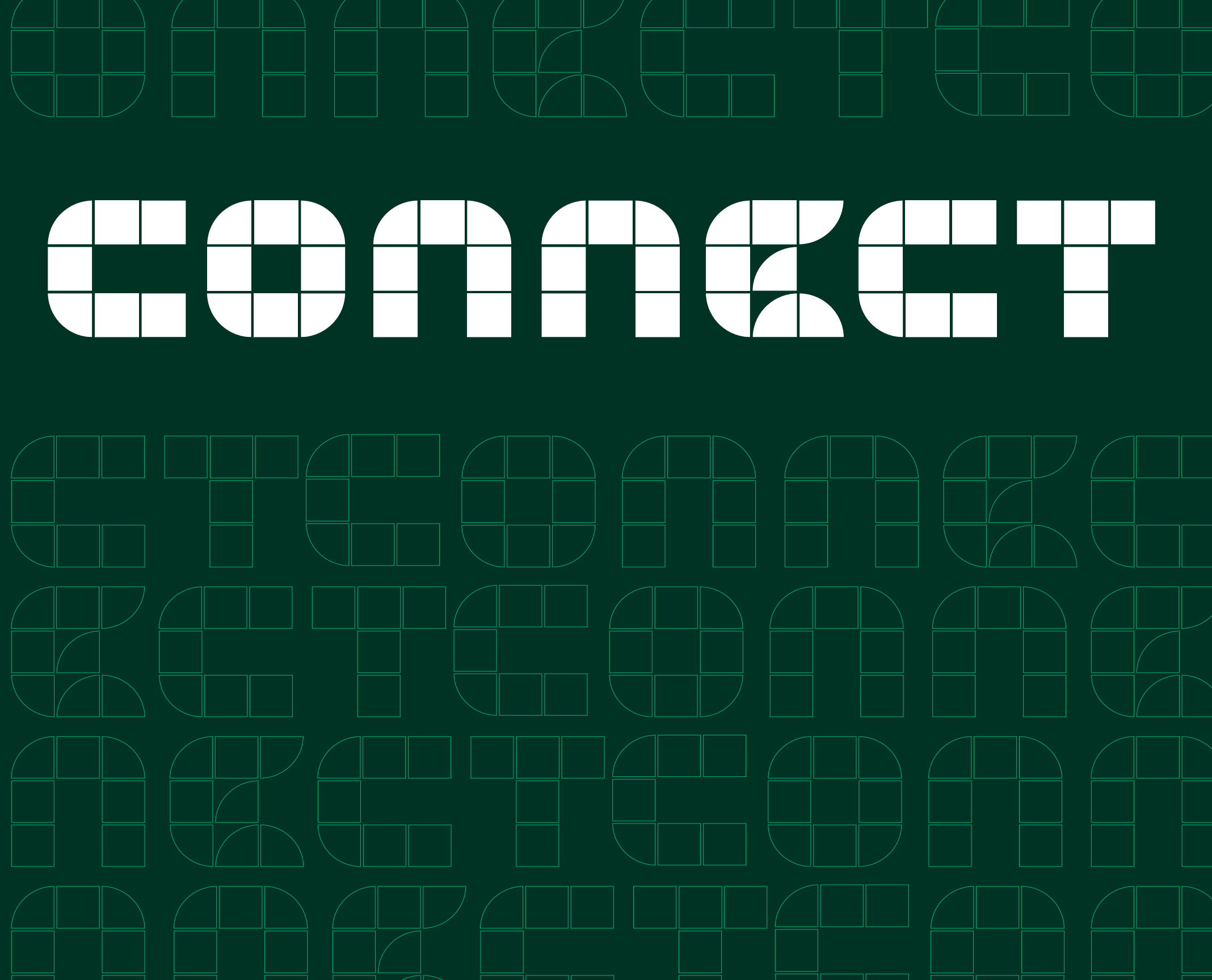
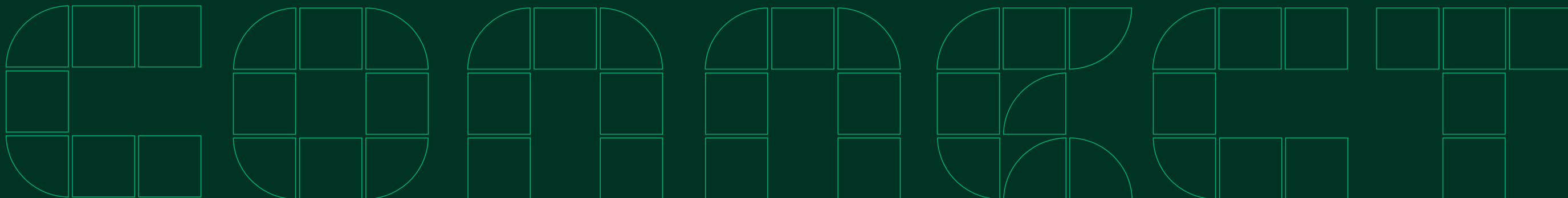


connect





Demystifying gRPC

A dive into the mysterious engine behind
InstrumentStudio Measurement Plug-Ins

Agenda

- Overview of gRPC
- Uses of gRPC at NI
- InstrumentStudio Measurement Plug-Ins



History of gRPC

- Created by Google
- Released in 2016
- Replacement of Stubby
- Performant server-to-server communication

RPC



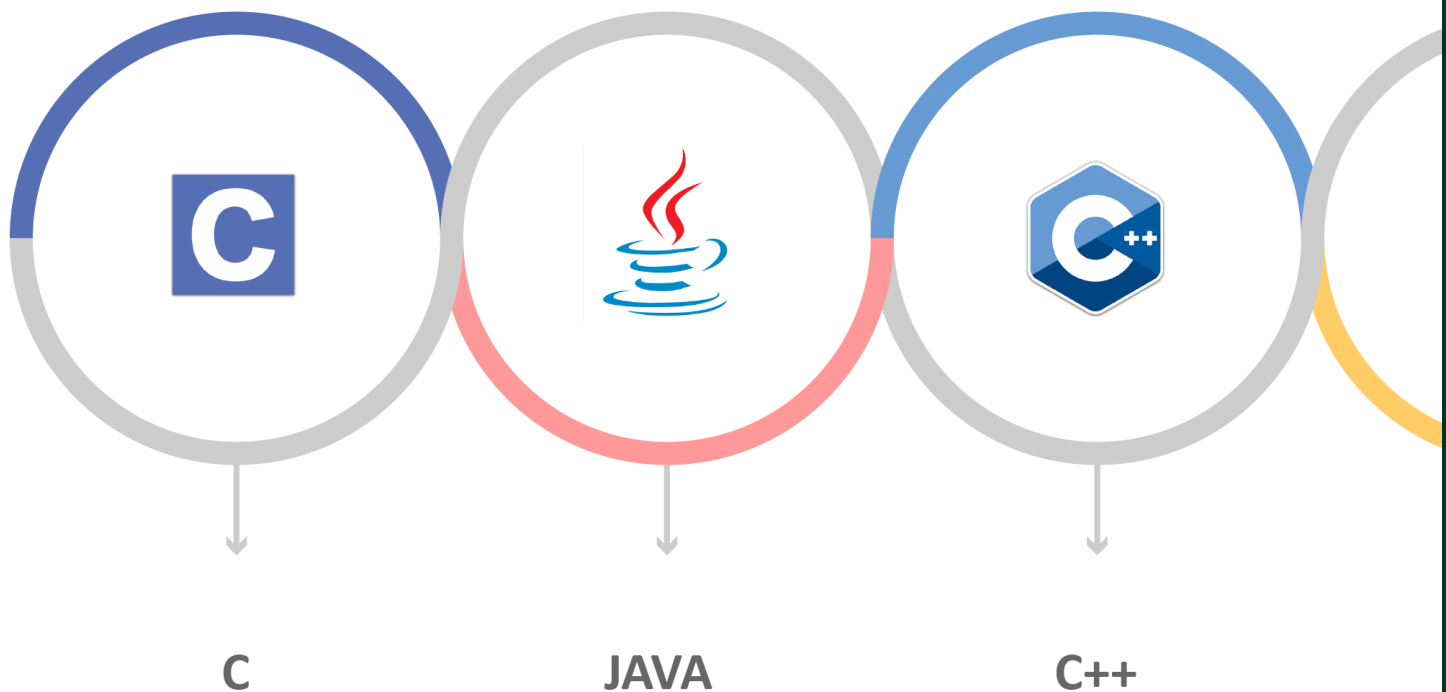
**Remote
Procedure
Call**



'g'

'g' in gRPC

- 1.0 'g' stands for 'gRPC'
- 1.1 'g' stands for 'good'
- 1.2 'g' stands for 'green'
- 1.3 'g' stands for 'gentle'
- 1.4 'g' stands for 'gregarious'
- 1.6 'g' stands for 'garcia'
- 1.7 'g' stands for 'gambit'
- 1.8 'g' stands for 'generous'
- 1.9 'g' stands for 'glossy'
- 1.10 'g' stands for 'glamorous'
- 1.11 'g' stands for 'gorgeous'
- 1.12 'g' stands for 'glorious'
- 1.13 'g' stands for 'gloriosa'
- 1.50 'g' stands for 'galley'
- 1.51 'g' stands for 'galaxy'
- 1.52 'g' stands for 'gribkoff'
- 1.53 'g' stands for 'glockenspiel'
- 1.54 'g' stands for 'gracious'
- 1.55 'g' stands for 'grandslam'
- 1.56 'g' stands for 'galvanized'
- 1.57 'g' stands for 'grounded'
- 1.58 'g' stands for 'goku'
- 1.59 'g' stands for 'generative'
- 1.60 'g' stands for 'gjallarhorn'
- 1.61 'g' stands for 'grand'
- 1.62 'g' stands for 'guardian'



Open and Secure

Open-source Apache-2.0 license

Uses HTTP/2

Supports SSL/TLS

Support in many programming languages including LabVIEW



Performant

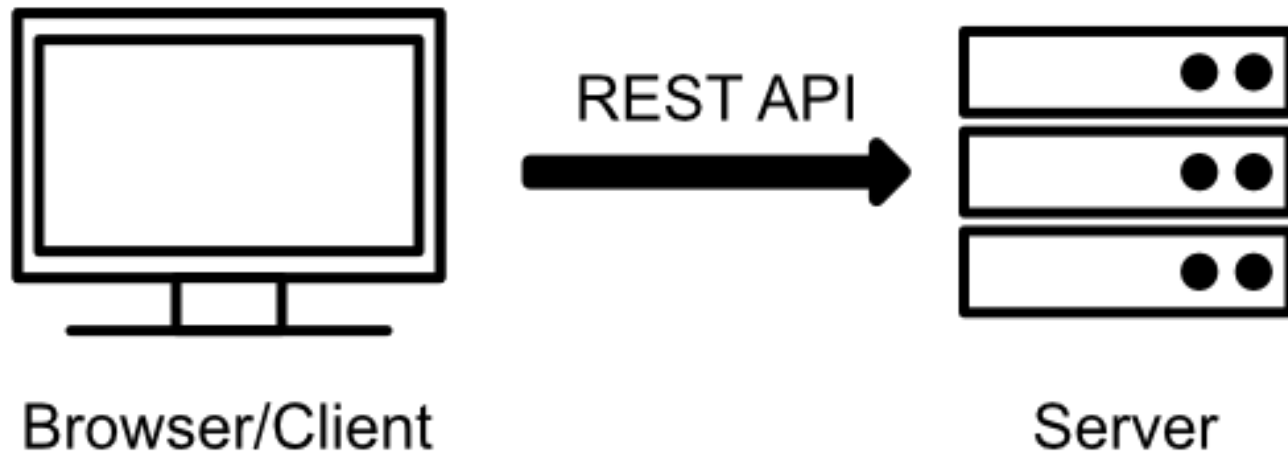
- Uses Protocol Buffers
- Binary serialization
- High performance
- Supports server to client streaming and bidirectional streaming.



