
MIT502 – GUIDE TO APPLICATION CERTIFICATION

29 June 2026
Issue 19.0





LONDON
STOCK
EXCHANGE

Contents

1.0	Introduction	4
1.1	Purpose	4
1.2	Readership	4
1.3	Document Series	4
1.4	Document History	4
1.5	Document History	5
2.0	Certification Programme	6
2.1	Access to the live service	6
2.2	Software identification	6
2.3	Market data applications	6
2.4	Market data applications	6
2.5	Test scenario exception policy	7
2.6	Non-conformance behaviour on the live service	7
2.7	Test charges	7
3.0	Certification Process	8
3.1	Full Certification Test	8
4.0	Trading and Order Management Certification Scenarios	10
4.1	Trading and Order Management – Auction Orders	10
4.2	Trading and Order Management - Continuous Trading Orders	11
4.3	Trading and Order Management - RFQ Scenarios	25
5.0	Next Functional Release 1.26.9 Certification Scenario	41
6.0	Post-Trade Gateway Certification Scenarios	45
7.0	Drop Copy Gateway Certification Scenarios	46
8.0	Mandatory Scenarios for New Software Applications	47

1.0 Introduction

1.1 Purpose

The purpose of this document is to provide customers with a detailed overview of the certification service.

LSEG, London Stock Exchange Group, restricts access to its live services to software applications that have not completed the software certification process. The live services are all trading and information services on an eligible LSEG venue. The eligible venues are currently London Stock Exchange and Turquoise.

1.2 Readership

This document is particularly relevant to software developers, system testers and anyone else involved in developing a software application for use on any of the live services.

This document should be read in conjunction with LSEG millennium technical specifications and the product specific 'Guide to testing services' documents. Together, these documents provide all the relevant information that customers require to develop and certify to the services.

1.3 Document Series

This document is part of a series of customer testing documents that are available to customers. The other documents are:

- MIT501 Guide to testing services
- MIT503 Certification test report

1.4 Document History

This document has been through the following iterations:

Issue	Date	Description
2.0	23 May 2011	Second issue of this document published via the Exchange's website
3.0	24 August 2012	Second issue of this document published via the Exchange's website to include information to the Group Ticker Plant.
4.0	19 October 2012	This version also outlines an amended process. Group Ticker Plant has now been incorporated into the general policy
5.0	30 January 2013	Updated LCON Policy added
6.0	12 February 2013	Updated test cases and improved layout
7.0	15 March 2013	Added test case for LSE-046 for Randomized Iceberg orders. Added clarification around LCON policy for NSP clients.
8.0	14 June 2013	Updated Policy for Annual Certification
9.0	16 July 2013	Updated test case IDs
9.1	29 July 2013	Updated certification passport
9.2	24 January 2014	Updated section 2.3
10	29 August 2014	Document rebranded. Updated test cases 5.5.18, 5.5.19 and 5.5.20

10.1	31 October 2014	Added test cycles for Turquoise Block Discovery™ MIT-073, MIT-074, MIT-075. Amended to reflect rebranding ITCH to MITCH
10.2	15 April 2015	Added test cycles MIT-076, MIT-077
11.0	26 April 2015	Amended annual certification policy. Removed application passporting. Removed non-London Stock Exchange test cases
12.0	12 September 2015	Added test cycles for Request for Quote (RFQ) functional. Test cases 5.5.27, 5.5.28, 5.5.29
13.0	6 June 2017	Amended certification policy
14.0	24 July 2017	Updated section 2.3
15.0	29 August 2017	Section 2.0 – Updated wording Section 2.3 – Updated to reflect changes to certification policy Reformatted document Added new test cycle for MiFID II MIT-106, MIT-107, MIT-108
16.0	8 October 2018	Updated document to support the new RFQ Auto Complete Model Updated MIT126. Added MIT126A & MIT127A
17.0	20 April 2020	Removed MITCH test cases after MITCH decommission 4.2.7 – Added test case for ATC offset order 4.5.1 – Changed to mandatory-only for new software
17.1	13 May 2020	Added 4.2.8 – Order modification – ATC offset order
17.2	27 January 2021	Added 4.2.31 – Added test case for RFQ market maker ranking
18.0	2 February 2024	Updated Certification Process section to reflect the new conformance testing tool and the introduction of OKTA to access the new conformance testing Portal.
19.0	29 June 2026	Removed Section 4.1 Market Data Group Ticker Plant (GTP) test cases to avoid duplication with the GTP Testing Guide Removed test cases for BTF and Cross Orders MIT-118 – MIT-125 Removed test cases MIT-128 Updated 'Trading and Order Management' test cycles with messages or tags for 'Client Action' and 'Server Response' Added Section 5.0 – New test cases for Next Functional Release 1.26.9

In subsequent issues, where amendments have been made to the previous version, these changes will be highlighted in Red.

1.5 Document History

Please contact either the Technical Account Management Team or your Technical Account Manager if you have any questions about the Millennium Exchange services outlined in this document. Client Technology Services (UK) can be contacted at:

- Telephone: +44 (0)20 7797 3939 (Option 1)
- Email: londontam@lseg.com

2.0 Certification Programme

The certification programme is based on regulatory compliance supporting interoperability against the three eligible LSEG venues.

The following certification programme applies to anyone connecting a software application to an LSEG live service. A Live service is any production trading or information services environment across LSEG.

Under EU and national regulatory requirements (including the ESMA Guidelines on Systems and Controls in a Highly Automated Trading Environment and MiFID II Regulatory Technical Standards), the eligible LSEG venues are required to have procedures and arrangements to ensure fair and orderly trading. This includes requirements for physical and electronic security to protect systems from misuse or unauthorised access and to ensure the integrity of the data that is part of, or passes through, the systems. The eligible venues are required to undertake standardised certification testing to ensure that members' and participants' systems used to access the venues have a minimum level of functionality that is compatible with fair and orderly trading on those venues.

Customer non-compliance with this certification programme may constitute a breach of the eligible venue terms and conditions or rules.

2.1 Access to the live service

Access to LSEG live services is permitted only when a customer's software application has been certified as being fit for purpose.

2.2 Software identification

All customer software must be identifiable by a software name and version number. Software applications that do not have both a name and version number will not be certified. Certification is limited to a single version of the named software.

2.3 Market data applications

All customers must confirm that their market data applications are fully conformant to all aspects of the GTP technical specifications and technically behaves as described in the GTP002 document. Customer also confirms that its application correctly receives and manages the market data messages.

2.4 Market data applications

Customers will be required to perform a certification test under the following conditions:

- Prior to use of a new application on the live environment
- The customer modifies the software in any way that directly impacts a LSEG interface. This includes but is not limited to gateways, order management, execution management and quote management software.
- The exchange venue upgrades its production environment to a later version of software
- The customer is requested to re-certify their application by the relevant venue. If not certification has to be performed during a solar year because any of the previous conditions did not occur, the customer will be asked to self-certify by e-mail that no changes have been made to their application since the last certification test.

2.5 Test scenario exception policy

Customers only need to complete the test cases relating to the functionality that they will use on the live services. If a customer's application does not support the functionality described in a particular test scenario and they do not intend to complete the scenario during the test, this must be agreed before the start of the certification test.

2.6 Non-conformance behaviour on the live service

Any non-conformant behaviour by a customer's software application on the live services may lead to the software application being disconnected. It will not be re-connected until it has been re-certified, and the non-conformant behaviour has been corrected.

2.7 Test charges

The published venue-specific certification charges apply to all certification testing, regardless of the test being assisted or not.

3.0 Certification Process

The Certification process is an integral part of the Certification Programme.

Mandatory testing will be required for all applications that wish to connect to the production environment and mandatory functions will need to be tested to confirm conformant behaviour.

Customers should consider their software applications 'production ready' before attempting the test. Customers should also read and familiarise themselves with this document and the market and product-specific testing documents. These documents contain important guidance that must be read before completing the certification process. When ready to take the test, customers should complete the following steps:

Customers should contact EQ Technical Onboarding (eqtechnonboarding@lseg.com) to provide the software name and version, a brief description of the application and the gateways that it will connect to.

3.1 Full Certification Test

Before Taking the Test

Customers will need to be set-up on OKTA to manage their conformance test on the new front-end GUI – <https://conformance.lseg.com>.

