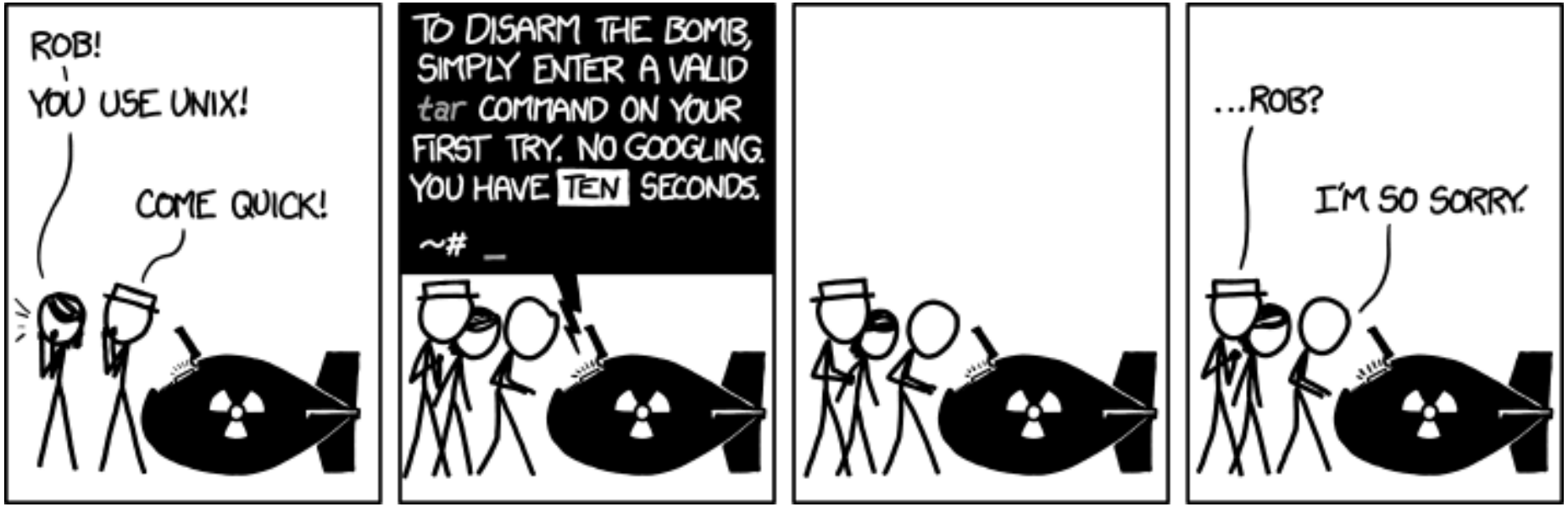
ThermoFisher SCIENTIFIC

Instrument API and XML Method Modification Interface

ASMS Orbitrap Power User Meeting – June 2nd, 2018, San Diego, CA

Derek Bailey, Ph.D.

ASMS 2018 Warm Up Slide



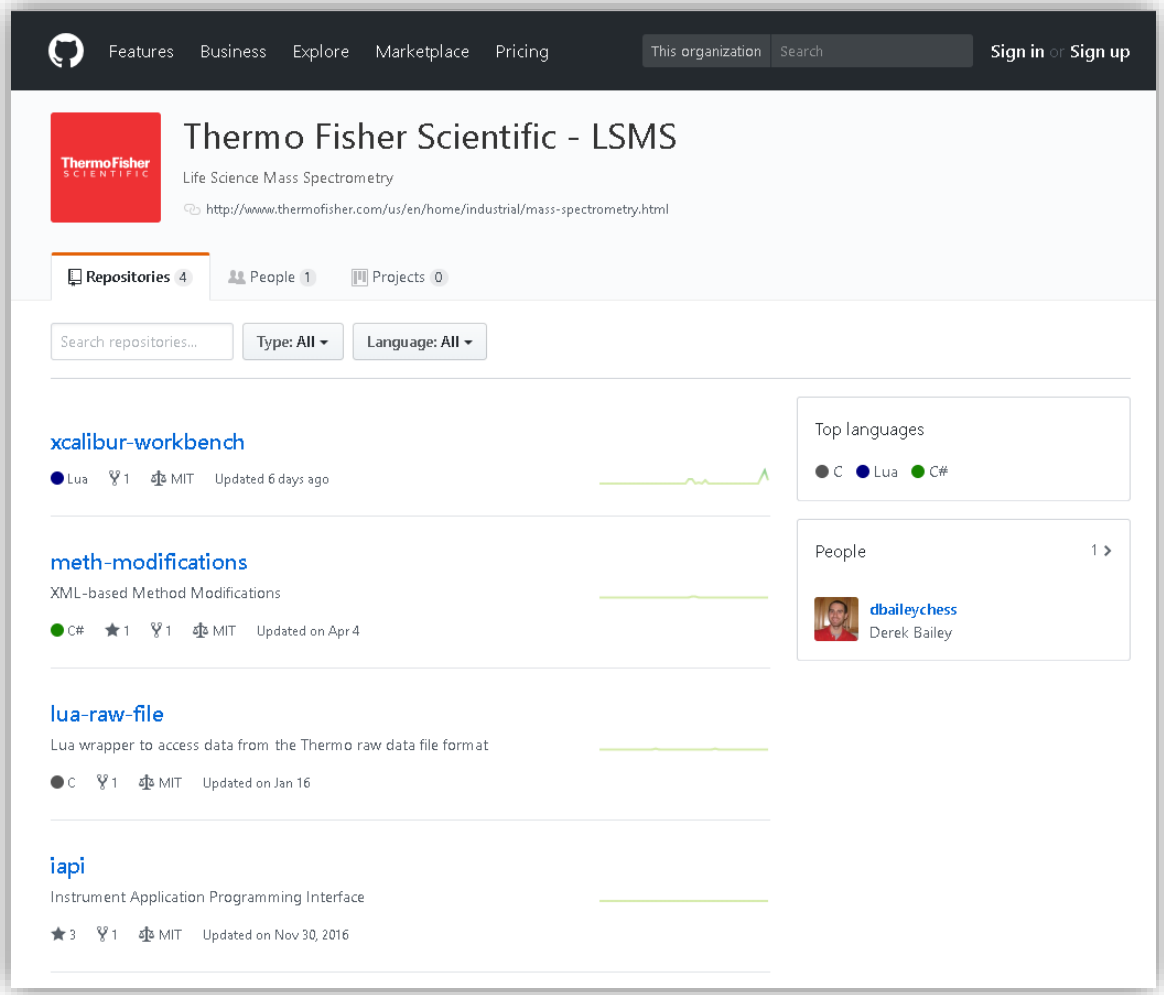
https://xkcd.com/1168/

Overview

1. **Instrument Application Programming Interface (IAPI)**
   - Real-time control and acquisition of Orbitrap Mass Spectrometers using .NET
2. **XML Method Modification Interface (XMMI)**
   - Programmatically modify Fusion-series method files
3. **Applications of IAPI**
   1. **Devin Schweppe** – Fusion API
   2. **Christoph Wichmann** – Q Exactive API
4. **Examples and FAQ for IAPI and XMMI**
5. **Applications of XMMI**
   1. **Pavel Shliaha** – Fusion Method Creation

Thermo Fisher Scientific LSMS GitHub Page

https://github.com/thermofisherlsms



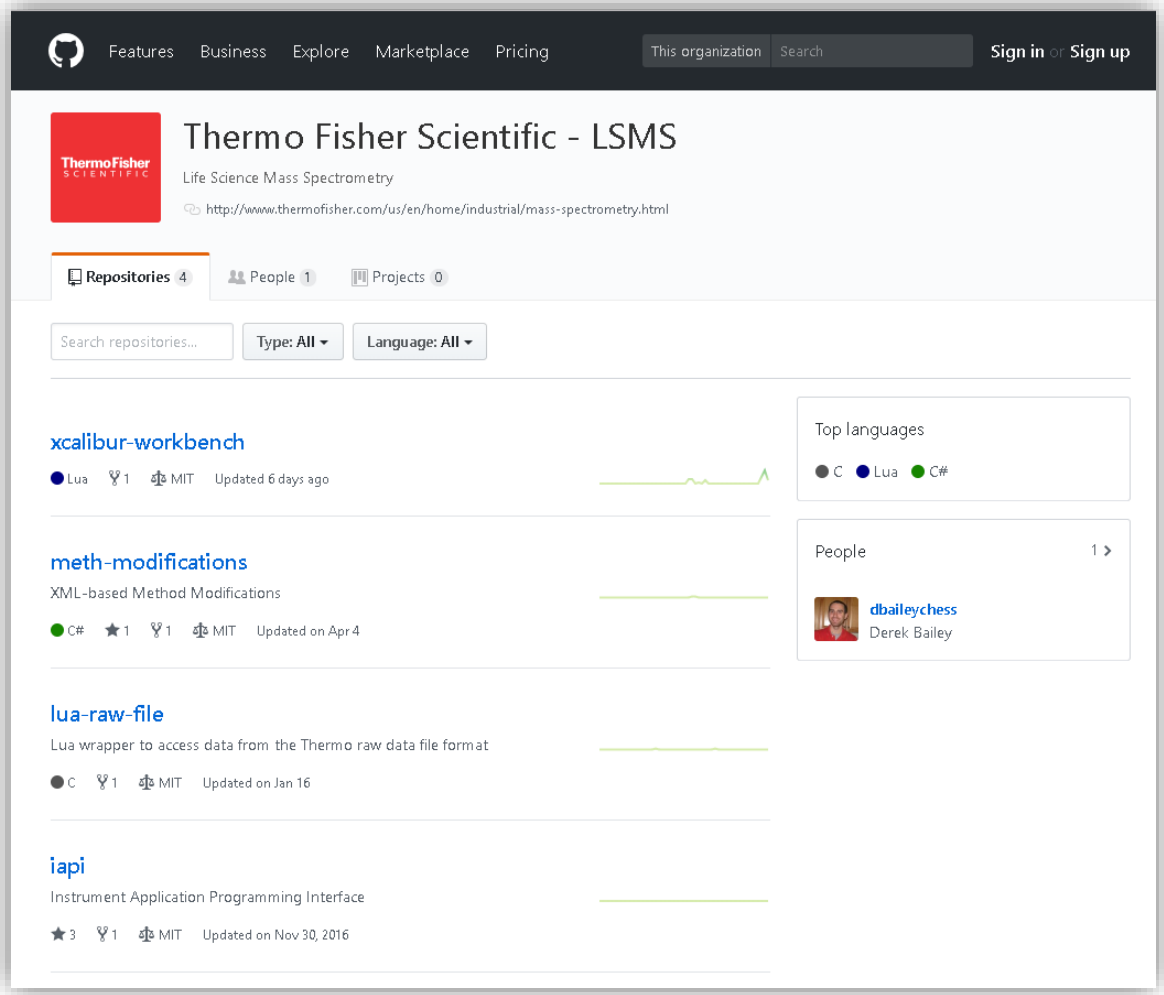
Thermo Fisher Scientific LSMS GitHub Page

https://github.com/thermofisherlsms

Mike Senko
Tuesday Poster

Covered Today

Covered Today



ThermoFisher SCIENTIFIC

Overview

1. **Instrument Application Programming Interface (IAPI)**
   - Real-time control and acquisition of Orbitrap Mass Spectrometers using .NET
2. **XML Method Modification Interface (XMMI)**
   - Programmatically modify Fusion-series method files
3. **Applications of IAPI**
   1. **Devin Schweppe** – Fusion API
   2. **Christoph Wichmann** – Q Exactive API
4. **Examples and FAQ for IAPI and XMMI**
5. **Applications of XMMI**
   1. **Pavel Shliaha** – Fusion Method Creation

Why an API?

"I wish I could do …"

"Can I do this in method editor?"

"I only want to select peaks that have odd masses …"

"I have this awesome new experiment …"

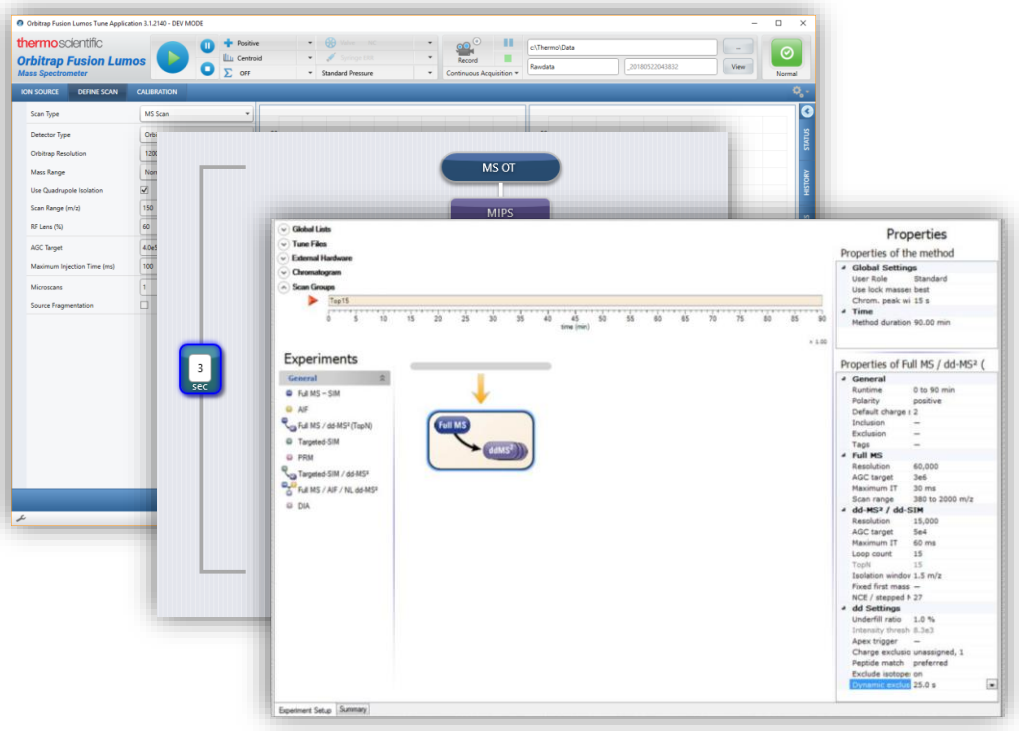
Why an API?

"I wish I could do …"

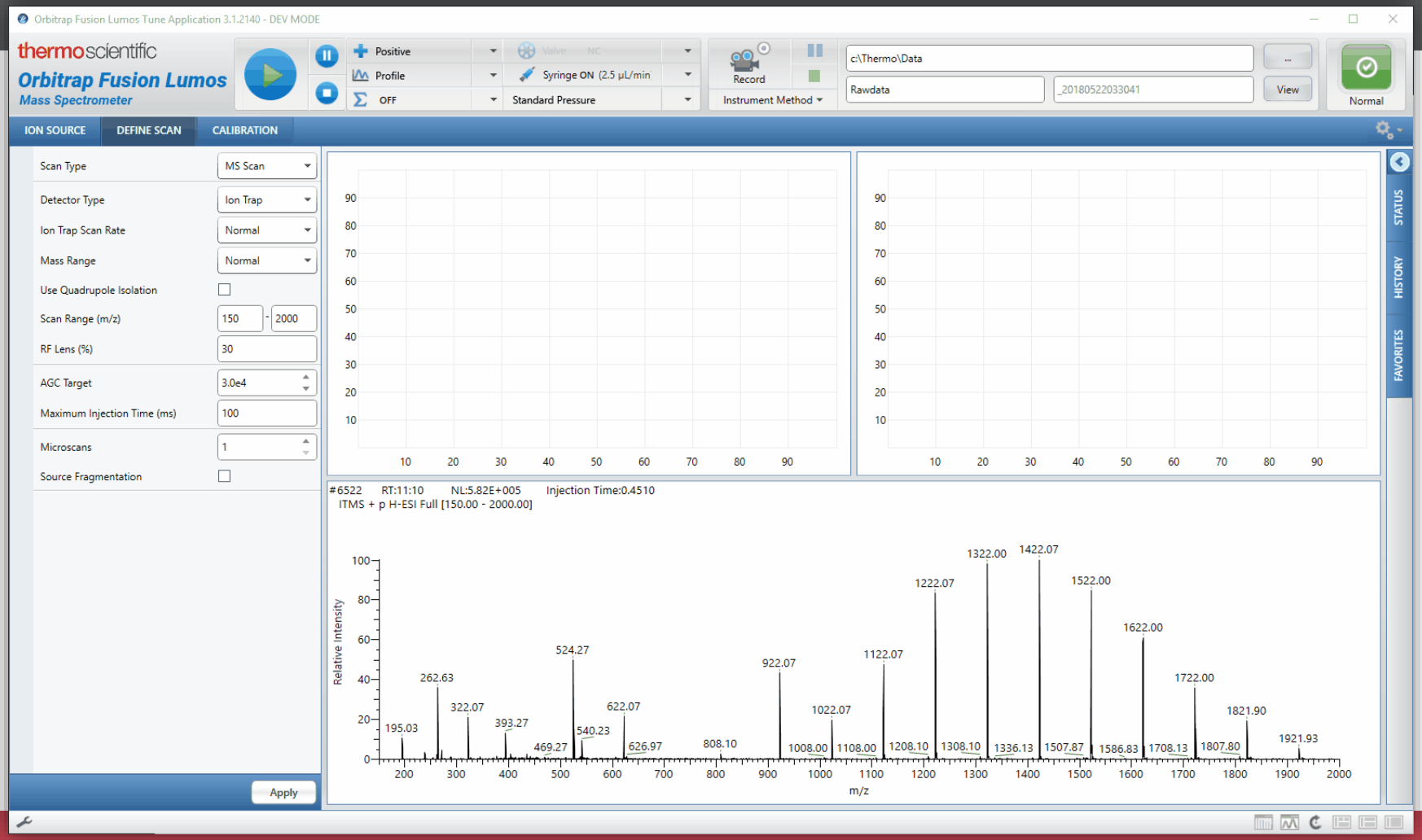
"Can I do this in method editor?"

"I only want to select peaks that have odd masses …"

"I have this awesome new experiment …"



- Users can only control the Mass Spectrometer through our **Tune** application and **Method** acquisition.
- Although both provide **great flexibility** in acquisition strategies they don't encompass all possibilities.
- Providing an Application Programming Interface enables users to **extend** the capabilities of the Orbitrap mass spectrometers.
- Allows **real-time** data acquisition and control.



The **IAPI** empowers users to **programmatically** control acquisition of both the **Exactive** and **Tribrid**-series Mass Spectrometers

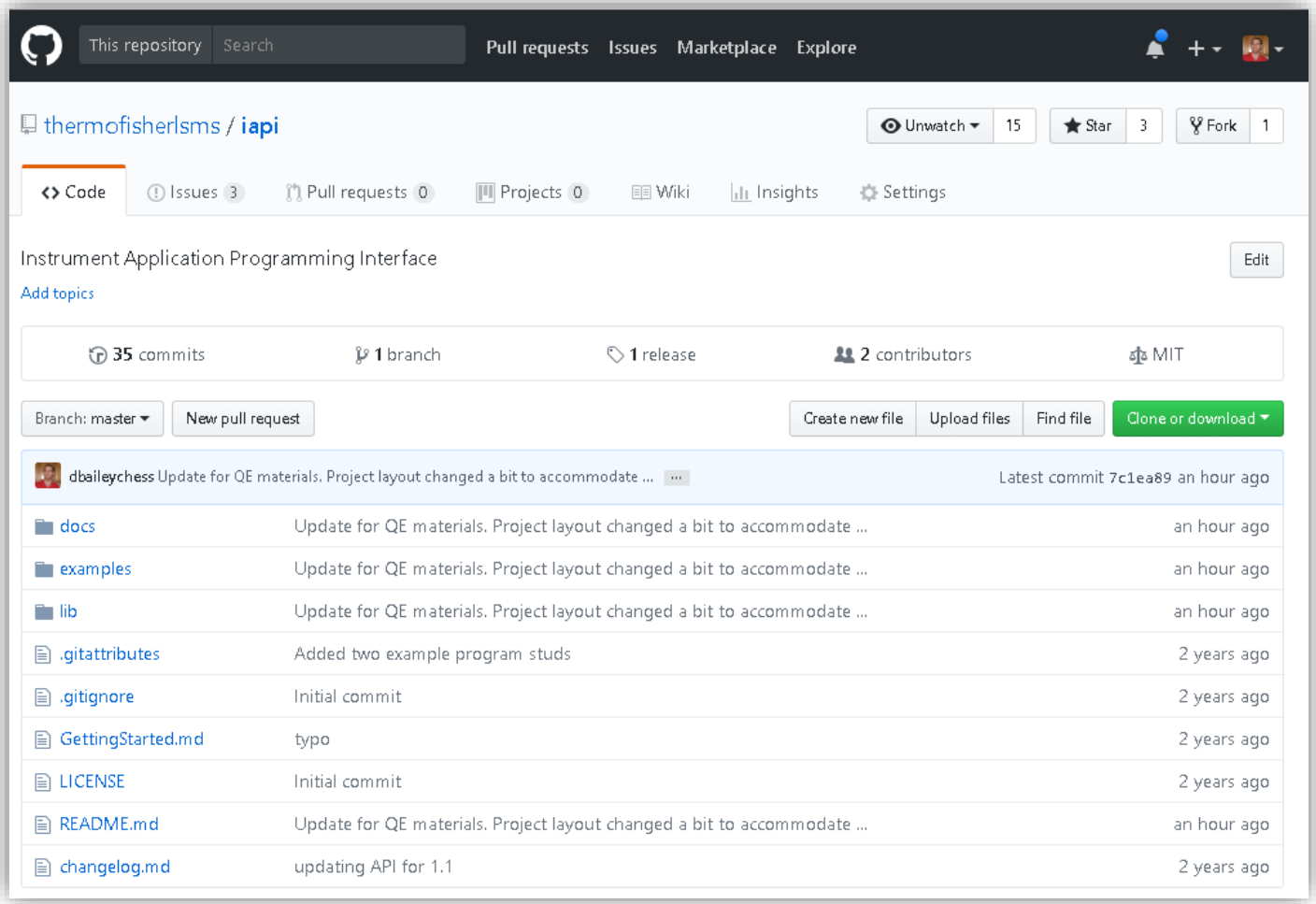
ThermoFisher SCIENTIFIC

IAPI Outline

1. **IAPI Architecture**
   - Interfaces
   - Data and Control Flow
2. **Receiving Data from the MS**
   - Interfaces
   - Scan Data Stream Subscription
3. **Controlling the MS**
   - Sending Scan Definitions
   - Changing other MS parameters
4. **Getting Started**

Instrument API Git Repository

https://github.com/thermofisherlsms/iapi



- Public interfaces for
  - Tribrid
  - Exactive
- Documentation
  - Previous ASMS Posters
  - Help files
- VS Solution of Example Applications

External Requirements

- Tune Installation
- .NET 4.6.2+
- License Agreement in place
- IAPI License Key Activated

IAPI Outline

1. **IAPI Architecture**
   - Interfaces
   - Data and Control Flow
2. **Receiving Data from the MS**
   - Interfaces
   - Scan Data Stream Subscription
3. **Controlling the MS**
   - Sending Scan Definitions
   - Changing other MS parameters
4. **Getting Started**

Interfaces

Exactive Series	IAPI Version	Instrument Extensions	Misc.
	API-1.0.dll	ESAPI-1.0.dll	VI-1.0.dll
	API-1.1.dll	ESAPI-1.1.dll	

Tribrid Series

API-2.0.dll

Fusion.API-1.0.dll

Spectrum-1.0.dll

Thermo.TNG.Factory.dll

All interfaces are in the git repository, in the lib/ directory

Differences and Similarities between Exactive and Tribrid Series IAPI

- Although IAPI 1.X is **not binary compatible** with IAPI 2.X, they share considerable amount of structure
  - Namespaces and hierarchies are virtually unchanged
  - A few name changes or minor reorganization
  - Removal of some interfaces
  - Spectrum-related interfaces moved into own assembly

Differences and Similarities between Exactive and Tribrid Series IAPI

- Although IAPI 1.X is **not binary compatible** with IAPI 2.X, they share considerable amount of structure
  - Namespaces and hierarchies are virtually unchanged
  - A few name changes or minor reorganization
  - Removal of some interfaces
  - Spectrum-related interfaces moved into own assembly
- The logical flow for both receiving scan data and sending scans definitions to the instrument is **identical**

Differences and Similarities between Exactive and Tribrid Series IAPI

- Although IAPI 1.X is **not binary compatible** with IAPI 2.X, they share considerable amount of structure
  - Namespaces and hierarchies are virtually unchanged
  - A few name changes or minor reorganization
  - Removal of some interfaces
  - Spectrum-related interfaces moved into own assembly
- The logical flow for both receiving scan data and sending scans definitions to the instrument is **identical**
- Different methods for instantiating the IAPIs
  - Tribrid-series uses a **Factory** method model with Microsoft's Managed Extensibility Framework (MEF)
  - Exactive-series uses the **System.Reflection.Assembly** API

Differences and Similarities between Exactive and Tribrid Series IAPI

- Although IAPI 1.X is **not binary compatible** with IAPI 2.X, they share considerable amount of structure
  - Namespaces and hierarchies are virtually unchanged
  - A few name changes or minor reorganization
  - Removal of some interfaces
  - Spectrum-related interfaces moved into own assembly
- The logical flow for both receiving scan data and sending scans definitions to the instrument is **identical**
- Different methods for instantiating the IAPIs
  - Tribrid-series uses a **Factory** method model with Microsoft's Managed Extensibility Framework (MEF)
  - Exactive-series uses the **System.Reflection.Assembly** API
- Some interfaces are not implemented due to technical reasons or deemed not immediately useful

The rest of this presentation will focus on the **Tribrid**-series IAPI.

Most topics are applicable to the **Exactive**-series with minor differences.

MS Architecture

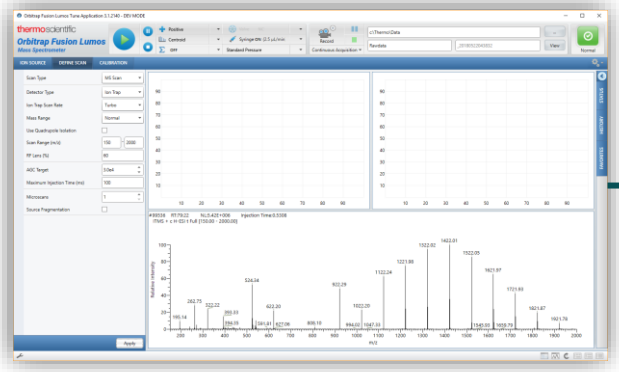
Mass Spectrometer

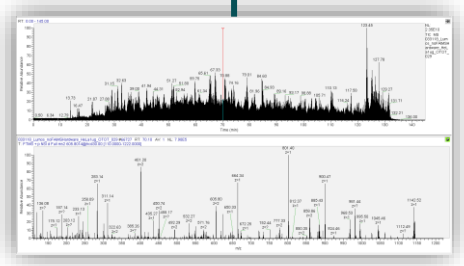
Local Switch

Data System

Instrument Service

Tune Application





Rawfile

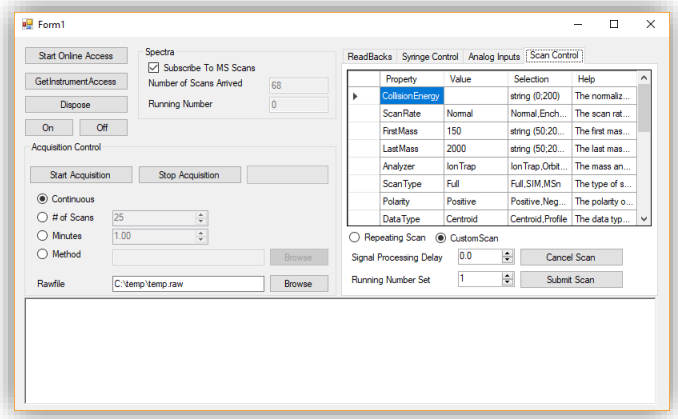
IAPI Architecture

Mass Spectrometer

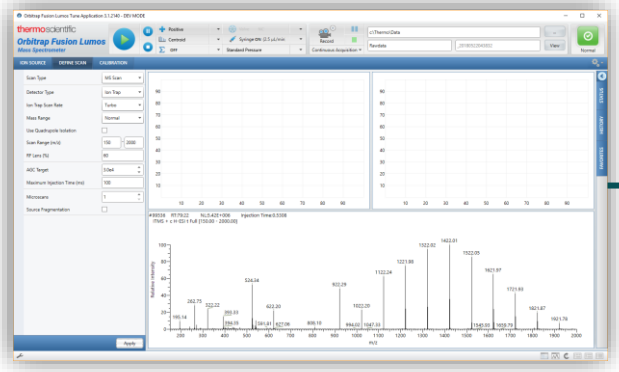
Local Switch

Data System

IAPI-Enabled App



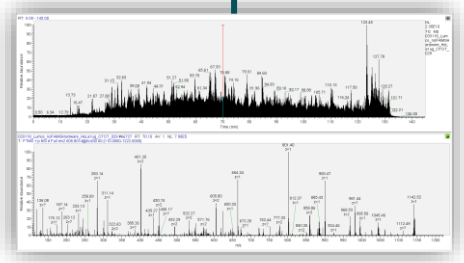
Tune Application



Instrument Service

3rd Party Application

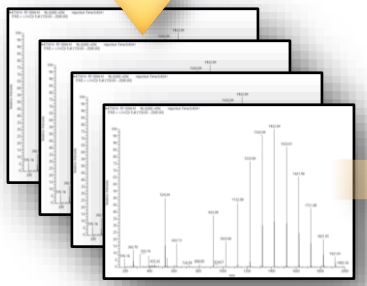
API-2.0.dll



Rawfile

IAPI Data and Control Flow

Mass Spectrometer



Scan Data Stream

Data System

Instrument Service

API-2.0.dll

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!

