

Core binding with affinity mask and threads management in FIX Antenna C++ products

- [Affinity mask settings](#)
 - [Application properties](#)
 - [Session properties](#)
 - [Supported affinity formats](#)
 - [Decimal](#)
 - [Hexadecimal](#)
 - [Cores-set format \(like Linux' taskset command\)](#)
 - [Support ranges](#)
 - [Backward compatibility](#)
 - [Troubleshooting](#)
 - [How to get information about threads](#)
- [Thread names in FIX Antenna C++](#)
 - [Add the threads' names to application logs' records.](#)
 - [The List of FIX Antenna spawned threads](#)
 - [Using the top for viewing thread names](#)
 - [How to change the thread name via FIX Antenna API](#)

Affinity mask settings


The Antenna allows users to change affinity masks for threads with different purposes. This allows the application to run functionality on a critical path at dedicated threads/CPU cores. It results in response time spreading reduction and helps to keep it minimal.

FIX Antenna C++ supports affinity masks up to 64 cores

FIX Antenna C++ threads:

- dedicated threads for session processing
 - [SendCpuAffinity](#)
 - [RecvCpuAffinity](#)
 - [CpuAffinity](#)
- threads pool dedicated to messages processing that is sharable between all sessions
 - [WorkerCpuAffinity](#)
- auxiliary antenna threads
 - [HelperCpuAffinity](#)
- Note: uncontrollable threads, e.g some timers

FIX Antenna C++ supports various affinity configuration formats: Decimal, Hexadecimal, Cores-set formats.

 NOTE: Some of the affinity masks don't use for FAST sessions

Affinity is specified as bitmask where a serial number of the bit is the serial number of the core. The default value is 0. It means that the engine will not change the thread affinity mask.

Application properties

- mask for all worker pool threads

```
WorkerCpuAffinity = 0
```

- mask for dispatcher and other auxiliary threads

```
HelperCpuAffinity = 0
```

Session properties

- mask for dedicated sending thread of a session. It makes sense only for an aggressive send mode

```
Session.Default.SendCpuAffinity = 0
```

- mask for the dedicated receiving thread of a session. It makes sense only for an aggressive receive mode

```
Session.Default.RecvCpuAffinity = 0
```

- mask for dedicated threads of a session. It makes sense only for aggressive modes

```
Session.Default.CpuAffinity = 0
```

Supported affinity formats

1. Decimal

```
WorkerCpuAffinity = 31
```

The setting above exposes cores #0,1,2,3,4
(see [Backward compatibility](#))


2. Hexadecimal

 Introduced in FA C++ 2.27.1

```
WorkerCpuAffinity = 0x1F3A
```

The setting above exposes cores #1,3,4,5,8,9,10,11,12

3. Cores-set format (like [Linux' taskset command](#))


 Introduced in FA C++ 2.27.1

```
# core numbers: integer values of set [0..63]
Session.Default.RecvCpuAffinity = 2-4

# separator character: ','
HelperCpuAffinity = 1,4

# grouping characters: '{' '}'
WorkerCpuAffinity = {0}
```

The settings above expose cores respectively #2,3,4; #1,4; #0;

 NOTE: To avoid erroneous reading with the cores-set format, use a grouping of characters in case of single-core in the mask:

```
WorkerCpuAffinity = 31 - it means cores #0,1,2,3,4
```

```
WorkerCpuAffinity ={31} - it means core #31
```

Support ranges

- Use comma-separated list instead of bitmask (decimal values in the range [0..63] separated by comma)

```
Session.Default.RecvCpuAffinity = {1,3,4,5,8,9,10,11,12}
```

- Supports ranges (like [Linux' taskset command](#))

```
Session.Default.RecvCpuAffinity = 1,3-5,8-12
```

- Supports initializing list brackets to separate sense groups

```
Session.Default.RecvCpuAffinity = {1,3-5,7},{12-15}
```

Backward compatibility

The new implementation does not affect the old version of the configuration.

The old Decimal and taskset-like formats have a different meaning of single number value. For any conflicting cases please use curly brackets '{}' notation.

by default - the decimal format is used.