To be set-up on OKTA, please provide the following information to EQ Technical Onboarding:

- Name and email address of the individual(s) performing the certification tests.
- Name of the company performing the certification
- Which interface(s) is requested for certification Native, FIX Trading, Post Trade, Drop Copy

Once the user certifying has been set-up, customers should expect to receive emails from OKTA to access the conformance testing tool via noreply@identity.lseg.com.

To perform conformance testing, customers will need to target the new customer conformance tool IP addresses. However, Native Ports are unique for each customer. Please contact your Technical Account Manager or the Technical Account Management team (londontam@lseg.com) for your allocated Port.

The EQ Technical Onboarding team will be available to assist participants in completing part of their certification test by performing both mandatory and optional functions.

Gateway connections and User IDs should be checked at least two working days before the planned execution date.

The [MIT503 Certification Report](#) must be filled in with the Software Version, Name and scenarios the customer wishes to execute and sent to: eqtechnonboarding@lseg.com.

This is a formal record of the software application that is being certified.

During the Test

Customers should complete the full set of uncoordinated test cycles and inform the relevant venue of LSEG when this has been completed. If required, limited support can be provided during this phase.

When all the uncoordinated test scenarios have been completed, the customer must then complete a set of

coordinated test scenarios, assisted by a tester.

When documenting the test results, customers should include all information as requested in each of the steps, such as Order IDs, QuoteMsg IDs and Security IDs.

If the result does not match the expected result, the customer should still record it in the relevant section of the certification test report and in the comments section including the following:

- Order IDs/QuoteMsg IDs (if any)
- Security ID
- Time of the order/quote/operation submission/execution
- Any information which may be helpful in troubleshooting the error

Unless a log off or disconnect request is specified in the test case, the customer should complete all test cases without logging off or disconnecting. If there are any unexpected log offs or disconnects, these should be documented in the comments section of the certification test report.

Note: All test scenarios marked as mandatory must be completed, unless an exception has been granted by the Technical Account Manager.

The customer can repeat this part as many times as necessary. However, additional test sessions may have to be booked if the testing exceeds two hours.

When all test scenarios have been completed, the customer should email the completed certification test report to the tester. The e-mail should be sent to eqtechnonboarding@lseg.com.

After the Test

After the test, the tester will check the details on the completed certification test report and review the log files for the completed test scenarios. A report is generated detailing how the customer's application has performed during the test. If no re-testing is required, the report will be sent to the customer to sign off the application as fit for purpose and ready for access to the live services.

The following items are checked:

- The functional behaviour and message sequencing in each scenario
- If the session was maintained for the entire period or if it dropped
- If any errors were produced over the time period

If multiple interfaces were included in the test, the results are concatenated into a single report. The report will be run, specifying a time range limited to a single day.

If the customer is required to repeat any of the testing, they should do so within 24 hours of the original test.

By returning the signed copy of the certification test report certification checklist, the customer is confirming that their application is fully conformant to all aspects of the millennium technical specifications and technically behaves as described in the guidance given in both this document and the guide to testing.

4.1 Trading and Order Management – Auction Orders

Objective:

Validate the behaviour of '**Limit**' Order during an **Opening or Closing Auction**.

Preconditions:

- The client must establish a valid connection.
- The system must be in an **Auction Phase** (Opening or Closing Auction).
- Ensure **matching orders exist** in the order book for execution (If there is no match for the order in the order book, please create a matching order).
- If no auction session is available, request LSEG support to manually move an instrument into auction.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a day 'Limit' order during opening or closing auction session. Message Type = D (New Order) Order Type = 2 (Limit Order) Price = greater than 0.00	Step 2: Server sends an 'Execution Report' confirming that the Order has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New)
(No client action in this step)	Step 3: Server sends an 'Execution Report' confirming that there has been an execution. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) or 2 (Filled) Trade Liquidity Indicator = C (Auction)

Objective:

Validate the behaviour of **'Market' Order** during an **Opening** or **Closing Auction**.

Preconditions:

- The client must establish a valid connection.
- The system must be in an **Auction Phase** (Opening or Closing Auction).
- Ensure **matching orders exist** in the order book for execution (If there is no match for the order in the order book, please create a matching order).
- If no auction session is available, request LSEG support to manually move an instrument into auction.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a day 'Market' order during Opening or Closing Auction session. Message Type = D (New Order) Order Type = 1 (Market) TimeInForce (TIF) = 0 (Day Order)	Step 2: Server sends an 'Execution Report' confirming that the Order has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New)
<i>(No client action in this step)</i>	Step 3: Server sends an 'Execution Report' confirming that there has been an execution. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) or 2 (Filled) Trade Liquidity Indicator = C (Auction)

4.2 Trading and Order Management - Continuous Trading Orders

Test Case ID: MIT-102 Market order

Objective:

Validate 'Market' Order behaviour during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Ensure **matching orders exist** in the order book for execution (If there is no match for the order in the order book, please create a matching order).

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a day 'Market' order during Continuous Trading session. Message Type = D (New Order) Order Type = 1 (Market) TimeInForce (TIF) = 0 (Day Order)	Step 2: Server sends an 'Execution Report' confirming that there has been an execution. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) or 2 (Filled)

Test Case ID: MIT-103 Limit Passive Only Order

Objective:

Validate the behaviour of 'Passive Only' 'Limit' Order during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Ensure **resting orders exist** in the order book for execution (If there are no orders in the order book, please create some).

Comments:

This is an order-level parameter that allows clients to require that their order must rest prior to execution, with flexibility for visible orders to rest at a specified price level on the orderbook.

Value Meaning

0	No constraint (Default)
99	Only accept order if it will not match with visible contra order. Otherwise expire order.
100	Only accept order if setting new visible BBO. Otherwise expire order.
1	Only accept order if setting new BBO or joining existing BBO. Otherwise expire order.
2	Only accept order if will be at BBO or within one visible price point. Otherwise expire order.
3	Only accept order if will be at BBO or within two visible price points. Otherwise expire order.

Note:

Setting PassiveOnlyOrder to values other than '0' or 'null' may still result in the order aggressively matching against hidden orders resting on the order book.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'Limit' 'Passive Only' order during Continuous Trading session with a price that violates selected conditions. Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 0 (Day Order) Price = price that violates selected conditions Passive Only Order <> 0 and <> 99 and Not Blank	Step 2: Server sends an 'Execution Report' confirming that the order has been expired. Native: Message Type = 8 (Execution Report) Execution Type = C (Expired) Order Status = 6 (Expired) FIX: Message Type = 8 (Execution Report) Execution Type = C (Expired) Order Status = C (Expired) Text = "Cannot Add Order at Requested Price Level"
Step 3: Client submits a 'Limit' 'Passive Only' order during Continuous Trading session with a price that would result in an execution. Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 0 (Day Order) Price = price that would result in an execution Passive Only Order <> 0 and Not Blank	Step 4: Server sends an 'Execution Report' confirming that the order has been expired. Native: Message Type = 8 (Execution Report) Execution Type = C (Expired) Order Status = 6 (Expired) FIX: Message Type = 8 (Execution Report) Execution Type = C (Expired) Order Status = C (Expired) Text = "Order Would Execute with Visible Order on Aggression"

Test Case ID: MIT-104 Aggressive Limit Order

Objective:

Validate behaviour of an **aggressive 'Limit' order** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Ensure **matching orders exist** in the order book for execution (If there is no match for the order in the order book, please create a matching order).

Comments:

There should be **NO** Execution Report confirming that the order has been Accepted (Execution Type = 0) **only** ER confirming execution (Execution Type = F)

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an aggressive 'Limit' order during continuous trading with a price that will trigger an immediate execution. Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 3 (IOC) or 4 (FOK) Price = price that will trigger an immediate execution	Step 2: Server sends an 'Execution Report' confirming that there has been an execution (there should be NO 'Execution Report' confirming that the order has been accepted (Execution Type = 0)) Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) or 2 (Filled) Trade Liquidity Indicator = R (Removed Liquidity)

Test Case ID: MIT-104A Passive Limit Order

Objective:

Validate behaviour of **passive 'Limit' orders** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session

Comments:

This test verifies entering a passive order without using the "PassiveOnlyOrder" field.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a passive 'Limit' order during Continuous Trading session with a price that is worse than the best available price in the order book without using the 'PassiveOnlyOrder' field. Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 0 (Day Order) Price = greater than 0.00 (but worse than the best available price) Passive Only Order = 0 or 'Blank'	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New)

Test Case ID: MIT-105 At the Close (ATC) Offset Order

Objective:

Validate the behaviour of 'At-the-Close' (ATC) 'Offset' order during the **Closing Auction**. The order is priced at the DRP +/- offset value specified in the order.

