Expanded Instrument Control Using the Orbitrap Tribrid IAPI

Jesse Canterbury, William Barshop, Graeme McAlister, and Shannon Eliuk, Thermo Fisher Scientific, San Jose CA 95134

ABSTRACT

Purpose: We have developed and expanded upon the Tribrid IAPI to provide greater flexibility and increased functionality to IAPI users.

Methods: Newly exposed functionality in this version of the IAPI was verified using experiments on an Orbitrap Tribrid mass spectrometer, with FlexMix providing analytes of interest.

Results: Expanded functionality includes access to a few new scan modes, as well as the ability to control the predictive automated gain control (pAGC) process. *Note, a patch is required to use some of this expanded functionality. See the note at the end of the poster.*

INTRODUCTION

Several years ago [1], we introduced an instrument application programmer's interface (IAPI). The IAPI allows users of Thermo Scientific™ Orbitrap™ Tribrid™ instruments access to the standard scan functionality, enabling them to build custom data acquisition schemes. While useful for many applications, this initial implementation nonetheless lacked a few important capabilities that have become essential as user needs have expanded. This work describes some of these expanded capabilities, which include access to mass range mode, control of injection time for multiplexed acquisitions, and limited control of predictive automatic gain control (pAGC) for custom scans.

MATERIALS AND METHODS

Instrumentation

Data were acquired on a software-modified Thermo Scientific™ Orbitrap Fusion™ Lumos™ Tribrid™ mass spectrometer. FlexMix analytes were delivered via microflow infusion (flow rate 3-5 uL/min) and the heated electrospray (HESI) emitter.

Software

A custom application was written in C#, using a new version of the instrument application programmer's interface (IAPI), and the new scans modes were exercised.

