EXIT GAMES®

**What:** **Photon Configuration (PhotonServer.config)**
**Version:** **3.2.x**
Author: developer@exitgames.com
Last update: 02.04.2013

| Section | Setting | PhotonServer .config | Default | Description |
|---|---|---|---|---|
| **<Instance ...>** | **Multiple instances are supported. Each instance has its own node in the config file. PhotonControl will currently only start "Instance1" but the .cmd files could be modified to start other instances.** | | | |
| | ConfigName | | | Optional "ConfigName" value from config file if present it will be logged. |
| | ReportOnJob | | FALSE | It causes photon to determine if it is running inside a win32 job object (something that is used by process starting code to place limits on what the process can do) and if so reports on the limits that are currently imposed. |
| | LogFileLocation | | | Sets the path for logfiles. Default is the location where the server is started. Can be absolute or relative to the PhotonSocketServer.exe |
| | MaxMessageSize | | 512000 | Determines the maximum number of bytes that a UDP, TCP or WebSockets message might have that is received or sent by the server. If the message exceeds the MaxMessageSize limit, the client will be disconnected from the server. MaxMessageSize can be overwritten in the appropriate listener. |
| | **[TCP]** | | | |
| | TCPBufferSize | | 4096 | The size of buffer used during TCP data transfer. Note that this does not in any way restrict the message size. It's ideal to have the buffer size be slightly more than the average message size. |
| | TCPBufferAllocatorPoolSize | | 500 | The number of buffers used for TCP data transfer that are kept in the pool for later reuse. This is a classic trade of speed vs resources. The pooled buffers are faster to allocate but take up memory all the time. |
| | StreamSocketAllocatorPoolSize | | 250 | The number of TCP sockets that are kept in the pool for later reuse. This is a classic trade of speed vs resources. The pooled sockets are faster to allocate but take up memory all the time. |
| | **[UDP/ENET]** | | | |
| | DatagramSocketAllocatorPoolSize | | 1000 | The number of datagram sockets that are kept in the pool for later reuse. This is a classic trade of speed vs resources. The pooled sockets are faster to allocate but take up memory all the time. |
| | PingIntervalMilliseconds | | 1000 | The Ping interval is how often a ping is sent from server to client when no data is flowing. Sending pings adds load to the server and so this can be tuned; a larger value leads to less load on the server. The ENet protocol usually uses a fixed value for this parameter. Defaults to the ENet standard value of 500ms. |
| | DataSendingDelayMilliseconds | | 5 | The data sending delay is how long the server will wait to accumulate more data to send to the client in the same datagram. A larger value can lead to fewer datagrams being sent as more data can be put into a single datagram, however this also increases latency as the server waits longer before sending responses. If using the 'standard' ENet code then data is accumulated until a call to enet_host_flush() or enet_host_service(). |
| | AckSendingDelayMilliseconds | | 5 | The data sending delay is how long the server will wait before sending an ACK to the client. This allows it to potentially piggyback the ACK onto a data packet if a response packet is generated within the delay period. A larger value can lead to fewer datagrams being sent as an ACK can be put into the same datagram as a reply. This doesn't increase latency. If using the 'standard' ENet code then data is accumulated until a call to enet_host_flush() or enet_host_service(). |
| | MinimumRetransmitTimeout | | 200 | Enet retransmission times are calculated based on the accumulated RTT of the peer. This setting sets a minimum retransmission time. Peers with very low latency could want to retransmit before the value that this setting allows and will be delayed. This value allows you to reduce the speed at which data is retransmitted and reduce the load on the server. If you are sending reliable data and have low latency peers and require minimum latency then setting this value to anything but zero will likely increase latency and reduce the number of peers that you can support with given CPU resources. |
| | ENETBufferAllocatorPoolSize | | 5000 | The number of buffers that are kept in the pool for later reuse. This is a classic trade of speed vs resources. The pooled buffers are faster to allocate but take up memory all the time. |
