Configuring FIX Sessions

Overview

The settings of FIX sessions in the FIXEdge are defined in two configuration files: *'FIXEdge.properties'* and *'engine.properties'*. There are common settings for all FIX sessions and individual settings for each registered FIX session. The common settings are located in both configuration files and described in the part 'Common Settings of Sessions'. The individual settings of registered FIX session are located in the file *'FIXEdge.properties*' and described in the part 'Settings of registered FIX Sessions'.

Basic concepts related to the FIX session and its configuring are described in part 'Introduction and Terminology'

Introduction and Terminology

A FIX session is a bi-directional stream of ordered FIX messages between two counterparties (FIX applications). The FIX message transmission is performed within the FIX connection. Each FIX connection comprises of three parts: *logon*, message exchange and *logout*. Parts logon and logout are used for initiation and termination of the FIX session, correspondingly.

Initiator and Acceptor

In the FIX session, one counterparty plays the role of an Initiator and other - plays the role of an Acceptor. The **Initiator** is responsible for establishing the connection (TCP/IP) and sends the first **Logon** message to the Acceptor. The **Acceptor** has a responsibility to perform authentication and replies the **confirming Logon message** to the Initiator as a declaration that FIX session has been established. In case of a connection failure, the Acceptor waits when the Initiator will initiate reconnection.

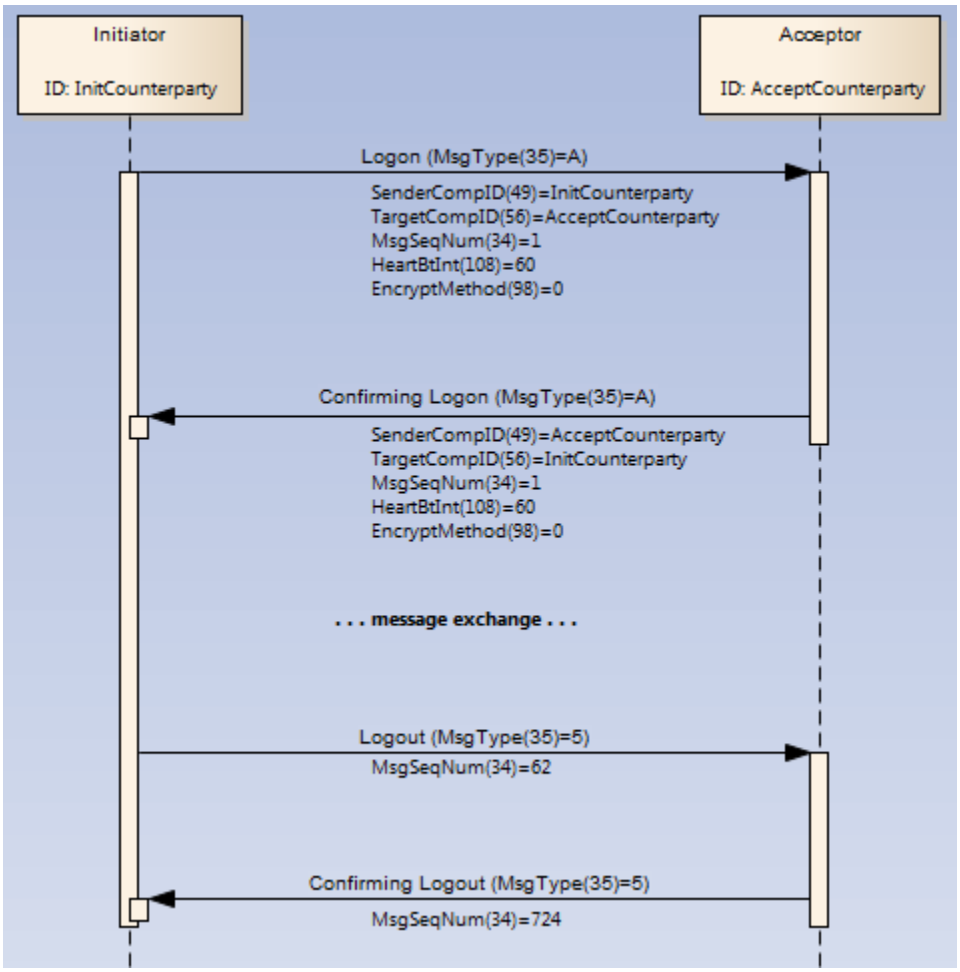

Figure 1. FIX connection

When the counterparty decides to terminate the FIX session, it sends the **Logout** message to other party. The **confirming Logout message** should be generated in response.

Session Identification

Both the Initiator and the Acceptor have own logical textual identifiers that must be bilaterally agreed between counterparties. These identifiers are included in each FIX message: *SenderCompID* (49) and *TargetCompID* (56) tags.

All FIX messages must have the same version of the FIX protocol (*BeginString* (8) tag) in the one FIX session. Therefore, message's tags (***SenderCompID*** , ***TargetCompID*** and ***BeginString***) identify a FIX session.

Registered and Unregistered FIXEdge Sessions

FIXEdge can work with registered FIX sessions as well as with unregistered FIX sessions.

Working with registered sessions is a standard mode. This mode assumes that before using, each FIX session must be registered in the configuration file ‘ *FIXEdge.properties*’, where the session has own session name and a set of properties that describes specifics of the session behavior. Refer to part ‘Settings of Registered FIX Sessions’ for more information.

Working with unregistered sessions is an auxiliary mode for FIXEdge. The configuration property *UnregisteredAcceptor.CreateSession* can be set to ‘true’ or ‘false’ to turn this mode on or off, correspondingly.

All registered FIX sessions are created in the FIXEdge memory at the time of FIXEdge starting.

The session - initiator sends the Logon messages to establish the connection and waits for a confirming Logon message.

The session-acceptor waits for incoming logon right after creation. When FIXEdge receives incoming Logon message it tries to find the corresponding session-acceptor in the session list, i.e. session-acceptor with *SenderCompID* and *TargetCompID* equal to the *TargetCompID* and *SenderCompID* extracted from the received Logon message accordingly. If such a session exists FIXEdge checks credentials and then links incoming message flow to the found session-acceptor. Otherwise, FIXEdge treats this situation as an "**unknown connection**".

If *UnregisteredAcceptor.CreateSession* property is set to ‘false’, then incoming Logon messages from unknown connections are ignored.

If *UnregisteredAcceptor.CreateSession* property is set to ‘true’ then each incoming Logon message from unknown connection initiates creation of the session-acceptor.

Such FIX sessions that are created without preliminary settings are called ‘**unregistered acceptor**.’ All unregistered sessions have similar behavior because is described with the same set of properties (refer to the part ‘Settings of Unregistered Acceptors’ for more information).

> **Note:** All incoming application messages having been received before session initiation (before Logon message) are rejected

Session Name

Each FIX session registered in FIXEdge has its own name. The session name must be unique among names of FIX Sessions and Transport Adaptor Sessions because the session name can be also used for session definition as a Source or a Destination in the business layer rules.
It must contain characters with codes from ASCII [30] to ASCII [128] and must begin with a letter, digit or underlining symbol ‘_’.

Full list of Session names are registered in property *FixLayer.FixEngine.Sessions*.

Session Time Schedule

It is recommended that a new FIX session is established once within each 24 hour period. The counterparties must bilaterally agree as to when FIX sessions are to be started/stopped based upon individual system and time zone requirements. There are *DefaultStartTime* and *DefaultTerminateTime* properties to specify the default session working time

Multiple logins

The several FIX connections could be sequentially established (not concurrent) in the one FIX session. I.e. counterparties can connect and disconnect multiple times while maintaining a single FIX session, meaning that a FIX session spans *multiple logins*.

For registered sessions, the property *RecreateOnLogout* defines the possibility of multiple logins. If the property is set to ‘false’ the session is removed from the list of sessions after successful disconnection (by Logout message). If it is set to 'true' the session will be recreated after disconnection.

> **Note**: Disconnection without the exchange of Logout messages is interpreted as an abnormal condition (failure connection). In case of when the connection was broken without a Logout message, such a session is considered as deliberately disconnected "**non-gracefully**".

The Initiator is responsible for restoring connection once it is broken. When connection error is detected the Initiator starts a **reconnection process**: restoring the telecommunication link and exchanging of Logon messages. The number of reconnection attempts and delay can be set in the properties *Reconnect.MaxTries* and *Reconnect.Interval*.

Username and Password

Username (553) and Password (554) can be sent in Logon message for a security purpose for both Active and Backup connections. It is also possible to send Username and Password in different tags other than standard ones (for instance in order to increase security). It can be done by setting the parameters UsernameTag and Passwordtag (available for both Active and Backup connections).

Example: FixLayer.FixEngine.Session.Session_Name.UsernameTag=10553

FixLayer.FixEngine.Session.Session_Name.PasswordTag=10554

FixLayer.FixEngine.Session.Session_Name.Username=user

FixLayer.FixEngine.Session.Session_Name.Password=111

In this case, Logon message would contain the custom tags 10553 and 10554 instead of standard Username (553) and Password (554) tags:
|10553=user|10554=111|

UsernameTag and PasswordTag can be defined in a custom Logon message (for Active connection only). In case they are not present in a custom Logon message but exist in FixEdge.properties file, the actual Logon message would contain those two predefined tags and corresponding values.

Default values: If any of these parameters is not set the default value will be used, i.e. the default value for UsernameTag is 553 and the default value for PasswordTag is 554.

Sequence Number Handling

The FIX protocol assumes complete ordered delivery of messages between counterparties. It means there is no acknowledgement of individual message delivery. Instead of this, the following control mechanism is used:

- Each FIX message has a unique sequence number (*MsgSecNum* (34) tag).
- Counterparties must control that sequence numbers of outgoing messages and incoming messages, are continuous and sequential.

  For this, the counterparty establishes an independent incoming and outgoing sequence numbers for each FIX session. Sequence numbers are initialized at the start of the FIX session starting at 1 (one) and increment throughout the session.

  Incoming sequence number is a counter for incoming messages.

  Outgoing sequence number is a counter for outgoing messages.

Note: It is possible to change starting number of the sequence counters, refer to *InSeqNum* and *OutSeqNum* for details.