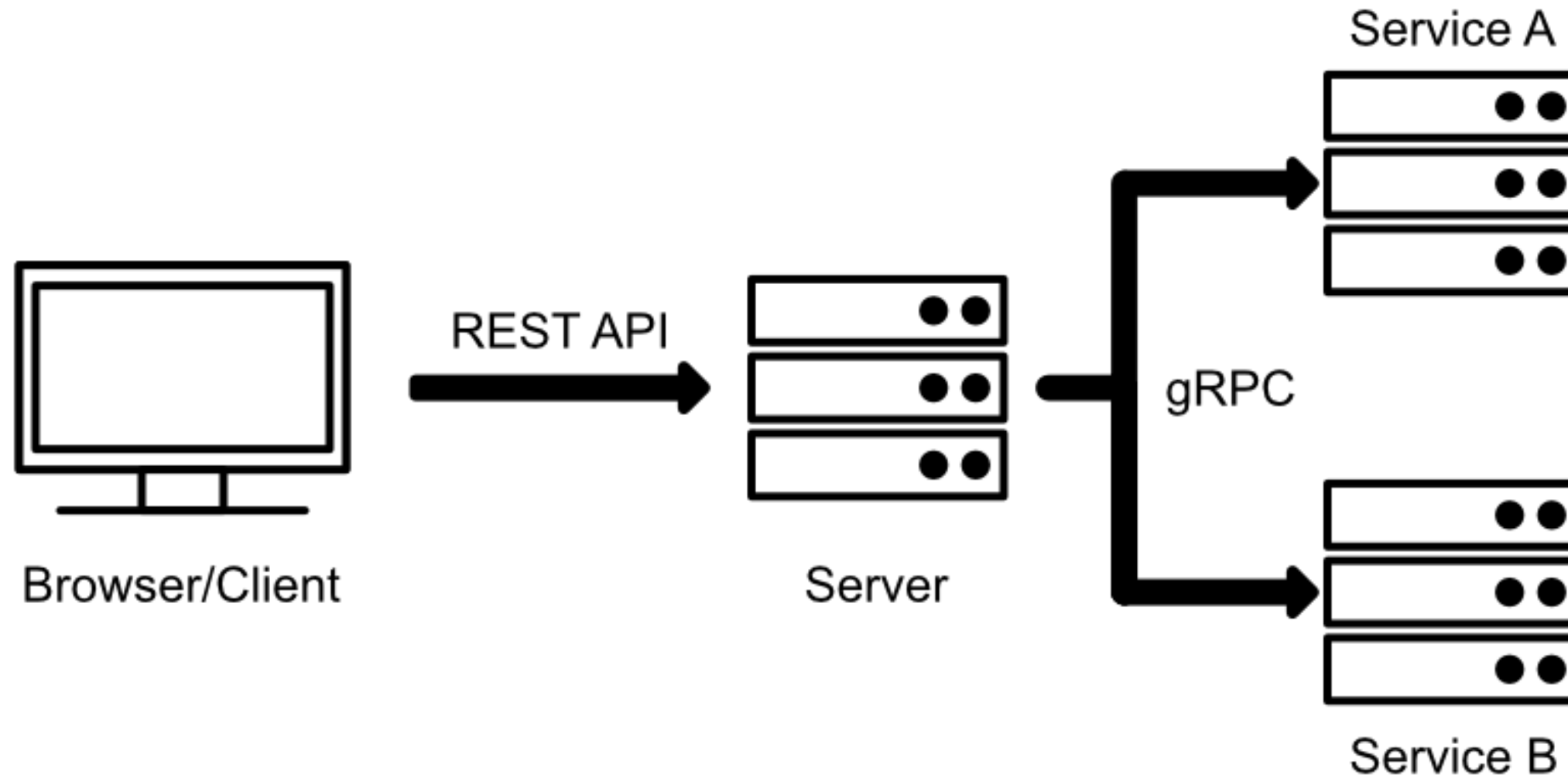
Contract First Design

- Development **starts** with the contract or interface

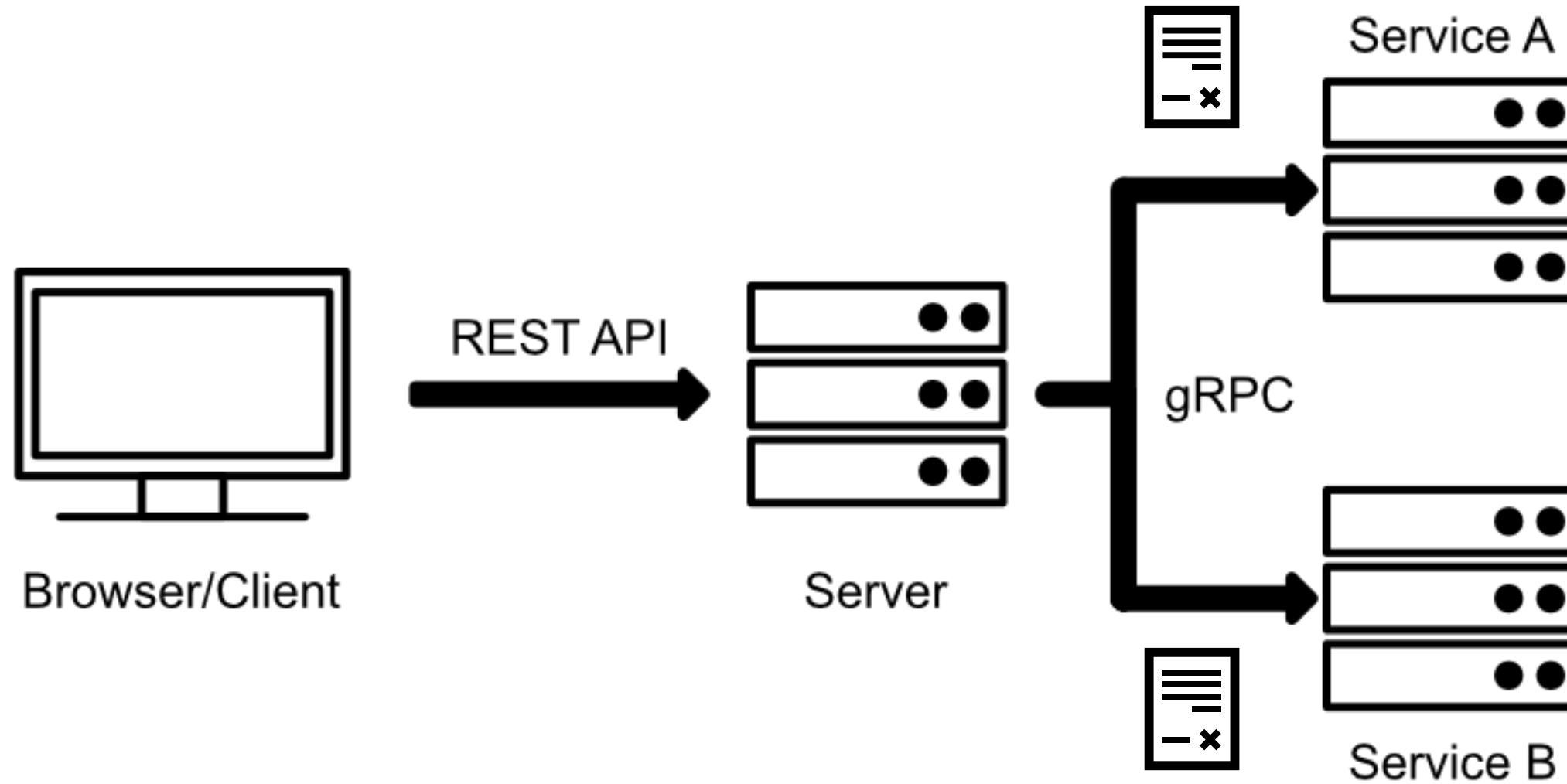
gRPC Usage



gRPC Usage

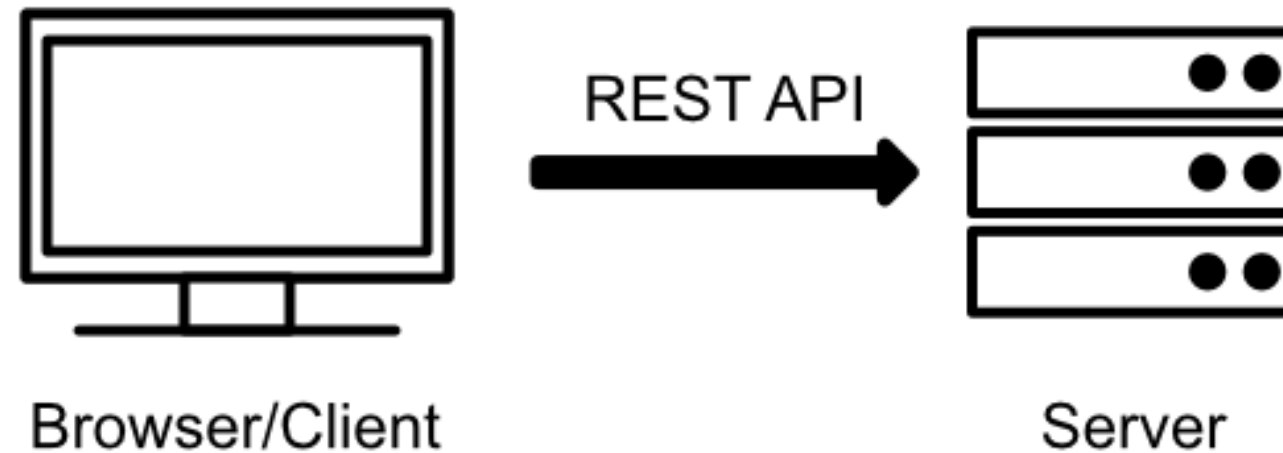


gRPC Usage



HTTP/1.1 and Text Encoding

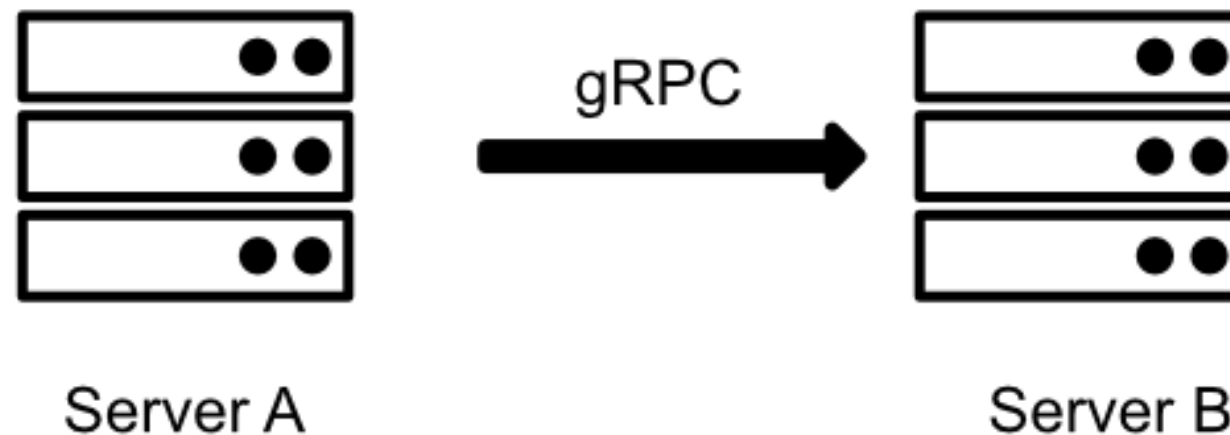
```
JSON
{
  "name": "John",
  "age": "32",
}
```



HTTP/2 and Binary Encoding

Binary Serialization

Type	Field Number	Value
0b string	00 01	4A 6F 68 6E J o h n
0a i64	00 02	32



REST vs gRPC

REST

- Suited for use in web browser.
- Uses JSON serialization
- Simple
- Stateless
- Flexible

gRPC

- Suited for server-to-server communication
- Uses binary serialization
- Supports streaming
- Strong typing
- Language agnostic API definition

Best of both

- grpc-web
 - Client-side and Bi-directional streaming not supported
- gRPC JSON transcoding
 - Expose gRPC services as a REST service



Alternatives

Thrift

GraphQL

WebSockets

RabbitMQ

Kafka

Protobuf

```
1  // The greeter service definition.
2  service Greeter {
3    // Sends a greeting
4    rpc SayHello (HelloRequest) returns (HelloReply) {}
5  }
6
7  // The request message containing the user's name.
8  message HelloRequest {
9    string name = 1;
10 }
11
12 // The response message containing the greetings
13 message HelloReply {
14   string message = 1;
15 }
```


Protobuf

Service Definition

```
1 // The greeter service definition.
2 service Greeter {
3   // Sends a greeting
4   rpc SayHello (HelloRequest) returns (HelloReply) {}
5 }
6
7 // The request message containing the user's name.
8 message HelloRequest {
9   string name = 1;
10 }
11
12 // The response message containing the greetings
13 message HelloReply {
14   string message = 1;
15 }
```


Protobuf

RPC Definition

```
1 // The greeter service definition.
2 service Greeter {
3     // Sends a greeting
4     rpc SayHello (HelloRequest) returns (HelloReply) {}
5 }
6
7 // The request message containing the user's name.
8 message HelloRequest {
9     string name = 1;
10 }
11
12 // The response message containing the greetings
13 message HelloReply {
14     string message = 1;
15 }
```

Protobuf

Request

Response

```
1 // The greeter service definition.
2 service Greeter {
3     // Sends a greeting
4     rpc SayHello (HelloRequest) returns (HelloReply) {}
5 }
6
7 // The request message containing the user's name.
8 message HelloRequest {
9     string name = 1;
10 }
11
12 // The response message containing the greetings
13 message HelloReply {
14     string message = 1;
15 }
```

Protobuf

Message Definition →

```
1  // The greeter service definition.
2  service Greeter {
3    // Sends a greeting
4    rpc SayHello (HelloRequest) returns (HelloReply) {}
5  }
6
7  // The request message containing the user's name.
8  message HelloRequest {
9    string name = 1;
10 }
11
12 // The response message containing the greetings
13 message HelloReply {
14   string message = 1;
15 }
```

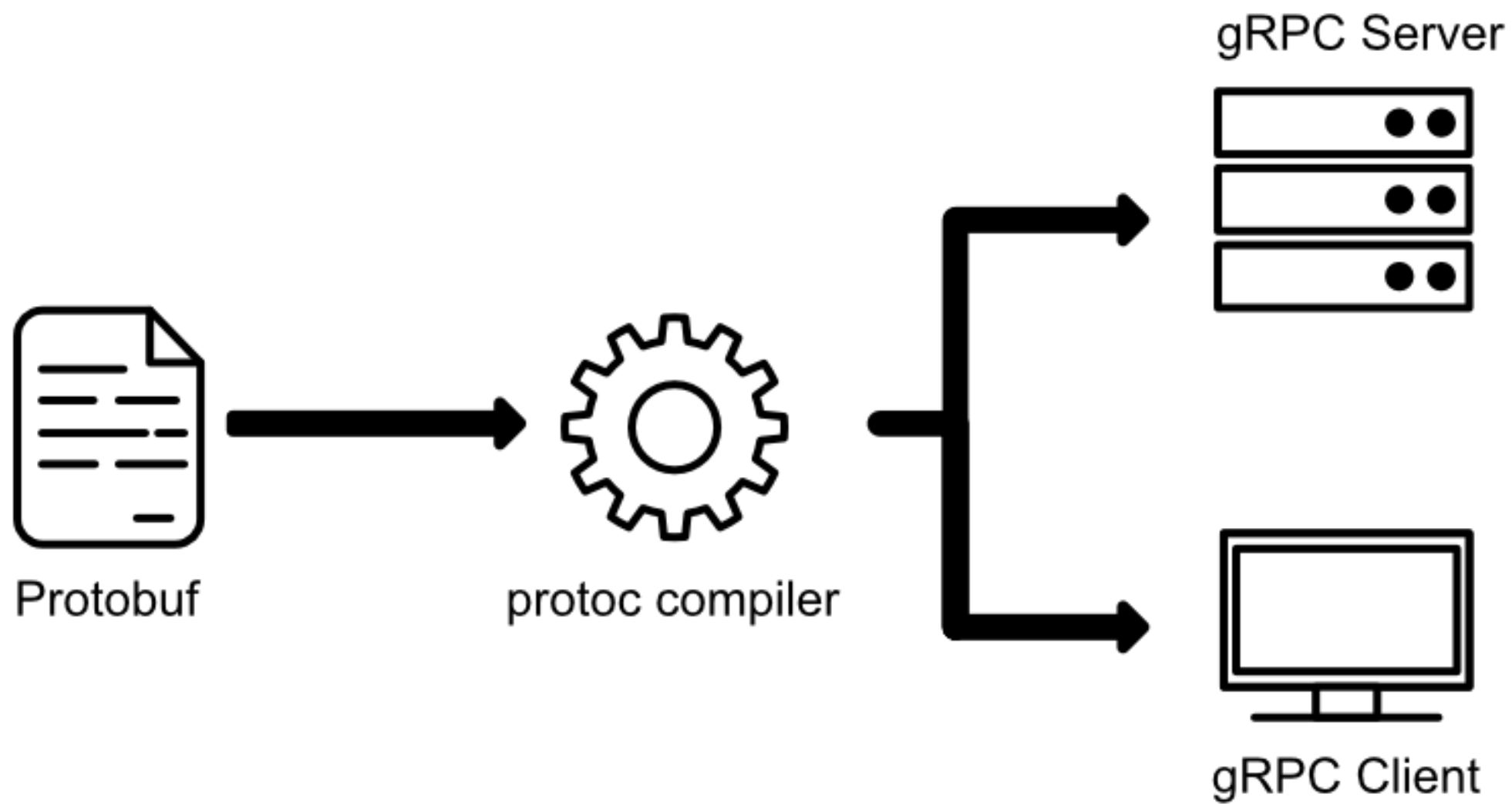
Protobuf

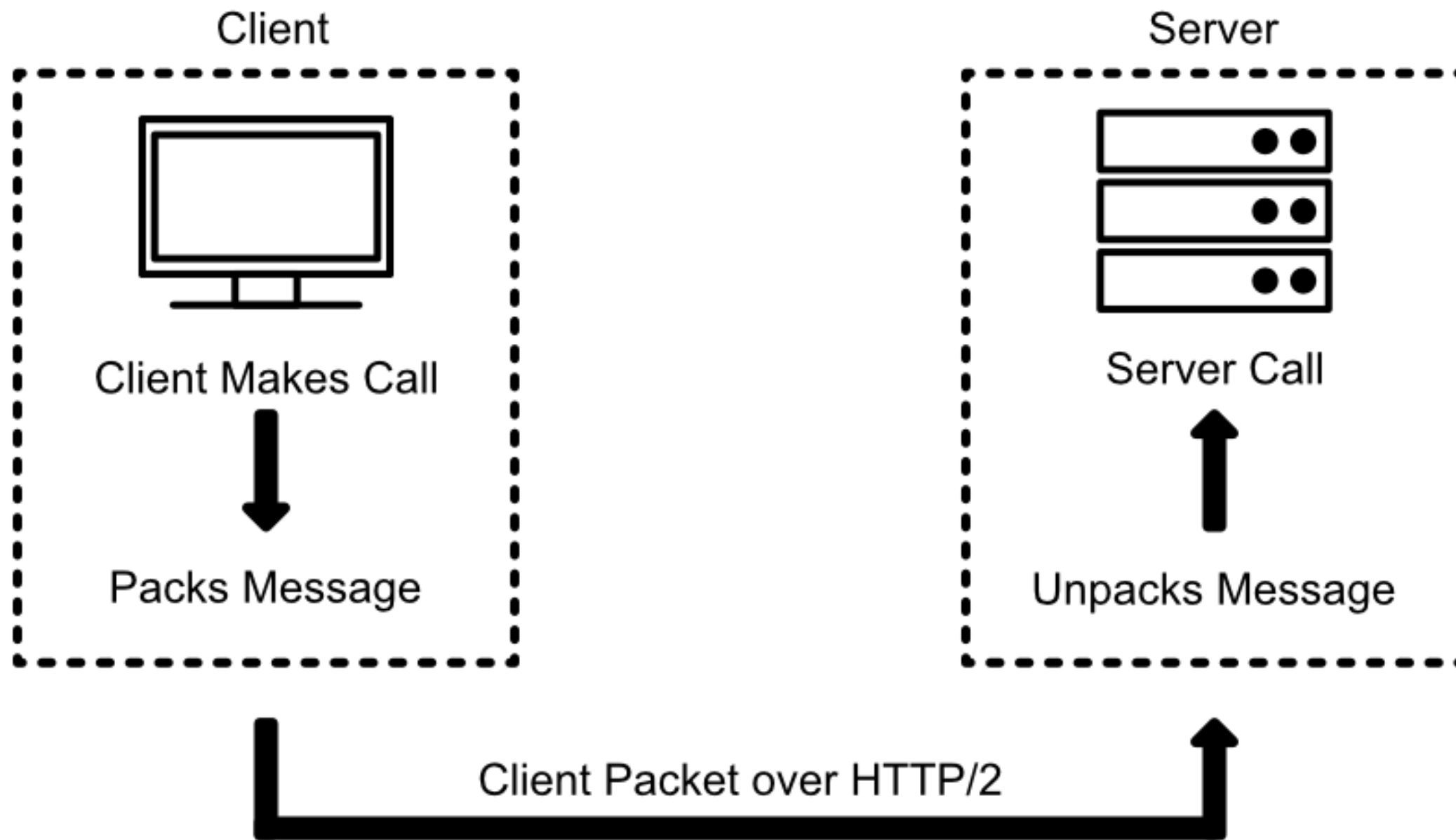
Fields

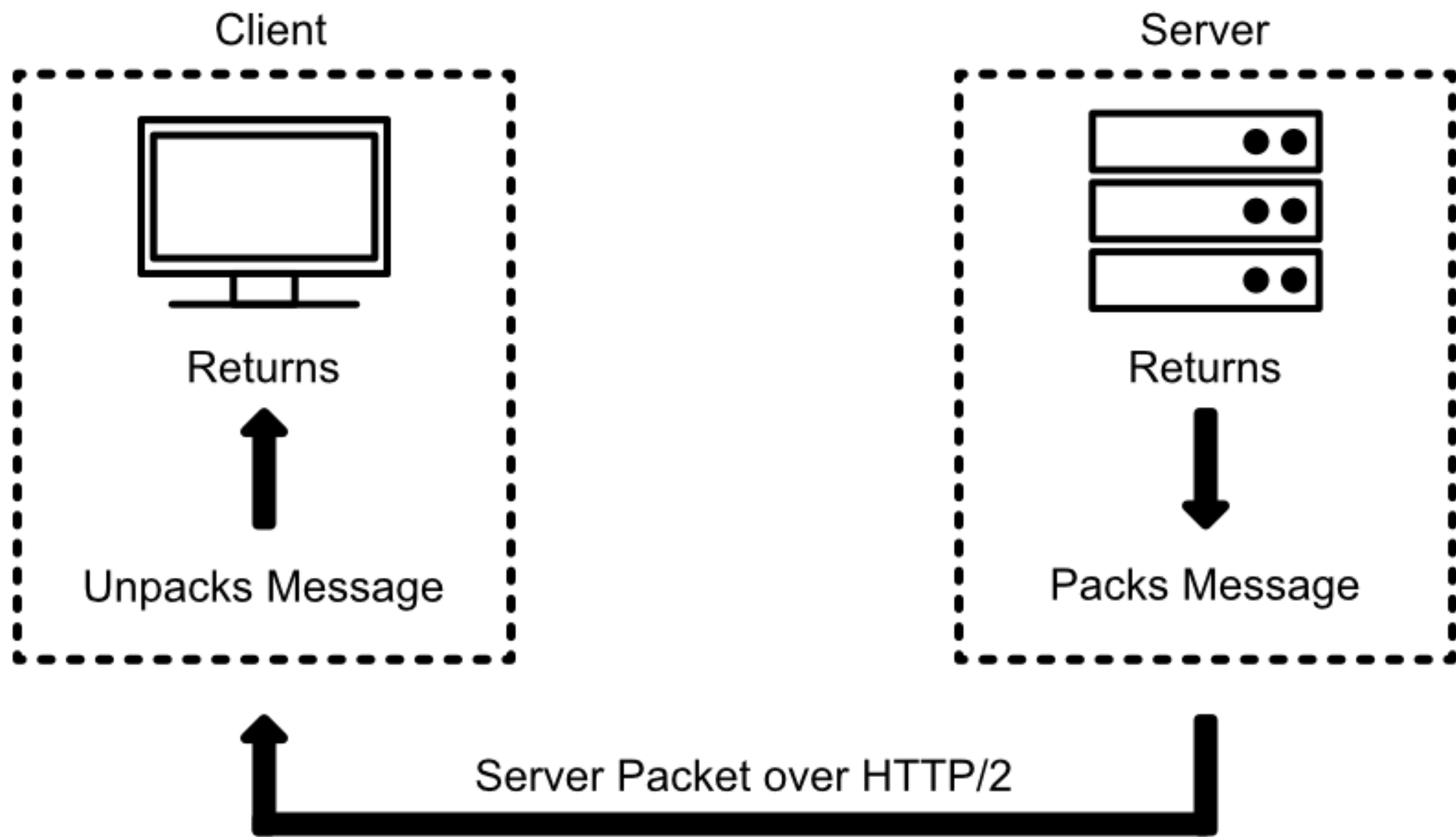
```
1  // The greeter service definition.
2  service Greeter {
3    // Sends a greeting
4    rpc SayHello (HelloRequest) returns (HelloReply) {}
5  }
6
7  // The request message containing the user's name.
8  message HelloRequest {
9    string name = 1;
10 }
11
12 // The response message containing the greetings
13 message HelloReply {
14   string message = 1;
15 }
```