| | MinimumTimeout | 5000 | 5000 | The minimum amount of time before an Enet peer's lack of response causes a timeout disconnect. Note that the actual time is determined dynamically per peer based on the RTT history. |
| | MaximumTimeout | 30000 | 30000 | The maximum amount of time before an Enet peer's lack of response causes a timeout disconnect. Note that the actual time is determined dynamically per peer based on the RTT history. |
| | SetValidateCRCifPresent | | FALSE | If enabled: if the incoming datagram contains an optional CRC value, the checksum is validated. If the peer had been created with the CRC value present (indicated by a "CRC" flag in the UDP header that is set by the client library), all outbound datagrams that are sent to the client will contain the CRC value, too. |
| | **[Udp/Enet Throtteling]** | | | |
| | PerPeerMaxReliableDataInTransit | 51200 | 16384 | The maximum amount of reliable data that a peer can send and which the peer has not as yet received an ACK for. In bytes. Once this amount of data has been sent all future reliable data will be queued. |
| | PerPeerTransmitRateLimitKBSec | 256 | 256 | The maximum amount of data (reliable AND unreliable) that can be sent in a second (in KB). This can be used to limit the amount of data that a peer can send. When the limit is reached further reliable data is queued and unreliable data is dropped. The default, 12, is a purely arbitrary value that has no meaning and has NOT been carefully calculated to be in any way special. |
| | PerPeerTransmitRatePeriodMilliseconds | | 200 | How often we check the transmit rate limit. By default we check every 250ms (i.e. 4 times per second) and we scale the PerPeerTransmitRateLimitKBSec by 4 for each check. A smaller value makes the data flow more consistent, a larger value makes the flow more jerky but uses less server resources. |
| | MaxQueuedDataPerPeer | 512000 | 163840 | The maximum amount of reliable data that a peer can queue. Reliable data is queued when either the PerPeerMaxReliableDataInTransit or the PerPeerTransmitRateLimitKBSec limits are exceeded. Once this limit is exceeded all future reliable sends will return an error code. |
| | **[DEBUGGING]** | | | |
| | ProduceDumps | | TRUE | Switch to enable or disable creation of "dump files" in case of a crash. Dump files are essential to find issues in the Photon Core. |
| | DumpType | | Full | Defines the type of crash dumps to write. The types are "Full", "Maxi" and "Mini" and they include less information from first to last but also require less space on the harddisk. |
| | MaxDumpsToProduce | | 10 | Configures how many dump files are written maximum. If the files are moved or deleted, new ones can be written. |

| | | | | |
|---|---|---|---|---|
| **<IOPool …>** | **This thread pool handles all I/O operations such as datagram sending and reception, tcp read completions, etc. This pool can also handle Enet protocol operations if ENetThreadPool\OnlyDispatchTimers is set to "true". Once inbound data has been deblocked into commands it is sent to the "Business Logic" thread pool for passing into the CLR and managed code.** | | | |
| | NumThreads | | 2 | This value determines the number of threads used for socket I/O. A value of 4-8 tends to be good but you should profile your server to see what works best. Use the I/O thread performance counters to see when all I/O threads are busy when the server is under load. If set to 0 then the I/O Pool uses 2 x the number of CPU cores as the number of threads; whilst this works well for up to a dual core it isn't especially useful if the resulting number of threads is more than 8. |
| | Priority | | Normal | Thread priority of the IO threads. Valid values are "Lowest", "Below Normal", "Normal", "Above Normal" and "Highest". |
| **<ThreadPool …>** | **This is the "Business Logic" thread pool. It's used for all CLR operations.** | | | |
| | InitialThreads | | 4 | This setting determines the number of threads that the thread pool starts with. |
| | MinThreads | | 4 | This setting determines the minimum number of threads that the thread pool will contain. |
| | MaxThreads | | 4 | This setting determines the maximum number of threads that the thread pool will contain. Note that if Initial, min and max are all the same then the pool is a static pool rather than a dynamic pool. Static pools are more efficient than dynamic pools as the work item dispatch process is optimised. Dynamic pools can grow and shrink as demand changes. If your work items are generally short lived and the pool rarely expands in general use then you can get a performance boost by setting the pool up as a static pool. |
