IVI Introduction

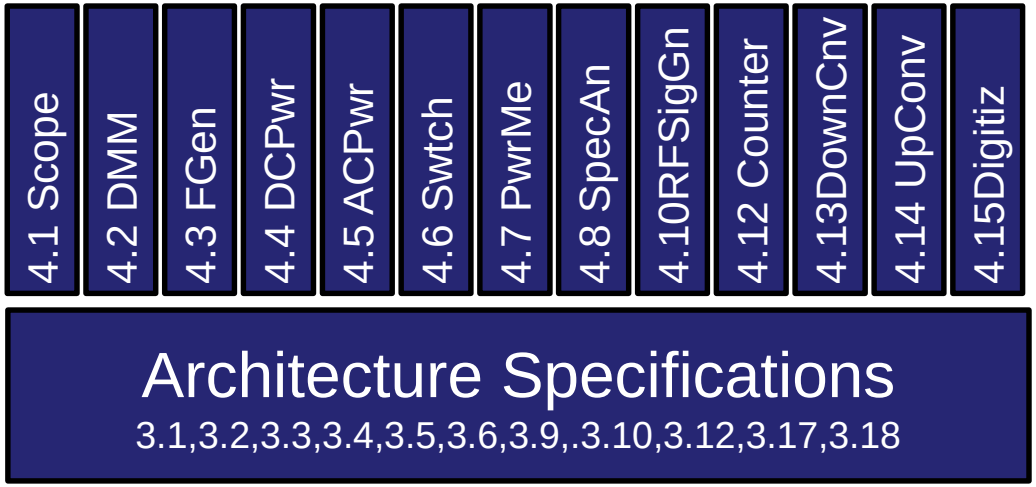
LXI™

It's about YOUR time.

www.lxistandard.org

What are IVI Drivers?

- Architecture specifications
- Instrument class specifications
- A library of shared software components