IAPI Data and Control Flow

Mass Spectrometer

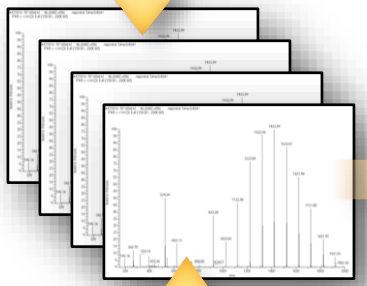
Data System

Instrument Service

API-2.0.dll

3rd Party Application

Scan Data Stream



- Displays scan in Tune
- Writes to Rawfile
- Forwards data to API

- Converts data into IAPI format
- Raises **Event** to any subscribers

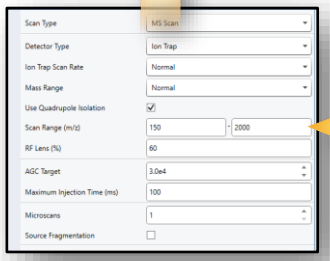
- Handles scan **Event**
- Processes the data
- Do whatever you want!

- 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

Scan Definition

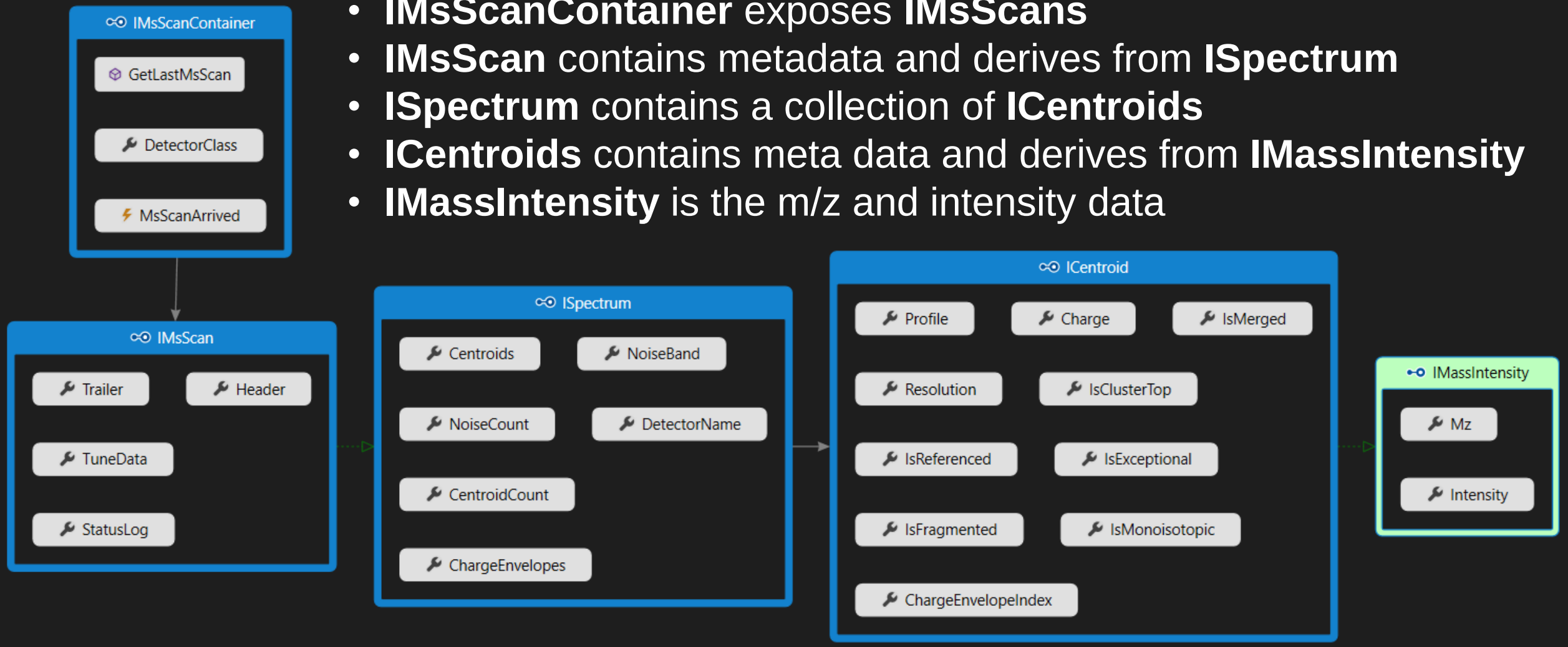


IAPI Outline

1. IAPI Architecture
   - Interfaces
   - Data and Control Flow
2. **Receiving Data from the MS**
   - Interfaces
   - Scan Data Stream Subscription
3. Controlling the MS
   - Sending Scan Definitions
   - Changing other MS parameters
4. Getting Started

MS Scan Interfaces

- **IMsScanContainer** exposes **IMsScans**
- **IMsScan** contains metadata and derives from **ISpectrum**
- **ISpectrum** contains a collection of **ICentroids**
- **ICentroids** contains meta data and derives from **IMassIntensity**
- **IMassIntensity** is the m/z and intensity data



Scan Data Stream

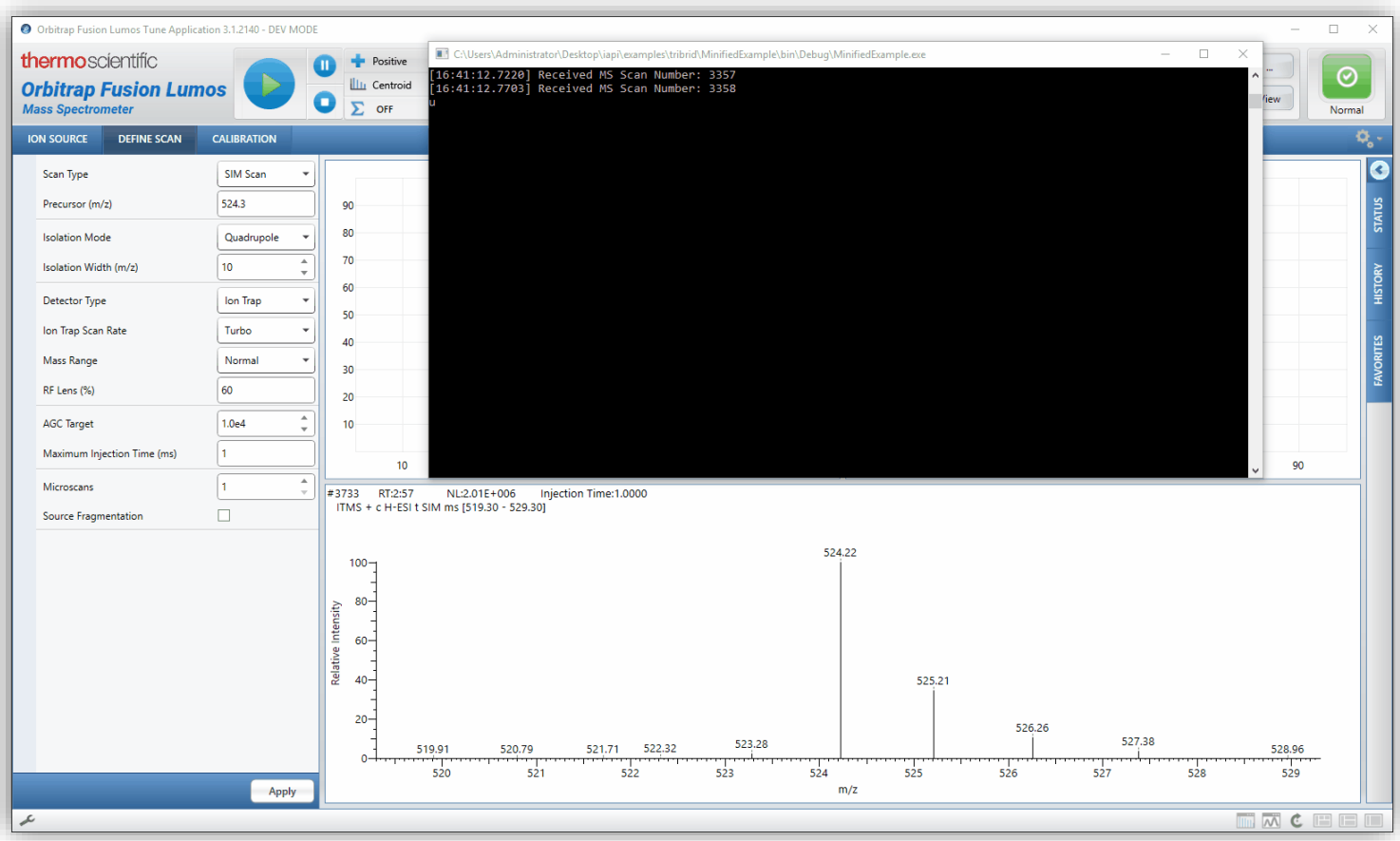
- Receiving the scan data is through the **Event Driven** programming model
- Scans are sent via the .NET **Event** and **EventHandler** mechanism
  - Multiple handlers per scan can be registered
  - Minimal overhead and latency
- Receives Scans from both Tune and Method acquisitions

```
26          // Get the MS Scan Container from the fusion
27          IFusionMsScanContainer fusionScanContainer = fusionAccess.GetMsScanContainer(0);
28
29          // Run forever until the user Escapes
30          ConsoleKeyInfo cki;
31          while ((cki = Console.ReadKey()).Key != ConsoleKey.Escape)
32          {
33              switch(cki.Key)
34              {
35                  case ConsoleKey.S:
36                      // Subscribe to whenever a new MS scan arrives
37                      fusionScanContainer.MsScanArrived += FusionScanContainer_MsScanArrived;
38                      break;
39                  case ConsoleKey.U:
40                      // Unsubscribe
41                      fusionScanContainer.MsScanArrived -= FusionScanContainer_MsScanArrived;
42                      break;
43                  default:
44                      Console.WriteLine("Unsupported Key: {0}", cki.Key);
45                      break;
46              }
47          }
48      }
49
        2 references | Thermo Scientific, 1 hour ago | 2 authors, 2 changes
50      private static void FusionScanContainer_MsScanArrived(object sender, MsScanEventArgs e)
51      {
52          // Print out the scan number of the scan received to console
53          Console.WriteLine("[{0:HH:mm:ss.ffff}] Received MS Scan Number: {1}",
54              DateTime.Now,
55              e.GetScan().Header["Scan"]);
56      }
57  }
58 }
```


This example is from the examples/tribrid/MinifiedExample/ directory in the Git repository

Scan Data Stream

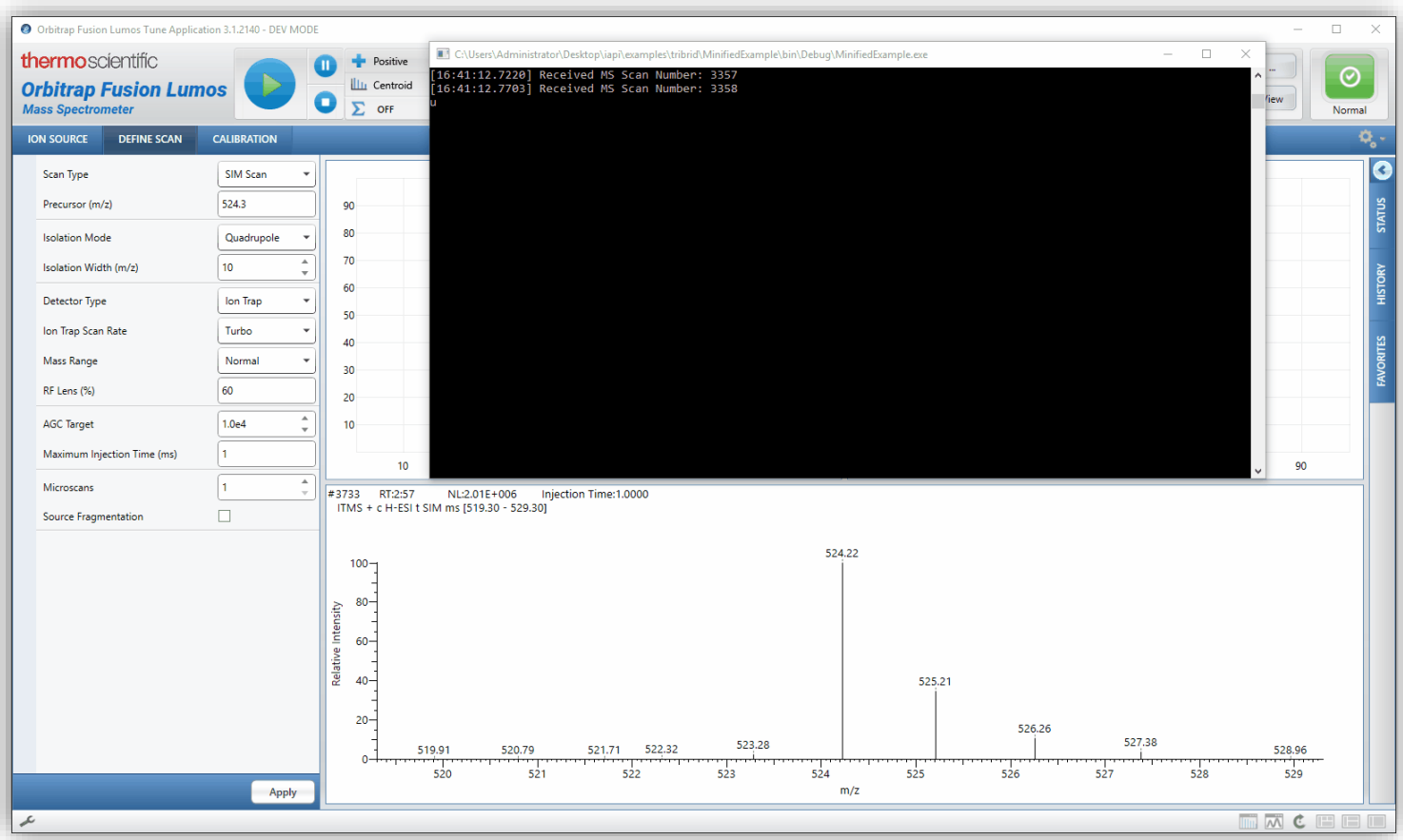
- Receiving the scan data is through the **Event Driven** programming model
- Scans are sent via the .NET **Event** and **EventHandler** mechanism
  - Multiple handlers per scan can be registered
  - Minimal overhead and latency
- Receives Scans from both Tune and Method acquisitions
- tSIM running ~21 Hz (**47.5** ms) from Tune.
  - Average time of **48.4** ms between scans received (n = 200)



This example is from the examples/tribrid/MinifiedExample/ directory in the Git repository

Scan Data Stream

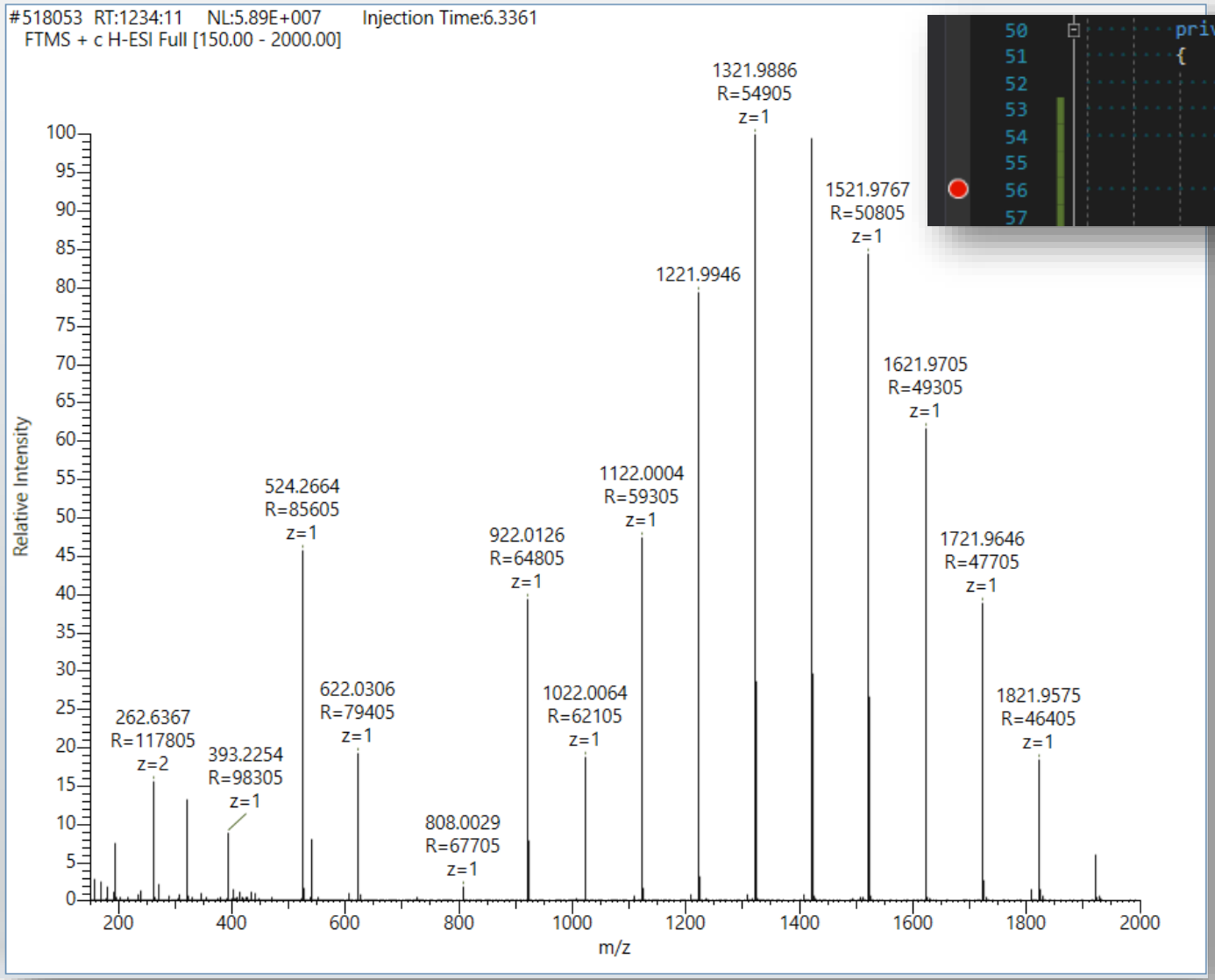
- Receiving the scan data is through the **Event Driven** programming model
- Scans are sent via the .NET **Event** and **EventHandler** mechanism
  - Multiple handlers per scan can be registered
  - Minimal overhead and latency
- Receives Scans from both Tune and Method acquisitions
- tSIM running ~21 Hz (**47.5** ms) from Tune.
  - Average time of **48.4** ms between scans received (n = 200)



The IAPI is responsive and has no impact on the MS acquisition

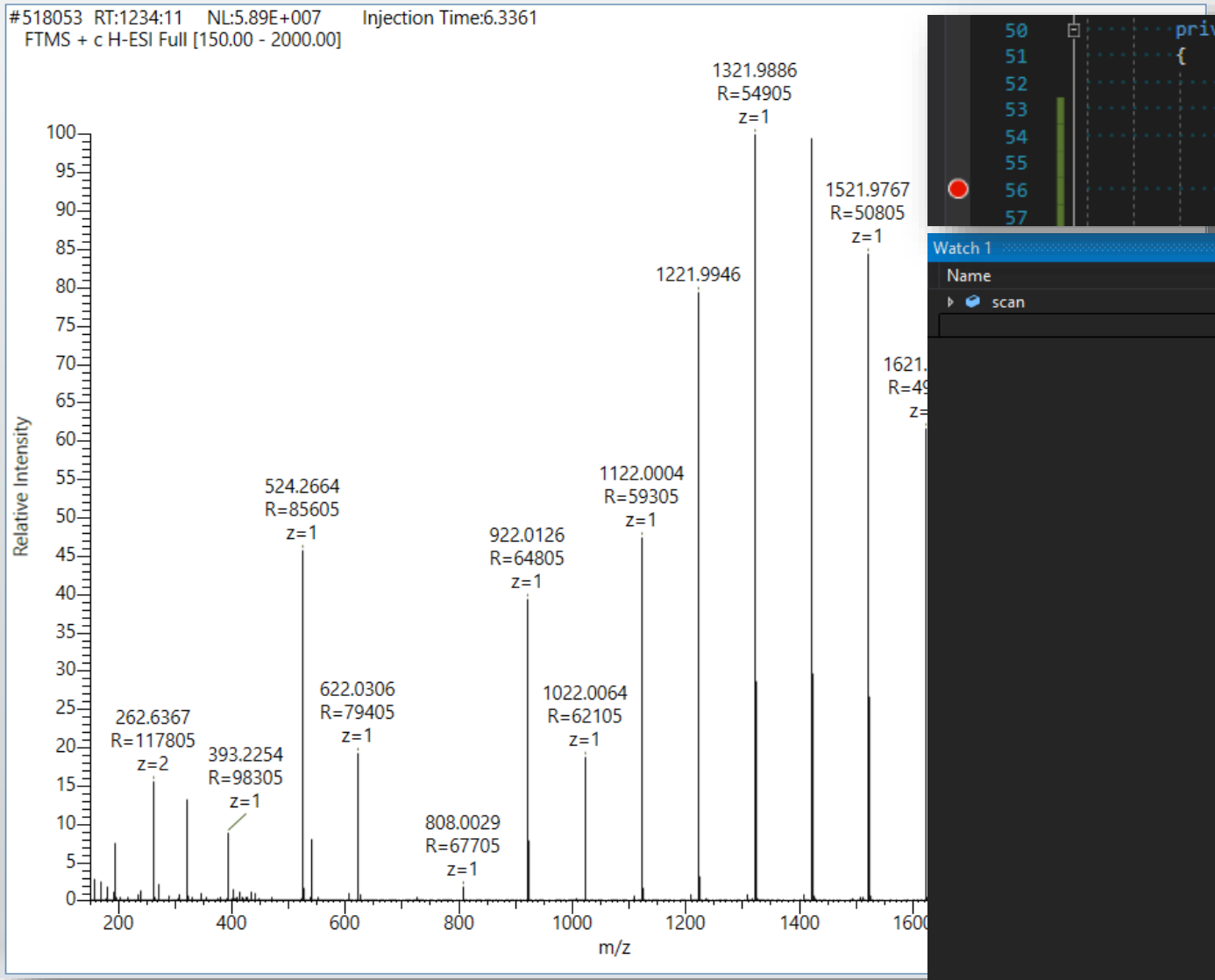
This example is from the examples/tribrid/MinifiedExample/ directory in the Git repository

Scan Data Contents




```
50  private static void FusionScanContainer_MsScanArrived(object sender, MsScanEventArgs e)
51  {
52      // Print out the scan number of the scan received to console
53      Console.WriteLine("[{0:HH:mm:ss.ffff}] Received MS Scan Number: {1}",
54          DateTime.Now, e.GetScan().Header["Scan"]);
55
56      var scan = e.GetScan();
57
```

Scan Data Contents




```
50    private static void FusionScanContainer_MsScanArrived(object sender, MsScanEventArgs e)
51    {
52        // Print out the scan number of the scan received to console
53        Console.WriteLine("[{0:HH:mm:ss.ffff}] Received MS Scan Number: {1}",
54            DateTime.Now, e.GetScan().Header["Scan"]);
55
56        var scan = e.GetScan();
57
```

Watch 1

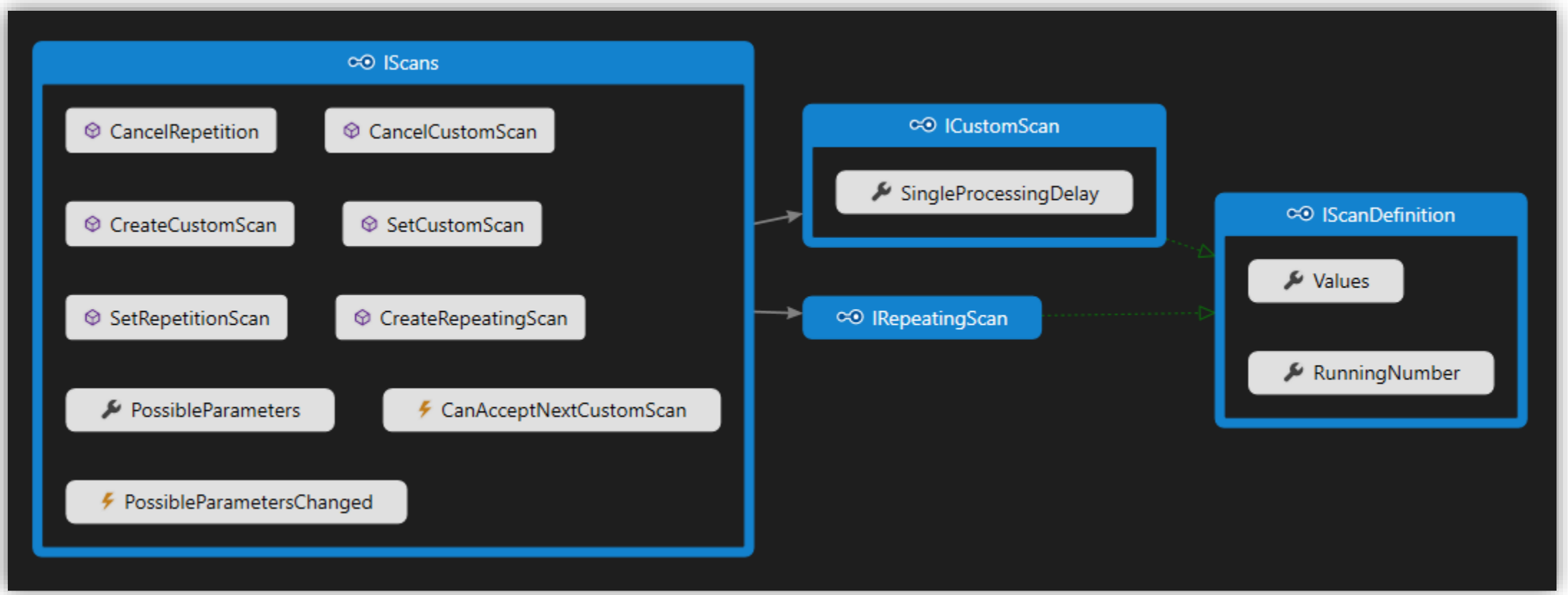
Name	Value	Type
scan	{Thermo.TNG.Client.API.MsScanContainer.MsScan}	Thermo.

IAPI Outline

1. **IAPI Architecture**
   - Interfaces
   - Data and Control Flow
2. **Receiving Data from the MS**
   - Interfaces
   - Scan Data Stream Subscription
3. **Controlling the MS**
   - Sending Scan Definitions
   - Changing other MS parameters
4. **Getting Started**

IScans Interface

- Sending scans to the instrument is through the **IScans** interface
  - Provides methods for creating different types of **IScanDefinitions**
  - Provides methods for setting and cancelling scans.



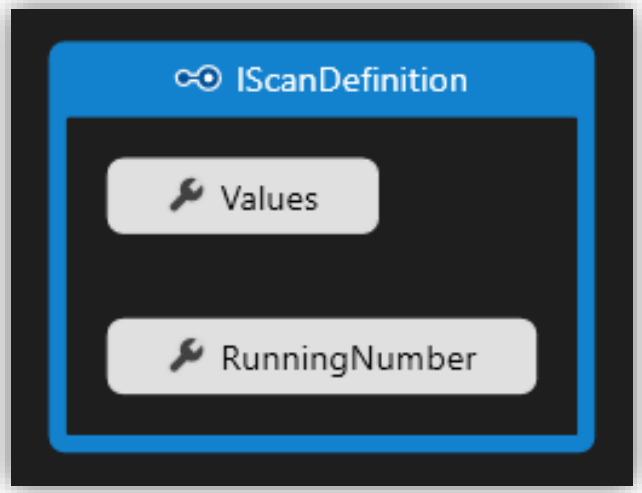
IScanDefinition

- Scans are defined in the **IScanDefinition** interface
  - The **RunningNumber** property is for general purpose use
    - It shows up in the Scan Trailer as the **'Access ID'** field

Scan Header 557 - Rawdata_20180522033041

Name	Value
Dependency Type:	0
Access ID:	1

  - The **Values** property is a Dictionary<string,string>
    - Keys are the scan parameters
    - Values are their associated values
    - Values can take different formats based on the parameter
    - Invalid, or nonsense values, are ignored



Example of MS^3 of 524@35 191@25
Scan 150 – 550 m/z

```
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";
```

IScanDefinition

Property	Value	Selection	Help
ChargeStates		string (0;25)	Charge states for HCD(default 0 is unknown) It is expressed as a string of values, with eac...
IsolationMode	Quadrupole	None,Quadrupole,IonTrap	Isolate using the quadrupole or ion trap
SourceCIDEnergy	0	0-100	Source CID Energy (0 = off).
ActivationQ	0.25	string (0.05;0.8)	The Activation Q value (0.05-0.8). It is expressed as a string of values, with each value se...
ActivationType	CID	string (CID;HCD)	The activiation type to use at a given MS stage. The available types are (CID,HCD). It is e...
AGCTarget	3000	3000-100000	The Automatic Gain Control (AGC) target value.
DataType	Centroid	Centroid,Profile	The data type to collect the scan in.
FirstMass	150	string (50;2000)	The first mass of the scan range. It is expressed as a string of values, with each value sep...
IsolationWidth	0.7	string (0.4;2000)	The isolation width (full-width) for a given MS stage It is expressed as a string of values, wit...
LastMass	2000	string (50;2000)	The last mass of the scan range. It is expressed as a string of values, with each value sep...
Analyzer	IonTrap	IonTrap,Orbitrap	The mass analyzer.
MaxIT	100	0.001-8000	The maximum injection time (ms)
CollisionEnergy		string (0;200)	The normalized collision energy (NCE) It is expressed as a string of values, with each value...
Microscans	1	1-6000	The number of microscans to collect (1 = don't use microscans)
OrbitrapResolution	120000	7500,15000,30000,50000,60000,120000,240000,500000	The Orbitrap Resolution
Polarity	Positive	Positive,Negative	The polarity of the scan.
PrecursorMass		string (50;2000)	The precursor m/z to isolate at a given MS stage. The first value will be the MS1->MS2 tra...
ReactionTime	10	string (0.001;100)	The reaction/activation time (ms) for CID activations. It is expressed as a string of values, ...
SrcRFLens	60	string (0;150)	The RF Lens (%) for the source. It is expressed as a string of values, with each value sepe...
ScanRate	Normal	Normal,Enchanced,Zoom,Rapid,Turbo	The scan rate of the ion trap
ScanType	Full	Full,SIM,MSn	The type of scan to perform.

