NIWeek

FUTURE FASTER

MAY 21–24, 2018, AUSTIN, TEXAS

OO: LLAMA: Leveraging Object-Oriented Programming for Designing a Logging Framework

LLAMA

Jesse Batsche

Project Manager, DMC

NATIONAL INSTRUMENTS LabVIEW™ Certified Architect

Christian Owen

Systems Engineer, DMC

NATIONAL INSTRUMENTS LabVIEW™ Certified Developer

DMC Overview

Established in 1996, DMC serves customers worldwide from offices in Chicago, Boston, Denver, Houston, New York, Reno, Seattle, and St. Louis.

170+

employees & growing

Industries Served

- Automotive
- Chemical
- Consumer Goods
- Defense Contracting
- Energy & Utilities
- Food & Beverage
- Oil & Gas Engineering
- Packaging Machinery Programming
- Pharmaceutical
- Printing
- Semiconductor
- Specialty Machinery
- Telecommunications
- Test & Measurement

Our NI Certifications

NATIONAL INSTRUMENTS™
Alliance Member

DMC has partnered with **National Instruments since 1997**

NATIONAL INSTRUMENTS
Silver Alliance Partner
Vision Specialty

DMC's team of certified LabVIEW engineers
is **one of the largest in the country**

NATIONAL INSTRUMENTS LabVIEW™
Certified Architect

12 Certifications

NATIONAL INSTRUMENTS LabVIEW™
Certified Developer

4 Certifications

NATIONAL INSTRUMENTS™
Certified Professional Instructor

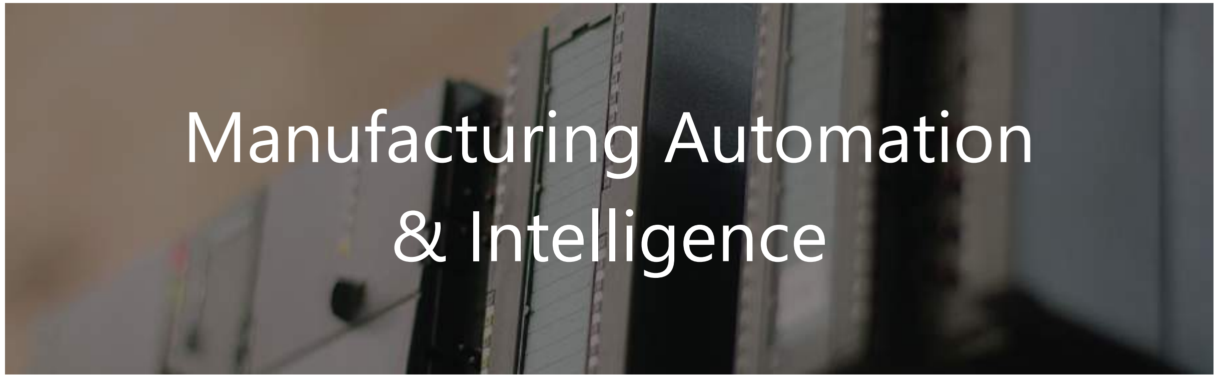
4 Certifications

NATIONAL INSTRUMENTS LabVIEW
CERTIFIED EMBEDDED SYSTEMS DEVELOPER

1 Certification

NATIONAL INSTRUMENTS™

Areas of Expertise



Manufacturing Automation & Intelligence

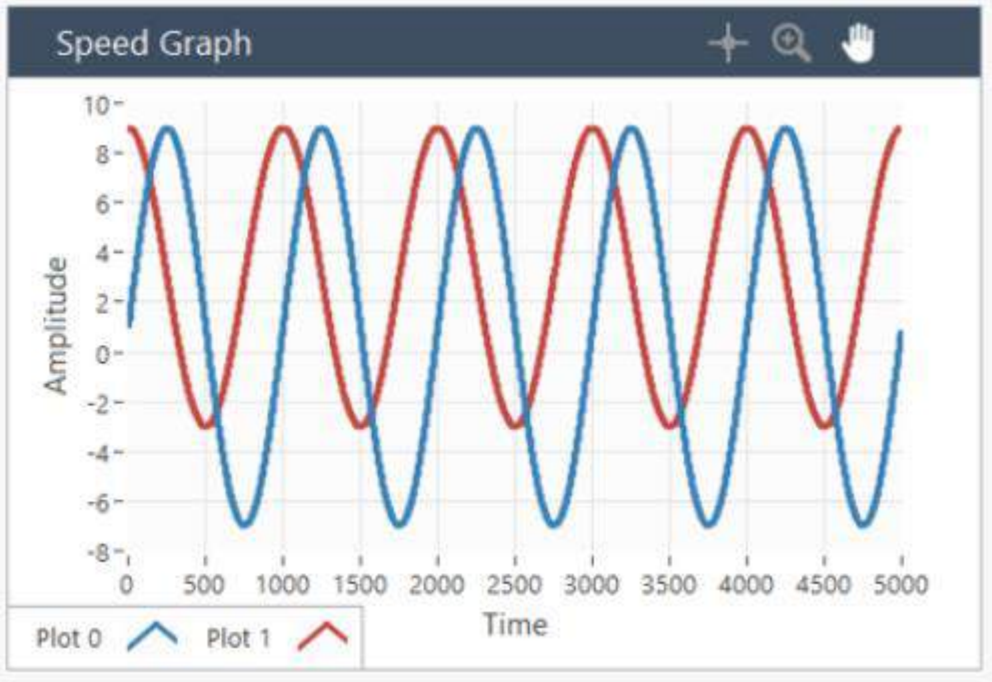
Test & Measurement Automation

Custom Software & Hardware Development

Microsoft Consulting Services

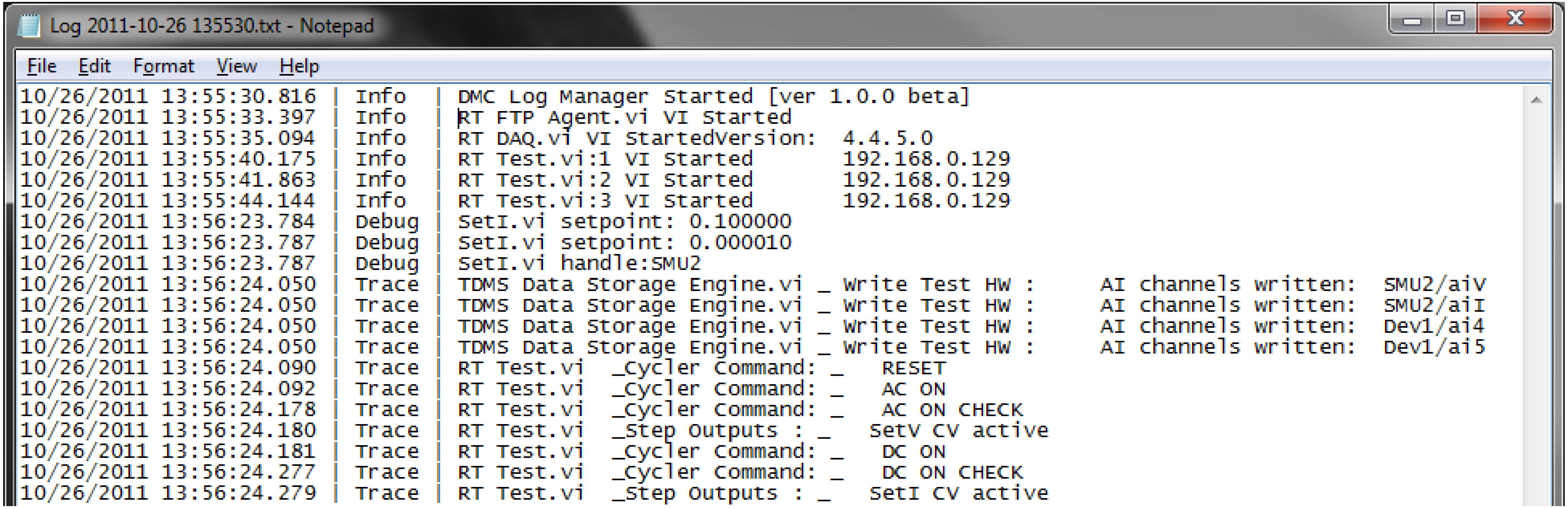
Today we are NOT talking about…

Logging?



Today we ARE talking about:

Event Logging. Execution Trace Logging.



```