Architecture and Data & Control Flow

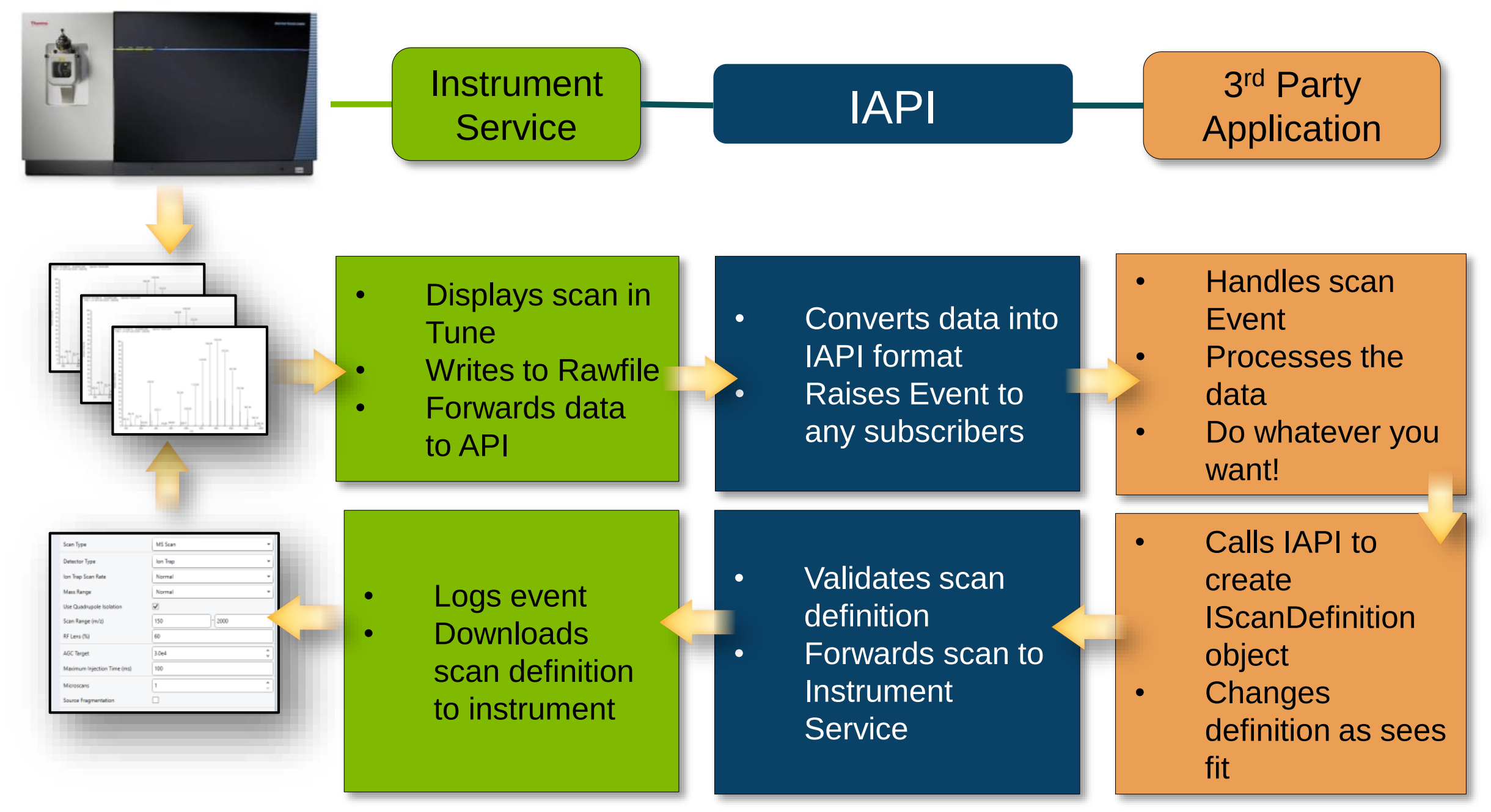

Instrument Service
IAPI
3rd Party Application
Displays scan in Tune
Writes to Rawfile
Forwards data to API
Converts data into IAPI format
Raises Event to any subscribers
Handles scan Event
Processes the data
Do whatever you want!
Calls IAPI to create IScanDefinition object
Changes definition as sees fit
Validates scan definition
Forwards scan to Instrument Service
Logs event
Downloads scan definition to instrument

Figure 1. The IAPI assemblies provide an event-driven interface to instrument functionality that can be exploited in user applications to design custom data acquisition strategies. The data flow proceeds as follows: (1) the Instrument Service handles tasks like displaying data and writing to raw files, in addition to forwarding data to the IAPI; (2) the IAPI converts data into the IAPI format and raises events that users' applications can subscribe to; (3) user application processes data, handles events, and generates new scan definitions; (4) user application can send scan definitions and other data to the IAPI, which (5) validates these data and sends them to the Instrument Service, which then (6) sends scan data and other data to the instrument firmware.

IAPI Overview

Scan Management

A central concept for use of the IAPI is the hierarchical system for scan management. It is assumed that the instrument is always scanning, and controlled either by Tune (i.e., "direct control") or by a Method acquisition. The figure below explains the hierarchy in detail.