Message Fields

- Numbered starting at 1
- Numbered to support binary serialization
- Can be marked optional
- Can be marked repeated
- Can be marked oneof
- Can be:
 - double, float, int32, int64, uint32, uint64, ..., bool, string, bytes.
 - Other message types
 - Enums
 - Maps
 - Any



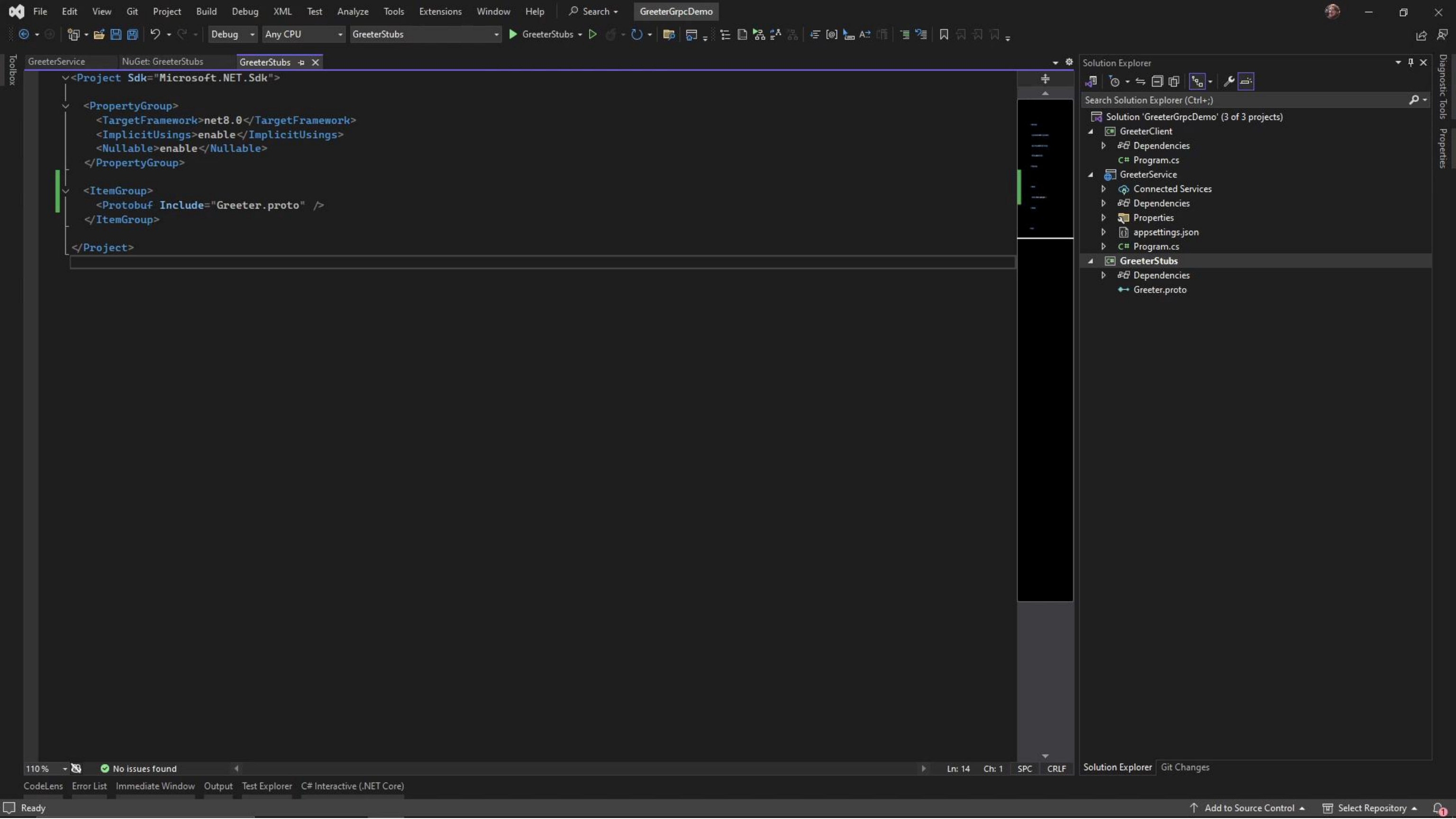




Calling a gRPC Service

- Protobuf or Client API
- IP address and port

```
1  with grpc.insecure_channel('localhost:5001') as channel:  
2  
3      stub = helloworld_pb2_grpc.GreeterStub(channel)  
4      response = stub.SayHello(helloworld_pb2.HelloRequest(name='you'))  
5      print("Greeter client received: " + response.message)  
6  
7      response = stub.SayHelloAgain(helloworld_pb2.HelloRequest(name='you'))  
8      print("Greeter client received: " + response.message)
```