10/26/2011 13:55:30.816 | Info  | DMC Log Manager Started [ver 1.0.0 beta]
10/26/2011 13:55:33.397 | Info  | RT FTP Agent.vi VI Started
10/26/2011 13:55:35.094 | Info  | RT DAQ.vi VI StartedVersion:  4.4.5.0
10/26/2011 13:55:40.175 | Info  | RT Test.vi:1 VI Started       192.168.0.129
10/26/2011 13:55:41.863 | Info  | RT Test.vi:2 VI Started       192.168.0.129
10/26/2011 13:55:44.144 | Info  | RT Test.vi:3 VI Started       192.168.0.129
10/26/2011 13:56:23.784 | Debug | SetI.vi setpoint: 0.100000
10/26/2011 13:56:23.787 | Debug | SetI.vi setpoint: 0.000010
10/26/2011 13:56:23.787 | Debug | SetI.vi handle:SMU2
10/26/2011 13:56:24.050 | Trace | TDMS Data Storage Engine.vi _ Write Test HW :     AI channels written:  SMU2/aiV
10/26/2011 13:56:24.050 | Trace | TDMS Data Storage Engine.vi _ Write Test HW :     AI channels written:  SMU2/aiI
10/26/2011 13:56:24.050 | Trace | TDMS Data Storage Engine.vi _ Write Test HW :     AI channels written:  Dev1/ai4
10/26/2011 13:56:24.050 | Trace | TDMS Data Storage Engine.vi _ Write Test HW :     AI channels written:  Dev1/ai5
10/26/2011 13:56:24.090 | Trace | RT Test.vi  _Cycler Command: _   RESET
10/26/2011 13:56:24.092 | Trace | RT Test.vi  _Cycler Command: _   AC ON
10/26/2011 13:56:24.178 | Trace | RT Test.vi  _Cycler Command: _   AC ON CHECK
10/26/2011 13:56:24.180 | Trace | RT Test.vi  _Step Outputs : _   SetV CV active
10/26/2011 13:56:24.181 | Trace | RT Test.vi  _Cycler Command: _   DC ON
10/26/2011 13:56:24.277 | Trace | RT Test.vi  _Cycler Command: _   DC ON CHECK
10/26/2011 13:56:24.279 | Trace | RT Test.vi  _Step Outputs : _   SetI CV active
```

Session Outline

- Why do we need a logging tool?
- What makes a good logging tool?
- Review community and industry options
- How could such a tool be programmed?
- Where do we go next?