Monitoring of incoming sequence number allows counterparties to identify and react onto disturbing the synchronization. There are two types of sequence numbers desynchronization:

- **Sequence number is too high**, i.e. a gap in sequence numbers is detected. It indicates that some messages were missed. The counterparty initiates standard message recovery process.
- **Sequence number is too low** – it indicates some serious problem and leads to immediate session termination

- In order to maintain the integrity of connection **Heartbeat** and **Test Request** messages are used.

In accordance with FIX protocol standard, after the correct FIX session termination (by the Logout message) sequence numbers are reset to 1. If the connection is terminated '*non-gracefully*' sequence numbers will continue when the session is restored.

In fact, a lot of service providers never reset sequence numbers during the day. There are also some, who reset sequence numbers once per week. There are several cases to handle such deviation from standard in FIXEdge:

- **Intraday Logout Tolerance mode** makes that FIX session sequence numbers always will continue on the next connection.
- **Force Sequence Number Reset mode** makes the session reset sequence number every time on logon and forces the counterparty to do the same.

Intraday Logout Tolerance Mode

In the **Intraday Logout Tolerance mode**, the FIXEdge will continue sequence numbers even in case the FIX session is re-established after correct termination (by the Logout message). The configuration property *IntradayLogoutTolerance* can be set to 'true' or 'false' to turn this mode on or off, correspondingly.

If set to 'false' then the session does not exist after correct termination and hence a newly created session with the same *SenderCompID* and *TargetCompID* starts with 1 as it is specified by the FIX protocol.

In the Intraday Logout Tolerance mode, the sequence number is never reset during the day It means that the Initiator should initiate session recovery by sending Logon message with *MsgSeqNum* = <last outgoing sequence number> + 1; and will expect confirming Logon message with *MsgSeqNum* = <last incoming sequence number> + 1. If a gap is detected, the standard message recovery or gap filling process takes place.

> **Note:** It is crucial that both sides – the initiator and the acceptor – work in the same mode. Otherwise, the '*sequence number is too low*' fatal error appears when the session is reestablished.

During the end-of-day procedure, session logs are archived. So, because the FIXEdge stores session state in logs files, absence of such file will be treated as the session being created from scratch and hence sequence numbers start from 1.

Force Sequence Number Reset Mode

In the **Force Sequence Number Reset mode**, the FIXEdge will reset outgoing and incoming sequence number on every time on logon and force the counterparty to do the same via sending the tag *ResetSeqNumFlag (141) = Y* in Logon message.

The configuration property *ForceSeqNumReset* is used to set the mode. The following values can be applied for the property:

- Value '0' or 'false' – mode is off, do not use tag *ResetSeqNumFlag (141)*
- Value '1' or 'true' – Enable SeqNumreset at the first time of session initiation
- Value '2' – reset sequence number at every logon

This mode is used to resolve '*sequence number is too low*' problem. It usually occurs as a result of applying the Intraday Logout Tolerance mode only by one of a counterparty or after '*non-graceful*' session termination log-files are cleared on the one side and kept on the other side.

> **Note:** This is not a recommended way since messages sent during inactivity time will be lost. Also, it is recommended to make sure that other counterparty supports this functionality before relying on it.

The matrix of different combinations

ForceSeqNumReset	**0**	**1**	**2**
IntradayLogoutTolerance			
true	FIXEdge will always use the same set of logs and will never reset sequence numbers. Afterlogscleanup FIXEdge will send Logon with SeqNum=1 and without ResetSeqNum flag. ⚠ This is the most safe combination, but you have to make sure that your counterparties don't reset sequence numbers on their side. Otherwise, they will send you confirming Logon with SeqNum=1 and FIXEdge will terminate the session.	FIXEdge will send only first Logon with ResetSeqNum flag (141=Y), after that it will always use the same set of logs and will reset sequence numbers only after logs cleanup. Whether this is the first Logon or not, is defined by presence of logs in log directory.	FIXEdge will always send Logon with ResetSeqNum flag (141=Y). IntradayLogoutTolerance doesn't work for this case,new set of logs will be created after each Logout.

false	FIXEdge will create new set of logs after Logout. Next Logon will be sent with SeqNum=1 and without ResetSeqNum flag.	FIXEdge will create new set of logs after Logout. Next Logon will be sent with SeqNum=1 and with ResetSeqNum flag (141=Y).	Each Logon will be sent with SeqNum=1 and with ResetSeqNum flag (141=Y).

Other Cases of Sequence Number Handling

The property *ResetSeqNumAfter24hours* can be used for sequence number resetting after each 24 hour period via sending the tag *ResetSeqNumFlag (141) = Y* in Logon message.

Heartbeat and Test Request Messages

During periods of message inactivity, counterparties have to generate the **Heartbeat** messages at a regular pre-defined time interval (**Heartbeat Interval**). The recipient also has to generate the Heartbeat message in response.

The heartbeat interval is declared by the Initiator using the *HeartBtInt* (108) tag in the Logon message. If *HeartBtInt* tag is set to zero then no regular Heartbeat messages will be generated.

When either counterparty has not received any messages for some period of inactivity (heartbeat interval + delta (reasonable transmission time)), it has to generate a **Test Request** message. The other counterparty has to send Heartbeat message in response to the Test Request message.

In this manner, counterparties regularly are checking sequence numbers or communication line status:

- When a gap in sequence numbers is detected, the counterparty initiates **the standard message recovery process**.
- It is considered that FIX Connection is lost (state "Telecommunication link failed") when response Heartbeat message is not received after the heartbeat interval plus delta.

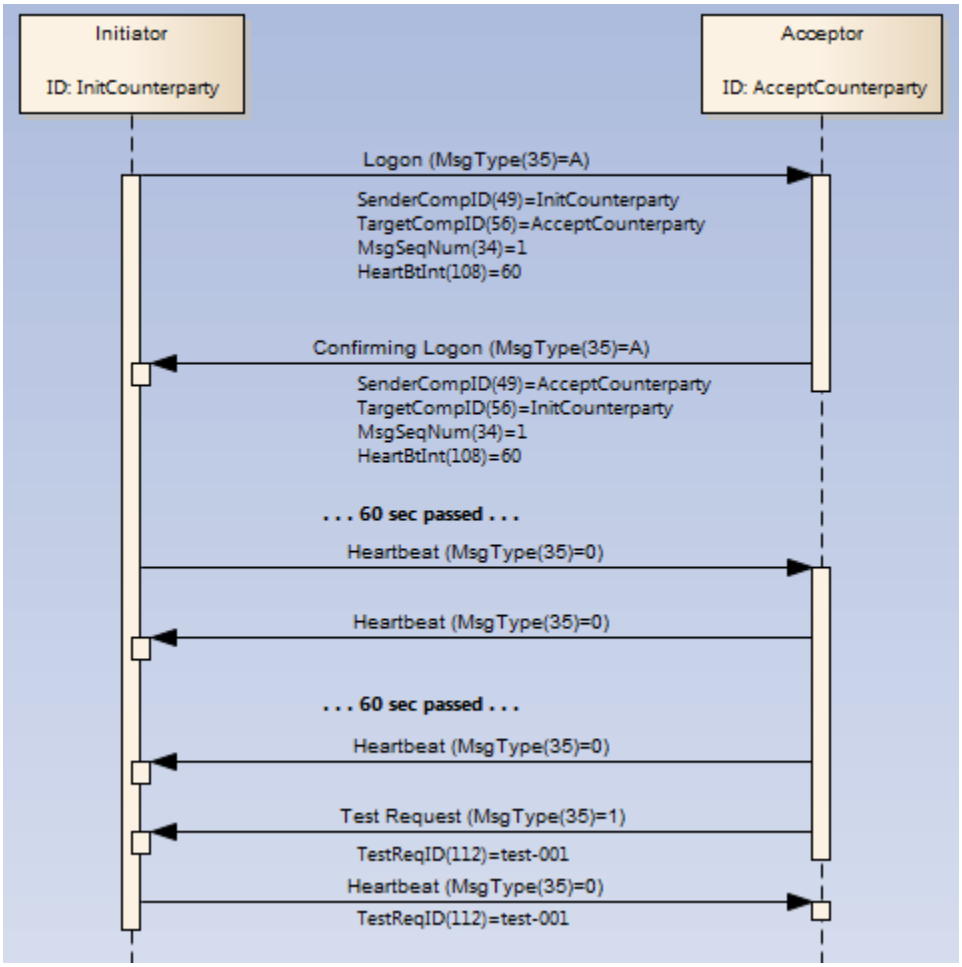

Figure ... Heartbeats and Test requests

Standard Message Recovery Process

When the gap in sequence numbers is detected the each counterparty can initiate a request for message retransmission - **Resend Request** message. The counterparty can request a single message or range of missed messages: *BeginSeqNo* (7) and *EndSeqNo* (16) tags are used.

In response to the Resend Request message the other counterparty retransmits the application messages or sends the **Sequence Reset** messages. The Sequence Reset message (in 'Gap Fill' mode) is used when sender wants to skip some messages for the following reasons: when missed messages were application messages and became obsolete or when missed messages were just session messages.

