



LIVE CODING



INTERMEDIATE AF SDK DEVELOPER

Getting the most out of AFSearch

Rick Davin, Sr. Technology Enablement Engineer
David Hearn, Group Leader, Software Engineering

Live Coding Session

- New offering to PI World 2018
- Neither a hands-on lab nor a one-way lecture
- Want this to be a ***two-way dialogue***
- Appreciate feed back on whether to do live coding again in future PI World events
 - Even if you love concept of Live Coding but hate this particular presentation!

Rules

- First rule of Live Coding is we **DO** talk about Live Coding
- Tablets/Laptops not needed but feel free to use
- Questions are not only allowed during the presentation but highly encouraged!
 - Planned for a 15 minute Q&A on the backend
 - If we have lots of questions during, we will borrow from the backend

Who is the target audience

An Intermediate level .NET developer with previous experience using AFSearch

Prerequisites - prior experience with

- .NET Development
- AF SDK
- AFSearch

Why you want to be here

- You faithfully copy our AFSearch examples but rarely diverge from them
- You are unsure of what's going on and are timid to change any parameters
- You wish you had a better understanding of what's going on with AFSearch so that you have confidence in tweaking your code to make it better/faster
- Your searches take longer than expected and you would like to speed them up

What are the learning goals

- Know about the new features in AF 2.9.5 and 2.10
- Walk away with thorough understanding of what's going on during a search
 - Which calls require a trip to the server
 - Which call is the most expensive
- Fully understand why caching performs better than not caching
- Know what GetTotalCount truly represents
- Know which LINQ calls are good and which are bad

Format

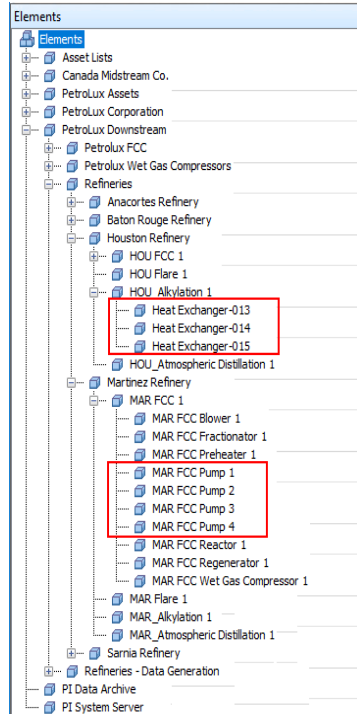
- A peek at our environment & database (5 mins)
- Assorted Topics with Demos (60 min)
- Q & A (15 min)
 - *Hit us with your best shot*
 - *Anyone who stumps David Hearn will be a Featured PI Geek!*
































Peek at the Client Environment

About the Environment

- Pre-release of PI AF 2018 Client (2.10)
- .NET Framework 4.6.2
- Visual Studio 2017 Community (free) and C#
- Sorry - can't share the database XML
- Review item counts and some pathing (*don't get bogged down spending too much time on our structure*)
- Focus on methods and techniques - Mentally play around with how you would use this on your own database

Sample of Structure and an Element



Heat Exchanger-722			
General Child Elements Attributes Ports Analyses Notification Rules Version			
Filter			
	   	Name	Value
		Tube Side Material	HC1500
	Category: Specification		
		Area	1425 ft2
	  	Calculated Heat Transfer Coefficient	Calc Failed
		Design Heat Transfer Coefficient	255.6 Btu/h/ft2/F
		Category: Temperatures	
		 Cold Side Inlet Temperature	136.7647705078
		 Cold Side Outlet Temperature	152.4381408691
	 	Cold Side Temperature Difference	15.67337036132
		 Hot Side Inlet Temperature	347.3962402343
		 Hot Side Outlet Temperature	368.3725891113
	 	Hot Side Temperature Difference	-20.9763488769
	 	LMTD	212.75710969832 delta T

One of the
searched
Category
than just
reference

One of the demos searches on Attribute Category to find more than just PIPoint data references.

Counts to consider

- Over 2200 elements
- Over 5.7 million event frames
- The 'Compressor State' event frame template:
 - Over 3.5 million event frames - greater than 60% of total
 - For date range of Feb 2 – March 14, we will find 320K event frames out of the 3.5 million
 - The underlying SQL Server must read all 3.5 million to settle on the 320K to be sent to the client

What's New

Summary of Previous Releases

Version	Highlight
PI AF 2016 AF SDK 2.8.0	Initial release with new AFSearch Class for Elements, Event Frames, Notification Rules, and Notification Rule Templates
PI AF 2016 R2 AF SDK 2.8.5	FindObjectIds Filters: GroupID, Source, and Destination
PI AF 2017 AF SDK 2.9.0	Implements IDisposable, Event Frame Aggregation, Lightweight FindObjectFields Filters: CreationDate, ModifyDate, TargetName, TimeContext, and TimeContextEnd

What's New in PI AF 2017 R2 (2.9.5)

- New search classes
 - AFNotificationContactTemplateSearch
 - AFAttributeSearch
- Nested Queries!
- New Search Tokens:
 - EventFrame, IsInternal, Parent, PlugIn, and PlugInName
- Both AF Server & Client must be at least 2.9.5