Why Do We Need A Logging Tool?

- Essential to debugging and monitoring deployed applications (no development environment)
    - Case study : Remote debug of Transmission EOL Tester in factory
- Bridging the gap : User stories <-> Log files <-> Code execution

Why Do We Need A Logging Tool?

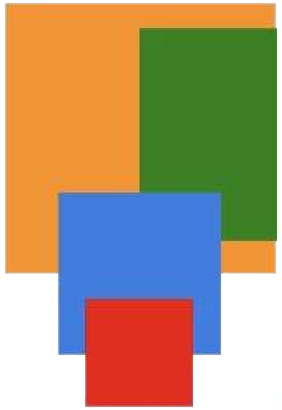
- Supplements and fills gaps in built in LabVIEW debug tools
- Debugging parallel operations
  - Highlight execution and breakpoints will change execution speed
- Probes aren’t convenient for iterative calls
- Event logging is always watching

What Makes A Good Logging Tool?

- Easy to use
  - Simple API
  - Low barrier to entry, easy to integrate into applications, small footprint
- Lightweight
  - Minimal impact on application performance
- Configurable
  - Configurable at run time and after building
- Easy to Extend
  - Easy to “plug in” new features

What Makes A Good Logging Tool?

- Organized Output
  - Categorization or grouping of Entries
    - Timestamped
    - Significance or Subsystem
    - Easy to read, parse, search
- Widely Accessible
  - PC or RT
  - Development or Runtime
  - Cross target



What Makes A Good Logging Tool?

- Provides LIVE and HISTORICAL feedback

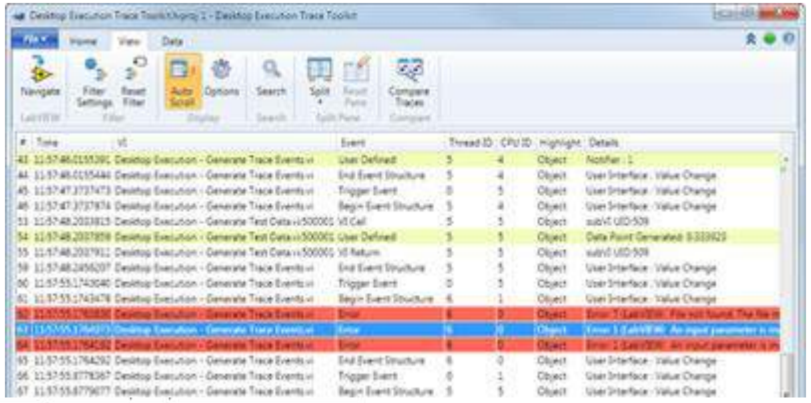
Your Thoughts?

- What are other characteristics that a good logging tool should have?

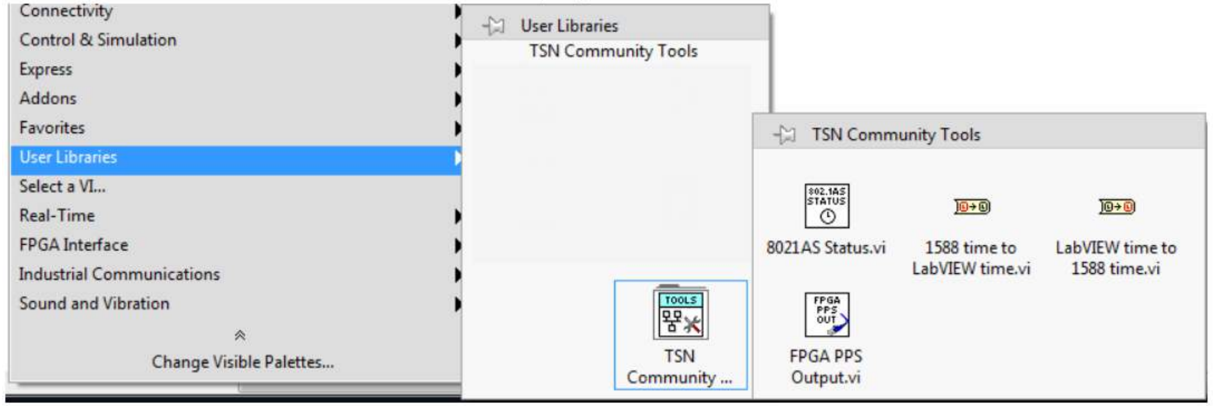
What’s Available?

