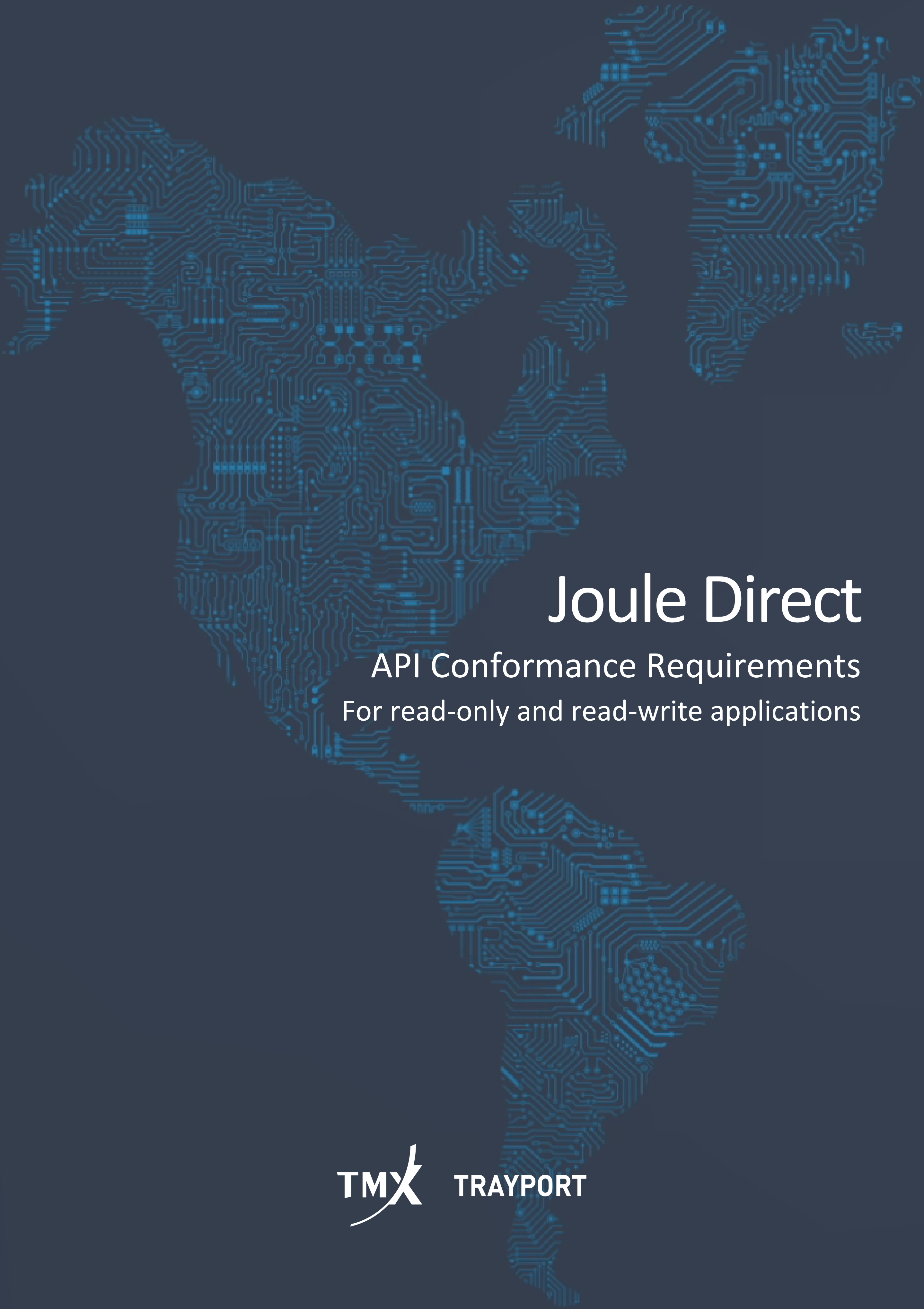


Joule Direct

API Conformance Requirements

For read-only and read-write applications

LEGAL NOTICE

CONTENTS

Introduction

Trayport Conformance Testing ensures that third-party or in-house applications are correctly interacting with the Trayport software and platforms. Testing is carried out for all major upgrades of external applications, with the primary aim of ensuring users receive the highest quality experience and the performance, reliability and security integrity of the Trayport environment is not put at risk. A major upgrade may consist of any fundamental changes in how the application is developed, changes in behaviour or functionality.

