ApplicationServerController" section, can be "Rebalance"-d

```
<AppServer host="<ServerHostName>" port="<ServerPort>" alias="<ServerAlias>"/>
```

  Where parameters are:

  - host="<ServerHostName>" (mandatory)

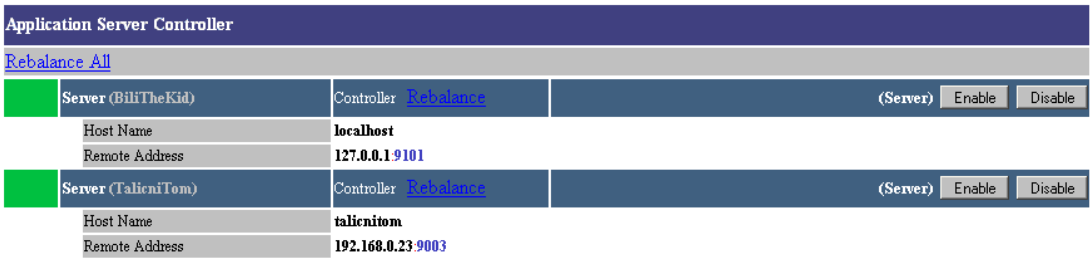
    - Name or IP adress of application server.

  - port="<ServerPort>" (mandatory)

    - Any port on application server with running TAC Connector (application, defined on application server).

  - alias="<ServerAlias>" (mandatory)

    - User defined name of server, can be any valid name (no space and spec. chars), but it is mandatory.



Application server controller actions are:

- **Rebalance all**

  Performs rebalance of all servers.

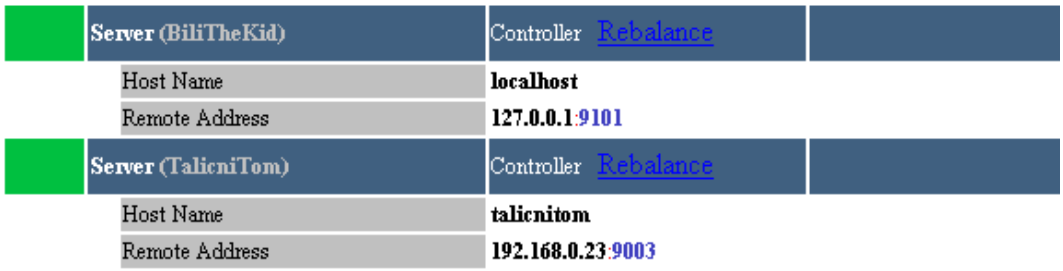
- **Disable/Enable Server**

  "Disable" button will disable redirection from TAC to all applications that are hosted on that server. If server is alredy disabled then action have no effects.

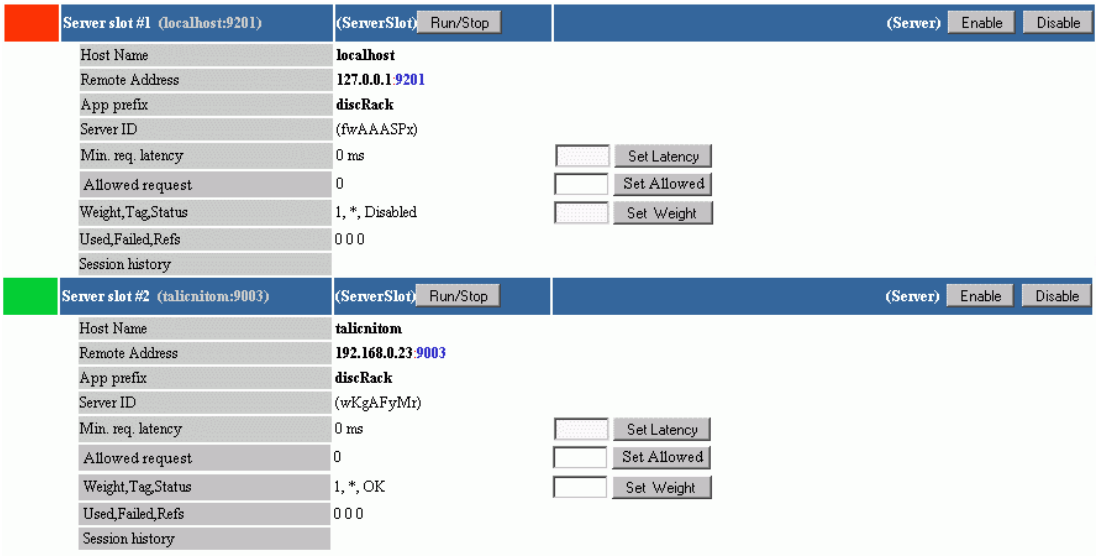
  "Enable" button will enable redirection from TAC to applications that are hosted on that server. If server is alredy enabled then action have no effects.