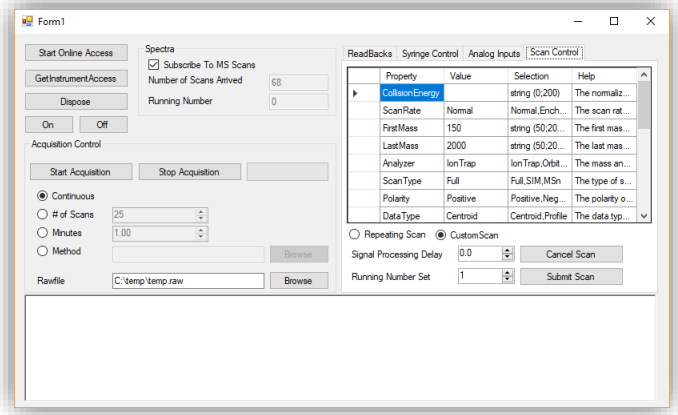
Scan Management

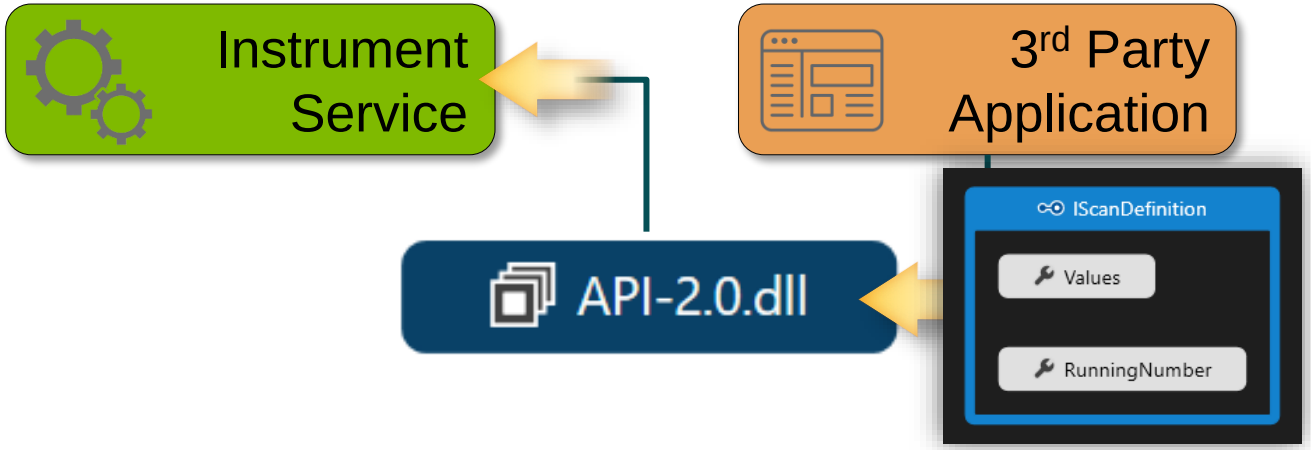
Mass Spectrometer

Local Switch

Data System

IAPI-Enabled App





Sending Scan to MS

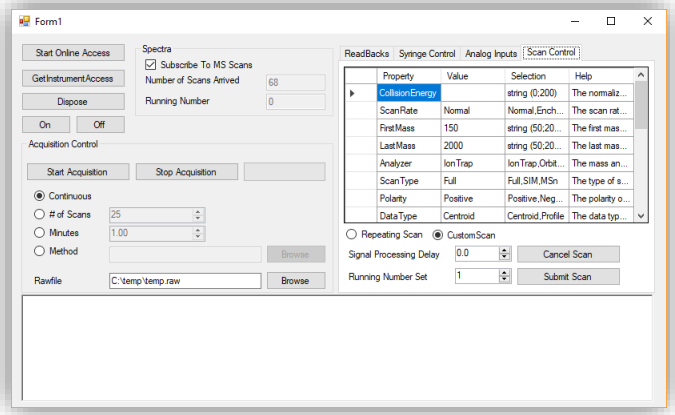
Scan Management

Mass Spectrometer

Local Switch

Data System

IAPI-Enabled App



Scan Hierarchy

Priority

- Custom
- Repeating
- Method
- Tune

Instrument Service

3rd Party Application

API-2.0.dll

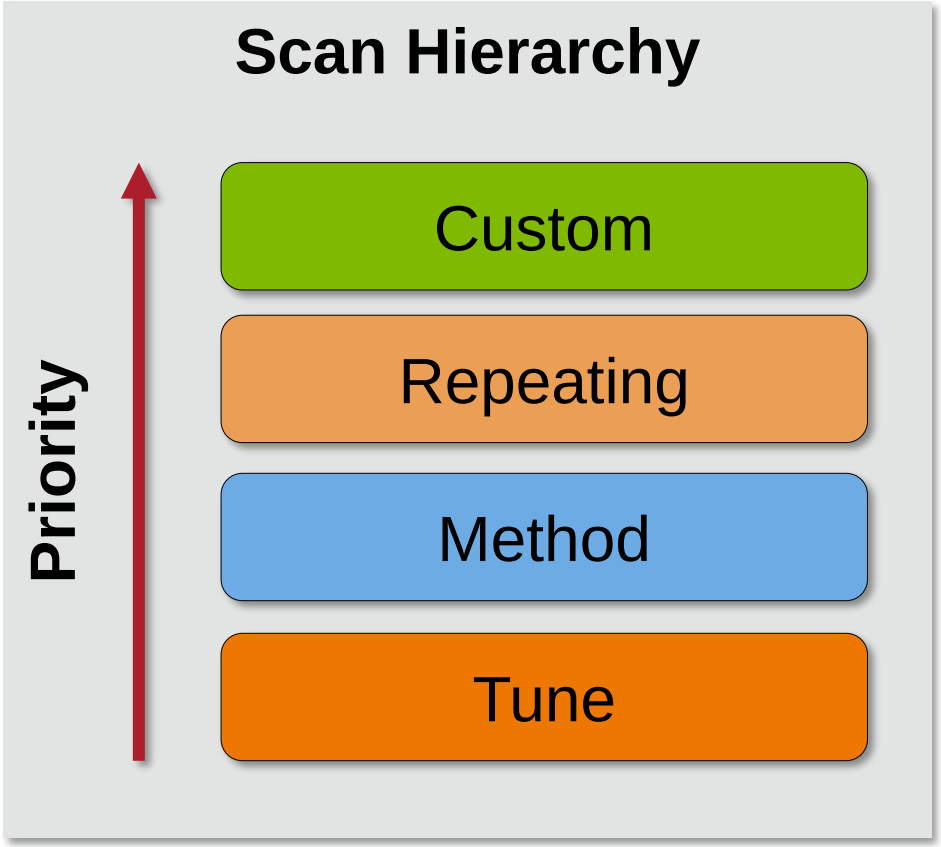
IScanDefinition

Values

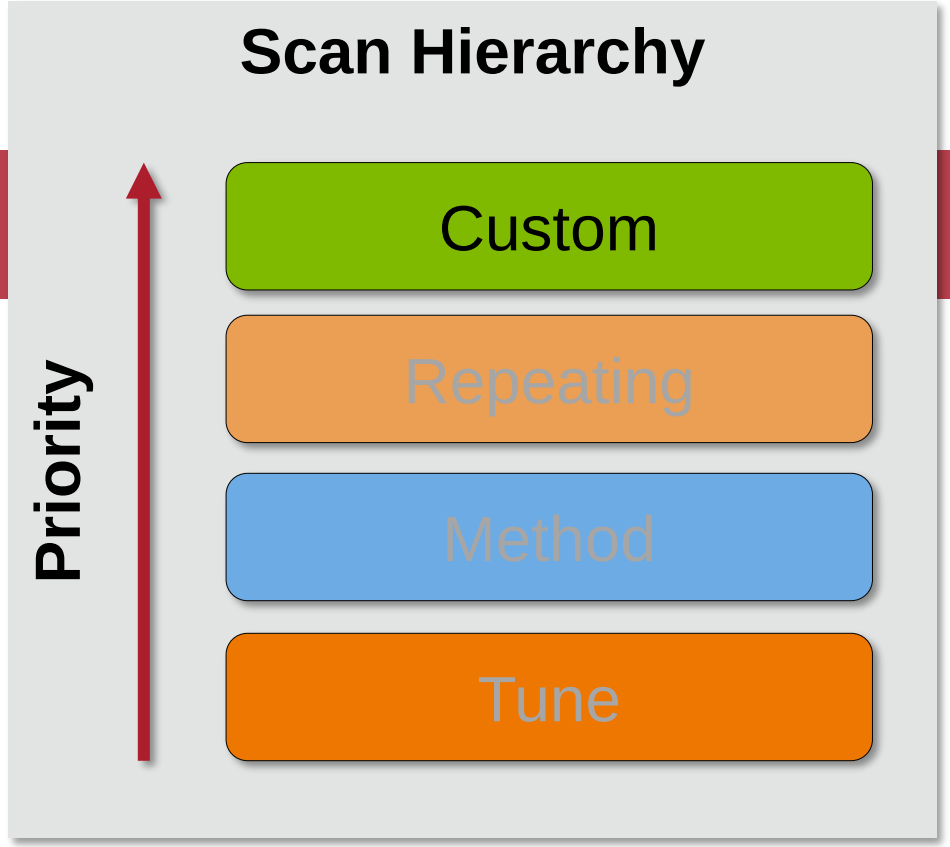
RunningNumber

Sending Scan to MS

Scan Management

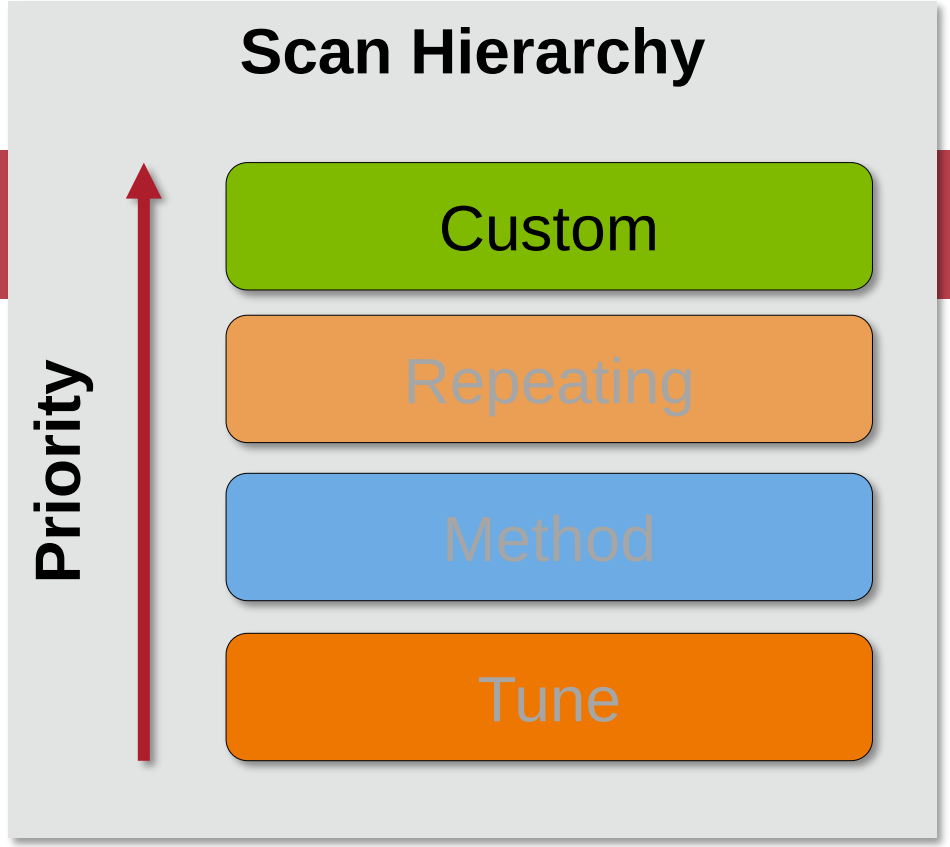


Scan Management - ICustomScan



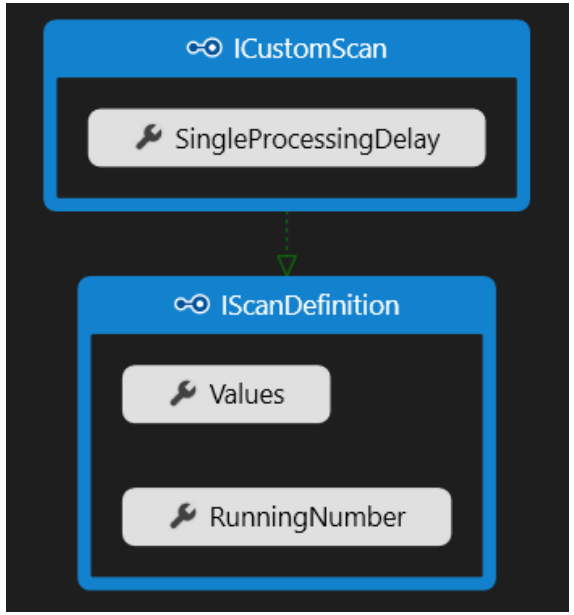
Single scan, drops to lower level after completion

Scan Management - ICustomScan



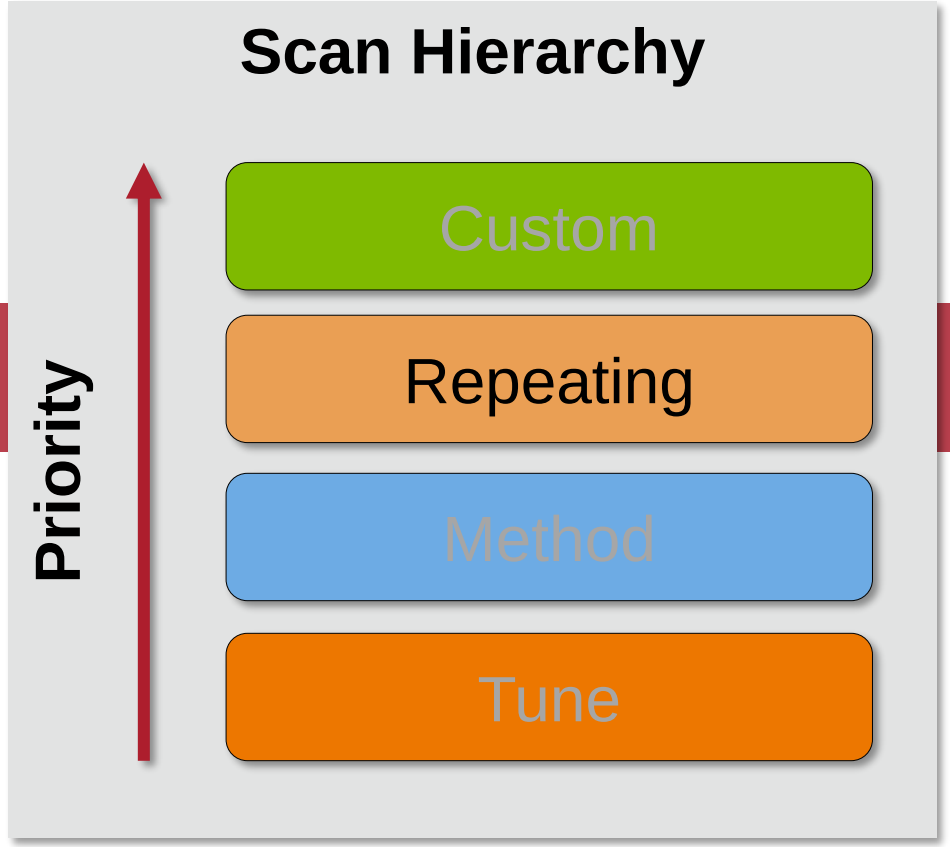
Single scan, drops to lower level after completion

- Once executed, waits the **Single Processing Delay** (s) for a new **Custom** scan to arrive*
- Default delay is 0 seconds (no waiting)
- Immediately ends the wait if a new **Custom** scan arrives or there are pending ones
  - The Tribrid uses a circular buffer (*40-capacity*) to store them



** Single Processing Delay is not enabled when occurring during Tribrid method acquisition, due to acquisition pipeline*

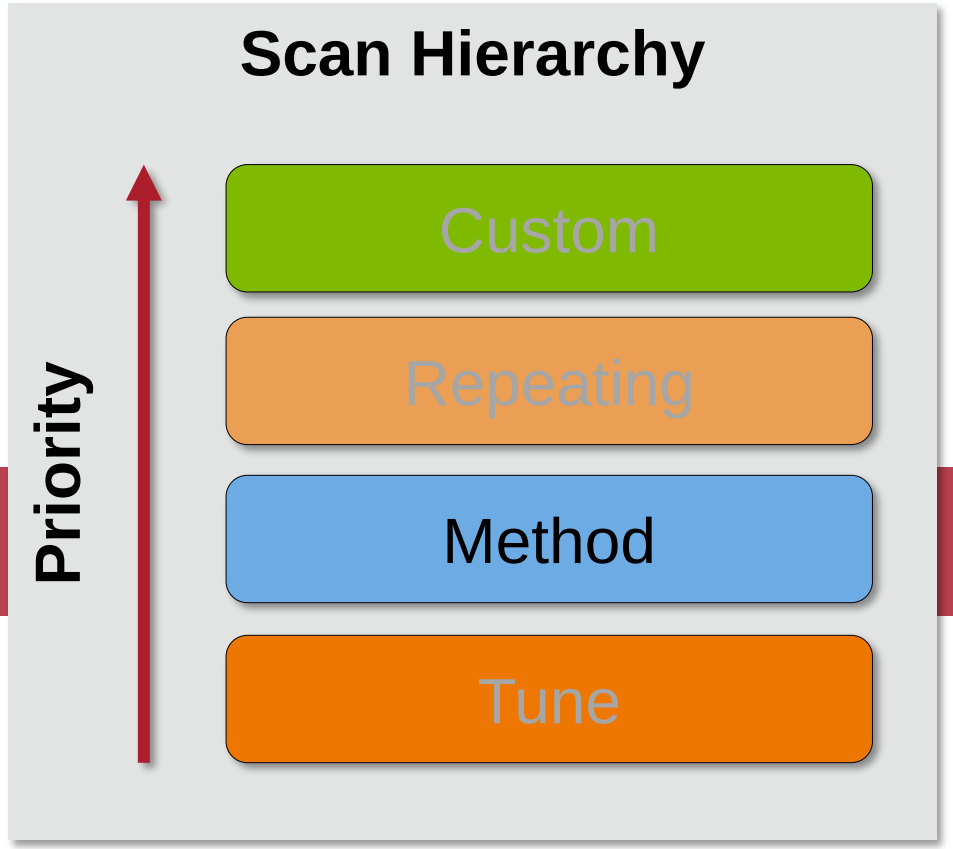
Scan Management - IRepeatingScan

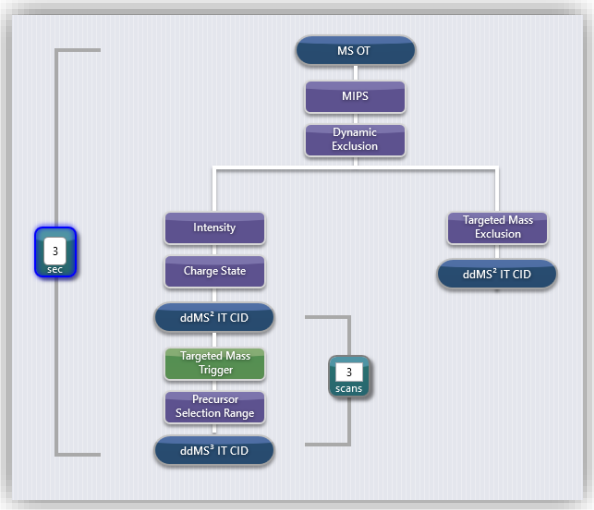


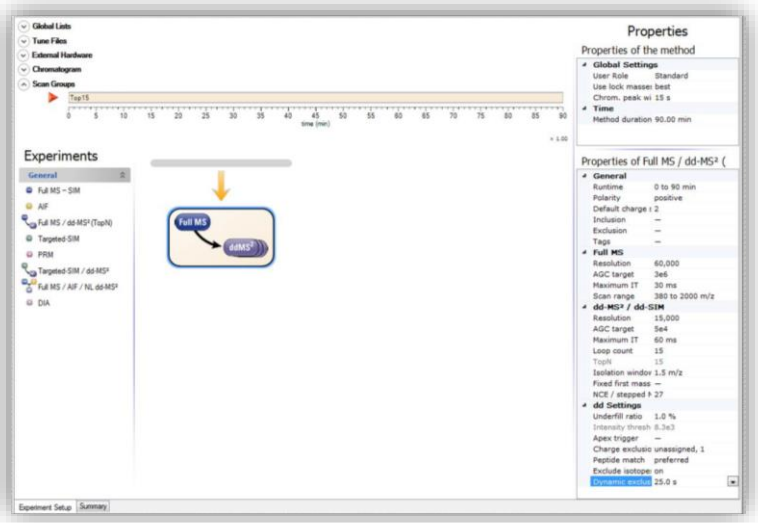
Repeats scan until explicitly canceled by user

- Continually runs the same scan until:
  1. Cancelled by the user
  2. Replaced by a new Repeating Scan
  3. Preempted by a Custom Scan