Upon successful completion of the test, Trayport will provide a Conformance Test Certificate Licence (CTCL) detailing the outcome of the test and the billable users. The CTCL must be signed before Joule Direct Production credentials are provided.

Conformance testing is not required for certified software partner (CSP) applications as the application is already known and approved by Trayport. You can find a list of our CSPs on our website here.

Benefits of Conformance Testing

- Better serve market participants by ensuring optimal connectivity of external applications with Trayport software.
- Provide market participants with assurance and confidence that conforming applications behave as expected and perform functions in a known manner.
- Consistency of standards across all applications, while all new versions are tested to ensure the highest standards are maintained.
- Reduce risk and cost by safeguarding the integrity (stability, security and reliability) of the Trayport environment for the benefit of all our clients.

Conformance Test Format

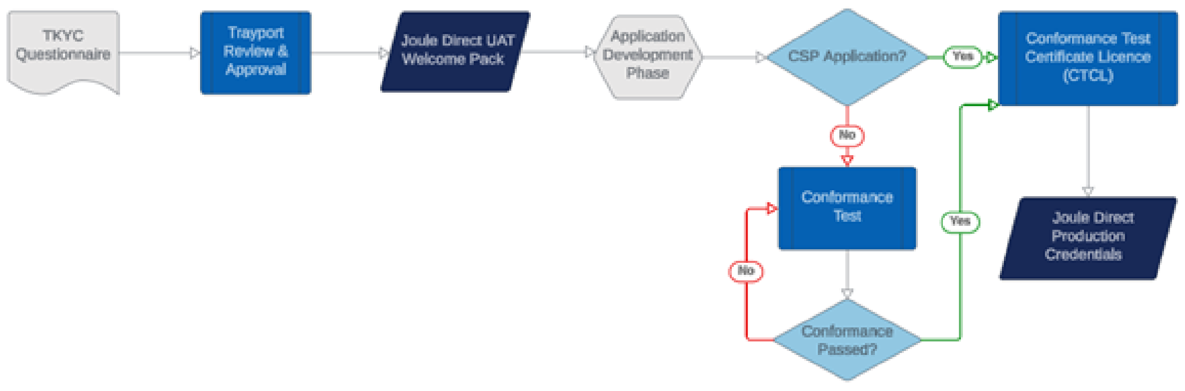
The Trayport conformance test is designed to be thorough but time efficient, thus reducing client overheads. Testing is carried out within the Joule Direct UAT environment for all new and major upgrades for applications connecting to the Trayport API.

The conformance test process typically takes up to 30 minutes to complete for each client application.

The test will be performed at a pre-agreed date and time with Trayport staff. The participant will be instructed to execute specified certification test cases at this time and Trayport will later verify that the relevant application fulfils the expected requirements and behaviour.

It is imperative that the development guidelines detailed in the 'Developer Best Practice Checklist' and 'Requirements' sections within this document are implemented from the very beginning of your development stage to ensure the application passes the conformance test. Your application will not be enabled in the Joule Direct Production environment until the conformance test is passed and the Conformance Test Certificate Licence is signed.

API Onboarding Workflow


TKYC Questionnaire
Trayport Review & Approval
Joule Direct UAT Welcome Pack
Application Development Phase
CSP Application?
Yes
No
Conformance Test Certificate Licence (CTCL)
Conformance Test
Yes
No
Joule Direct Production Credentials
Conformance Passed?