Preconditions:

- The client must establish a valid connection.
- The system must be in the **Closing Auction Phase**.
- If no closing auction session is available, request LSEG support to manually move an instrument into auction.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an 'ATC' 'Offset' limit order during Closing Auction with an Offset Price specified. Native: Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 10 (ATC) Order Sub Type = 55 (Offset) Offset <> 0 and Not Blank FIX: Message Type = D (New Order) Order Type = F (Offset) TimeInForce (TIF) = 7 (ATC) Offset <> 0 and Not Blank	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New)

Test Case ID: MIT-105A Modify ATC Offset Order

Objective:

Validate the behaviour of **modifying an 'At-the-Close' (ATC) 'Offset' order** during the **Closing Auction**. The order is priced at the DRP +/- offset value specified in the order.

Preconditions:

- The client must establish a valid connection.
- The system must be in the **Closing Auction** phase.
- If no closing auction session is available, request LSEG support to manually move an instrument into closing auction.
- An existing **ATC offset order** that will be modified must be available on the Order Book. If there are no existing orders, please enter one prior to testing this scenario.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client amends the 'ATC' 'Offset' limit order previously entered during Closing Auction. Native: Message Type = G (Order Cancel/Replace Request) Order ID = 'Order ID' of the offset order that is being amended Original Client Order ID = 'Client Order ID' of the Offset Order that is being amended (ignored if 'Order ID' has been provided) Order Type = 2 (Limit Order) TimeInForce (TIF) = 10 (ATC) Order Sub Type = 55 (Offset) Order Qty = Copied from the Offset Order that is being amended Offset = Different from the previous price and Not	Step 2: Server sends an 'Execution Report' confirming that the order has been replaced. Message Type = 8 (Execution Report) Execution Type = 5 (Replaced) Order Status = 0 (New)

Blank	
FIX: Message Type = G (Order Cancel/Replace Request) Order ID = 'Order ID' of the offset order that is being amended Original Client Order ID = 'Client Order ID' of the Offset Order that is being amended (ignored if 'Order ID' has been provided) Order Type = F (Offset) TimeInForce (TIF) = 7 (ATC) Order Qty = Copied from the Offset Order that is being amended Offset = Different from the previous price and Not Blank	

Test Case ID: MIT-106 Algorithm-Generated Order

Objective:

Validate the behaviour of **Algorithm-Generated Orders** submitted during a **Continuous Trading** session.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an order with Algo field set in Bit Field 'FXMiFID Flag ALGO' (Native) / 'OrderAttributeType' = '4' (Algorithm) (Fix) which indicates an order was generated via an algorithm. Native: Message Type = D (New Order) Order Type = 1 (Market) or 2 (Limit Order) TimeInForce (TIF) = 0 (Day) Price = greater than 0.00 (if Limit Order) MiFID Flags = ALGO FIX: Message Type = D (New Order) Order Type = 1 (Market) or 2 (Limit Order) TimeInForce (TIF) = 0 (Day) Price = greater than 0.00 (if Limit Order) OrderAttributeType = 4 (Algorithm)	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled)

Test Case ID: MIT-107 Liquidity Provision Order

Objective:

Validate the behaviour of **Liquidity Provision Orders** submitted during a **Continuous Trading** session.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an order with Liquidity Provision field set in Bit Field 'FXMiFID Flag Liquidity Provision' (Native) / 'OrderAttributeType' = '2' (Liquidity Provision) (Fix) which indicates an order was generated as part of liquidity provision activity. Native: Message Type = D (New Order) Order Type = 1 (Market) or 2 (Limit Order) TimeInForce (TIF) = 0 (Day) Price = greater than 0.00 (if Limit Order) MiFID Flags = Liquidity Provision FIX: Message Type = D (New Order) Order Type = 1 (Market) or 2 (Limit Order) TimeInForce (TIF) = 0 (Day) Price = greater than 0.00 (if Limit Order) OrderAttributeType = 2 (Liquidity Provision)	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled)

Test Case ID: MIT-108 Direct Electronic Access (DEA) Order

Objective:

Validate the behaviour of **Direct Electronic Access (DEA) Orders** submitted during a **Continuous Trading** session.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an order with DEA flag set in Bit Field 'FXMiFID Flag DEA' (Native) / 'OrderOrigination' = '5' (DEA) (Fix) which indicates that the order is a DEA Order. Native: Message Type = D (New Order) Order Type = 1 (Market) or 2 (Limit Order) TimeInForce (TIF) = 0 (Day) Price = greater than 0.00 (if Limit Order) MiFID Flags = DEA FIX: Message Type = D (New Order) Order Type = 1 (Market) or 2 (Limit Order) TimeInForce (TIF) = 0 (Day) Price = greater than 0.00 (if Limit Order) OrderOrigination = 5 (DEA)	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled)

Test Case ID: MIT-109 Mid-Point Pegged Order

Objective:

Validate the behaviour of **Mid-Point 'Pegged' order** submitted during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Ensure there are **buy/sell orders on the order book** to allow mid-point calculation.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Native - Client submits a 'Pegged' order, 'Order Sub Type' = '5' (Pegged) (Native) / 'Order Type' = 'P' (Pegged) (Fix). Native: Message Type = D (New Order) Order Type = 1 (Market) or 2 (Limit Order) TimeInForce (TIF) = 0 (Day) Price = greater than 0.00 (if Limit Order) Order Sub Type = 5 (Pegged Order) FIX: Message Type = D (New Order) Order Type = P (Pegged) TimeInForce (TIF) = 0 (Day) Price = greater than 0.00 (if Limit Order)	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled)

Test Case ID: MIT-110 Stop Order

Objective:

Validate the behaviour of 'Stop' market order submitted during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Ensure there are **buy/sell orders on the order book** to allow stop price trigger.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'Stop' market order with Stop Price better than the last traded automated trade price. Message Type = D (New Order) Order Type = 3 (Stop) TimeInForce (TIF) = 0 (Day) Price = 0 or Blank Stop Price = better than last traded automated trade price	Step 2: Server sends an 'Execution Report' confirming that the order has been parked (suspended). Message Type = 8 (Execution Report) Execution Type = 9 (Suspended) Order Status = 9 (Suspended) Note: Before Step 3, please execute a trade so that the last traded automated trade price is better than the Stop Price in order to trigger Stop Order injection onto the order book.
Step 3: Client or LSEG execute a trade so that the last traded automated trade price is better than the Stop Price in order to trigger Stop Order injection onto the order book.	
(No client action)	Step 4: Server sends an 'Execution Report' confirming that the order has been executed. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) or 2 (Filled) Trade Liquidity Indicator = R (Removed Liquidity)

Test Case ID: MIT-111 Stop Limit Order

Objective:

Validate the behaviour of 'Stop Limit' order submitted during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Ensure there are **buy/sell orders on the order book** to allow stop price trigger.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'Stop Limit' order with Limit Price > 0 and Stopped Price better than the last automated trade price. Message Type = D (New Order) Order Type = 4 (Stop Limit) TimeInForce (TIF) = 0 (Day) Price > 0 Stopped Price = better than the last automated trade price	Step 2: Server sends an 'Execution Report' confirming that the order has been parked (suspended). Message Type = 8 (Execution Report) Execution Type = 9 (Suspended) Order Status = 9 (Suspended)

Step 3: Client or LSEG execute a trade so that the last traded automated trade price is better than the Stop Price in order to trigger Stop Order injection onto the order book.

(No client action)

Step 4: Server sends an 'Execution Report' confirming that the order has been accepted/executed.

Message Type = 8 (Execution Report)
Execution Type = 0 (New) / F (Trade)
Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled)

Test Case ID: MIT-112 Hidden Order

Objective:

Validate the behaviour of 'Hidden' order submitted during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Ensure the **Display Qty** (visible portion of the order) is set to 0.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'Hidden' order during Continuous Trading session. Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 0 (Day Order) Price > 0.00 Order Qty > 0 Display Qty = 0	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled)

Test Case ID: MIT-113 Iceberg Order

Objective:

Validate the behaviour of 'Iceberg' order with a **Fixed Peak Size** submitted during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- Ensure that only visible portion of the order (**Display Qty**) is displayed in the Order Book.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an 'Iceberg' order with a Fixed Peak Size. Message Type = D (New Order) Order Type = 2 (Limit) TimeInForce (TIF) = 0 (Day) Price > 0.00 Order Qty = equal to or greater than EMS Display Qty = equal to or greater than the Minimum Peak Size and less than 50% of Order Qty Note: Make sure that only 'Display Qty' is visible in order book.	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New)
(No client action in this step)	Step 3: LSEG submit an order with 'Order Qty' = 'Display Qty' of the original Iceberg Order, so that the peak is fully executed, forcing a peak refresh.
(No client action in this step)	Step 4: Server sends an 'Execution Report' confirming that the peak has been fully executed. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) Type Of Trade = 0 (Visible)
	Step 5: Server sends an 'Execution Report' confirming that the peak has been replenished.

