Thermo SCIENTIFIC

Real-Time Instrument Control of the Orbitrap Tribrid Mass Spectrometer

Derek J. Bailey, Florian Grosse-Coosmann, Manish Doshi, Qingyu Song, Jesse D. Canterbury, Qiming Wan, and Michael W. Senko, Thermo Fisher Scientific, 355 River Oaks Parkway, San Jose, CA 95134

OVERVIEW

Presented here is an instrument application programming interface (IAPI) for real-time control of Thermo Scientific™ Orbitrap™ Tribrid™ mass spectrometers (MS). The IAPI provides a programmatic and interactive way to receive and send data between the MS and host computer system.

INTRODUCTION

In the past half-decade a growing emphasis has been placed on the ability to incorporate real-time data analysis and programmatic control during the mass spectrometer acquisition. To provide advanced, high-performance access for third-parties, we previously released an IAPI for the bench-top Orbitrap mass spectrometers (Exactive-series IAPI (ES-IAPI)). We have now enabled the Orbitrap Tribrid family of instruments to use a similar, expanded programming interface to extend real-time capabilities to researchers on additional platforms.

METHODS

The Tribrid IAPI is written for the Microsoft.NET Framework (Version 4.5.1) and is fully integrated with the Tune instrument control software (Version 2.2). It uses an event-driven architecture where users subscribe to instrument generated events throughout the acquisition (e.g., spectra, messages, device readbacks, etc.). Users are able to control the instrument in real-time by modifying instrument parameters or submitting custom scan definitions. The IAPI is efficient and does not impact the performance of the instrument. The Tribrid IAPI is a direct extension of the previous bench-top Orbitrap interface, but various improvements and refinements break binary compatibility. All example code and applications are written in C#.

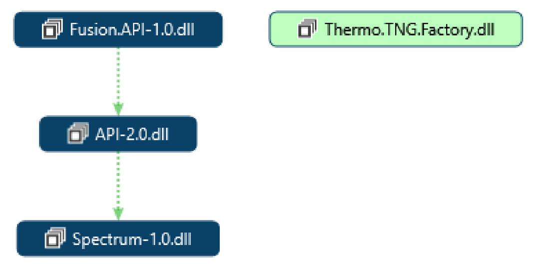
RESULTS

Figure 1. Compatible Mass Spectrometers

Orbitrap Fusion　　Orbitrap Fusion Lumos

The IAPI is compatible with both the Thermo Scientific™ Orbitrap™ Fusion™ mass spectrometer and the Thermo Scientific™ Orbitrap™ Fusion Lumos™ mass spectrometer

Figure 2. IAPI Architecture



- The architecture of the Tribrid IAPI (Figure 2.) is derived from the ES-IAPI with some slight modifications.
- We have separated the spectrum-related interfaces into a separate assembly (*Spectrum-1.0.dll*).
- The core API assembly (*API-2.0.dll*) was updated from the ES-IAPI and new features were added and others removed.
- A Fusion factory assembly (*Thermo.TNG.Factory.dll*) provides a clean way to create an instance of the IAPI main entry point (Figure 2).
- A new Fusion API assembly (*Fusion.API-1.0.dll*) derives from the core API and provides Fusion-specific extensions and implementations.

Figure 3. IAPI Instance Creation Example

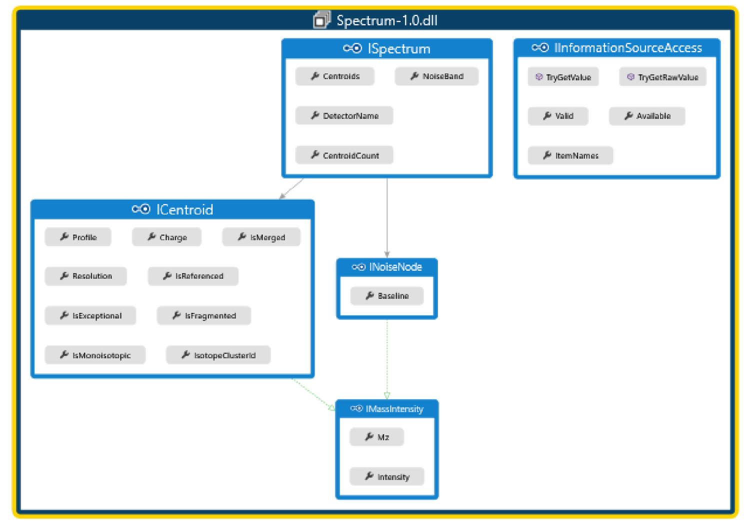
```
IFusionInstrumentAccessContainer iAccessContainer
    = Factory<IFusionInstrumentAccessContainer>.Create();
```

Creating an instance of the Tribrid IAPI is straightforward using the factory method pattern provided by the *Thermo.TNG.Factory.dll* assembly (Figure 3). There is no longer a need of using COM or manually searching the registry to create an instance from an explicitly loaded assembly. All other access to the IAPI can be obtained through the factory-generated *IFusionInstrumentAccessContainer* instance.