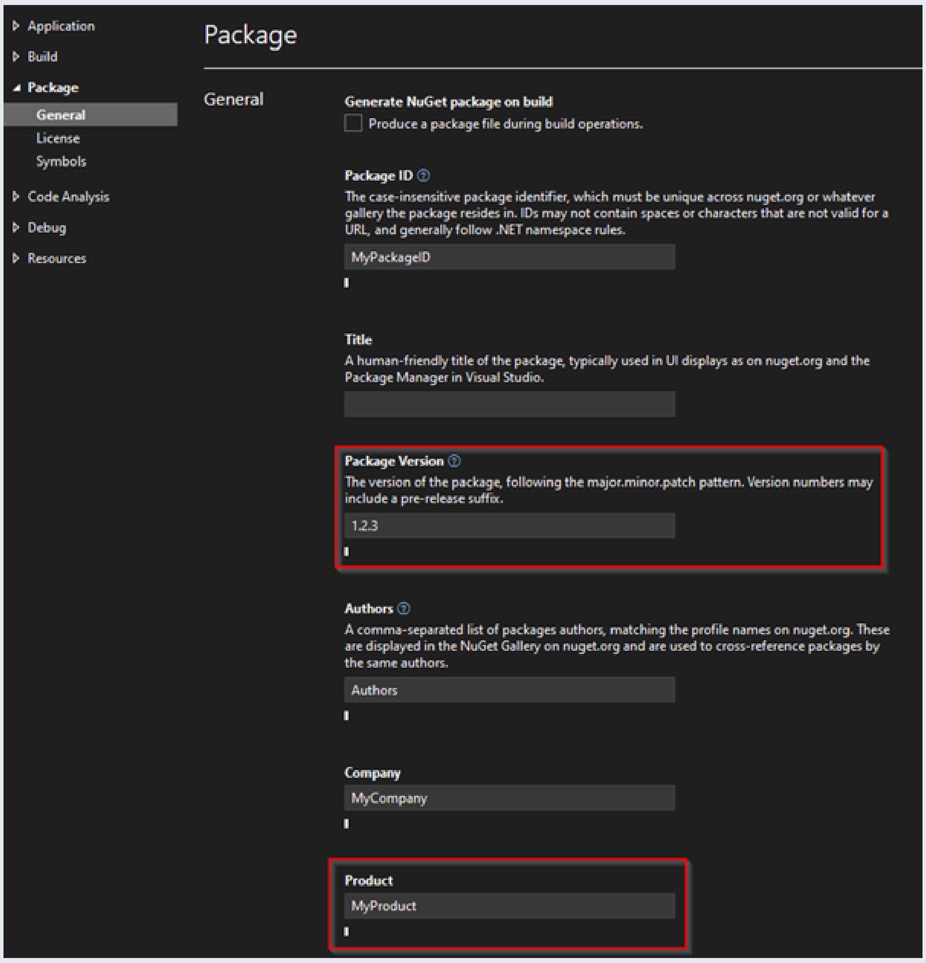
Developer Best Practice

Below is a list of the recommendations and considerations you should make during your development phase. This list must be completed prior to Trayport Support being able to provide in-depth assistance with investigations and before scheduling the Conformance Test.

Any failure to do so could result in inaccurate and unexpected behaviour in your application.

Activity	Description
API Version	The API version must be greater than 2.135.0.14685. You must continue to regularly upgrade your API version, annually as a minimum. This ensures access to the newest functionality and lowest latency performance. Trayport continually makes improvements to its software and you are encouraged to take advantage of these. For the end-of-life dates for the API, our other products, and previous versions, please refer to the Lifecycle Support section on our website.
API Lifecycle	The API maintains an internal copy of all deals that have occurred since it was first connected to Joule Direct. Any deals queried from before this time are also held within the API in order to make subsequent queries faster and remove the need to contact Joule Direct for the data. Because of this, it is recommended that you gracefully close the connection and create a new instance of the API at least once a week. The API should establish a long-lived connection to Joule Direct, rather than reconnecting every time a query is needed. Data retrieved from Events (for example, subscription results into OnNewData) should be handled asynchronously to the thread which owns the API instance so that the event handler finishes execution as soon as possible. Not doing this may result in missed events or COM exceptions being thrown.

Activity	Description
Connectivity	When connecting to Trayport's infrastructure, you must use the following aliases: • jouledirecttest - UAT environment for testing new implementations. • jouledirect – PROD environment for conformance-tested applications only. Further connectivity information can be found in the Joule Direct Connectivity Guide. Before building your application, be sure to read all documentation and understand the available configurable **Connection Options**. These options will impact how your application functions, and include critical options such as whether or not your application can view Implied Pricing and what actions you want the system to take in the event of your application disconnecting from Joule Direct. Ensure your application clearly displays connection status to all end users, as confirming an active session is one of the first steps towards troubleshooting.
Security	Login Names and Passwords must be stored in a secure manner.