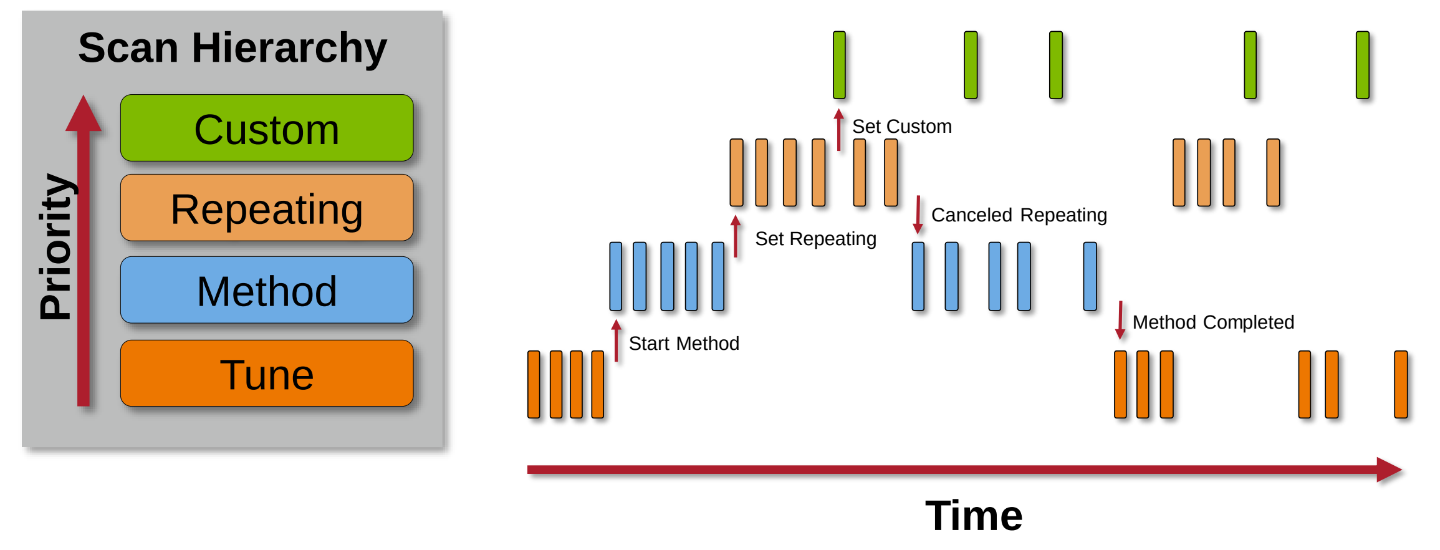


Figure 3. The above diagram illustrates the scan hierarchy. IAPI scans are defined as either "repeating" or "custom"; if there are no repeating or custom scans pending, the instrument will continue running the current method. If no method is running, the instrument will use whatever scan definition is in Tune. If the instrument is in Standby or Off, no IAPI scans will be processed by the instrument.

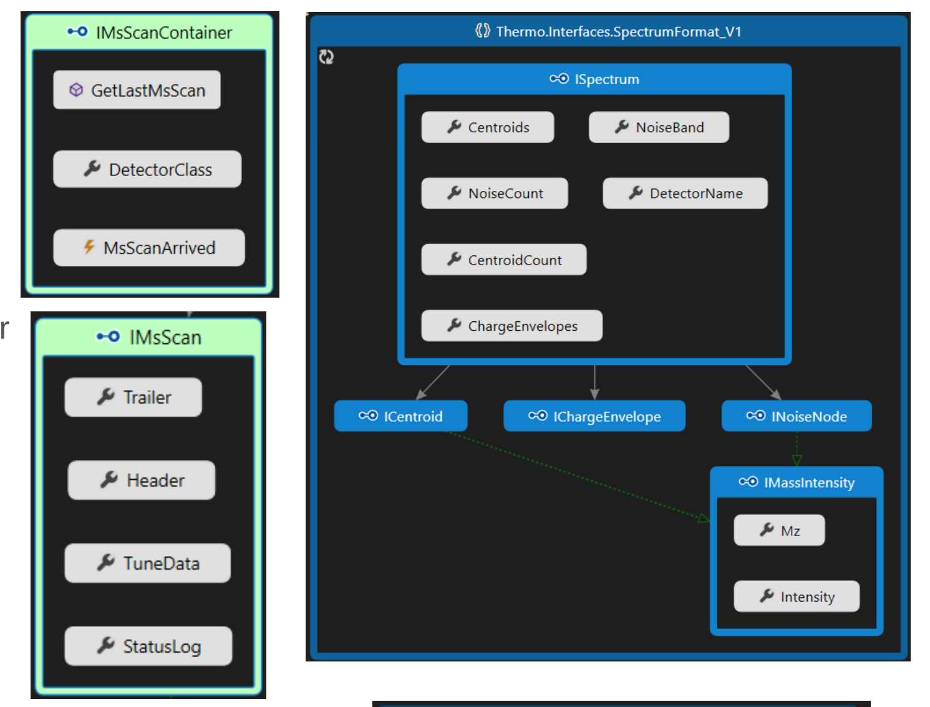
Interfaces

Several interfaces are provided for instrument control using the IAPI. Below we describe the most relevant; however, other important interfaces for connecting to the instrument service can be understood by exploring the example code on github (see the "Download and Notes" section).