GreeterService NuGet: GreeterStubs GreeterStubs

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>
  <ItemGroup>
    <Protobuf Include="Greeter.proto" />
  </ItemGroup>
</Project>
```

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'GreeterGrpcDemo' (3 of 3 projects)

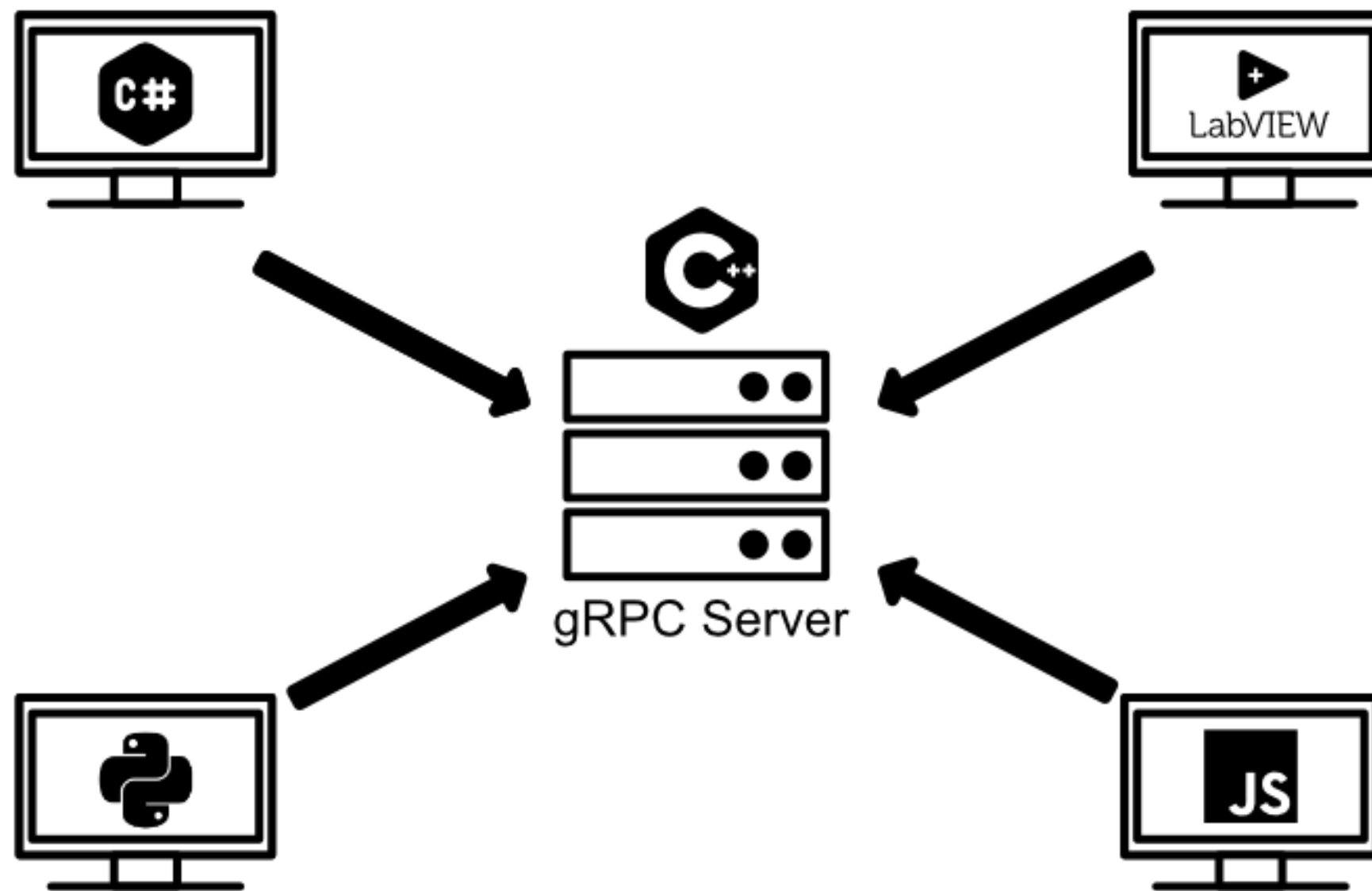
- GreeterClient
 - Dependencies
 - Program.cs
- GreeterService
 - Connected Services
 - Dependencies
 - Properties
 - appsettings.json
 - Program.cs
- GreeterStubs
 - Dependencies
 - Greeter.proto

110 % No issues found Ln: 14 Ch: 1 SPC CRLF

CodeLens Error List Immediate Window Output Test Explorer C# Interactive (.NET Core)

Solution Explorer Git Changes

Ready Add to Source Control Select Repository



gRPC use at NI

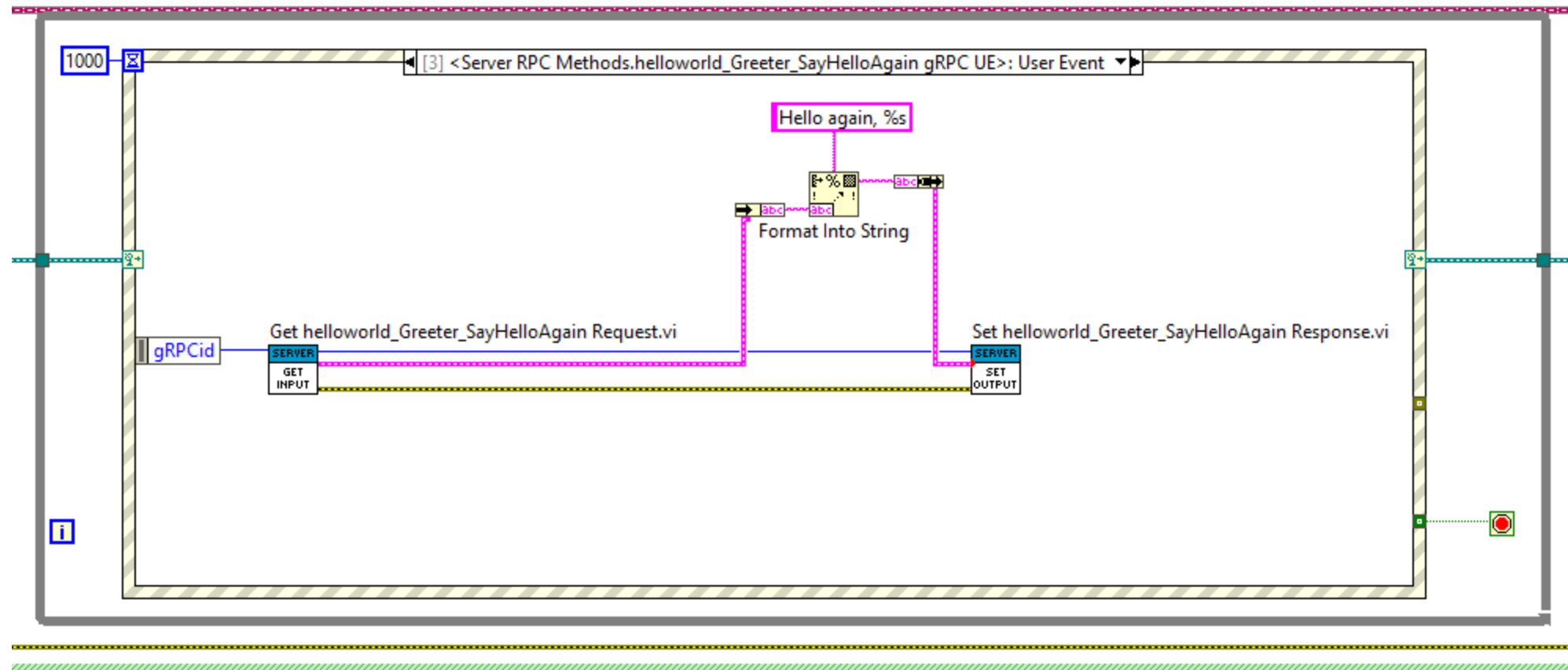
ni-sujain and github-actions[bot] VIPB version bump ✓

.github/workflows	Update sync_issues_to_azdo.y
build-it	fix the build scripts to handle
docs	Fixed gRPC Client Hang issue
examples	Added comments (#329)
labview source	VIPB version bump
src	fixing crashes related to Repe
tests	Improved cmake tests config
third_party	Revert "update to grpc 1.62.0
.gitignore	Revert "update to grpc 1.62.0
.gitmodules	Fix linux build

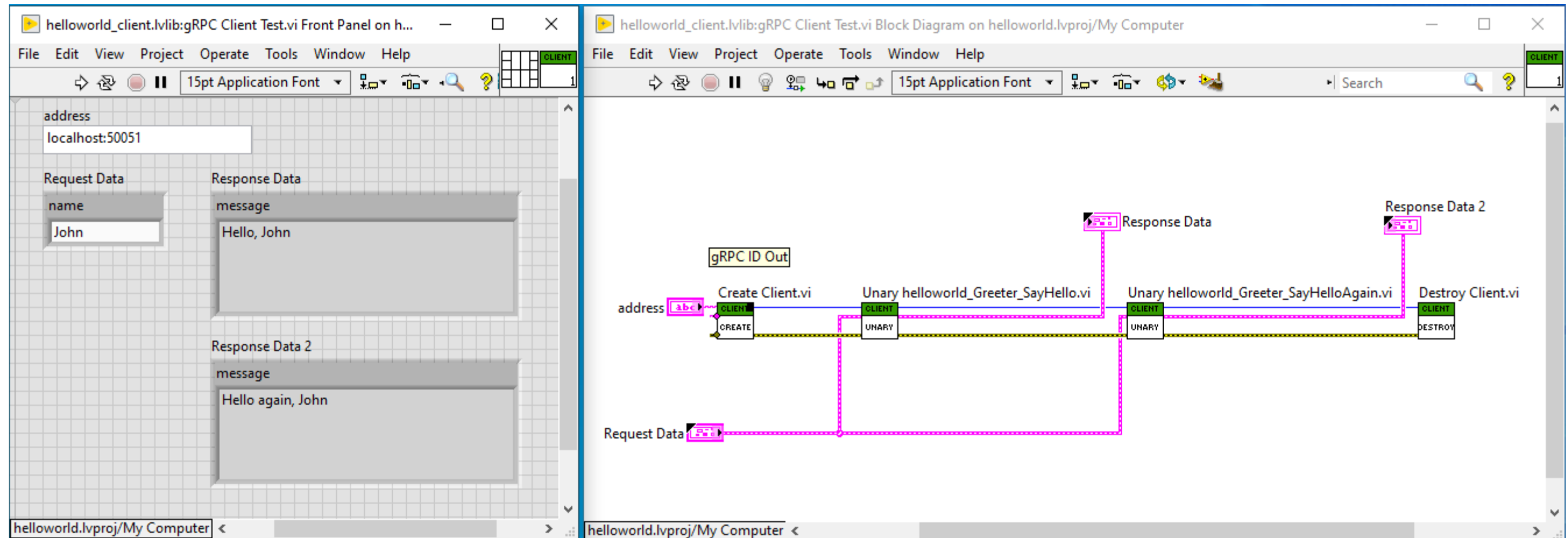
grpc-labview

- Create gRPC clients and servers in LabVIEW
- Open source
- MIT licensed

gRPC Server - LabVIEW



gRPC Client - LabVIEW



RahulBenaka and RahulNagaraju15 Updated gRPC Device code for BT, LTE an...	
.github	Update AzDO area path (#104
cmake	Fix venv creation on GH runner
examples	Adding 6 python examples to
generated	Updated gRPC Device code fo
imports	Updated gRPC Device code fo
source	Updated gRPC Device code fo
third_party	Update grpc to v1.51.1 (#819)
.clang-format	[Owners] Update clang-forma
.gitignore	Add new executable flag to ch
.gitmodules	Convert between UTF-8 (gRPC
CMakeLists.txt	Bumped version in CMakeList

grpc-device

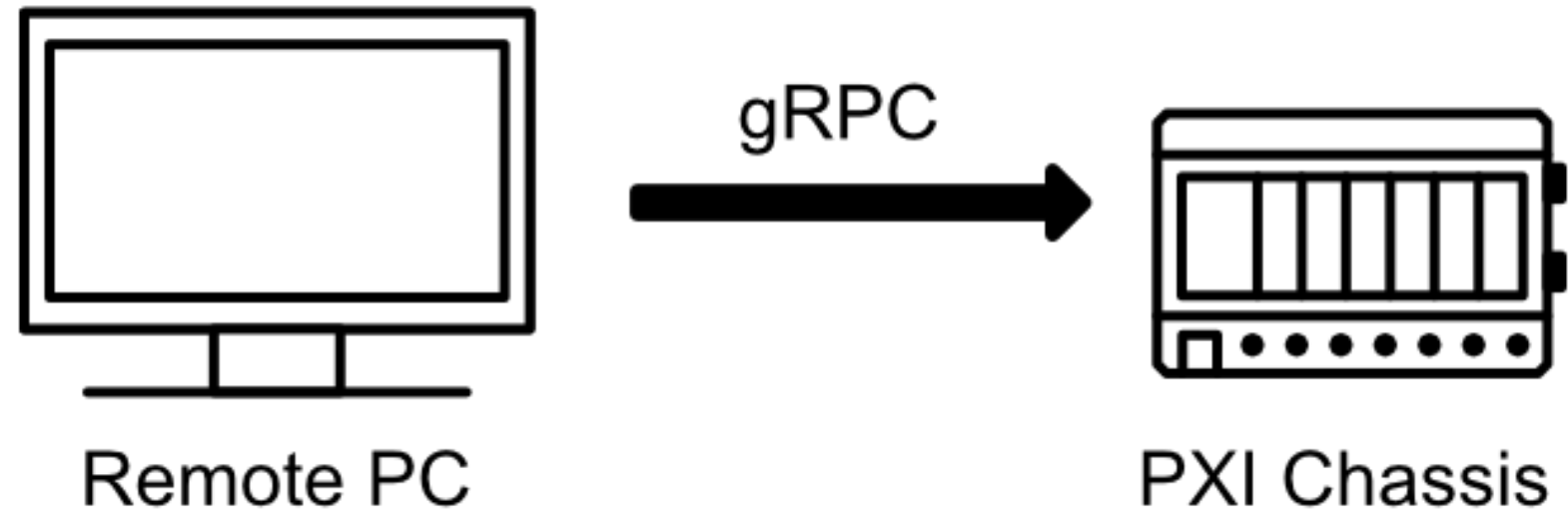
- gRPC Server and Client APIs allowing NI's instrumentation to be accessed and controlled remotely
- Open source
- MIT licensed