Scan Management - IRepeatingScan



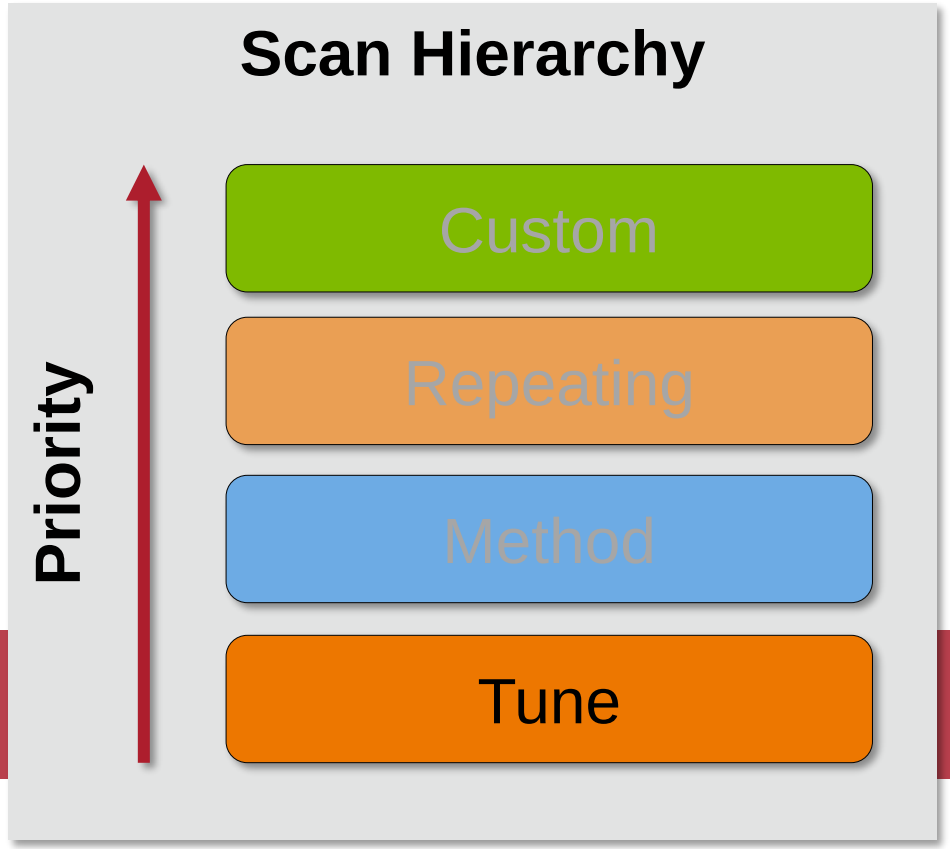


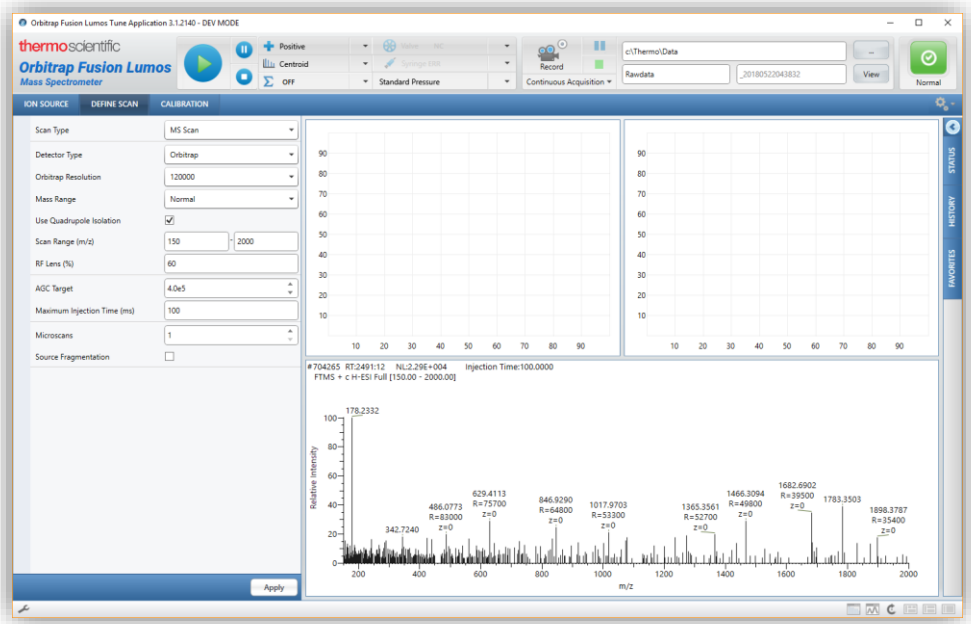


Runs Scans and logic defined in the method

- Runs the normal method logic until completion or new IAPI scan arrives

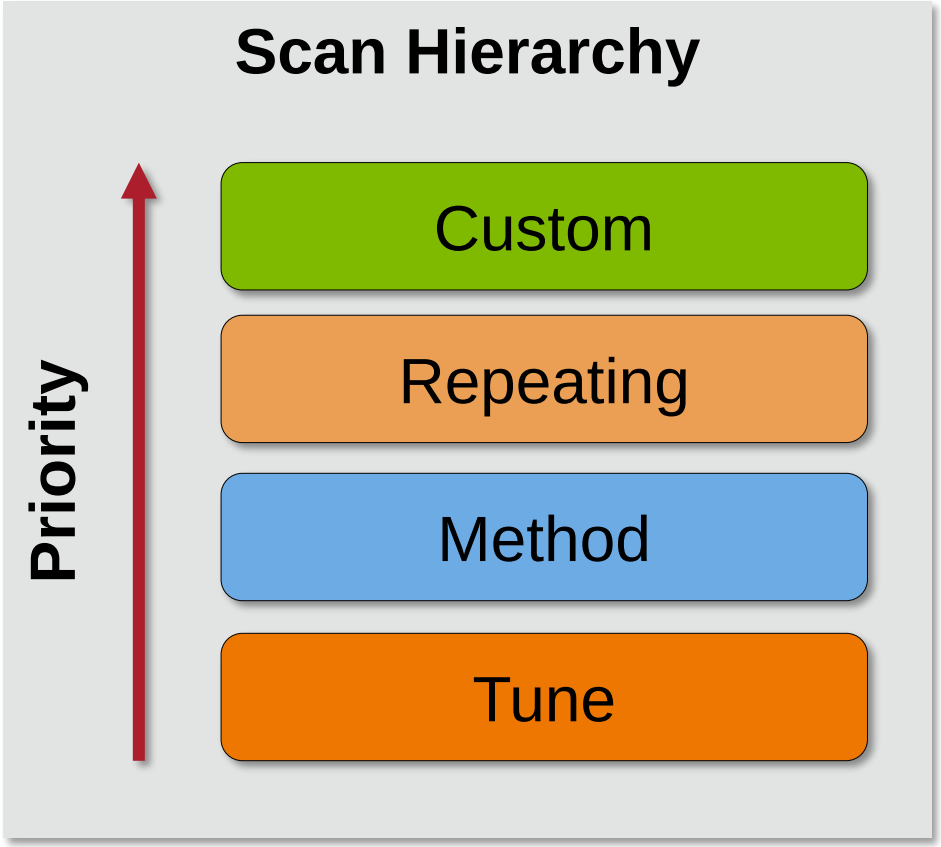
Scan Management - IRepeatingScan

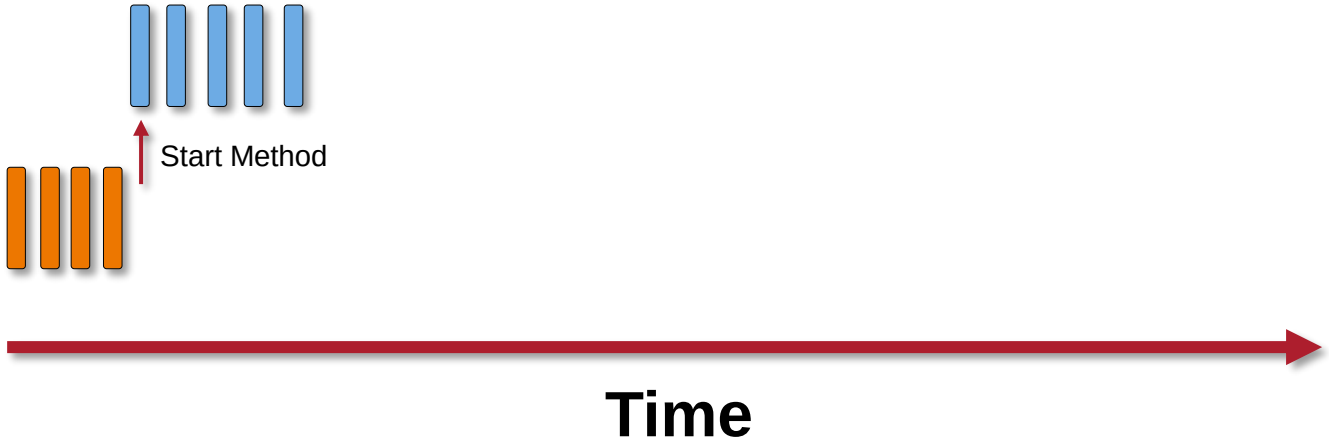




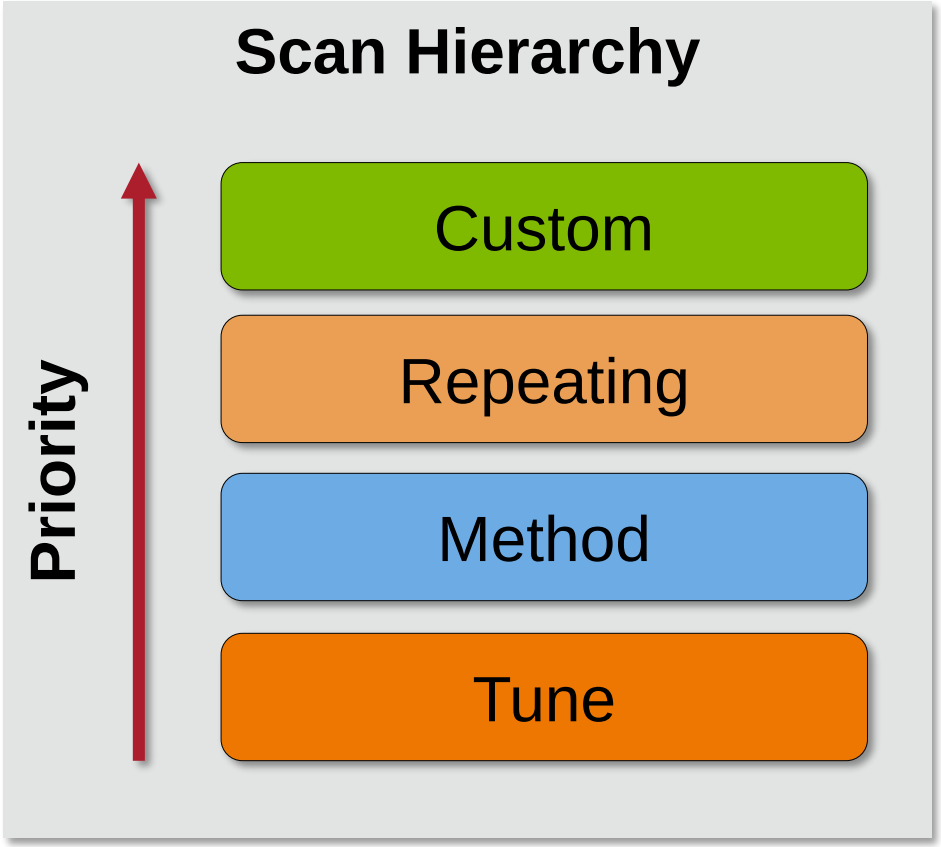
Runs the Scan defined in Tune

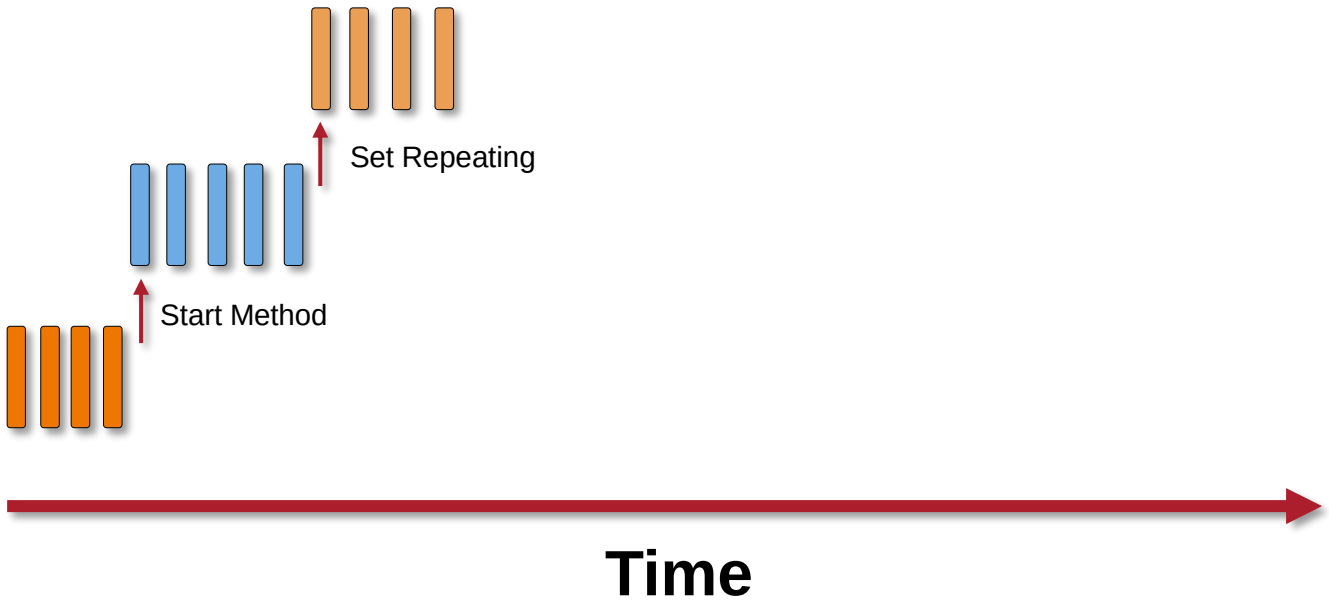
Scan Management



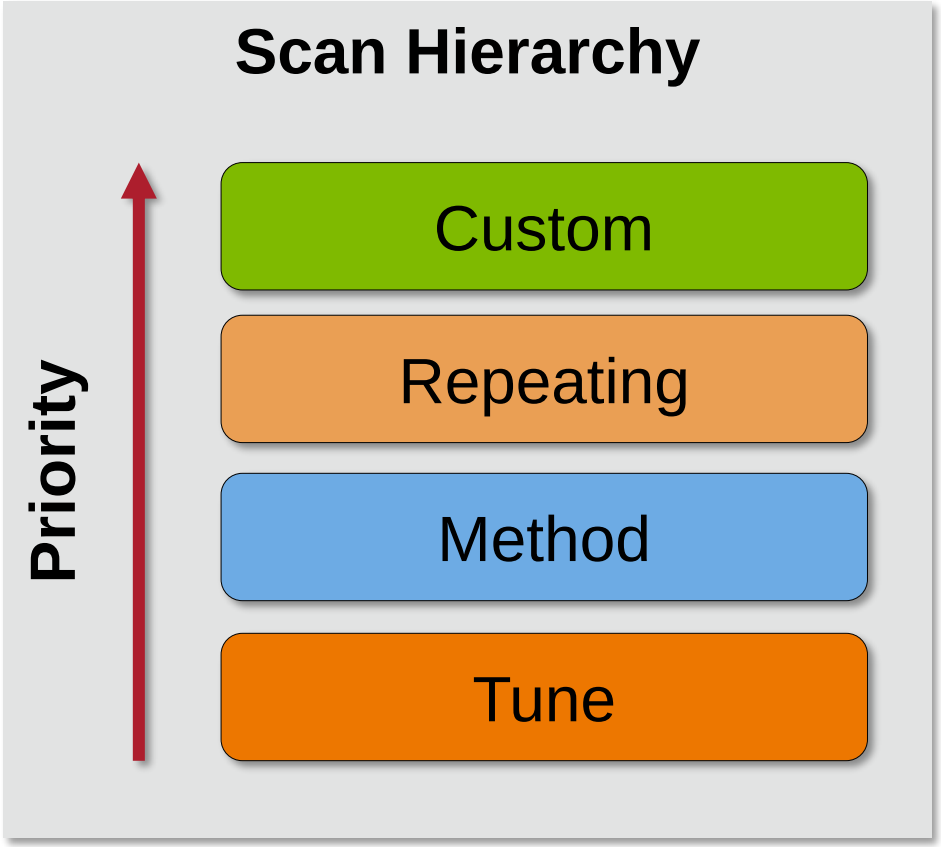


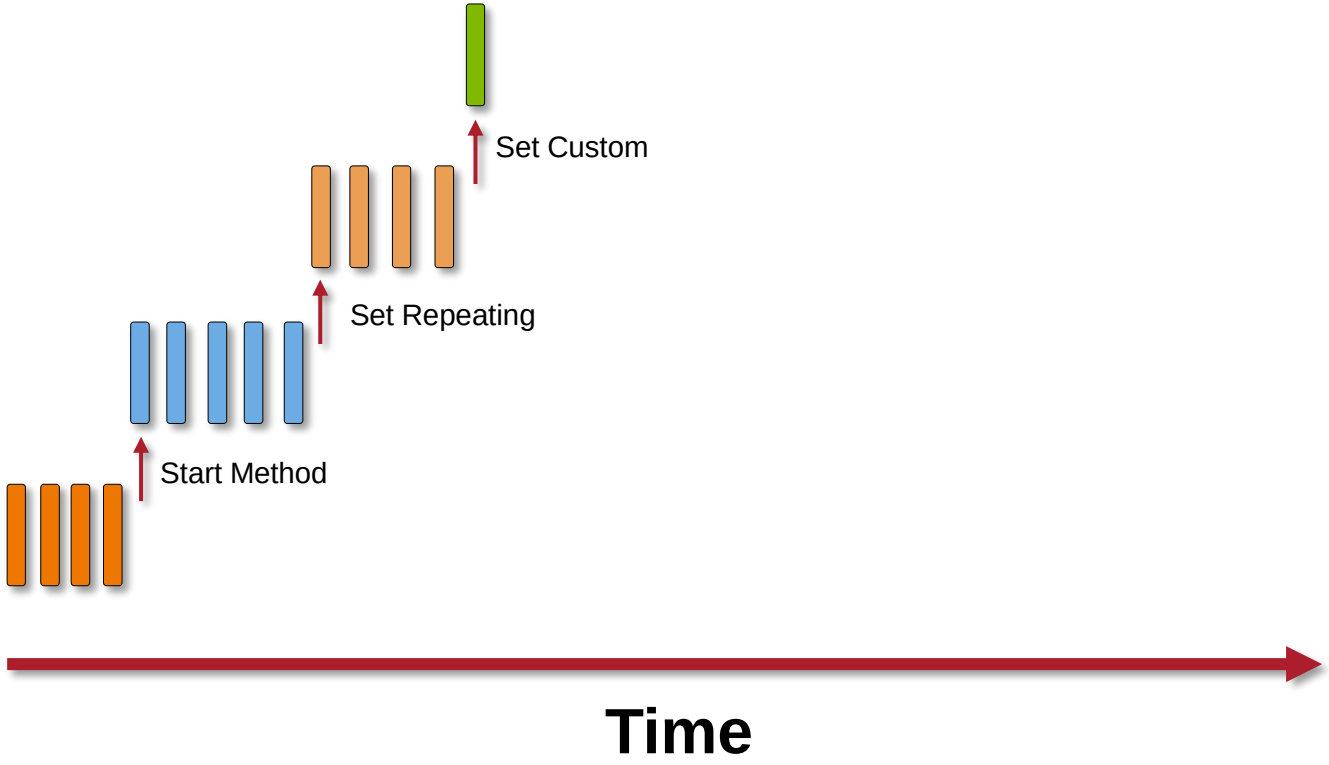
Scan Management



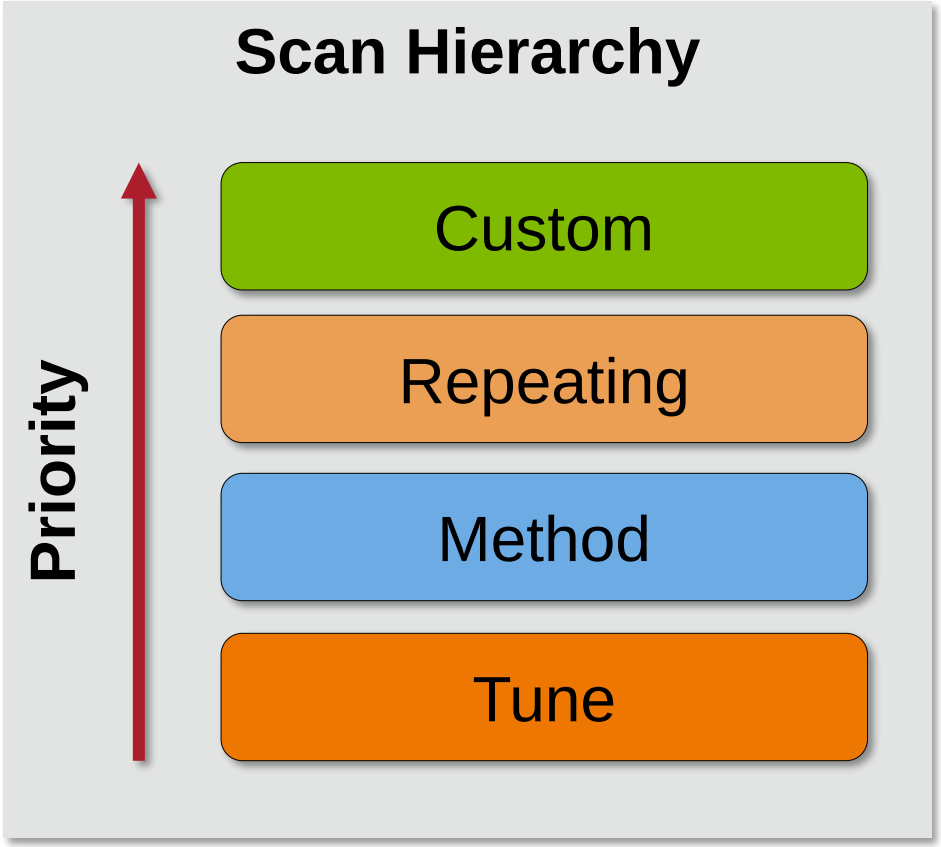


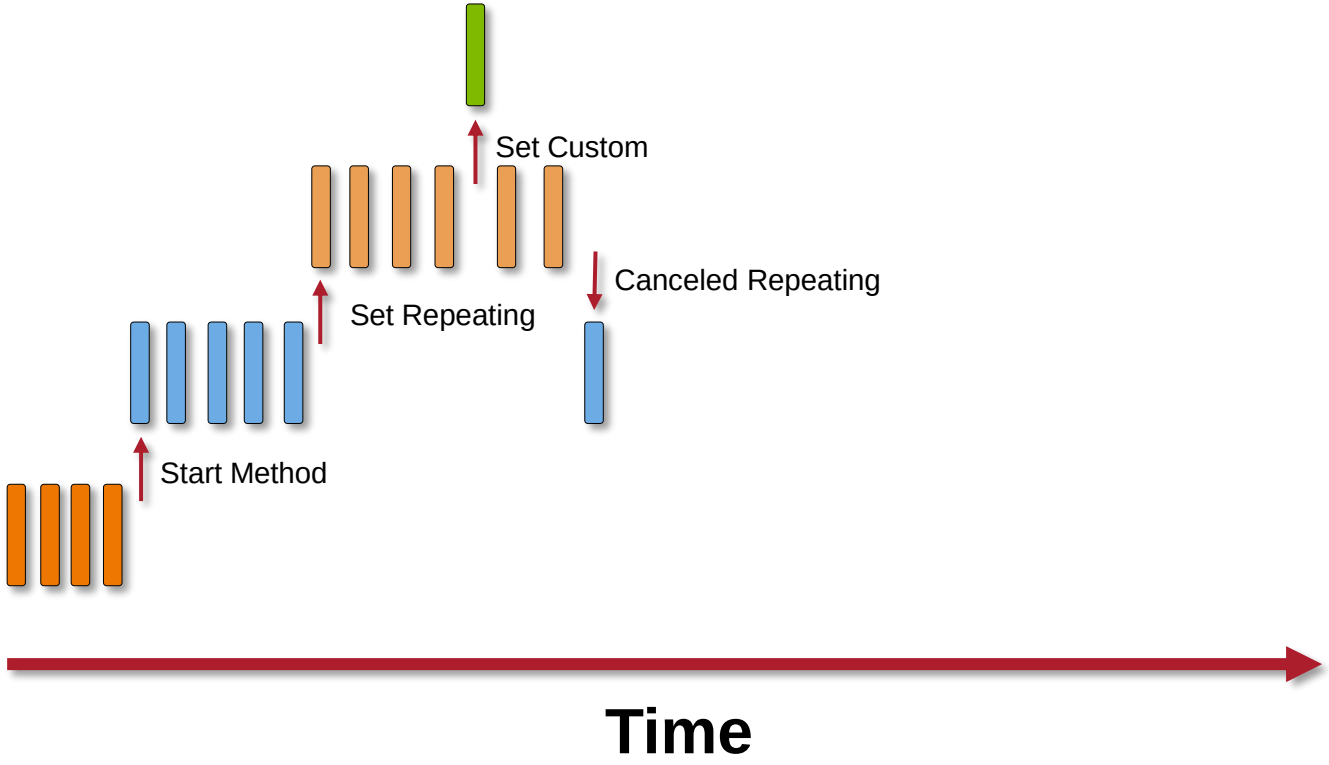
Scan Management



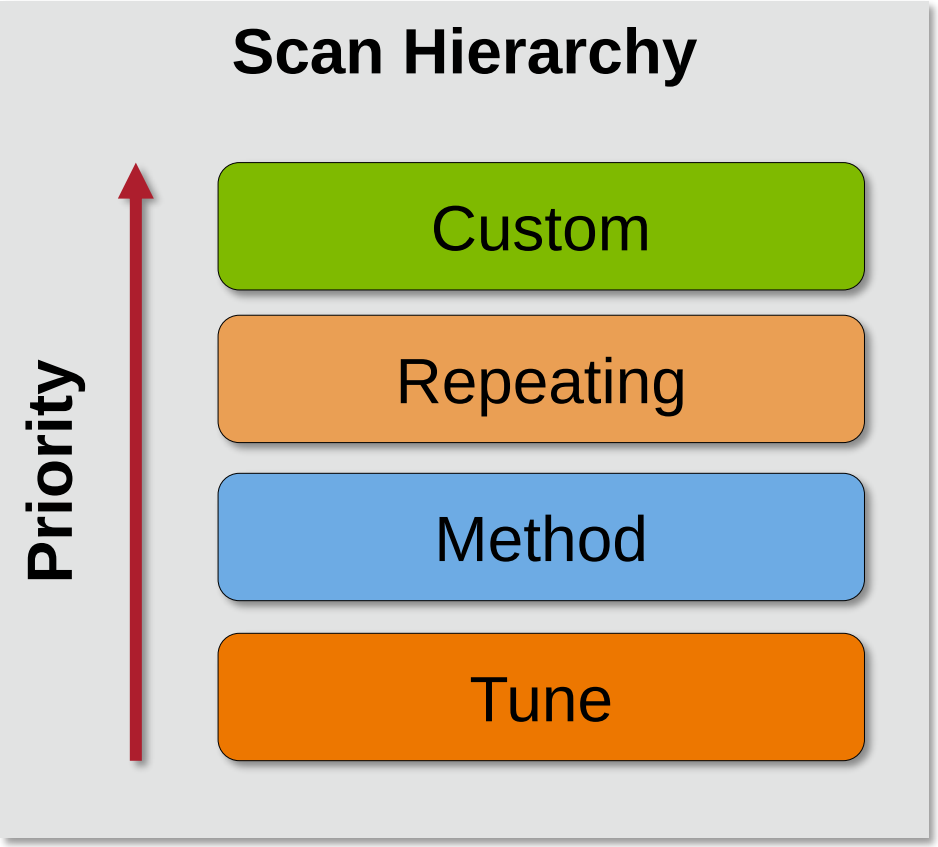


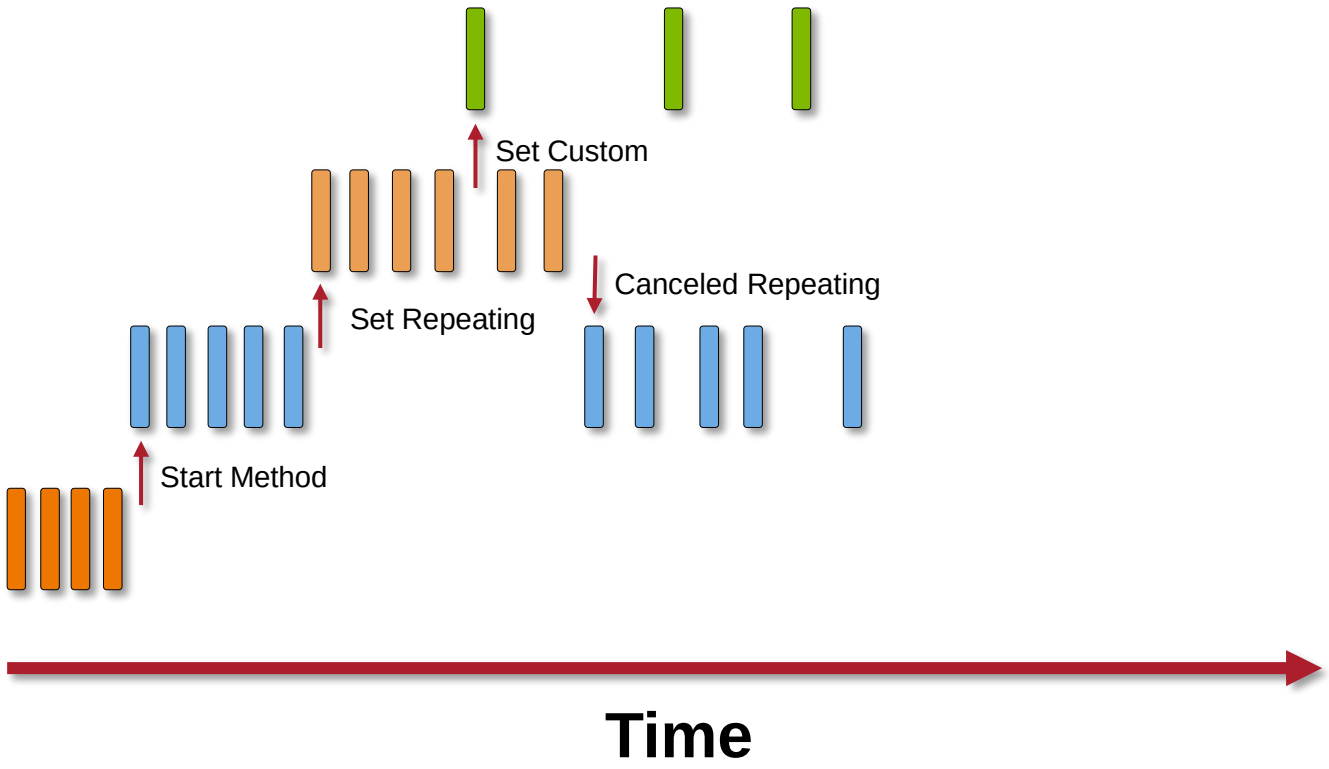
Scan Management



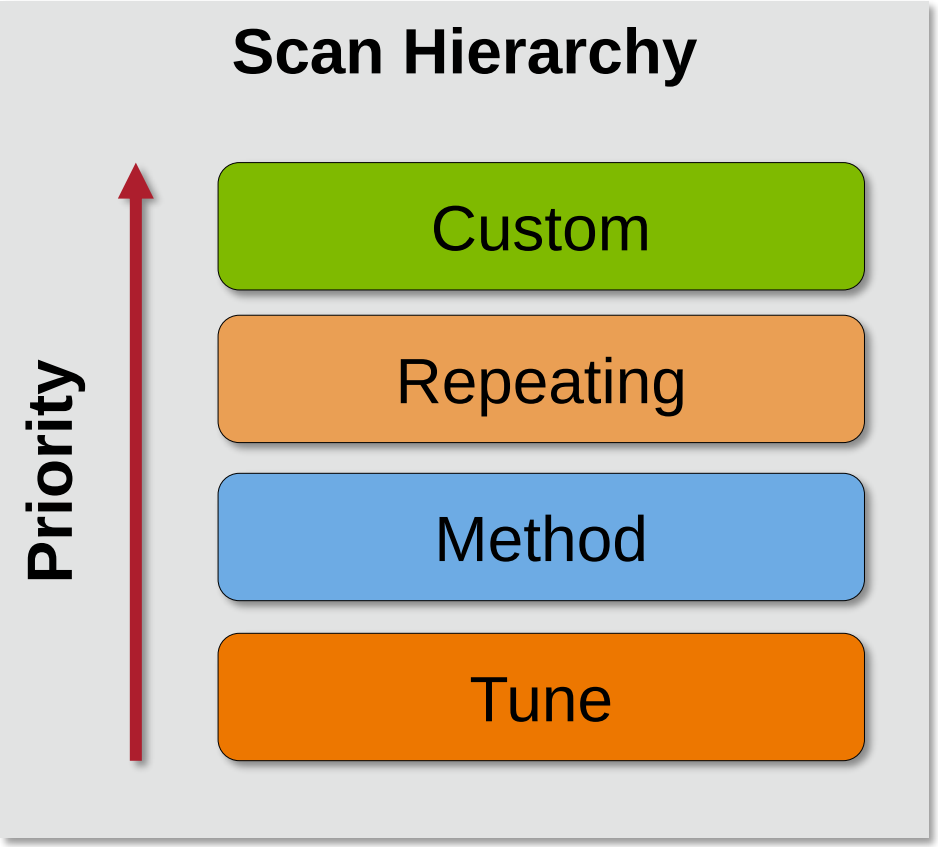


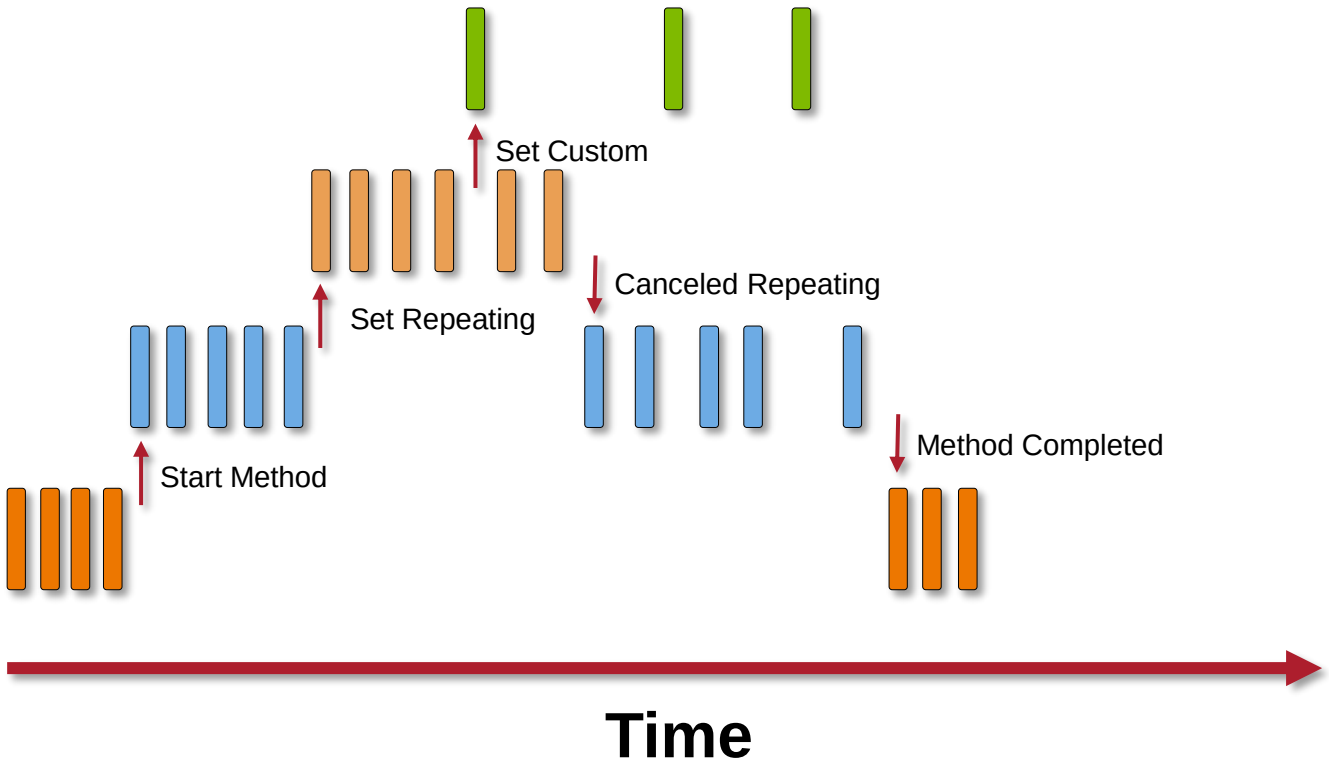
Scan Management



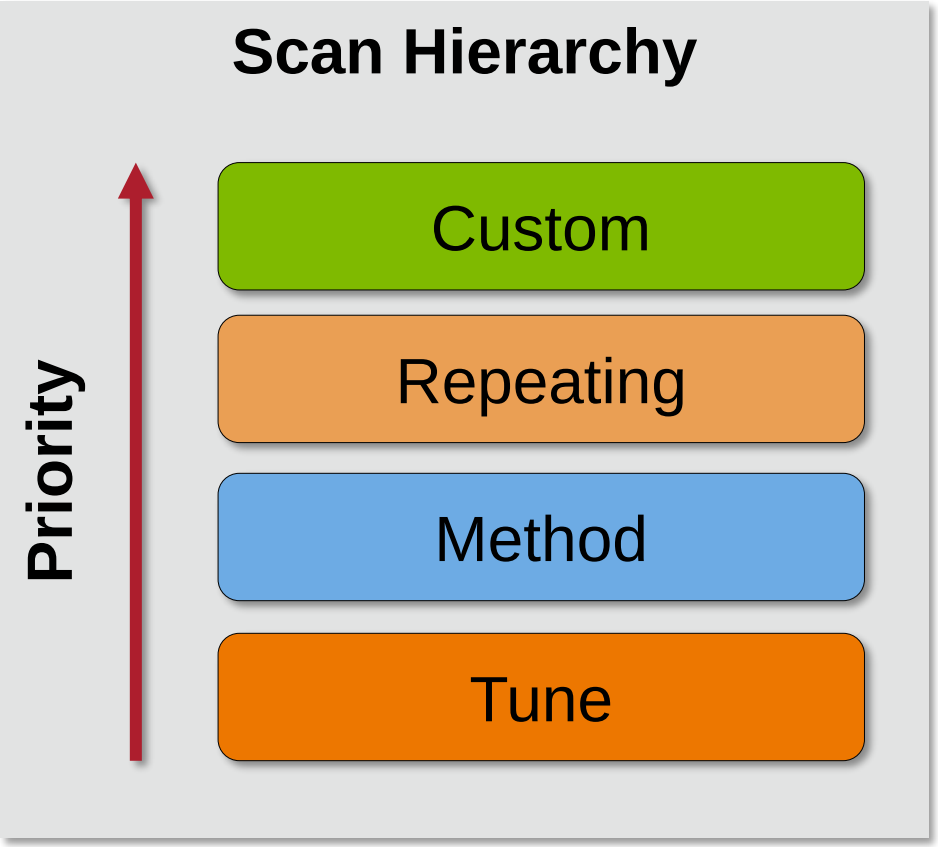


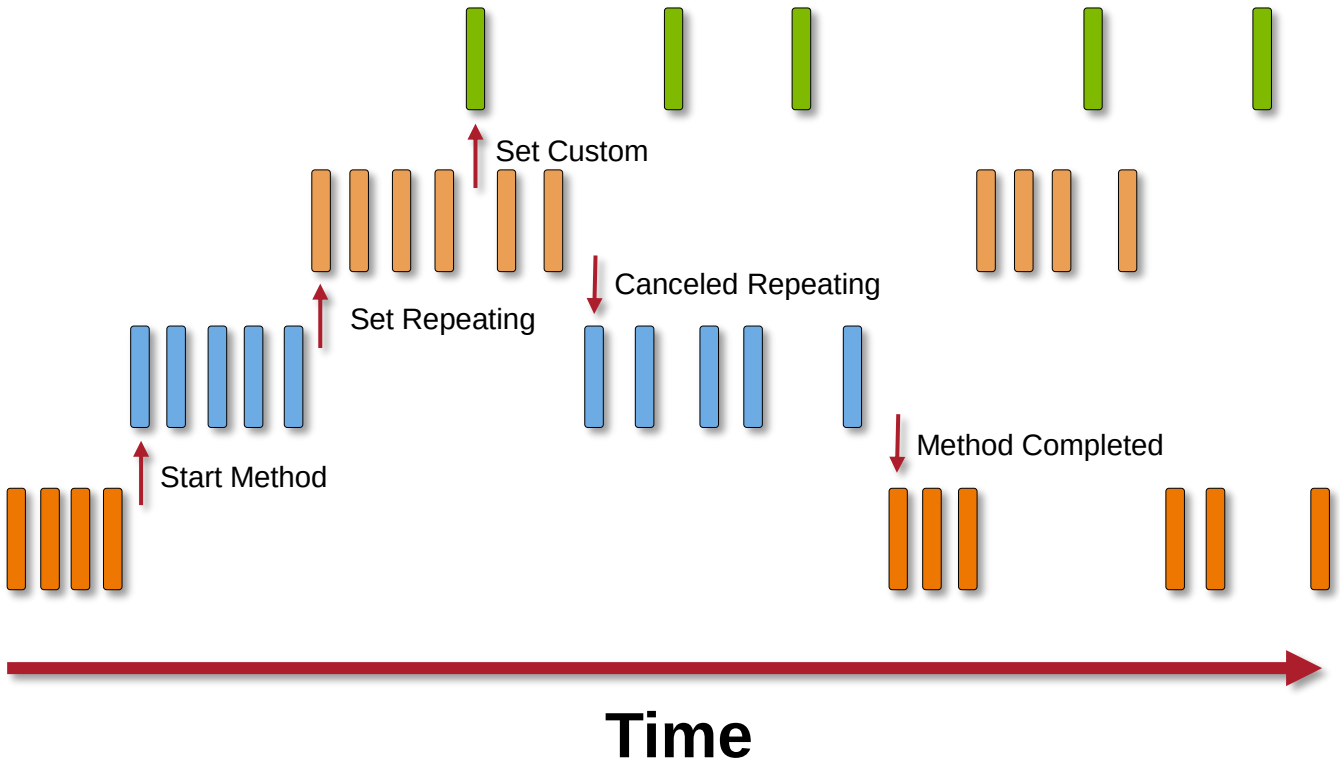
Scan Management





Scan Management



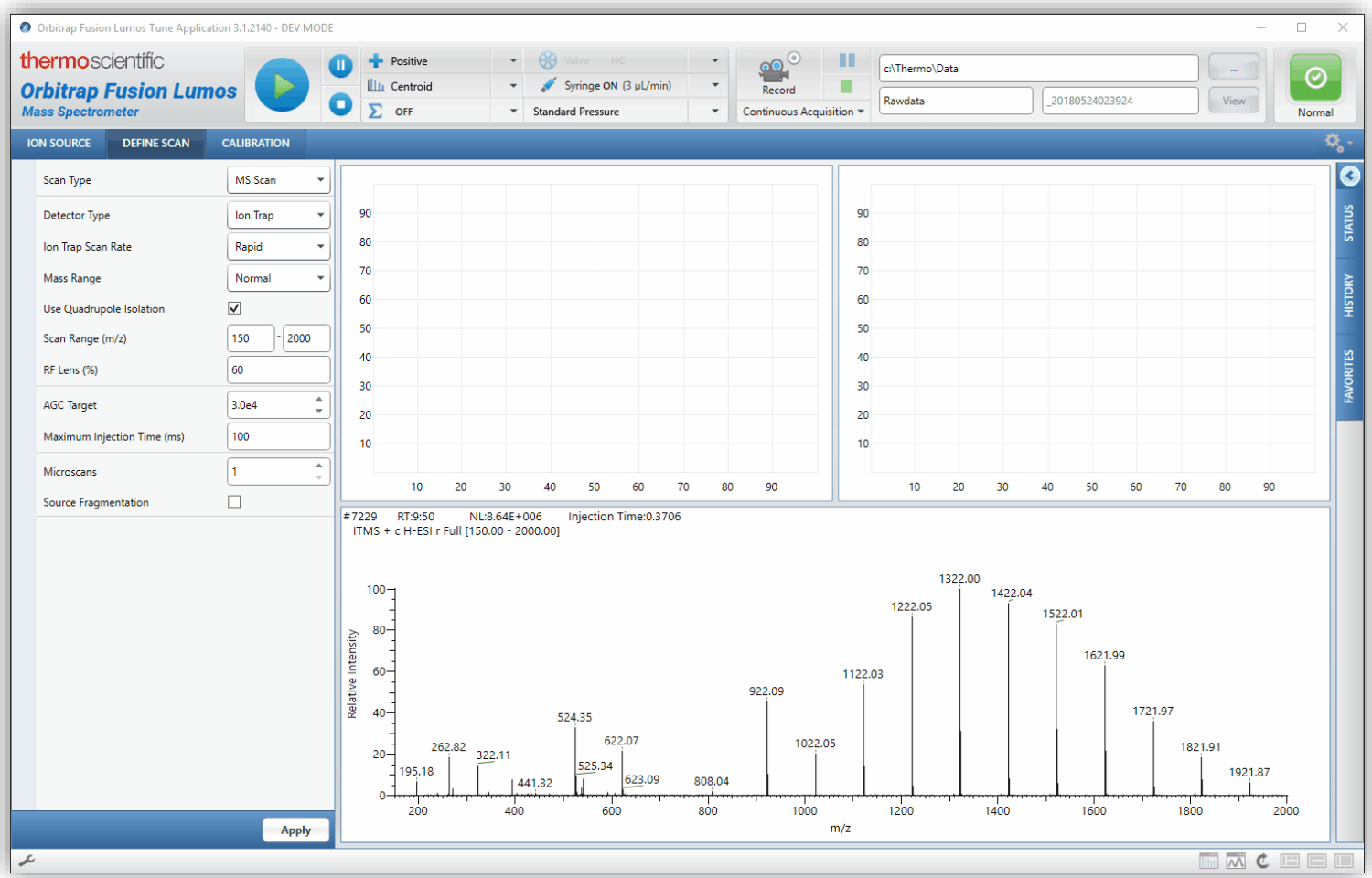


Scan Management

Scan Hierarchy

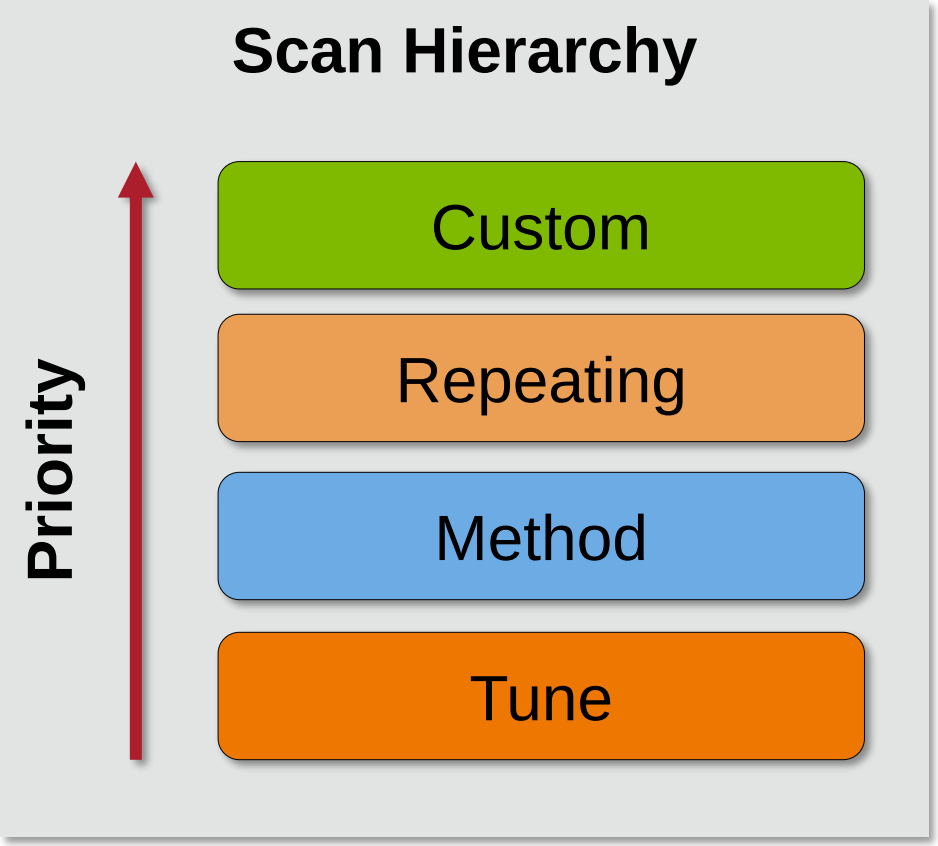
Priority

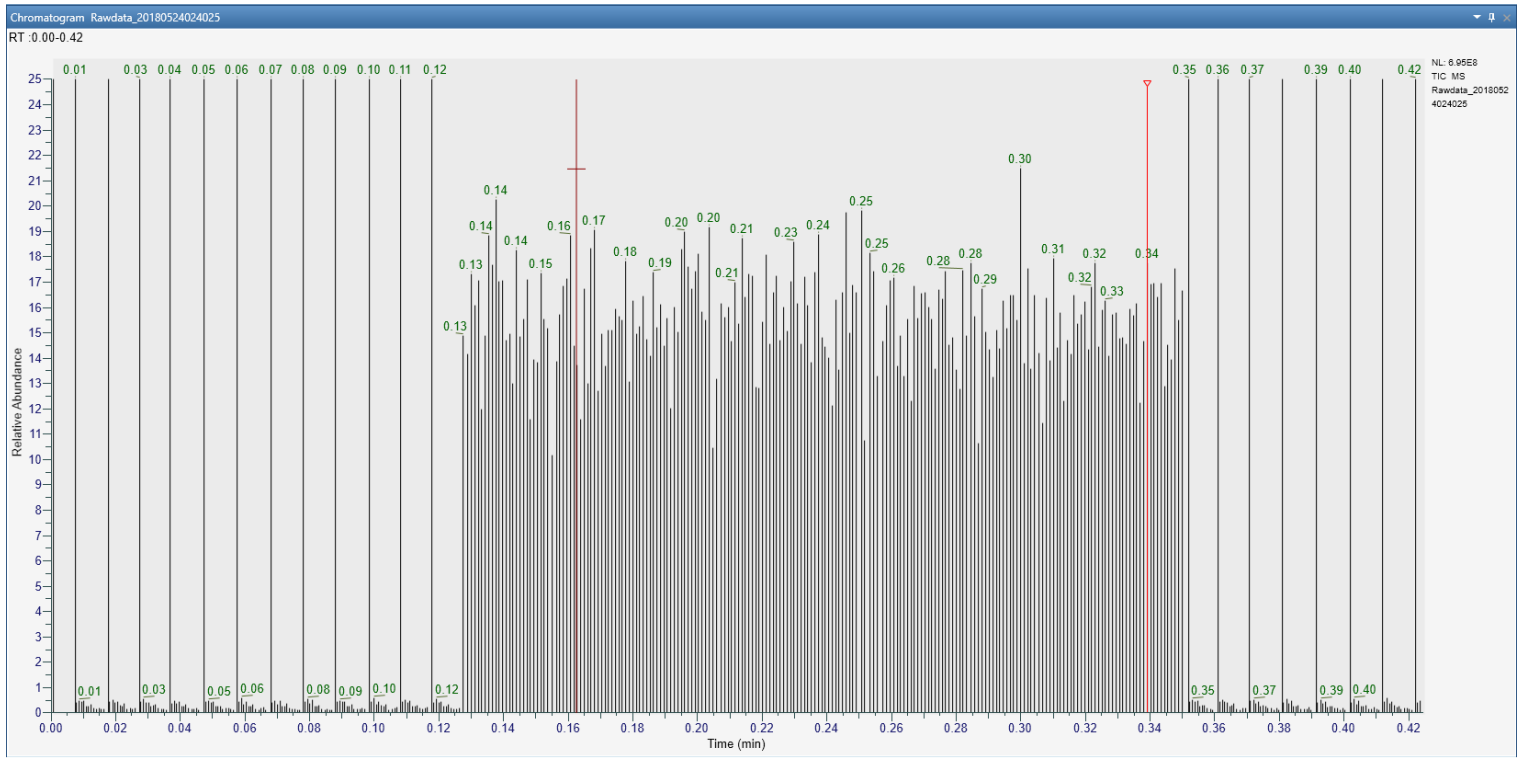
- Custom
- Repeating
- Method
- Tune



ThermoFisher SCIENTIFIC

Scan Management

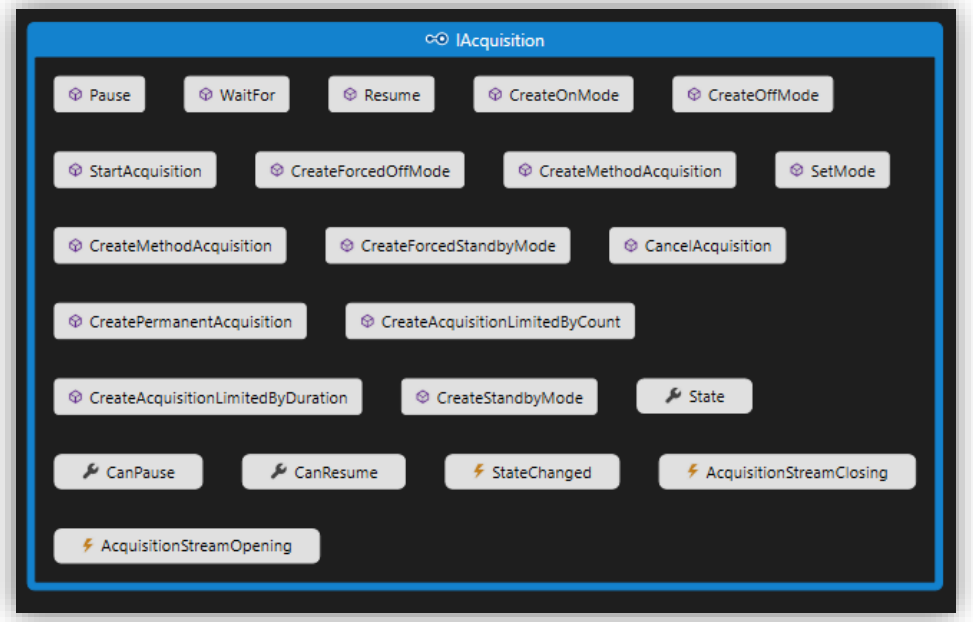




Method Scans | **IAPI Scans** | **Method Scans**

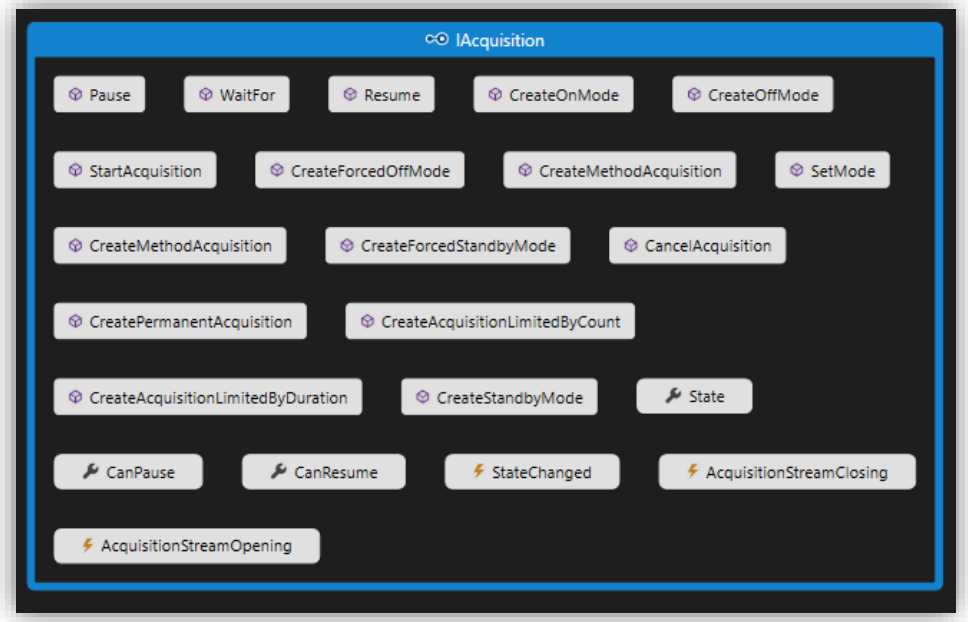
Starting Acquisitions from the IAPI

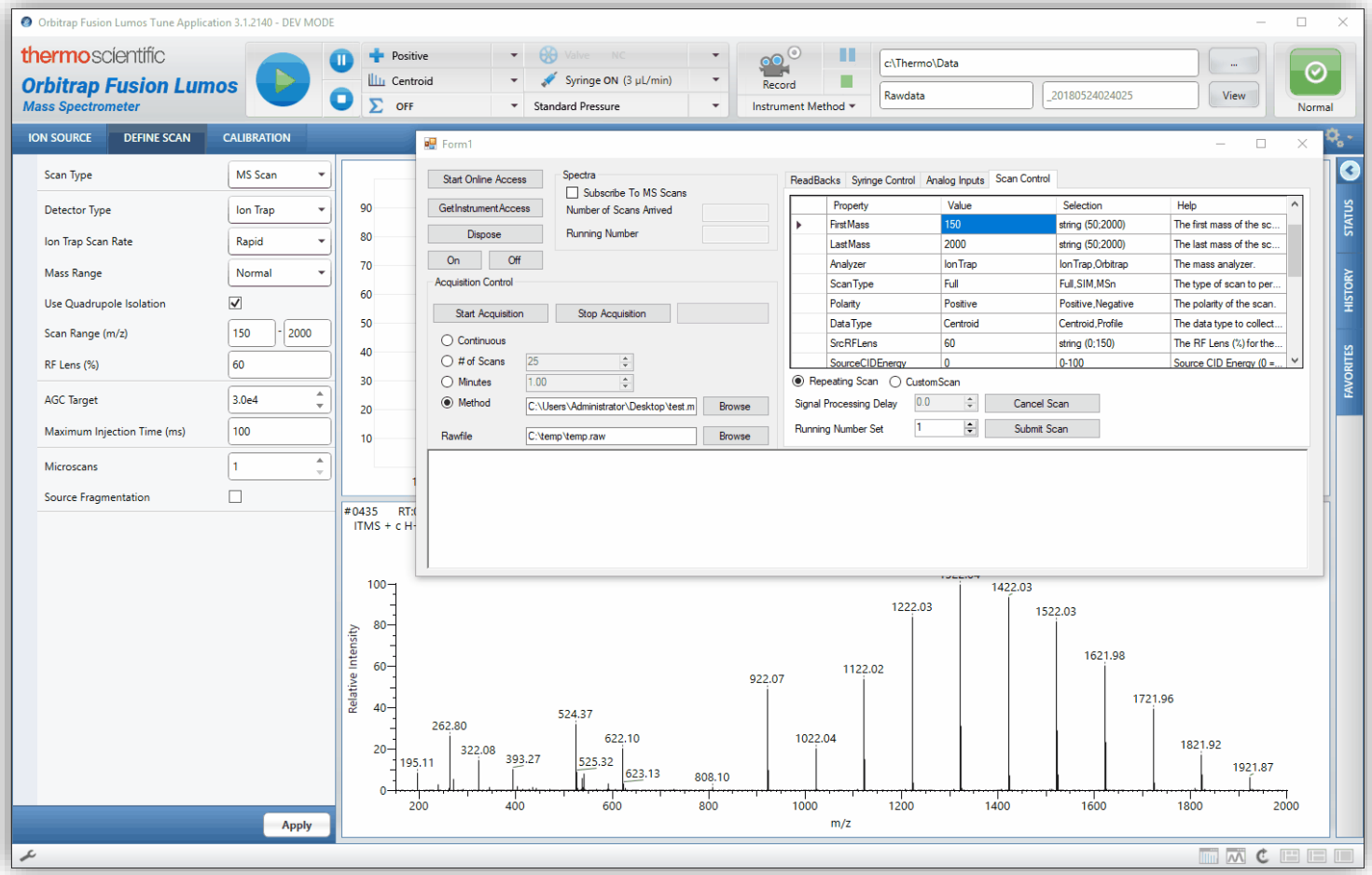
- The **IAcquisition** interface enables the IAPI to start/stop acquisitions
- It also has **Events** for listening to when an acquisition starts and stops



Starting Acquisitions from the IAPI

- The **IAcquisition** interface enables the IAPI to start/stop acquisitions
- It also has **Events** for listening to when an acquisition starts and stops