Receiving Scans

Figure 4. Incoming scans are signaled by an event called MsScanArrived. This event can be subscribed to, and the scan itself received as part of the event arguments.

The scan contains header and trailer information, as well as noise and, through the Centroids property, the m/z and intensity values.

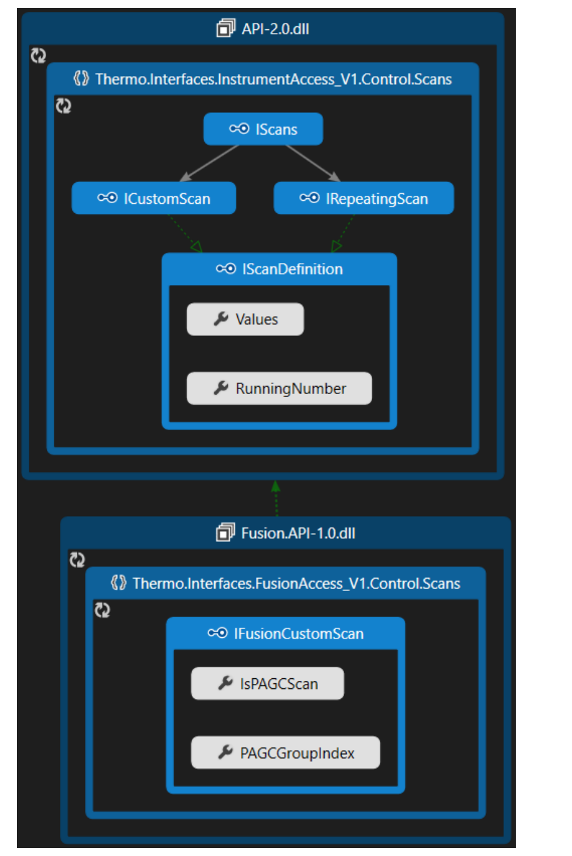


Sending Scans

Figure 5. Scans within the IAPI are created as either *repeating* or *custom,* as described above. In either case, scan values are held in a C# dictionary where both keys and values are strings. Scan parameters are set by setting appropriate values for specific keys, as in the example below, which defines an MS3 scan of *m/z* 524, with *m/z* 191 as the MS3 precursor.

```
scan.RunningNumber = 123456789;
scan.Values["FirstMass"] = "150";
scan.Values["LastMass"] = "550";
scan.Values["ScanType"] = "MSn";
scan.Values["PrecursorMass"] = "524.3;191.2";
scan.Values["CollisionEnergy"] = "35;25";
```

The RunningNumber property can be used for any purpose; it does not affect the scan acquisition. The *FusionCustomScan* is a Tribrid-specific extension of the CustomScan, and defines two additional properties, IsPAGCScan and PAGCGroupIndex, discussed in the next section.



Control of pAGC via the IAPI

AGC Overview

In normal AGC, a prescan precedes every analytical scan. With predicted AGC (pAGC), a prescan is done only prior to the MS1 scan. Injection times for subsequent scans are predicted from this prescan

As described above, IAPI scans take priority over all other scans. If many IAPI scans are being executed, the time between the last prescan and the most recent IAPI scan can increase to the point that the prescan information no longer accurately represents the actual ion flux, resulting in inaccurate injection times. We now provide a way to control the pAGC mechanism using the IAPI, described below.

Property	Type	Purpose	Note
IsPAGCScan	bool	Tells instrument that this scan should be used for pAGC calculations	Scan is restricted to ion trap full scan
PAGCGroupIndex	int	Links scans to pAGC scans; all scans with the same index will reference the same pAGC scan	Limited to 200 pAGC groups

Table 1. Using the parameters above, pAGC can be controlled for IAPI analytical scans by sending an ion trap full scan with IsPAGCScan set to true, and PAGCGroupIndex set to an integer. This instructs the instrument to save pAGC data based on that scan. Subsequent scans with IsPAGCScan set to false and PAGCGroupIndex set to that same integer will use the saved pAGC data for that PAGCGroupIndex to compute injection times.