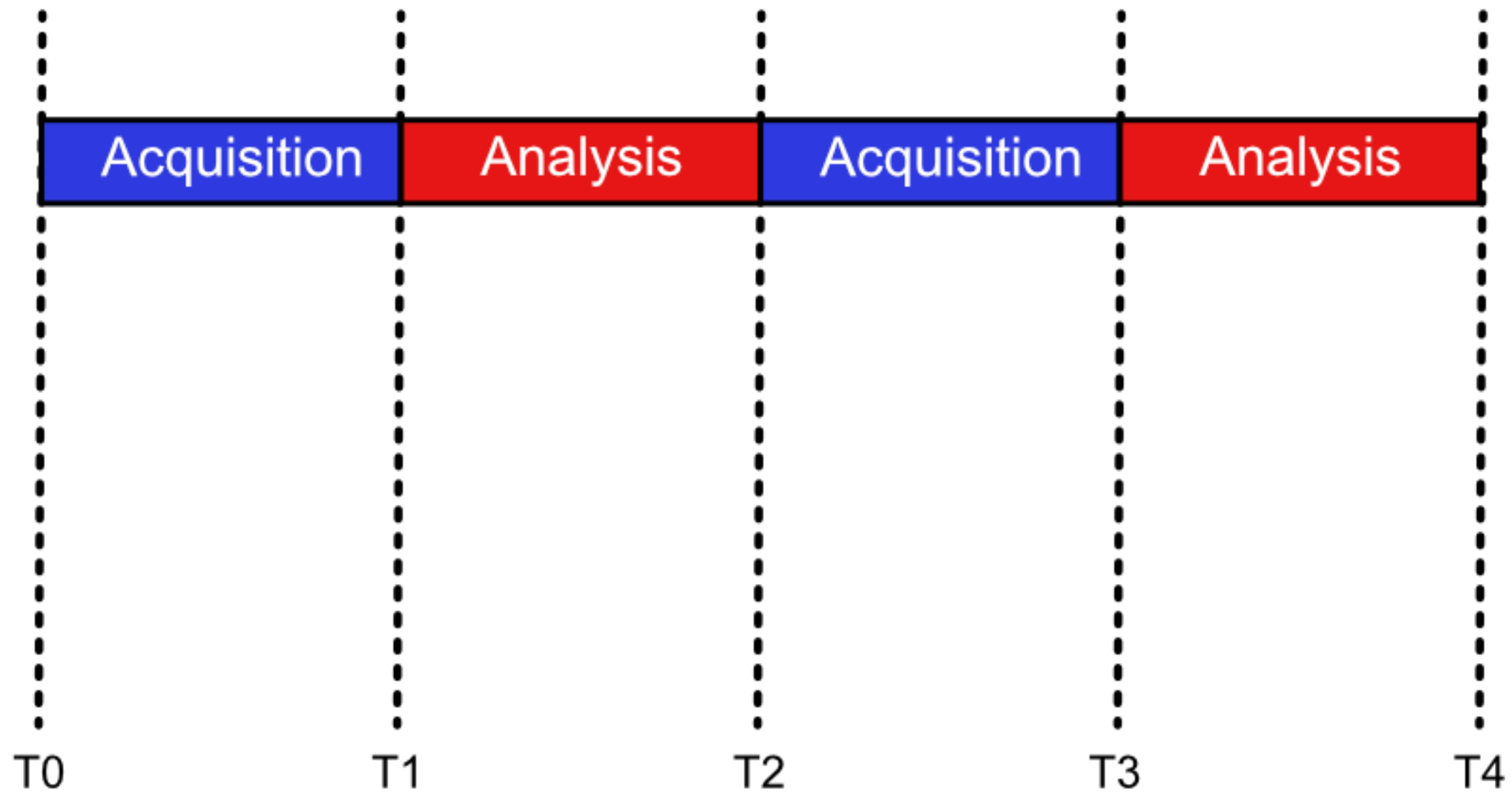
providing remote access to
APIs.

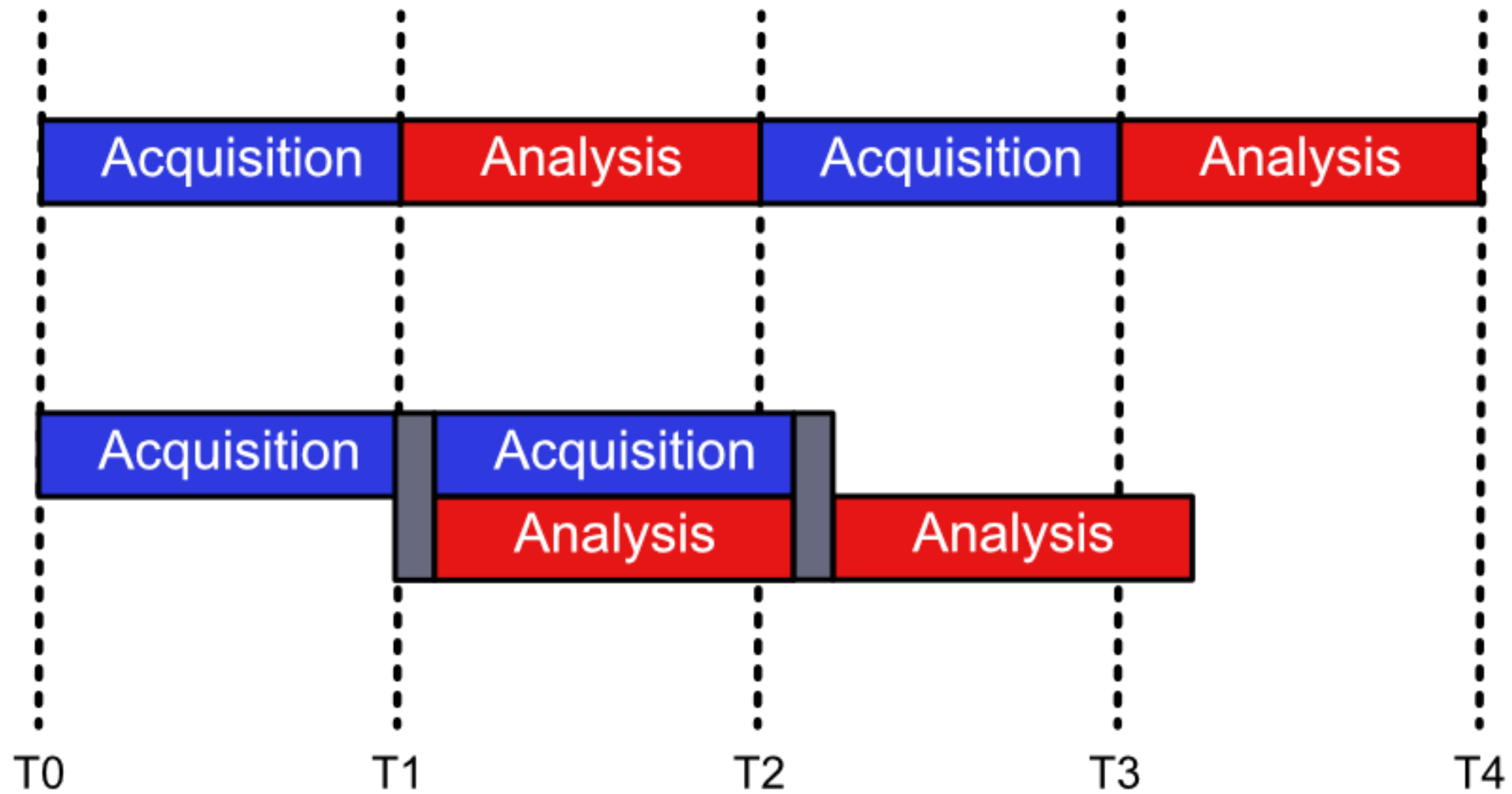
- ni-daqmx
- ni-dmm
- ni-switch
- ni-digital
- ni-xnet
- ni-tclk
- ni-sync
- ni-rfsa
- ni-rfmw-nr
- ni-rfmw-lte
- ni-rfmw-wlan

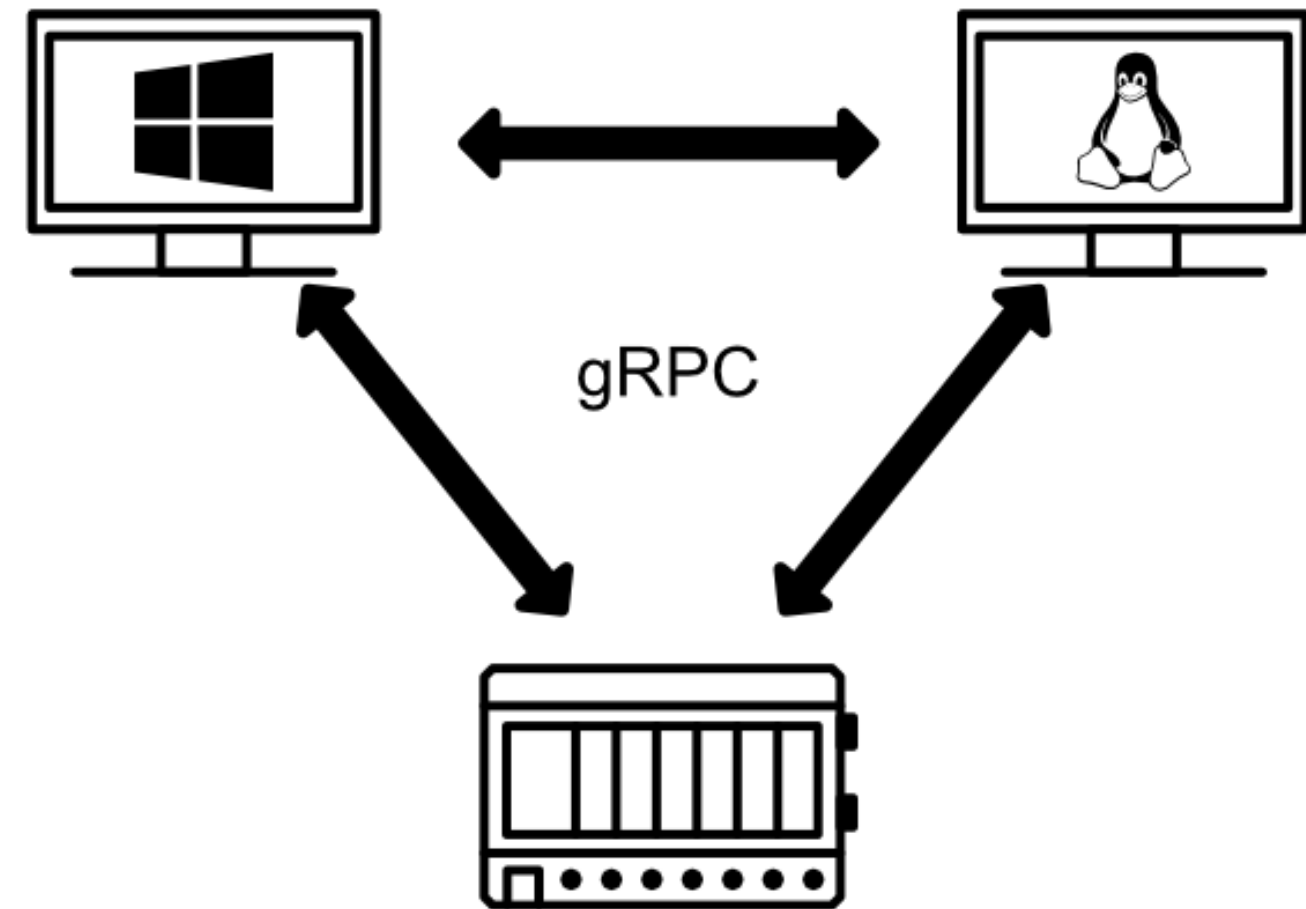
NI Driver	Version Tested (Windows)	Version Tested (Linux)	Version Tested (Linux RT)
NI-DAQmx	2023 Q1	2023 Q1	2023 Q1
NI-DCPower	2023 Q1	2023 Q1	2023 Q1
NI-Digital Pattern Driver	2023 Q1	Not Supported	Not Supported
NI-DMM	2023 Q1	2023 Q1	2023 Q1
NI-FGEN	2023 Q1	2023 Q1	2023 Q1
NI-RFmx Bluetooth	2024 Q2	Not Supported	Not Supported
NI-RFmx CDMA2k	2023 Q1	Not Supported	Not Supported
NI-RFmx Demod	2023 Q1	Not Supported	Not Supported
NI-RFmx GSM	2023 Q1	Not Supported	Not Supported
NI-RFmx LTE	2024 Q2	Not Supported	Not Supported
NI-RFmx NR	2024 Q1	Not Supported	Not Supported
NI-RFmx SpecAn	2024 Q2	Not Supported	Not Supported
NI-RFmx TD-SCDMA	2023 Q1	Not Supported	Not Supported
NI-RFmx WCDMA	2023 Q1	Not Supported	Not Supported
NI-RFmx WLAN	2024 Q1	Not Supported	Not Supported
NI-RFSA	21.0.0	21.0.0	Not Supported
NI-RFSG	21.0.0	21.0.0	Not Supported
NI-SCOPE	2023 Q2	2023 Q2	2023 Q2
NI-SWITCH	2023 Q1	2023 Q1	2023 Q1
NI-TClk	2023 Q1	2023 Q1	2023 Q1
NI-VISA	2024 Q1	Not Supported	Not Supported
NI-XNET	21.5.0	21.5.0	21.5.0

grpc-device





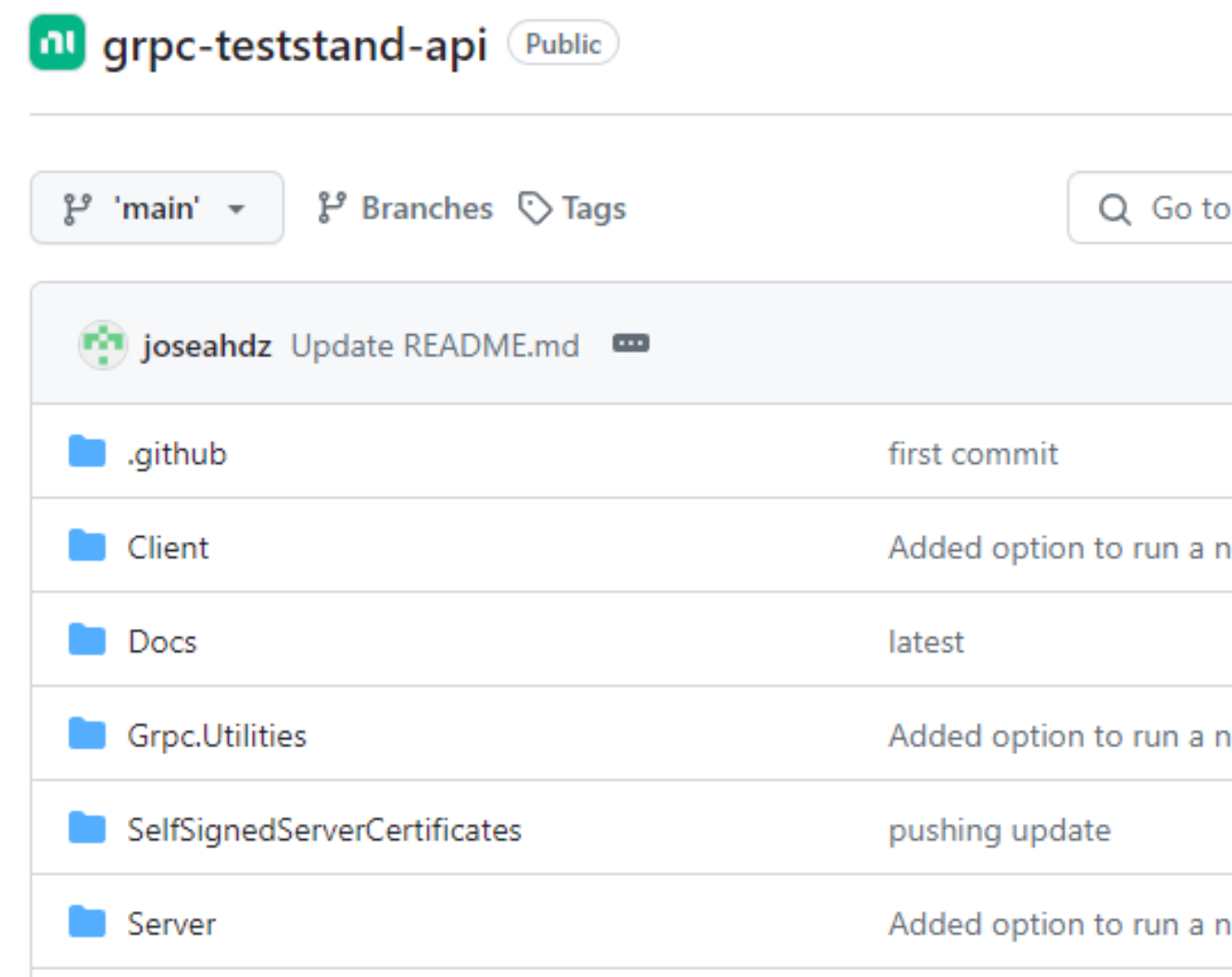




Customer Anecdote

- Standardization
- Cross-platform support
- Test time optimization

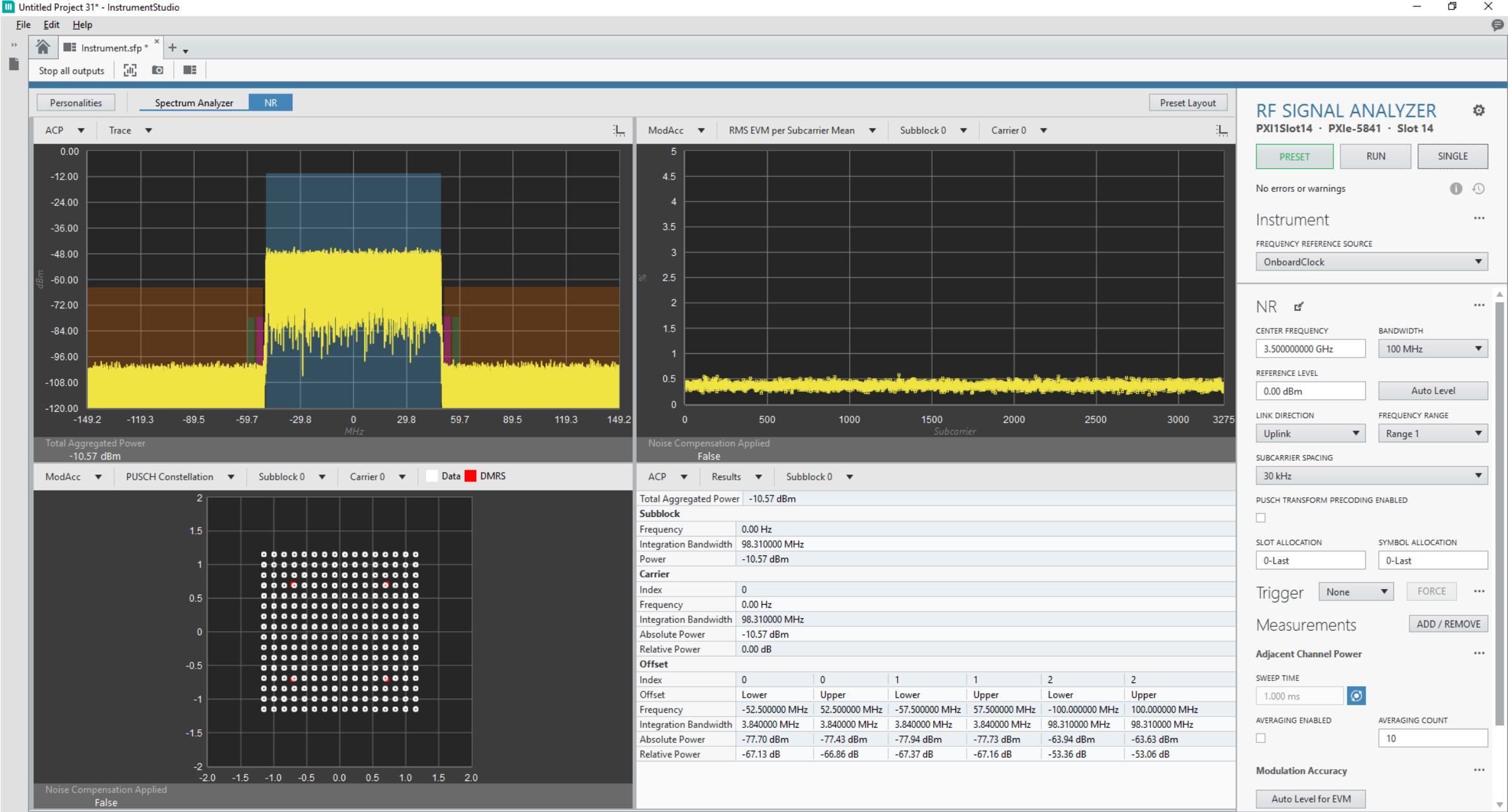
Chad Erickson – California FAE



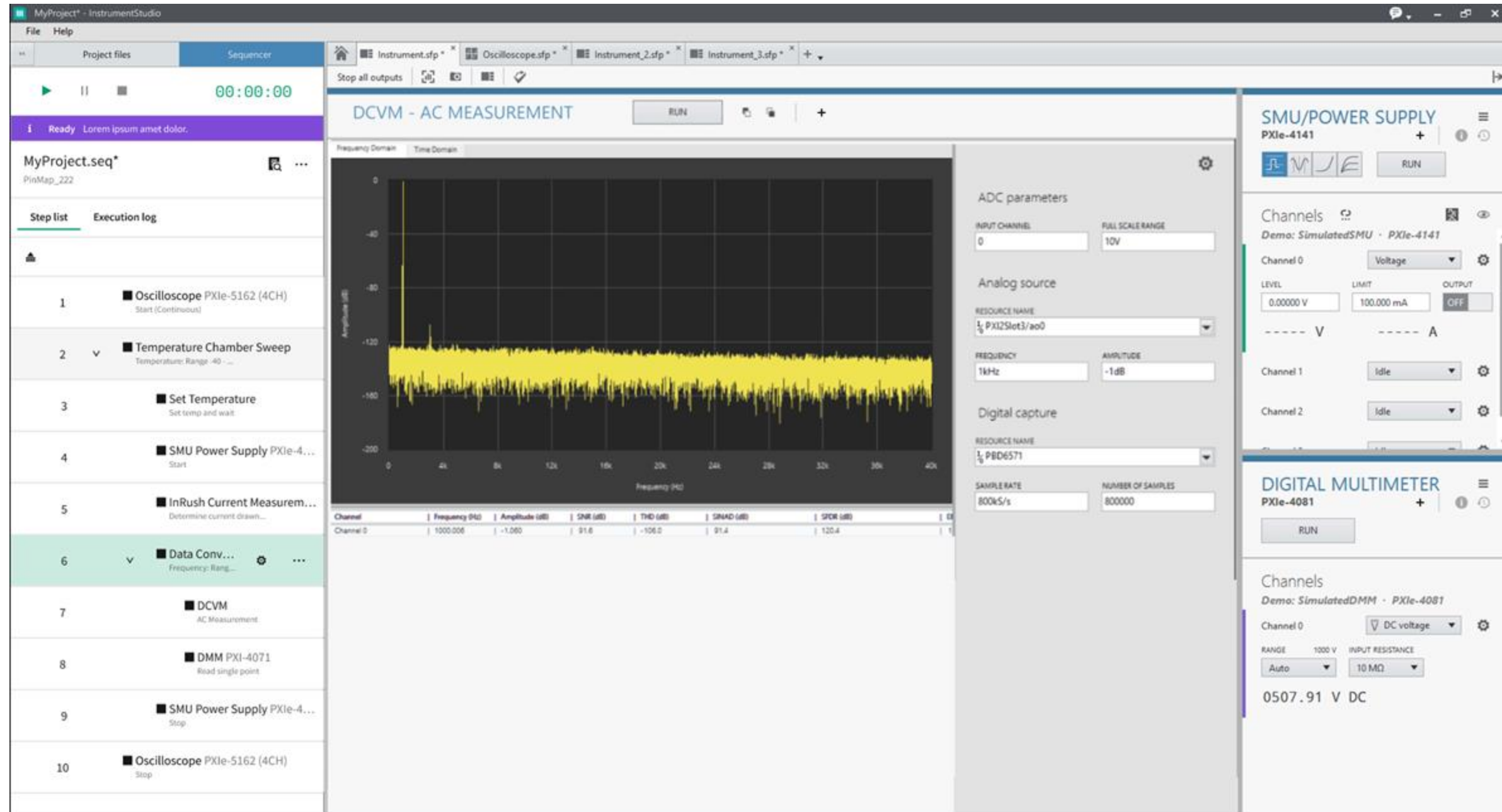
grpc-teststand-api

- Execute TestStand sequences remotely
- Early access
- Open source
- MIT licensed

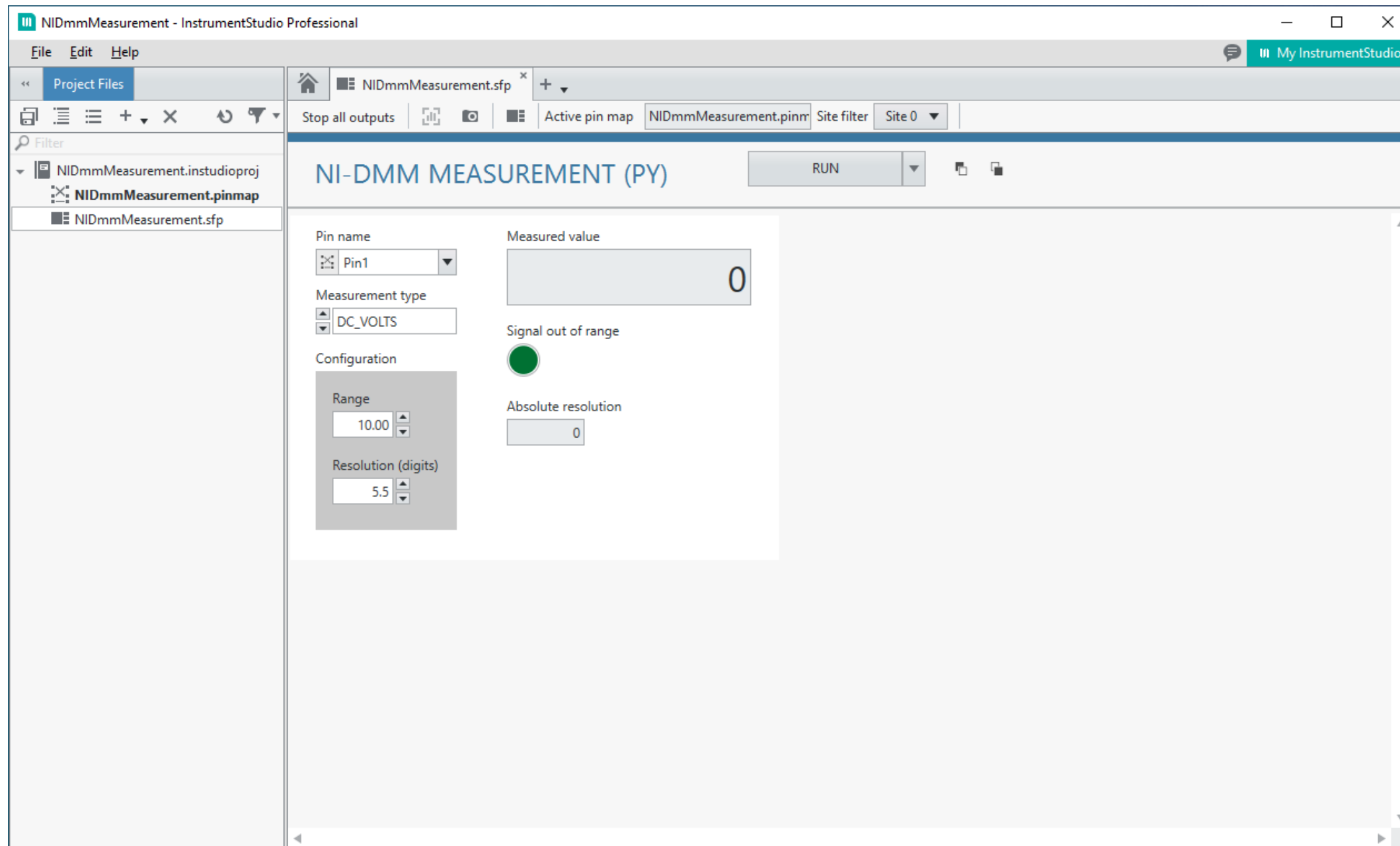
InstrumentStudio RF Remote Panel



InstrumentStudio Pro Sequencer



InstrumentStudio Measurement Plug-Ins

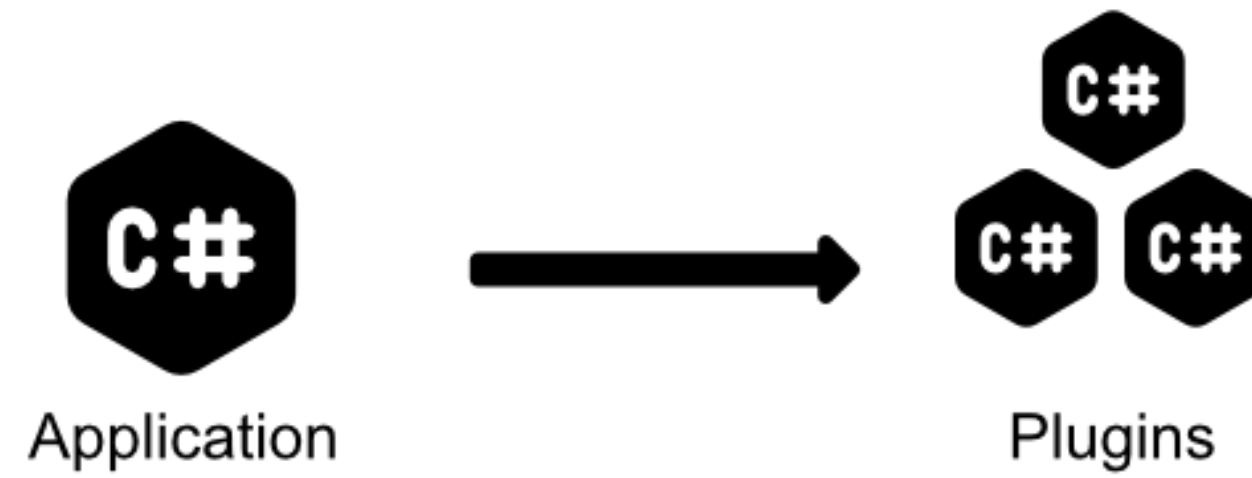


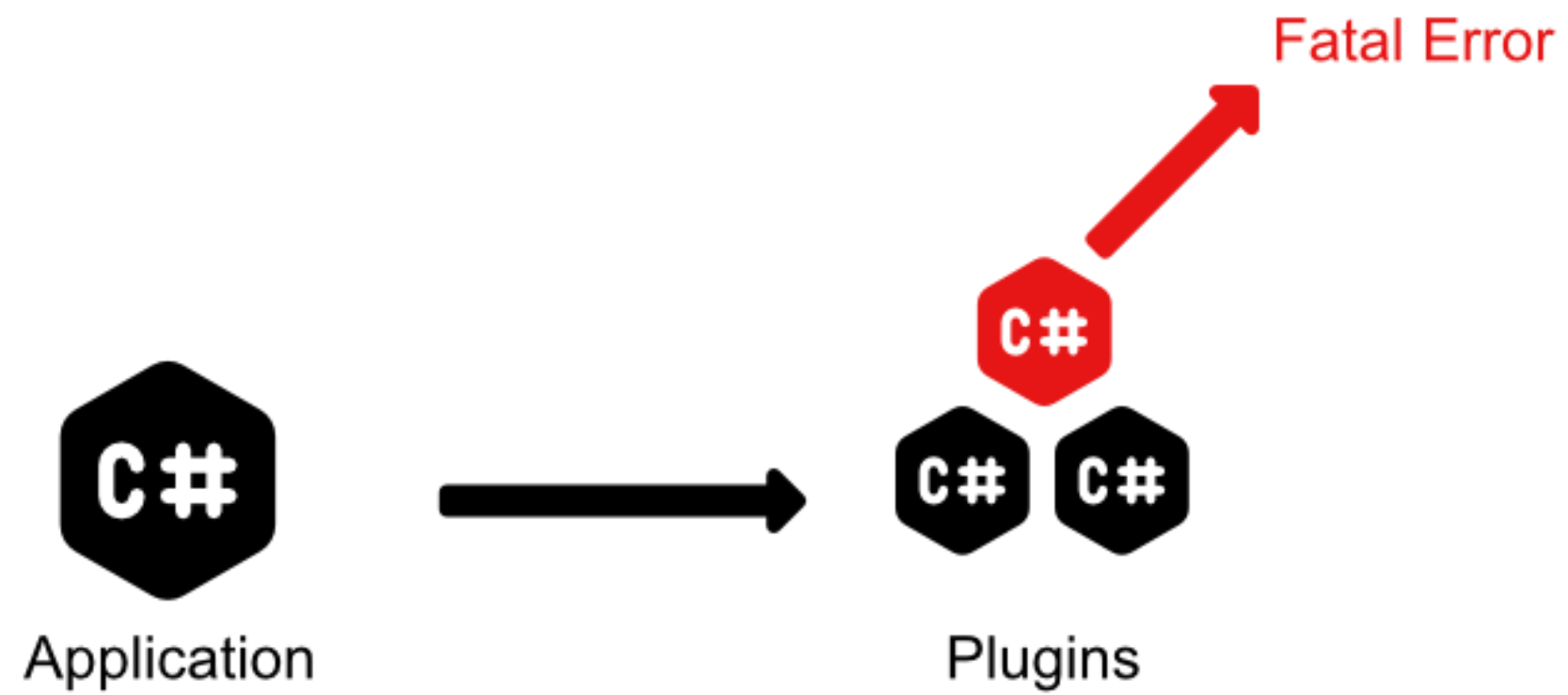
InstrumentStudio Measurement Plug-Ins

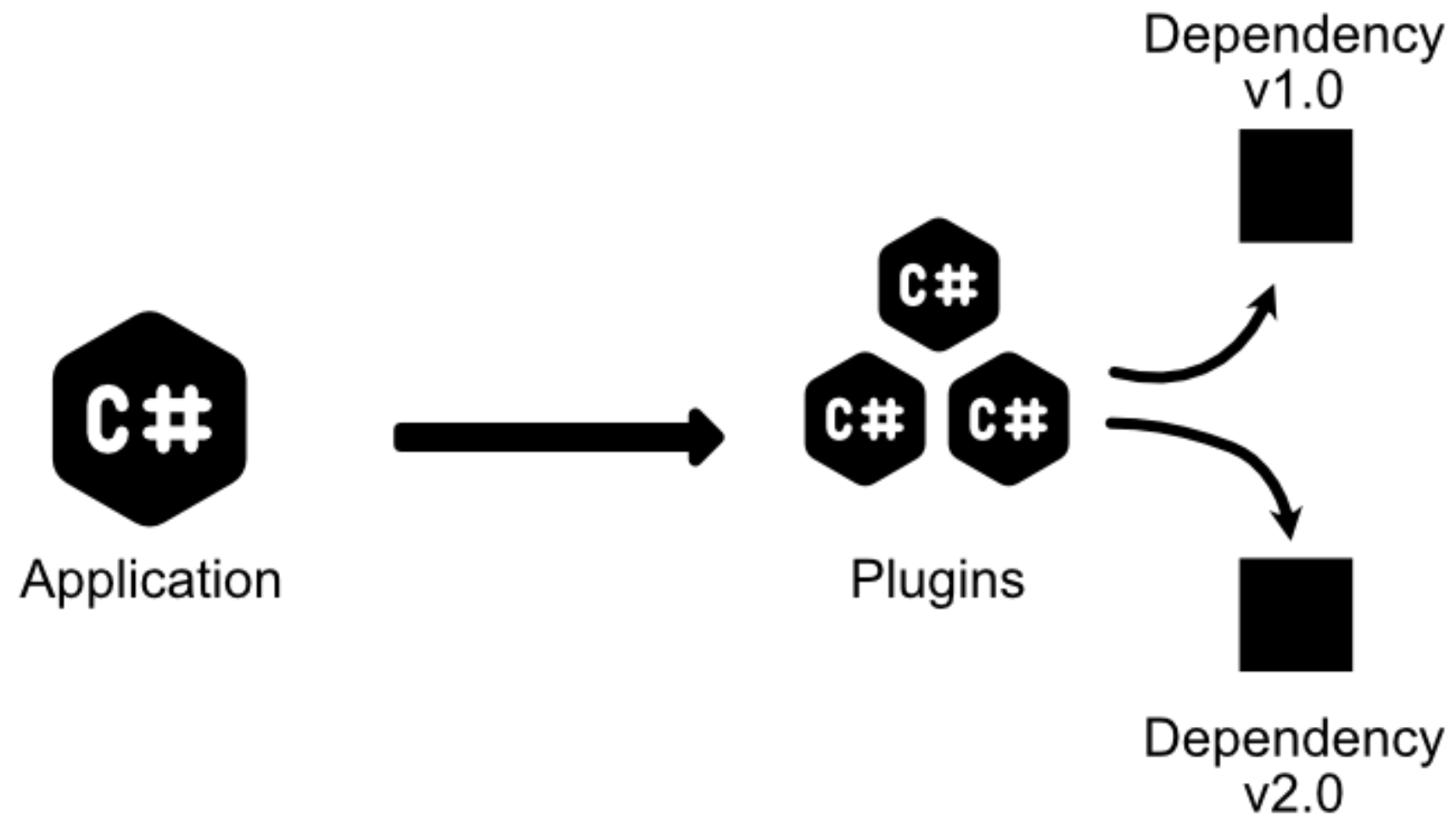


Problems with Traditional Plugin Model

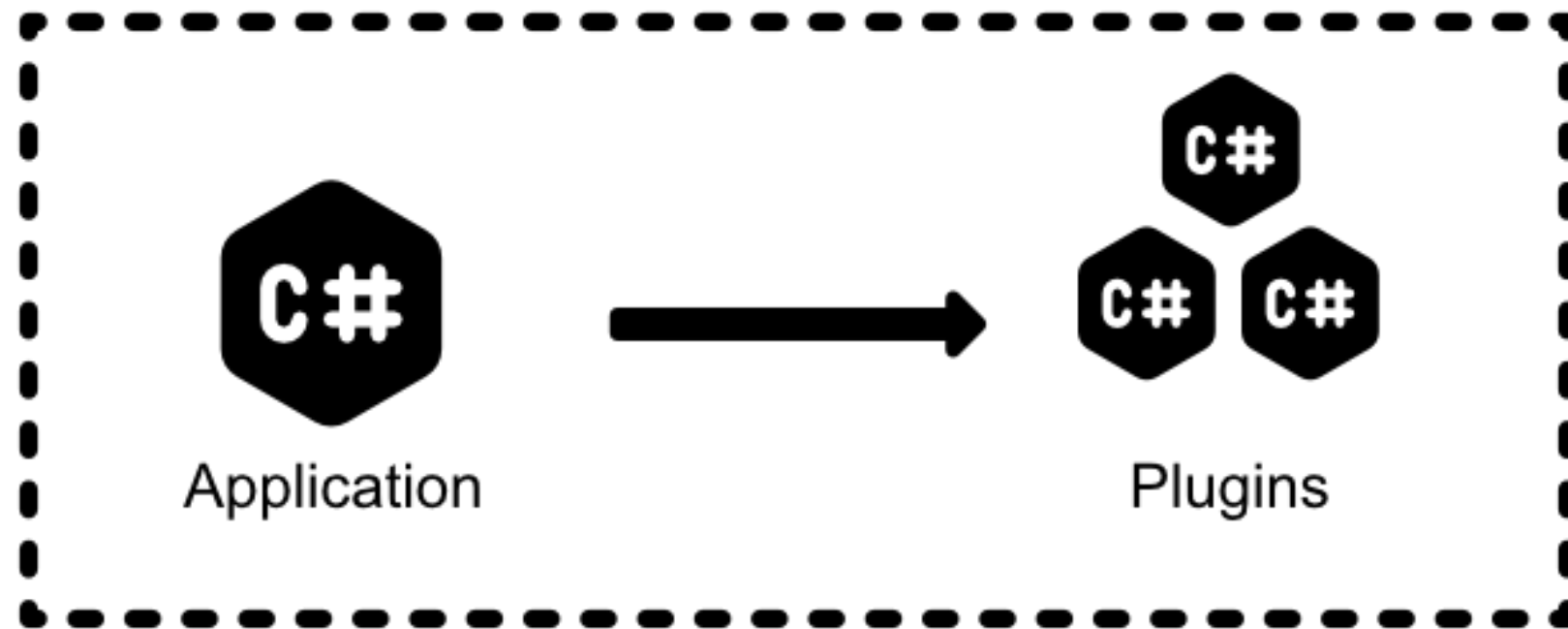