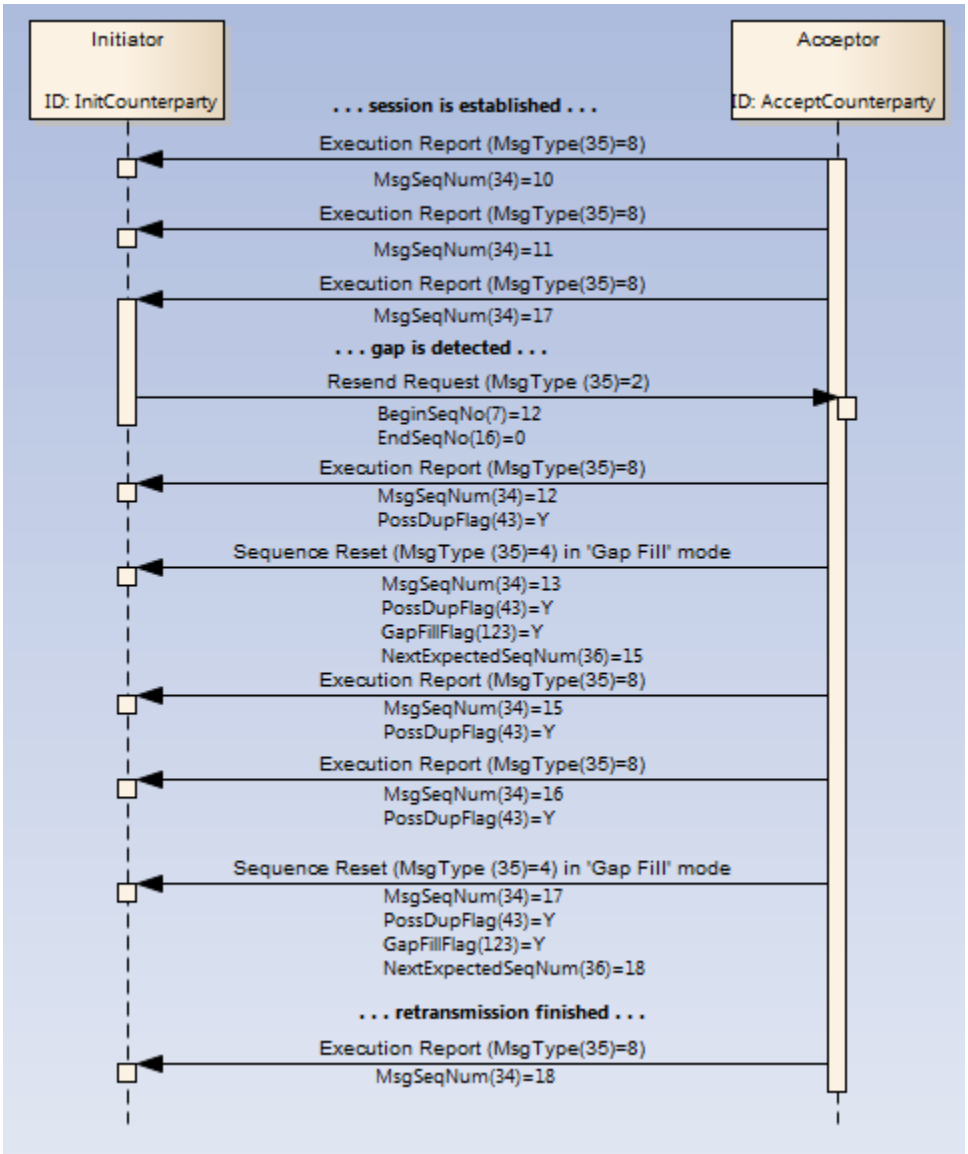

Figure ... Message recovery

The properties *OutgoingMessagesStorageSize* and *ResendMessagesBlockSize* are used for limiting the number of requested and sent messages.

Logging Storage Type

According to the FIXprotocol session is responsible for outgoing message flow. This means that any time counterparty may request retransmission of the pr eviously sent messages. Therefore, the session must store all sent messages to some storage.

Moreover, to successful restore the session after failure the following logs of data have to be stored:

- Session parameters (SenderCompID, TargetCompID, FIX version, etc.)
- Session incoming and outgoing sequence numbers.
- Incomming messages. Incoming message is stored to the log after the business layer processes it (optional).
- Outgoing messages. Outgoing message is stored to the log before sending.

Incoming messages are stored in files with the '.in' extension. Outgoing messages are stored in files with the '.out' extension.

There are the following modes of storage that can be defined for the FIX session:

Type	Description

Persistent	In case of the persistent session, the logs are stored to the files. This guarantees that session can recover its state (sent messages and latest sequence numbers) after connection failure but it decreases the performance.
PersistentMM	In this case, all related information is stored to the persistent memory mapped files.
Split Persistent	Like persistent session, in this case the information is stored to the log-files but there is a log-file size limitation. When log-file size exceeds the limit then new log-file will be created. At the same time, the previous log-files become available to change or delete because the FIXEdge does not lock them. So, it is possible to recover the session state and to re-transmit of the previously sent messages from old log-files if these files have not been changed.
Transient	Transient session stores all related information in memory. This means that this type of session has greater throughput and lower latency. However, such session cannot be restored after failure, and hence message loss is possible. This session type is recommended when recovery after failure is not required, i.e. when every time session is connected from scratch. This applicable for market data sessions, when client has to subscribe and request snapshot on every logon and hence recovery is not required. Another variant is when persistent already exists in application so having yet another storage of recovery information is redundant. In the latter case the application is responsible to restore session state after creation and before connect.
Null	No storage is used. This means that incoming and outgoing messages are not stored. This session type is recommended when messages resend, recovery after failure and logging are not required.

The mode of storage is defined by the different properties *FixLayer.FixEngine.Session.Session_Name.StorageType* and *UnregisteredAcceptor.SessionStorageType*, for registered and unregistered sessions correspondingly. Refer also to part 'Logging and Backup Settings' for more details.

Encryption

The exchange of sensitive data across public carrier networks may make it advisable to employ data encryption techniques to mask the application messages. FIXEdge can perform message encryption in the session by means of several cryptographic technologies: PGP, PEM, PKCS, DES or DES-MD5.

The choice of encryption method is determined by mutual agreement of the two counterparties of FIX session.

The required encryption method is defined in the session parameter EncryptMethod. FIXEdge requires GNU Privacy Guard command-line utility to use encryption. Others settings of encryption are set in a special configuration file.

> **Note:** Problems in use of encryption may occur in case of separated configuration of sessions that will be connected subsequently (i.e. when the related sessions have different encryption methods). In this case a session will be started, but errors will be logged during the action of message sending. The server does not decrypt the encrypted messages on the business layer, so only unencrypted tags (such as *TargetCompID*, *SenderCompID*, etc.) could be used for routing purposes.

Usage the SSL is an alternative to encryption of FIX messages. In case of SSL, the protection will be spread to the whole connection, and encryption of messages is not required.

Common Settings of Sessions

This part contains a list of common properties and their detailed description related to registered and unregistered sessions. These properties are stored in both configuration files '*FIXEdge.properties*' and *'engine.properties'*.

Business Rules parameters

BusinessLayer.RoutingRules

Mandatory.PathtoBusiness Layer routing rules configuration file in XML format.

Default value: *BusinessLayer.RoutingRules = FIXEdge1/conf/BL_Config.xml*

BusinessLayer.JS.ExecuteInParallel

Optional.Upperlimit of JS scripts that can be executed in parallel. Each parallel executed script requires separate JS environment. Set a positive number or "auto".

Defaultvalueis*auto*.

Configuration File FIXEdge.properties

The file '*FIXEdge.properties'* contains four common properties that describe registered sessions, see below:

FixLayer.FixEngine.Sessions

Mandatory. Contains a comma separated list of names of FIX Sessions registered in the FIXEdge

Example: FixLayer.FixEngine.Sessions = TestFIXAcceptor, TestFIXInitiator

Default value: FixLayer.FixEngine.Sessions = **TestFIXAcceptor - this session is defined in installation procedure**

FixLayer.FixEngine.Sessions.ArchivePath

Optional. Defines the path to move log files after the session is closed.

> **Note**: directory must exist otherwise the backup procedure will fail.
>
> **Attention**: it shouldn't point to path from **BackupDirectory** (defined in engine.properties), otherwise an error message will occur on session start, and session won't start

Default value: FixLayer.FixEngine.Sessions.ArchivePath = **FixEdge1/log/archive**

Configuration File Engine.properties

Backup Settings

HiddenLogonCredentials

Optional. Provides an option to mask password in in/out log files.

Valid values: "true" | "false"

Default value: HiddenLogonCredentials = **false**

LogDirectory

Mandatory. Defines the path of the directory in which the logs for all incoming (if *LogIncomingMessages* is set to 'true') and outgoing FIX messages are stored. A path is specified relative to the EngineRoot directory.

Default value: LogDirectory = **FixEdge1/log**

LogIncomingMessages

Mandatory. Provides an option to log incoming FIX messages (those received) from a counterparty FIXEdge.Log-file will be stored in the directory specified by the LogDirectory property in a file with extension "in".

Valid values: "true" | "false"

Default value: LogIncomingMessages = **true**

TotalOutgoingStorageMemoryLimit

Mandatory. Defines the total amount of the memory (in MB) that active session may use. If the limit is exceeded then the sessions will be closed '*non-gracefully*'.

Valid values: The valuemustbeinteger and not negative. The value "0' means that allocated memory is unlimited.

Default value: TotalOutgoingStorageMemoryLimit = **0**

EnableIncrementalLogFileCreation

Mandatory. Defines whether the size of disc space for session logging must be reserved to 10Mb.

Valid values: "true" | "false". The value 'true' means that the 10 Mb of disc space will be reserved for logging.Iflog-files reaches 10Mbs, another 10Mbs will be reserved and so on.

> **Note**: If set to 'true' the FIXEdge performance is greatly increased.

Default value: EnableIncrementalLogFileCreation = **false**

BackupDirectory

Mandatory. Defines a relative path to the backup folder. This folder will be used for message storage files of the backup connections.

Default value: BackupDirectory = **FixEdge1/log/backup**

Backup Connection

Optional. Provides an option to switch from an active session to a backup one on a predefined backup port/host. Backup session has its own configuration settings, yet not every option is supported for Backup connection (for example UsernameTag and PasswordTag).

Example:

FixLayer.FixEngine.Session.Session_Name.Backup.Host = XXXX
FixLayer.FixEngine.Session.Session_Name.Backup.Port = YYYY

Note: if the primary connection is an SSL connection, then the backup connection will use the same SSL parameters (certificate, protocols) as the primary connection.

Logging Settings

Detailed information about Logging parameters is available here: FIX Engine parameters#Loggingparameters

Integrity Control Settings

LogonTimeFrame

Mandatory. Defines the time period (in seconds) after which a session is 'non-gracefully terminated' when confirming Logon message is not received to a Logon message. The corresponding Logout message must be sent to the counterparty.

Valid values: The value must be integer and not negative. The recommended value is 30 seconds for dedicated connections or private networks. Trading connections via the internet will require calibration. If it is set to "0", then the time period is unlimited.

Default value: LogonTimeFrame = **10**

LogoutTimeFrame

Mandatory. Defines the time period (in seconds) after which a session is automatically terminated when confirming Logout message is not received to a Logout message.

Valid values: The value must be an integer and greater than 0. The recommended value is 10 seconds for dedicated connections or private networks. Trading connections via the internet will require calibration.

Default value: LogoutTimeFrame = **10**

IntradayLogoutTolerance

Mandatory. Defines whether the Intraday Logout Tolerance Mode will be used. The property is applied to all of the sessions which have not possessed the own parameter IntradayLogoutTolerance.