- NLog (for .NET)
  - Supports multiple targets
  - Structured logging and categorization by severity
- NI Desktop Execution Trace toolkit
  - Excellent for profiling memory and performance
  - Runs outside of LabVIEW (better for diagnosing crashes)
- LabVIEW Community Tools
  - Logger by Field R&D Services, LLC
  - LabQT LGPL Open Source Library Logger
  - …

NLog Advanced .NET Logging



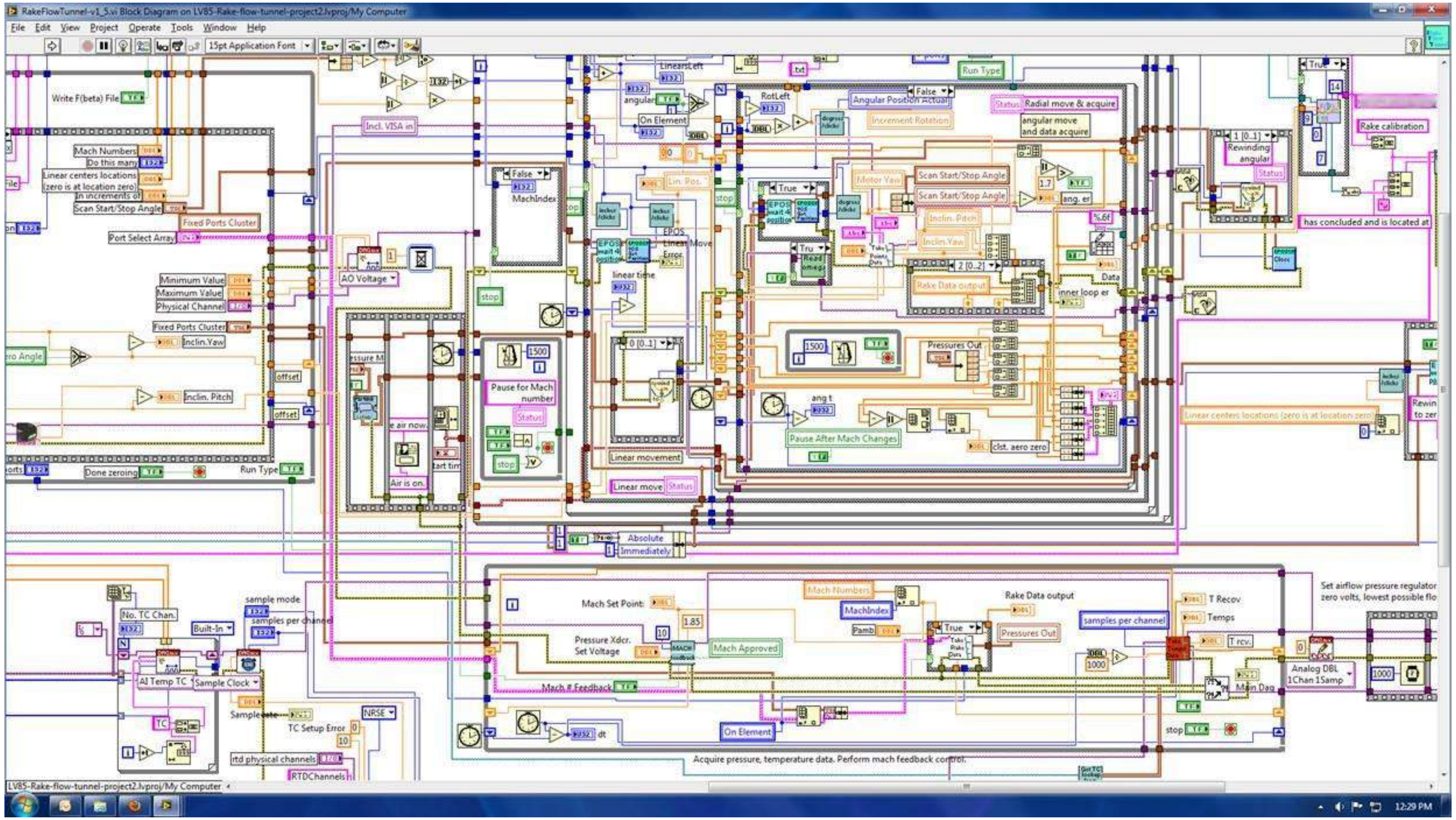
#	Time	VI	Event	Thread ID	CPU ID	Highlight	Details
43	11:57:46.0155391	Desktop Execution - Generate Trace Events.vi	User Defined	5	4	Object	Notifier : 1
44	11:57:46.0155444	Desktop Execution - Generate Trace Events.vi	End Event Structure	5	4	Object	User Interface : Value Change
45	11:57:47.3737473	Desktop Execution - Generate Trace Events.vi	Trigger Event	0	5	Object	User Interface : Value Change
46	11:57:47.3737874	Desktop Execution - Generate Trace Events.vi	Begin Event Structure	5	4	Object	User Interface : Value Change
53	11:57:48.2033815	Desktop Execution - Generate Test Data.vi:500001	VI Call	5	5	Object	subVI UID:509
54	11:57:48.2037859	Desktop Execution - Generate Test Data.vi:500001	User Defined	5	5	Object	Data Point Generated: 0.333923
55	11:57:48.2037911	Desktop Execution - Generate Test Data.vi:500001	VI Return	5	5	Object	subVI UID:509
59	11:57:48.2456207	Desktop Execution - Generate Trace Events.vi	End Event Structure	5	5	Object	User Interface : Value Change
60	11:57:55.1743040	Desktop Execution - Generate Trace Events.vi	Trigger Event	0	5	Object	User Interface : Value Change
61	11:57:55.1743478	Desktop Execution - Generate Trace Events.vi	Begin Event Structure	6	1	Object	User Interface : Value Change
62	11:57:55.1763836	Desktop Execution - Generate Trace Events.vi	Error	6	0	Object	Error 7 (LabVIEW): File not found. The file m
63	11:57:55.1764373	Desktop Execution - Generate Trace Events.vi	Error	6	0	Object	Error 1 (LabVIEW): An input parameter is in
64	11:57:55.1764382	Desktop Execution - Generate Trace Events.vi	Error	6	0	Object	Error 1 (LabVIEW): An input parameter is in
65	11:57:55.1764292	Desktop Execution - Generate Trace Events.vi	End Event Structure	6	0	Object	User Interface : Value Change
66	11:57:55.8779367	Desktop Execution - Generate Trace Events.vi	Trigger Event	0	1	Object	User Interface : Value Change
67	11:57:55.8779677	Desktop Execution - Generate Trace Events.vi	Begin Event Structure	5	5	Object	User Interface : Value Change