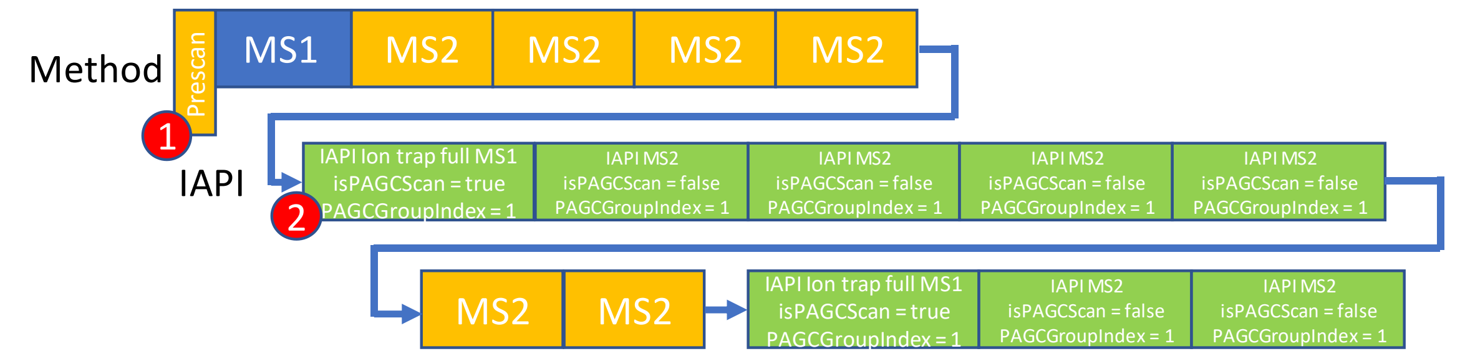

Method
Prescan
MS1
MS2
1
2
IAPI
IAPI Ion trap full MS1 isPAGCScan = true PAGCGroupIndex = 1
IAPI MS2 isPAGCScan = false PAGCGroupIndex = 1

Figure 7. The diagram above illustrates how pAGC can be controlled with the IAPI. All the method MS2 scans use the initial prescan information (1) to update their injection times. During the scan cycle, when the method prescan (1) may be too far back in time, an IAPI application may insert a IAPI scan set to ion trap, full scan (2), with parameters passed as shown. Subsequent IAPI scans with the same PAGCGroupIndex will use the pAGC data derived from IAPI scan (2).

Additional pAGC groups with separate pAGC data may be created by linking scans with the same PAGCGroupIndex. Up to 200 separate groups are allowed.