- Language lock-in
- Fatal errors
- Dependency management
- Debugging
- Sharing code

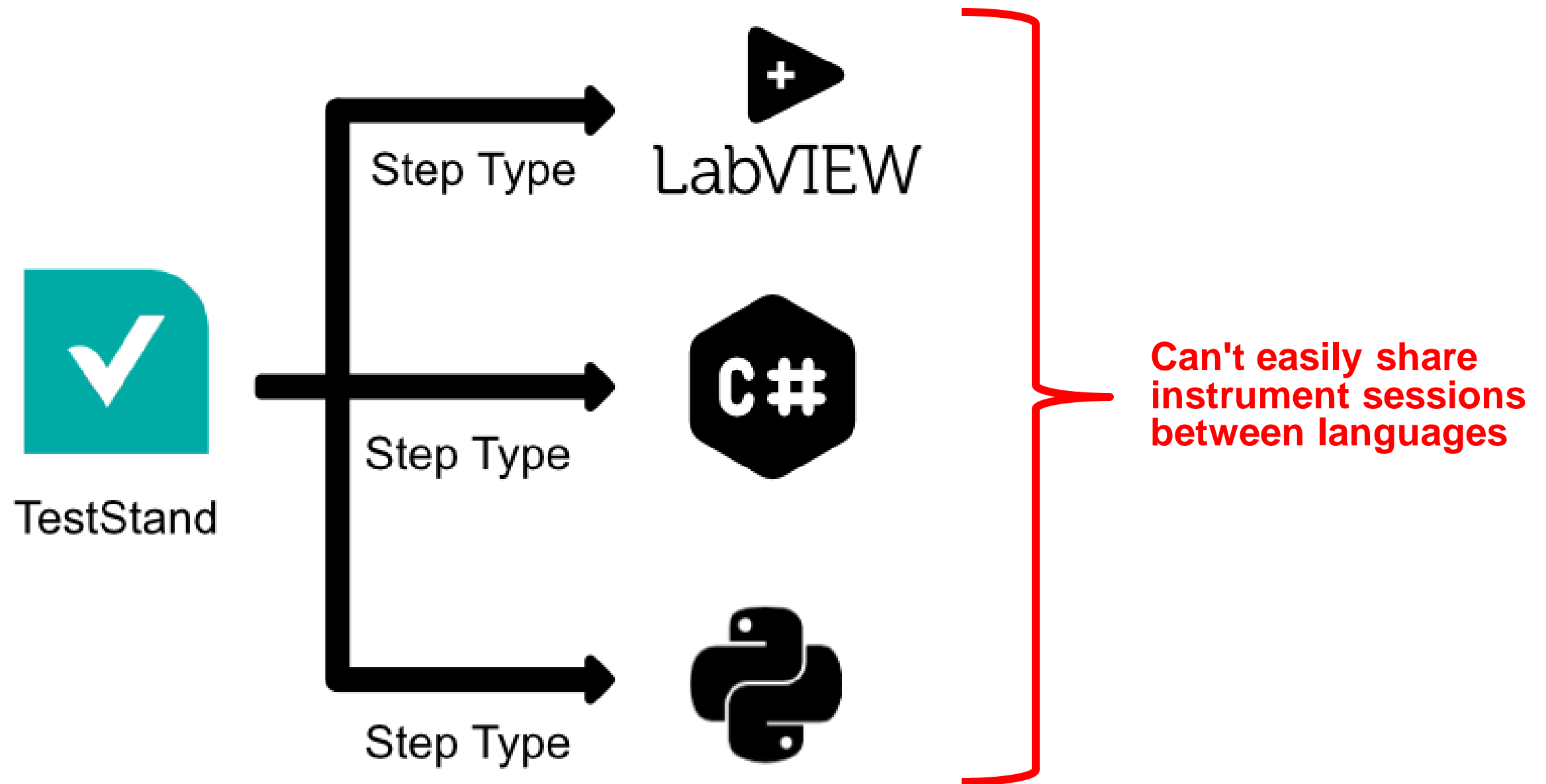


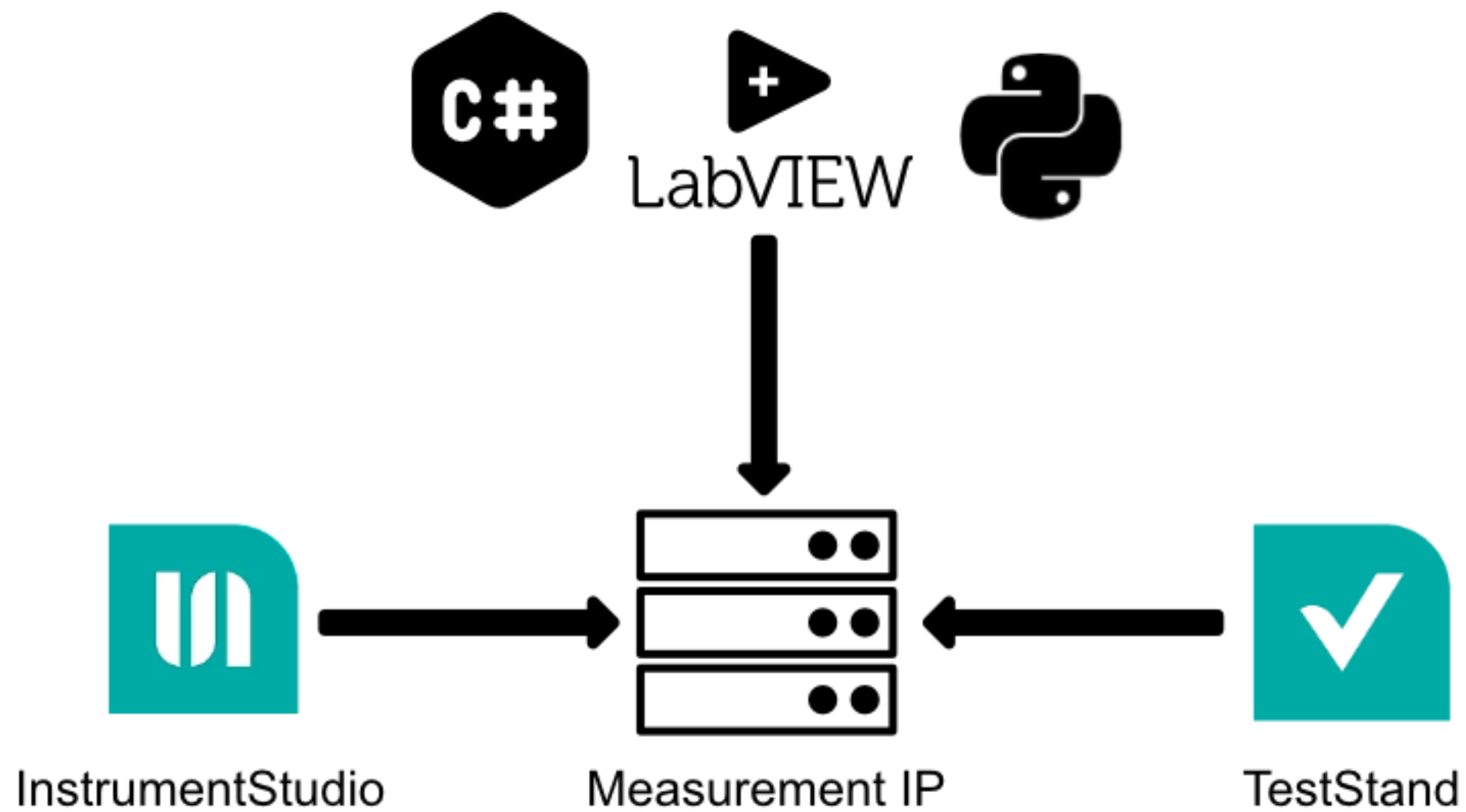




Debugging









FileHomeShareView

Pin to Quick accessCopyPasteCutCopy pathPaste shortcutMove toCopy toDeleteRenameNew folderNew itemEasy accessPropertiesOpenHistoryStart backupBackup

ClipboardOrganizeNewOpenSelectBackup

examplesNI-SCOPE Acquire Waveform

Search NI-SCOPE Acquire Wa...

Name	Date modified	Type	Size
.cache	5/17/2024 2:15 PM	File folder	
NI-SCOPE Acquire Waveform	5/10/2024 6:47 AM	File folder	
NIScopeAcquireWaveform.aliases	5/17/2024 2:14 PM	ALIASES File	1 KB
NIScopeAcquireWaveform	5/17/2024 2:15 PM	InstrumentStudio ...	2 KB
NIScopeAcquireWaveform.lvls	5/10/2024 2:22 PM	LVLPS File	1 KB
NIScopeAcquireWaveform	5/10/2024 2:22 PM	LabVIEW Project	24 KB
NIScopeAcquireWaveform.pinmap	3/26/2024 9:04 AM	PINMAP File	1 KB
NIScopeAcquireWaveform	3/26/2024 9:04 AM	TestStand Sequen...	8 KB
NIScopeAcquireWaveform.sfp	5/17/2024 2:15 PM	SFP File	5 KB
README.md	3/26/2024 9:04 AM	MD File	2 KB

10 items1 item selected23.0 KB

Microservice Challenges

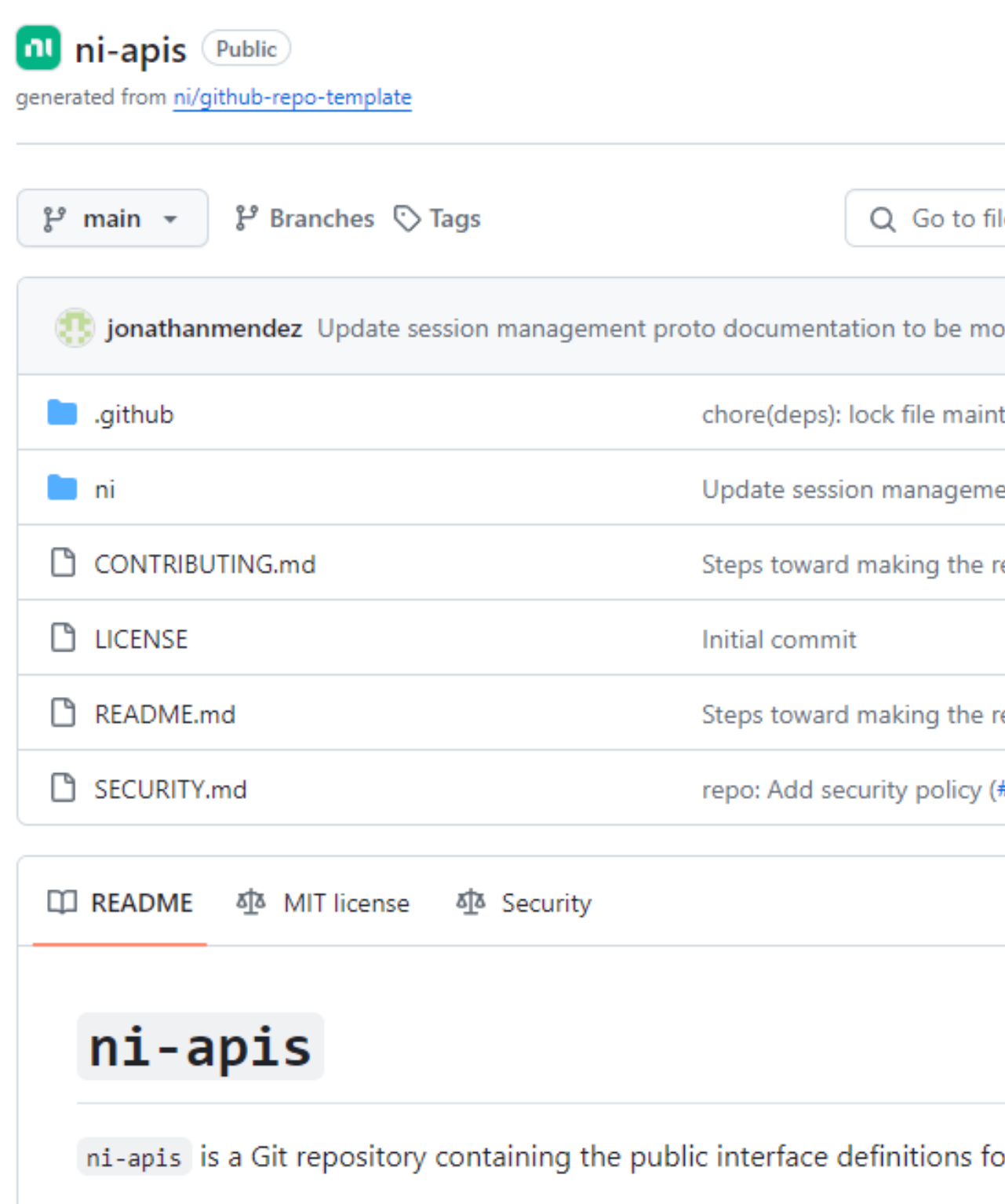
- Managing state between applications
 - Registering and reserving instrument sessions
 - Using pin maps
- Discovery and calling services

Utility Services

- **Discovery Service**
 - Discovery services on the machine
- **Monitoring Service**
 - Monitor measurements between Instrument Studio and Test Stand
- **Pin Map Service**
 - Share pin map between services and applications
- **Session Management Service**
 - Manage hardware between services and applications
- **I/O Discovery Service**
 - Discovery hardware I/O on system.

How to create a measurement

- Create a gRPC service
- Implement the measurement service proto
- Create serviceconfiguration file
- Create user interface
- Register with the DiscoveryService

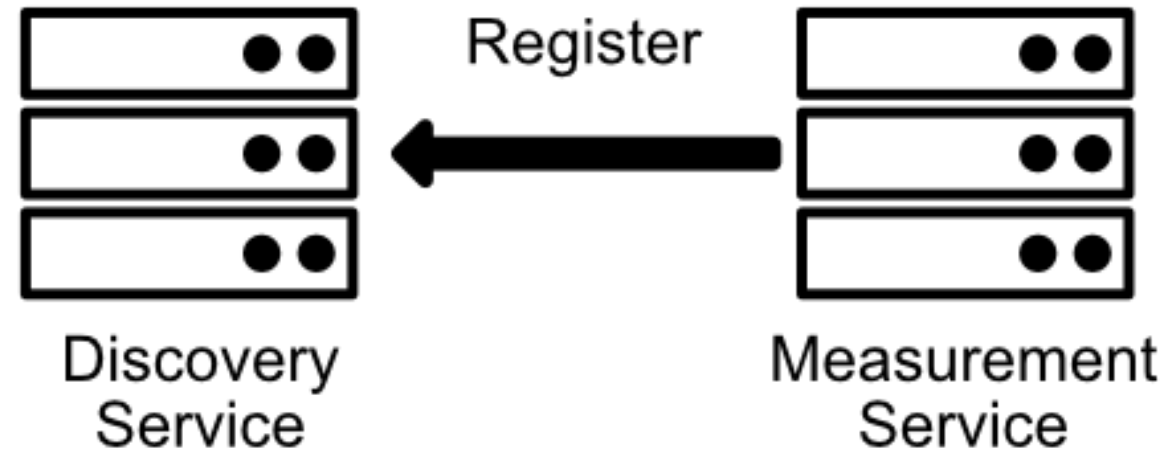


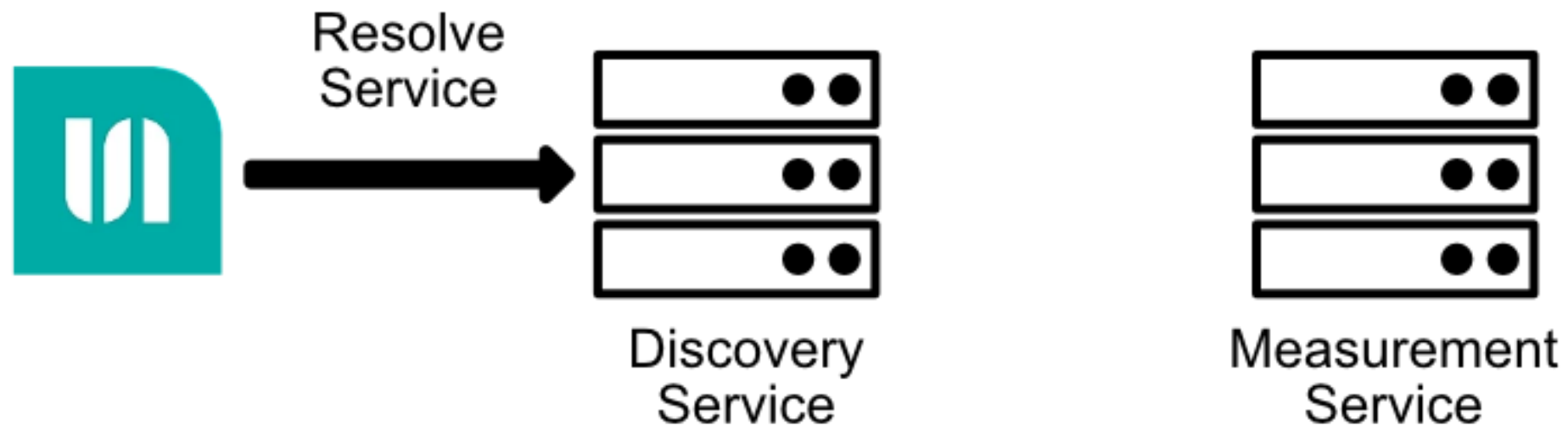
ni-apis

- Collection of gRPC protos
- Open source

Measurement Interface