- **Rebalance Server**

  Pressing "Rebalance" link will cause a lost of all users session on selected application server and redistribution of users sessions across all avaliable servers by "round-robin" algorithm.



- **Application servers slots actions and informations.**



- **Run/Stop**

  "Run/Stop" button action will cause enabling and disabling of server slot (ServerSlot = Server+Application).

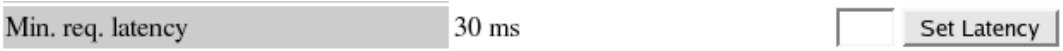
  If server slot is enabled then they will have green markup.

  if server slot is disabled then markup is red.

  Together Application Cluster will never attempt to send request to disabled server slot - even if they are explicitly asked from user application (thought "alias" parameter).

- **Set Latency**

  (ServerSlot) "Set Latency" action will set minimum time in milliseconds between two requests in the same session. If time between two requests is less then parameter value, second request will be rejected, this feature is added to avoid "F5" problem . If this parameter is set to zero(0) then this feature is disabled (default). This parameter also exists in TAC's configuration file.



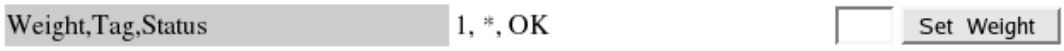
- **Set Allowed**

  (ServerSlot) "Set Allowed" action will set number of allowed requests in time specified with Set Latency button. This will only have effect if latency is enabled. If this parameter is set to zero(0) then this feature is disabled (default). This parameter also exists in TAC's configuration file.

- **Set Weight.**

  Under light load, the servers will be selected in round-robin order. However, under higher load, when more than one request must be dispatched to a server, a weight factor is used to determine the proportional number of simultaneous active connections each application server gets. This allows newer, faster servers to be added, and to allow administrators to assign higher loads to such servers such that each server is loaded in rough proportion to its capacity.

"Set Weight" allows any unsigned 32 bit integer number to be chosen as a weight without danger of causing unpredictable behavior due to integer overflow on multiplication. Also, the proportional relationships between the server weights is preserved.



- **Server Disable Enable**

  "Disable" button will disable redirection from TAC to all applications that are hosted on same Server like current ServerSlot. If server is already disabled then action have no effects.

  "Enable" button will enable redirection from TAC to applications that are hosted on same server like current ServerSlot.. If server is already enabled then action have no effects.

Chapter 6. Together Application Cluster Configuring and Caching

Configuring Apache (and IIS) front-end to serve static objects.

The basic idea is to configure the Apache front-end web server with Together Application Cluster to act as a resource cache for static object such as gif, jpeg pictures and css files in order to decrease enhydra app servers load (decreasing number of requests). Apache might be used to serve pre-configured static objects (gifs, jpegs, css) from its local storage. In other words, small static objects, such as small gifs, jpgs and css, could be placed within web server#s local storage (local disk) and sent to the clients when it is requested.

The advantage of this approach it that clients requests for static objects are accomplished immediately by the Apache itself. No external app server is involved for this task. Minimum response time, minimum resource usage are benefits. Drawback is that all resource files (gifs, jpegs, etc.) must be located within original app server, extracted from jar archives and deployed to the web server's local disk. As the Apache is unaware of any modification on app server side, any modification of resources in an application produces the need to repeat the previous steps.

