The HTTP Connector

Table of Contents

Introduction

The **HTTP Connector** element represents a **Connector** component that supports the HTTP/1.1 protocol. It enables Catalina to function as a stand-alone web server, in addition to its ability to execute servlets and JSP pages. A particular instance of this component listens for connections on a specific TCP port number on the server. One or more such **Connectors** can be configured as part of a single Service, each forwarding to the associated Engine to perform request processing and create the response.

If you wish to configure the **Connector** that is used for connections to web servers using the AJP protocol (such as the `mod_jk 1.2.x` connector for Apache 1.3), please refer to the AJP Connector documentation.

Each incoming request requires a thread for the duration of that request. If more simultaneous requests are received than can be handled by the currently available request processing threads, additional threads will be created up to the configured maximum (the value of the `maxThreads` attribute). If still more simultaneous requests are received, they are stacked up inside the server socket created by the **Connector**, up to the configured maximum (the value of the `acceptCount` attribute). Any further simultaneous requests will receive "connection refused" errors, until resources are available to process them.

如果同时到达的请求数超过当前可用的请求处理线程，则额外的线程会被创建（`maxThreads`）
如果请求数超过`maxThreads`值，则超过的请求都会被堆积在“服务器套接字”里（`acceptCount`）
如果请求数超过`acceptCount`值，则会接收到“连接拒绝”错误

Attributes

Common Attributes

All implementations of **Connector** support the following attributes:

Attribute	Description
`allowTrace`	A boolean value which can be used to enable or disable the TRACE HTTP method. If not specified, this attribute is set to false.
`asyncTimeout`	The default timeout for asynchronous requests in milliseconds. If not specified, this attribute is set to 10000 (10 seconds). 异步请求的超时时间
`enableLookups`	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.
`maxHeaderCount`	The maximum number of headers in a request that are allowed by the container. A request that contains more headers than the specified limit will be rejected. A value of less than 0 means no limit. If not specified, a default of 100 is used.
`maxParameterCount`	The maximum number of parameter and value pairs (GET plus POST) which will be automatically parsed by the container. Parameter and value pairs beyond this limit will be ignored. A value of less than 0 means no limit. If not specified, a default of 10000 is used. Note that `FailedRequestFilter` filter can be used to reject requests that hit the limit.
`maxPostSize`	The maximum size in bytes of the POST which will be handled by the container FORM URL parameter parsing. The limit can be disabled by setting this attribute to a value less than or equal to 0. If not specified, this attribute is set to 2097152 (2 megabytes).
`maxSavePostSize`	The maximum size in bytes of the POST which will be saved/buffered by the container during FORM or CLIENT-CERT authentication. For both types of authentication, the POST will be saved/buffered before the user is authenticated. For CLIENT-CERT authentication,