NATIONAL INSTRUMENTS

How could such a tool be programmed?

- A practical discussion of LabVIEW programming techniques



NATIONAL INSTRUMENTS™

LLAMA

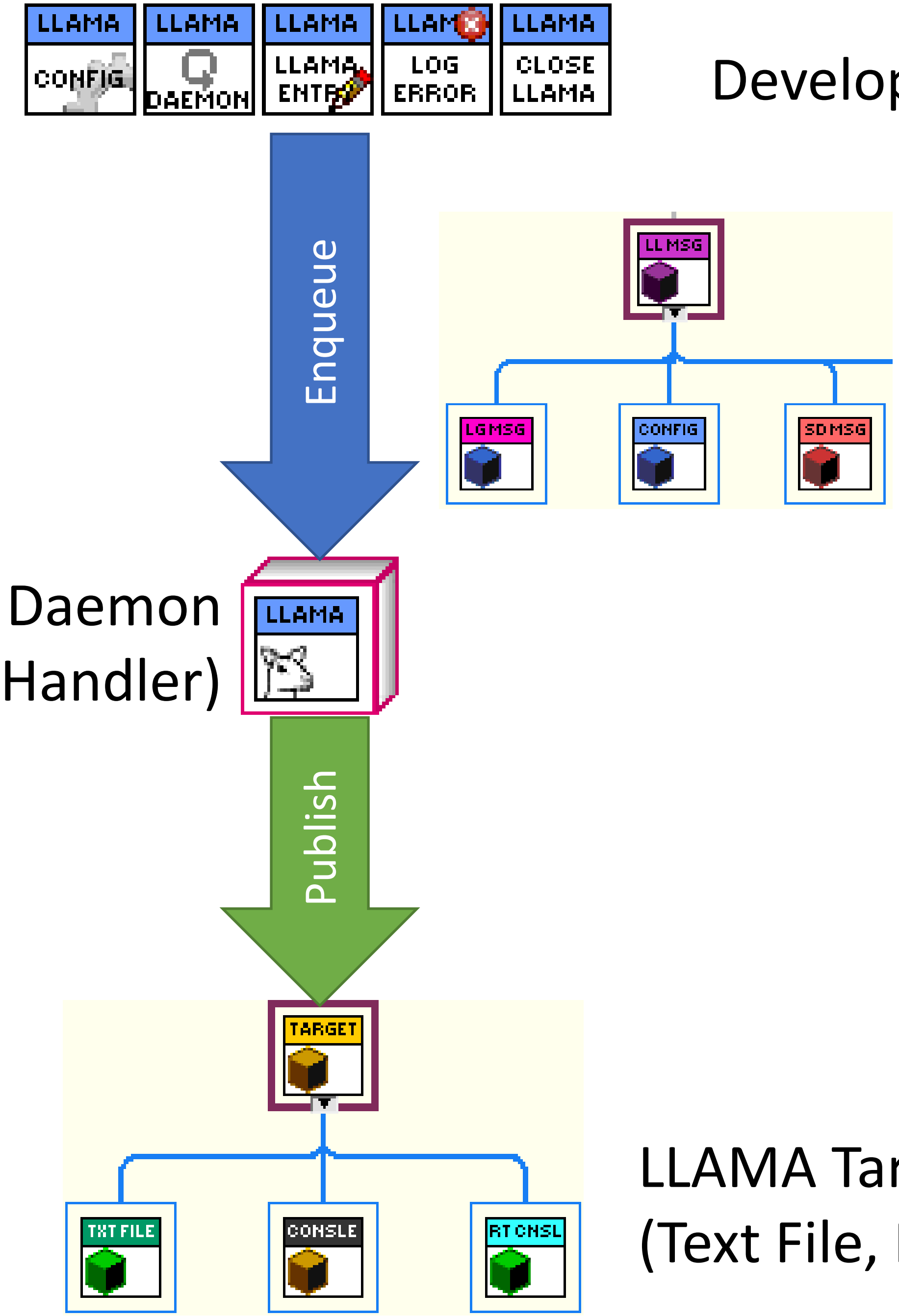
- LLAMA Logs AlMost Anything

_(ツ)_/¯

- Design Intent
  - Pursue the ideals discussed earlier
  - Keep a simple task simple

- DMC is sharing LLAMA (full source code) for you to download

DOWNLOAD



Developer API

LLAMA Message.lvclass
(Log Entry, Config, Shutdown, …)

LLAMA Daemon
(Queued Message Handler)