13 specs @
~220 pages

~1140 pages
of specs

IVI Driver Standards

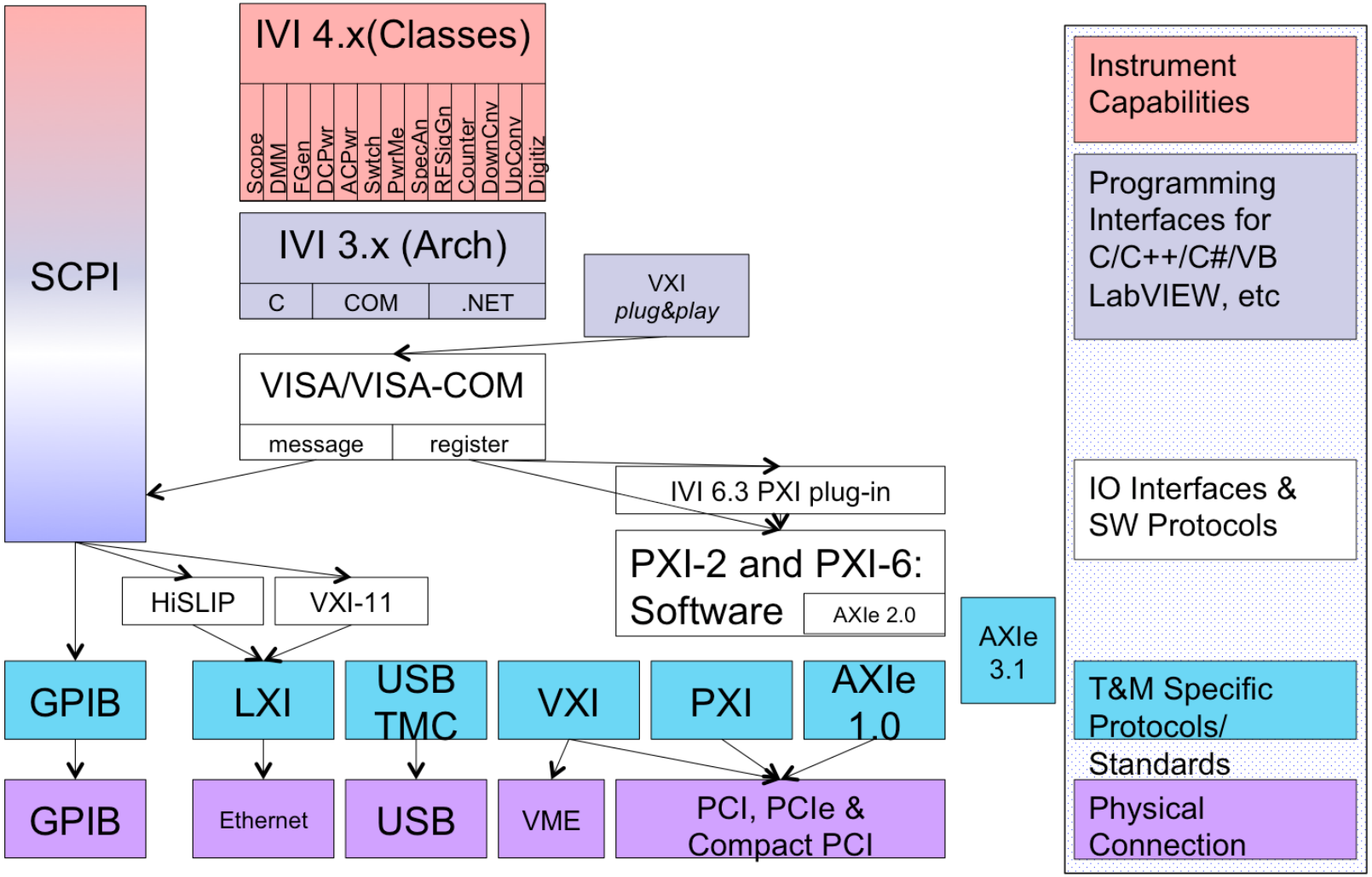
Architecture Specs

- Requirement for all drivers
- Ensures all work together
- Important for any driver
- Common Functionality
- Common Components
- Common Style
- Installation
- Driver types: C/COM/.NET