	the POST is buffered for the duration of the SSL handshake and the buffer emptied when the request is processed. For FORM authentication the POST is saved whilst the user is re-directed to the login form and is retained until the user successfully authenticates or the session associated with the authentication request expires. The limit can be disabled by setting this attribute to -1. Setting the attribute to zero will disable the saving of POST data during authentication. If not specified, this attribute is set to 4096 (4 kilobytes).
`parseBodyMethods`	A comma-separated list of HTTP methods for which request bodies will be parsed for request parameters identically to POST. This is useful in RESTful applications that want to support POST-style semantics for PUT requests. Note that any setting other than `POST` causes Tomcat to behave in a way that goes against the intent of the servlet specification. The HTTP method TRACE is specifically forbidden here in accordance with the HTTP specification. The default is `POST`
`port`	The TCP port number on which this **Connector** will create a server socket and await incoming connections. Your operating system will allow only one server application to listen to a particular port number on a particular IP address. If the special value of 0 (zero) is used, then Tomcat will select a free port at random to use for this connector. This is typically only useful in embedded and testing applications.
`protocol`	Sets the protocol to handle incoming traffic. The default value is `HTTP/1.1` which uses an auto-switching mechanism to select either a blocking Java based connector or an APR/native based connector. If the `PATH` (Windows) or `LD_LIBRARY_PATH` (on most unix systems) environment variables contain the Tomcat native library, the APR/native connector will be used. If the native library cannot be found, the blocking Java based connector will be used. Note that the APR/native connector has different settings for HTTPS than the Java connectors. To use an explicit protocol rather than rely on the auto-switching mechanism described above, the following values may be used: `org.apache.coyote.http11.Http11Protocol` - blocking Java connector `org.apache.coyote.http11.Http11NioProtocol` - non blocking Java connector `org.apache.coyote.http11.Http11AprProtocol` - the APR/native connector. Custom implementations may also be used. Take a look at our Connector Comparison chart. The configuration for both Java connectors is identical, for http and https. For more information on the APR connector and APR specific SSL settings please visit the APR documentation
`proxyName`	If this **Connector** is being used in a proxy configuration, configure this attribute to specify the server name to be returned for calls to `request.getServerName()`. See Proxy Support for more information.
`proxyPort`	If this **Connector** is being used in a proxy configuration, configure this attribute to specify the server port to be returned for calls to `request.getServerPort()`. See Proxy Support for more information.
`redirectPort`	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.
`scheme`	Set this attribute to the name of the protocol you wish to have returned by calls to `request.getScheme()`. For example, you would set this attribute to "`https`" for an SSL Connector. The default value is "`http`".
`secure`	Set this attribute to `true` if you wish to have calls to `request.isSecure()` to return `true` for requests received by this Connector. You would want this on an SSL Connector or a non SSL connector that is receiving data from a SSL accelerator, like a crypto card, a SSL appliance or even a webserver. The default value is `false`.
`URIEncoding`	This specifies the character encoding used to decode the URI bytes, after %xx decoding the URL. If not specified, ISO-8859-1 will be used. **用于解码URI字节（UTF-8）**
`useBodyEncodingForURI`	This specifies if the encoding specified in contentType should be used for URI query parameters, instead of using the URIEncoding. This setting is present for compatibility with Tomcat 4.1.x, where the encoding specified in the contentType, or explicitly set using Request.setCharacterEncoding method was also used for the parameters from the URL. The default value is `false`.
`useIPVHosts`	Set this attribute to `true` to cause Tomcat to use the IP address that the request was received on to determine the Host to send the request to. The default value is `false`.
`xpoweredBy`	Set this attribute to `true` to cause Tomcat to advertise support for the Servlet specification using the header recommended in the specification. The default value is `false`.

Standard Implementation

The standard HTTP connectors (BIO, NIO and APR/native) all support the following attributes in addition to the common Connector attributes listed above.

Attribute	Description
`acceptCount` 连接请求缓冲区	The maximum queue length for incoming connection requests when all possible request processing threads are in use. Any requests received when the queue is full will be refused. The default value is 100. 传入的连接请求的最大队列长度
`acceptorThreadCount` 用于接收连接的主线程数	The number of threads to be used to accept connections. Increase this value on a multi CPU machine, although you would never really need more than `2`. Also, with a lot of non keep alive connections, you might want to increase this value as well. Default value is `1`.
`acceptorThreadPriority`	The priority of the acceptor threads. The threads used to accept new connections. The default value is `5` (the value of the `java.lang.Thread.NORM_PRIORITY` constant). See the JavaDoc for the `java.lang.Thread` class for more details on what this priority means.
`address`	For servers with more than one IP address, this attribute specifies which address will be used for listening on the specified port. By default, this port will be used on all IP addresses associated with the server.
`bindOnInit`	Controls when the socket used by the connector is bound. By default it is bound when the connector is initiated and unbound when the connector is destroyed. If set to `false`, the socket will be bound when the connector is started and unbound when it is stopped.
`compressableMimeType`	The value is a comma separated list of MIME types for which HTTP compression may be used. The default value is `text/html,text/xml,text/plain`.
`compression`	The **Connector** may use HTTP/1.1 GZIP compression in an attempt to save server bandwidth. The acceptable values for the parameter is "off" (disable compression), "on" (allow compression, which causes text data to be compressed), "force" (forces compression in all cases), or a numerical integer value (which is equivalent to "on", but specifies the minimum amount of data before the output is compressed). If the content-length is not known and compression is set to "on" or more aggressive, the output will also be compressed. If not specified, this attribute is set to "off". *Note*: There is a tradeoff between using compression (saving your bandwidth) and using the sendfile feature (saving your CPU cycles). If the connector supports the sendfile feature, e.g. the NIO connector, using sendfile will take precedence over compression. The symptoms will be that static files greater that 48 Kb will be sent uncompressed. You can turn off sendfile by setting `useSendfile` attribute of the connector, as documented below, or change the sendfile usage threshold in the configuration of the DefaultServlet in the default `conf/web.xml` or in the `web.xml` of your web application.
`compressionMinSize`	If **compression** is set to "on" then this attribute may be used to specify the minimum amount of data before the output is compressed. If not specified, this attribute is defaults to "2048".
`connectionLinger`	The number of seconds during which the sockets used by this **Connector** will linger when they are closed. The default value is `-1` which disables socket linger.
`connectionTimeout` 在接收一条连接后，用于等待请求URI的毫秒数	The number of milliseconds this **Connector** will wait, after accepting a connection, for the request URI line to be presented. Use a value of -1 to indicate no (i.e. infinite) timeout. The default value is 60000 (i.e. 60 seconds) but note that the standard server.xml that ships with Tomcat sets this to 20000 (i.e. 20 seconds). Unless **disableUploadTimeout** is set to `false`, this timeout will also be used when reading the request body (if any).
`connectionUploadTimeout`	Specifies the timeout, in milliseconds, to use while a data upload is in progress. This only takes effect if **disableUploadTimeout** is set to `false`.
`disableUploadTimeout`	This flag allows the servlet container to use a different, usually longer connection timeout during data upload. If not specified, this attribute is set to `true` which disables this longer timeout.