ThermoFisher SCIENTIFIC

IAPI Outline

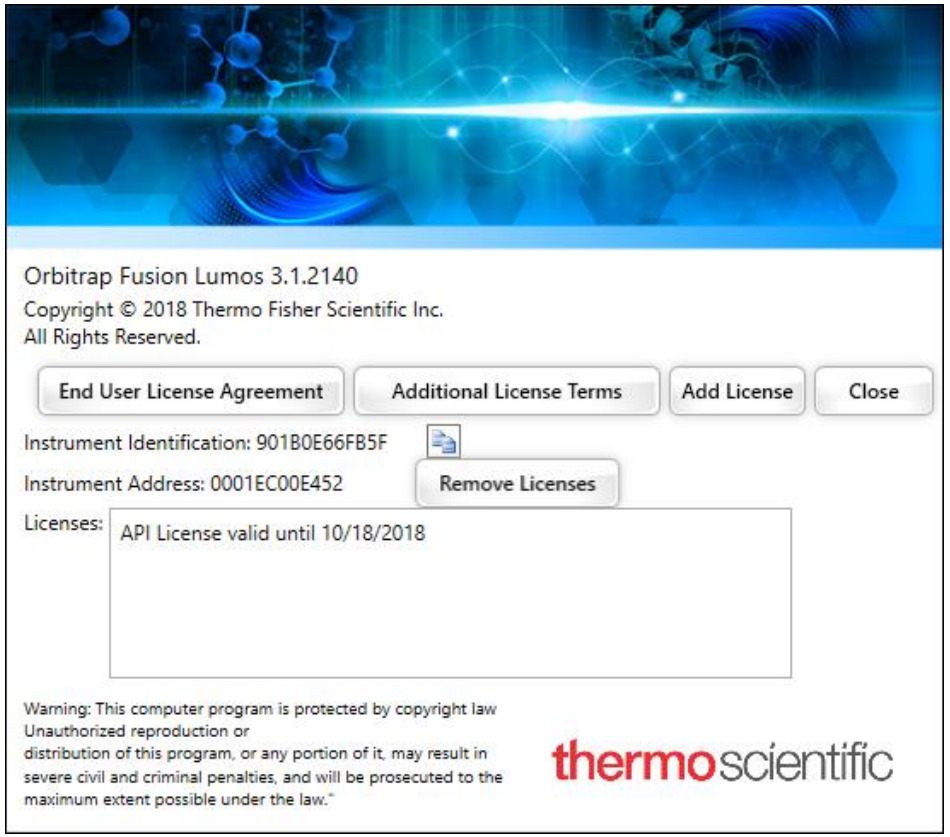
1. **IAPI Architecture**
   - Interfaces
   - Data and Control Flow
2. **Receiving Data from the MS**
   - Interfaces
   - Scan Data Stream Subscription
3. **Controlling the MS**
   - Sending Scan Definitions
   - Changing other MS parameters
4. **Getting Started**

License Activation

- Access to the IAPI is given after a fully executed IAPI License is in force.
- Contact ThermoMSLicensing@thermo.com to get started
  - Please include which IAPI (**Tribrid** or **Exactive**) you would like access to in your initial correspondence

License Activation

- Access to the IAPI is given after a fully executed IAPI License is in force.
- Contact ThermoMSLicensing@thermo.com to get started
  - Please include which IAPI (**Tribrid** or **Exactive**) you would like access to in your initial correspondence

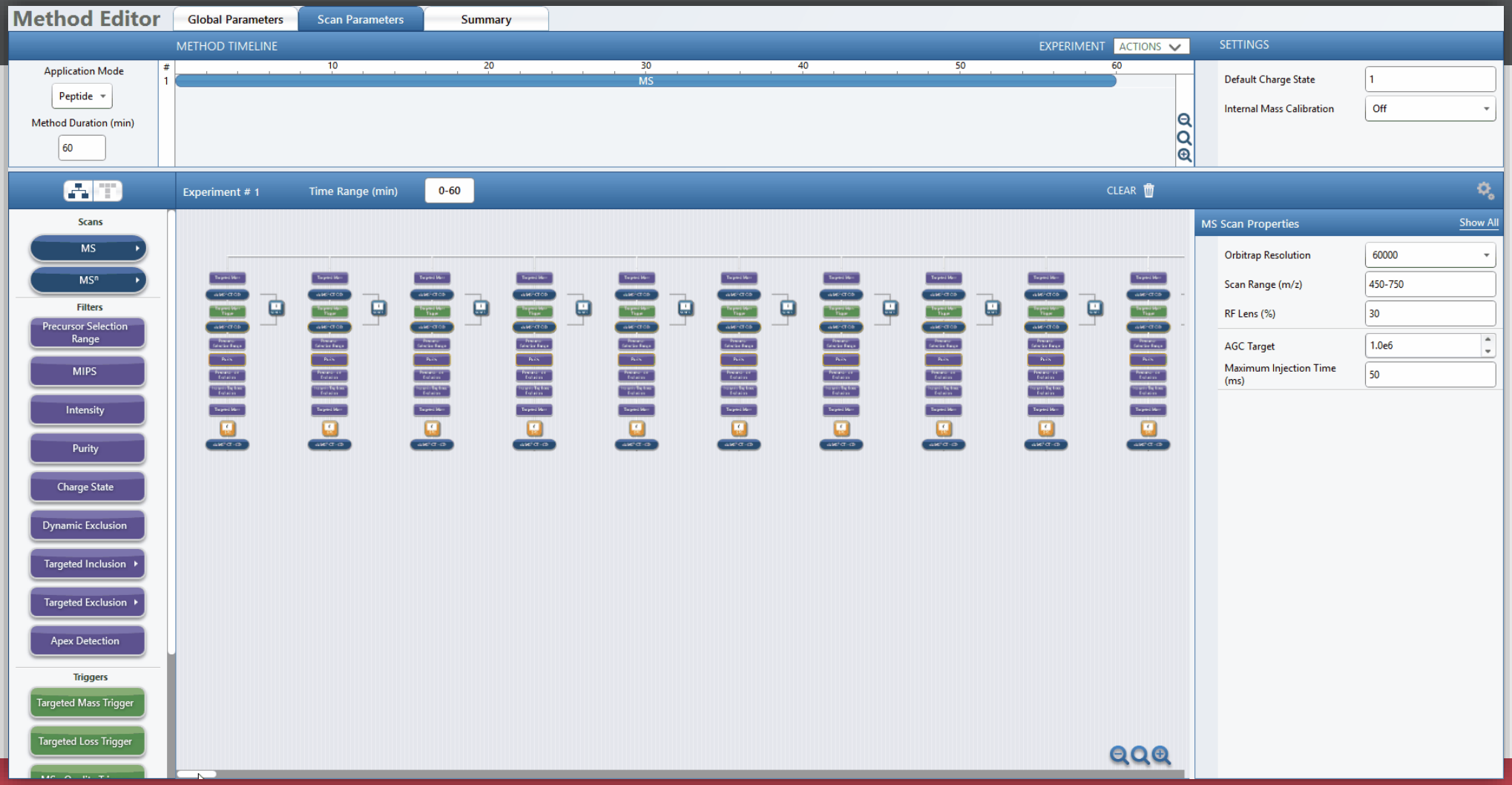


Activating the License on the Tribrids

- A license key will be provided to you and is entered in Tune's "About Tune" dialog
- Restart both the Instrument and Data system computer and the API will be enabled

Overview

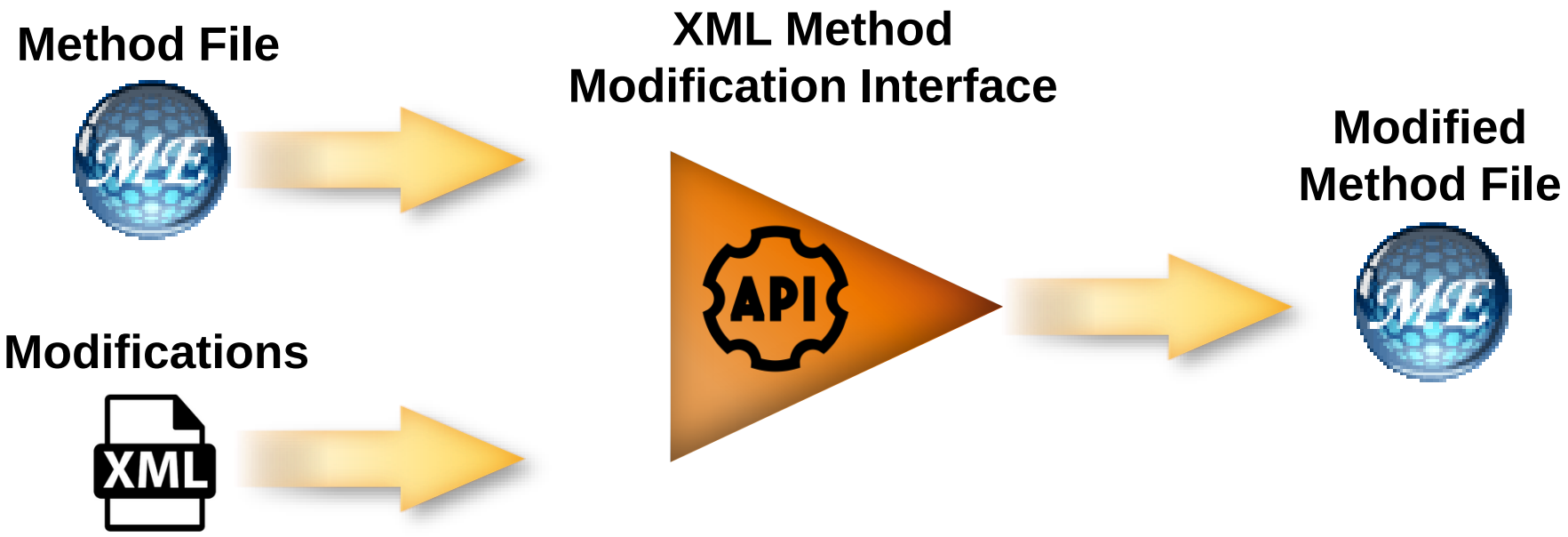
1. **Instrument Application Programming Interface (IAPI)**
   - Real-time control and acquisition of Orbitrap Mass Spectrometers using .NET
2. **XML Method Modification Interface (XMMI)**
   - Programmatically modify Fusion-series method files
3. **Applications of IAPI**
   1. **Devin Schweppe** – Fusion API
   2. **Christoph Wichmann** – Q Exactive API
4. **Examples and FAQ for IAPI and XMMI**
5. **Applications of XMMI**
   1. **Pavel Shliaha** – Fusion Method Creation



The Method Editor offers tremendous **flexibility** in method creation; sometimes manually editing complex methods is laborious.