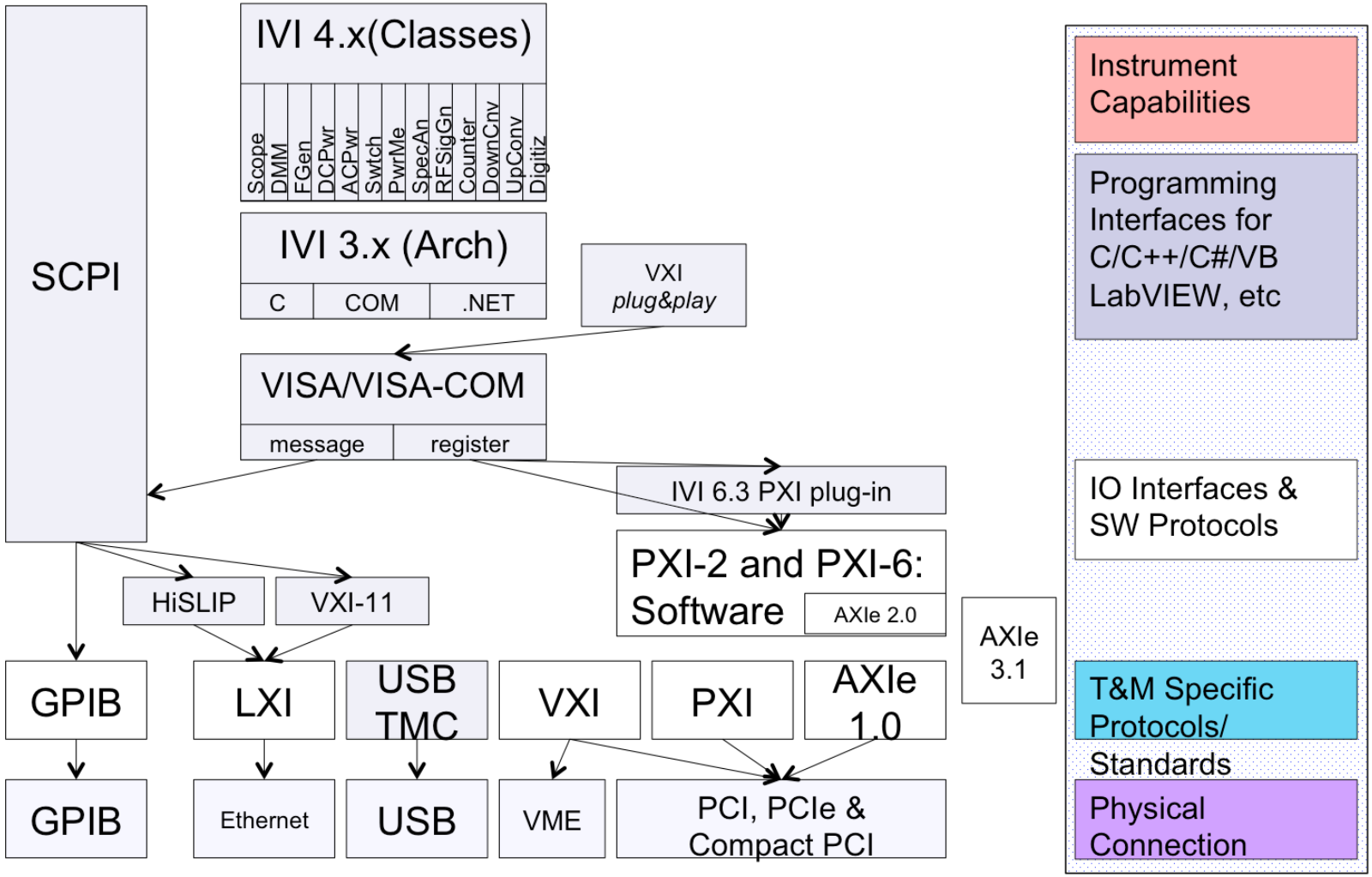
Class Specs

- Requirements for a type of instrument
- Provides syntactic interchangeability
- Establishes common paradigms for consistency
- Limited to common functionality