In order to set up the Apache web server to serve static objects, the following actions are necessary:

- Locating and extracting the resources
  - The jar files that include static objects (*.gif, *.jpg, *.css) that will be copied to the web server (somewhere below the document root directory, e.g. <Document Root>/graphic/)
- Configuring Apache (httpd.conf)
  - mod_rewrite should be loaded and added before mod_enhydra_director (mod_enhydra_director should be loaded with LoadModule directive after mod_rewrite and it should be added with AddModule directive before mod_rewrite)
  - rewriting engine should be enabled (RewriteEngine on)
  - appropriate rewriting rule(s) should be added (e.g. RewriteRule ^/graphic/(.*) /graphic/$1 [L])

Example : The TAC is configured in a way that url prefix "/app" is connected with an enhydra app server. Let us assume that all static application resources are contained below url /app/graphic/ (e.g. / app/graphic/Enhydra.gif). Than we locate jar file that contains /graphic/*. Let us assume that the Apache#s DocumentRoot is /var/www/. We extract files from the jar in to the directory / var/www/app-graphic/. Than we introduce the following url rewriting rule for Apache:

```
RewriteRule ^/graphic/(.*) /app-graphic/$1
```

After that, a client requests url http://ourserver/app/graphic/Enhydra.gif . This url will match our rewrit-

ing rule and will be translated into "/app-graphic/Enhydra.gif". As mod_rewrite is consulted by the Apache before mod_enhydra_director, this url will not be considered by the TAC and will be served by the Apache itself.

IIS might be used for this task in a similar way. There are many implementations of IIS third-party extension (commercial, license-free, ...) that could perform url rewriting.TAC Filter has a very basic url rewriting function that also can be used for this task. This TAC's feature can be controlled by its config files. There are the following params in enhydra_director.conf:

- <Options urlrewrite="<on/off>"/> Controls url rewriting process.
- <Options urlrewritepattern="<url suffix pattern list>"/> Defines suffix based mechanism for matching urls that will be rewritten (e.g. #.gif .jpg res.jpeg# will match all url that ends with .gif, .jpg or #res.jpeg#)
- <Options urlrewriteprefix="<new url prefix>"/> Specifes a prefix that will be add in front of the original url.

Example : Enhydra_director.conf:

- <Options urlrewrite="on"/>
- <Options urlrewritepattern=".gif"/>
- <Options urlrewriteprefix="/app-resource"/>

An URL http://ourserver/app/graphic/Enhydra.gif , for example, matchs, in first place, the condition that it is an URL for our app server (/app prefix), and, in second place, the url rewriting condition (it ends with ".gif"). It will be translated into /app-resource/graphic/Enhydra.gif and will be further processed by the IIS as a request for local resource.

Together Application Cluster Cache (IIS only)

TAC Cache is built-in simple build-in cache for TAC. It is implemented only for IIS TAC. Its purpose is to cache small static object (such gif, jpeg, css) and to decrease app server's load. It is not designed as a general purpose caching system.

The cache is controlled through the options within TAC Config [../IIS/enhydra_director.conf.cache_example] settings (TAC configuration file). The following is description of these options:

- *<Options debug=0x01000000/>* - enables TAC cache normal debug output. If not present, normal debug cache messages are disabled. This option is optional.
- *<Options debug=0x02000000/>* - enables TAC cache verbose debug output. If not present, verbose debug cache messages are disabled. This option is optional.
- *<Options cache="on|off"/>* - the "cache" param tells the TAC to enable or disable caching. This field is optional. The caching is disabled by default.

- *<Options cachepattern="<uri suffix pattern list>"*> - the "cachepattern" param is a list of URI suffix patterns that will be match against client requests to decide which request should be cached. This field is optional. If omitted the default is ".gif .jpg". If cache is disabled this option is ignored.

- *<Options cachesize="<an integer value>"*> - the "cachesize" param tells the TAC the max size of memory in KBytes that could be allocated by the cache system (if enabled). This field optional. If omitted and if the cache is enabled the defualt size is 512 (512KBytes).