What's New in PI AF 2018 (2.10)

- Remove restrictions on attribute value conditions:
 - DataReference does not need to be defined at template level
 - Template no longer required
- New Search Token:
 - ID supports Equal or IN(...)
- New Search Fields:
 - ConfigString, DisplayDigits, IsManualDataEntry
 - Trait, UOM, UOMID, UOMAbbr
 - SecurityString, SecurityToken

What's New in PI AF 2018 (2.10)

- These updates along with previous versions gives a lot of flexibility to developers
 - Recent PI Square post filtering on Parent
 - Lots of possibilities to filter on PlugIn or PlugInName
 - CheckSecurity using 'SecurityToken' field without loading object

What you will see in Demo 1

- Example #1 is an AFAttributeSearch using a nested query string for filtering the elements
- Example #2 uses FindObjectFields for a skinny search showing current attribute Value. If the server is 2.10, PlugIn and ConfigString are included as output fields.

Not shown but code is included on GitHub: examples identical to Example #1 but for Tokens and an Interpolated String for the nested query.

Code Demo # 1



Demo 1 Wrap up

- AFSearch objects are disposable and are best wrapped inside a Using Block
- A CacheTimeout of 10 minutes is more than generous for most applications but will immediately dispose of resources if wrapped inside of Using Block
- Nested queries are easy (*and cool too!*)
- To show newer fields such as PlugIn or ConfigString, both the AF Client and Server must be at 2.10 or later

Breaking It Down Line by Line

Breaking it down line-by-line

- Start with simple example
- Cover line-by-line
- Fully understand what's really happening on the client versus what's happening on the server

What you will see in Demo 2

- A few simple lines of code to perform an AFAttributeSearch followed by the same lines of code with LOTS of comments and explanation
- First example similar to Demo 1 using same nested query string
- Second example similar to first except we add a GetTotalCount() before FindAttributes()
- Third example skips a Using Block but shows an explicit Close() on the search to immediately dispose of server resources

Code Demo # 2



Demo 2 Wrap up

- Many commands you think might make a trip to the server actually don't
- AFSearch tries to be as lazy as possible
- The first RPC issued ...
 - Is the most expensive
 - Creates the cached list of ID's on the server
 - Begins the CacheTimeout countdown
- A RPC is made for each page of data

Demo 2 Wrap up continued

- The end of the Using Block sends an RPC so that the server immediately dispose of resources
- If you omit a Using Block, you should
 - Wrap your code inside a Try-Catch-Finally Block
 - Issue an explicit Close() inside the Finally to immediately dispose of resources

Caching versus Not Caching

Caching vs. Not Caching

- Why server-side caching is good
- Why not using server-side caching performs poorly
- The special case of when not caching is a good thing
- Things to approach with **caution and aforethought**
 - pageSize: int.MaxValue
 - ToList() or ToArray()