SPECTRUM ACCESS

The spectrum-related interfaces are broken into 5 separate parts as outlined in Figure 4.

Figure 4. IAPI Spectrum Interfaces



- **IMassIntensity**
  Represents a m/z-intensity pair
- **INoiseNode**
  Represents a noise baseline of a spectrum. It derives from *IMassIntensity*
- **ICentroid**
  Represents the instrument-centroided peak and associated data. If profile data is present, an array of *IMassIntensity* will represent those points. It derives from *IMassIntensity*.
- **ISpectrum**
  Represents a mass spectrum of *ICentroid* peaks and its associated polygonal noise-baseline. The mass analyzer that collected the data is indicated by the detector name (e.g., Ion Trap or Orbitrap).
- **IInformationSourceAccess**
  A dictionary-like container to store various metadata fields (e.g., Scan Trailer, Status Log, Tune Data, etc.) that are readily present in the Thermo raw data file. Provides access to both the string representation as well as the raw-values (.NET object) for each item.

Figure 5. Example Spectrum Access

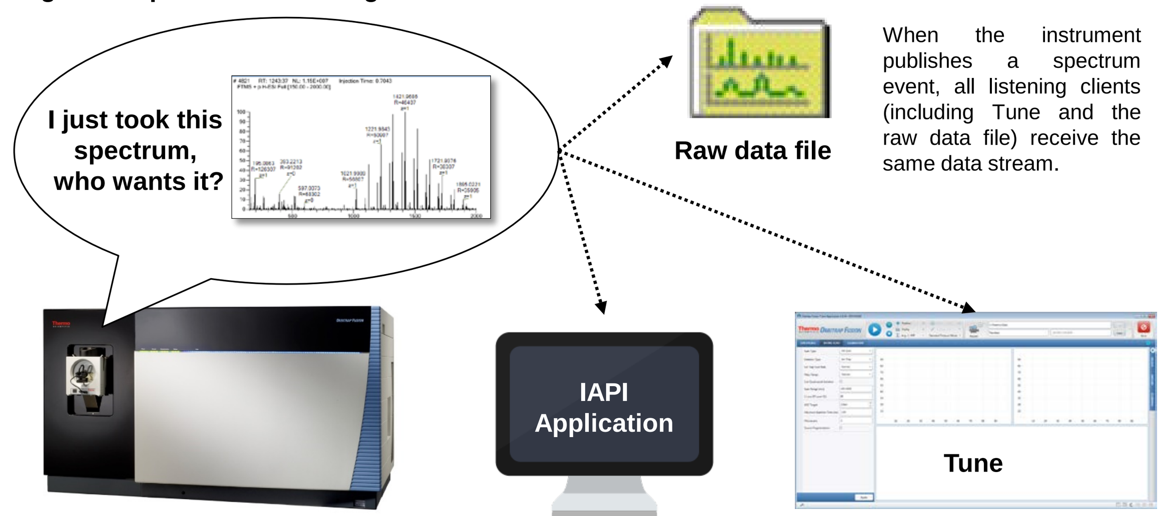
```
private void ExampleSpectrum(ISpectrum spectrum)
{
    // Check for emptiness
    if (!spectrum.CentroidCount.HasValue) return;

    // Loop over the spectrum centroids
    foreach(ICentroid centroid in spectrum.Centroids)
    {
        // Print some information about each peak to console
        Console.WriteLine("MZ: {0} Intensity: {1} Charge: {2}",
            centroid.Mz,
            centroid.Intensity,
            centroid.Charge.HasValue ? centroid.Charge.Value.ToString() : "?" );
    }
}
```