(No client action in this step)	Native: Message Type = 8 (Execution Report) Execution Type = D (Restated) Order Status = 1 (Partially filled) Restatement Reason = 100 (Order Replenishment) FIX: Message Type = 8 (Execution Report) Execution Type = D (Restated) Order Status = 1 (Partially filled) ExecRestatement = 100 (Order Replenishment)
(No client action in this step)	Step 6: LSEG submit an order with 'Order Qty' > 'Display Qty' of the restated Iceberg Order, so that both visible and hidden portions of the Iceberg Order are executed.
(No client action in this step)	Step 7: Server sends an 'Execution Report' confirming that the peak has been fully executed. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) Type Of Trade = 0 (Visible)
(No client action in this step)	Step 8: Server sends an 'Execution Report' confirming that the hidden portion of the Iceberg Order has been executed. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) / 2 (Filled) Type Of Trade = 1 (Hidden)

Test Case ID: MIT-114 Randomised Iceberg Order

Objective:

Validate the behaviour of 'Iceberg' order with a **Randomised Peak Size** submitted during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- Ensure that only visible portion of the order (**Display Qty**) is displayed in the Order Book.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an 'Iceberg' order with a Randomised Peak Size. Message Type = D (New Order) Order Type = 2 (Limit) Order Sub Type = 51 (Random Peak size) (Native only) Display Method = 3 (Randomise Value) (Fix only) TimeInForce (TIF) = 0 (Day) Price > 0.00 Order Qty = equal to or greater than EMS Display Qty = equal to or greater than the Minimum Peak Size and less than 50% of Order Qty Note: Make sure that only Peak Size/Display Qty is visible in order book	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New)
(No client action in this step)	Step 3: LSEG submit an order with 'Order Qty' = 'Display Qty' of the original Iceberg Order, so that the peak is fully executed, forcing a peak refresh.

(No client action in this step)	Step 4: Server sends an 'Execution Report' confirming that the peak has been fully executed. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled) TypeOfTrade = 0 (Visible)
(No client action in this step)	Step 5: Server sends an 'Execution Report' confirming that the peak has been replenished. Native: Message Type = 8 (Execution Report) Execution Type = D (Restated) Order Status = 1 (Partially filled) Restatement Reason = 100 (Order Replenishment) FIX: Message Type = 8 (Execution Report) Execution Type = D (Restated) Order Status = 1 (Partially filled) ExecRestatement = 100 (Order Replenishment)

Note: Please check if the Peak Size has been randomised (new 'Display Qty' > previous 'Display Qty')

Test Case ID: MIT-115 Named Order

Objective:

Validate the behaviour of **'Named' order** submitted during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.

Comments:

'Named' order is an order for which the identity of the submitting member is disclosed in the pre-trade market data feed (GTP).

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'Named' order during Continuous Trading. Native: Message Type = D (New Order) Order Type = 2 (Limit Order) Price > 0.00 Anonymity = 1 (Named) FIX: Message Type = D (New Order) Order Type = 2 (Limit Order) Price > 0.00 PreTrade Anonymity = N (Named)	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled)

Test Case ID: MIT-116 Order Modification

Objective:

Validate the behaviour of 'Order Cancel/Replace Request' submitted during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- An existing Limit Order that will be modified must be available on the Order Book. If there are no existing orders, please enter one prior to testing this scenario.

Comments:

For purpose of this scenario please modify one or both of the following attributes of a live order:

- (i) Order quantity
- (ii) Limit price

The client should identify the order being amended by either the 'Original Client Order ID' or 'Order ID'. If an 'Order Cancel/Replace Request' contains values for both 'Original Client Order ID' and 'Order ID', the server will only process the Order ID.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client amends an existing 'Limit' order. Native: Message Type = G (Order Cancel/Replace Request) Order ID = 'Order ID' of the order that is being amended (or provide 'Original Client Order ID' / 'OrigClOrdID' instead) Price = Different from the original Limit Price (if price is being modified) Order Qty = Different from the original Order Qty (if quantity is being modified) Original Client Order ID = 'Client Order ID' of the Order that is being amended (ignored if 'Order ID' has been provided) FIX: Message Type = G (Order Cancel/Replace Request) Order ID = 'Order ID' of the order that is being amended (or provide 'Original Client Order ID' / 'OrigClOrdID' instead) Price = Different from the original Limit Price (if price is being modified) Order Qty = Different from the original Order Qty (if quantity is being modified) OrigClOrdID = 'ClOrdID' of the Order that is being amended (ignored if 'Order ID' has been provided)	Step 2: Server sends an 'Execution Report' confirming that the order has been replaced. Message Type = 8 (Execution Report) Execution Type = 5 (Replaced) Order Status = 0 (New) / 1 (Partially filled)

Test Case ID: MIT-117 Order Cancel

Objective:

Validate the behaviour of 'Order Cancel Request' submitted during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- An existing Limit Order that will be cancelled must be available on the Order Book. If there are no existing orders, please enter one prior to testing this scenario.

Comments:

The client should identify the order being cancelled by either the 'Original Client Order ID' or 'Order ID'. If an 'Order Cancel Request' contains values for both 'Original Client Order ID' and 'Order ID', the server will only process the 'Order ID'.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client cancels an existing 'Limit' order. Native: Message Type = F (Order Cancel Request) Order ID = 'Order ID' of the Order that is being cancelled (or provide 'Original Client Order ID' instead) Original Client Order ID = 'Client Order ID' of the order that is being amended (ignored if 'Order ID' has been provided) FIX: Message Type = F (Order Cancel Request) Order ID = 'Order ID' of the Order that is being cancelled (or provide 'Original Client Order ID' instead) OrigClOrdID = 'ClOrdID' of the order that is being amended (ignored if 'Order ID' has been provided)	Step 2: Server sends an 'Execution Report' confirming that the order has been replaced. Message Type = 8 (Execution Report) Execution Type = 4 (Cancelled) Order Status = 4 (Cancelled)

4.3 Trading and Order Management - RFQ Scenarios

Test Case ID: MIT-126 Submit RFQ (Manual Model) Requestor

Objective:

Validate the behaviour of 'Request for Quote' (RFQ) submission in Manual Mode by the Requestor during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- The participant must be authorized to **submit RFQs**.
- The instrument must be eligible for **RFQ trading**.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'Quote Request' message for manual RFQ model. Native: Message Type = a (Quote Request) Quote Request Type = 0 (Manual) FIX: Message Type = R (Quote Request) Quote Request Type = 1 (Manual)	Step 2: Server sends a 'Quote Status Report' confirming that the quote has been accepted. Native: Message Type = c (Quote Status Report) Quote Status = 1 (Accepted) FIX: Message Type = AI (Quote Status Report) Quote Status = 0 (Accepted)

Test Case ID: MIT-126A Submit RFQ (Auto-Complete Model) Requestor

Objective:

Validate the behaviour of 'Request for Quote' (RFQ) submission in Auto-Complete Mode by the Requestor during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- The participant must be authorized to **submit RFQs**.
- The instrument must be eligible for **RFQ trading**.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'Quote Request message' for Auto-Complete RFQ model. Native: Message Type = a (Quote Request) Quote Request Type = 1 (Automatic) FIX: Message Type = R (Quote Request) Quote Request Type = 2 (Automatic)	Step 2: Server sends a 'Quote Status Report' confirming that the quote has been accepted. Native: Message Type = c (Quote Status Report) Quote Status = 1 (Accepted) FIX: Message Type = AI (Quote Status Report) Quote Status = 0 (Accepted)

Test Case ID: MIT-126B Submit RFQ with Market Maker Rank - Requestor

Objective:

Validate the behaviour of 'Request for Quote (RFQ)' submission with Market Maker Ranking by the Requestor during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- The participant must be authorized to **submit RFQs**.
- The instrument must be eligible for **RFQ trading**.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'Quote Request' message with Market Maker Rank > 0 for Auto-Complete RFQ model. Native: Message Type = a (Quote Request) Quote Request Type = 1 (Automatic) Market Maker Rank > 0 FIX: Message Type = R (Quote Request) Quote Request Type = 2 (Automatic) Market Maker Rank > 0	Step 2: Server sends a 'Quote Status Report' confirming that the quote has been accepted. Native: Message Type = c (Quote Status Report) Quote Status = 1 (Accepted) FIX: Message Type = AI (Quote Status Report) Quote Status = 0 (Accepted)

Test Case ID: MIT-127a RFQ Response by Market Maker (Reject)

Objective:

Validate the behaviour of **Market Maker Response to a 'Request for Quote (RFQ)'** during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- The participant must be authorized to **submit RFQs**.
- The instrument must be eligible for **RFQ trading**.

