Extensible Session Framework 1.0.0

Agenda

- Intro: What is a session?
- Common session design patterns
- Extensible Session Framework (ESF)
  - Optional: Refresher on LVOOP
- Examples using ESF
- Q&A

What is a session?

Many uses in computing:

- **OSI**: “A semi-permanent information exchange over a network”
- **Websites**: “A set of client-host interactions with fixed scope and duration”
- **IVI**: “A set of configuration parameters that control an instrument

What is a session?

Common elements:

1. A session contains information and allows actions to be taken on that information
2. A session has a limited lifespan
   - It is impermanent
3. A session provides an API to the applications that need to manage it

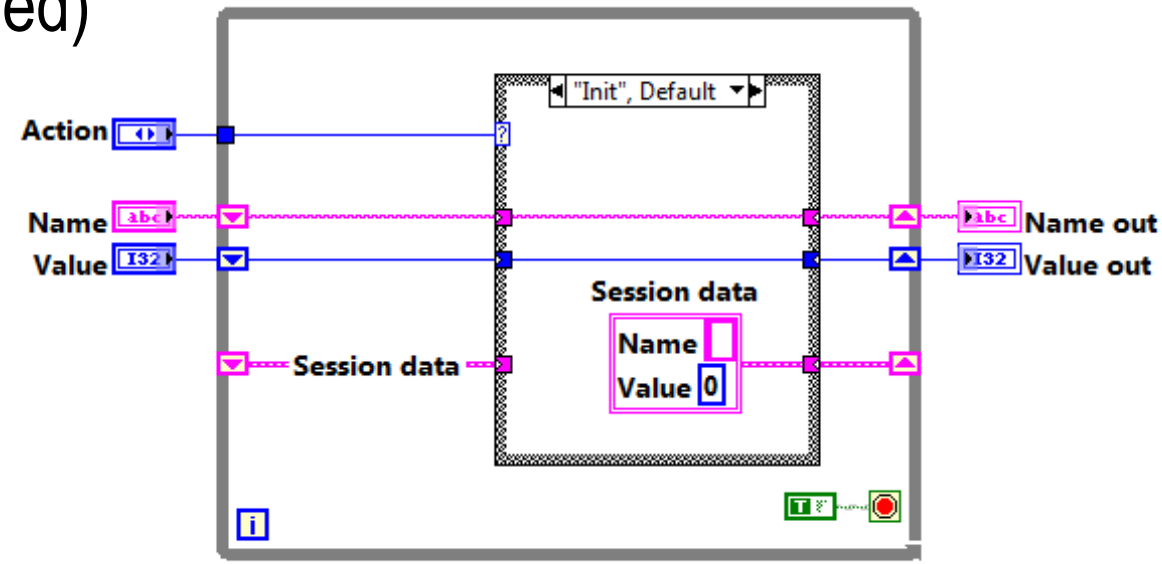
Examples of Sessions

- IVI instrument drivers establish a session to configure and control the instrument
- File I/O operations establish a session when a file is opened for access
- NI SE toolkits establish communication sessions between devices (STM) and processes (AMC)

Common session design patterns

Functional Global Variable (FGV)

- Use uninitialized shift register to manage session data
- Place wherever needed on the diagram (no handle required)