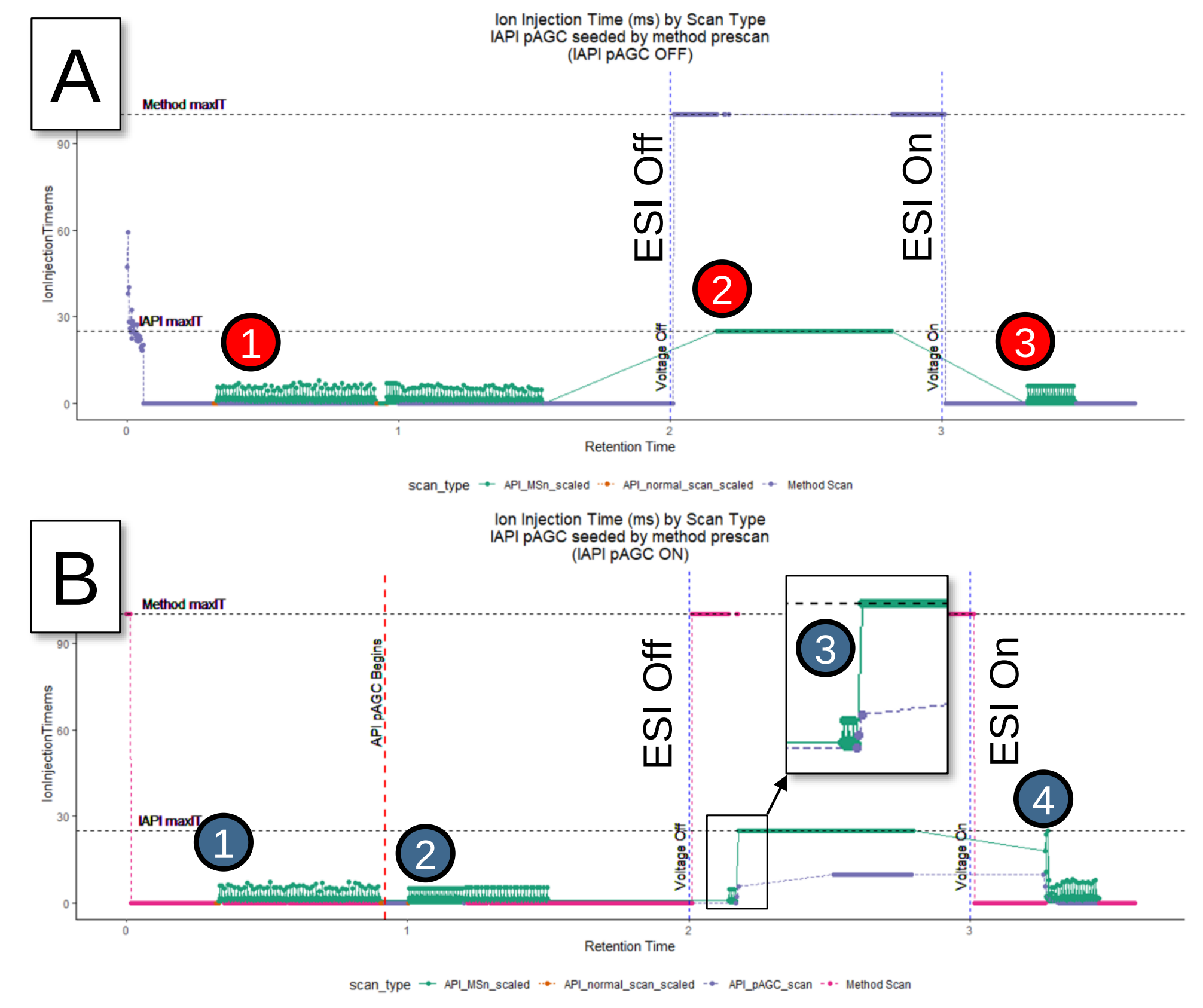
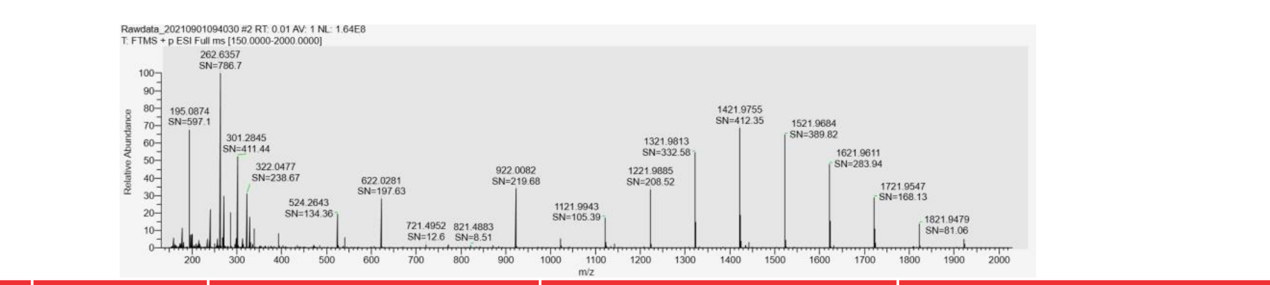


Figure 8. At point **A1**, API MSn scans are sent, and inherit pAGC data from the method scans. At **A2**, the spray turns off, and both the method scans and API scans hit their max injection times, as both are using the same set of pAGC data. At **A3**, the situation of A1 returns. In contrast, at point **B3**, API scans use pAGC data from point **B2**, until another IAPI pAGC scan runs, updating the pAGC info. It's only at that point that the IAPI scans hit the max injection time. The reverse happens at **B4**, where the IAPI scans reset their injection times only after the IAPI pAGC scan runs, updating the pAGC information.