executor	A reference to the name in an Executor element. If this attribute is set, and the named executor exists, the connector will use the executor, and all the other thread attributes will be ignored. Note that if a shared executor is not specified for a connector then the connector will use a private, internal executor to provide the thread pool.
executorTerminationTimeoutMillis	The time that the private internal executor will wait for request processing threads to terminate before continuing with the process of stopping the connector. If not set, the default is 0 (zero) for the BIO connector and 5000 (5 seconds) for the NIO and APR/native connectors.
keepAliveTimeout	The number of milliseconds this **Connector** will wait for another HTTP request before closing the connection. The default value is to use the value that has been set for the **connectionTimeout** attribute. Use a value of -1 to indicate no (i.e. infinite) timeout. 在关闭这条连接之前，等待其他HTTP请求的毫秒数
maxConnections <= acceptCount	The maximum number of connections that the server will accept and process at any given time. When this number has been reached, the server will accept, but not process, one further connection. This additional connection be blocked until the number of connections being processed falls below **maxConnections** at which point the server will start accepting and processing new connections again. Note that once the limit has been reached, the operating system may still accept connections based on the acceptCount setting. The default value varies by connector type. For BIO the default is the value of **maxThreads** unless an Executor is used in which case the default will be the value of maxThreads from the executor. For NIO the default is 10000. For APR/native, the default is 8192. Note that for APR/native on Windows, the configured value will be reduced to the highest multiple of 1024 that is less than or equal to maxConnections. This is done for performance reasons. If set to a value of -1, the maxConnections feature is disabled and connections are not counted.
maxExtensionSize	Limits the total length of chunk extensions in chunked HTTP requests. If the value is -1, no limit will be imposed. If not specified, the default value of 8192 will be used.
maxHttpHeaderSize	The maximum size of the request and response HTTP header, specified in bytes. If not specified, this attribute is set to 8192 (8 KB).
maxKeepAliveRequests	The maximum number of HTTP requests which can be pipelined until the connection is closed by the server. Setting this attribute to 1 will disable HTTP/1.0 keep-alive, as well as HTTP/1.1 keep-alive and pipelining. Setting this to -1 will allow an unlimited amount of pipelined or keep-alive HTTP requests. If not specified, this attribute is set to 100.
maxSwallowSize	The maximum number of request body bytes (excluding transfer encoding overhead) that will be swallowed by Tomcat for an aborted upload. An aborted upload is when Tomcat knows that the request body is going to be ignored but the client still sends it. If Tomcat does not swallow the body the client is unlikely to see the response. If not specified the default of 2097152 (2 megabytes) will be used. A value of less than zero indicates that no limit should be enforced.
maxThreads =768	The maximum number of request processing threads to be created by this **Connector**, which therefore determines the maximum number of simultaneous requests that can be handled. If not specified, this attribute is set to 200. If an executor is associated with this connector, this attribute is ignored as the connector will execute tasks using the executor rather than an internal thread pool. 请求处理线程的最大数量，它决定同一时刻请求处理的最大数量
maxTrailerSize	Limits the total length of trailing headers in the last chunk of a chunked HTTP request. If the value is -1, no limit will be imposed. If not specified, the default value of 8192 will be used.
minSpareThreads	The minimum number of threads always kept running. If not specified, the default of 10 is used. 保留的最小活动闲线程数
noCompressionUserAgents	The value is a regular expression (using java.util.regex) matching the user-agent header of HTTP clients for which compression should not be used, because these clients, although they do advertise support for the feature, have a broken implementation. The default value is an empty String (regexp matching disabled).