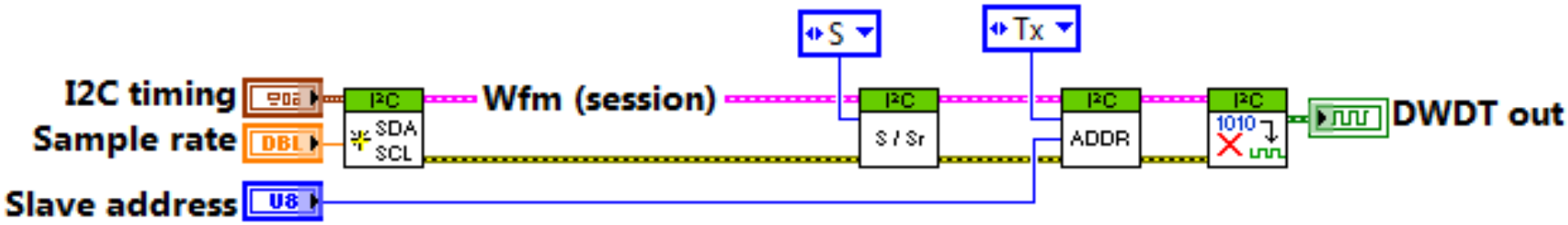


FGV – Dis/advantages

- ⊕ Easy to provide multiple access, no handle req'd
- ⊖ Confusing user interface, but can improve with wrapper functions
- ⊖ Extra work to ensure no re-initialization
- ⊖ Memory copies on non-required inputs

Data wire

- Single wire that passes through all API VIs
- Usually implemented using a cluster



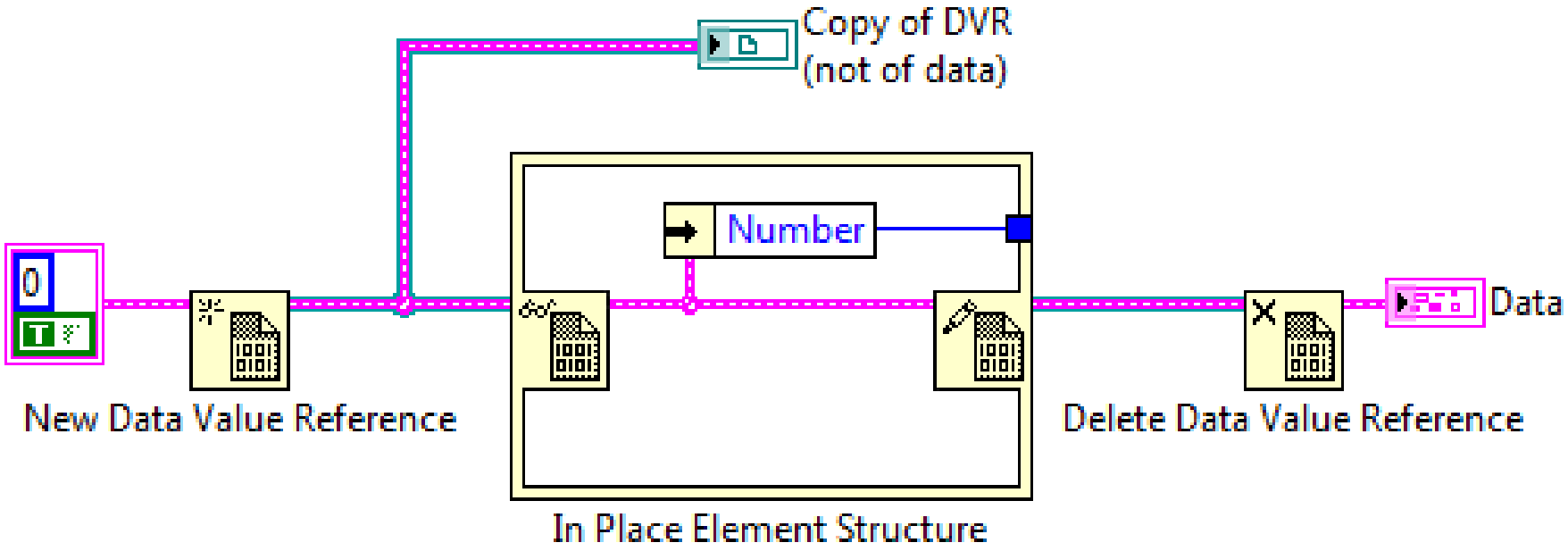
Data wire – Dis/advantages

- ⊕ Each function gets its own parameter list
- ⊕ One-time initialization by first caller
- ⊕ Inputs can be made required to avoid data copies in memory
- ⊖ No multiple access without routing wire to other diagrams
  - Wire could become invalid in a race condition

Extensible Session Framework (ESF)