- *<Cache prefix="<cache flush url>"*> - specifes the url that triggers cache flushing actions. When this url is requested, if the cache is enabled, all object stored in cache will be wiped out. This param is very similar to the status param. The <Restrict> tag can also be included in order to restrict access to this url. This tag is optional and, if not specifed, the default url would be set to "/cacheflush".

TAC Cache example configuration file :

```
<EnhydraDirectorConfig> <!-- This tag is required. Do not remove. -->

    <!-- <Options debug="0x00000001" /> --> <!-- Function Entry -->
    <!-- <Options debug="0x00000002" /> --> <!-- Function Exit -->
    <!-- <Options debug="0x00000004" /> --> <!-- State Machine Entry -->
    <!-- <Options debug="0x00000008" /> --> <!-- State Machine Exit -->
    <!-- <Options debug="0x00000010" /> --> <!-- Sparse Debugging -->
    <!-- <Options debug="0x00000020" /> --> <!-- Normal Debugging -->
    <!-- <Options debug="0x00000040" /> --> <!-- Verbose Debugging -->
    <!-- <Options debug="0x00000080" /> --> <!-- Very Verbose Debugging -->
    <!-- <Options debug="0x00000100" /> --> <!-- Socket Debugging -->
    <!-- <Options debug="0x00000200" /> --> <!-- Verbose Socket Debugging-->
    <!-- <Options debug="0x00000400" /> --> <!-- HexDump All Outgoing Packets-->
    <!-- <Options debug="0x00000800" /> --> <!-- HexDump All Incoming Packets -->
    <!-- <Options debug="0x80000000" /> --> <!-- Fake AppServer Connections-->
    <!-- <Options debug="0x01000000" /> --> <!-- Normal cache debuging -->
    <!-- <Options debug="0x02000000" /> --> <!-- Verbose cache debuging -->

    <Options cachestatus="on"/>     <!--To enable cache info on status page -->
    <Options cache="on"/>
    <Options cacheentries="100"/>
    <Options cachepattern=".jpg .gif.css"/>

    <Application prefix="/proba">
       <AppServer host="localhost" port="40000" />
       <CacheCtrl cache="on"/>
    </Application>

    <Application prefix="/calc">
       <AppServer host="localhost" port="41000" />
       <CacheCtrl cache="off"/>
    </Application>

    <Status prefix="/status">
       <Restrict server="127.0.0.1" />
       <Restrict client="127.0.0.1" />
    </Status>

    <Cache prefix="/cacheflush">
       <Restrict client="127.0.0.1"/>
    </Cache>

</EnhydraDirectorConfig> <!-- This tag is required. Do not remove. -->
```

Chapter 7. Together Application Cluster Tuning

NOTE: This part of document is a bit terse at this point, but it contains some important information for those building large, high-throughput internet sites using Together Application Cluster. During load testing we discovered that a great many connections are left in the TIME_WAIT state after closure. This document describes how to greatly reduce the impact of these lingering connections.

The TIME_WAIT traffic jam

The TCP protocol used by Together Application Cluster has in its design a "cooling off period" for connections that have recently been closed. This is the infamous "TIME_WAIT" state. It is demanded by the TCP specification in order to be "sure" that stray data from an old defunct connection on the same ports does not find its way onto the new connection. The specification RFC 793 "Transmission Control Protocol" states that this should be four minutes.

The problem is that the four minute wait is far too conservative for high performance transaction-oriented server implementations. Suppose we have a server that is in steady state receiving and processing 200 requests per second. The current implementation of TAC opens a new connection for each transaction to the back-end server, much like Apache JServ. This means that 200 connections per second on average are going into the "TIME_WAIT" state when they finish. After four minutes, older TIME_WAIT connections begin to disappear as expected. This means that there will, on average, be about four minutes' worth of TIME_WAIT connections at any given time. Doing the math, we get 200 * 4 * 60 = 48000 connections!