processorCache	The protocol handler caches Processor objects to speed up performance. This setting dictates how many of these objects get cached. -1 means unlimited, default is 200. If not using Servlet 3.0 asynchronous processing, a good default is to use the same as the maxThreads setting. If using Servlet 3.0 asynchronous processing, a good default is to use the larger of maxThreads and the maximum number of expected concurrent requests (synchronous and asynchronous).
restrictedUserAgents	The value is a regular expression (using java.util.regex) matching the user-agent header of HTTP clients for which HTTP/1.1 or HTTP/1.0 keep alive should not be used, even if the clients advertise support for these features. The default value is an empty String (regexp matching disabled).
server	Overrides the Server header for the http response. If set, the value for this attribute overrides the Tomcat default and any Server header set by a web application. If not set, any value specified by the application is used. If the application does not specify a value then Apache-Coyote/1.1 is used. Unless you are paranoid, you won't need this feature.
socketBuffer	The size (in bytes) of the buffer to be provided for socket output buffering. -1 can be specified to disable the use of a buffer. By default, a buffers of 9000 bytes will be used.
SSLEnabled	Use this attribute to enable SSL traffic on a connector. To turn on SSL handshake/encryption/decryption on a connector set this value to true. The default value is false. When turning this value true you will want to set the scheme and the secure attributes as well to pass the correct request.getScheme() and request.isSecure() values to the servlets See SSL Support for more information.
tcpNoDelay	If set to true, the TCP_NO_DELAY option will be set on the server socket, which improves performance under most circumstances. This is set to true by default.
threadPriority	The priority of the request processing threads within the JVM. The default value is 5 (the value of the java.lang.Thread.NORM_PRIORITY constant). See the JavaDoc for the java.lang.Thread class for more details on what this priority means.
upgradeAsyncWriteBufferSize	The default size of the buffer to allocate to for asynchronous writes that can not be completed in a single operation. Data that can't be written immediately will be stored in this buffer until it can be written. If more data needs to be stored than space is available in the buffer than the size of the buffer will be increased for the duration of the write. If not specified the default value of 8192 will be used.

Java TCP socket attributes

The BIO and NIO implementation support the following Java TCP socket attributes in addition to the common Connector and HTTP attributes listed above.

Attribute	Description
socket.rxBufSize	(int)The socket receive buffer (SO_RCVBUF) size in bytes. JVM default used if not set.
socket.txBufSize	(int)The socket send buffer (SO_SNDBUF) size in bytes. JVM default used if not set.
socket.tcpNoDelay	(bool)This is equivalent to standard attribute **tcpNoDelay**.
socket.soKeepAlive	(bool)Boolean value for the socket's keep alive setting (SO_KEEPALIVE). JVM default used if not set.
socket.ooBInline	(bool)Boolean value for the socket OOBINLINE setting. JVM default used if not set.
socket.soReuseAddress	(bool)Boolean value for the sockets reuse address option (SO_REUSEADDR). JVM default used if not set.
socket.soLingerOn	(bool)Boolean value for the sockets so linger option (SO_LINGER). A value for the standard attribute **connectionLinger** that is >=0 is equivalent to setting this to true. A value for the standard attribute **connectionLinger** that is <0 is equivalent to setting this to false. Both this attribute and soLingerTime must be set else the JVM defaults will be used for both.

socket.soLingerTime	(int)Value in seconds for the sockets so linger option (SO_LINGER). This is equivalent to standard attribute **connectionLinger**. Both this attribute and soLingerOn must be set else the JVM defaults will be used for both.
socket.soTimeout	This is equivalent to standard attribute **connectionTimeout**.
socket.performanceConnectionTime	(int)The first value for the performance settings. See Socket Performance Options All three performance attributes must be set else the JVM defaults will be used for all three.
socket.performanceLatency	(int)The second value for the performance settings. See Socket Performance Options All three performance attributes must be set else the JVM defaults will be used for all three.
socket.performanceBandwidth	(int)The third value for the performance settings. See Socket Performance Options All three performance attributes must be set else the JVM defaults will be used for all three.
socket.unlockTimeout	(int) The timeout for a socket unlock. When a connector is stopped, it will try to release the acceptor thread by opening a connector to itself. The default value is 250 and the value is in milliseconds

BIO specific configuration

The following attributes are specific to the BIO connector.

Attribute	Description
disableKeepAlivePercentage	The percentage of processing threads that have to be in use before HTTP keep-alives are disabled to improve scalability. Values less than 0 will be changed to 0 and values greater than 100 will be changed to 100. If not specified, the default value is 75.