```
1 // Service that implements a measurement. Unlike other services, a MeasurementService is designed to be a plugin
2 // where there can be multiple implementations of the service that provide different measurement capabilities.
3 service MeasurementService {
4     // Returns information that describes the measurement.
5     rpc GetMetadata (GetMetadataRequest) returns (GetMetadataResponse);
6
7     // API used to perform a measurement.
8     rpc Measure (MeasureRequest) returns (stream MeasureResponse);
9 }
10
```

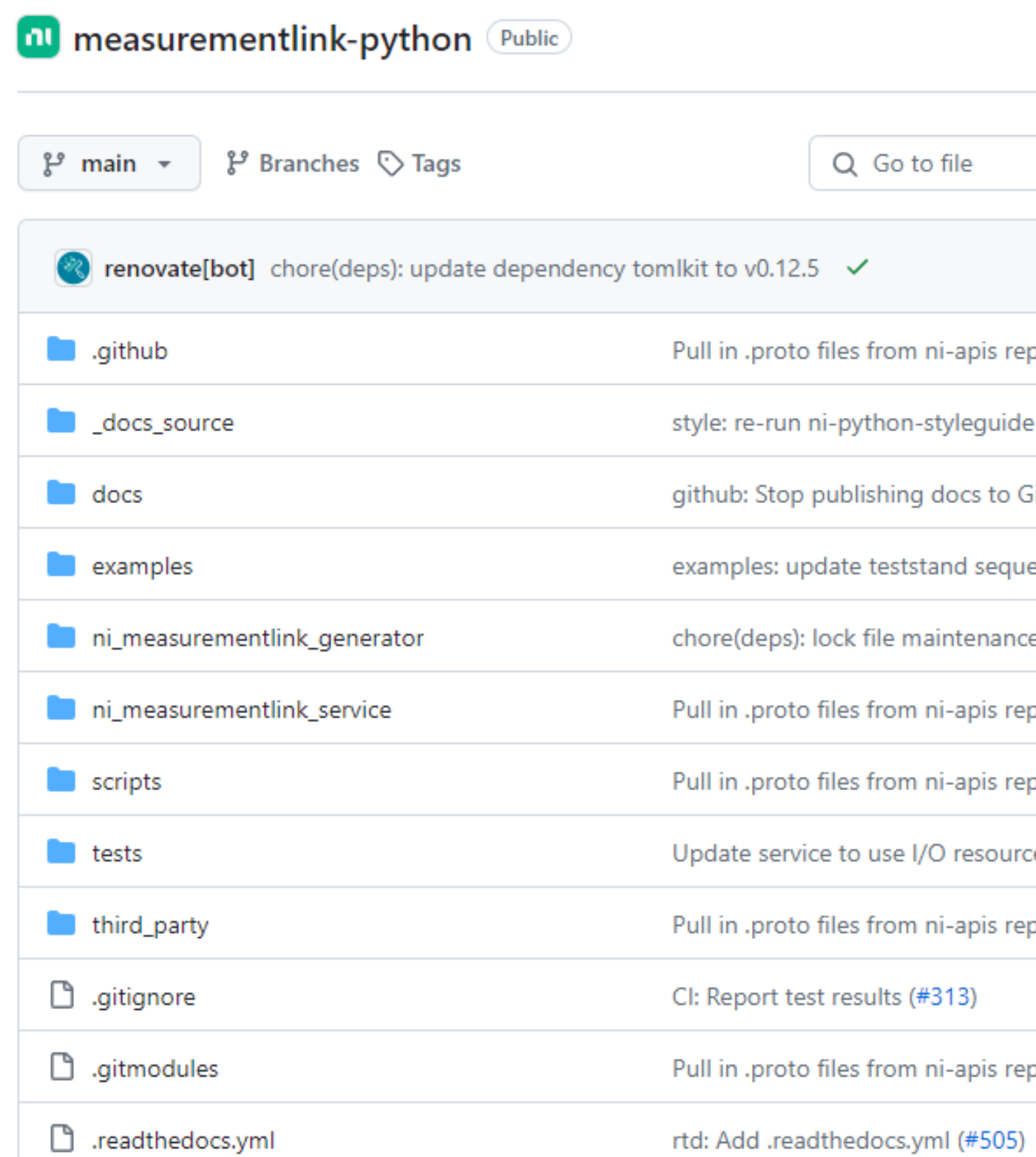






Python and LabVIEW

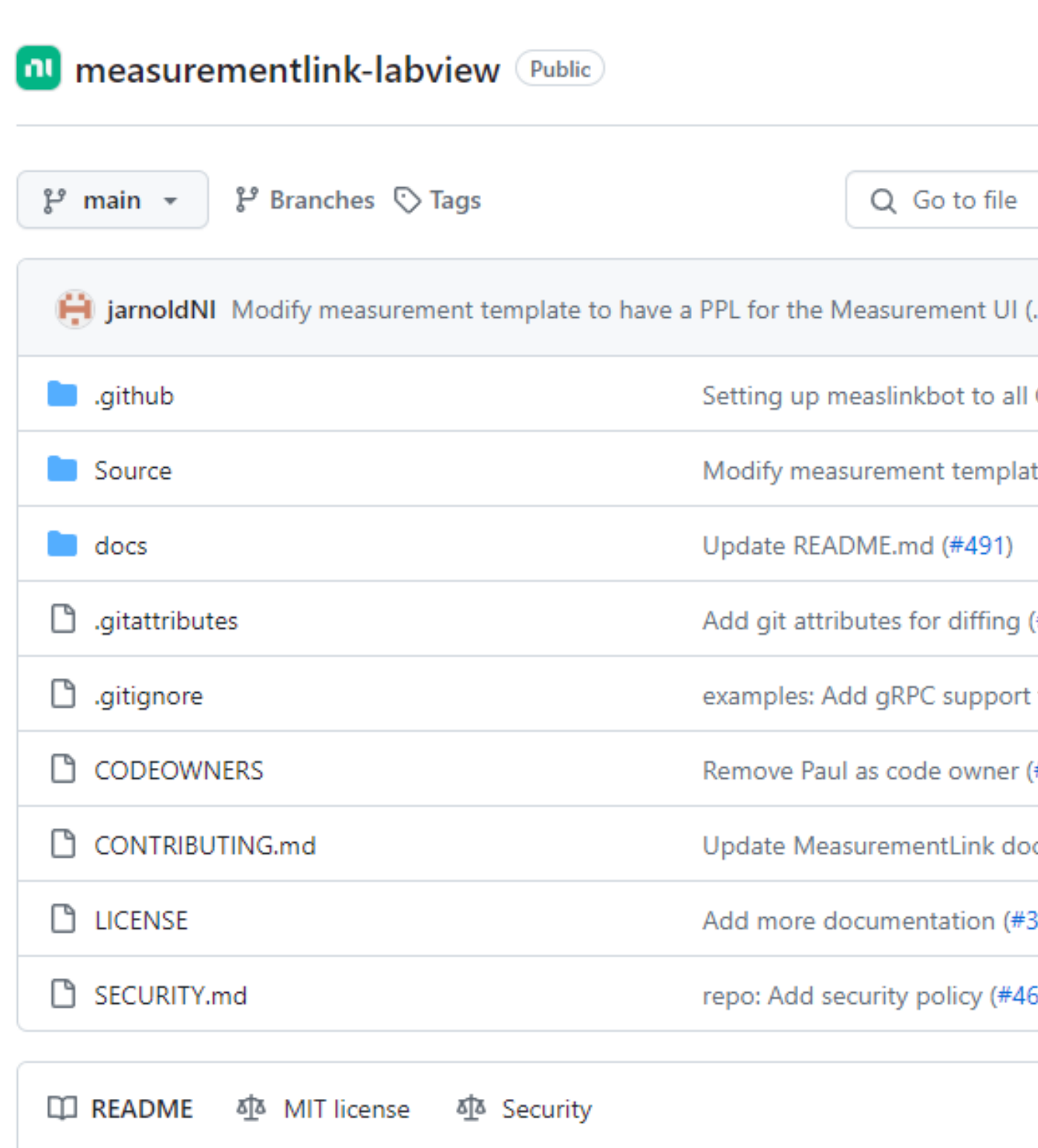
- First class support
- Abstracts gRPC away
- Handles DiscoveryService registration
- Handles channel creation



Python support

- Open source
- MIT Licensed





LabVIEW support

- Open source
- MIT Licensed

Service Configuration