Test Steps & Expected Results

Client Action	Server Response
(No client action in this step)	Step 1: Server sends a 'Quote Request' message to the Client. Native: Message Type = a (Quote Request) FIX: Message Type = R (Quote Request)
Step 2: Client (Market Maker) sends a 'Quote Request Reject' message confirming that the 'Quote Request' has been rejected. Native: Message Type = b (Quote Request Reject) FIX: Message Type = AG (Quote Request Reject)	(No server response in this step)

Test Case ID: MIT-127b RFQ Response by Market Maker (Accept)

Objective:

Validate the behaviour of **Market Maker Response to a 'Request for Quote (RFQ)'** during Continuous Trading.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- The participant must be authorized to **submit RFQs**.
- The instrument must be eligible for **RFQ trading**.

Test Steps & Expected Results

Client Action	Server Response
<i>(No client action in this step)</i>	Step 1: Server sends a 'Quote Request' message to the Client. Native: Message Type = a (Quote Request) FIX: Message Type = R (Quote Request)
Step 2: Client (Market Maker) sends an 'RFQ Quote' confirming that the 'Quote Request' has been accepted. Native: Message Type = d (RFQ Quote) FIX: Message Type = S (Quote) Quote Type = 2 (RFQ Quote)	Step 3: Server sends an 'Execution Report' confirming that the 'RFQ Quote' has been executed. Native: Message Type = g (RFQ Execution Report) Execution Type = F (Trade) Order Status = 2 (Filled) FIX: Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 2 (Filled) TrdType = 99 (RFQ trade)

Test Case ID: MIT-127c Requester Hits/Lifts an RFQ Quote (manual model only)

Objective:

Validate the behaviour of **Requester Hitting/Lifting a Market Maker 'RFQ Quote' in Manual Mode** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- The participant must be authorized to **submit RFQs**.
- The instrument must be eligible for **RFQ trading**.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client sends a 'Quote Request' for Manual RFQ model. Native: Message Type = a (Quote Request) Quote Request Type = 0 (Manual) FIX: Message Type = R (Quote Request) Quote Request Type = 1 (Manual)	Step 2: Server sends a 'Quote Status Report' confirming that the 'Quote Request' has been accepted. Native: Message Type = c (Quote Status Report) Quote Status = 1 (Accepted) FIX: Message Type = AI (Quote Status Report) Quote Status = 0 (Accepted)
(No client action in this step)	Step 3: Server sends an 'RFQ Quote' confirming that the 'Quote Request' has been accepted by the Market Maker. Native: Message Type = d (RFQ Quote) FIX: Message Type = S (Quote) Quote Type = 2 (RFQ Quote)
Step 4: Client accepts (Hit/Lift) the 'RFQ Quote'. Native: Message Type = f (Quote Response) Quote Response Type = 1 (Hit/Lift) FIX: Message Type = AJ (Quote Response) Quote Response Type = 1 (Hit/Lift)	Step 5: Server sends an 'Execution Report' confirming that the 'RFQ Quote' has been executed. Native: Message Type = g (RFQ Execution Report) Execution Type = F (Trade) Order Status = 2 (Filled) FIX: Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 2 (Filled) TrdType = 99 (RFQ trade)

Test Case ID: MIT-129 Cancel on Disconnect/Log-Out

Objective:

Validate the behaviour of 'Cancel on Disconnect' / 'Cancel on Log-Out' functionality during **Continuous Trading**.

Preconditions:

- The CompID must be logged in.
- The system must be in a **Continuous Trading** session.
- **Cancel on Disconnect** feature must be enabled for the CompID used for testing.
- CompID that is being tested must have **open orders** on the order book (at least 1 per each Partition/AppID).

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client terminates the session by either TCP disconnect for 'Cancel on Disconnect' or logging out for 'Cancel on Log-Out'.	<i>(No server response in this step – flow differs for each protocol below)</i>
Step 2: Client re-establishes a connection and logs on.	FIX: Execution reports for all the cancelled orders are received automatically.
Native only: Step 3: Client successfully logs on to the Recovery Channel.	<i>(No server response in this step)</i>
Step 4: Client sends a 'Missed Message Request' for Load ID 1 (also known as App ID 1) specifying the 'Last Msg Seq Num' that is known for that Load ID. Message Type = M (Missed Message Request) App ID = 1 (For Load ID 1, = 2 for Load ID 2 and so on) Last Msg Seq Num = Last received Sequence No. for that Load ID	Step 5: Server sends a 'Missed Message Request Ack' confirming that the request has been accepted. Message Type = N (Missed Message Request Ack) Response Type = 0 (Successful)
<i>(No client action in this step)</i>	Step 6: Server sends an 'Execution Report(s)' (one for each order) confirming that the Order(s) has/have been expired. Message Type = 8 (Execution Report) Execution Type = C (Expired) Order Status = 6 (Expired)
<i>(No client action in this step)</i>	Step 7: Server sends an 'Missed Message Report' confirming that all requested messages have been sent. Message Type = P (Missed Message Report) Response Type = 0 (Download Complete)

For Native repeat Steps 4–7 for each Load ID/App ID

Test Case ID: MIT-130 Exchange Cancels Client's Order

Objective:

Validate the behaviour of **cancelling Client's active Order** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a passive 'Limit' order during Continuous Trading session with a price that ensures that the order will reside on the order book. Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 0 (Day Order) Price = price that ensures that the order will reside on the order book	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New)
Step 3: Client requests that the Exchange cancel the order submitted in the previous step.	Step 4: Server sends an 'Execution Report' confirming that the order has been cancelled by the Exchange. Native: Message Type = 8 (Execution Report) Execution Type = 4 (Cancelled) Order Status = 4 (Cancelled) Restatement Reason = 8 (Order is cancelled by market operations) FIX: Message Type = 8 (Execution Report) Execution Type = 4 (Cancelled) Order Status = 4 (Cancelled) ExecRestatement = 8 (Order is cancelled by market operations)

Test Case ID: MIT-131 Exchange Cancels Client's Quote

Objective:

Validate the behaviour of **cancelling Client's active Quote** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a 'New Quote' / 'Quote' during Continuous Trading session. Native: Message Type = S (New Quote) Fix: Message Type = S (Quote) QuoteType = 1 (Electronic Quote)	Step 2: Native - Server sends Two 'Execution Reports' (one for each side) confirming that the quote has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New) FIX - Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted) QuoteType = 1 (Electronic Quote)
Step 3: Client requests that the Exchange cancel the quote submitted in the previous step.	Step 4: Native - Server sends Two 'Execution Reports' confirming that the quote has been cancelled by the Exchange. Message Type = 8 (Execution Report) Execution Type = 4 (Cancelled) Order Status = 4 (Cancelled) FIX - Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 6 (Removed from Market)

Test Case ID: MIT-132A Exchange Cancels Client's Trade – fully filled

Objective:

Validate the behaviour of **cancelling a fully filled Client's Trade** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an aggressive 'Limit' order during Continuous Trading session - order <u>must be fully executed</u> on entry. Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 0 (Day Order) Price = aggressive price - order must be fully executed on entry	Step 2: Server sends an 'Execution Report' confirming that the order has been fully executed. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 2 (Filled)
Step 3: Client requests that the Exchange cancel the trade.	Step 4: Server sends an 'Execution Report' confirming that the trade has been cancelled by the Exchange. Message Type = 8 (Execution Report) Execution Type = H (Trade Cancel) Order Status = 0 (New)
(No client action in this step)	Step 5: Server sends an 'Execution Report' confirming that the original order has been cancelled by the Exchange. Native: Message Type = 8 (Execution Report) Execution Type = 4 (Cancelled) Order Status = 4 (Cancelled) Restatement Reason = 8 (Trade is cancelled by market operations) FIX: Message Type = 8 (Execution Report) Execution Type = 4 (Cancelled) Order Status = 4 (Cancelled) ExecRestatement = 8 (Trade is cancelled by market operations)