The problem here is that the maximum number of connections that any one server can have in any state is 65535. This is because the TCP specification (RFC 793) specifies that the "port" which defines the local endpoint for the connection is only a 16 bit unsigned integer, and the largest such number is 65535. Worse, ports 0-1024 are usually reserved for well-known services and are not available as dynamically assigned (or "ephemeral") ports. In the best of all cases, we have 64511 ports available. With a 4 minute TIME_WAIT, we can compute 64511 ports / (4 minutes * 60 seconds per minute) = 268 conn. per sec. This means that no matter how fast your machine, connections will start failing at 268 connections per second average load. Actually the failures will usually starte a little before that threshold is reached. The situation here is similar to having a brand new Ferrari on a crowded freeway at rush hour. You might be able to do 180 MPH, but you're still going no faster than 20.

The current (quick) solution

The current solution to this problem is to reduce the amount of time old connections spend in TIME_WAIT. Some purists will sound off that doing so is a violation of RFC 793. Technically this is true, but I argue that on modern networks, this violation is a very venial sin, compared to the performance improvements it gives. Here's why.

The purpose of TIME_WAIT is to prevent lingering stale duplicates from old connections from finding their way into an existing connection, causing data corruption. Back in the days of 9600 baud X.25 networks, where a packet could very well circulate around for many tens of seconds among remote routers, four minutes was a good conservative number. The Web had not been invented yet, and if it had, no one would have accused a PDP/11 of being capable of sustaining 200 or more transactions per second anyhow.

On a typical Enhydra high throughput server, all of the back-end TAC connections will be taking place on a fast 100Base-T or Gig ethernet subnet. In such a situation, a packet is either going to make it to its destination or die within milliseconds if not microseconds, unless severe routing misconfigurations are present.

On the Internet side of the server, connections are coming from all different places. It is quite possible that stray packets could linger for quite some time. For a single client-server pair, the four minute TIME_WAIT is not going to be a problem and may be reasonable if the connection is being reestablished for each transaction. However, a web server is a server with many different clients. HTTP requests from a single client are in most cases piggybacked into one TCP session using HTTP "Keep-Alive" semantics. Most of the new connections are coming from different clients, which will prevent the 'stray segment' problem anyhow.

The bottom line is, for a stray segment to make it into a new connection, the following conditions must ALL be met.

- The local port of the new connection MUST match the local port of the old connection.
- The local IP address of the new connection MUST match the local IP address of the old connection.
- The remote port of the new connection MUST match the remote port of the old connection.
- The remote IP address of the new connection MUST match the remote IP address of the old connection.
- The TCP sequence number in the old segment packet MUST such that the new connection will think it is valid. This is highly unlikely, even in the already unlikely event that the other four conditions have been met.

Given these conditions, it should be very safe to set a TIME_WAIT interval that is quite low. We recommend that you choose a TIME_WAIT value that keeps the average number of outstanding TIME_WAIT connections below 30000. That is, TIME_WAIT_SECS * CONN_PER_SEC < 30000. A value of 60 seconds will allow a theoretical maximum of 1072 connections per second, and up to 500 connections per second without going over 30000 TIME_WAIT lingerers. If you need more, then lower TIME_WAIT even more. I have seen thirty seconds quoted as a good value, and have heard of one vendor suggesting one second. I doubt you will have problems if you choose either of these values for a typical Together Application Cluster server setup.

How to fix TIME_WAIT on Linux (RedHat 6.1)

On RedHat 6.1, run the following command:

```
/sbin/sysctl net.ipv4.ip_local_port_range
```

If you haven't already changed this value, it is 1024 to 4999. This is the range of ephemeral ports available by default, and as you can see it is too low for a high performance server. Use the command

```
/sbin/sysctl -w net.ipv4.ip_local_port_range="1024 65535"
```

to change the range to 1024-65535. By default the TIME_WAIT interval on Redhat Linux 6.1 is one minute. This will allow up to 1072 connections per second and 500 per second without exceeding 30000 TIME_WAITS. If you need more, you have to change a kernel header file and recompile your Linux kernel. Assuming /usr/src/linux is a symlink to your current kernel, the file to change is:

```
/usr/src/linux/include/net/tcp.h
```

Change the line:

```
#define TCP_TIMEWAIT_LEN (60*HZ)
```

to:

```
#define TCP_TIMEWAIT_LEN ( * HZ)
```

where:

```
TIMEWAITSECS is the number of seconds for connections to remain in the TIME_WAIT state.
```

Example to change the TIME_WAIT period to 15 seconds:

```
#define TCP_TIMEWAIT_LEN (15 * HZ)
```

Once you have saved 'tcp.h', do a COMPLETE rebuild and install of the Linux kernel following the usual Linux kernel rebuild procedure.