NIO specific configuration

The following attributes are specific to the NIO connector.

Attribute	Description
pollerThreadCount	(int)The number of threads to be used to run for the polling events. Default value is 1 per processor up to and including version 7.0.27. Default value as of version 7.0.28 is 1 per processor but not more than 2. When accepting a socket, the operating system holds a global lock. So the benefit of going above 2 threads diminishes rapidly. Having more than one thread is for system that need to accept connections very rapidly. However usually just increasing acceptCount will solve that problem. Increasing this value may also be beneficial when a large amount of send file operations are going on.
pollerThreadPriority	(int)The priority of the poller threads. The default value is 5 (the value of the java.lang.Thread.NORM_PRIORITY constant). See the JavaDoc for the java.lang.Thread class for more details on what this priority means.
selectorTimeout	(int)The time in milliseconds to timeout on a select() for the poller. This value is important, since connection clean up is done on the same thread, so do not set this value to an extremely high one. The default value is 1000 milliseconds.
useComet	(bool)Whether to allow comet servlets or not. Default value is true.
useSendfile	(bool)Use this attribute to enable or disable sendfile capability. The default value is true.
socket.directBuffer	(bool)Boolean value, whether to use direct ByteBuffers or java mapped ByteBuffers. Default is false. When you are using direct buffers, make sure you allocate the appropriate amount of memory for the direct memory space. On Sun's JDK that would be something like -XX:MaxDirectMemorySize=256m.
socket.appReadBufSize	(int)Each connection that is opened up in Tomcat get associated with a read ByteBuffer. This attribute controls the size of this buffer. By default this read buffer is sized at 8192 bytes. For lower concurrency, you can increase this to buffer more data. For an extreme amount of keep alive connections, decrease this number or increase your heap size.
socket.appWriteBufSize	(int)Each connection that is opened up in Tomcat get associated with a write ByteBuffer. This attribute controls the size of this buffer. By default this write

	buffer is sized at `8192` bytes. For low concurrency you can increase this to buffer more response data. For an extreme amount of keep alive connections, decrease this number or increase your heap size. The default value here is pretty low, you should up it if you are not dealing with tens of thousands concurrent connections.
`socket.bufferPool`	(int)The NIO connector uses a class called NioChannel that holds elements linked to a socket. To reduce garbage collection, the NIO connector caches these channel objects. This value specifies the size of this cache. The default value is `500`, and represents that the cache will hold 500 NioChannel objects. Other values are `-1` for unlimited cache and `0` for no cache.
`socket.bufferPoolSize`	(int)The NioChannel pool can also be size based, not used object based. The size is calculated as follows: NioChannel `buffer size = read buffer size + write buffer size` SecureNioChannel `buffer size = application read buffer size + application write buffer size + network read buffer size + network write buffer size` The value is in bytes, the default value is `1024*1024*100` (100MB).
`socket.processorCache`	(int)Tomcat will cache SocketProcessor objects to reduce garbage collection. The integer value specifies how many objects to keep in the cache at most. The default is `500`. Other values are `-1` for unlimited cache and `0` for no cache.
`socket.keyCache`	(int)Tomcat will cache KeyAttachment objects to reduce garbage collection. The integer value specifies how many objects to keep in the cache at most. The default is `500`. Other values are `-1` for unlimited cache and `0` for no cache.
`socket.eventCache`	(int)Tomcat will cache PollerEvent objects to reduce garbage collection. The integer value specifies how many objects to keep in the cache at most. The default is `500`. Other values are `-1` for unlimited cache and `0` for no cache.
`selectorPool.maxSelectors`	(int)The max selectors to be used in the pool, to reduce selector contention. Use this option when the command line `org.apache.tomcat.util.net.NioSelectorShared` value is set to false. Default value is `200`.
`selectorPool.maxSpareSelectors`	(int)The max spare selectors to be used in the pool, to reduce selector contention. When a selector is returned to the pool, the system can decide to keep it or let it be GC'd. Use this option when the command line `org.apache.tomcat.util.net.NioSelectorShared` value is set to false. Default value is `-1` (unlimited).
`command-line-options`	The following command line options are available for the NIO connector: `-Dorg.apache.tomcat.util.net.NioSelectorShared=true|false` - default is `true`. Set this value to `false` if you wish to use a selector for each thread. When you set it to `false`, you can control the size of the pool of selectors by using the **selectorPool.maxSelectors** attribute.
`oomParachute`	(int)The NIO connector implements an OutOfMemoryError strategy called parachute. It holds a chunk of data as a byte array. In case of an OOM, this chunk of data is released and the error is reported. This will give the VM enough room to clean up. The `oomParachute` represents the size in bytes of the parachute(the byte array). The default value is `1024*1024`(1MB). Please note, this only works for OOM errors regarding the Java Heap space, and there is absolutely no guarantee that you will be able to recover at all. If you have an OOM outside of the Java Heap, then this parachute trick will not help.