Valid values: "true" | "false"

Default value: IntradayLogoutTolerance = **false**

ReasonableTransmissionTime

Mandatory. This parameter specifies the delta (additional time) to the heartbeat interval that FIXEdge has to wait before sending Test Request message or will wait for a Heartbeat message in response to Test Request message. Refer to part 'HeartBeat and Test Request messages' for more information.

The parameter is specified in a percentage of Heartbeat Interval (Heartbeat Interval/100). The recommended value is twenty percent.

Default value: ReasonableTransmissionTime = **20**

ForceSeqNumReset

Mandatory. Defines using of the Force Sequence Number Reset Mode for all sessions by default. Used when parameter *ForseSeqNumReset* is not set for a particular session

Valid values: "0" or "false" | "1" or "true" | "2"

Default value: ForceSeqNumReset = **false**

ResetSeqNumAfter24hours

Optional. Defines whether FIXEdge sends a Logon message with the *ResetSeqNumFlag* (141) tag after each 24 hour period of session's activity to establish a new set of sequence numbers (starting with 1).

Valid values: "true" | "false"

> **Note**: This option does not affect sessions that use version 4.0 of the FIX protocol.

Default value: ResetSeqNumAfter24hours = **false**

DuplicateResendRequestLimit

Mandatory. Defines how many the same Resend Request messages can be received before calling the callback Application::onResendRequestLoop.

Valid values: The value must be an integer and not less than 0. This option is disabled if the value is less than 2.

Default value: DuplicateResendRequestLimit = **0**

Settings of Session Reconnection

Reconnect.MaxTries

Mandatory. Defines the number of attempts to restore the session (see also ForceReconnect). The session is considered as restored if the telecommunication link is restored and the exchange of Logon messages is successful.

Valid values: 0, 1, 2, ... The value "-1" means that the number of attempts is unlimited.

Default Value: Reconnect.MaxTries = **5**

Reconnect.Interval

Mandatory. Defines the time interval (in milliseconds) between reconnection attempts in order to restore a communications link.

Valid values: The value must be integer and greater than 0. The recommended value is 1000 for dedicated connections and private networks. Internet connections require calibration.

Default value: Reconnect. Interval = **5000**

Settings of Message Validation

MessageMustBeValidated

Mandatory. Defines whether the each application message is checked for existence of required tags.

Valid values: "true" | "false"

If set to ‘true’ then all application level messages are checked. If set to ‘false’ then the responsibility for message validity rests with the counterparty.

> **Note:** Session level messages are checked in all cases. The recommended setting is ‘false’. Changing this value will impact up on the FIXEdge performance. Use ‘true’ for debugging and ‘false’ to increase performance.

Default value: MessageMustBeValidated = **false**

VerifyTagsValues

Optional. Defines whether the each application message is verified for tag values

Valid values: "true" | "false"

If set to ‘true’ then all messages will be verified. If set to 'false' then the responsibility for message verification rests with the counterparty.

> **Note:** Ignored if *MessageMustBeValidated* = false

Default value: VerifyTagsValues = **true**

ProhibitUnknownTags

Optional. Defines whether the each application message is checked for the unknown tags.

Valid values: "true" | "false".

If set to ‘false’ then all application messages with unknown tags will be handled. If set to ‘true’ then FIXEdge will reject messages with unknown tags.

> **Note**: Ignored if *MessageMustBeValidated* = false

Default value: ProhibitUnknownTags = **false**

VerifyReperatingGroupBounds

Optional. Defines whether the each application message is verified for repeating group size.

Valid values: "true" | "false".

If set to ‘true’ then FIXEdge will reject messages with incorrect repeating group size.Otherwise message will be passed to the application layer. In this case the responsibility for message validity rests with the counterparty.

> **Note**: Ignored if *MessageMustBeValidated* = false

Default value: VerifyReperatingGroupBounds = **true**

IgnoreUnknownFields

Mandatory. If true then any field that is unknown to dictionary will be ignored. Should be false for development, true for production because if false FA stores all unknown fields in the FIXMessage in the list on the heap. With true it would ignore these fields thus limit storage on heap

Valid values: "true" | "false".

> **Important**: this value has priority over all other validation settings: Even If (*MessageMustBeValidated* = true and (*ProhibitUnknownTags* = true or *VerifyTagsValues*= true)) unknown fields are ignored.
>
> **Warning**: Changing this value will impact upon the performance of FIXEdge.
>
> **Note**: Ignored if *MessageMustBeValidated* = false

Default value: IgnoreUnknownFields = **false**

Validation.CheckRequiredGroupFields

Optional. Controls the validation of required fields in repeating group.

Valid values: "true" | "false".

If set to ‘true’ then repeating groups will be checked for presence of required fields. If set to ‘false’ then the responsibility for repeating group validity rests with the counterparty. The recommended setting is 'true'.

> **Note**: Ignored if *MessageMustBeValidated* = false

Default value: Validation.CheckRequiredGroupFields = **true**

AllowEmptyFieldValue

Mandatory. Defines whether the raw message may contain tags without values.

Valid values: "true" | "false".

If set to ‘true’, the raw message may contain tags without values that will be ignored.Otherwise an exception will be fired.

Default value: AllowEmptyFieldValue = **false**

ProhibitDuplicatedTags

Optional. This parameter is an option whereby incoming messages with duplicated tags can be processed. It's used for incoming messages.

If set to **“true”** incoming messages with duplicated tags will be rejected with the following message:

`"Tag appears more than once [RefSeqNum: _, RefTagID: _, RefMsgType: _]".`

If set to **“false”** raw message may contain duplicated field definitions. The **default** value is **“true”**.

In *FixEdge.properties* file, the option can be defined in the following way:

```
FixLayer.FixEngine.Session.<SessionID>.Validation.ProhibitDuplicatedTags = true
```

where SessionID is one of sessions defined in FixLayer.FixEngine.Sessions section.

Settings of Unregistered Acceptors

The following properties describe the behavior of unregistered acceptors:

UnregisteredAcceptor.CreateSession

Mandatory. Defines whether the FIXEdge works with Unregistered Acceptors.

Valid values: "true" | "false"

Default value: UnregisteredAcceptor.CreateSession = **false**

UnregisteredAcceptor.SessionStorageType

Optional. Commented. Defines the storage type for unregistered sessions.

Valid values: "persistent" | "persistentMM" | "splitPersistent" | "transient" | "null"

Default value: UnregisteredAcceptor.SessionStorageType = **persistent**

UnregisteredAcceptor.IgnoreSeqNumTooLowAtLogon

Optional. Commented. Defines how the FIXEdge should resolve '*Sequence number is too low*' problem at a Logon message.

Valid values: "true" | "false". When it set to 'true' then the session ignores error and continues to work with received sequence number.

Default value: UnregisteredAcceptor.IgnoreSeqNumTooLowAtLogon = **false**

UnregisteredAcceptor.RejectMessageWhileNoConnection

Optional.

UnregisteredAcceptor.tcpBufferDisabled

Optional.

UnregisteredAcceptor.maxMessagesAmountInBunch

Optional.

Other Settings

EncryptionConfigFile

Optional. Defines path to encryption config file.

Default value: EncryptionConfigFile=**encryption.properties**

ThirdPartyRoutingIsEnabled

Mandatory. Defines whether the FIXEdge supports the "Deliver To On Behalf Of" mechanism as specified by the FIX protocol.

Valid values: "true" | "false".

If set to 'true' then the FIXEdge will redirect FIX messages automatically to other FIX sessions that it maintains when *OnBehalfOfCompID* (115) tag in the incoming message is defined.

If set to 'false' then the FIXEdge directs all received messages in accordance with business rules.

Default value: ThirdPartyRoutingIsEnabled = **false**

DelayedProcessing.MaxDeliveryTries

Optional. Defines the number of attempts that will be made to deliver an application level message to the registered client application. If this value is exceeded then the session will be closed with the logout reason "Application not available".

Valid values: The value must be integer and not negative. The recommended value is 10.

Default value: DelayedProcessing.MaxDeliveryTries = **2**

DelayedProcessing.DeliveryTriesInterval

Optional. Defines the time interval (in milliseconds) between attempts to deliver an application level message to a registered client application in the event the application does not confirm receipt and operation upon the message.

Valid values: The value must be integer and greater than 0.

> **Note**: This parameter is required only if the *DelayedProcessing.MaxDeliveryTries* property is specified.

Default value: DelayedProcessing.DeliveryTriesInterval = **500**

MessageTimeToLive

Optional. Defines the time period (in milliseconds) after which a message rejecting is starting while session doesn't exist.

Valid values: The value must be integer and greater than 0.

Default value: MessageTimeToLive = **500**

OutgoingMessagesStorageSize

Mandatory. Defines the upper limit to the number of outgoing messages that are resent in the event of a Resend Request message.

Valid values: The value must be integer and greater than 0. The value "-1" means that the number of the messages is unlimited. The recommended value is 1000 if there is no data on mean.

Default value: OutgoingMessagesStorageSize = **1000**

ResendMessagesBlockSize

Mandatory. Defines how many missed messages can be requested in the one Resend Request message. If number of missed messages exceeds the value then several Resend Request messages will be sent.

Valid values: The value must be integer and not less than 0. The value “0” means that all required messages can be requested in one Resend Request message.

Default value: ResendMessagesBlockSize = **1000**

CheckVersionOfOutgoingMessages

Mandatory. Defines whether the FIXEdge must validate the the version of FIX protocol (*BeginString* (8) tag) used for the outgoing message against the version that is established for session.

Valid values: "true" | "false"

If set to ‘true’ then the application must use the same version of the protocol as the established session otherwise an error occurs. If set to ‘false’ then the application level message will be sent to the counterparty. The recommended value is "true".

Default value: CheckVersionOfOutgoingMessages = **false**

DictionariesFilesList

Mandatory: Defines a semicolon (or comma) separated list of XML files with FIX protocol definitions. In additional, this parameter could containlistof XML files with additional protocol customization, if required.

> **Note**: FIX protocol customization files **must** be specified **after** protocol definition files.

Example: DictionariesFilesList = FixEdge1/conf/fixdic44.xml;FixEdge1/conf/fixdic50.xml;FixEdge1/conf/fixdic50sp1.xml;FixEdge1/conf/fixdic50sp2.xml; FixEdge1/conf/fixdict11.xml;additional.xml

CustomRawDataTagStrategies

Optional: Array of Length* fields interpretation strategies.