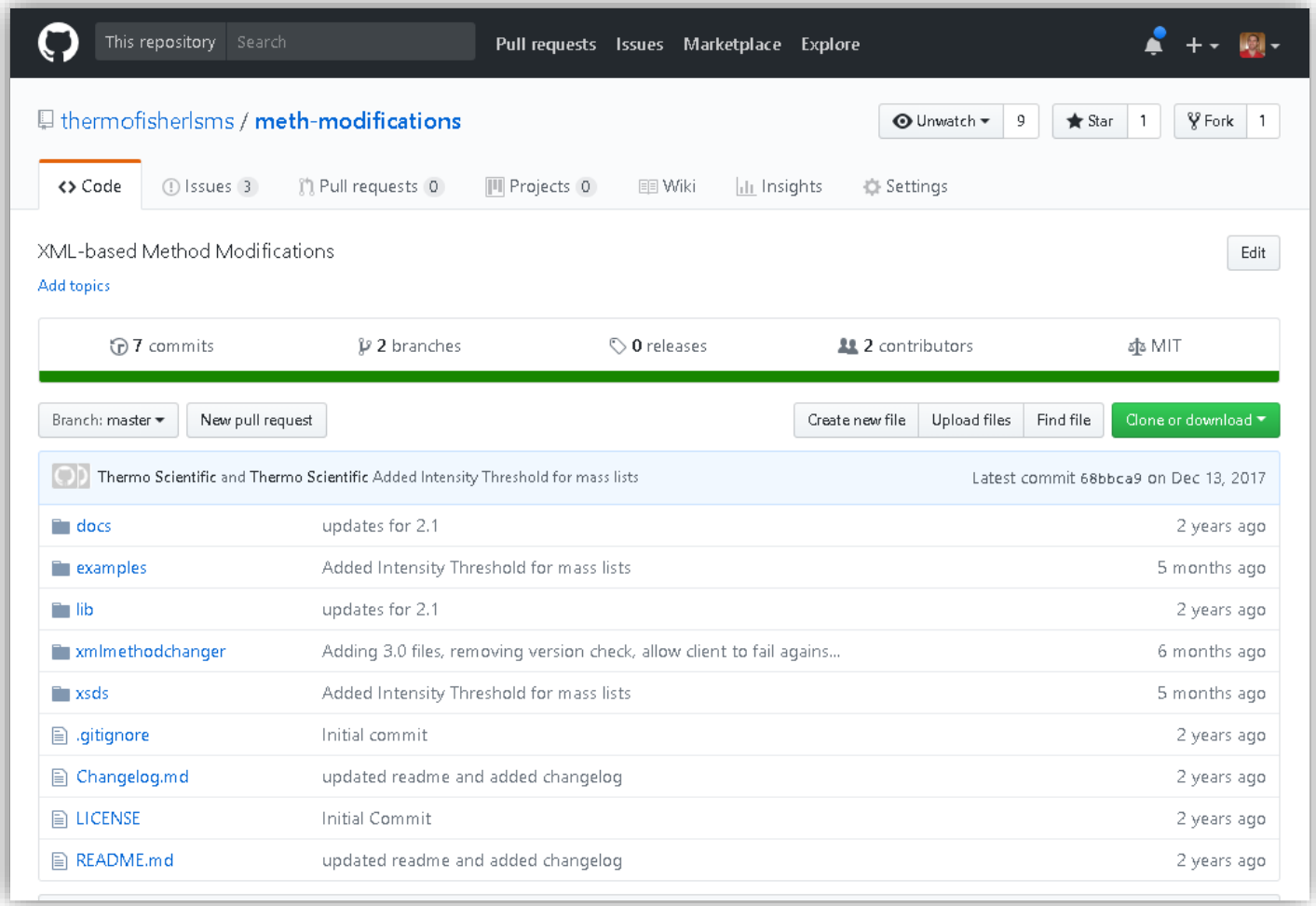
Method courtesy of Christopher Rose

Programmatically Modify Fusion Methods



Method Modifications Git Repository

https://github.com/thermofisherlsms/meth-modifications

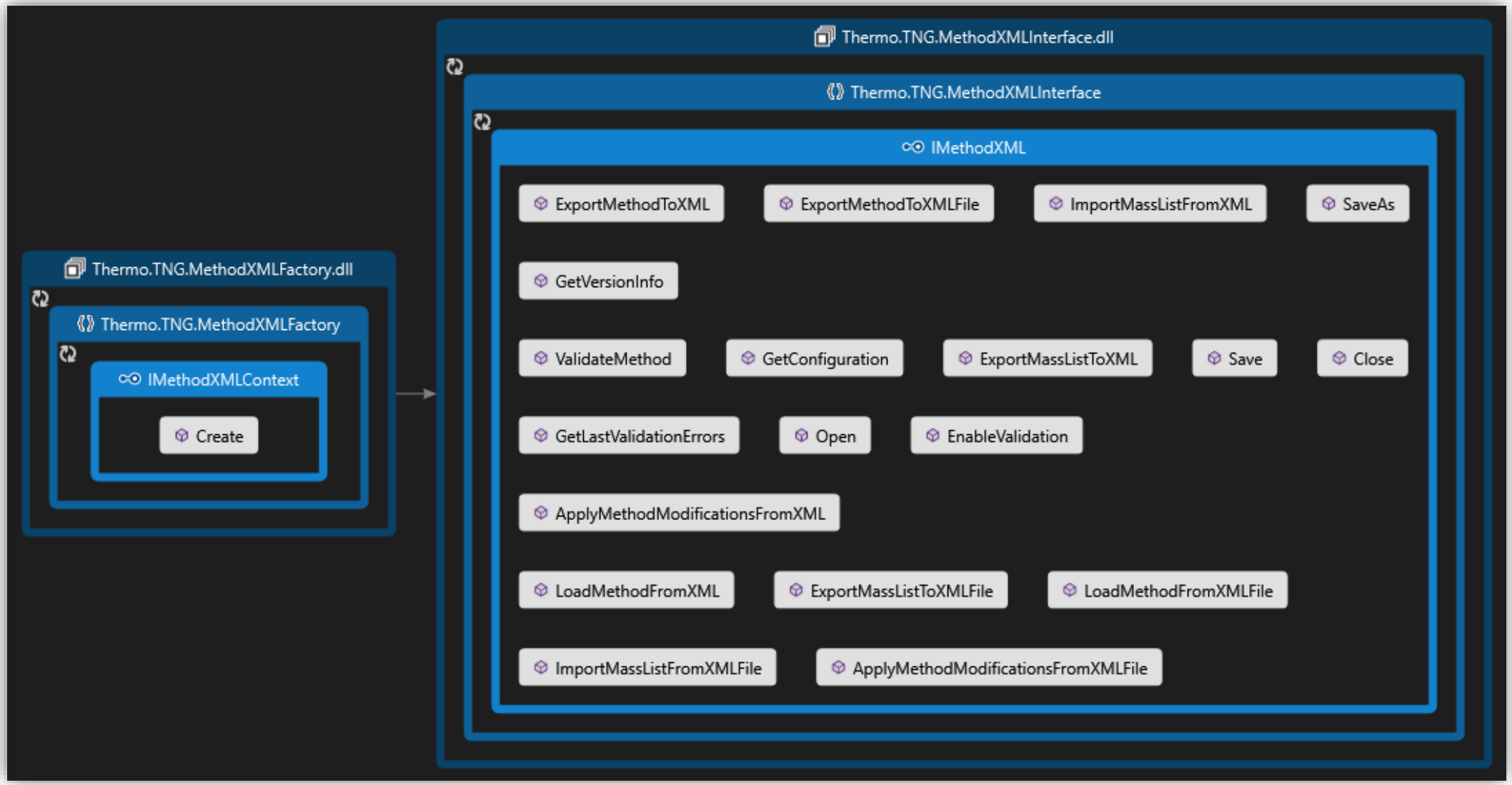


- Supports Fusion and Triple-Quad Methods
  - Versions **1.2, 2.0, 2.1, 3.0** and **3.1**
- Documentation
- Complete Examples
- VS Solution
  - Command Line Program
  - Heavily commented code
- Schemas

External Requirements

- Tune Installation
  - Full or Workstation
- .NET 4.6.2+

XML Modification Interfaces

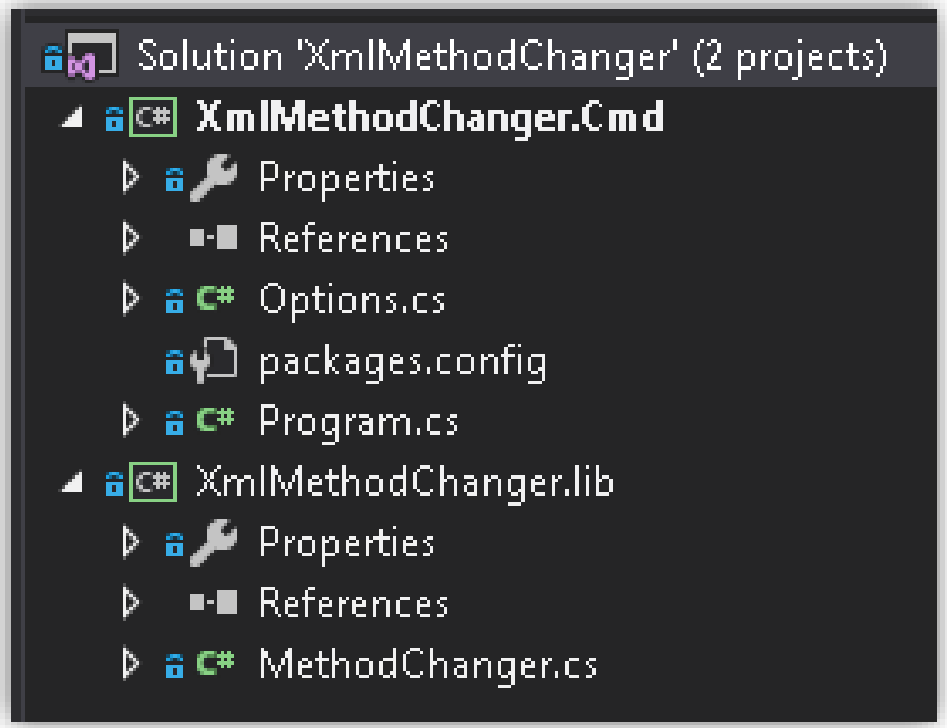


Complete Code Example

```
1   using Thermo.TNG.MethodXMLFactory;
2   using Thermo.TNG.MethodXMLInterface;
3
4   string templateMethod = @"Files/Fusion/DIA/DIATemplate.meth";
5   string modificationXML = @"Files/Fusion/DIA/DIA.xml";
6   string ouputMethod = @"Files/Fusion/DIA/DIA_output.meth";
7
8   string instrumentModel = "OrbitrapFusion"; // alternatively "TSQEndura" or "TSQQuantiva"
9   string instrumentVersion = "1.2"; // just the version number as a string
10
11  using(IMethodXMLContext mxc = MethodXMLFactory.CreateContext(instrumentModel, instrumentVersion))
12  using(IMethodXML mx = mxc.Create()) {
13
14      mx.Open(templateMethod); // Open the template method file
15
16      mx.ApplyMethodModificationsFromXMLFile(modificationXML); // Apply the xml modifications
17
18      mx.SaveAs(outputMethod); // Save the method in memory to disk
19  }
20
```

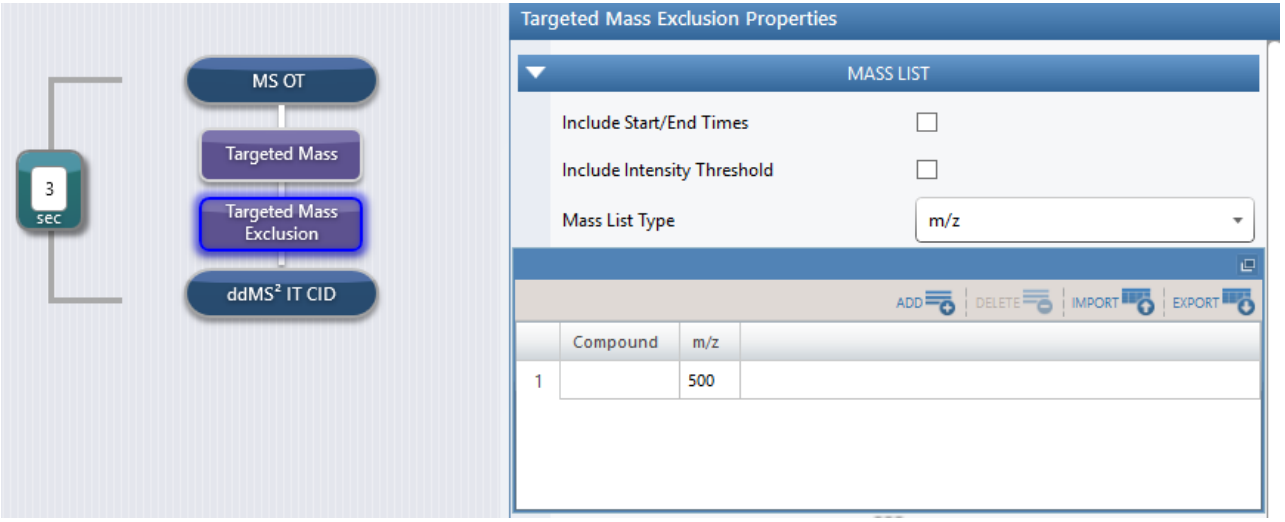
Method Modifications Visual Studio Solution

- The **XmlMethodChanger** Solution contains two projects
  1. XmlMethodChanger.lib
     - A public, static class library (**MethodChanger)**
     - Wraps the two Thermo.TNG.*.dll assemblies
     - Provides public methods for common tasks
     - Serves as an example of the API Usage
  2. XmlMethodChanger.cmd
     - Command-line program that uses XmlMethodChanger.lib
     - Uses third-party NuGet package (Command Line Parser Library)
     - Can be used as a standalone tool for modifying methods



Demonstration

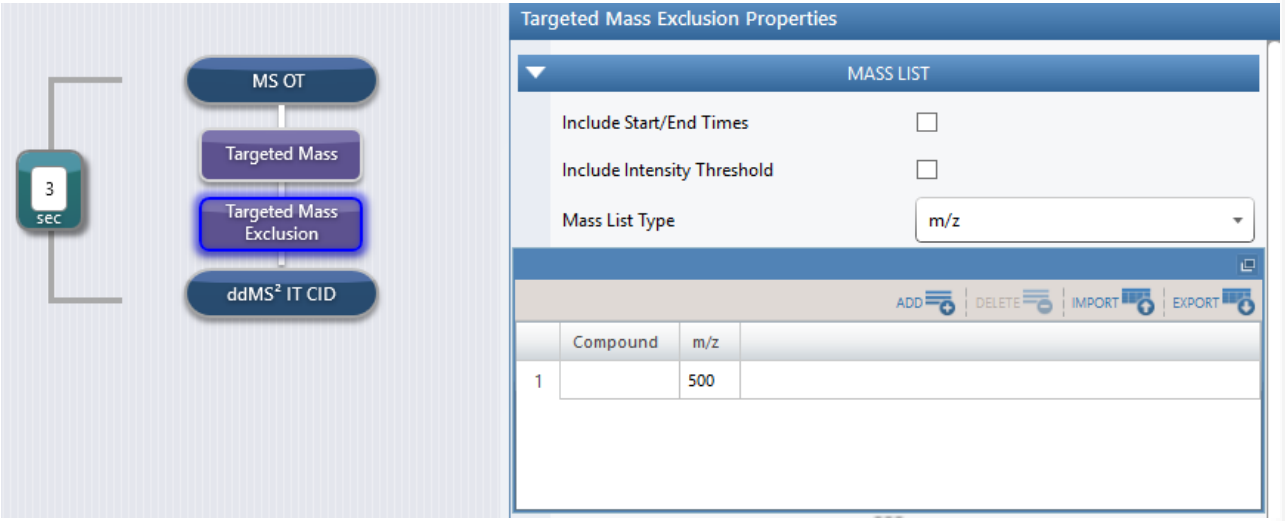
- The “template” method contains the structure of the experiment



This example is from the examples/Fusion/IntensityThreshold/ directory in the Git repository

Demonstration

- The “template” method contains the structure of the experiment



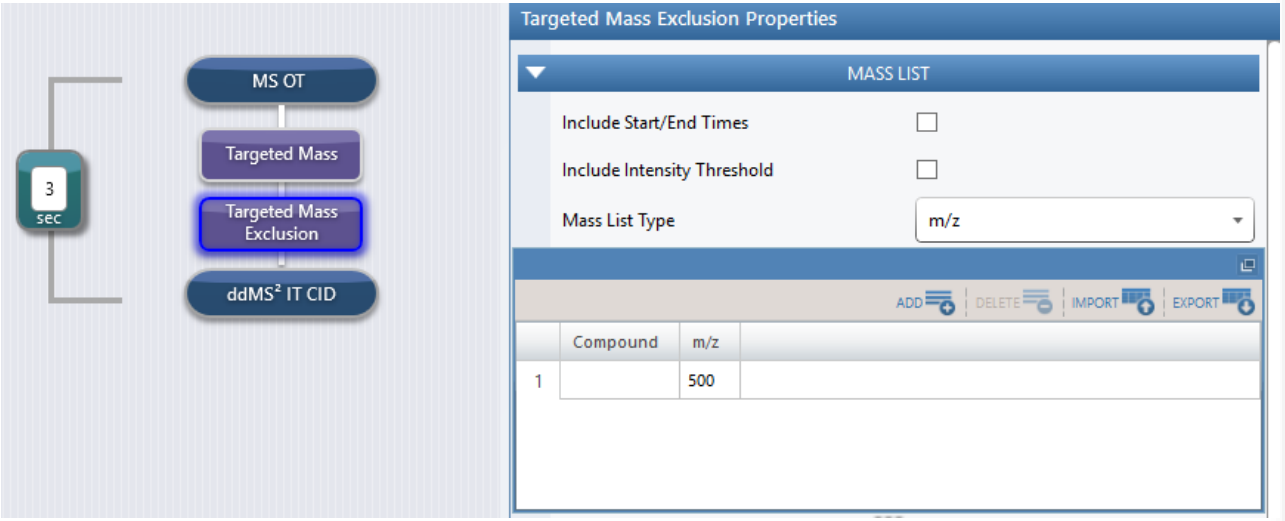
- The modification XML specifies a list of transformations to perform on the “template” method file

```xml
<?xml version="1.0" encoding="utf-8" ?>
<MethodModifications Version="2" Model="OrbitrapFusionLumos" Family="Calcium" Type="SL">
    <Modification Order="1">
        <Experiment ExperimentIndex="0">
            <MassListFilter MassListType="TargetedMassExclusion">
                <MassList IntensityThreshold="true">
                    <MassListRecord>
                        <MOverZ>195.12</MOverZ>
                        <Z>1</Z>
                        <IntensityThreshold>5e6</IntensityThreshold>
                    </MassListRecord>
                    <MassListRecord>
                        <MOverZ>262.3</MOverZ>
                        <Z>2</Z>
                        <IntensityThreshold>1e4</IntensityThreshold>
                    </MassListRecord>
                </MassList>
            </MassListFilter>
        </Experiment>
    </Modification>
    <Modification Order="2">
        <Experiment ExperimentIndex="0">
            <MassListFilter MassListType="TargetedMassInclusion">
                <MassList IntensityThreshold="true">
                    <MassListRecord>
                        <MOverZ>195.12</MOverZ>
                        <Z>1</Z>
                        <IntensityThreshold>5e7</IntensityThreshold>
                    </MassListRecord>
                    <MassListRecord>
                        <MOverZ>262.3</MOverZ>
                        <Z>2</Z>
                        <IntensityThreshold>1e2</IntensityThreshold>
                    </MassListRecord>
                </MassList>
            </MassListFilter>
        </Experiment>
    </Modification>
</MethodModifications>
```


This example is from the examples/Fusion/IntensityThreshold/ directory in the Git repository

Demonstration

- The “template” method contains the structure of the experiment

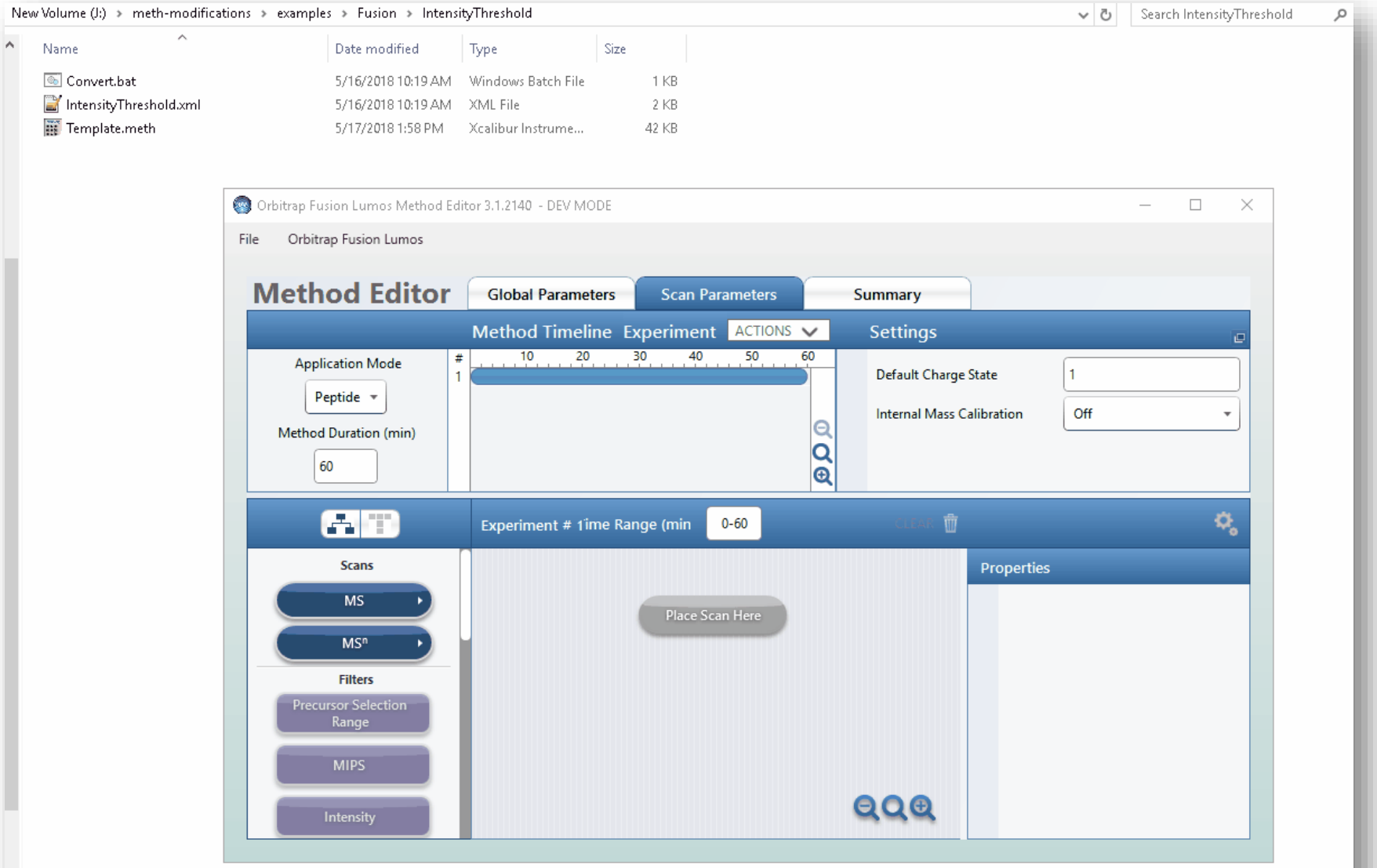


- The modification XML specifies a list of transformations to perform on the “template” method file
- The transformed method can then be saved to a separate .meth file

```xml
<?xml version="1.0" encoding="utf-8" ?>
<MethodModifications Version="2" Model="OrbitrapFusionLumos" Family="Calcium" Type="SL">
    <Modification Order="1">
        <Experiment ExperimentIndex="0">
            <MassListFilter MassListType="TargetedMassExclusion">
                <MassList IntensityThreshold="true">
                    <MassListRecord>
                        <MOverZ>195.12</MOverZ>
                        <Z>1</Z>
                        <IntensityThreshold>5e6</IntensityThreshold>
                    </MassListRecord>
                    <MassListRecord>
                        <MOverZ>262.3</MOverZ>
                        <Z>2</Z>
                        <IntensityThreshold>1e4</IntensityThreshold>
                    </MassListRecord>
                </MassList>
            </MassListFilter>
        </Experiment>
    </Modification>
    <Modification Order="2">
        <Experiment ExperimentIndex="0">
            <MassListFilter MassListType="TargetedMassInclusion">
                <MassList IntensityThreshold="true">
                    <MassListRecord>
                        <MOverZ>195.12</MOverZ>
                        <Z>1</Z>
                        <IntensityThreshold>5e7</IntensityThreshold>
                    </MassListRecord>
                    <MassListRecord>
                        <MOverZ>262.3</MOverZ>
                        <Z>2</Z>
                        <IntensityThreshold>1e2</IntensityThreshold>
                    </MassListRecord>
                </MassList>
            </MassListFilter>
        </Experiment>
    </Modification>
</MethodModifications>
```


This example is from the examples/Fusion/IntensityThreshold/ directory in the Git repository