APR/native specific configuration

The following attributes are specific to the APR/native connector.

Attribute	Description
`deferAccept`	Sets the `TCP_DEFER_ACCEPT` flag on the listening socket for this connector. The default value is `true` where `TCP_DEFER_ACCEPT` is supported by the operating system, otherwise it is `false`.
`pollerSize`	Amount of sockets that the poller responsible for polling kept alive connections can hold at a given time. Extra connections will be closed right away. The default value is 8192, corresponding to 8192 keep-alive connections. This is a synonym for maxConnections.
`pollTime`	Duration of a poll call in microseconds. Lowering this value will slightly decrease latency of connections being kept alive in some cases, but will use more CPU as more poll calls are being made. The default value is 2000 (2ms).

sendfileSize	Amount of sockets that the poller responsible for sending static files asynchronously can hold at a given time. Extra connections will be closed right away without any data being sent (resulting in a zero length file on the client side). Note that in most cases, sendfile is a call that will return right away (being taken care of "synchronously" by the kernel), and the sendfile poller will not be used, so the amount of static files which can be sent concurrently is much larger than the specified amount. The default value is 1024.
threadPriority	(int)The priority of the acceptor and poller threads. The default value is `5` (the value of the `java.lang.Thread.NORM_PRIORITY` constant). See the JavaDoc for the `java.lang.Thread` class for more details on what this priority means.
useComet	(bool)Whether to allow comet servlets or not. Default value is `true`.
useSendfile	(bool)Use this attribute to enable or disable sendfile capability. The default value is `true`.

Nested Components

None at this time.

Special Features

HTTP/1.1 and HTTP/1.0 Support

This **Connector** supports all of the required features of the HTTP/1.1 protocol, as described in RFC 2616, including persistent connections, pipelining, expectations and chunked encoding. If the client (typically a browser) supports only HTTP/1.0, the **Connector** will gracefully fall back to supporting this protocol as well. No special configuration is required to enable this support. The **Connector** also supports HTTP/1.0 keep-alive.

RFC 2616 requires that HTTP servers always begin their responses with the highest HTTP version that they claim to support. Therefore, this **Connector** will always return `HTTP/1.1` at the beginning of its responses.

Proxy Support

The `proxyName` and `proxyPort` attributes can be used when Tomcat is run behind a proxy server. These attributes modify the values returned to web applications that call the `request.getServerName()` and `request.getServerPort()` methods, which are often used to construct absolute URLs for redirects. Without configuring these attributes, the values returned would reflect the server name and port on which the connection from the proxy server was received, rather than the server name and port to whom the client directed the original request.

For more information, see the Proxy Support HOW-TO.

SSL Support

You can enable SSL support for a particular instance of this **Connector** by setting the `SSLEnabled` attribute to `true`.

You will also need to set the `scheme` and `secure` attributes to the values `https` and `true` respectively, to pass correct information to the servlets.

The BIO and NIO connectors use the JSSE SSL whereas the APR/native connector uses OpenSSL. Therefore, in addition to using different attributes to configure SSL, the APR/native connector also requires keys and certificates to be provided in a different format.

For more information, see the SSL Configuration HOW-TO.

SSL Support - BIO and NIO

The BIO and NIO connectors use the following attributes to configure SSL:

Attribute	Description
algorithm	The certificate encoding algorithm to be used. This defaults to `KeyManagerFactory.getDefaultAlgorithm()` which returns `SunX509` for Sun JVMs. IBM JVMs return `IbmX509`. For other vendors, consult the JVM documentation for the default value.
allowUnsafeLegacyRenegotiation	Is unsafe legacy TLS renegotiation allowed which is likely to expose users to CVE-2009-3555, a man-in-the-middle vulnerability in the TLS protocol that allows an attacker to inject arbitrary data into the user's request. If not specified, a default of `false` is used. This attribute only has an effect if the JVM does not support RFC 5746 as indicated by the presence of the pseudo-ciphersuite TLS_EMPTY_RENEGOTIATION_INFO_SCSV. This is available