Attention: this option is introduced as workaround for exchanges, which doesn't follow FIX standard and pass length in symbols, instead of length in bytes for Length* fields, if you don't interact with such exchange,youdont' need it

Example:

engine.properties

```
CustomRawDataTagStrategies.Count = 1
CustomRawDataTagStrategies.1.Tag = 213
CustomRawDataTagStrategies.1.Strategy = BY_CHARS
CustomRawDataTagStrategies.1.Encoding = UTF8
```

CustomRawDataTagStrategies.Count

Mandatory: Count of strategies,whichwilldefined

Example: CustomRawDataTagStrategies.Count = 5

Valid values: integer not less than 1.

CustomRawDataTagStrategies.<index>.Tag

Mandatory: Define raw data tag, which Length tagged field interpretation strategy is changed

Example: CustomRawDataTagStrategies.1.Tag = 213

Valid values: integer not less than 1.Iftag doesn't have Length field - this strategy will be ignored.

CustomRawDataTagStrategies.<index>.Strategy

Mandatory: Define how to interpret corresponding length field

Example: CustomRawDataTagStrategies.5.Strategy = BY_CHARS

Valid values: BY_CHARS - use count of characters | BY_BYTES - use count of bytes

CustomRawDataTagStrategies.<index>.Encoding

Mandatory: Defines which encoding should be used if to obtain raw data field value size in bytes if BY_CHARS strategy is used

Example: CustomRawDataTagStrategies.3.Encoding = UTF8

Valid values: UTF8 | ASCII

Monitoring.ListenPort

Optional. Allows to define port which will be added to set of engine listen ports and administrative sessions will be accepted on this port only. At the same time common session acceptance is permitted on the port.

Note: if this port is configured FIXICC agent (**2.7.23.3 version only**) will use it to establish the administrative session.

Example:

engine.properties

```
# The engine TCP listen port that is enabled for administrative sessions
# The parameter is not required.
# If port is configured it will be added to set of engine listen ports and administrative sessions will be
accepted on this port only
# If not exists or empty common engine listen ports are used to accept administrative sessions
Monitoring.ListenPort = 8901
```


Hint: if you don't know what FIXICC agent version is installed in your system, you can put Monitoring.ListenPort to list of listen ports at the first place. Therefore FE behavior will be the same as in example regardless of FIXICC agent version.

Hint example:

```
ListenPort = 8900, 8901
Monitoring.ListenPort = 8900
```

Settings of Registered FIX Sessions

This part describes set of properties related to each registered FIX session ('*FIXEdge.properties*' configuration file). Each property's name contains within itself the name of FIX session. The format is: FixLayer.FixEngine.Session.<**Session_Name**>.<ParameterName>.

Properties for Acceptors and Initiators

FixLayer.FixEngine.Session.Session_Name.Description

Optional. Contains the session description.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.Description = Just a FIX session for testing.

FixLayer.FixEngine.Session.Session_Name.Version

Mandatory. Defines version of FIX protocol or custom protocol of the registered session

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.Version = FIX44

FixLayer.FixEngine.Session.TestFIXAcceptor.Version = FIXT11:FIX50SP2

FixLayer.FixEngine.Session.Session_Name.Role

Mandatory. Defines a role of the registered session

Valid values: "Acceptor" | "Initiator"

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.Role = Acceptor

FixLayer.FixEngine.Session.Session_Name.SenderCompID

Mandatory Key parameter for purposes of session identification. Sets value of *SenderCompID* (49) tag in outgoing FIX messages.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.SenderCompID = FIXEDGE

FixLayer.FixEngine.Session.Session_Name.TargetCompID

Mandatory. Key parameter for purposes of session identification. Sets value of *TargetCompID* (56) tag in outgoing FIX messages.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.TargetCompID = FIXCLIENT

FixLayer.FixEngine.Session.Session_Name.SessionQualifierValue

Optional. Allows to set the name of SessionQualifier.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.SessionQualifierValue = Q1

FixLayer.FixEngine.Session.Session_Name.StorageType

Mandatory. Defines the session storage type.

Valid values: "persistent" | "persistentMM" | "splitPersistent" | "transient" | "null"

Default value: FixLayer.FixEngine.Session.TestFIXAcceptor.StorageType = **persistentMM**

FixLayer.FixEngine.Session.Session_Name.Username

Optional. Sets value of *Username* (553) tag in Logon message

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.Username = user

FixLayer.FixEngine.Session.Session_Name.Password

Optional. Sets value of *Password* (554) tag in Logon message

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.Password = foobar

FixLayer.FixEngine.Session.Session_Name.UsernameTag

Optional. This parameter of Username field sets tag value of Username in Logon message.

Example: FixLayer.FixEngine.Session.Session_Name.UsernameTag=10533 – sends Username in a custom tag 10533

Default value: FixLayer.FixEngine.Session.Session_Name.UsernameTag=553

FixLayer.FixEngine.Session.Session_Name.PasswordTag

Optional. This parameter of Password field sets tag value of Password in Logon message.

Example: FixLayer.FixEngine.Session.Session_Name.PasswordTag=10554 – sends Password in a custom tag 10544

Default value: FixLayer.FixEngine.Session.Session_Name.PasswordTag=554

FixLayer.FixEngine.Session.Session_Name.SenderSubID

Optional. Sets value of *SenderSubID* (50) tag in outgoing FIX messages

FixLayer.FixEngine.Session.Session_Name.TargetSubID

Optional. Sets value of *TargetSubID* (57) tag in outgoing FIX messages

FixLayer.FixEngine.Session.Session_Name.SenderLocationID

Optional. Sets value of *SenderLocationID* (142) tag in outgoing FIX messages

FixLayer.FixEngine.Session.Session_Name.TargetLocationID

Optional. Sets value of *TargetLocationID* (143) tag in outgoing FIX messages

FixLayer.FixEngine.Session.Session_Name.EncryptMethod

Mandatory. Defines method of encryption for registered session. Sets value of *EncryptMethod* (98) tag in Logon message.

Valid values:

- 0 - None / other
- 1 - PKCS (proprietary)
- 2 - DES (ECB mode)
- 3 - PKCS/DES (proprietary)
- 4 - PGP/DES (defunct)
- 5 - PGP/DES-MD5 (see app note on FIX web site)
- 6 - PEM/DES-MD5 (see app note on FIX web site)

Default value: FixLayer.FixEngine.Session.TestFIXAcceptor.EncryptMethod = **0**

FixLayer.FixEngine.Session.Session_Name.InSeqNum

Optional. Defines the initial incoming sequence number. It is expected that the first incoming message will have the specified sequence number.

Valid values: 0, 1, 2, ... Value "0" means that initial sequence number = 1.

Default value: FixLayer.FixEngine.Session.TestFIXAcceptor.InSeqNum = **0**

FixLayer.FixEngine.Session.Session_Name.OutSeqNum

Optional. Defines the initial outgoing sequence number. The first outgoing message will be sent with the specified sequence number.

Valid values: 0, 1, 2, ... Value "0" means that initial sequence number = 1.

Default value: FixLayer.FixEngine.Session.TestFIXAcceptor.OutSeqNum = **0**

FixLayer.FixEngine.Session.Session_Name.RecreateOnLogout

Mandatory. Defines whether the session must be recreated on the logout.

Valid values: "true" | "false"

If set to 'false' then the session is removed from the list of sessions after successful disconnection.

If set to 'true' then the session will be recreated after disconnection, it means:

- the Acceptor will wait for the Logon message from the counterparty
- the Initiator will send the Logon message to the counterparty

Note : Recreation will take place *only* if disconnection is initialized by the counterparty by the logout message.

Default value: FixLayer.FixEngine.Session.TestFIXAcceptor.RecreateOnLogout = **false**

FixLayer.FixEngine.Session.Session_Name.TerminateOnLogout

Optional.

Valid values: "true" | "false"

FixLayer.FixEngine.Session.Session_Name.IntradayLogoutTolerance

Optional. Defines using of the Intraday Logout Tolerance mode in particular registered session.

Valid values: "true" | "false"

This property overrides the *IntradayLogoutTolerance* property in configuration file '*engine.properties*' for the particular session.

Note : If a new session is created via FIXICC then the *default value* is taken from the *IntradayLogoutTolerance* property.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.IntradayLogoutTolerance = true

FixLayer.FixEngine.Session.Session_Name.ForceSeqNumReset

Optional. Defines using of the Force Sequence Number Reset mode for particular session.

Valid values: "0" or "false" | "1" or "true" | "2". If property is set in another value - the value from property *ForceSeqNumReset* in configuration file *'engine.properties'* is used.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.ForceSeqNumReset = **2**

FixLayer.FixEngine.Session.Session_Name.HandleSeqNumAtLogon

Optional.

Valid values: "true" | "false"

FixLayer.FixEngine.Session.Session_Name.SocketPriority

Optional. Defines priority of the socket Send \ Receive operations.

Valid values:

- EVEN - Share worker thread among all session in the Engine
- AGGRESSIVE_SEND - Use dedicated thread to send outgoing messages
- AGGRESSIVE_RECEIVE - Use dedicated thread to receive incoming messages
- AGGRESSIVE_SEND_AND_RECEIVE - Enables the both aggressive sending and aggressive receiving options
- DIRECT_SEND - use the current thread for sending, if this would block, performs as "EVEN".

Note: If parameter not specified for session then '*SocketOpPriority*' engine parameters will be used instead.

Default value: FixLayer.FixEngine.Session.TestFIXAcceptor.SocketPriority = **EVEN**

FixLayer.FixEngine.Session.Session_Name.AggressiveReceiveDelay

Optional. Defines a polling interval in microseconds for reading data from the socket.

Works only with aggressive sessions i.e. Engine::AGGRESSIVE_RECEIVE_SOCKET_OP_PRIORITY or Engine::AGGRESSIVE_SEND_AND_RECEIVE_SOCKET_OP_PRIORITY modes.

Setting lower value reduces the latency but causes high CPU utilization.

Default value: 500 microseconds.

FixLayer.FixEngine.Session.Session_Name.TcpBufferDisabled

Optional.

Valid values: "true" | "false"

FixLayer.FixEngine.Session.Session_Name.RejectMessageWhileNoConnection

Optional.

Valid values: "true" | "false"

FixLayer.FixEngine.Session.Session_Name.ForceReconnect

Optional.