Test Case ID: MIT-132B Exchange Cancels Client's Trade – partial fill

Objective:

Validate the behaviour of **cancelling a partially filled Client's Trade** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an aggressive 'Limit' order during Continuous Trading session - order <u>must be partially executed</u> on entry. Message Type = D (New Order) Order Type = 2 (Limit Order) TimeInForce (TIF) = 0 (Day Order) Price = aggressive price - order must be partially executed on entry (with some quantity remaining on the order book)	Step 2: Server sends an 'Execution Report' confirming that the order has been partially executed. Message Type = 8 (Execution Report) Execution Type = F (Trade) Order Status = 1 (Partially filled)
Step 3: Client requests that the Exchange cancel the trade.	Step 4: Server sends an 'Execution Report' confirming that the trade has been cancelled by the Exchange. Message Type = 8 (Execution Report) Execution Type = H (Trade Cancel) Order Status = 0 (New)
(No client action in this step)	Step 5: Server sends an 'Execution Report' confirming that the original order has been restated. Native: Message Type = 8 (Execution Report) Execution Type = D (Restated) Order Status = 0 (New) Restatement Reason = 8 (Trade is cancelled by market operations) Quantity = (original) 'Order Qty' – 'Executed Qty' FIX: Message Type = 8 (Execution Report) Execution Type = D (Restated) Order Status = 0 (New) ExecRestatement = 8 (Trade is cancelled by market operations) Quantity = (original) 'Order Qty' – 'LastQty'

Test Case ID: MIT-133 New Executable Quote

Objective:

Validate the behaviour of **submitting a new executable quote** during **Continuous Trading**.

Preconditions:

- The system must be in the **Continuous Trading** phase.
- The participant must be authorized to **submit executable quotes**.
- The instrument/security must be eligible for **quote-driven trading**.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a new executable quote message. Native: Message Type = S (New Quote) FIX: Message Type = S (Quote) QuoteType = 1 (Electronic Quote)	Step 2: <u>Native</u> - Server sends <u>Two</u> 'Execution Reports' (one for each side) confirming that the quote has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New) <u>FIX</u> - Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted)

Test Case ID: MIT-134 Modify Executable Quote

Objective:

Validate the behaviour of **modifying an existing executable quote** during **Continuous Trading**.

Preconditions:

- The system must be in the **Continuous Trading** phase.
- The participant must be authorized to **submit executable quotes**.
- The instrument/security must be eligible for **quote-driven trading**.

Comments:

As there can only be one quote for any given instrument, a quote is modified by sending a new quote which replaces the existing quote.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a new executable quote message. Native: Message Type = S (New Quote) FIX: Message Type = S (Quote) QuoteType = 1 (Electronic Quote)	Step 2: Native - Server sends Two 'Execution Reports' (one for each side) confirming that the quote has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New) FIX - Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted)
Step 3: Client sends a new executable quote that amends/replaces the existing one. Native: Message Type = S (New Quote) FIX: Message Type = S (Quote) QuoteType = 1 (Electronic Quote)	Step 4: Native - Server sends Two 'Execution Reports' (one for each side) confirming that the quote amendment has been accepted. Message Type = 8 (Execution Report) Execution Type = 5 (Replaced) Order Status = 0 (New) FIX - Server sends 'Quote Status Report' confirming that the quote amendment has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted)

Test Case ID: MIT-135 Cancel Executable Quote

Objective:

Validate the behaviour of **cancelling an active executable quote** during **Continuous Trading**.

Preconditions:

- The system must be in the **Continuous Trading** phase.
- The participant must be authorized to **submit executable quotes**.
- The instrument/security must be eligible for **quote-driven trading**.
- The Market Maker must have an **active executable quote** on the order book

Comments:

Native – Quote is cancelled by sending a 'Order Cancel Request' message. Client can identify the quote being cancelled by either the 'Original Client Order ID' or 'Order ID', if message contains both, server will only process the 'Order ID'.

Fix – Quote is cancelled by sending a 'Quote Cancel' message. The message should include a 'QuoteCancelType' (298) = 1 (Cancel for Instruments), a 'NoQuoteEntries' (295) = 1 and the instrument to which the cancellation applies 'SecurityID' (48). It is necessary to include the 'NoTargetPartyIDs' (1461) block in the message and the 'TargetPartyRole' (1464) should be 'Trader Group' (76).

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a new executable quote message. Native: Message Type = S (New Quote) FIX: Message Type = S (Quote) QuoteType = 1 (Electronic Quote)	Step 2: Native - Server sends Two 'Execution Reports' (one for each side) confirming that the quote has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New) FIX - Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted)
Step 3: Native - Client sends an 'Order Cancel Request to cancel the existing quote. Message Type = F (Order Cancel Request) Order ID = 'Order ID' of the quote that is being cancelled (or provide Original Client Order ID instead) Original Client Order ID = Client Order ID of the Quote that is being cancelled (ignored if Order ID has been provided) FIX - Client sends a 'Quote Cancel' to cancel the existing quote. Message Type = Z (Quote Cancel) QuoteCancelType = 1 (Cancel for Instruments), NoQuoteEntries = 1 SecurityID = Instrument ID for which the quote is being cancelled	Step 4: Native - Server sends Two 'Execution Reports' (one for each side) confirming that the quote has been cancelled. Message Type = 8 (Execution Report) Execution Type = 4 (Cancelled) Order Status = 4 (Cancelled) FIX - Server sends 'Quote Status Report' confirming that the quote has been cancelled. Message Type = AI (Quote Status Report) QuoteStatus = 17 (Cancelled)

Test Case ID: MIT-136 New Firm Quote (FIX only)

Objective:

Validate the behaviour of **submitting a new firm quote** during **Continuous Trading**.

Preconditions:

- The system must be in the **Continuous Trading** phase.
- The participant must be authorized to **submit firm quotes**.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a new firm quote message. Message Type = S (Quote) QuoteType = 0 (Firm Quote)	Step 2: Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted)

Test Case ID: MIT-137 Modify Firm Quote (FIX only)

Objective:

Validate the behaviour of **modifying firm quote** during **Continuous Trading**.

Preconditions:

- The system must be in the **Continuous Trading** phase.
- The participant must be authorized to **submit firm quotes**.

Comments:

As there can only be one quote for any given instrument, a firm quote is modified by sending a new quote which replaces the existing quote.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a new firm quote message. Message Type = S (Quote) QuoteType = 0 (Firm Quote)	Step 2: Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted)
Step 3: Client submits a new firm quote message that amends/replaces the existing one. Message Type = S (Quote) QuoteType = 0 (Firm Quote)	Step 4: Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted)

Test Case ID: MIT-138 Cancel Firm Quote (FIX only)

Objective:

Validate the behaviour of **cancelling firm quote** during **Continuous Trading**.

Preconditions:

- The system must be in the **Continuous Trading** phase.
- The participant must be authorized to **submit firm quotes**.
- The Market Maker must have an **active firm quote** on the order book

Comments:

Quote is cancelled by sending a 'Quote Cancel' message. The message should include a 'QuoteCancelType' (298) = 1 (Cancel for Instruments), a 'NoQuoteEntries' (295) = 1 and the instrument to which the cancellation applies 'SecurityID' (48). It is necessary to include the 'NoTargetPartyIDs' (1461) block in the message and the 'TargetPartyRole' (1464) should be 'Trader Group' (76).

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a new firm quote message. Message Type = S (Quote) QuoteType = 0 (Firm Quote)	Step 2: Server sends 'Quote Status Report' confirming that the quote has been accepted. Message Type = AI (Quote Status Report) QuoteStatus = 0 (Accepted)
Step 3: Client sends a 'Quote Cancel' to cancel the existing quote. Message Type = Z (Quote Cancel) QuoteCancelType = 1 (Cancel for Instruments), NoQuoteEntries = 1 SecurityID = Instrument ID for which the quote is being cancelled	Step 4: Server sends 'Quote Status Report' confirming that the quote has been cancelled. Message Type = AI (Quote Status Report) QuoteStatus = 17 (Cancelled)

Test Case ID: MIT-139 Mass Cancellation

Objective:

Validate the behaviour of **Mass Order Cancellation** during **Continuous Trading**.