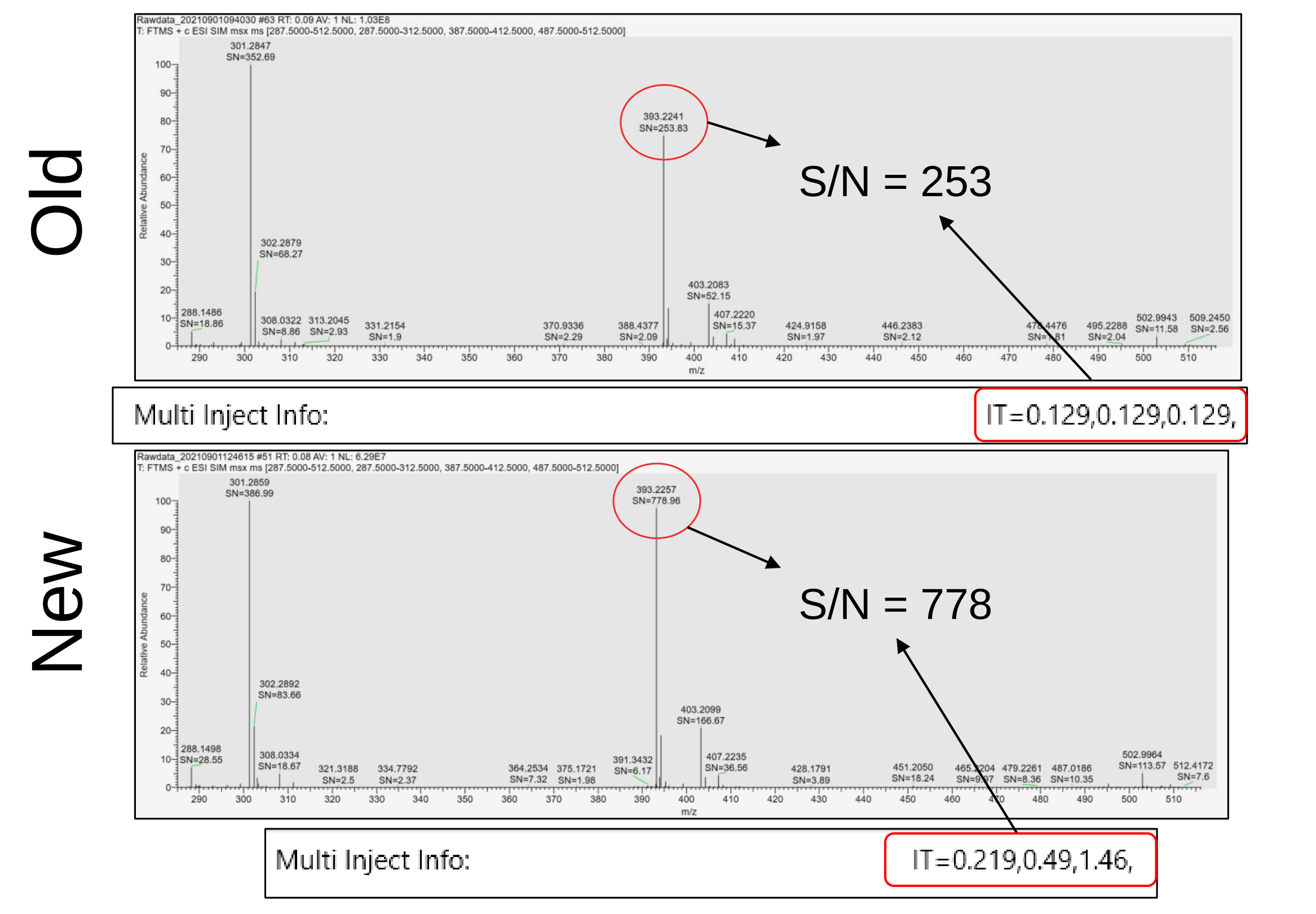
Additional New Features

Separate injection times for MSX SIM windows

Figure 9. In previous versions of the IAPI, multiplexed (MSX) SIM windows were possible, but of limited utility due to the fact that each window was assigned the same injection time as the first window in the series. Now separate windows have separate injection times, as determined by the AGC or pAGC mechanisms, resulting in enhanced signal-to-noise ratios for peaks that would've otherwise been underrepresented. In the data below, arbitrary MSX windows with width 25 Da and centers were chosen as shown. Enhanced signal-to-noise is observed for peaks that were less abundant in full scan mode.



Window center	Base peak *m/z*	Base peak S/N full scan	Base peak S/N MSX, same IT	Base peak S/N MSX, individual IT
300	301	411	352	386
400	393	58	253	778
500	503	4	12	113



Support for mass range changes

Previously, IAPI scans have been limited to operation in Normal mass range, which extends up to *m/z* 2000. With newer software, the mass range is now exposed as a parameter, enabled IAPI access to High mass range. This can extend the mass range up to *m/z* 4000 (ion trap) and *m/z* 6000 (Orbitrap) or up to *m/z* 8000 (Orbitrap) on a properly licensed instrument.

A scan can be prepared for high mass range by using the code snippet at right.

```
scan.Values["MassRange"] = "High";
```

Support for multistage activation (MSA)

With newer software, multistage activation (MSA) is now supported. In MSA, an extra stage of activation is done for resonant CID at the designated mass value. The analyzer must be set to IonTrap, and the activation type must be CID.

A scan can be prepared for MSA by using the code snippet at right.

```
scan.Values["PrecursorMass"] = "524.3";
scan.Values["ScanType"] = "MSn";
scan.Values["ActivationType"] = "CID";
scan.Values["CollisionEnergy"] = "20";
scan.Values["MSANeutralLossMass"] = "506";
```