	JRE/JDK 6 update 22 onwards. Where RFC 5746 is supported the renegotiation - including support for unsafe legacy renegotiation - is controlled by the JVM configuration.
ciphers	The comma separated list of encryption ciphers to support for HTTPS connections. If specified, only the ciphers that are listed and supported by the SSL implementation will be used. By default, the default ciphers for the JVM will be used. Note that this usually means that the weak export grade ciphers will be included in the list of available ciphers. The ciphers are specified using the JSSE cipher naming convention. The special value of `ALL` will enable all supported ciphers. This will include many that are not secure. `ALL` is intended for testing purposes only.
clientAuth	Set to `true` if you want the SSL stack to require a valid certificate chain from the client before accepting a connection. Set to `want` if you want the SSL stack to request a client Certificate, but not fail if one isn't presented. A `false` value (which is the default) will not require a certificate chain unless the client requests a resource protected by a security constraint that uses `CLIENT-CERT` authentication.
clientCertProvider	When client certificate information is presented in a form other than instances of `java.security.cert.X509Certificate` it needs to be converted before it can be used and this property controls which JSSE provider is used to perform the conversion. For example it is used with the AJP connectors, the HTTP APR connector and with the org.apache.catalina.valves.SSLValve. If not specified, the default provider will be used.
crlFile	The certificate revocation list to be used to verify client certificates. If not defined, client certificates will not be checked against a certificate revocation list.
keyAlias	The alias used to for the server certificate in the keystore. If not specified the first key read in the keystore will be used.
keyPass	The password used to access the server certificate from the specified keystore file. The default value is "`changeit`".
keystoreFile	The pathname of the keystore file where you have stored the server certificate to be loaded. By default, the pathname is the file "`.keystore`" in the operating system home directory of the user that is running Tomcat. If your `keystoreType` doesn't need a file use `""` (empty string) for this parameter.
keystorePass	The password used to access the specified keystore file. The default value is the value of the `keyPass` attribute.
keystoreProvider	The name of the keystore provider to be used for the server certificate. If not specified, the list of registered providers is traversed in preference order and the first provider that supports the `keystoreType` is used.
keystoreType	The type of keystore file to be used for the server certificate. If not specified, the default value is "`JKS`".
sessionCacheSize	The number of SSL sessions to maintain in the session cache. Use 0 to specify an unlimited cache size. If not specified, a default of 0 is used.
sessionTimeout	The time, in seconds, after the creation of an SSL session that it will timeout. Use 0 to specify an unlimited timeout. If not specified, a default of 86400 (24 hours) is used.
sslEnabledProtocols	The comma separated list of SSL protocols to support for HTTPS connections. If specified, only the protocols that are listed and supported by the SSL implementation will be enabled. If not specified, the JVM default is used. The permitted values may be obtained from the JVM documentation for the allowed values for `SSLSocket.setEnabledProtocols()` e.g. Oracle Java 6 and Oracle Java 7. Note: There is overlap between this attribute and `sslProtocol`.

sslImplementationName	The class name of the SSL implementation to use. If not specified, the default of `org.apache.tomcat.util.net.jsse.JSSEImplementation` will be used which wraps JVM's default JSSE provider. Note that the JVM can be configured to use a different JSSE provider as the default.
sslProtocol	The the SSL protocol(s) to use (a single value may enable multiple protocols - see the JVM documentation for details). If not specified, the default is `TLS`. The permitted values may be obtained from the JVM documentation for the allowed values for algorithm when creating an `SSLContext` instance e.g. Oracle Java 6 and Oracle Java 7. Note: There is overlap between this attribute and `sslEnabledProtocols`.
trustManagerClassName	The name of a custom trust manager class to use to validate client certificates. The class must have a zero argument constructor and must also implement `javax.net.ssl.X509TrustManager`. If this attribute is set, the trust store attributes may be ignored.
trustMaxCertLength	The maximum number of intermediate certificates that will be allowed when validating client certificates. If not specified, the default value of 5 will be used.
truststoreAlgorithm	The algorithm to use for truststore. If not specified, the default value returned by `javax.net.ssl.TrustManagerFactory.getDefaultAlgorithm()` is used.
truststoreFile	The trust store file to use to validate client certificates. The default is the value of the `javax.net.ssl.trustStore` system property. If neither this attribute nor the default system property is set, no trust store will be configured.
truststorePass	The password to access the trust store. The default is the value of the `javax.net.ssl.trustStorePassword` system property. If that property is null, no trust store password will be configured. If an invalid trust store password is specified, a warning will be logged and an attempt will be made to access the trust store without a password which will skip validation of the trust store contents.
truststoreProvider	The name of the truststore provider to be used for the server certificate. The default is the value of the `javax.net.ssl.trustStoreProvider` system property. If that property is null, the value of `keystoreProvider` is used as the default. If neither this attribute, the default system property nor `keystoreProvider`is set, the list of registered providers is traversed in preference order and the first provider that supports the `truststoreType` is used.
truststoreType	The type of key store used for the trust store. The default is the value of the `javax.net.ssl.trustStoreType` system property. If that property is null, the value of `keystoreType` is used as the default.