| | MaxDormantThreads | | 4 | This setting determines the maximum number of threads that can be 'dormant', i.e. not currently performing work. Once the number of dormant threads exceeds this amount some threads will be shut down. |
| | PoolMaintPeriod | | 5000 | This setting determines how often the pool is maintained. The value is in milliseconds. A maintenance thread will wake up every X ms to check on and possibly shut down dormant threads. |
| | DispatchTimeout | | 100 | This setting determines when a new thread may be started. If a work item is dispatched and this number of milliseconds passes before a worker thread picks up the item and begins processing it then a new thread may be started. |
| | Priority | | Normal | Thread priority of the business logic threads. Valid values are "Lowest", "Below Normal", "Normal", "Above Normal" and "Highest". |
| **<ENetThreadPool …>** | **The Enet Thread Pool is one of three thread pools in the server. This is used for all Enet protocol operations (i.e. deblocking inbound datagrams, acking on and sending ACKs, dealing with** | | | |
| | InitialThreads | | 2 | See ThreadPool. |
| | MinThreads | | 2 | See ThreadPool. |
| | MaxThreads | | 2 | See ThreadPool. |
| | MaxDormantThreads | | 5 | See ThreadPool. |
| | PoolMaintPeriod | | 5000 | See ThreadPool. |
| | DispatchTimeout | | 100 | See ThreadPool. |
| | Priority | | Normal | Thread priority of the enet threads. Valid values are "Lowest", "Below Normal", "Normal", "Above Normal" and "Highest". |
| | OnlyDispatchTimers | | FALSE | If set to true the only Enet timer operations use this pool. If set to false then all Enet operations (i.e. protocol level operations such as deblocking inbound data and processing ACKs etc) are dispatched to this pool. |
| **<Runtime …>** | **Defines the Photon Runtime behavior.** | | | |
| | Assembly | PhotonHostRuntime, Culture=neutral | | The details of the Photon Runtime assembly to use. Normal .Net assembly loading rules are used to locate the assembly. The assembly MUST be signed by the Exit Games Photon Runtime key and the PublicKeyToken part of the assembly name must NOT be specified as we add that on ourselves before we load the assembly. |
| | CLRVersion | | | By default, the latest DotNet runtime is used to host Photon Applications. Per instance, this setting allows to load a select version. Values are "v4.0" or "v2.0" or complete versions with build numbers. Example: CLRVersion = "2.0" |
| | EnableMDA | | FALSE | Enables the managed debugging assistants. Configure these with a PhotonSocketServer.exe.mda.config file in the same directory as the exe. |
| | Type | PhotonHostRuntime .PhotonDomain Manager | | The name of the type that is used as the Photon Runtime's domain manager. This type MUST be located in the assembly detailed by the Runtime value. MUST derive from PhotonHostRuntimeInterfaces.IPhotonDomainManager. |
| | UnhandledExceptionPolicy | Ignore | Ignore | Ignore, ReloadAppDomain, TerminateProcess |
| **<Applications …>** | | | | |
| | Default | | | A default application to use if no application is specified. |
| | PassUnknownAppsToDefaultApp | | TRUE | If true when an unknown application name is supplied the default application is used. If false then this is a fatal error which will terminate a connection. |
| **___<Application …>** | | | | |
| | Name | | | Arbitrary name for the application, used for dispatch. |
| | SharedDirectory | | "Shared" | Used to contain 'shared' assemblies. WILL BE located under the Base Directory. |
| | BaseDirectory | | | This folder should contain a "bin" folder with all the application assemblies and config files, etc. |
| | Assembly | | | Details of the assembly to use. Normal .Net assembly loading rules are used to locate this assembly and it can contain a complete assembly specification string (including PublicKeyToken if requried). |
| | Type | | | The name of the type that will be loaded as the application. MUST derive from PhotonHostRuntimeInterfaces.IPhotonApplication. |
| | EnableShadowCopy | | FALSE | If enabled then the CLR will create copies of the assembly files in a private directory when they are loaded. This allows you to overwrite the originals, for update, say. The copies are used until the application domain that's using them is shut down. |