EVENTS

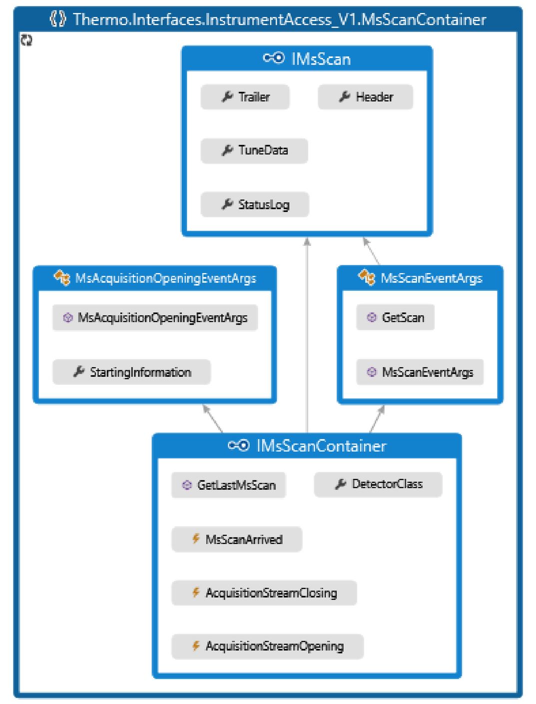
The IAPI uses an event-driven model for informing the user application of instrument-related events. When the instrument has finish collecting a spectrum, it publishes the event and listening clients can consume it (Figure 6). The same spectrum event the IAPI uses is what the Tune application and the raw data file internally subscribe to; this ensures the data displayed in Tune and stored in the raw data file is 100% identical to the one the IAPI user would receive.

Figure 6. Spectrum Receiving Interfaces



The spectrum receiving event for the IAPI is controlled through the *IMsScanContainer* interface (Figure 7.).

Figure 7. Spectrum Receiving Interfaces

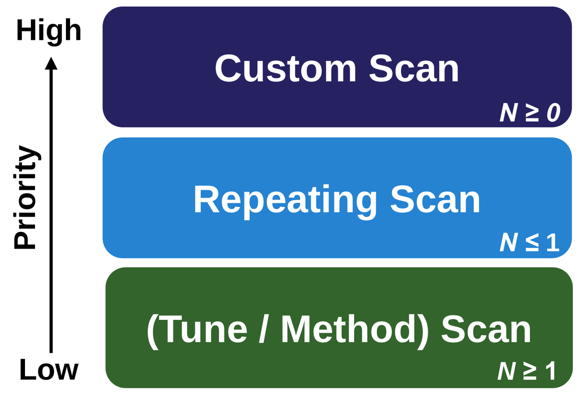


- **IMsScanContainer**
  This interface has 3 events, the most commonly subscribed event is the *MsScanArrived* event.
  - Whenever the instrument collects a spectrum, this event is raised with a method to get the generated *IMsScan*.
  - The IAPI also generates events when the acquisition stream starts or finishes being acquired into a raw data file.
- **IMsScan**
  Represents a spectrum with associated meta-data. The Header, Trailer, TuneData, and StatusLog are all *IInformationSourceAccess* objects. All the data available in the Thermo raw data file is accessible through this interface. It derives from *ISpectrum*.

Instrument Control

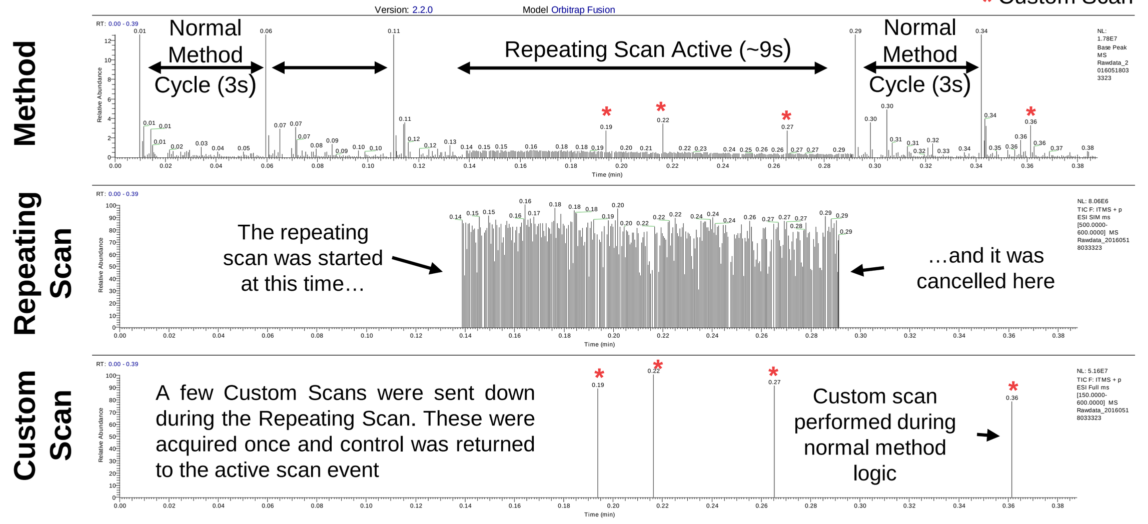
The IAPI allows users to submit scans, at will, for the instrument to perform. This capability is available all the time—when running a method or controlled by Tune. There are two types of user-submitted scans: **Repeating Scans** and **Custom Scans**.