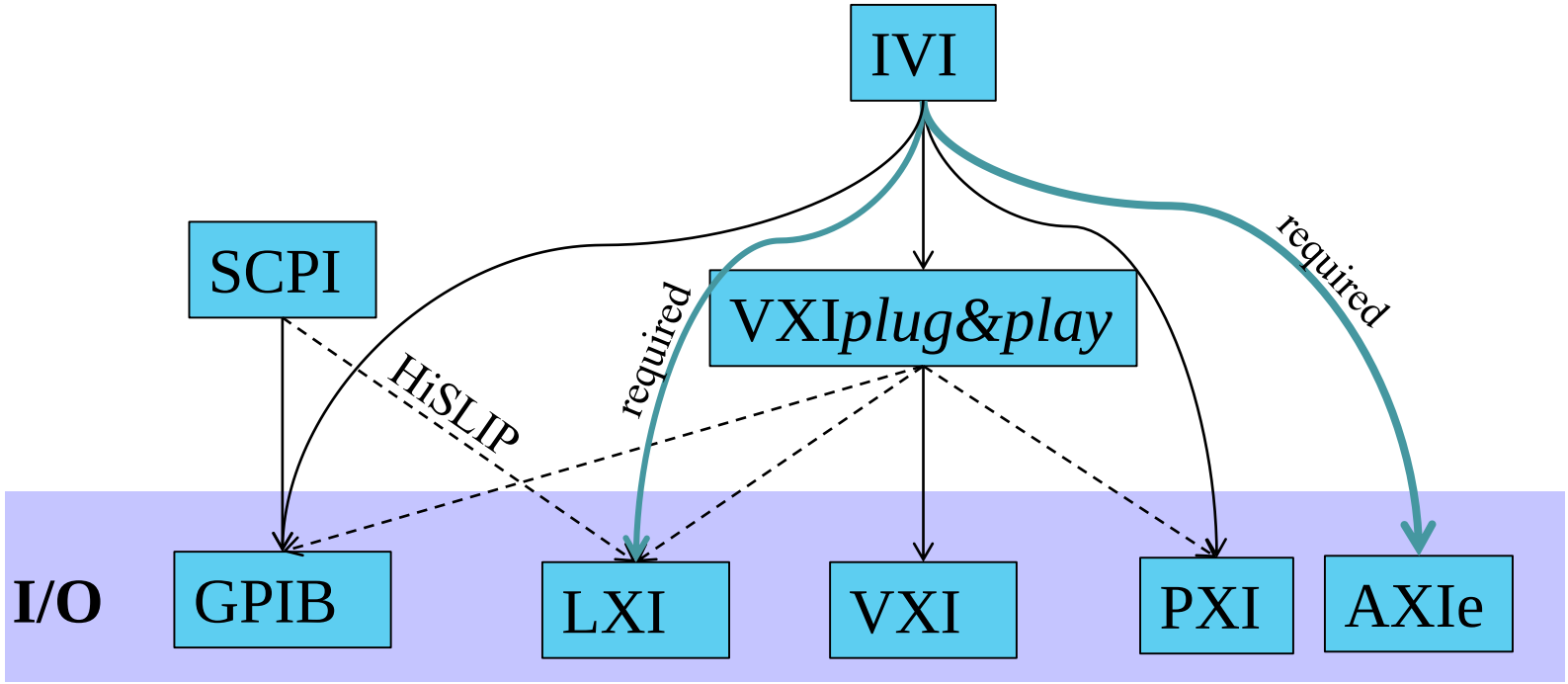
IVI Fit With Other Specs



IVI Fit With Other Specs



Instrument Control Standards



- SCPI provided necessary standards based on GPIB needs
  - Command strings natural match to GPIB
  - HiSLIP allows connection to LXI
- VXI*plug&play* added drivers necessary for VXI
  - Used with other I/O to provide necessary driver
- IVI enhances VXI*plug&play* with new features

Comparing Drivers and SCPI

Programming with SCPI

```
Status = viPrintf(vi, "MEAS:VOLT? %f, %f", range, resolution);
Status = viScanf(vi, &reading);
```

- Program deals with strings sent to/from the instrument
- Syntax errors caught by instrument when program is run
- Checking for errors requires another sequence to read error
- Simple model that requires no driver install

Programming with IVI-C

```
Status = Ag34410_MeasureDCVolt(vi, range, resolution, &reading);
```

- Program variables sent directly – no chance for syntax errors
- Syntax errors caught by compiler or editor
- No performance impact due to string manipulation
- Uses debug tools and techniques the programmers knows

Why LXI and AXIe Require IVI

- Need standard programmatic access for customers to construct multi-vendor systems
- Improves on SCPI since:
  - No performance impact due to string parsing and formatting
  - Modular instruments have no processor to parse SCPI
  - Easier to use since driver provides direct programming interface to programmer
- IVI Improves over VXIplug&play with
  - Simulation
  - Object Oriented Capabilities (for COM and .NET versions)
  - Common functions in every driver
  - Class specifications for common product types

The IVI Architectures

IVI Provides: C, COM, and .NET

- C dll for environments that use DLL's
- COM Components for COM and .NET ADE's
- .NET Assemblies for .NET ADE's

Architectures make use of same class definition

Architectures have specific rules for installation, style, etc.

Why IVI for Customers?

Uniform way of doing common tasks

- Instantiation, initialization, shutdown
- Common driver features: simulation, status checking
- Common control of driver features – state caching, error query, simulation, etc.
- Configuration and installation
  - Fixed locations for binaries, source, headers, documentation, examples
  - Proper registry entries always made
  - Common protocol to open close (standard I/O address is a big benefit)
  - Consistent solution for managing driver versions
- Standard mechanism for handling multi-channel devices

Why IVI for Customers?

Key Capabilities that simplify program development

- Syntactic Interchangeability
- Simulation
- Fine grained control through properties
- Usable in many ADE's
- Documentation of SCPI commands used by function
- DirectIO (drivers provide access to SCPI)
- Attributes for all parameters (fine grained control)
- Buildable source for message based instruments (SCPI)
- Tested using a IVI-defined process

IVI-2014

Why IVI?

One driver for any ADE

- IVI Drivers (C/COM/.NET) provide a first class experience in ***nearly any ADE***
  - Visual Basic 6
  - Visual C++
  - Visual C# and Visual Basic.NET
  - VBA (Excel, Word, PowerPoint)
  - LabVIEW
  - LabWindows/CVI
  - MATLAB
  - Agilent VEE

What is IVI Compliant?

IVI Compliant

- Common behavior model
- Support for IVI Features
  - Simulation, IO, doc, ….
- Standard install
- Common API for common tasks
  - ~40 common functions
  - Simulation, Caching, Open, Close, Initialize, SW Trigger, Status check, Version ….
- Consistent API
  - Common organization, data types, naming

Class Compliant

- Instrument Class API
- Custom API still available
  - Especially for capabilities beyond the class
- Simplifies exchanging instruments