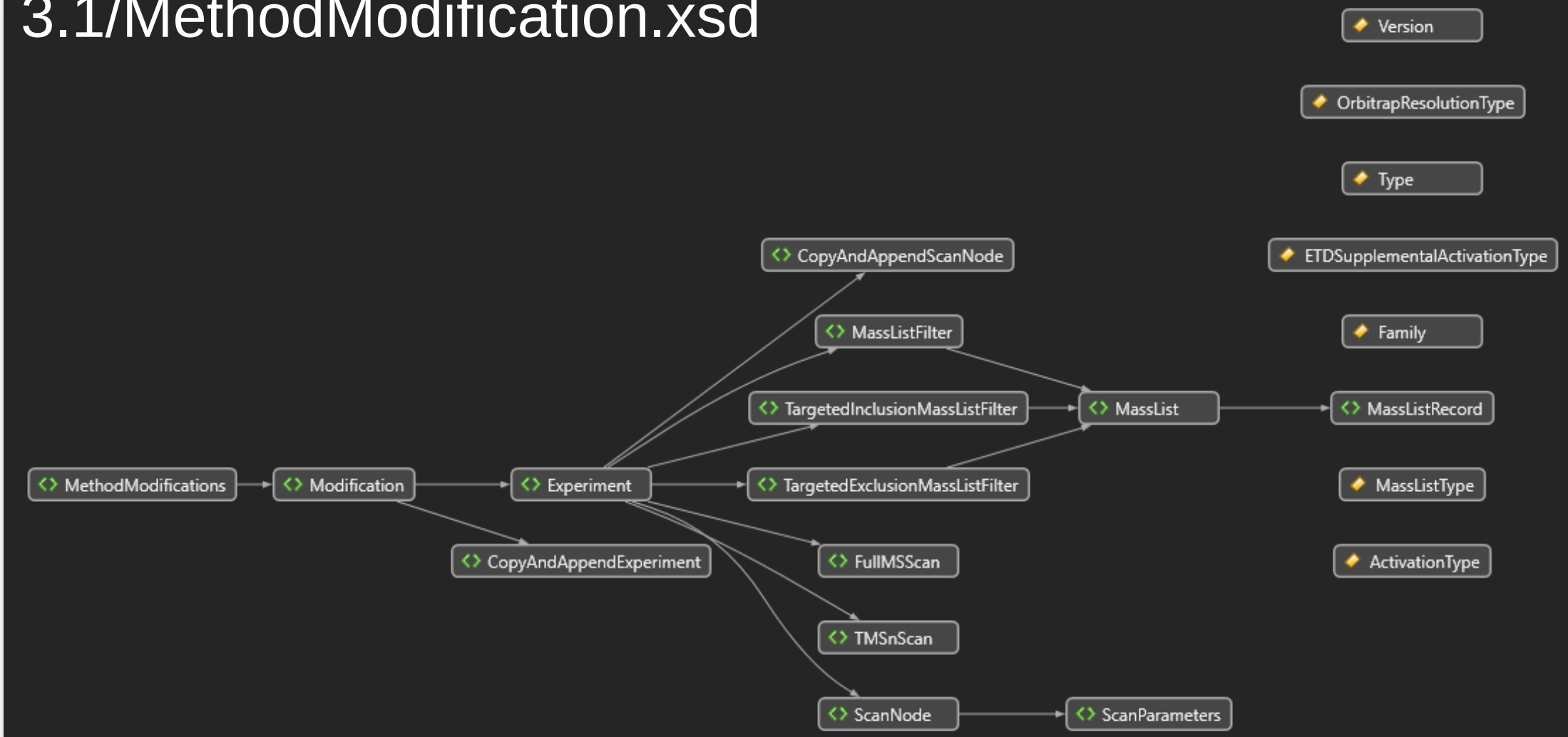
Demonstration



This example is from the examples/Fusion/IntensityThreshold/ directory in the Git repository

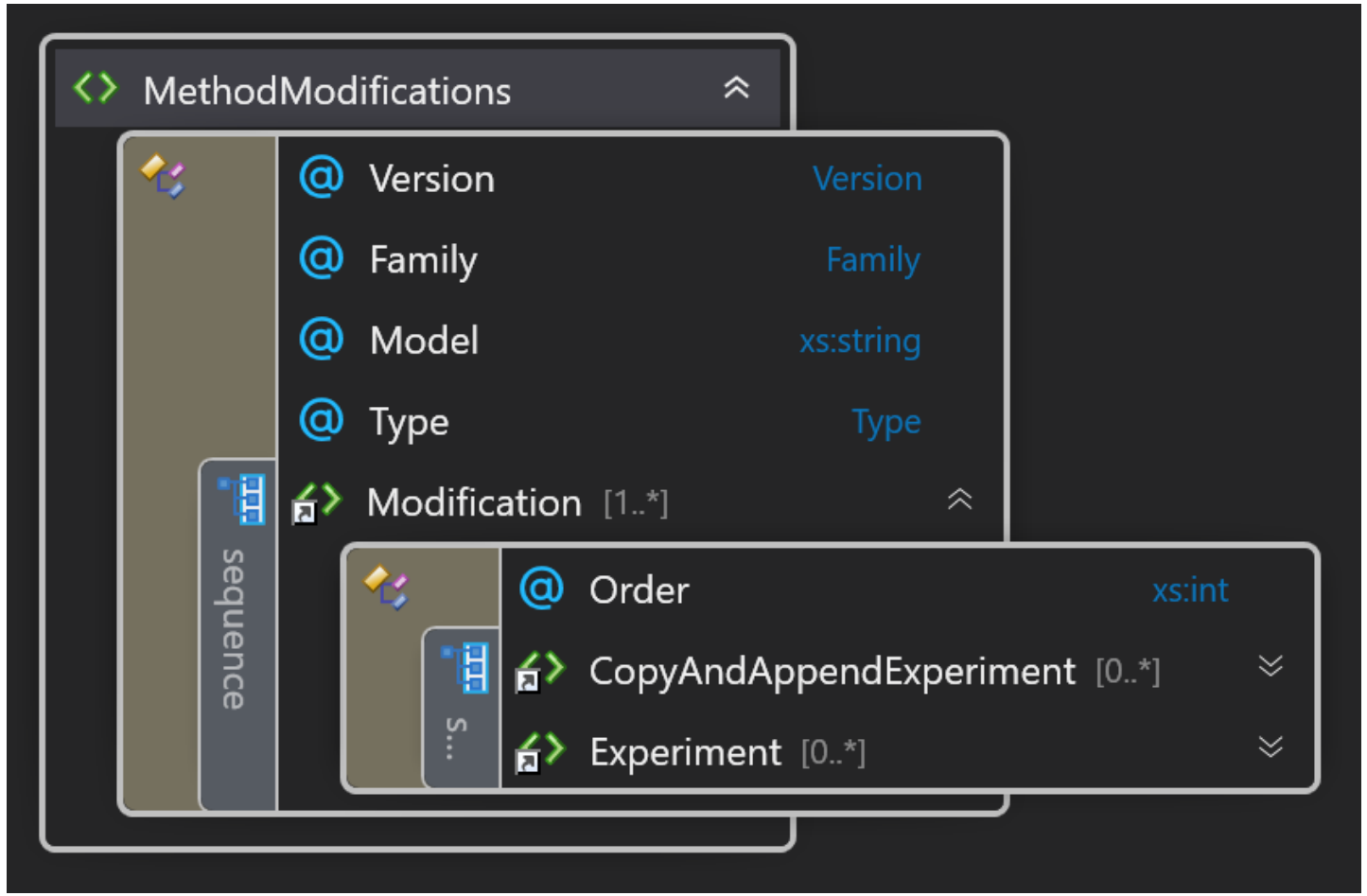
Fusion 3.1 Method Modification Schema

3.1/MethodModification.xsd



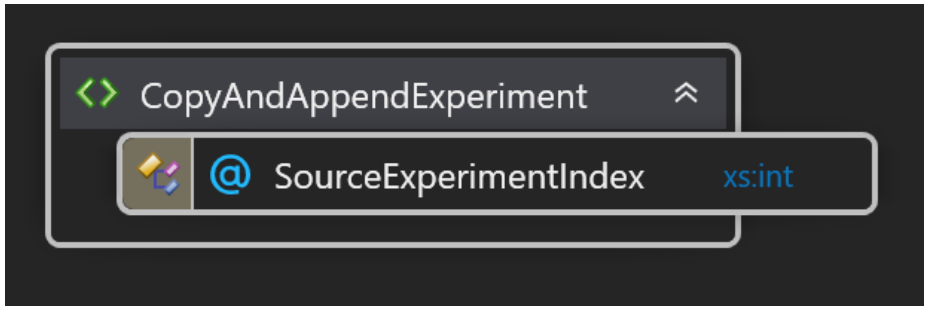
Method Modifications

- The XML contains a collection of **Modifications**
- Each **Modification** contains:
  - An **Order** attribute
    - This is the order that this modification will take effect, from low to high.
  - Collection of **Actions** elements
    - This is what the **Modification** performs on the method
    - **Experiments** are performed first, in the order they are defined in the xml
    - **CopyAndAppendExperiment** are performed second, in the order they are defined in the xml

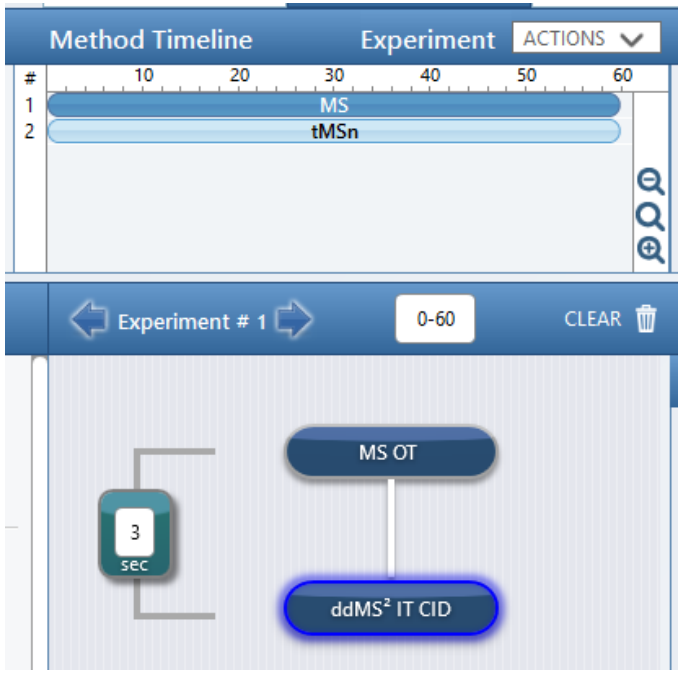


Copy And Append Experiment

- Copies a method experiment and appends it the end of all the experiments
- The attribute **SourceExperimentIndex** (0-based) specifies which experiment to copy

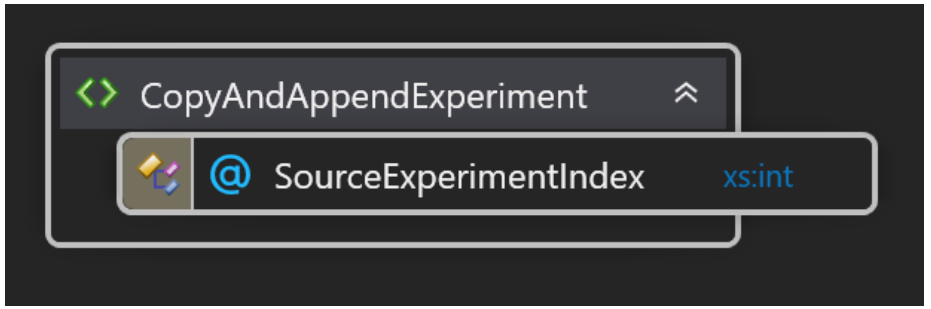



```
<CopyAndAppendExperiment SourceExperimentIndex="0" />
```

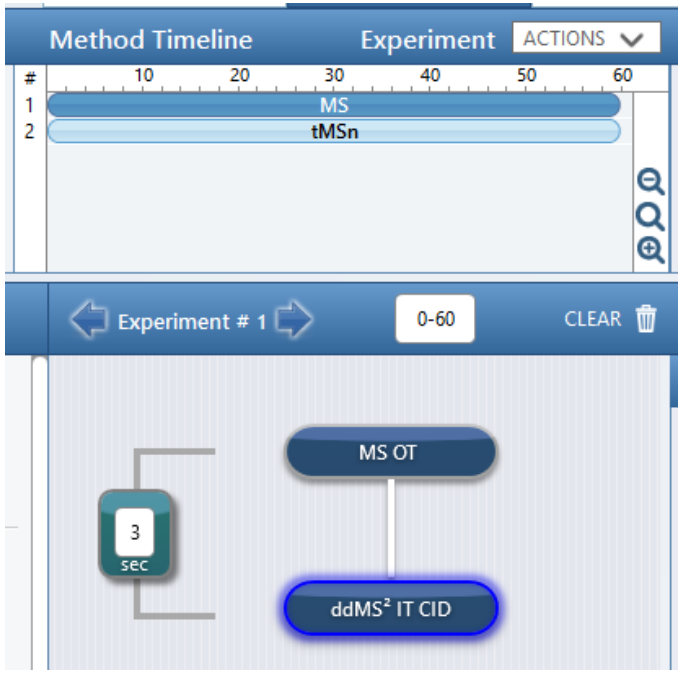


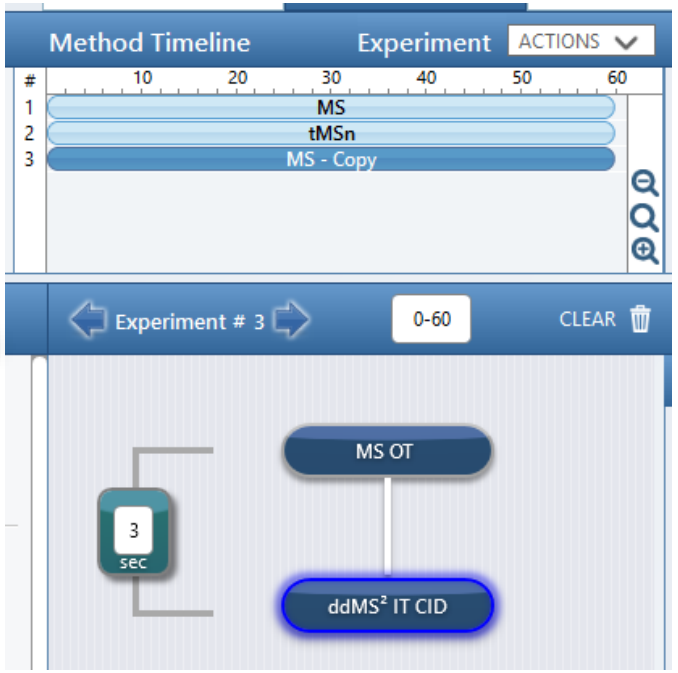
Copy And Append Experiment

- Copies a method experiment and appends it the end of all the experiments
- The attribute **SourceExperimentIndex** (0-based) specifies which experiment to copy



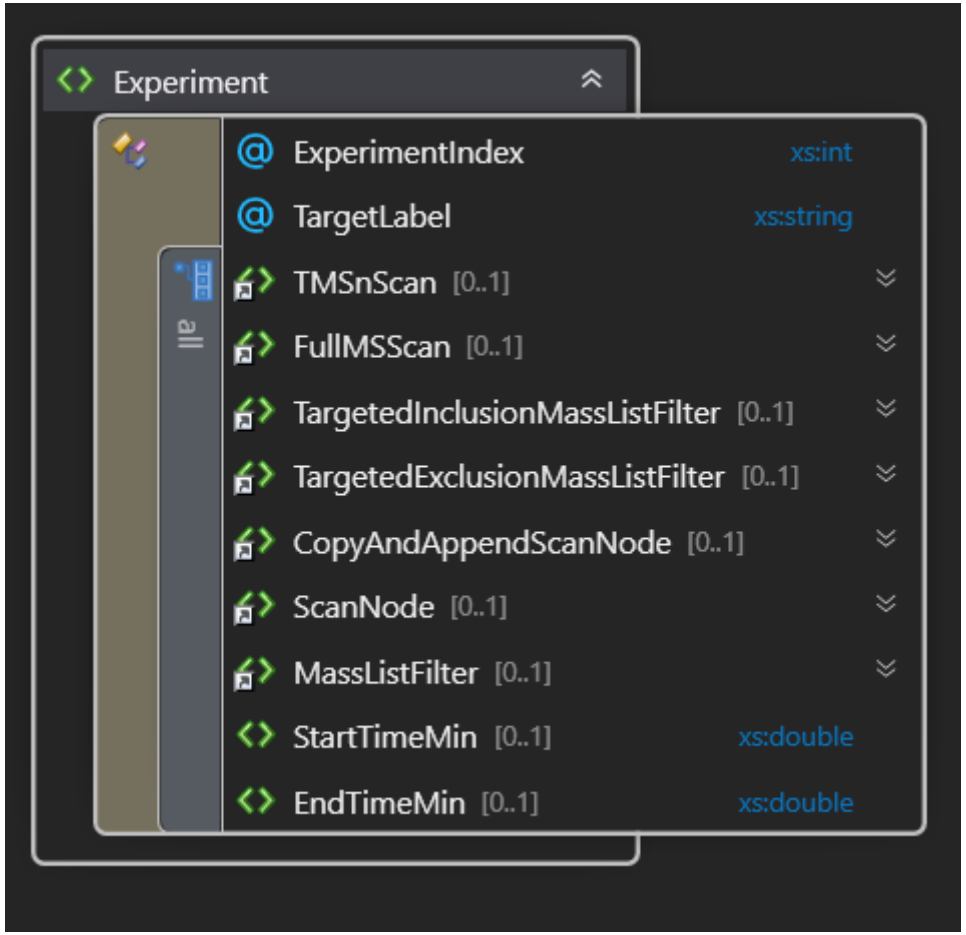

```
<CopyAndAppendExperiment SourceExperimentIndex="0" />
```





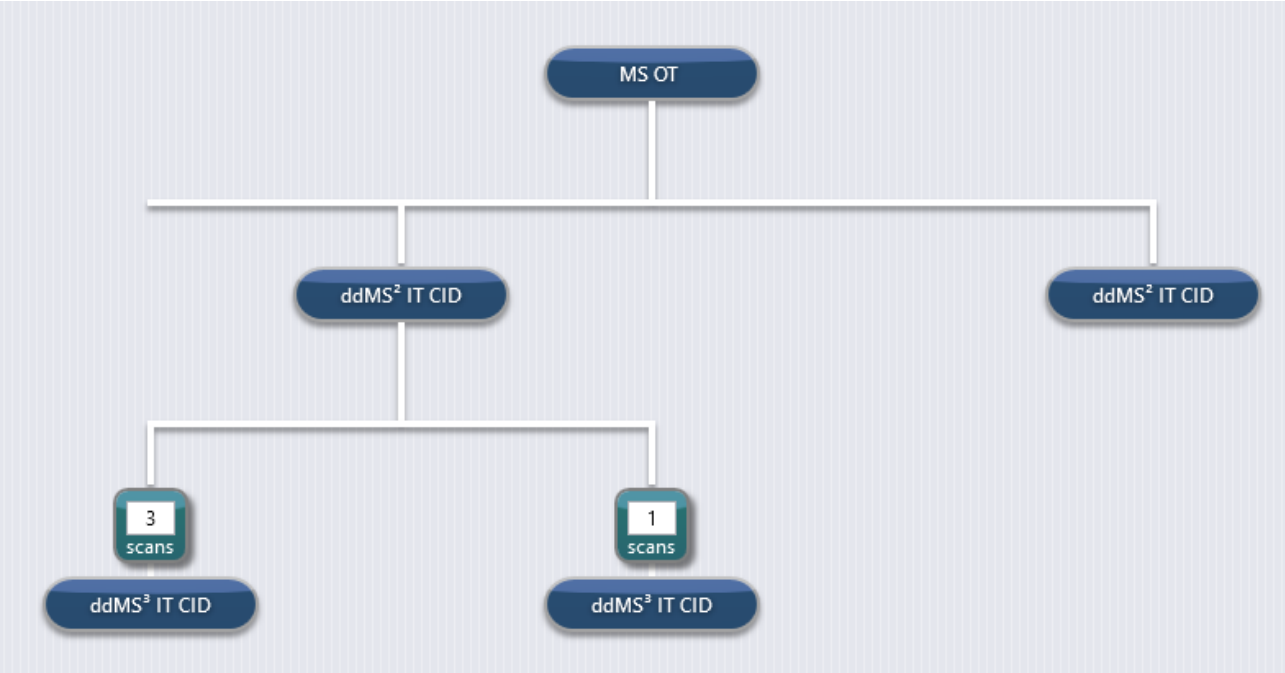
Experiment Modifications

- Modifies an experiment based on the attribute **ExperimentIndex** (0-based)
- Here you can target certain nodes/filters of the selected experiment and modify their contents
- You can also modify the experiment time frame with the **StartTimeMin** and **EndTimeMin** fields.



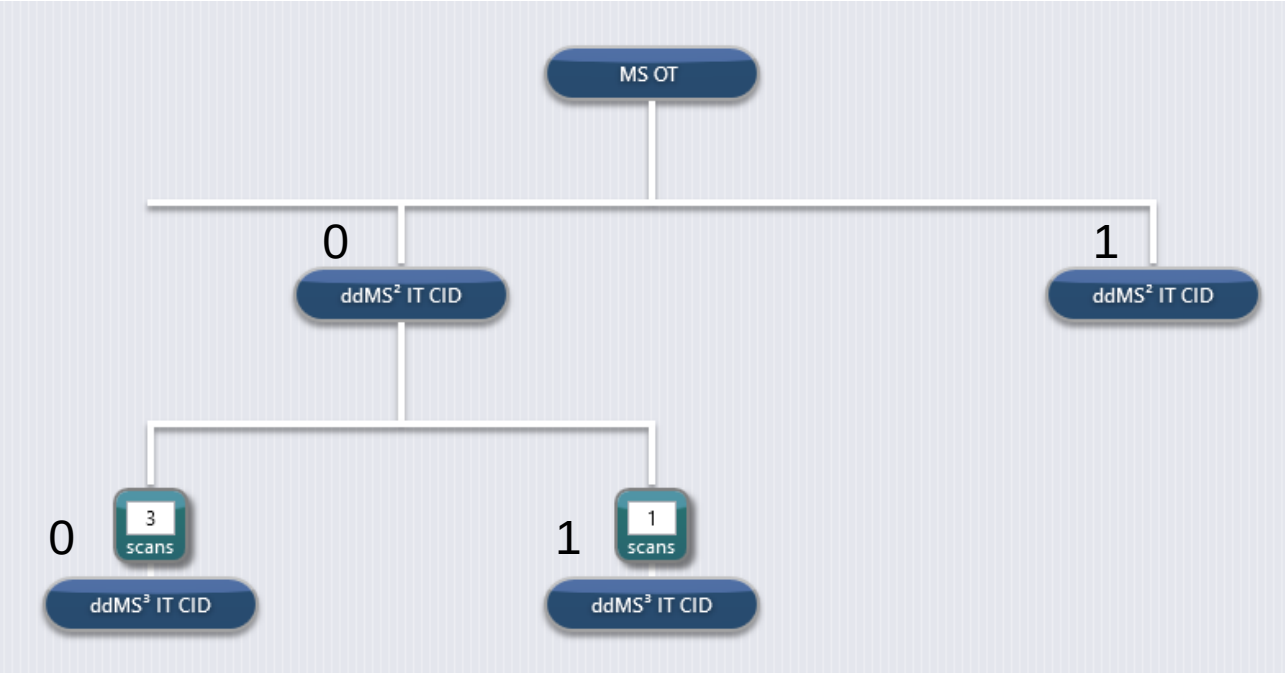
Source Node Position

- **SourceNodePosition** is a mechanism to specify a specific node in scan tree



Source Node Position

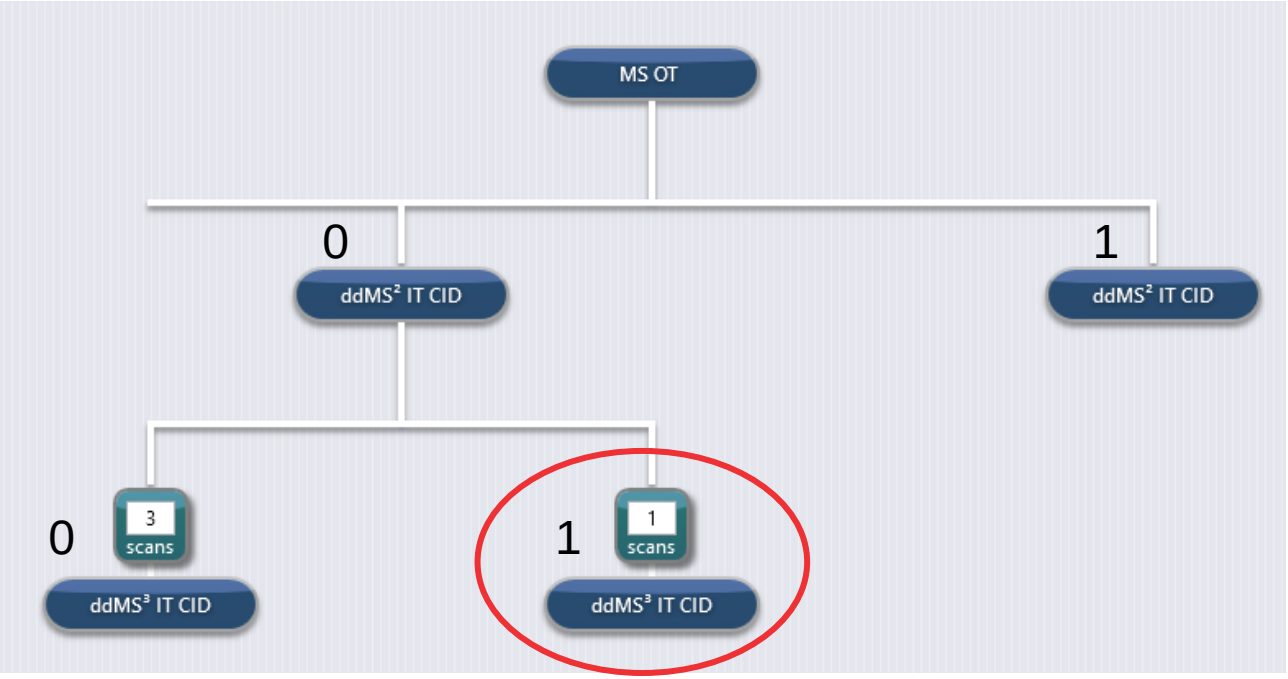
- **SourceNodePosition** is a mechanism to specify a specific node in scan tree
- Each instance of **SourceNodePosition** selects which child node to select; numbered left-to-right and 0-based.



Source Node Position

- **SourceNodePosition** is a mechanism to specify a specific node in scan tree
- Each instance of **SourceNodePosition** selects which child node to select; numbered left-to-right and 0-based.
- To specify the highlighted node, you would need to do:

```
<SourceNodePosition>0</SourceNodePosition>
<SourceNodePosition>1</SourceNodePosition>
```

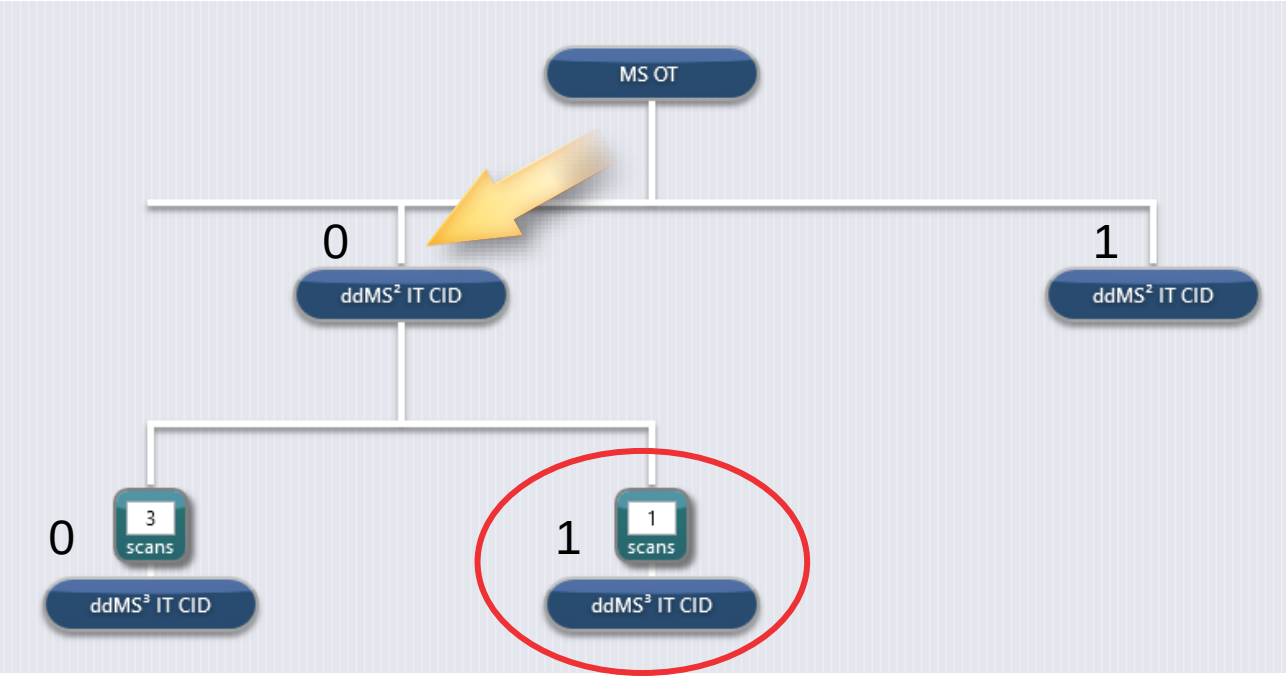


Source Node Position

- **SourceNodePosition** is a mechanism to specify a specific node in scan tree
- Each instance of **SourceNodePosition** selects which child node to select; numbered left-to-right and 0-based.
- To specify the highlighted node, you would need to do:

```
<SourceNodePosition>0</SourceNodePosition>
<SourceNodePosition>1</SourceNodePosition>
```

  - The first instance selects the left branch off the MS1.