LVOOP Refresher

Data Value Reference



LVOOP class – Better than a cluster

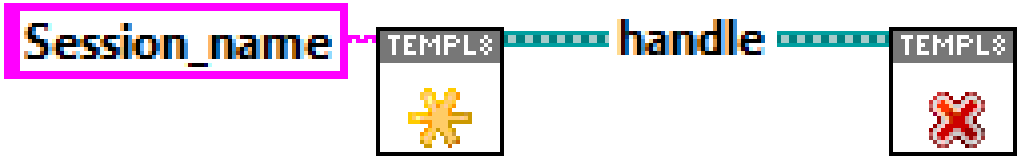
- **Inheritance**: can create a hierarchical API
  - Parent methods can act on a child session
- **Encapsulation**: session data only accessible via controlled interface
  - MUST use class methods to access object's data

LVOOP class – Better than a cluster

- **Protection**: can password-protect the class
  - Prevent snooping on private methods
- **Property nodes**: in LV 2010, can create true property nodes for data member access

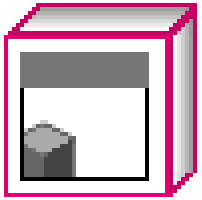
ESF structure

- Data wire with multiple-access capability



- Your session is an LVOOP class
  - Parent class is the framework's "Session – Root"
  - Handle is a Data Value Reference (DVR) to the object
  - Method VIs pass the handle as a parameter (usually)

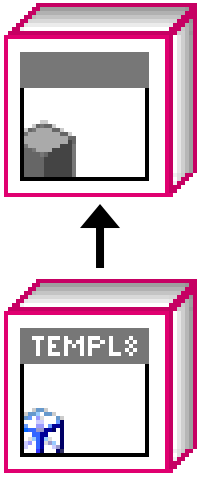
ESF's root class



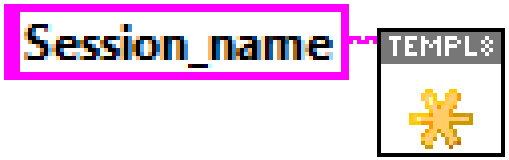
<u>Manages sessions</u>

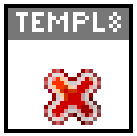
- Creates a new session or obtains a handle (DVR) to an existing one
- Keeps track of number of open handles
- "Destroys" a session when all handles have been released

ESF's template class



- Starting point for your API development
- Only two predefined methods:
  "Obtain Session" and "Release Session"





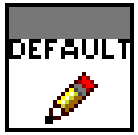
- You create a copy of it (clone), which has its own typedef but keeps the root class as its parent

ESF's optional utility methods

- "initialize Session Data"
  - Executed on first call of "Obtain Session"

- "clear Session Data"
  - Executed on last call of "Release Session"

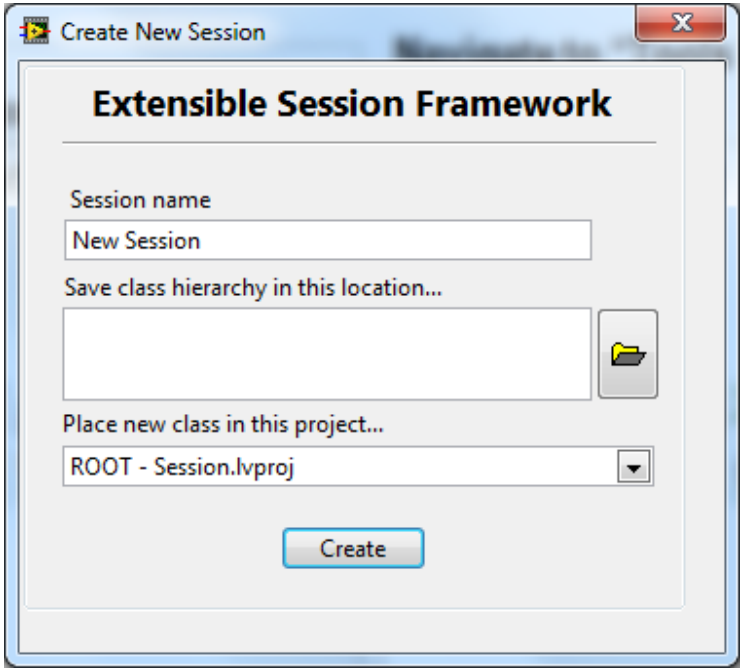
- "set Defaults"