Figure 8. Instrument Scan Priority Stack



- Custom and repeating scans can be explicitly cancelled via an IAPI call.
- Scans higher on the priority stack (Figure 8) will get executed by the instrument first, before ones residing below it.
- The Tune- and Method-define scans are always at the bottom of the stack and are not removable from the IAPI.
- Multiple custom scans can be defined at a given point. As soon as a custom scan is acquired by the instrument, it is immediately popped from the stack. Multiple custom scans are performed in FIFO manner.
- Only 1 repeating scan can be defined at a time. This scan does *not* get popped from the stack after being acquired.

Figure 9. Example IAPI scans During Method Acquisition



A top speed (3s) data-dependent method was acquired while an IAPI application sent both repeating and custom scans. The method logic was suspended until the IAPI scan events were completed and then resumed where it left off. IAPI scans inserted during method acquisition are placed in analysis pipeline so that they can be parallelized for optimal performance. This insertion may lead to a small delay between when a scan is submitted and its acquisition.

INSTRUMENT CONTROL

The IAPI offers a convenient way for specifying the parameters of a scan. After creating a default repeating or custom scan through an IAPI method call, simply specify the parameters in a *Dictionary<string,string>* Values property of the scan (Figure 10). The allowable keys and values ranges for the scans are provided by another IAPI property that is not shown here.

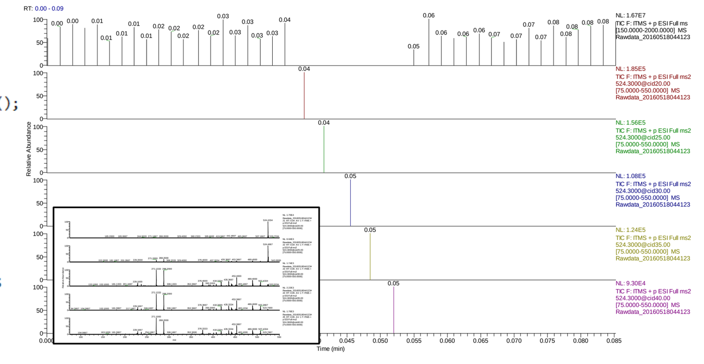
Figure 10. Example Energy Ramp of a Precursor Recorded from Tune

```
double startEnergy = 20;
double deltaEnergy = 5;

for (int i = 0; i < 5; i++)
{
    ICustomScan customScan = _scans.CreateCustomScan();

    customScan.Values["FirstMass"] = "75";
    customScan.Values["LastMass"] = "550";
    customScan.Values["ScanType"] = "MSn";
    customScan.Values["PrecursorMass"] = "524.3";
    customScan.Values["IsolationWidth"] = "2";
    customScan.Values["ChargeStates"] = "1";
    customScan.Values["CollisionEnergy"]
        = (startEnergy + deltaEnergy * i).ToString();

    _scans.SetCustomScan(customScan);
}
```