Activity	Description
Application Name	Applications using the Joule Direct API must have an identifiable Application Name. The Application Name must be visible in Joule Direct and align with the Application Name in your **Technical Know Your Client (TKYC)** questionnaire. The exact method to set the name depends on the language and frameworks used to develop the application. In most .NET applications, the Application Name is controlled by the 'Assembly Name' configured for the application. Refer to the 'Assembly Name' field within Project Properties in Visual Studio. Alternatively, in older versions of Visual Studio, the ApplicationName can be defined directly in the `AssemblyProduct` attribute of the `AssemblyInfo.cs` file. **Note:** The `AssemblyInfo.cs` file may have a different file extension depending on the chosen language. For the Joule Direct .NET API, this can be set by configuring the Product and Package Version fields under the Package settings for the project in Visual Studio:  Application Build Package General License Symbols Code Analysis Debug Resources Package General **Generate NuGet package on build** ☐ Produce a package file during build operations. **Package ID** The case-insensitive package identifier, which must be unique across nuget.org or whatever gallery the package resides in. IDs may not contain spaces or characters that are not valid for a URL, and generally follow .NET namespace rules. MyPackageID **Title** A human-friendly title of the package, typically used in UI displays as on nuget.org and the Package Manager in Visual Studio. **Package Version** The version of the package, following the major.minor.patch pattern. Version numbers may include a pre-release suffix. 1.2.3 **Authors** A comma-separated list of packages authors, matching the profile names on nuget.org. These are displayed in the NuGet Gallery on nuget.org and are used to cross-reference packages by the same authors. Authors **Company** MyCompany **Product** MyProduct For the Joule Direct .NET API gRPC server interface, the Application Name value must be defined in the **`application_id`** and

Activity	Description
	application_version fields in the LoginRequest call made to gRPC.
Market Activity	The markets you follow may have the tendency to be volatile and have spikes in activity. As such, it is critical to ensure that your application is able to handle a high number of actions per second. For Read-Write applications, it is extremely important that you add throttling to your API application so as to not hit our established limits and become blocked from performing actions. Your application should not exceed 50 upstream actions per second.

Activity	Description
Queries vs. Subscriptions	Use Queries only when necessary for the best experience: • Queries should be used to provide you with the latest state of the market for the instruments you are interested in. Upon reconnecting, your application should: o Utilize order book snapshots to ensure all prices are up-to-date. o Utilize trades snapshots to ensure no trades are missed. Subscriptions provide ongoing real- time updates as soon as new data is available in the API and should generally be used much more frequently than queries. Users of the .NET API also have the ability to use combined SubscribeAndQuery functionality. When configuring queries and subscriptions for your API application, Trayport recommends the following best practices: • Wherever possible, subscriptions should be used to retrieve data from Joule Direct. • Only data that is within the deal cache on Joule Direct is accessible to the API (120,000 records). If you need to regularly query data beyond this history, you should retain the results of a subscription in your own data store. • You cannot query for historic orders. The results of a subscription must be retained by your own application. Trayport Support may be able to produce a one-off historic version of this for you from historic log files if they are available. Contact Trayport support on +44(0) 20 7960 5555 or support@trayport.com if you require this information. **Note:** Retaining historic order or trade data may be subject to licensing requirements from either Trayport or the venue the data originated from. Contact your Trayport Client Relationship Manager to seek further guidance for your specific implementation. • Static information, such as the data used within the <INSTSPECIFIER> should be retained within your application rather than re-queried for every order or trade. A new API connection should not be established to query this information; this causes considerable network utilization due to the download upon login to Joule Direct. • If you require the full lifecycle of information for a trade, you must set-up a subscription rather than polling via queries, even

Activity	Description
	if the update suppression mode is disabled.
General Tips	Where possible, it is highly recommended to use IDs. Instrument, Broker & Company names can change, IDs are unique and will not change without advance warning from Trayport.
gRPC	The implementation of the gRPC wrapper in conjunction with the JD API has had the benefit of opening up more opportunities to leverage the languages that suit our clients best. However, this has also opened up a huge potential for languages and setups that Trayport has not dealt with before. Whilst Trayport aims to be able to support our clients as much as possible, it is not always going to be feasible for us to do so when it comes to implementation beyond what we've dealt with in the past. Please leverage Trayport Support for anything regarding the structure or behaviour of the JD API and/or how it interacts with the gRPC server, but we will not be able to directly support anything outside this scope.
Common Order & Trade Workflows	The most common workflow is that a user inserts an order and when this is dealt a subsequent trade is created. When this happens, order and trade records will be created accordingly. If a broker inserts a trade manually on-behalf of your company, they will set flags such as ManualDeal and VoiceDeal to true. It is important that your application is able to monitor the values of these attributes. We are aware that if a trade update message is received for a trade ID (that was previously deleted) where the `IsDeleted` flag is set to false, that this means the trade has been restored and is no longer deleted. A full list of attributes on order and trade records can be found in the API guide. It is important to note that Trayport aggregates various brokers and exchanges, therefore it is possible that some workflows may differ from venue to venue. Examples of this can be manual/voice deals, block trades and sleeve workflows. For full clarification, you may need to reach out to the venues to discuss further.