Examples of conflicting configuration:

```
# The new settings uses core #2
WorkerCpuAffinity = 4
WorkerCpuAffinity = 0x4
WorkerCpuAffinity = {2}

# The old behavior uses cores # 1, 2, 3. Not using core #0
WorkerCpuAffinity = 14
# The old behavior uses cores #0
WorkerCpuAffinity = 1

# The new behavior uses cores #2 and #4
WorkerCpuAffinity = 0x14
# The new behavior uses core #14
WorkerCpuAffinity = {14}
# The new behavior uses core #2 and #4
WorkerCpuAffinity = {2,4}

# The old behavior means all cores for mapping (default)
WorkerCpuAffinity = 0
# The new behavior uses core #0
WorkerCpuAffinity = {0}
# The new representation to map all cores:
WorkerCpuAffinity = {0-63}
WorkerCpuAffinity = 0xFFFFFFFFFFFFFFFF
```

Troubleshooting

The output format for affinity in logs and other outputs in decimal style:

- The correct Affinity value:

```
[INFO] 20200127-10:46:21.703 [13888] [Engine] - WorkerCpuAffinity = {1-3}
[INFO] 20200127-10:30:46.303 [11448] [Engine] - Session <ESVR, CTG> : New session created CpuAffinity = 0
```

- The incorrect value/range for Affinity value:

```
[ERROR] Affinity parse error at pos=1 : token value '2147483647' exceeds the range [0..63]
[ERROR] Affinity parse error at pos=1 : token can not be parsed as integer inp: {abc}
```

- The incorrect interval of Affinity value:

```
[ERROR] Affinity parse error at pos=0 : token is neither interval nor decimal value inp: 1.3,4/5
```

How to get information about threads

Example configuration engine.properties of FIX Antenna C++ application (e.g. EchoServer, SimpleClient samples)

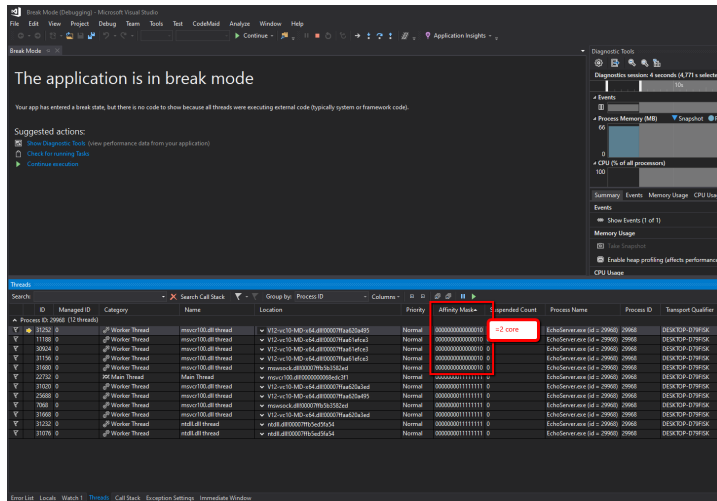
```
WorkerCpuAffinity = {1}    #(it's Worker# in CPU)  
HelperCpuAffinity = {1}   #(it's Dispatcher in CPU)
```

Check assigned CPU cores in Linux using htop:

```
$ htop -p `pidof EchoServer`
```

```
mc [user@ecs004008db.epam.com]~  
1 [ 0.0%] Tasks: 44, 134 thr; 2 running  
2 [|||||100.0%] Load average: 1.03 1.08 1.07  
Mem[|||||706M/7.30G] Uptime: 09:25:42  
Swp[ 0K/2.00G]  
  
CPU PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command  
2 8102 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.06 FA: Timer  
1 8100 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.64 ./EchoServer engine.properties  
1 8101 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.00 FA: AbsoluteTime  
2 8103 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.00 FA: Worker_0  
2 8104 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.00 FA: Worker_1  
2 8105 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.00 FA: Worker_2  
2 8106 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.00 FA: Dispatcher  
1 8107 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.00 FA: Timer  
1 8108 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.00 FA: Timer  
2 8109 user 20 0 955M 57476 9180 S 0.0 0.8 0:00.00 FA: Timer
```

Check assigned CPU cores in Windows using Visual Studio



i The internal threads of FIX Antenna were named firstly in FIX Antenna C++ v2.19.0 (r186).
The ability to rename the threads in FIX Antenna was introduced as well.

Thread names in FIX Antenna C++

Add the threads' names to application logs' records.

The thread name can be settled by `%(thread_name)` argument in `Log.File.Format` property in the `engine.properties`, e.g:

```
Log.File.Format = %date{ISO8601} %timezone %level%tablevel [%logger] %thread_name %thread %message
```

FIX Engine parameters#Log.File.Format

```
2019-08-16 12:35:53,663 UTC INFO [Engine] Worker_0 31940 -- "Session <MYAPP, CLIENT> : Asynchronously connecting to 127.0.0.1:9106"
```

The List of FIX Antenna spawned threads

FIX Antenna adds 'FA: ' prefix to a thread name for Linux.