Session will be reconnected in the following situations:

1) In WaitForConfirmLogon state when session received not a confirming logon

2) In WaitForConfirmLogon state when logon frame expired

3) In Reconnect state when session was closed

4) In Initial state of session-initiator, when there were no acceptor exists. In this case, this property works with the 'Reconnect.MaxTries' property in 'engine.properties' for registered session:

a) if FixLayer.FixEngine.Session.Session_Name.ForceReconnect = false, the parameter Reconnect.MaxTries will be ignored, Utils::Exception will be thrown

b) If FixLayer.FixEngine.Session.Session_Name.ForceReconnect = true, the parameter Reconnect.MaxTries will be used

Valid values: "true" | "false"

Default value: FixLayer.FixEngine.Session.FIXAcceptor.ForceReconnect = **false**

FixLayer.FixEngine.Session.Session_Name.DefaultApplicationProtocol

Optional.

FixLayer.FixEngine.Session.Session_Name.PredefinedMessages

Optional.

FixLayer.FixEngine.Session.Session_Name.PredefinedMessage

Optional.

FixLayer.FixEngine.Session.Session_Name.ActiveConnection

Optional.

FixLayer.FixEngine.Session.Session_Name.KeepState

Optional.

FixLayer.FixEngine.Session.Session_Name.EnableAutoSwitchToBackupConnection

Optional. If it is set as "true" an active session can automatically reconnect to its backup session (after ReconnectMaxTries attempts to connect) in case there is a problem to connect to host/port of the active one. In order to maintain the proper sequence number between two sessions HandleSeqNumAtLogon should be defined as "true".

Valid values: "true" | "false"

Default value: FixLayer.FixEngine.Session.Session_Name.EnableAutoSwitchToBackupConnection= **false**

FixLayer.FixEngine.Session.Session_Name.EnableCyclicSwitchBackupConnection

Optional. If it is set as "true" a backup session can automatically switch back to the active session (after ReconnectMaxTries attempts to connect) once the backup connection has problems.

Valid values: "true" | "false"

Default value: FixLayer.FixEngine.Session.Session_Name.EnableCyclicSwitchBackupConnection = **false**

FixLayer.FixEngine.Session.Session_Name.MaxMessagesAmountInBunch

Optional.

Default value: FixLayer.FixEngine.Session.TestFIXAceptor.MaxMessagesAmountInBunch = **0**

FixLayer.FixEngine.Session.Session_Name.SecurityGroups

Optional.

FixLayer.FixEngine.Session.Session_Name.Protocol

Optional. Defines underlying protocol of FIX session.

Valid values: "FIX_TCP" | " FIXT11_TCP" | "FIXT11_FAST_TCP"

Default value: FixLayer.FixEngine.Session.TestFIXAceptor.Protocol = FIX_TCP - if session Version is FIX4x

FixLayer.FixEngine.Session.TestFIXAceptor.Protocol = FIXT11_TCP - if session Version is FIX5x

FixLayer.FixEngine.Session.Session_Name.CustomRawDataTagStrategies

Optional: Array of references to CustomRawDataTagStrategies array entries

Example:

```
FixLayer.FixEngine.Session.Session1.CustomRawDataTagStrategies.Count = 2
FixLayer.FixEngine.Session.Session1.CustomRawDataTagStrategies.1.Strategy = 2
FixLayer.FixEngine.Session.Session1.CustomRawDataTagStrategies.2.Strategy = 5
```

FixLayer.FixEngine.Session.Session_Name.CustomRawDataTagStrategies.Count

Mandatory: Count of references to strategies, used by session

Valid values: integer not less than 1.

Example: FixLayer.FixEngine.Session.Session1.CustomRawDataTagStrategies.Count = 2

FixLayer.FixEngine.Session.Session_Name.CustomRawDataTagStrategies.<index>.Strategy

Mandatory: Reference to CustomRawDataTagStrategies array entry

Valid values: integer not less than 1.

Example: FixLayer.FixEngine.Session.Session1.CustomRawDataTagStrategies.3.Strategy = 2

FixLayer.FixEngine.Session.Session_Name.IncomingMessagesLimit

Optional. Define a limit for messages received by session. Both the session-level messages and application level messages are counted.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.IncomingMessagesLimit = 1000

Default value for parameter is '0' - unlimited, no events will be created on business level in such case.

For more information please read the article Overload protection in FIXEdge.

FixLayer.FixEngine.Session.Session_Name.IncomingThroughputLimit

Optional. Define a limit for incoming messages throughput.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.IncomingThroughputLimit = 25

Default value for parameter is '0' - unlimited, no events will be created on business level in such case.

For more information please read the article Overload protection in FIXEdge.

FixLayer.FixEngine.Session.Session_Name.OutgoingQueueSize

Optional. Define a limit for outgoing queue.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.OutgoingQueueSize = 10.

Default value for parameter is '0' - unlimited, no events will be created on the business level in such case.

For more information please read the article Overload protection in FIXEdge.

FixLayer.FixEngine.Session.Session_Name.StorageRecoveryStrategy

Define a recovery storage strategy in case of broken storage.

Valid values:

- NONE - exception on error in FA, don't start session in FixEdge
- CREATE_NEW_ON_ERROR - create new storage on any error

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.StorageRecoveryStrategy = CREATE_NEW_ON_ERROR

Default value for parameter is 'NONE'.

FixLayer.FixEngine.Session.Session_Name.HiddenLogonCredentials

Optional. Provides an option to mask password in in/out log files.

Valid values: "true" | "false".

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.HiddenLogonCredentials = true.

FixLayer.FixEngine.Session.Session_Name.MaskedTags

Optional. Provides an option to mask any tag value in the log files.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.MaskedTags = 554,553.

The message will be written to the log file in the following way: 8=FIX.4.4^9=85^35=A^49=a_t1^56=a_s1^34=1^52=20161005-17:29:14.339^98=0^108=10^141=Y^553=***^554=***^10=176^

FixLayer.FixEngine.Session.Session_Name.SendingTimestampUnit

⚠ Parameter is available since **FIXEdge 6.2.0**.

Optional. Defines the precision of the timestamp to be specified in 52 tag (SendingTime) and 122 tag (OrigSendingTime) in outgoing message.

Valid values: second | millisecond | microsecond | nanosecond

Defaults:

- For FIX 4.0 FIX 4.1:
  - if KeepMillisecondsInUTCTimeFields = 0, 52 tag (SendingTime) and 122 tag (OrigSendingTime) will have seconds precision
  - if KeepMillisecondsInUTCTimeFields = 1, 52 tag (SendingTime) will have seconds precision and 122 tag (OrigSendingTime) will have milliseconds precision
  - if KeepMillisecondsInUTCTimeFields = 2, 52 tag (SendingTime) and 122 tag (OrigSendingTime) will have seconds precision
- Millisecond for all other FIX versions

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.SendingTimestampUnit = nanosecond

FixLayer.FixEngine.Session.Session_Name.ResendMessagesLimit

Optional. Defines the maximum volume of messages sent as a reply on resend request.

Valid values:

- "-1" - disabled
- "0" - do not resend real messages, reply with GapFill
- positive number - max number of messages.

Default value: FixLayer.FixEngine.Session.Session_Name.ResendMessagesLimit = **-1**

Important: if **ResendMessagesLimit** property for the FIX session is set to "0" or to any positive number, then another mechanism for **ignoring duplicate resend requests** is automatically enabled.

For more information please read the article How to handle Resend Requests.

FixLayer.FixEngine.Session.Session_Name.CustomSessionType

Optional. Specifies whether a session should encrypt password each time.

Valid values:

- GENERIC - generic session
- LME_SELECT- encrypt password tag each time
- TOTAL_SSN_TYPES - represent number of sessions types
- CME_SECURE_LOGON - use CME secure logon scheme

Default value: **GENERIC**

FixLayer.FixEngine.Session.Session_Name.CMESecureKeysFile

Optional. Specifies a path to CME keys files used for CME secure logon

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.Version = CMEKeys.txt

FixLayer.FixEngine.Session.Session_Name.sendLastMsgSeqNumProcessed

Optional. Allows automatic adding the LastMsgSeqNumProcessed(369) tag to outgoing messages.

Valid values: "true" | "false"

Default value: **false**

The group of settings for FAST sessions

FixLayer.FixEngine.Session.Session_Name.FastTemplateFn

Mandatory for FAST session (Protocol = FIXT11_FAST_TCP). Defines path to file with FAST templates.

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.FastTemplateFn = FixEdge1/conf/fixt11-fix50sp2-templates.xml

FixLayer.FixEngine.Session.Session_Name.EnableFastScp

Optional. Enables or disables sending of FAST Hello message before Logon.

Default value: FixLayer.FixEngine.Session.TestFIXAcceptor.EnableFastScp = true

FixLayer.FixEngine.Session.Session_Name.FASTCodecParameters.BoolNtoString

Optional. Defines string representation of FIX boolean 'false' value in FAST encoded message

Default value: Character with ASCII code 0

FixLayer.FixEngine.Session.Session_Name.FASTCodecParameters.BoolYtoString

Optional. Defines string representation of FIX boolean 'true' value in FAST encoded message

Default value: Character with ASCII code 1

Additional properties for Acceptors only

FixLayer.FixEngine.Session.Session_Name.SourceIPaddress

Optional. Defines the expected value of the source IP address. If the real value is not equal to the expected one then the session is disconnected without sending a message and an error condition is generated in the log output.

> For more details read the following article: How to specify multiple IP addresses per FIX Session

FixLayer.FixEngine.Session.Session_Name.LogonMessageSessionQualifierTag

Optional. Allows to set a tag into the logon message in which the SessionQualifier will be sent. Even custom tags can be used but it's recommended to use standard tags for this purpose, e.g. 553 (Username) or 50 (SenderSubID).

Example: FixLayer.FixEngine.Session.TestFIXAcceptor.LogonMessageSessionQualifierTag = 553

Additional properties for Initiators only

FixLayer.FixEngine.Session.Session_Name.IgnoreSeqNumTooLowAtLogon

Optional. Defines how the FIXEdge should resolve '*Sequence number is too low*' problem at a logon message.

Valid values: "true" | "false". When it set to 'true' then the session ignores error and continues to work with received sequence number.