```
1  {
2    "services": [
3      {
4        "displayName": "NI-DMM Measurement (Py)",
5        "serviceClass": "ni.examples.NIDmmMeasurement_Python",
6        "descriptionUrl": "",
7        "providedInterfaces": [
8          "ni.measurementlink.measurement.v1.MeasurementService",
9          "ni.measurementlink.measurement.v2.MeasurementService"
10       ],
11        "path": "start.bat",
12        "annotations": {
13          "ni/service.description": "MeasurementLink example that performs a measurement using an NI DMM.",
14          "ni/service.collection": "NI.Examples",
15          "ni/service.tags": []
16        }
17      }
18    ]
19  }
```

Register Python Measurement

```
1 measurement_service = nims.MeasurementService(  
2     service_config_path=service_directory / "NIDmmMeasurement.serviceconfig",  
3     version="0.1.0.0",  
4     ui_file_paths=[service_directory / "NIDmmMeasurement.measui"],  
5 )
```

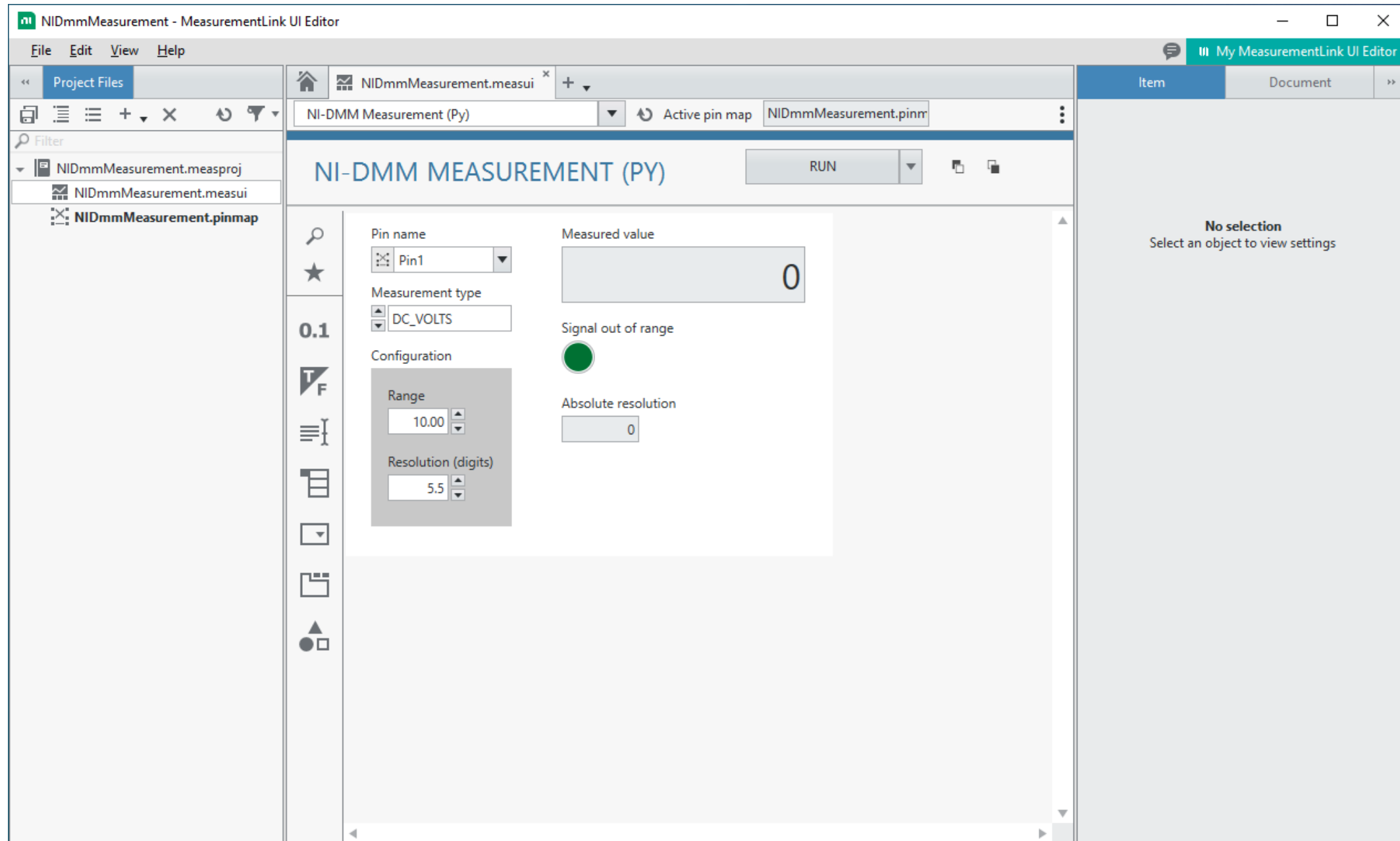
Define Python Measurement

```
1  @measurement_service.configuration(
2      "measurement_type", nims.DataType.Enum, Function.DC_VOLTS, enum_type=Function
3  )
4  @measurement_service.configuration("range", nims.DataType.Double, 10.0)
5  @measurement_service.configuration("resolution_digits", nims.DataType.Double, 5.5)
6  @measurement_service.output("measured_value", nims.DataType.Double)
7  @measurement_service.output("signal_out_of_range", nims.DataType.Boolean)
8  @measurement_service.output("absolute_resolution", nims.DataType.Double)
9  def measure(
10      pin_name: str,
11      measurement_type: Function,
12      range: float,
13      resolution_digits: float,
14  ) -> Tuple[float, bool, float]:
15      """Perform a measurement using an NI DMM."""
16
17      # Measurement Code
18
19      return (measured_value, signal_out_of_range, absolute_resolution)
20
```

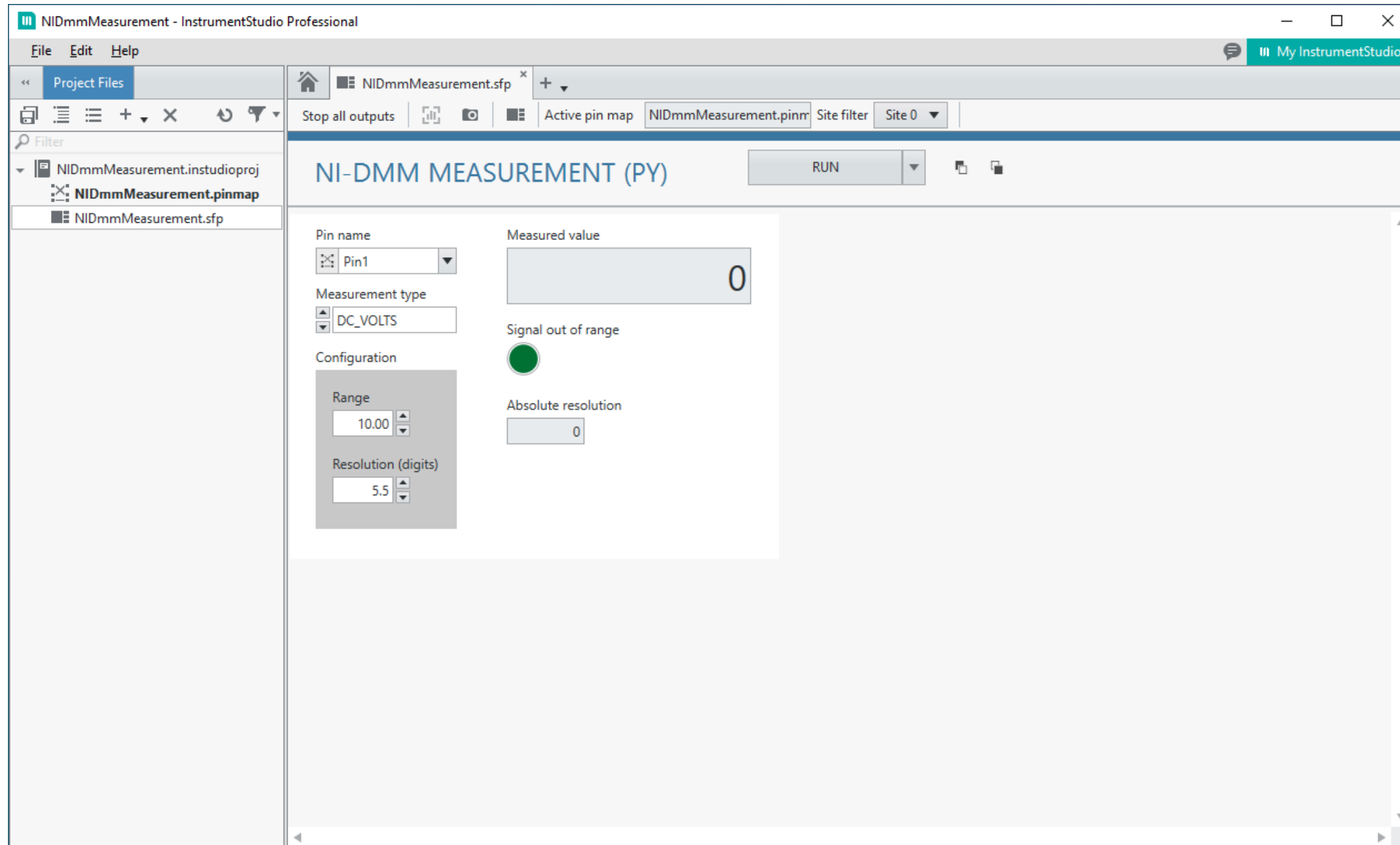
Define Python Measurement

```
1  # If the measurement type is not specified, use DC_VOLTS.
2  nidmm_function = nidmm.Function(measurement_type.value or Function.DC_VOLTS.value)
3
4  with measurement_service.context.reserve_session(pin_name) as reservation:
5      with reservation.initialize_nidmm_session() as session_info:
6          session = session_info.session
7          session.configure_measurement_digits(nidmm_function, range, resolution_digits)
8          measured_value = session.read()
9          signal_out_of_range = math.isnan(measured_value) or math.isinf(measured_value)
10         absolute_resolution = session.resolution_absolute
```

MeasurementUI



InstrumentStudio



TestStand

NI TestStand (64-bit) - Sequence Editor [Edit]

File Edit View Execute Debug Configure Source Control Tools Window Help

Insertion Palette

Step Types

LabVIEW

- Additional Results
- Sequence Call
- Statement
- Property Loader
- Label
- Message Popup
- Call Executable
- Flow Control
- Synchronization
- Database
- Data Streams
- LabVIEW Utility
- LabVIEW NXG Utility
- IO Configuration
- MeasurementLink
 - Measurement
 - Update Pin Map

Templates/IO Configurat...

Templates IO Configurations

- Steps
- Variables
- Sequences
- <Drag Template Here>

Sequence File 1*

Steps: MainSequence

STEP	DESCRIPTION	SETTINGS
+ Setup (0)		
- Main (1)		
A Measurement	Measurement	
<End Group>		
+ Cleanup (0)		

Variables Sequences

Variables

Filter by name

NAME	VALUE
Locals ('MainSequence')	
ResultList	
<Right click to insert Local>	
Parameters ('MainSequence')	
<Right click to insert Parameter>	
FileGlobals ('Sequence File 1')	
MeasurementLink	
<Right click to insert File Global>	

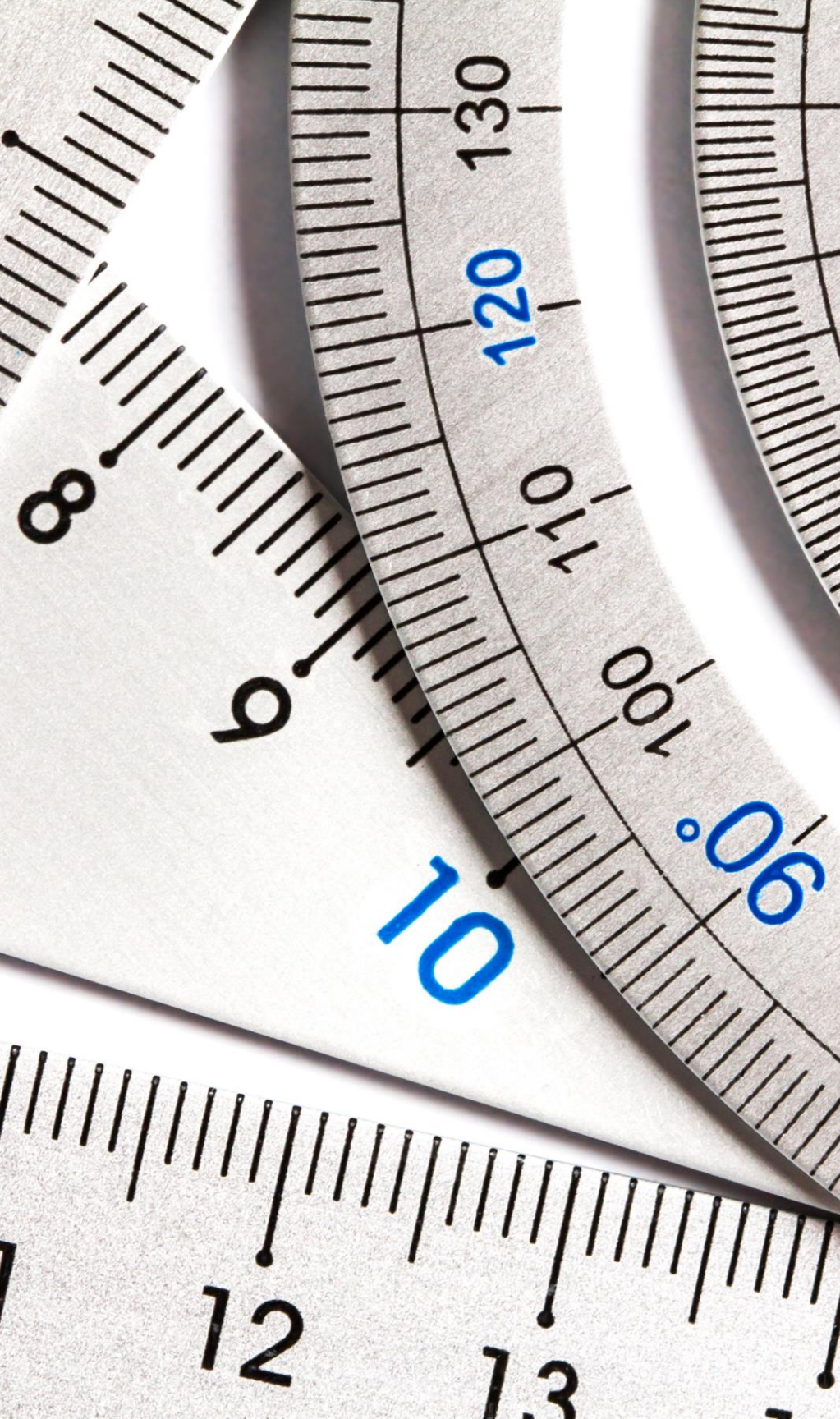
Step Settings for Measurement

Configure Measurement Properties

Measurement NI-DMM Measurement (Py)

PARAMETER NAME	TYPE	IN/OUT	LOG	VALUE	
pin_name	I/O Resource	In	<input checked="" type="checkbox"/>	"Pin1"	fx ✓
measurement_type	Enum	In	<input checked="" type="checkbox"/>	DC_VOLTS	fx ✓
range	Number (Double)	In	<input checked="" type="checkbox"/>	10	fx ✓
resolution_digits	Number (Double)	In	<input checked="" type="checkbox"/>	5.5	fx ✓
measured_value	Number (Double)	Out	<input checked="" type="checkbox"/>		fx ✓
signal_out_of_range	Boolean	Out	<input checked="" type="checkbox"/>		fx ✓
absolute_resolution	Number (Double)	Out	<input checked="" type="checkbox"/>		fx ✓

User: administrator Environment: <Global> Model: SequentialModel.seq 1 Step Selected [0] Number of Steps: 1



InstrumentStudio Measurement Plug-Ins

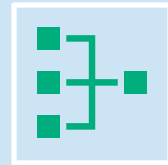
- Built on gRPC
- Easily develop measurements
- Easily reuse measurements between InstrumentStudio and TestStand
- Easily debug measurements

Recommended Presentations



InstrumentStudio Pro: A Low-code Environment for Manual and Automated Measurements

Tuesday, 3:15pm
Room 17 A



What's New in LabVIEW

Wednesday, 10:15am
Room 19A



**Make Measurement Reuse
your New Super Power**

**Wednesday, 2:45pm
Room 19B**

Hands-On

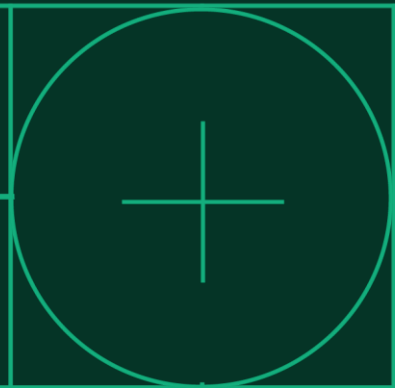
- Workflows from Instrument Bring up to Test Automation
 - Wednesday 1:30pm-3:45pm
 - Ballroom E

Questions?



ni connect

2024 AUSTIN



Certainly! Here are some common misconceptions about gRPC that you might want to address in your presentation:

1. gRPC is only for Google's use: While gRPC was initially developed by Google, it is an open-source framework and is used by many organizations outside of Google¹.

2. gRPC is too complex for simple applications: gRPC is designed for performance and scalability, but it can be used for simple applications as well, offering benefits like strong type checking and code generation¹.

3. gRPC can only be used with Protocol Buffers: gRPC typically uses Protocol Buffers for serialization; however, it can be configured to use other serialization formats like JSON or XML if needed¹.

4. gRPC is not suitable for browser clients: gRPC-Web enables gRPC to be used in browser-based applications, although it does have some limitations compared to gRPC on the server side¹.

5. gRPC is only for microservices architecture: gRPC is often used in microservices due to its efficiency, but it can be used in any client-server architecture where performance is a concern¹.

6. gRPC is not secure: gRPC supports robust security features. It can be used with SSL/TLS for encrypted transport, and it also supports authentication mechanisms¹.

7. gRPC is always better than REST: While gRPC has performance advantages, REST might be a better choice for public APIs due to its simplicity and widespread familiarity among developers².

8. gRPC doesn't support streaming: gRPC actually supports four kinds of RPCs, including server streaming, client streaming, bidirectional streaming, and the traditional unary call¹.

9. gRPC is difficult to debug: With the right tools and practices, gRPC services can be debugged effectively. It's important to use comprehensive logging and monitoring to troubleshoot issues³.

10. gRPC is not performant with large payloads: gRPC is designed for performance, including with large payloads. However, it's crucial to optimize message sizes and use streaming RPCs appropriately to avoid performance pitfalls³.

These misconceptions can lead to misunderstandings about gRPC's capabilities and use cases. Addressing them in your presentation can help clarify what gRPC is and how it can be effectively utilized in various scenarios.

Maybe talk about how chad uses grpc for other customer use cases