| | EnableAutoRestart | | FALSE | If enabled the app will restart automatically if files changes occur. A new copy of the application is started; existing connections to the old copy are allowed to continue until all connections disconnect at which point the app domain is unloaded. The IPhotonApplicationControl.OnStopRequested() method is called when the unmanaged code wishes to 'kindly' unload an app domain, the app could inform the clients at this point. |
| | ForceAutoRestart | TRUE | FALSE | ForceAutoRestart, if set to "true", implies "EnableAutoRestart" but aborts all existing connections rather than waiting for them to disconnect. |
| | RestartDelayMilliseconds | | 1000 | Restarts due to file changes can be delayed with this setting. |
| | WatchFiles | | "" | By default no "WatchFiles" entry means watch everything. Example: WatchFiles="dll;config;Lite.XML" |
| | ExcludeFiles | | "" | By default no "ExcludeFiles" entry means exclude nothing. Example: ExcludeFiles="log4net.config" |

| | | | | |
|---|---|---|---|---|
| **<TCPListeners …>** | | | | |
| **____<TCPListener …>** | | | | |
| | IPAddress | 0.0.0.0 | | The IP Address to listen on, e.g. 192.168.0.1 or 0.0.0.0 to listen on ALL interfaces. |
| | Port | 4530 | 5051 | The port to listen on. |
| | ListenBacklog | | 150 | The size of the listen backlog queue. This affects the number of connections that can be established simultaneously; so, for example, if 151 clients attempt to connect at exactly the same time then the last one could be rejected. Note that each connection establishment attempt takes very little time and once the connection IS established the connection is not affected by the limit. So you can set the value to 10 and still have 10,000 concurrent active connections as long as no more than 10 attempt to connect at exactly the same time. |
| | RecvBufferSize | | 0 | The size of the TCP recv buffer used by the TCP stack. This is also used to determine the TCP window size (which is used by the TCP stack for flow control). Defaults to zero which means "don't change" which causes the operating system's default value to be used. |
| | SendBufferSize | | 0 | The size of the TCP send buffer used by the TCP stack. Defaults to zero which means "don't change" which causes the operating system's default value to be used. |
| | MaxPendingWrites | | 50 | Flow control setting: configures the max amount of pending writes in buffers (see TCPBufferSize). |
| | MaxQueuedBuffers | | 200 | Flow control setting:: configures the amount of buffers used to queue writes (see TCPBufferSize). More queueued buffers than pending writes means that we use less non-paged pool (a scarce resource) as only pending writes use non-paged pool. |
| | DisableNagle | TRUE | TRUE | Determines if Nagle's algorithm is in use on the connection. If set to true then Nagle is disabled and outbound TCP data will be sent as soon as it reaches the TCP stack. If set to false then Nagle is in operation and the TCP stack will attempt to colasce outbound data into fewer datagrams. Setting this setting to true might improve the latency of your TCP connections a little, at the expense of there being more datagrams sent. |
| | InactivityTimeout | 5000 | 5000 | A time, in ms, which is allowed between receiving data on a connection. If this time is exceeded then the connection is deemed to be inactive and is aborted. To ensure that this timer doesn't fire inappropriately you should make sure that there is an inbound TCP message on each connection more frequently than the value set here. Set to zero to disable the inactivity timer. |
| | DisconnectTimeout | | 120000 | A time, in ms, which is allowed for the client to close the connection once the connection has been disconnected by the application. |
| | OverrideApplication | | | Can be used to override the application used for this connector. The application selection sent by clients will be ignored. The value can be the name of any setup application. |
| | DefaultApplication | | | Defines a default application for a Listener. This DefaultApplication is used, when a client tries to connect to a application that is not loaded. For this Listener, the DefaultApplication overrides the "Default" application set in the Applications node. The value can be the name of any setup application. |
| | PolicyApplication | | | Defines the application to be used in case policy requests are sent to this listener. (Bug of some flash clients) |
