FIX Antenna .NET Core: Release notes

FIX Antenna™ .NET Core 0.9.2

31/Aug/2020

FIX Antenna™ .NET Core 0.9.2 brings compatibility with the latest version of .NET Framework - 4.8.

The Engine was tested under .NET Core 3.1 and under .NET Framework 4.8 and is intended to be used under these platforms.
The Engine itself is based on .NET Standard 2.0 and shares the same set of features under both platforms.

> User should manually clean 'obj' and 'build' folders inside sample application's folder in case if it is needed to sequentially run the same sample for different target platform (for example, EchoServer for .NET Core after .NET Framework 4.8)

Features and improvements.

- Optimized memory allocation on message receiving.
- Added ability to choose IP protocol version between IPv4 or IPv6. Dual mode is used by default.

  > Please note: IPv4 should be specified for OpenOnload

Bug fixes.

- FixServer does not shutdown on calling Stop() function in case FixSessionManager was not explicitly stopped.

FIX Antenna™ .NET Core 0.9.0

FIX Antenna™ .NET Core is a high-performance low latency .NET FIX Engine.

The engine is based on .NET Standard 2.0 and is compatible with any platform that supports .NET Standard 2.0 and further.

The Engine is intended to be used under .NET Core starting from version 3.1 and further and was tested under this platform.

One of the key benefits of using .NET Standard and .NET Core is the ability to use Engine under different OS: Windows, Linux, Mac.

Requirements & Compatibility

1. Libraries compatibility: any OS with .NET platform which supports .NET Standard 2.0 and further (.NET Core is preferable).
2. Samples compatibility: any OS with installed .NET Core 3.1 and further.
3. Samples and libraries were tested with .NET Core under Windows 10 and Linux Ubuntu 16.04.

FIX Engine

1. Supports FIX 4.0 - FIX 4.4, FIX 5.0, FIX 5.0 SP1, FIX 5.0 SP2
2. Supports all FIX message types (pre-trade, trade, post-trade, market data, etc.)
3. Customizable FIX protocol with user-defined tags and message types
4. Sequence numbers management
5. Microseconds in FIX tags (MIFID II)
6. Smart resend request handling
7. Validation against FIX dictionaries
8. Unregistered acceptors handling
9. Standard FIX routing based on DeliverTo and OnBehalfOf fields
10. Store messages while disconnected
11. Switch to the backup connection
12. Round-robin reconnect option to multiple backup destinations for an initiator session
13. Administrative Plugin for Fix Antenna

Sessions Processing

1. Session Qualifiers
2. Multiple FIX sessions
3. Initiator/Acceptor Sessions
4. Auto-reconnect Session
5. Transient Sessions
6. Restore state after failure (Persistent Session)

7. Pluggable session level

Storage

1. Supported Storage Types:
    a. Null storage
    b. In-memory storage
    c. Persistent storage
    d. Persistent Memory Mapped based storage
2. Storage management (get a message by seq. num, get creation time)

Performance Tuning

1. Ability to enable or disable Nagle's algorithm to minimize latency or maximize throughput
2. Ability to manipulate the internal outgoing queue size to get maximum throughput (process messages in batch) or lower latency (minimal time in queue)
3. Ability to use different levels of message validation to balance between reasonable correctness and good performance:
    a. Well formedness validation
    b. Validation of allowed message fields
    c. Validation of required message fields
    d. Validation of message fields order
    e. Validation of duplicated message fields
    f. Validation of field values according to defined data types
    g. Validation of repeating group fields
    h. Conditionally required fields
4. Ability to choose the sending mode. Synchronous sending gives lower latency, but asynchronous is preferable for getting a better throughput
5. Ability to configure CPU affinity to enable the binding and unbinding a thread to a CPU, so that the thread will execute only on the designated CPU rather than any CPU
6. Setting send and receive socket buffer sizes for TCP transport options

Configuration

1. Supports configuration in properties file format
2. Ability to configure default behavior and custom sessions in the same configuration file

Monitoring and administration

1. Proprietary built-in remote monitoring and administrative interface for sessions monitoring and management
2. Public API to override and/or extend existing administrative instructions
3. Rich monitoring and administration GUI (FIXICC) out of the box

Message composition API

1. Provides the following functionality:
    a. Creating a message from a raw FIX string (parse)
    b. Serializing a FIX object message to raw FIX string
    c. Adding, removing, modifying FIX fields and repeating groups
    d. The ability of internal pooling to reduce memory allocations
    e. FIX flat message model - generic model to work with the abstract FIXMessage class via fields and groups getters and setters, which gives the highest performance
2. RG API - API for working with repeating groups. Similar to FIX flat message model but allow to work with structures
3. Prepared messages - message template for the faster sending the messages with the same structure but with different values
4. Efficient getters and setters for working with values as with primitive type to reduce garbage production

Security

1. Logon Customization
2. Standard FIX authorization utilizing username and password fields in FIX Logon message
3. Strategies for accepting incoming sessions:
    a. Auto-accept incoming sessions (to simplify development and testing)
    b. Auto-decline non-pre-configured sessions
4. IP range based whitelisting for incoming connections
5. CME Secure Logon support

Options

1. Option "disconnectOnLogonHeartbeatMismatch"
2. "resetThreshold" configuration option (handling of counterparties which ignores sequence reset and continues to send old sequence numbers)
3. Option "resetQueueOnLowSequence"

Services

1. Licensing system

2. Logging to text files
3. Logging to the socket (Splunk) via NLog
4. ASCII codes in FIX logs file names
5. Configuration of session parameters from .XML

Packaging

1. A zip archive package is provided
2. Sample EchoServer
3. Sample Connect to gateway
4. User’s guide
5. Reference Manual (with a description of all classes and methods)

Safety

1. Listen port management (Listen port per session)

For more information please visit Engine documentation.