Recommend: *Never!*

Recommend: *Sometimes
(if small number of items)*

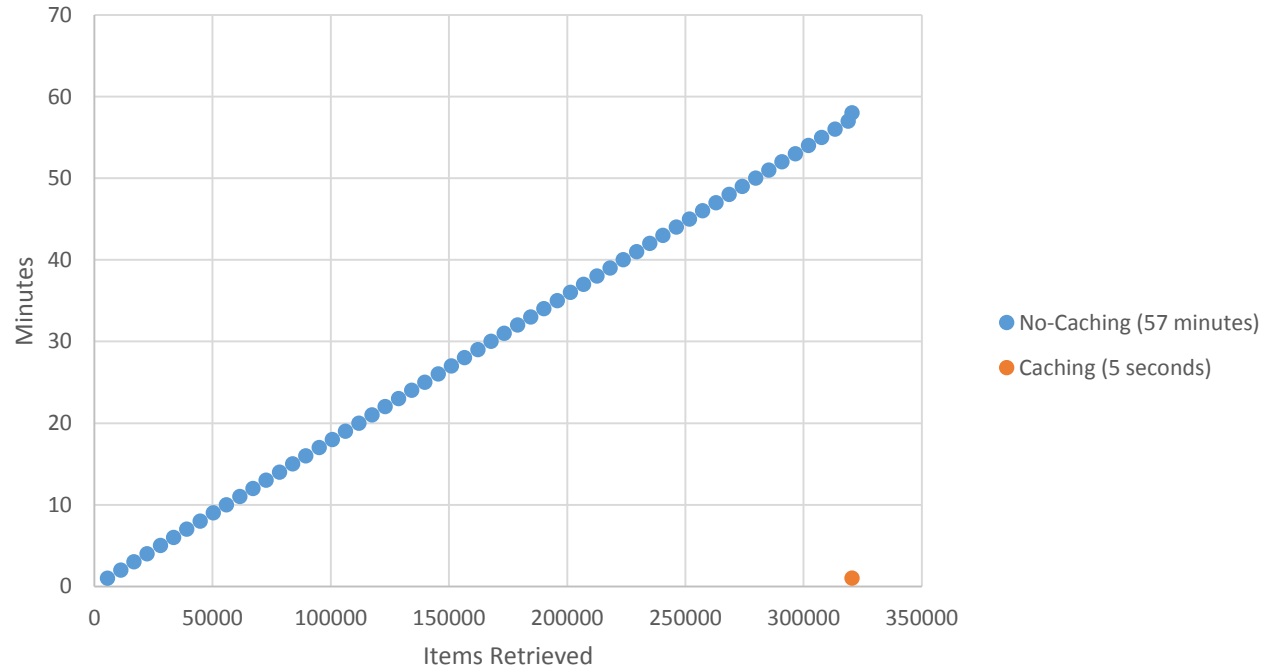
What you will see in Demo 3

- Compare back-to-back examples of same code except the 1st uses caching and the 2nd does not
- 3rd example shows the special case where not caching is an acceptable practice

Code Demo # 3



Compare Caching vs Not Caching



Demo 3 Wrap up

- When caching, the first RPC is the most expensive since it must create the cached list first
- Without caching, every RPC is expensive since each RPC repeatedly creates and disposes of the list
- If you only process the first page of results once - *and only once* - you do not need to cache
- A concern of caching is creating *huge* lists with *long* timeouts consuming resources on server – which underscores importance of disposing of resources immediately when no longer needed

Accurate Filter Counts

Accurate Filter Counts

- GetTotalCount() is count of items to be sent from server to client, i.e. count of items that passed server-side filters
- Filtering on Attribute Values depends upon data reference
- Data References are resolved client-side
 - Exception: event frames with captured values
- GetTotalCount() is not the same as filtered count whenever client-side filtering is at play

What you will see in Demo 4

- Compare back-to-back examples of similar code
- First example searches on element without attribute filtering
- Second example same as first except it includes an attribute filter
- What to look for – the count of elements passing the filters in each example

Code Demo # 4



Demo 4 Wrap up

- Both examples return same `GetTotalCount()`, i.e. the server sends same number of elements to the client for each example
- Due to attribute value filter, data references must be evaluated client side in the 2nd example
- In 2nd example, despite sending `GetTotalCount()` number of elements, fewer elements are processed in the For Each loop because fewer elements passed the client-side filtering

Are you your own worst enemy?

Are you your own worst enemy?

- Don't be the cause of your own slowdowns
 - Beware the hidden perils of many LINQ calls
 - Did you neglect to opt-in to caching?
- Try to make fewest trips to server
- Consider how many results you expect
 - Less than 10K or over 1 million?
 - How many full list enumerations will you perform?
 - Can any be easily avoided?

Full List Enumeration

- A full list enumeration is any call that enumerates over the complete list of search results
- It may not be apparent from the concise “syntactic sugar” offered by LINQ but most LINQ extension methods perform a full list enumeration under the hood

Safe versus Unsafe LINQ

- Always safe: First(), FirstOrDefault(), Skip()
- Usually safe: Take() *for small number of items*
- Not safe: Almost all other LINQ calls, including Count(), Select(), Where(), etc. are not safe with AFSearch

It's not that LINQ is inherently unsafe but many calls loop over entire collection. That's fine if the collection resides in client memory, but with AFSearch it means lots of RPC's to read all the data from the remote server.

What you will see in Demo 5

Time Permitting

- Example #1 where someone displayed the count to both a log file and console window before iterating over all results.
- The count displayed was using LINQ Count()
- Example #2 shows a re-working of #1 but reduces the full list enumerations from 3 to 1

Code Demo # 5



Demo 5 Wrap up

- GetTotalCount() more efficient than LINQ Count()
- If displaying the count more than once, you should fetch it once and save to a local variable
- LINQ calls are short and concise but may unwittingly cause more RPC's than needed

Additional Resources

When you return home

- PI Square blogs
 - [What's New in AFSearch 2.9.5 \(PI AF 2017 R2\)](#)
 - What's New in AFSearch 2.10
- See PI Square blog on FindObjectFields
 - Search “[AFFrameSearch Aggregating](#)”
- Expect to see best practices white paper on AFSearch later this year!

When you return home *(continued)*

- GitHub Repository
 - <https://github.com/Rick-at-OSIsoft/AF-SDK-TechCon-2018-AFSearch-LiveCoding>
- And finally, HELP with AFSearch is always at [Live Library](#) or your local CHM file!



- **Rick Davin**
- rdavin@osisoft.com
- Sr. Technology Enablement Engineer
- OSIsoft



- **David Hearn**
- dhearn@osisoft.com
- Group Leader, Software Engineering
- OSIsoft

Questions

Please wait for the **microphone** before asking your questions

State your **name & company**

And version of AF Client & Server



Please remember to...

Complete the Online Survey for this session



Download the Conference App for OSISOFT Users Conference 2017

- View the latest agenda and create your own
- Meet and connect with other attendees



search **OSISOFT** in the app store

Merci

谢谢

Спасибо

Danke

Gracias

Thank You

감사합니다

ありがとう

Grazie

Obrigado

See you at PI Square!