Default value: FixLayer.FixEngine.Session.TestFIXAcceptor.IgnoreSeqNumTooLowAtLogon = **false**

FixLayer.FixEngine.Session.Session_Name.CustomLogonFileName

Optional. Defines a path to the custom logon file. It is possible to customize the Logon message to be sent by the initiator during connection process.

Note: the path is related to the 'FIXEdge.RootDir'.

Example: FixLayer.FixEngine.Session.TestFIXInitiator.CustomLogonFileName = logon.msg

FixLayer.FixEngine.Session.Session_Name.Host

Mandatory. Defines a network IP address of the computer, to which connection is established.

Default value: FixLayer.FixEngine.Session.TestFIXInitiator.Host = **localhost**

FixLayer.FixEngine.Session.Session_Name.Port

Mandatory. Defines a port's network number on the computer, to which connection is established.

Default value: FixLayer.FixEngine.Session.TestFIXInitiator.Port = **9106**

FixLayer.FixEngine.Session.Session_Name.HBI

Mandatory. Defines the heartbeat interval (seconds) – a time interval between Heartbeat messages.

The recommended value is 10 seconds for dedicated connections or private networks. Trading connections via the internet will require calibration. '0' means that no Heartbeat messages will be sent.

Default value: FixLayer.FixEngine.Session.TestFIXInitiator.HBI = **60**

FixLayer.FixEngine.Session.Session_Name.ReconnectMaxTries

Optional. Defines the number of attempts to restore the session if telecommunication error is detected *during one connection attempt*. "-1" means unlimited number of attempts.

Default value: FixLayer.FixEngine.Session.TestFIXInitiator.ReconnectMaxTries = 3 (defined in engine.properties)

FixLayer.FixEngine.Session.Session_Name.ReconnectInterval

Optional. Defines the time interval in milliseconds between reconnection attempts. The value must be greater than 0 if specified.

Default value: FixLayer.FixEngine.Session.TestFIXInitiator.ReconnectInterval = 1000 (defined in engine.properties)

FixLayer.FixEngine.Session.Session_Name.ReconnectGroup

Optional. Defines the name of reconnection group. All sessions in one reconnection group will be connected one by one with ReconnectInterval delay.

Note: disable ReconnectMaxTries (=0) to avoid simultaneous connecting of sessions on FIXAntenna level

Additional properties for SSL configuration

This section describes set of parameters to be added for FIX session (initiator or acceptor) to configure secured SSL connection.
Configuration is applicable to FIXEdge 5.9.x and higher
Note: self-signed SSL certificate is required for this configuration and have to be created separately.

For full configuration instructions refer to here: How to configure SSL connection

SSL parameters specified in FIXEdge.properties have a higher priority than similar parameters specified in engine.properties

FixLayer.FixEngine.Session.Session_Name.SSL

Optional. Turns ON/OFF secure connection. Applicable for initiator and acceptor sessions. Valid values: true, false (default).

Example: FixLayer.FixEngine.Session.TestFIXSession.SSL = true

FixLayer.FixEngine.Session.Session_Name.SSLProtocols

Mandatory, if FixLayer.FixEngine.Session.Session_Name.SSL = true. Contains a comma separated list of supported SSL protocol versions. Applicable for initiator session. Valid values: SSLv2, SSLv3, TLSv1, TLSv1_1, TLSv1_2.

Example: FixLayer.FixEngine.Session.TestFIXSession.SSLProtocols = TLSv1_2

FixLayer.FixEngine.Session.Session_Name.SSLCheckPrivateKey

Optional. Specifies whether private key should be checked. Applicable for initiator session. Valid values: true (default if SSL support is turned on), false.

*Example:*FixLayer.FixEngine.Session.TestFIXSession.SSLCheckPrivateKey = false

Deprecated since FIXEdge 6.8.0. Correspondence between the key and certificate will be checked automatically. FIXEdge does not establish session if the key doesn't correspond to the certificate.

FixLayer.FixEngine.Session.Session_Name.SSLValidatePeerCertificate

Optional. If set to "true" then client connections without a certificate are rejected. SSLCACertificate parameter must be specified. Applicable for initiator session. Valid values: true, false (default).

*Example:*FixLayer.FixEngine.Session.TestFIXSession.SSLValidatePeerCertificate = true

FixLayer.FixEngine.Session.Session_Name.SSLCertificate

Optional, if FixLayer.FixEngine.Session.Session_Name.SSL = true. Path to SSL certificate. Applicable for initiator session.

Example: FixLayer.FixEngine.Session.TestFIXSession.SSLCertificate = FIXEdge/FixEdge1/conf/cert.pem

FixLayer.FixEngine.Session.Session_Name.SSLCACertificate

Optional. This is a path to the file containing CA certificates (all in one file on after another, PEM format only). Applicable for initiator session.

Example: FixLayer.FixEngine.Session.TestFIXSession.SSLCACertificate = FIXEdge/FixEdge1/conf/cert.pem

Introduced in FIXEdge 6.8.0

FixLayer.FixEngine.Session.Session_Name.SSLCertificatePassword

Optional. Certificate's password. Applicable for initiator session.

Example: FixLayer.FixEngine.Session.TestFIXSession.SSLCertificatePassword = CertificatePassword

Introduced in FIXEdge 6.8.0

FixLayer.FixEngine.Session.Session_Name.SSLPrivateKey

Optional. Path to SSL private key. Applicable for initiator session. If it is omitted Engine tries to load private key from the same file as SSLCertificate parameter states.

Example: FixLayer.FixEngine.Session.TestFIXSession.SSLPrivateKey = FIXEdge/FixEdge1/conf/cert.pem

FixLayer.FixEngine.Session.Session_Name.SSLPrivateKeyPassword

Optional. Private key's password. Applicable for initiator session.

Example: FixLayer.FixEngine.Session.TestFIXSession.SSLPrivateKeyPassword = CertificateKeyPassword

Introduced in FIXEdge 6.8.0

FixLayer.FixEngine.Session.Session_Name.SSLCiphersList

Optional. The default OpenSSL built-in configuration is used in case of the parameter absence.

The list is passed as is to OpenSSL during engine initialization in case of acceptor sessions (acceptor sessions are sharing the same configuration) and during session initialization in case of initiator sessions. See a list of the supported ciphers and list format is at official OpenSSL site: https://www.openssl.org/docs/man1.0.2/apps/ciphers.html

Introduced in FIXEdge 6.8.0

Session Schedule Settings

Default Session Schedule Properties

This section describes default session schedule properties which are used for session schedule management functionality.

All the properties from this section will be ignored if Cron-Like Session Schedule Properties are used.

Since FIXEdge 6.0 all the properties from this section can be defined in both CRON format and standard ***HH:MM*** format. See Cron-Like Session Schedule Properties for CRON syntax details.

FixLayer.FixEngine.Sessions.DefaultStartTime

Optional. Defines local time to start FIX sessions (HH:MM). Used when a registered session doesn't have StartTime parameter. If the FIXEdge start-up time is greater than the specified value then the sessions will be started immediately.

E.g.: FixLayer.FixEngine.Sessions.DefaultStartTime = **08:00**

FixLayer.FixEngine.Sessions.DefaultTerminateTime

Optional. Defines local time to terminate FIX sessions (HH:MM). Used when registered session doesn't have TerminateTime parameter.

E.g.: FixLayer.FixEngine.Sessions.DefaultTerminateTime = **23:59**

Old-Style Session Schedule Properties

This section describes session schedule properties which are used for session schedule management functionality.

All the properties from this section will be ignored if Cron-Like Session Schedule Properties are used.

Since FIXEdge 6.0 all the properties from this section can be defined in both CRON format and standard ***HH:MM*** format. See Cron-Like Session Schedule Properties for CRON syntax details.

FixLayer.FixEngine.Session.Session_Name.StartTime

Optional. Defines the local time (HH:MM) when the session is registered by FIXEdge. Session can be used for routing purposes (via BL rules) but it doesn't make the session available for connection.

If FIXEdge's start-up time is greater than the specified value then the session will be started immediately. See also default property *DefaultStartTime* to know when a default value is applied.

E.g.: FixLayer.FixEngine.Session.TestFIXAcceptor.StartTime = **08:00**

FixLayer.FixEngine.Session.Session_Name.ConnectTime

Optional. Defines the local time (HH:MM) when FIXEdge determines the registered session as ready for connection.

StartTime vs ConnectTime

E.g. if StartTime = 08:00, ConnectTime = 09:00, then Session will be registered at 08:00, but will be not be ready for connection until 09:00.

Due to the fact that parameters are optional, below are some if-then scenarios:

- If ConnectTime parameter is not specified and StartTime is specified, then ConnectTime = StartTime. In other words, on the StartTime the session will be registered and then will be immediately ready for connection;
- If ConnectTime parameter is specified and StartTime is not specified, then session becomes registered right after FIXEdge start, but is ready for connection only after ConnectTime comes;
- If both ConnectTime and StartTime are not specified, then session becomes registered and ready for connection right after FIXEdge start.

E.g.: FixLayer.FixEngine.Session.TestFIXAcceptor.ConnectTime = **09:00**

FixLayer.FixEngine.Session.Session_Name.DisconnectTime

Optional. Defines the local time (HH:MM) when FIXEdge sends logout message. In opposite to *TerminateTime*, when DisconnectTime comes, session remains registered and logs are not archived.

E.g.: FixLayer.FixEngine.Session.TestFIXAcceptor.DisconnectTime = **23:58**

FixLayer.FixEngine.Session.Session_Name.TerminateTime

Optional. Defines the local time (HH:MM) when FIXEdge sends logout message, removes the session from the list of registered sessions and archives logs, so that after the session restore SeqNums will be reset.

TerminateTime should be 1-2 minutes earlier than FIXEdge shutdown to ensure that all logs moved correctly.

If FIXEdge's start-up time is greater than the specified value then the session will not be started. See also default property *DefaultTerminateTime* to know when default value is applied.

E.g.: FixLayer.FixEngine.Session.TestFIXAcceptor.TerminateTime = **23:59**

Cron-Like Session Schedule Properties

The section below describes new session schedule management functionality that was introduced in ***FIXEdge 6.0***. See How to upgrade Session Schedule to Cron-Like format article for upgrade and functionality details.