	Linux thread name	FIX Antenna thread name	Description
1	FA: Timer	Timer	Several timer threads which can be used for: <ul style="list-style-type: none">• Internal scheduling events. Stop, Start, Reconnect a session• Scheduling for delayed message delivery. (non-public functionality).• Scheduled backup logs (non-public functionality).• Some internal health checks (non-public functionality)• Timer for connection timeout checks (non-public functionality)• Public health checks like control of used memory for windows.• Heartbeats timer.
2	FA: Worker_0 ... FA: Worker_N	Worker_0 ... Worker_N	Several threads. Session processing (receive/send) threads from thread pool used by EVEN and DIRECT_SEND modes and configuring by NumberOfWorkers parameter
3	FA: Dispatcher	Dispatcher	Thread that is listening and handling incoming and outgoing connections. Performs protection from abnormal behavior (DDOS protection)
4	FA: AbsolutTime	AbsolutTimer	Public timer scheduling tasks.
5	FA: ExclusiveRec	ExclusiveReceiveTaskDedicatedWorker	A dedicated thread for receiving in aggressive mode.
6	FA: NewInLinkTa	NewInLinkTask	Thread that accepts a new connection.
7	FA: ExclusiveSen	ExclusiveSendTaskDedicatedWorker	A dedicated thread to send in the aggressive mode if sending try in the current thread has failed.
8	FA: AsyncConnec	AsyncConnectTask	Asynchronous connection to the initiator.
9	FA: DBLDispatcher	DBLDispatcher	Thread that is listening and handling incoming and outgoing connections for Myricom DBL network cards.
10	FA: LSCommunica	LSCommunicator_Impl	<ul style="list-style-type: none">• Some internal utility threads.
11		<Unnamed threads>	<ul style="list-style-type: none">• Some internal utility threads.• TCP logging thread (TCPAppender for log4cpp)

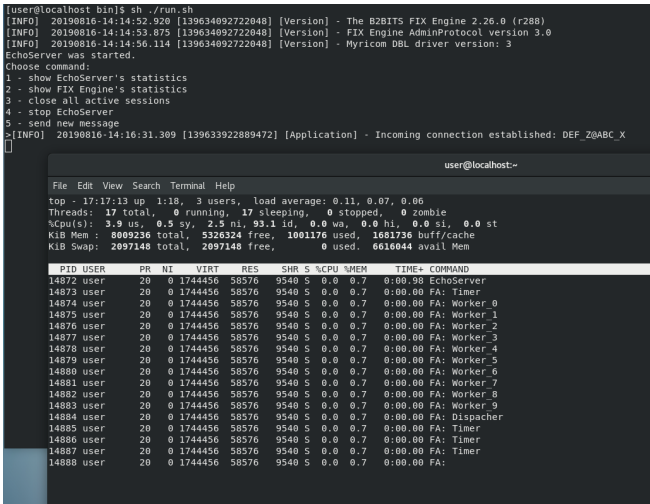
Using the top for viewing thread names

The threads names can be viewed using a command:

```
top -H
```

```
[user@localhost bin]$ sh -x ./run.sh
[INFO] 20190816-14:14:52.920 [139634092722048] [Version] - The B2BITS FIX Engine 2.26.0 (r288)
[INFO] 20190816-14:14:53.875 [139634092722048] [Version] - FIX Engine AdminProtocol version 3.0
[INFO] 20190816-14:14:56.114 [139634092722048] [Version] - Myricon DBL driver version: 3
EchoServer was started.
Choose command:
1 - show EchoServer's statistics
2 - show FIX Engine's statistics
3 - close all active sessions
4 - stop EchoServer
5 - send new message
> [INFO] 20190816-14:16:31.309 [139633922889472] [Application] - Incoming connection established: DEF_Z0ABC_X

```



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
14872	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.98 EchoServer
14873	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Timer
14874	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_0
14875	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_1
14876	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_2
14877	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_3
14878	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_4
14879	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_5
14880	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_6
14881	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_7
14882	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_8
14883	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Worker_9
14884	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Dispatcher
14885	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Timer
14886	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Timer
14887	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Timer
14888	user	20	0	1744456	58576	9540	S	0.0	0.7	0:00.00 FA: Timer

How to change the thread name via FIX Antenna API

The thread name can be changed with a function:

```
static void System::Thread::setName ( char const * name );
```