How to fix TIME_WAIT for Windows NT 4.0

I haven't noticed the TIME_WAIT issue as a problem during our load testing, in NT, but it is possible you'll run into it. The way to set the TIME_WAIT interval in WinNT is to edit the registry key under HKEY_LOCAL_MACHINE:

```
SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\TcpTimedWaitDelay
```

If this key does not exist you can create it as a DWORD value. The numeric value is the number of seconds to wait and may be set to any value between 30 and 240. If not set, WinNT defaults to 240 seconds for TIME_WAIT. I've also heard that Microsoft put undocumented hooks into WinSock to allow IIS to get around the TIME_WAIT problem, but I have not personally substantiated this.

How to fix TIME_WAIT for Compaq Tru64 Unix

This information was obtained from the Compaq Tru64 kernel tuning FAQ [http://tru64unix.compaq.com/faqs/publications/base_doc/DOCUMENTATION/V50A_HTML/ARH9GATE/TITLE.HTM]. This is a very useful page that describes numerous tuneable parameters for the Tru64 OS.

To change the TIME_WAIT and Port Range at runtime: (as root of course)

The easy way: Run 'Kernel Tuner'. From root's CDE Desktop...

CDE Menu Bar --> Manager Icon --> System_Admin --> MonitoringTuning --> Kernel Tuner --> 'inet' subsystem

For TIME_WAIT period, change the 'tcp_msl' attribute to the number of seconds. Let's say 30.

For port range, change 'ipport_userreserved_min' to 1024, and 'ipport_userreserved' to 64511. This will allow all ports from 1024 to 65534 to be used.

You should be able to make these changes 'permanent' meaning that they will take effect automatically even if the machine is rebooted.

The hard way #1: (At runtime manually every time you reboot)

Arrange for the following script to be run by or as an '/etc/rc3.d' script at boot time.

```
#!/bin/sh

# Change TIME_WAIT period to 30 seconds. Factory default is 60.

/sbin/sysconfig -r inet tcp_msl=30

# Change Port Range to 1024-65534 (Total 64511 available ephemeral ports)

/sbin/sysconfig -r inet ipport_userreserved_min=1024

/sbin/sysconfig -r inet ipport_userreserved=64511
```

The hard way #2: (Permanent using the sysconfigdb command)

To permanently set these values every time you boot, you use the /sbin/sysconfigdb command to read a stanza file into the bootup config. To change the TIME_WAIT, create a file called 'msl.txt' that contains:

```
inet:
tcp_msl = 30
```

Then run '/sbin/sysconfigdb -a -f msl.txt' to register the change. Finally, reboot.

To change the ephemeral port range, create 'port.txt' as follows:

```
inet:
ipport_userreserved_min = 1024
ipport_userreserved = 64511
```

Then run '/sbin/sysconfigdb -a -f port.txt' to register the change. Finally, reboot.

Chapter 8. Together Application Cluster Configuration Parameters

This document describes TAC configuration file for use with the TAC module. The elements definied in this configuration file tell TAC the location of its resources and servers participating in the load balancing protocol (ie. multiservers listening for request with the EnhydraConnectionMethod).

The TAC Config is composed of none, one or more "Options" element; one or more "Application" elements; none, one or more "Status" elements; and none or one "Cache" element.