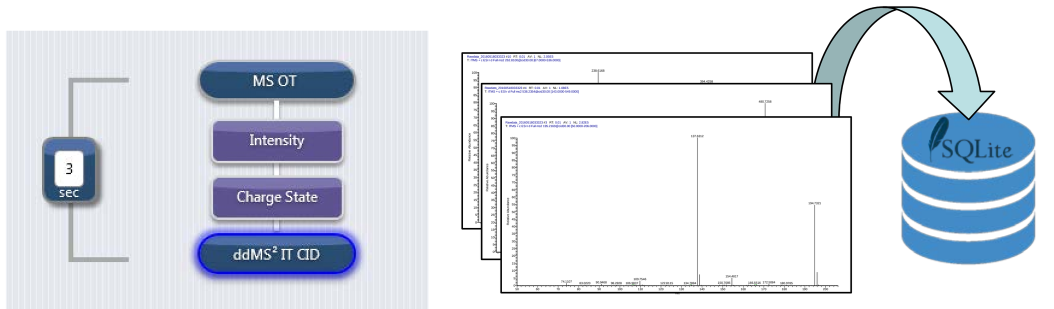


While acquiring spectra from Tune, the above code was executed from an IAPI application. Here 5 custom scans were created, targeting a peak at 524.3 m/z for MS/MS analysis. This peak was fragmented between 20 and 40 normalized collision energy (NCE) in increments of 5. These custom scans were immediately executed and saved in the acquiring raw data file. After these custom scans were acquired, the Tune-defined scan continued for the rest of the acquisition.

EXAMPLE IAPI APPLICATIONS

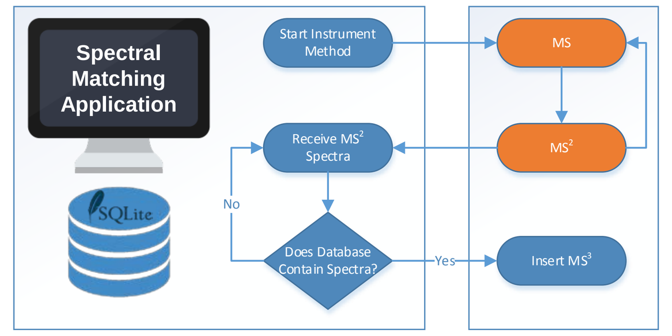
Spectral Matching in Real Time
The Tribrid Method Editor offers complex experimental design and real-time filtering of mass spectrum, but is limited to the built-in filters and execution workflow. The IAPI offers a way for researchers to extend the capabilities by implementing their own decision-making logic and control. Here we demonstrate real-time spectral matching to decide whether to take an MS3 scan.

Figure 11. Spectral Library Creation



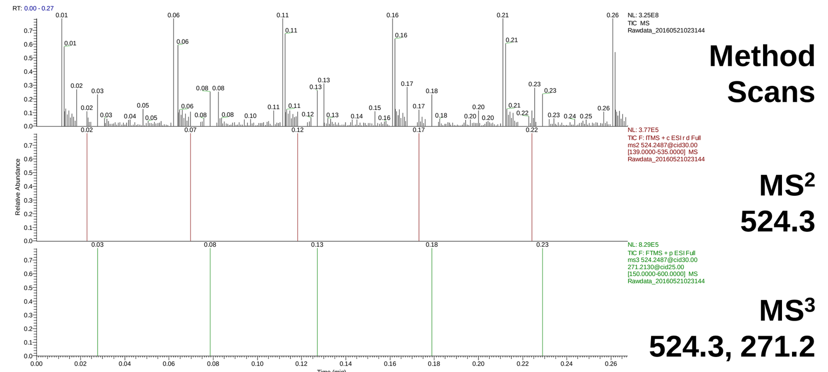
A standard MS/MS experiment was constructed in the Method Editor application and acquired on the Fusion mass spectrometer. Some of the resulting MS/MS spectra were extracted from the raw data file and imported in a local SQLite database. The spectra were binned into single Dalton bins and intensities were normalized from 0 to 100.

Figure 12. Spectral Matching Application



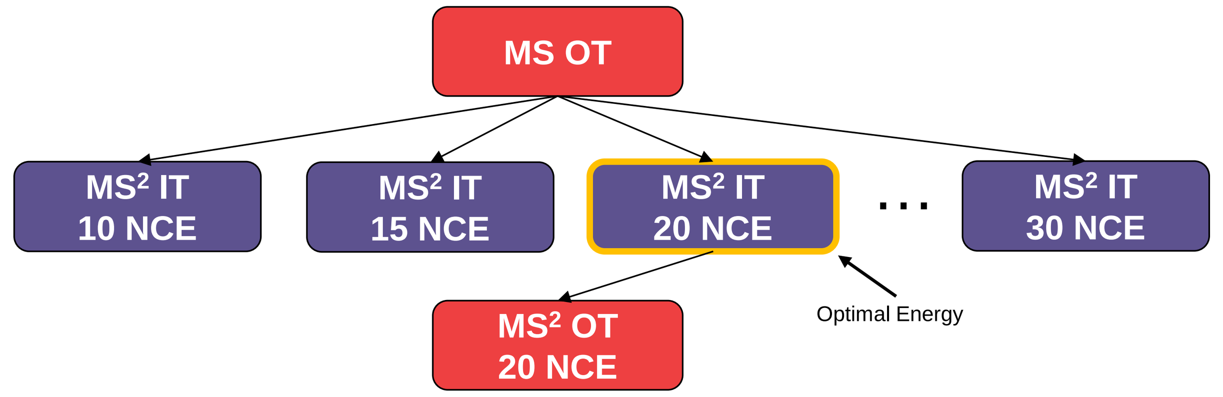
An IAPI application was written and listened to the MS/MS spectral stream of a executing method. When the application finds a match in the database, it triggered an MS^3 scan event of the most intense ion.