Activity	Description
Executing Trades	**General** • Avoid sending repeat commands for the same action. For example, sending trade order multiple times for the same ID will only ever be possible once. • Some venues may only allow execution of the best price available to you. In this case, you may execute a different order to the specific one you request. **Auto-matched and Non-Trayport Exchange Markets** • Perform the minimum actions possible to complete your desired outcome. For example, if there are several prices you wish to hit, send a single command for the entire quantity rather than a separate command for each order. Joule Direct handles executions sequentially, dependant on the broker or exchange system, blocking until each completes. Because of the round trip time from Joule Direct to the exchange your actions will be seen by other market participants before you have finished sending the commands. • In general, exchanges send aggregated volume; this often comes from implied orders where the underlying legs are not visible to you. Whilst exchanges may guarantee that the transaction will complete, there is a risk of accidentally trading through your own position unless self-trade protection is enabled on the exchange. Contact the exchange to request this feature be enabled if appropriate. • Consider building specific logic for each exchange's order types. Joule Direct converts a command to hit an order into the most appropriate exchange action to gain the required price and volume. For example: ICE allows a Fill or Kill, but EEX only allows Fill and Kill, which may result in an unexpected volume depending on your desired outcome.

Read-Write Application Requirements

In addition to the requirements detailed in the Read-Only Application Requirements section, writable application requirements are as follows:

If the application is to be able to manage orders, it must be able to:

- Enter orders
- Update the broker, price and quantity of an order
- Delete orders
- Enter an order that would create a choice market
- Enter an order that would cross the market
- Enter an order with a negative price
- Enter an order with a decimal quantity
- Enter an order with a hidden quantity
- Update, delete, withhold or firm orders inserted by other users at the same company
- Handle its own orders being updated, withheld or removed by other users at the same company
- Handle its own orders being fully or partially aggressed
- Handle an order being instantly aggressed, updated or removed upon insertion
- Handle the following errors when attempting an order action:
  - Insufficient instrument permissions at the venue
  - Choice market not allowed
  - Order would create an invalid spread
  - Venue disconnected.

If the application is to be able to deal orders, it must be able to:

- Deal a specific order
- Deal a given volume (if this feature is required)
- Deal a given volume across multiple venues (if this feature is required)
- Partially deal a specific order.

In addition to the above, all writable applications must:

- Display their connection status to the end user
- Reconnect automatically if disconnected (this is usually done automatically by the API)
- Stop attempting to reconnect if the account is disabled
- Be able to handle changes to orders and trades that occurred while the application is disconnected
- Throttle actions to 50/sec or 1000/min

- Act in line with the regulatory requirements of the connected venues
- Permission is granted from all venues upon which actions will be executed via the API. For Market Data Entitlements to be enabled by Trayport Support for your application to view ICE and CME markets:
  - ICE - written permission from ICE Support is required.
  - CME - you need to request approval from CME's GAM team at gamemea@cmegroup.com.
- Store passwords securely
- Use the subscription method rather than repeated polling if real-time data access is required
- Have protection against feedback loops if using algorithmic trading.

Read-Only Application Requirements

If the application is to view market orders, it must be able to:

- Handle an empty order book
- View prices on both sides of the market
- Properly display price and quantity updates to own and market orders
- Handle the removal of prices from an order book
- Handle changes that occur to an order book while the application is disconnected
- Handle choice markets
- Handle crossed markets
- Handle multiple prices on the same level
- Handle negative prices
- Handle decimal quantities
- Handle venue implied prices
- Handle prices from multiple venues for the same contract
- Display the last traded prices correctly.

If the application is to view private orders, it must also be able to:

- Handle aggression (full or partial) of private orders
- Handle the insertion, update or removal of private orders by the same or other accounts at the same company
- Handle private orders with negative prices
- Handle private orders with decimal quantities.

If the application is to view market trades, it must be able to:

- Handle the insertion, update, deletion and restoration of trades
- Handle manual trades
- Handle voice trades
- Handle trades that are not linked to any order but are not marked as manual trades.

If the application is to view private trades, it must also be able to:

- Display the counterparties to trades where provided
- Handle private trades that have not been provided with counterparty information.

In addition to the above, all read-only applications must:

- Display their connection status to the end user
- Reconnect automatically if disconnected (this is usually done automatically by the API)
- Stop attempting to reconnect if the account is disabled

- Be able to handle changes to orders and trades that occurred while the application is disconnected
- Act in line with the regulatory requirements of the connected venues
- Store passwords securely
- Use the subscription method rather than repeated polling/querying of real-time data
- Have protection against feedback loops if using algorithmic trading