The "Options" element contains the following attributes:

Atribute	**Description**	**Required**
daemon	Parameter is the location of the TAC daemon program.	This field is optional and is ignored on multithreaded platforms where the daemon is not used.
daemoninterval	Parameter tells the TAC how often the scoreboard daemon should check the scoreboard.	This fild is optional and is ignored on multithreaded platforms.
daemontimeout	Parameter tells the TAC daemon its idle timeout value.	This fild is optional and is ignored on multithreaded platforms.
daemondebug	Parameter tells the TAC daemon its debug mask.	This field is optional.
debug	Parameter tells the TAC module its debug mask.	This field is optional.
retrytime	Parameter tells the TAC daemon how long to keep trying to contact a server if it is down and session affininty is in effect for a session.	This field is optional.
sessionhistory	Number of entries in sessionid history list per application server. Value of this parameter must be integer between 0 and 100. Default value is 0, TAC works without sessionid list. Sessionid list is displayed on status page.	This field is optional.
minnextrequesttime	Minimum time in miliseconds between two requests in the same session. If time between two requests is less then parameter value, second request will be rejected. Default value is 0 (disabled).	This field is optional.
noofallowedrequests	Number of allowed requests in time interval specified with minnextrequesttime. Default value is 0	This field is optional.
cache	Parameter tells the TAC module to enable or disable caching globally (module-wide). The caching is disabled by default.	This field is optional.

Atribute	Description	Required
cacheentries	param specifies the number of cache table entries that will be allocated for the caching system. The default value is 100. If set to 0, the cache system will be disabled. As the cache system uses the linear search algorithm for the cache lookup, in order to keep cache performances, this param should not be to large.	This field is optional.
cachepattern	Parameter is a list of URI suffix patterns that will be match against client requests to decide which request should be cached. If omitted the default is ".gif .jpg .css".	This field is optional.
cachestatus	Parameter controls viewing cache status at TACstatus page (/status). Displaying status is disabled by default.	This param is optional.
urlrewrite	Parameter controls the TAC url rewriting process. By default it is "off". If enabled the TAC filter compares app suffixs against "urlrewritepattern" list. If match found, the original uri is modified adding "urlrewriteprefix" and proceeded back to IIS. "urlrewriteprefix" default is an empty string. "urlrewritepattern" is ".gif .jpg" by default.	This field is optional.
urlrewriteprefix	Parameter - see "urlrewrite". The default value is an empty string.	This param is optional.
urlrewritepattern	Parameter - see "urlrewrite". The default value is ".gif ..jpg".	This param is optional.

The "Application" element tells the TAC daemon what applications are participating in load balancing. Each Application contains one or more "AppServer" elements, none or one CacheCtrl element and has the following attributes:

Atribute	Description	Required
prefix	Parameter assigns a URI prefix to the application and is used for identifying requests and routing the request to the associated application.	This field is required.

The "Status" element tells TAC to create a "status" prefix and to respond to requests for that prefix with a dump of the load balancing scoreboard and/or other stats. Restrict elements may be included to limit the scope of the status URL to specific local IP addresses, ports, or virtual hosts. The following attribute is required:

Atribute	Description	Required
prefix	Attribute assigns a URL prefix to the status handler.	This field is required.

The "Cache" element tells TAC to create a "cache" prefix and to respond to requests for that prefix with flushing the cache entries. Restrict elements may be included to limit the scope of the status URL to specific local IP addresses, ports, or virtual hosts. The following attribute is required:

Atribute	**Description**	**Required**
prefix	The "Prefix" attribute assigns a URL prefix to the status handler. If cache element is omitted, cache flush prefix is "/cacheflush".	This field is required.

The "AppServer" element describes the servers participating in load balancing. Each AppServer must be configured with an EnhydraConnectionMethod listening as described by the following attributes:

Atribute	**Description**	**Required**
host	Parameter gives the domain name or IP number of the server.	This field is required.
port	Parameter gives the port number that the EnhydraConnectionMethod is listening to.	This field is required.
weight	Parameter is an integer value that wieghs the performance of this machine as a ratio with that of all the remianing AppServer specified.	This field is optional.
auth	Parameter specifies the authentication type required.	This field is optional.
alias	Parameter specifies alias for server.Value of this attribute can be used to specify from application which application server will be used to serve request. This can be done by setting query parameter DIRECTTOSERVER to value of alias attribute of wanted server.Default value is host:port for specific server. IMPORTANT NOTE: If wanted server is not available,than TAC will not find other server.It will throw SERVER ERROR exception. So,if in request is specified DIRECTTOSERVER parameter,than or that server will serve request or error will occure. -e.g http://localhost/MyApp/StartPO.po?DIRECTTOSERVER=serverB In this case,server with alias 'serverB' will serve request.If this server is not available,error will occure. This is implemented only for IIS.	This field is optional.
startMode	Defines startup status of connection. Values are 'active' and 'inactive'. Default value is 'active'.	This field is optional.
hotStandByHost	Defines host that has backup server runing on. If this application server	This field is optional.

Atribute	Description	Required
	fails or if connection to this server is manualy disabled, connection to this server will automaticaly be put in active mode	
hotStandByPort	Defines port on hotStandByHost machine.	This parameter is required only when hotStandByHost parameter is defined.
hotStandByInactivate	Describes behaviour of hot stand by connection. Values are 'true' and 'false'. If set to 'true' and original connection is established again, hot stand by connection will be set to 'inactive' otherwise it will remain active. Default value is true.	This parameter is optional and counted only if hotStandByHost parameter is defined.
hotStandByMode	This parameter allows selection of hot stand by modes. Possible values are "normal", "noMasterSlave" and "noAutoSlaveShutdown". *Normal* mode means that if this application server fails or if connection to this server is manualy disabled, connection to this server hot stand by will automaticaly be put in active mode. After 'Fail retry' timeout, new request will again go to this server and hot stand by will be deactiveted (depending on hotStandByInactivate parameter). *noMasterSlave* differs from normal mode in a way that these two servers are equal and only one of them is active at time. If connection is lost, there will be no retry time, but this server connection will be deactivated and hot stand by connection will be activated. *noAutoSlaveShutdown* is similar to noMasterSlave mode. Only difference is behaviour of slave server. If slave server fails, it will not automatically be disabled and it's master will not be automatically enabled. In that situation it will behave like normal server (as if he doesn't have hot stand by).	This parameter is optional and counted only if hotStandByHost parameter is defined. If ommited, default value is "normal". If this parameter is set to "noMasterSlave" or "noAutoSlaveShutdown", parameters startInactive and hotStandByInactivate will not be counted
noOfAllowed	This param will act as noofallowedrequests but only for specific appserver. This param has higher priority than noofallowedrequests.	This parameter is optional.

The "Restrict" element describes a restriction of an application to a particular IP/Port or VirtualHost/Port. Semantically, only one of "server", "virtualserver", or "client" may specified in any one "Restrict" entry. Also Either the port or one of the above three params or both must be specified. That is, an "empty" Restrict entry is not allowed.

Atribute	Description
server	Parameter gives the domain name or IP number of the server to which the enclosing Application configuration applies.
virtualserver	Parameter gives the name of the virtual server to which the enclosing Application configuration applies.
client	Parameter specifies which clients can connect to the application. This is either a DNS name or dotted IP or dotted IP/maskbits.
port	Parameter gives the TCP Port number of the server to which the enclosing Application configuration applies. Note that this param is illegal if the host is specified with 'client'.

The "CacheCtrl" element are placed within an Application element and enables or disables caching for this particular application.

Atribute	Description	Required
cache	This parameter is ignored if global cache option is disabled.The dafault value is off.	This element is optional.

The "Options" element contains the debug attribute. Here is overview of it's values:

Value	Description
0x00000001	Function Entry
0x00000002	Function Exit
0x00000004	State Machine Entry
0x00000008	State Machine Exit
0x00000010	Sparse Debugging
0x00000020	Normal Debugging
0x00000040	Verbose Debugging
0x00000080	Very Verbose Debugging
0x00000100	Socket Debugging
0x00000200	Verbose Socket Debugging
0x00000400	HexDump All Outgoing Packets
0x00000800	HexDump All Incoming Packets
0x80000000	Fake AppServer Connections
0x01000000	Normal Cache Debugging
0x02000000	Verbose Cache Debugging