Preconditions:

- The system must be in the **Continuous Trading Phase**.
- The participant must be authorized to **perform mass order cancellations**.
- The participant must have **multiple active orders on** the order book.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an 'Order Mass Cancel Request' message. Native: Message Type = q (Order Mass Cancel Request) Mass Cancel Request Type = 8 (All firm orders) Order Sub Type = 0 (Order) FIX: Message Type = q (Order Mass Cancel Request) Mass Cancel Request Type = 7 (Cancel All Orders)	Step 2: Server sends 'Order Mass Cancel Report(s)' (one for each App ID) confirming that the request has been accepted. Native: Message Type = r (Order Mass Cancel Report) Mass Cancel Response = 7 (Accepted) FIX: Message Type = r (Order Mass Cancel Report) Mass Cancel Response = 7 (Cancelled All Orders)
<i>(No client action in this step)</i>	Step 3: Server sends 'Execution Report(s)' one for each order that has been cancelled. Message Type = 8 (Execution Report) Execution Type = 4 (Cancelled) Order Status = 4 (Cancelled) The Client Order ID = The Client Order ID of the Order Mass Cancel Request

Test Case ID: MIT-140 Requesting Missed Messages (Native only)

Objective:

Validate that the 'Missed Message Request' (Message Recovery Mechanism) on Native Recovery works as expected.

Preconditions:

- The CompID must be logged in to the **Native Trading Gateway**.
- The system must be in a **Continuous Trading** session.
- CompID that is being tested must have **open orders** on the order book (at least 1 per each Load ID/App ID).

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client terminates the session by dropping the connection/log-out if 'Cancel on Disconnect'/'Cancel on Log-out' is enabled. Alternatively, Client logs out and contacts LSEG and request them to cancel the open orders on their behalf.	(No server response in this step)
Step 2: Client re-establishes a connection and logs on to the Real-time Channel.	(No server response in this step)
Step 3: Client successfully logs on to the Recovery Channel.	(No server response in this step)
Step 4: Client sends a 'Missed Message Request' for Load ID 1 (also known as App ID 1) specifying the 'Last Msg Seq Num' that is known for that Load ID. Message Type = M (Missed Message Request) App ID = 1 (For Load ID 1, = 2 for Load ID 2 and so on) Last Msg Seq Num = Last received Sequence No. for that Load ID	Step 5: Server sends a 'Missed Message Request Ack' confirming that the request has been accepted. Message Type = N (Missed Message Request Ack) Response Type = 0 (Successful)
(No client action in this step)	Step 6: Server sends an 'Execution Report(s)' (one for each order) confirming that the Order(s) has/have been expired. Message Type = 8 (Execution Report) Execution Type = C (Expired) Order Status = 6 (Expired)
(No client action in this step)	Step 7: Server sends an 'Missed Message Report' confirming that the Order(s) has/have been expired. Message Type = P (Missed Message Report) Response Type = 0 (Download Complete)

Repeat Steps 4–7 for each Load ID/App ID

5.0 Next Functional Release 1.26.9 Certification Scenario

Test Case: Mid-Priced Pegged Orders (PeggedExecInst)

Objective:

Validate behaviour of **Mid-Priced 'Pegged' Orders** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Executions should happen on Venue MIC XLOX

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a Mid-Priced 'Pegged' order, 'Order Sub Type' = '5' (Pegged) (Native) / 'Order Type' = 'P' (Pegged) (FIX). Native: Message Type = D (New Order) Order Sub Type = 5 (Pegged) Price = greater than 0.00 (if Limit Order) Pegged Exec Inst = 0 (Default), 1 (Permitted to execute), FIX: Message Type = D (New Order) Order Type = P (Pegged) Price = greater than 0.00 (if Limit Order) PeggedExecInst = 0 (Default), 1 (Permitted to execute),	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) LastMkt = XLOX On Exchange Mid-Priced Executions Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled) PeggedExecInst = 1 Permitted for execution against mid-priced pegged orders

Test Case: Mid-Point Pegged Quote (PeggedExecInst)

Objective:

Validate behaviour of **Mid-Price 'Pegged' Quote** during **Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- Quote must be an electronic quote.
- The system must be in a **Continuous Trading** session
- Executions should happen on Venue MIC XLOX

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a Mid-Price 'Pegged' Quote, 'Order Sub Type' = '5' (Pegged) (Native) / 'Order Type' = 'P' (Pegged) (FIX). Native: Message Type = D (New Quote) Order Sub Type = 5 (Pegged) Pegged Exec Inst = 0 (Default), 1 (Permitted to execute), FIX: Message Type = S (New Quote) Quote Type = 1 (Electronic Quote) PeggedExecInst = 0 (Default), 1 (Permitted to execute), 2 (	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled) Pegged exec = 1 (Permissioned for execution against mid-priced pegged orders) Last Market = 25 (XLOX On Exchange Mid-Priced Executions)

Test Case: Non-Pegged Orders - PeggedExecInst set to execute against any order type including Pegged Orders

Objective:

Validate behaviour of **PeggedExecInst = 1 (Permitted to execute against mid-priced pegged orders)** during **Continuous Trading** when a Non-Pegged Order is submitted. s

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session
- Pegged Orders should be available in the system
- Executions should happen on Venue MIC XLOX

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a Non-Pegged order. Native: Message Type = D (New Order) Order Sub Type = 0 (Order) Price = Price of Mid-Priced Pegged Order/ Market Order Pegged Exec Inst = 1 (Permitted to execute against mid-priced pegged orders) FIX: Message Type = D (New Order) Order Type = 1 (Market), 2 (Limit) Price = Price of Mid-Priced Pegged Order/ Market Order PeggedExecInst = 1 (Permitted to execute against mid-priced pegged orders)	Step 2: Server sends an 'Execution Report' confirming that the order has been executed. Message Type = 8 (Execution Report) Execution Type = F (Trade) LastMkt = XLOX On Exchange Mid-Priced Executions Order Status = 1 (Partially filled) or 2 (Filled) PeggedsExecInst = 1 Permissioned for execution against mid-priced pegged orders

Test Case: Ability to flag an order as Retail Order

Objective:

Validate the behaviour of **Retail Order** submitted during a **Continuous Trading** session.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.
- Ensure **matching orders exist** in the order book for execution (If there is no match for the order in the order book, please create a matching order).
- Only member firm who have signed up to the "Retail Order Book Trading Scheme" AND is submitting an order that is eligible for that scheme can use this functionality.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a Retail Liquidity Provider order Native: Message Type = D (New Order) Owner Type = 1 (Retail investor) FIX: Message Type = D (New Order) OwnerType = 1 (Retail investor)	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled) Owner Type = 1 (Retail Investor) or 2 (Retail Liquidity Provider) – It should match the same value as what was put on the order.

Test Case: Ability to flag an order as Retail Liquidity Provider

Objective:

Validate the behaviour of **Retail Liquidity Provider Order** submitted during a **Continuous Trading** session.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits a Retail Liquidity Provider order Native: Message Type = D (New Order) Only permitted for orders that are; Mid-priced Pegged Orders, Mid-priced Pegged Limit orders, Hidden Limit Orders or Orders with IOC or FOK TIF Owner Type = 2 (Retail Liquidity Provider) FIX: Message Type = D (New Order) Only permitted for orders that are; Mid-priced Pegged Orders, Mid-priced Pegged Limit orders, Hidden Limit Orders or Orders with IOC or FOK TIF OwnerType= = 2 (Retail Liquidity Provider)	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted/executed. Message Type = 8 (Execution Report) Execution Type = 0 (New) / F (Trade) Order Status = 0 (New) / 1 (Partially filled) or 2 (Filled)

Test Case ID: Auction Volume Discovery - AVD / AVC Orders

Objective:

Validate behaviour of **AVD / AVC orders** during **Auction call or Continuous Trading**.

Preconditions:

- The client must establish a valid connection.
- The system must be in a **Continuous Trading** session

Comments:

A new auction event will be introduced to uncross at the closing auction price, immediately after a closing auction execution has successfully completed. Purpose of this test is to ensure an AVD / AVC order can be entered and accepted on the order book until the closing auction.