SSL Support - APR/Native

When APR/native is enabled, the HTTPS connector will use a socket poller for keep-alive, increasing scalability of the server. It also uses OpenSSL, which may be more optimized than JSSE depending on the processor being used, and can be complemented with many commercial accelerator components. Unlike the HTTP connector, the HTTPS connector cannot use sendfile to optimize static file processing.

The HTTPS APR/native connector has the same attributes than the HTTP APR/native connector, but adds OpenSSL specific ones. For the full details on using OpenSSL, please refer to OpenSSL documentations and the many books available for it (see the Official OpenSSL website). The SSL specific attributes for the APR/native connector are:

Attribute	Description
SSLCACertificateFile	See the mod_ssl documentation.
SSLCACertificatePath	See the mod_ssl documentation.
SSLCARevocationFile	See the mod_ssl documentation.
SSLCARevocationPath	See the mod_ssl documentation.

SSLCertificateChainFile	See the mod_ssl documentation.
SSLCACertificateFile	Name of the file that contains the concatenated certificates for the trusted certificate authorities. The format is PEM-encoded.
SSLCACertificatePath	Name of the directory that contains the certificates for the trusted certificate authorities. The format is PEM-encoded.
SSLCARevocationFile	Name of the file that contains the concatenated certificate revocation lists for the certificate authorities. The format is PEM-encoded.
SSLCARevocationPath	Name of the directory that contains the certificate revocation lists for the certificate authorities. The format is PEM-encoded.
SSLCertificateChainFile	Name of the file that contains concatenated certifcates for the certificate authorities which form the certifcate chain for the server certificate. The format is PEM-encoded.
SSLCertificateFile	Name of the file that contains the server certificate. The format is PEM-encoded.
SSLCertificateKeyFile	Name of the file that contains the server private key. The format is PEM-encoded. The default value is the value of "SSLCertificateFile" and in this case both certificate and private key have to be in this file (NOT RECOMMENDED).
SSLCipherSuite	Ciphers which may be used for communicating with clients. The default is "ALL", with other acceptable values being a list of ciphers, with ":" used as the delimiter (see OpenSSL documentation for the list of ciphers supported).
SSLDisableCompression	Disables compression if set to `true` and OpenSSL supports disabling compression. Default is `false` which inherits the default compression setting in OpenSSL.
SSLHonorCipherOrder	Set to `true` to enforce the server's cipher order (from the `SSLCipherSuite` setting) instead of allowing the client to choose the cipher (which is the default).
SSLPassword	Pass phrase for the encrypted private key. If "SSLPassword" is not provided, the callback function should prompt for the pass phrase.
SSLProtocol	Protocol which may be used for communicating with clients. The default value is `all`, which is equivalent to `SSLv3+TLSv1` with other acceptable values being `SSLv2`, `SSLv3`, `TLSv1` and any combination of the three protocols concatenated with a plus sign. Note that the protocol `SSLv2` is inherently unsafe.
SSLVerifyClient	Ask client for certificate. The default is "none", meaning the client will not have the opportunity to submit a certificate. Other acceptable values include "optional", "require" and "optionalNoCA".
SSLVerifyDepth	Maximum verification depth for client certificates. The default is "10".

Connector Comparison

Below is a small chart that shows how the connectors differentiate.

	Java Blocking Connector	Java Non Blocking Connector	APR/native Connector
	BIO	NIO	APR
Classname	Http11Protocol	Http11NioProtocol	Http11AprProtocol
Tomcat Version	3.x onwards	6.x onwards	5.5.x onwards
Support Polling	NO	YES	YES
Polling Size	N/A	maxConnections	maxConnections
Read HTTP Request	Blocking	Non Blocking	Blocking
Read HTTP Body	Blocking	Sim Blocking	Blocking
Write HTTP Response	Blocking	Sim Blocking	Blocking
Wait for next Request	Blocking	Non Blocking	Non Blocking
SSL Support	Java SSL	Java SSL	OpenSSL
SSL Handshake	Blocking	Non blocking	Blocking
Max Connections	maxConnections	maxConnections	maxConnections