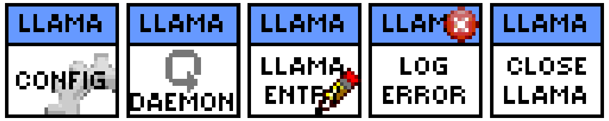
LLAMA Target.lvclass
(Text File, PC Console, RT Console, …)

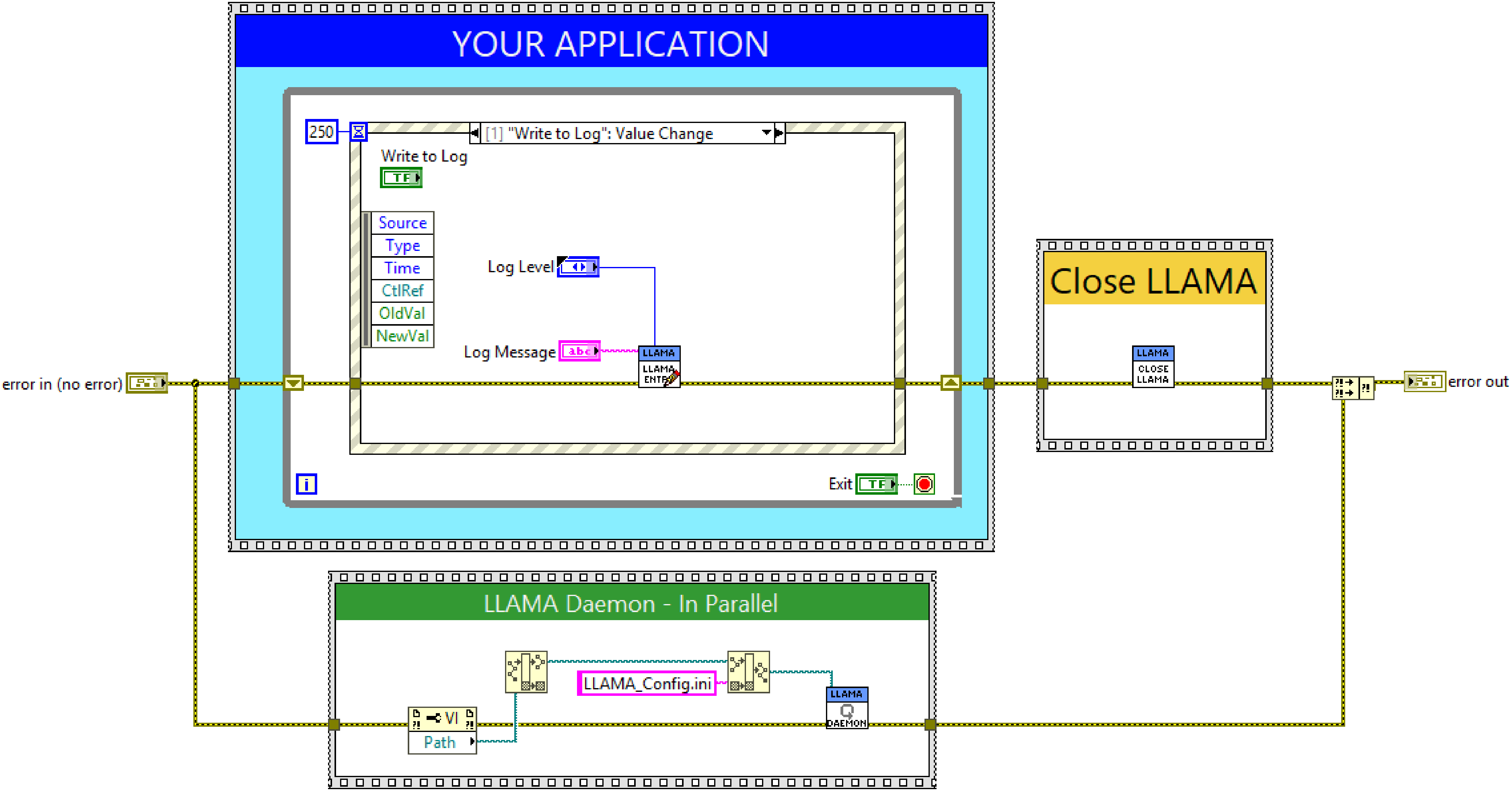
Live / Historical Feedback

- Console Log target provides live feedback
- File targets can be used to capture historical information

Easy to Use

- Straightforward API
- Small code footprint, portable
- Easy to integrate into an application
  - Drop Daemon
  - Drop API calls




YOUR APPLICATION
250
[1] "Write to Log": Value Change
Write to Log
TF
Source
Type
Time
CtlRef
OldVal
NewVal
Log Level
Log Message
abc
LLAMA
LLAMA ENTRY
error in (no error)
i
Exit
TF
Close LLAMA
LLAMA
CLOSE LLAMA
error out
LLAMA Daemon - In Parallel
VI
Path
LLAMA_Config.ini
LLAMA
DAEMON

LLAMA: Overview

- The LLAMA daemon
  - Launched at startup, runs in background
  - Consumes API messages
  - Hangs on to an array of Target objects and a queue reference
- The LLAMA Messaging VIs
  - Enqueues Message objects, which are consumed by the daemon
  - Configure, Log Entry, Shutdown
  - Low overhead