Support for scan range mode

Table 2. With newer software, different types of scan range modes are now supported, as described in the table below. The default is Auto.

Mode	String to send	Purpose
Auto	"Auto"	Scan range for MSn scans is automatically determined. Last mass is to precursor*charge+10; first mass is lowest mass that retains ions at last mass.
Defined First Mass	"DefineFirstMass"	First mass is user-determined, last mass is precursor*charge+10
Defined Range	"DefineMZRange"	Scan range for MSn scans is user-determined

Code example:

```
scan.Values["ScanRangeMode"] = "DefineMZRange";
```

Download and Notes

Prerequisites for use of the IAPI with features described in this poster:

- Installation of Instrument Control Software version 3.5.3881.18. Earlier versions back to 3.0 are supported but will not support features discussed here.
- IAPI License agreement. This may be obtained by clicking the "gear" icon at the upper right of Tune, clicking "About," selecting "License…" and then "Get API License."
- Active IAPI license key.
- .NET 4.6.2+
- Public interfaces, downloadable at

  http://github.com/thermofisherlsms/iapi

- Exactive series and Exploris series IAPI are also available at the above github location.
- Example code has been updated to illustrate how to use the new functionality.
- A patch is necessary for some of the above functionality, available at the github site.

Note: Patch Necessary!

- Some of the functionality described here requires a patch, available here:

  http://github.com/thermofisherlsms/iapi/tree/master/misc

REFERENCE

1. Derek Bailey et al., TP 393, ASMS 2016.

ACKNOWLEDGEMENTS

We gratefully acknowledge the feedback and contributions of Qing Yu in Steve Gygi's lab at Harvard, Devin Schweppe of the University of Washington, Dennis Goldfarb of Washington University, and John McGee and Ken Durbin in Neil Kelleher's lab at Northwestern University.

TRADEMARKS/LICENSING

PO66127 EN0921S