Test Steps & Expected Results

Client Action	Server Response
Step 1: Client submits an order (e.g. Limit, Market, Iceberg or hidden) with the TIF AVD or AVC (both should be submitted if member firm plans to use the functionality).	Step 2: Server sends an 'Execution Report' confirming that the order has been accepted. Message Type = 8 (Execution Report) Execution Type = 0 (New) Order Status = 0 (New)

6.0 Post-Trade Gateway Certification Scenarios

Test Case ID: MIT-200 Establish Logon for Post-Trade Gateway

Objective:

Establish a connection and log on.

Preconditions:

- The client must establish a valid connection to the **Post-Trade Gateway**.
- Ensure the correct IP address/Port are targeted.

Client Action	Server Response
Step 1: Establish a connection and log on to the post-trade gateway	Step 2: Logon acknowledgment confirming a successful connection to the post-trade gateway

Test Case ID: MIT-201- Own Trade Book Download

Objective:

Validate the ability to request and receive an Own Trades Book Download using the **Post-Trade Gateway**.

Preconditions:

- The client must establish a valid connection to the **Post-Trade Gateway**.
- Ensure the correct IP address/Port are targeted.

Client Action	Server Response
Step 1: Send a trade capture report request. <i>Note: Please provide TradeRequestID in the certification report.</i>	Step 2: Receive a trade capture report request ack.
(No client action in this step)	Step 3: Receive trade capture report(s) as requested.

7.0 Drop Copy Gateway Certification Scenarios

Test Case ID: MIT-300 - Establish Logon for Drop Copy Gateway

Objective:

Establish a connection and log on.

Preconditions:

- The client must establish a valid connection to the **Drop Copy Gateway**.
- Ensure the correct IP address/Port are targeted.

Client Action	Server Response
Step 1: Establish a connection and log on to the Drop Copy gateway	Step 2: Logon acknowledgment confirming a successful connection to the Drop Copy gateway

Test Case ID: MIT-301 - Own Order Book Download

Objective:

Validate the ability to request and receive an Own Order Book Download using the **Drop Copy Gateway**.

Preconditions:

- The client must establish a valid connection to the **Drop Copy Gateway**.
- Ensure the correct IP address/Port are targeted.

Client Action	Server Response
Step 1: Send a mass order status request <i>Note: Please provide MassStatus ReqID and the trader group in the certification report.</i>	Step 2: Receive Execution Report for each active order for the specified trader group..

8.0 Mandatory Scenarios for New Software Applications

Test Case ID: MIT-400 Error Handling on Logon after Suspension

Objective:

Validate the logon behaviour after suspension / un-suspension of User ID.

Preconditions:

- **Mandatory for new software.**
- This test case must be completed (where applicable) for the services listed below:
 - FIX gateway
 - Native gateway
 - Native recovery
 - Drop Copy gateway
 - Post-trade gateway
 - Group ticker plant replay service (any one channel)
 - Group ticker plant recovery service (any one channel)
- Ensure the correct IP address/Port are targeted for each gateway.
- Ensure the Comp ID / User ID is logged off.

Client Action	Server Response
Step 1: Request LSEG to suspend the CompID.	Step 2: CompID is suspended.
Step 3: Establish a connection and log on to the applicable gateway(s). Customer to verify the reject message.	Step 4: Logon is rejected. Customer application should automatically stop trying to log on and display the reason the logon was unsuccessful.
Step 5: Verify application behaviour conforms to the connectivity policy as outlined in the technical specification.	Step 6: Customer application attempts to connect a maximum of three times to the primary gateway, with a minimum timeout value of three seconds between attempts before attempting to connect to the secondary gateway – and this should be retried a maximum of a further three times.
Step 7: Request LSEG to unsuspend CompID.	Step 8: CompID unsuspended.
Step 9: Establish a connection and log on to gateway.	Step 10: Successful logon.

Test Case ID: MIT-500 Password Change

Objective:

Validate the behaviour of changing the logon password from the system default.

Preconditions:

- **Mandatory for new software.**
- This test case must be completed (where applicable) for the services listed below:
 - FIX gateway
 - Native gateway
 - Drop Copy gateway
 - Post-trade gateway
- Ensure the correct IP address/Port are targeted for each gateway.

Client Action	Server Response
Step 1: Establish a connection and log on to the relevant gateway(s). In your logon message, specify a new password of at least eight characters in length including one integer. <i>(Please provide details of ComplID(s) and gateway(s) tested in the comments section of the certification report.)</i>	Step 2: Successful logon, with current default password and a valid new password specified.
Step 3: Log out and disconnect.	Step 4: Logout message sent by client and connection terminated.
Step 5: Log on using the new password.	Step 6: Successful logon, using the new password specified in step one.

Test Case ID: MIT-501 Heartbeat Exchange

Objective:

Validate behaviour of the client and the server exchanging heartbeat messages during the period of client's inactivity.

Preconditions:

- **Mandatory for new FIX software only.**
- Customer application can suspend sending all messages except heartbeats. Heartbeats are only sent during periods of inactivity.
- The client must establish a valid connection to the **FIX Gateway**.
- Ensure the correct IP address/Port are targeted.
- Document the start and end time of test execution in the certification report.

Client Action	Server Response
Step 1: Establish a connection and log on to the gateway.	Step 2: Successful logon.
Step 3: Client refrains from sending messages. Stay inactive, i.e., make sure that there are no client initiated messages sent to the server. Make sure there are no unexpected disconnects within a reasonable timeframe, i.e., a few minutes.	Step 4: Logout message sent by client and connection terminated.
Step 5: Log on using the new password.	Step 6: Successful logon, using the new password specified in step one.

Test Case ID: MIT-502 Sequence Reset

Objective:

Validate the behaviour of a FIX message sequence reset.

Preconditions:

- **Mandatory for new FIX software only.**
- The client must establish a valid connection to the **FIX Gateway**.
- Ensure the correct IP address/Port are targeted.
- Document the start and end time of test execution in the certification report.

Client Action	Server Response
Step 1: Establish a connection and log on to the gateway.	Step 2: Successful logon.
Step 3: Make sure that the client and server are exchanging heartbeat messages. Send a sequence reset message MsgType=4. Make sure that the value of the NewSeqNo tag is greater than the sequence number of the latest incoming heartbeat message.	Step 4: Server is sending heartbeat messages to the client. Client is responding to heartbeat messages sent by the server. Gateway resets messages sequence number to the value specified in NewSeqNo tag sent by the client.

Test Case ID: MIT-503 Sequence Reset at Logon

Objective:

Validate the behaviour of FIX messages sequence reset initiated by the client at logon.

Preconditions:

- **Mandatory for new FIX software only.**
- The client must establish a valid connection to the **FIX Gateway**.
- Ensure the correct IP address/Port are targeted.
- Document the start and end time of test execution in the certification report.

Client Action	Server Response
Step 1: Establish a connection and log on to the gateway.	Step 2: Successful logon.
Step 3: Make sure that the client and server are exchanging heartbeat messages and inbound and outbound MsgSeqNum is bigger than 10.	Step 4: Server is sending heartbeat messages to the client. Client is sending heartbeat messages to the server.
Step 5: Disconnect from the server.	Step 6: FIX session is terminated.
Step 7: Client to set the next expected inbound and outbound MsgSeqNumbers to 1. Send a Logon message with ResetSeqNumFlag tag set to "Y".	Step 8: Server is responding with a logon message with the ResetSeqNumFlag tag set to "Y". Messages sequence is reset to "1". Server and client sending heartbeat messages.

Test Case ID: MIT-504 Resend Request

Objective:

Validate the behaviour of FIX messages sequence reset initiated by the client at logon.

Preconditions:

- **Mandatory for new FIX software only.**
- The client must establish a valid connection to the **FIX Gateway**.
- Ensure the correct IP address/Port are targeted.
- Document the start and end time of test execution in the certification report.

Client Action	Server Response
Step 1: Establish a connection and log on to the gateway.	Step 2: Successful logon.
Step 3: Submit an order that results in a fully filled trade. Note down the message sequence number received for this execution report.	Step 4 The trade execution report received.
Step 5: Disconnect from the server.	Step 6: FIX session is terminated.
Step 7: Client to set the next expected inbound and outbound MsgSeqNumbers to 1. Send a Logon message with ResetSeqNumFlag tag set to "Y".	Step 8: Server is responding with a logon message with the ResetSeqNumFlag tag set to "Y". Messages sequence is reset to "1". Server and client sending heartbeat messages.



**LONDON
STOCK
EXCHANGE**