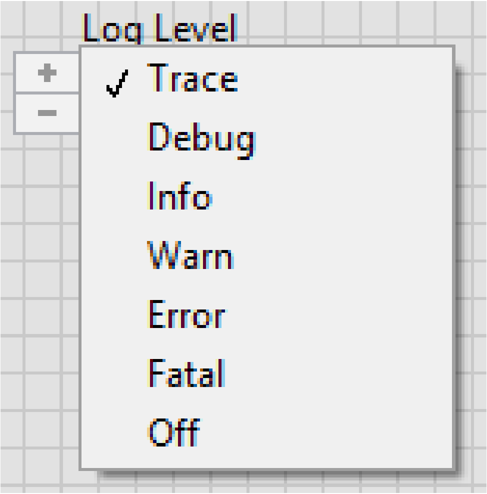
Lightweight

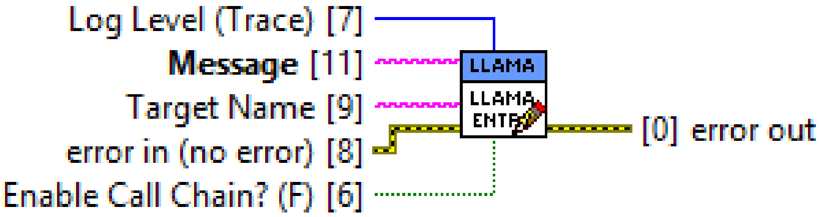
- Heavy lifting occurs in daemon
  - API calls are merely enqueueing a message object
  - Daemon handles messages as time allows

- A quick benchmark : Log string to Text File
- 2 steps
  - Generate string to log : enqueue
    - <2% of time (3.3 us)
  - Compose full log string, write to text file
    - 98% of time (185 us)

Organized Output

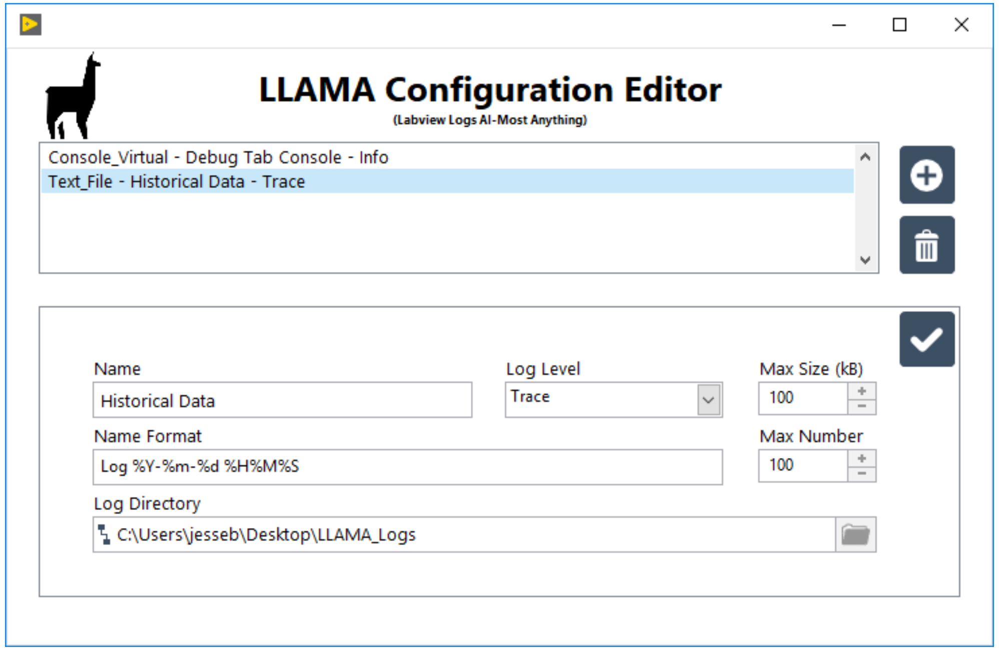
- Categorization in terms of Severities
  - Trace Debug Info Warn Error Fatal
- Categorization by functional area
- Headers allow for simple sorting of entries
  - Timestamp and Log Level included here
- Simplest Case : Text File
  - Flexible, easily searched/parsed
- Extension Case : Structured (e.g. cluster)
  - Avoid string parsing, more compact representation





Configurable at Runtime/Post-Build

- LLAMA can be configured to write to a number of log targets
- Log targets can be added or removed without restarting application
- Configuration can be updated after building



LLAMA Configuration Editor
(Labview Logs Al-Most Anything)

Console_Virtual - Debug Tab Console - Info
Text_File - Historical Data - Trace

Name: Historical Data
Log Level: Trace
Max Size (kB): 100
Name Format: Log %Y-%m-%d %H%M%S
Max Number: 100
Log Directory: C:\Users\jesseb\Desktop\LLAMA_Logs

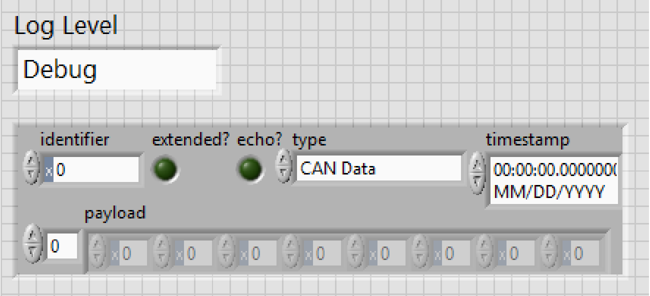
Categorized Output, Configurable Tool

Widely Accessible

- Compatible with RT targets and PC targets
- Can run in development environment and executables