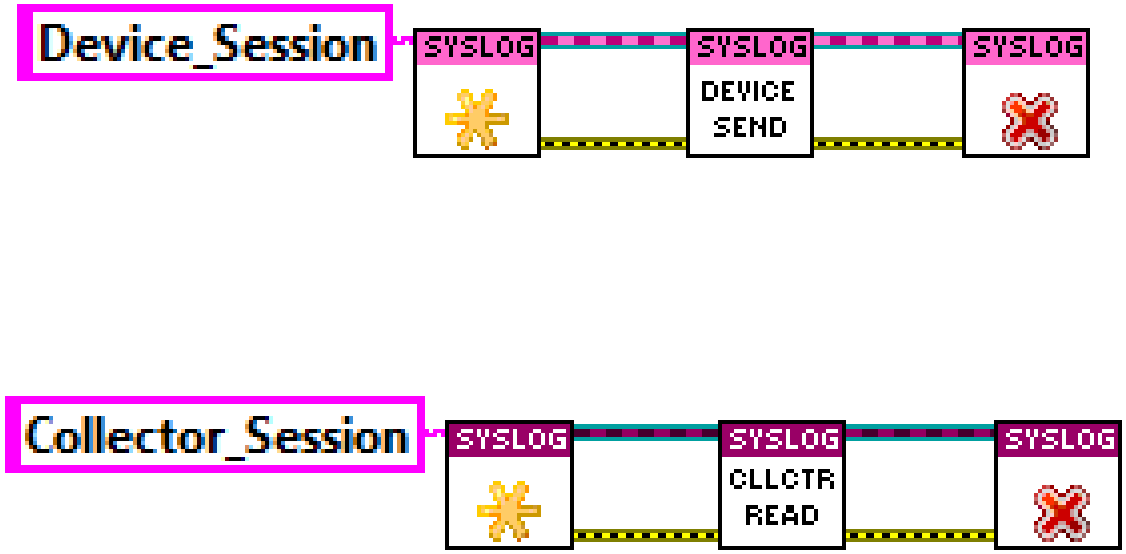


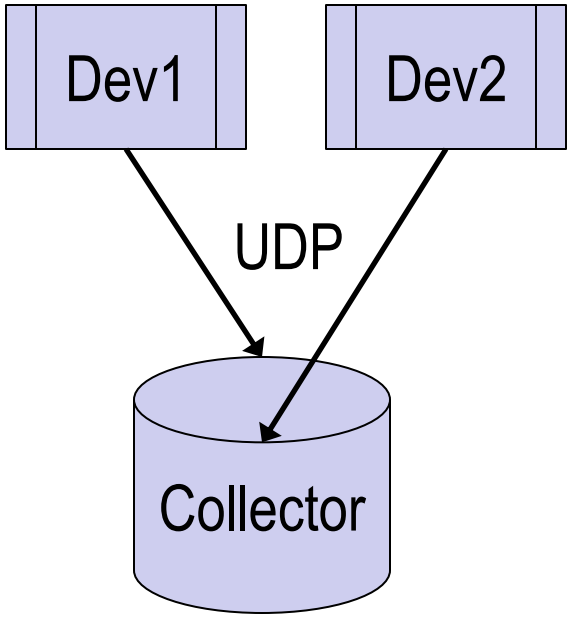
- These are called as appropriate by the root class, but defined (by you) in your API class

ESF's clone tool

To clone the template class, navigate to "Tools >> Create New Session Class…"