| | MaxInboundMessageSize | | | Determines the maximum number of bytes that a message might have which is **received** by the server through this Listener. If the message exceeds the MaxInboundMessageSize limit, the client will be disconnected from the server. Default is the configured value in MaxMessageSize. |
| | MaxOutboundMessageSize | | | Determines the maximum number of bytes that a message might have which is **sent** by the server through this Listener. If the message exceeds the MaxOutboundMessageSize limit, the client will be disconnected from the server. Default is the configured value in MaxMessageSize. |
| **<UDPListeners …>** | To define a list of UDP/Enet listeners. | | | |
| **____<UDPListener …>** | | | | |
| | IPAddress | 0.0.0.0 | | The IP Address to listen on, e.g. 192.168.0.1 or 0.0.0.0 to listen on ALL interfaces. |
| | Port | 5055 | 5055 | The port to listen on. |
| | ListenBacklog | | 500 | See TCPListener. |
| | RecvBufferSize | | | See TCPListener. |
| | SendBufferSize | | | See TCPListener. |
| | OverrideApplication | | | See TCPListener. |
| | DefaultApplication | | | See TCPListener. |
| | MaxInboundMessageSize | | | See TCPListener. |
| | MaxOutboundMessageSize | | | See TCPListener. |
| **<WebSocketListeners …>** | | | | |
| **____<WebSocketListener …>** | | | | |
| | IPAddress | 0.0.0.0 | | The IP Address to listen on, e.g. 192.168.0.1 or 0.0.0.0 to listen on ALL interfaces. |
| | Port | | 5051 | The port to listen on. |
| | ListenBacklog | | | See TCPListener. |
| | RecvBufferSize | | | See TCPListener. |
| | SendBufferSize | | | See TCPListener. |
| | DisableNagle | TRUE | TRUE | See TCPListener. |
| | InactivityTimeout | 10000 | 10000 | See TCPListener. |
| | OverrideApplication | | | See TCPListener. |
| | DefaultApplication | | | See TCPListener. |
| | MaxInboundMessageSize | | | See TCPListener. |
| | MaxOutboundMessageSize | | | See TCPListener. |
| | MaxPendingWrites | | | See TCPListener. |
| | MaxQueuedBuffers | | | See TCPListener. |
| | DisconnectTimeout | | | See TCPListener. |
| | **[SSL]** | | | |
| | Secure | | FALSE | True defines a listener is secured by ssl. |
| | StoreName | | MY | Name of store where certificate can be found. |
| | CertificateName | | Photon | Name of certificate to be used. |
| | UseMachineStore | | FALSE | Defines if Machinestore should be used. |

| | | | | |
|---|---|---|---|---|
| **<TCPFlashListeners …>** | | | | |
| **____<TCPFlashListener …>** | Deprecated use TCPPolicyListener instead. | | | |
| | | | | |
| **<TCPPolicyListeners …>** | | | | |
| **____<TCPPolicyListener …>** | | | | |
| | Address | | | The IP Address to listen on, e.g. 192.168.0.1 or 0.0.0.0 to listen on ALL interfaces. |
| | Port | | 943 | The port to listen on. |
| | Application | | | Defines the application for a Listener. |
| | ListenBacklog | | | See TCPListener. |
| | InactivityTimeout | | 1000 | See TCPListener. |
| **<ConfigServer …>** | If a <ConfigServer> node is present then you can configure a listener which presents a text menu based interface to telnet clients. The menu lets you stop, start and restart apps and also allows you to shut the server down. | | | |
| | IPAddress | | | The IP Address to listen on, e.g. 192.168.0.1 or 0.0.0.0 to listen on ALL interfaces. |
| | Port | | 5051 | The port to listen on. |
| | ListenBacklog | | | See TCPListener. |
| **<S2S …>** | | | | |
| | PingFrequency | | 2500 | The Ping frequency is how often a ping is sent out when no data is flowing. |
| | MaxInboundMessageSize | | | See TCPListener. |
| | MaxOutboundMessageSize | | | See TCPListener. |
| | RecvBufferSize | | | See TCPListener. |
| | SendBufferSize | | | See TCPListener. |
| | MaxPendingWrites | | | See TCPListener. |
| | MaxQueuedBuffers | | | See TCPListener. |
| | MaxPendingWritesMUX | | | same as MaxQueuedBuffers, but for MUX connections. Only applies if TuneFlowControlForMUX is not set. |
| | MaxQueuedBuffersMUX | | | same as MaxQueuedBuffers, but for MUX connections. Only applies if TuneFlowControlForMUX is not set. |
| | TuneFlowControlForMUX | | | MUX connections are configured with X * MaxPendingWrites and X * MaxQueuedBuffers |
| | DisableNagle | | TRUE | See TCPListener. |
| **<OutboundENet …>** | | | | |
| | GenerateOutboundCRC | | FALSE | If enabled, a 32bit CRC value is appended to each UDP header. |
| | MaxInboundMessageSize | | | See TCPListener. |
| | MaxOutboundMessageSize | | | See TCPListener. |
| | MTU | | 1200 | Maximum transmition unit at UDP level. Defaults 1200 - same default the client sdk's have. |
| | MaxChannels | | 255 | Maximum number of Channles the peer should support. Default is 255. |