All the properties mentioned in this section are available since ***FIXEdge 6.0.***

CRON expressions which should be assigned to schedule properties have the following syntax:

CRON syntax

```
 second (0 - 59)
  minute (0 - 59)
|   hour (0 - 23)
|    day of month (1 - 31)
|     month (1 - 12)
|      day of week (1 - 7: Sunday to Saturday)
|


* * * * * *
```

Support of the CRON expressions is limited by quartz. In particular, specifying both a day-of-week and a day-of-month values is prohibited (you'll need to use the '*' character in one of these fields).

The only note here is that '*' character is used instead of '?' character in compare with quartz.

Several CRON expressions can be assigned for each property, they should be delimited by semicolon.

FixLayer.FixEngine.Session.Session_Name.Schedule

Optional. Defines a schedule for the session. This property can be assigned to each registered FIX session and it should contain only one value of the schedule name to be used.

Pay your attention to the following rules of new session schedule management functionality usage:

- If FixLayer.FixEngine.Session.Session_Name.Schedule is specified then the default and old-style session schedule's properties will be ignored.
- If FixLayer.FixEngine.Session.Session_Name.Schedule is not specified, old-style and default session schedule properties will be applied.

Example: FixLayer.FixEngine.Session.Session_Name.Schedule = TestFIXSchedule

Schedules.Schedule_Name.StartTime

Conditionally required. It must be specified if FixLayer.FixEngine.Session.Session_Name.Schedule is defined and none of Schedules.Schedule_Name.ConnectTime, Schedules.Schedule_Name.DisconnectTime, FixLayer.FixEngine.Session.Session_Name.TerminateTime properties is specified.

The meaning of the property is the same as for old-style StartTime property.

Valid values: The property should be configured using CRON expression.

Example: Schedules.TestFIXSchedule.StartTime = 0 0 8 * * 2-6

Schedules.Schedule_Name.ConnectTime

Conditionally required. It must be specified if FixLayer.FixEngine.Session.Session_Name.Schedule is defined and none of Schedules.Schedule_Name.StartTime, Schedules.Schedule_Name.DisconnectTime, FixLayer.FixEngine.Session.Session_Name.TerminateTime properties is specified.

The meaning of the property is the same as for old-style ConnectTime property.

Valid values: The property should be configured using CRON expression.

Example: Schedules.TestFIXSchedule.ConnectTime = 0 0 9 * * 2-6

Schedules.Schedule_Name.DisconnectTime

Conditionally required. It must be specified if FixLayer.FixEngine.Session.Session_Name.Schedule is defined and none of Schedules.Schedule_Name.StartTime, Schedules.Schedule_Name.ConnectTime, FixLayer.FixEngine.Session.Session_Name.TerminateTime properties is specified.

The meaning of the property is the same as for old-style DisconnectTime property.

Valid values: The property should be configured using CRON expression.

Example: Schedules.TestFIXSchedule.DisconnectTime = 0 0 21 * * 2-6

Schedules.Schedule_Name.TerminateTime

Conditionally required. It must be specified if FixLayer.FixEngine.Session.Session_Name.Schedule is defined and none of Schedules.Schedule_Name.StartTime, Schedules.Schedule_Name.ConnectTime, Schedules.Schedule_Name.DisconnectTime properties is specified.

The meaning of the property is the same as for old-style TerminateTime property.

Valid values: The property should be configured using CRON expression.

Example: Schedules.TestFIXSchedule.TerminateTime = 0 0 21 * * 6

Schedules.Schedule_Name.DaysOff

Optional. Defines the date and/or time when a schedule should not be applied, i.e. should be ignored.

Valid values: The property should be configured using CRON expression.

Example: Schedules.TestFIXSchedule.DaysOff = * * * 25 12 *

It means that the schedule *TestFIXSchedule* will not be executed on the December 25th of each year. In other words, if session was active before as December 25th came, it would remain active on December 25th, and the further schedule will be applied only on December 26th.

Schedules.Schedule_Name.TimeZone

Optional. Defines the time zone of the schedule.

Valid values: UTC/Local or TimeZone name from standard JAVA timezone list that depends on the used version of JRE. You can find some part of the supported time zones on the Oracle page.

To get the complete list of time zones, execute the following command: *"java <path>/timelinegenerator.jar --tzlist"*

Example: Schedules.TestFIXSchedule.TimeZone = America/New_York

Default value: Schedules.TestFIXSchedule.TimeZone = Local

Examples of Usage

Use Case #1

Say, you have the following configuration setup in FIXEdge.properties:

```
_____________
FixLayer.FixEngine.Sessions.DefaultStartTime = 0 0 8 * * *
FixLayer.FixEngine.Sessions.DefaultTerminateTime = 0 59 23 * * *
_____________
FixLayer.FixEngine.Session.Session1.Version = FIX44
FixLayer.FixEngine.Session.Session1.Role = Acceptor
FixLayer.FixEngine.Session.Session1.SenderCompID = FIXEdge
FixLayer.FixEngine.Session.Session1.TargetCompID = SampleClient
_____________
```

Note: StartTime/ConnectTime/DisconnectTime/TerminateTime as well as Schedule are not specified for the session.

Result:

The defaults should be applied. Thus session will be started at 8:00 and will be terminated at 23:59 every day of the week (Monday - Sunday) in local time. Every day it will be started with empty log files and sequence numbers reset.

Use Case #2

Say, you have the following configuration setup in FIXEdge.properties:

```
_____________
FixLayer.FixEngine.Session.Session1.Version = FIX44
FixLayer.FixEngine.Session.Session1.Role = Acceptor
FixLayer.FixEngine.Session.Session1.SenderCompID = FIXEdge
FixLayer.FixEngine.Session.Session1.TargetCompID = SampleClient
_____________
FixLayer.FixEngine.Session.Session1.ConnectTime = 0 0 8 * * 2-6
FixLayer.FixEngine.Session.Session1.DisconnectTime = 0 0 21 * * 2-6
FixLayer.FixEngine.Session.Session1.TerminateTime = 0 0 21 * * 6
_____________
```

Note: StartTime, as well as Schedule, are not specified for the session. Default Session Schedule Properties are also not set.

Result:

The session will connect at 8:00 and disconnect at 21:00 every working day of the week (on Monday, Tuesday, Wednesday, Thursday and Friday) in local time. Every Monday it will start with empty log files and sequence numbers will be reset.

Use Case #3

Say, you have the following configuration setup in FIXEdge.properties:

```
_____________
FixLayer.FixEngine.Sessions.DefaultStartTime = 0 0 8 * * *
FixLayer.FixEngine.Sessions.DefaultTerminateTime = 0 59 23 * * *
_____________
FixLayer.FixEngine.Session.Session1.Version = FIX44
FixLayer.FixEngine.Session.Session1.Role = Acceptor
FixLayer.FixEngine.Session.Session1.SenderCompID = FIXEdge
FixLayer.FixEngine.Session.Session1.TargetCompID = SampleClient
_____________
FixLayer.FixEngine.Session.Session1.ConnectTime = 0 0 8 * * 2-6
FixLayer.FixEngine.Session.Session1.DisconnectTime = 0 0 21 * * 2-6
FixLayer.FixEngine.Session.Session1.TerminateTime = 0 0 21 * * 6
_____________
```

Note: StartTime as well as Schedule are not specified for the session.

Result:

Session will connect at 8:00 and disconnect at 21:00 every working day of the week (on Monday, Tuesday, Wednesday, Thursday and Friday) in local time. Every Monday it will start with empty log files and sequence numbers will be reset. Default properties will be ignored.

Use Case #4

Say, you have the following configuration setup in FIXEdge.properties:

```
_____________
Schedules.TestFIXSchedule.StartTime = 0 0 7 * * 2-6
Schedules.TestFIXSchedule.DisconnectTime =  0 0 23 * * 2-6
Schedules.TestFIXSchedule.TerminateTime =  0 0 23 * * 6
# catholic Christmas:
Schedules.TestFIXSchedule.DaysOff = * * * 25 12 *
# TimeZone
Schedules.TestFIXSchedule.TimeZone = EST
_____________
FixLayer.FixEngine.Session.Session1.Version = FIX44
FixLayer.FixEngine.Session.Session1.Role = Acceptor
FixLayer.FixEngine.Session.Session1.SenderCompID = FIXEdge
FixLayer.FixEngine.Session.Session1.TargetCompID = SampleClient
_____________
FixLayer.FixEngine.Session.Session1.Schedule =  TestFIXSchedule
_____________
```

Note: Old-style StartTime/ConnectTime/DisconnectTime/TerminateTime as well as Default Session Schedule Properties are not specified.

Result:

Session will connect at 7:00 and disconnect at 23:00 on Monday, Tuesday, Wednesday, Thursday and Friday in EST time. Every Monday it will start with empty log files and sequence numbers will be reset. The schedule will not be executed on the December 25th of each year.

Use Case #5

Say, you have the following configuration setup in FIXEdge.properties:

```
_____________
FixLayer.FixEngine.Sessions.DefaultStartTime = 0 0 9 * * *
FixLayer.FixEngine.Sessions.DefaultTerminateTime = 0 59 23 * * *
_____________
Schedules.TestFIXSchedule.StartTime = 0 0 7 * * 2-6
Schedules.TestFIXSchedule.DisconnectTime =  0 0 23 * * 2-6
Schedules.TestFIXSchedule.TerminateTime =  0 0 23 * * 6
# catholic Christmas:
Schedules.TestFIXSchedule.DaysOff = * * * 25 12 *
# TimeZone
Schedules.TestFIXSchedule.TimeZone = EST
_____________
FixLayer.FixEngine.Session.Session1.Version = FIX44
FixLayer.FixEngine.Session.Session1.Role = Acceptor
FixLayer.FixEngine.Session.Session1.SenderCompID = FIXEdge
FixLayer.FixEngine.Session.Session1.TargetCompID = SampleClient
_____________
FixLayer.FixEngine.Session.Session1.Schedule =  TestFIXSchedule
_____________
FixLayer.FixEngine.Session.Session1.ConnectTime = 0 0 8 * * 2-6
FixLayer.FixEngine.Session.Session1.DisconnectTime = 0 0 21 * * 2-6
FixLayer.FixEngine.Session.Session1.TerminateTime = 0 0 21 * * 6
_____________
```

Result:

The session will connect at 7:00 and disconnect at 23:00 on Monday, Tuesday, Wednesday, Thursday and Friday in EST time. Every Monday it will start with empty log files and sequence numbers will be reset. Default Session Schedule Properties and Old-Style Session Schedule Properties will be ignored. The schedule will not be executed on December 25th of each year.