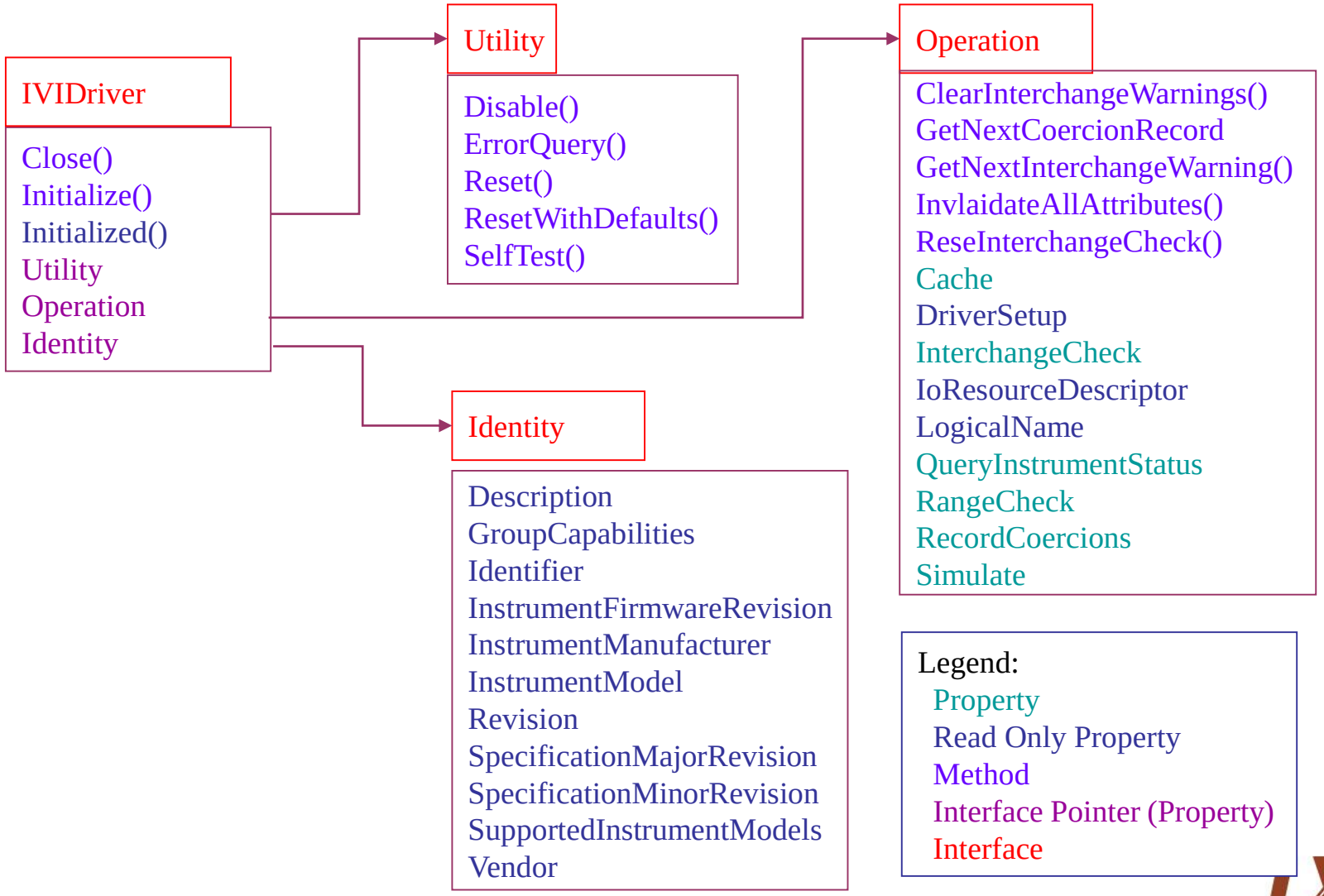
IVI Training Available this Afternoon

- IVI Foundation has training class to help understanding
- Class covers:
  - Using IVI Drivers describing C, COM and .NET
  - Examples using IVI in C and C#
  - Accessing repeated capabilities (for example multiple channels or multiple traces)
  - Using the IVI Features: Hierarchy, Simulation, Error handling, State Caching
  - IVI IO requirements
  - IVI Configuration Server and Instrument interchangeability

Conclusion

- For more information
  - IVI Web Forum: forums.ivifoundation.org
  - IVI Website: www.ivifoundation.org
  - IVI Getting Started guides: www.ivifoundation.org
  - IVI Specifications: www.ivifoundation.org
  - IVI Registration page: www.ivifoundation.org
- Most vendors have documentation and drivers on their website
- For questions on these slides, contact
  - Kirk Fertitta kirk@pacificmindworks.com
  - Joe Mueller joe_mueller@agilent.com

Intrinsic IVI API (all drivers)



Full IVI API for DMM