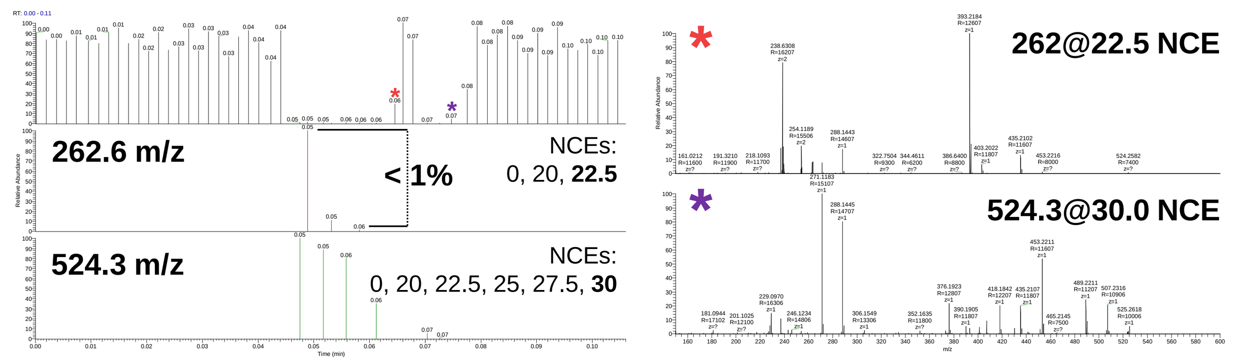
Every time a MS/MS was taken of 524.3, the resulting spectrum was matched against the database and it triggered a MS^3 scan on the 271.2 peak. The inserted scan was an Orbitrap MS^3 and had to wait a short amount of time for the analyzer to become available.



Real Time Compound Collision Energy Optimization
In typical data-dependent (DD) experiments, the MS/MS collision energy (CE) is set to a static value that performs well over a broad range of precursors. In order to better optimize the collision energy on a precursor-by-precursor basis, we wrote a simple IAPI application that performs a series of rapid ion trap scans over a CE range. The application then determined the optimal CE and subsequently performed a high resolution Orbitrap scan of the same precursor at the optimal CE (Figure 11.).

Figure 11. Compound Collision Energy Optimization





While acquiring spectra from Tune, both 262.6 and 524.3 m/z were targeted for MS/MS analysis. A series of rapid ion trap (IT) scans were performed on each precursor at increasing NCEs. Once the TIC of the of the IT survey scan was less than 1% of the TIC at 0 NCE, the optimization was stopped and a high resolution Orbitrap MS/MS analysis was queued at the optimal NCE. Both of these compounds were optimized in tandem and arrived at different optimal NCEs. The 262 precursor broke apart quicker than the 524 precursor.

CONCLUSIONS

The Instrument Application Programming Interface (IAPI) for the Orbitrap Tribrid Mass Spectrometer family provides advanced control and acquisition to enable custom experimentation and method development for mass spectrometry. This increased flexibility the IAPI provides users has the potential to spawn creative new experimental design and experiments not possible with the standard software.

The IAPI offers a lot of functionality not mentioned in this document. For complete listing of the capabilities, as well as all the example applications shown, please visit our Thermo Fisher Scientific Life Science Mass Spectrometry GitHub page: https://github.com/thermofisherlsms/iapi

ACKNOWLEDGEMENTS

We thank Phil Remes for his suggestions and technical support implementing the IAPI on the Tribrid system. We also thank Balaram Barange for contributing to the example programs.

TRADEMARKS/LICENSING

Access to the API is not part of the standard software delivered with the instrument and requires a special license agreement. Contact derek.bailey@thermofisher.com for details.

PO64748-EN 0516S