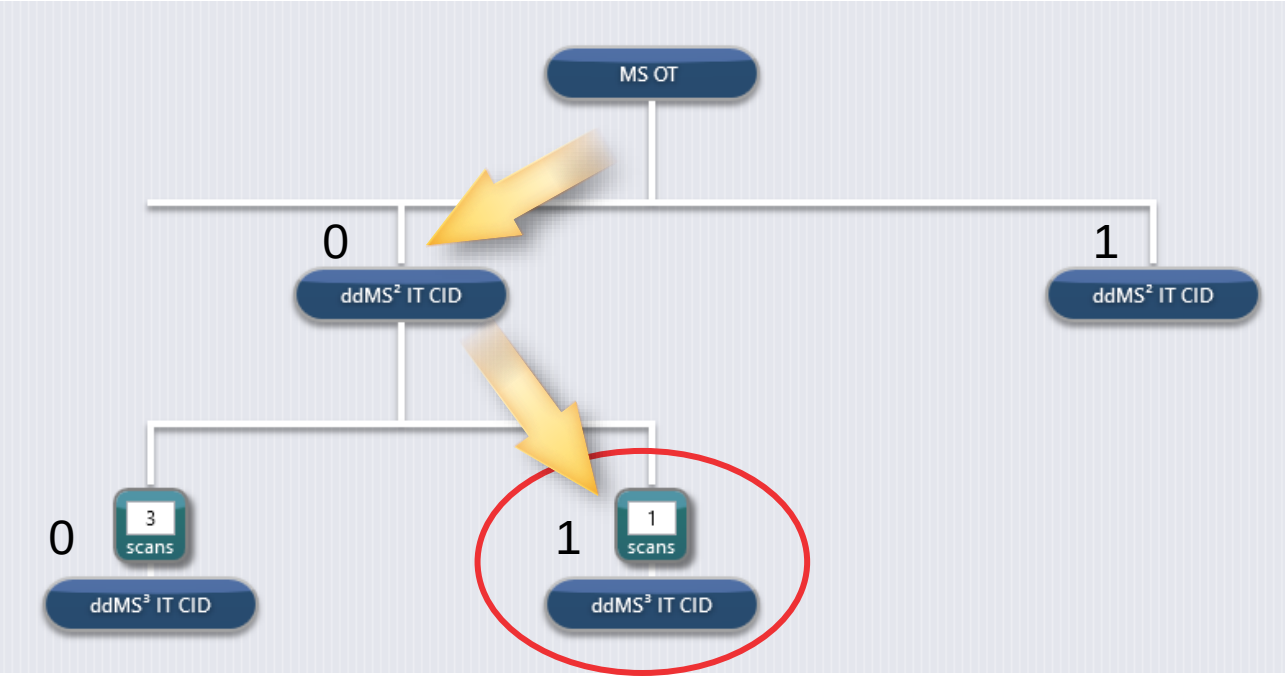


Source Node Position

- **SourceNodePosition** is a mechanism to specify a specific node in scan tree
- Each instance of **SourceNodePosition** selects which child node to select; numbered left-to-right and 0-based.
- To specify the highlighted node, you would need to do:

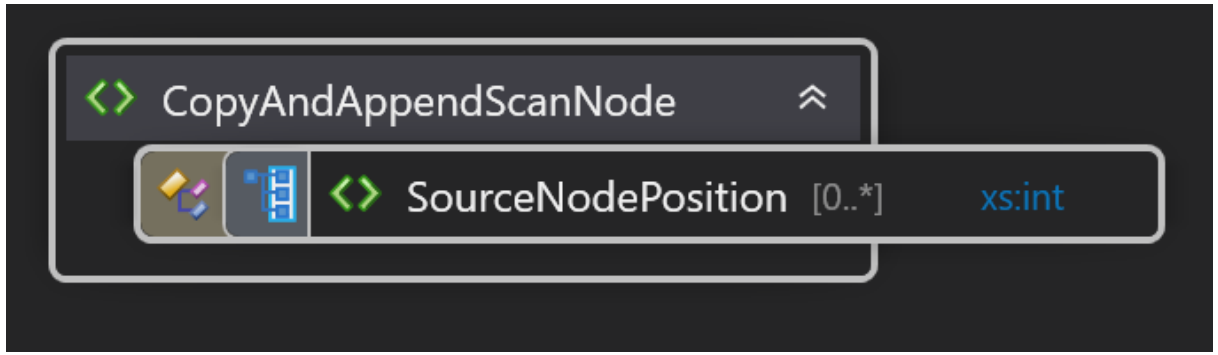
```
<SourceNodePosition>0</SourceNodePosition>
<SourceNodePosition>1</SourceNodePosition>
```

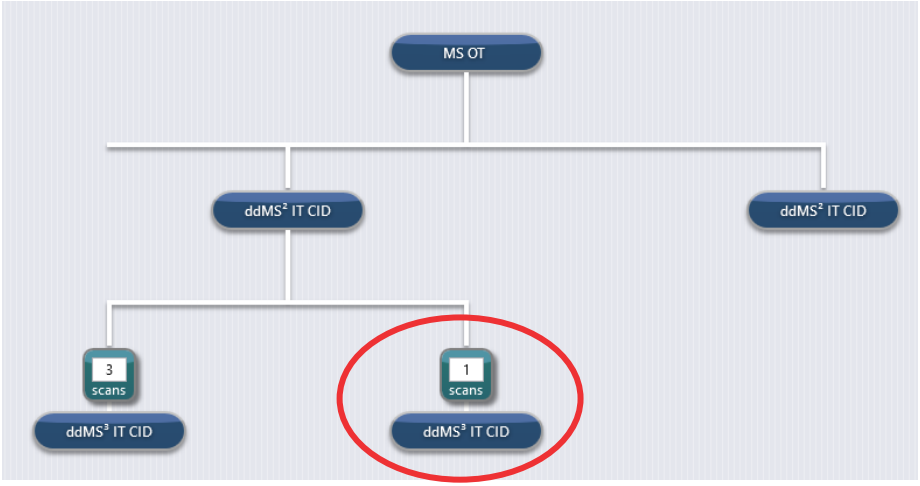
  - The first instance selects the left branch off the MS1.
  - The second instance selects the right branch off the ddMS2



Copy And Append Scan Node

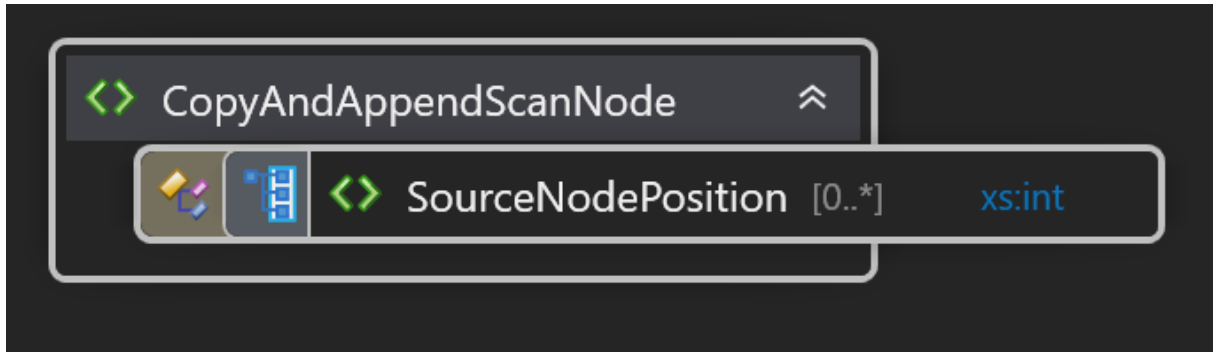
- Copies a scan node in the experiment and appends it to its parent node
- Uses **SourceNodePosition** to specify the specific node in the tree



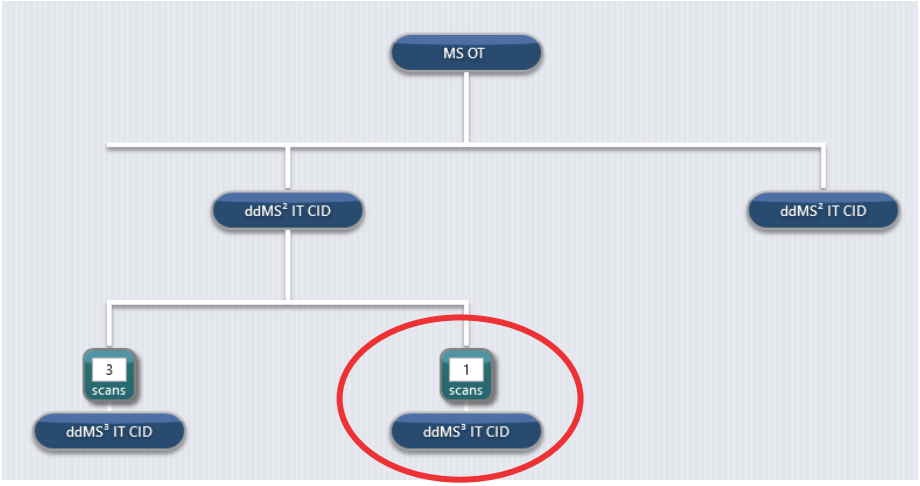


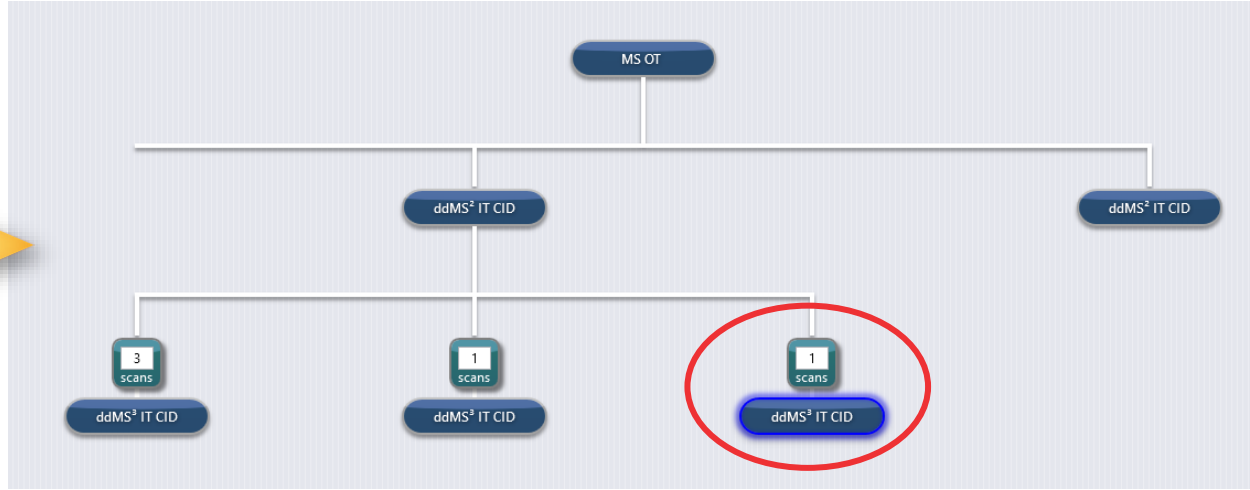
Copy And Append Scan Node

- Copies a scan node in the experiment and appends it to its parent node
- Uses **SourceNodePosition** to specify the specific node in the tree



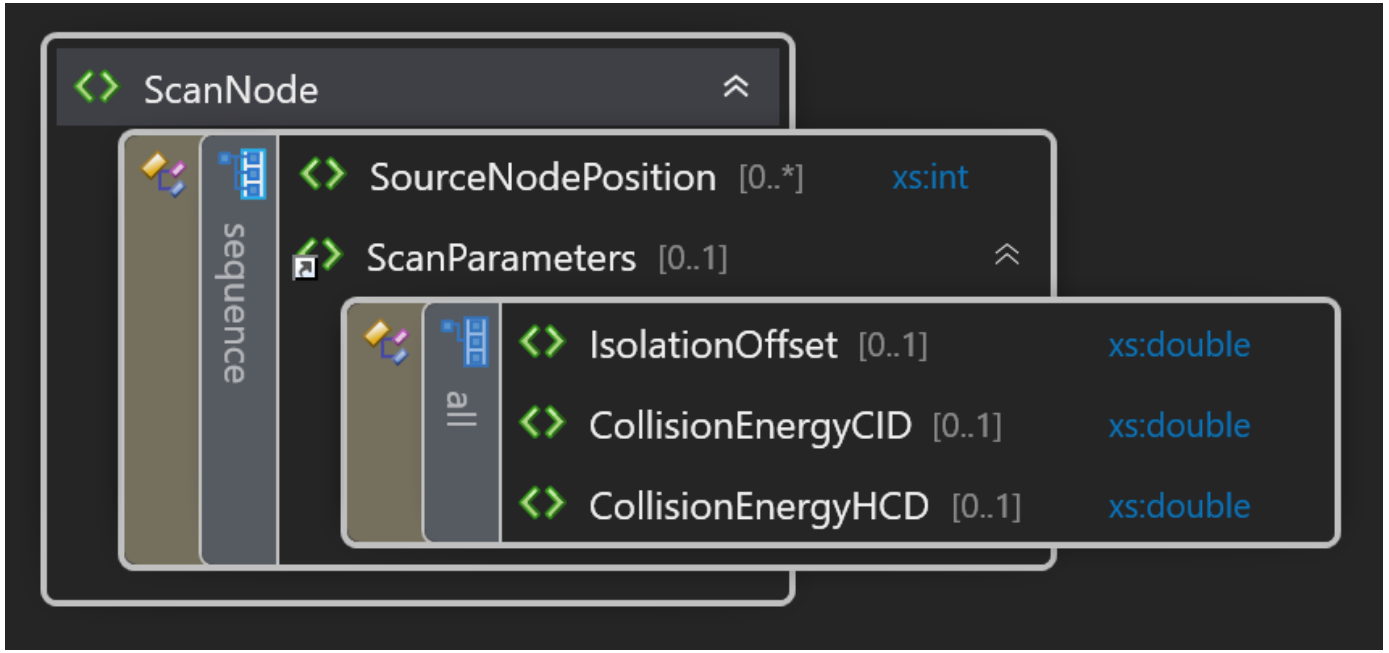

```
<CopyAndAppendScanNode>
  <SourceNodePosition>0</SourceNodePosition>
  <SourceNodePosition>1</SourceNodePosition>
</CopyAndAppendScanNode>
```





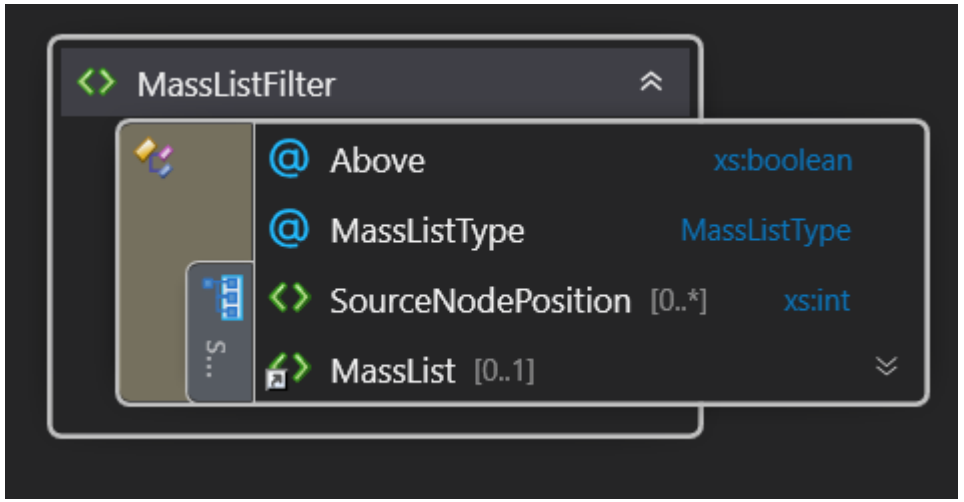
Scan Node Modification

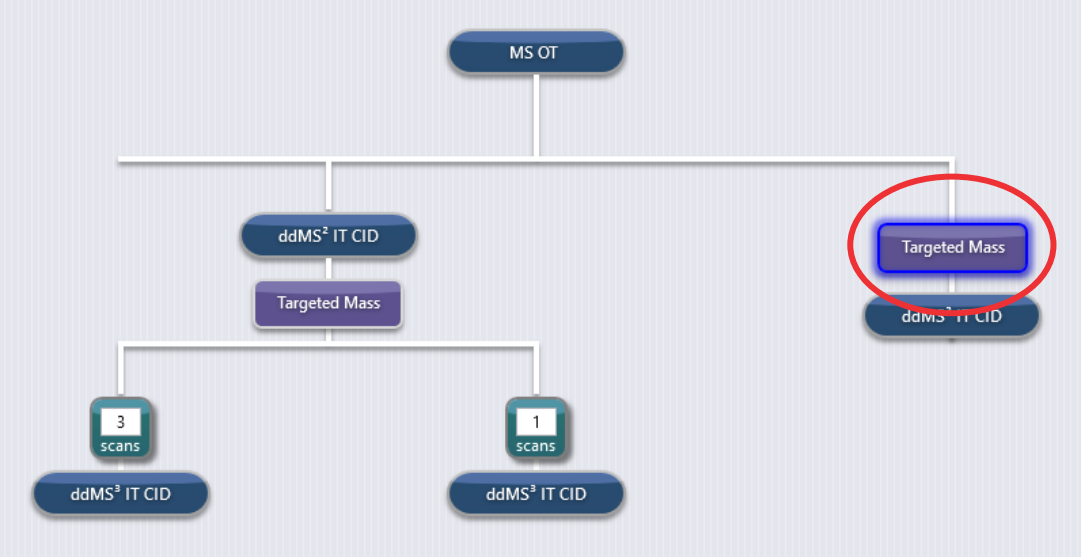
- Modifies a scan based on the specified **SourceNodePosition**.
- You can currently change the
  - Isolation offset
  - CID Collision Energy
  - HCD Collision Energy



Mass List Filter Modification

- Modifies a filter with a mass list based on the specified **SourceNodePosition** and **Above** attribute
  - To specify the filter, you first select the node the filter is attached to, and then indicate if the filter is above or below the node.



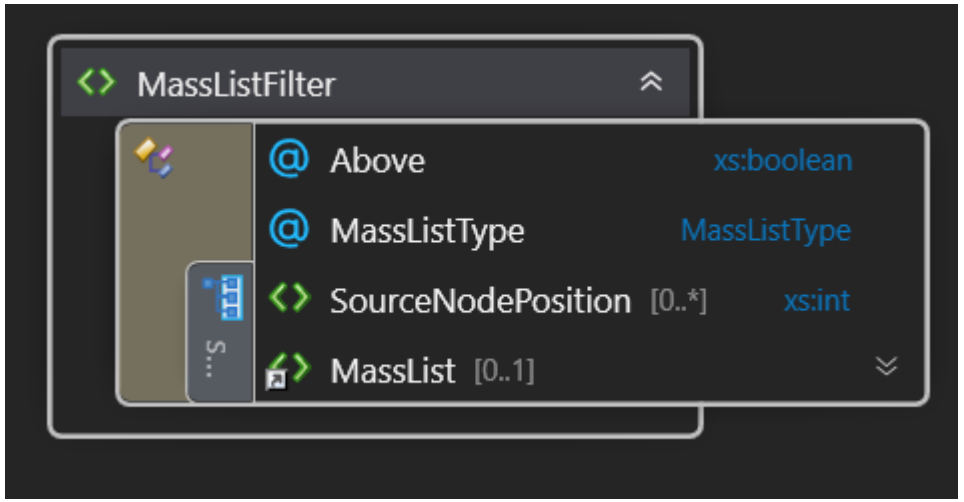


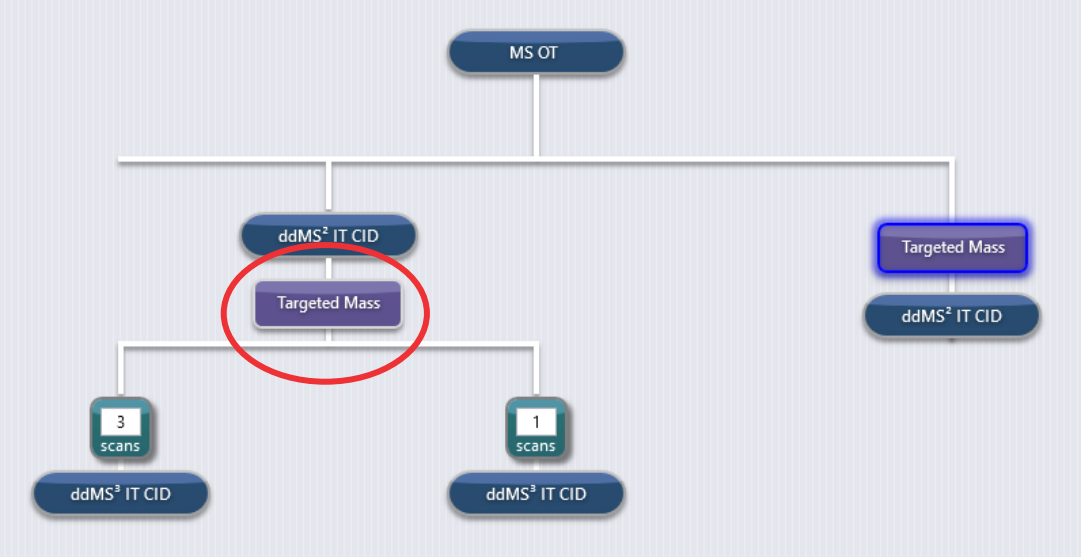
- This filter would be defined as

```
<MassListFilter Above="true" MassListType="TargetedMassInclusion">
  <SourceNodePosition>1</SourceNodePosition>
  <MassList />
</MassListFilter>
```

Mass Filter Modification

- Modifies a filter with a mass list based on the specified **SourceNodePosition** and **Above** attribute
  - To specify the filter, you first select the node the filter is attached to, and then indicate if the filter is above or below the node.





- This filter would be defined as

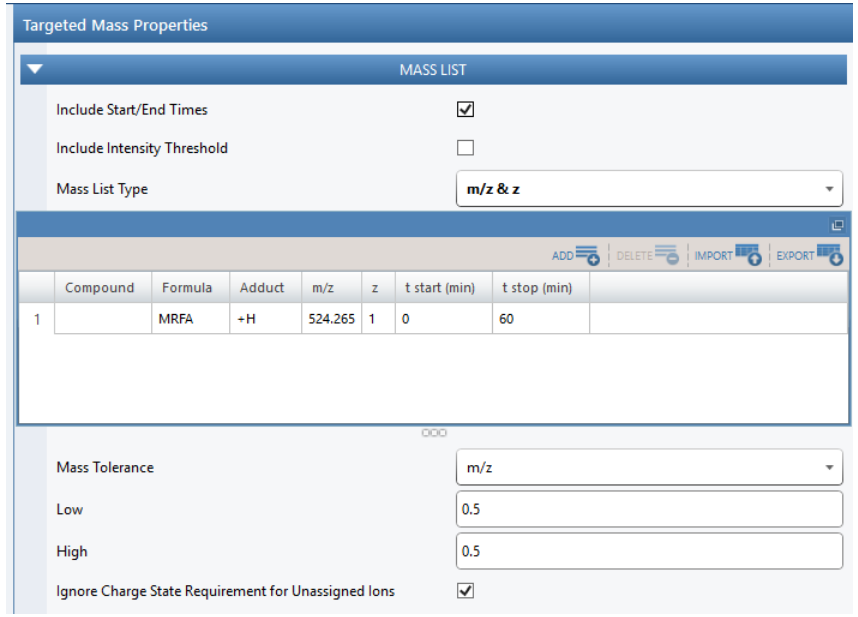
```
<MassListFilter Above="true" MassListType="TargetedMassInclusion">
  <SourceNodePosition>1</SourceNodePosition>
  <MassList />
</MassListFilter>
```

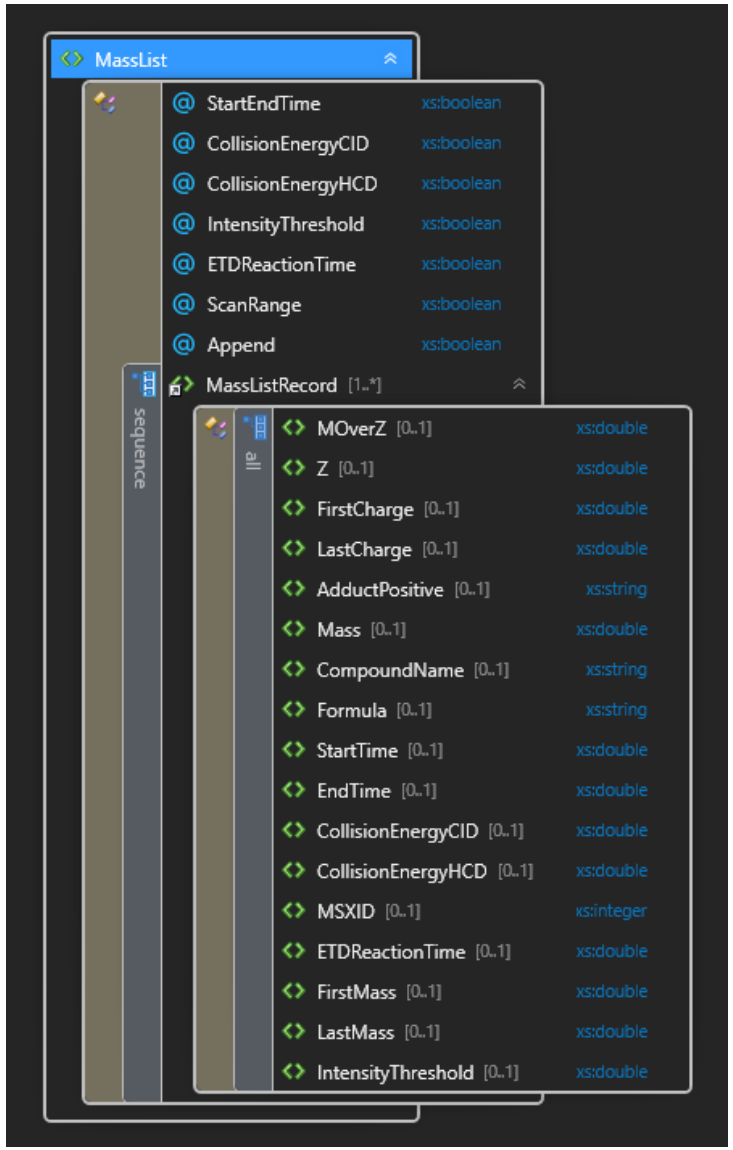
- And this filter would be defined as

```
<MassListFilter Above="false" MassListType="TargetedMassInclusion">
  <SourceNodePosition>0</SourceNodePosition>
  <MassList />
</MassListFilter>
```

Mass List Modification

- **MassList** contains a collection of **MassListRecords** which are the entries in the table
- The different Boolean attributes control which columns in the table are used
- The **Append** attribute indicates if the entries are append to the list or are overwritten





Overview

1. **Instrument Application Programming Interface (IAPI)**
   - Real-time control and acquisition of Orbitrap Mass Spectrometers using .NET
2. **XML Method Modification Interface (XMMI)**
   - Programmatically modify Fusion-series method files
3. **Applications of IAPI**
   1. **Devin Schweppe** – Fusion API
   2. **Christoph Wichmann** – Q Exactive API
4. **Examples and FAQ for IAPI and XMMI**
5. **Applications of XMMI**
   1. **Pavel Shliaha** – Fusion Method Creation

Overview

1. **Instrument Application Programming Interface (IAPI)**
   - Real-time control and acquisition of Orbitrap Mass Spectrometers using .NET
2. **XML Method Modification Interface (XMMI)**
   - Programmatically modify Fusion-series method files
3. **Applications of IAPI**
   1. **Devin Schweppe** – Fusion API
   2. **Christoph Wichmann** – Q Exactive API
4. **Examples and FAQ for IAPI and XMMI**
5. **Applications of XMMI**
   1. **Pavel Shliaha** – Fusion Method Creation

IAPI FAQ

- **Is the IAPI Free?**
  - Yes! Though it requires a license agreement with Thermo.
- **Are the Q Exactive and Fusion IAPIs the same?**
  - They are not **binary** compatible, but share the same structure, namespacing, and basic data and control paths.
  - It only requires a minimal amount of work to convert code between the two APIs.
- **How do we obtain the IAPI?**
  - Interfaces are shared on https://github.com/thermofisherlsms/iapi
  - Contact ThermoMSLicensing@thermo.com to start the process.
- **Can you run the IAPI while acquiring an acquisition from a method?**
  - Yes, the IAPI works even when in idle, or actively running a method.

IAPI FAQ

- **Are IAPI scans parallelized?**
  - When running a method, IAPI scans are inserted into the acquisition pipeline that parallelizes the scans with other scans.
  - When running without a method, IAPI scans are **not** parallelized.
- **Are the results of the method filters published through the IAPI?**
  - The IAPI only publishes the raw spectrum, without any of the method filters taking effect.
- **Can we obtain profile data through the IAPI?**
  - Currently only the centroided data is returned to the IAPI.
- **Is the IAPI 64-bit enabled?**
  - No, the main instrument service is a 32-bit service and cannot export a 64-bit interface

XMMI FAQ

- **Can you modify every parameter?**
  - No, only a subset of parameters in a method can be modified using this API.
- **I want to modify parameter X, Y and Z, can you provide it?**
  - Please file an **issue** with the github account describing the desired changes.
  - https://github.com/thermofisherlsms/meth-modifications/issues
- **Do you need be attached to an instrument to modify methods?**
  - No, if you have installed the **Workstation** configuration, you can create and modify methods on any computer.
- **Does this work for Exactive methods?**
  - This interface does not currently support Exactive method modification.

IAPI Example – Tribrid Series

- Use **Thermo.TNG.Factory.Factory.Create()** to create instrument access container
- Connect to the main service using the **StartOnlineAccess()** call
- Wait until server has connected and get access to a particular instrument

```
using Thermo.TNG.Factory;
using Thermo.Interfaces.FusionAccess_V1;
using Thermo.Interfaces.FusionAccess_V1.MsScanContainer;
using Thermo.Interfaces.InstrumentAccess_V1.MsScanContainer;
using System;

namespace MinifiedExample
{
    0 references | Thermo Scientific, 7 minutes ago | 2 authors, 2 changes
    class Program
    {
        0 references | Thermo Scientific, 7 minutes ago | 2 authors, 2 changes
        static void Main(string[] args)
        {
            // Use the Factory creation method to create a Fusion Access Container
            IFusionInstrumentAccessContainer fusionContainer = Factory<IFusionInstrumentAccessContainer>.Create();

            // Connect to the service by going 'online'
            fusionContainer.StartOnlineAccess();

            // Wait until the service is connected
            // (better through the event, but this is nice and simple)
            while (!fusionContainer.ServiceConnected);

            // From the instrument container, get access to a particular instrument
            IFusionInstrumentAccess fusionAccess = fusionContainer.Get(1);

```


This example is from the examples/tribrid/MinifiedExample/ directory in the Git repository

Overview

1. **Instrument Application Programming Interface (IAPI)**
   - Real-time control and acquisition of Orbitrap Mass Spectrometers using .NET
2. **XML Method Modification Interface (XMMI)**
   - Programmatically modify Fusion-series method files
3. **Applications of IAPI**
   1. **Devin Schweppe** – Fusion API
   2. **Christoph Wichmann** – Q Exactive API
4. **Examples and FAQ for IAPI and XMMI**
5. **Applications of XMMI**
   1. **Pavel Shliaha** – Fusion Method Creation