- Limitations
    - No native support for log consolidation between targets

Extensible

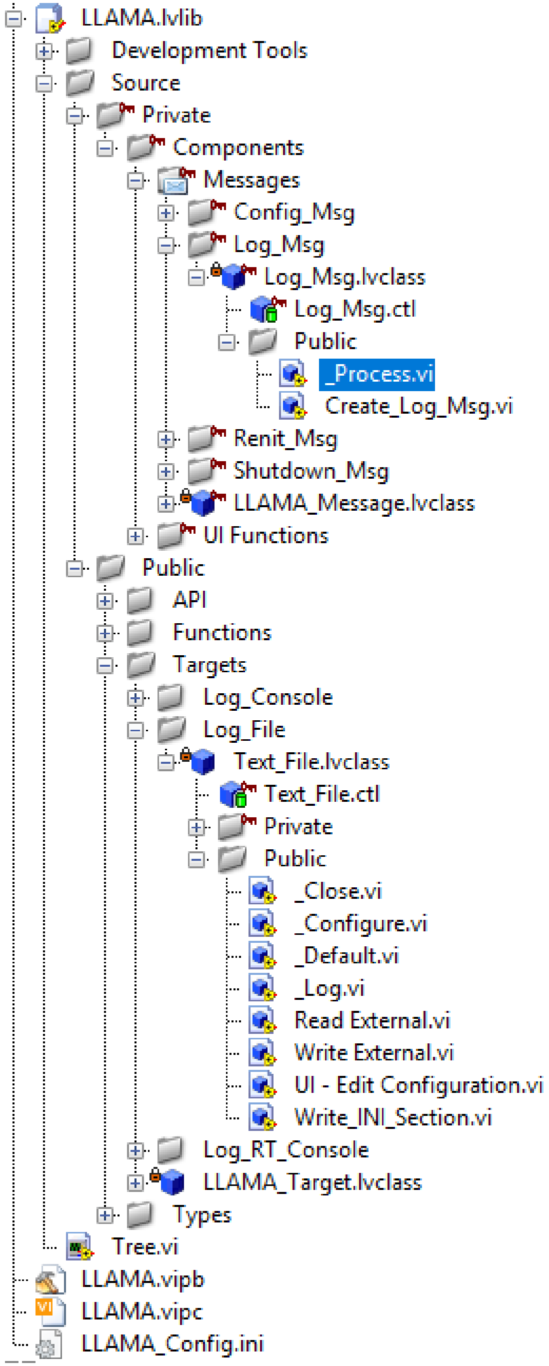


Log Level: Debug

identifier: 0 | extended? | echo? | type: CAN Data | timestamp: 00:00:00.000000 MM/DD/YYYY

payload: 0 0 0 0 0 0 0 0 0

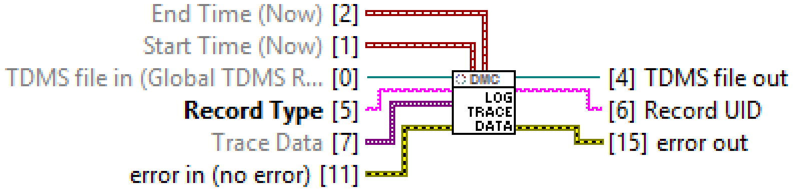
- Example: Instead of string to a text file… We want CAN Frame Message to a Database Target
- Creating a Message plug-in
  - Create a new child of Message and override “Process” method
  - Add a new VI to the API which enqueues the Message object
- Creating a Target plug-in
  - Create a new child of Target and override “Log” and “Configure” methods



```
LLAMA.lvlib
  Development Tools
  Source
    Private
      Components
        Messages
          Config_Msg
          Log_Msg
            Log_Msg.lvclass
              Log_Msg.ctl
              Public
                _Process.vi
                Create_Log_Msg.vi
          Renit_Msg
          Shutdown_Msg
          LLAMA_Message.lvclass
        UI Functions
    Public
      API
      Functions
      Targets
        Log_Console
        Log_File
          Text_File.lvclass
            Text_File.ctl
            Private
            Public
              _Close.vi
              _Configure.vi
              _Default.vi
              _Log.vi
              Read External.vi
              Write External.vi
              UI - Edit Configuration.vi
              Write_INI_Section.vi
        Log_RT_Console
        LLAMA_Target.lvclass
      Types
    Tree.vi
LLAMA.vipb
LLAMA.vipc
LLAMA_Config.ini
```

Where Do We Go Next?

- More destinations via new Target child classes
  - Database
  - Structured TDMS execution trace
- Cross-target consolidation
  - Syslog
- Expanded API via new Message child classes



Audience Input

- What would you find useful?
- What have you done in your own logging toolchains?

LLAMA

DOWNLOAD

https://www.dmcinfo.com/services/test-and-measurement-automation/labview-programming/llama-an-extensible-logging-framework-for-labview

Questions? Contact Us…

www.dmcinfo.com
sales@dmcinfo.com
312.255.8757

DMC

Smart People. Expert Solutions.®

NIWeek

FUTURE FASTER

MAY 21–24, 2018, AUSTIN, TEXAS