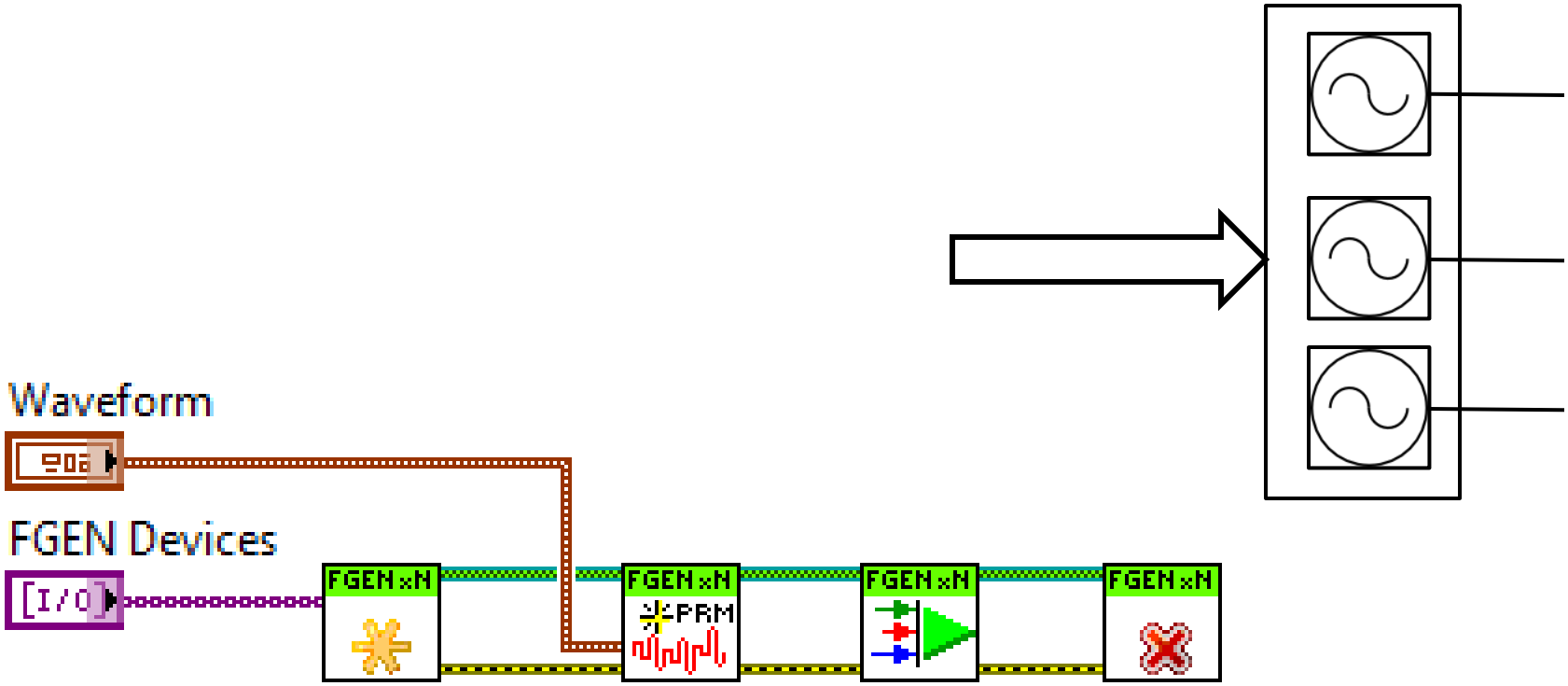






Examples using ESF

Syslog API



Examples using ESF

Multi_FGEN API

Benefits of using ESF

- All the best features of common design patterns
- Create professional software development tools
  - Property nodes!
- LVOOP-enabled if you choose to use it

Benefits of using ESF

- DVR handle makes interfacing easy
  - TestStand

Parameter Name	Type	In/Out	Default	Value
Session Name ()	ASCII String	in	☑	""
+ error in (no error)	Container	in	☑	
Session Handle	Number (U32)	out		
+ error out	Container	out		Step.Result.Error

Benefits of using ESF

- DVR handle makes interfacing easy
  - C DLL

MultiFgen.h*

(Unknown Scope)

```
void __cdecl InitializeWithOptions(char SessionName[],
    LVRefNum *SessionHandle, TD1Hdl *ResourceNames, LVBoolean *ResetDevice,
    LVBoolean *IdQuery, char OptionString[], TD2 *error);

void __cdecl Close(LVRefNum *SessionHandle, LVBoolean *forceDestroy,
    TD2 *error);

void __cdecl CreateWaveformParams(LVRefNum *SessionHandle, TD3 *Waveform,
    char ChannelName[], TD2 *error);
```