IVI Repeated Capabilities

Purpose: Cover IVI repeated capabilities

Topics:

- Describe what IVI repeated capabilities are
- Discuss 3 ways that they are implemented
- Use with the IVI Config Store
- Show their use in some examples

Repeated Capabilities

- Many instruments contain multiple instances of the same type of functionality – IVI terms these *repeated capabilities*
  - Example: Channels in an oscilloscope
  - Example: Traces or markers in a spectrum analyzer
- An instrument may have multiple sets of repeated capabilities
  - Example: A scope with channels and traces
  - Example: A device with analog channels and digital channels
- Repeated capabilities can be nested
  - Example: Traces within displays
- IVI specifies 3 ways drivers can implement repeated capabilities
- Classes partially specify repeated capabilities
  - Defines which functions and attributes apply to repeated capabilities

Repeated Capability Concepts

- Repeated capability name
  - Unique designator for a specific repeated capability in an instrument class
  - Example: IviScope spec defines “Channel” as a repeated capability name
  - Example: IviSpecAn spec defines “Trace” as a repeated capability name
- Repeated capability identifier
  - Unique designator for an instance of a particular repeated capability
  - Examples: “CH1”, “CH2” represent different instances of the “Channel” repeated capability
  - Two types exist to facilitate interchangeability: physical and virtual repeated capability identifiers
- Physical repeated capability identifier
  - Defined by specific driver
  - Placed in IVI Configuration Store by specific driver installer
- Virtual repeated capability identifier
  - Defined by end-user
  - End user maps virtual name to physical name in IVI Configuration Store
  - Required for interchangeable code

Repeated Capability Concepts

Repeated Capability Name: Channels

Defined by driver or class

Physical Repeated Capability Identifier: Chan1 Chan2 Chan3

Virtual Repeated Capability Identifier: Antenna PowerAmp Rotor

Defined by application

3 Ways to Expose Repeated Capabilities

- Parameter-style (pass element to every call)
  - Most common technique in IVI-C drivers
  - First parameter to each applicable function is a repeated capability identifier
  - Must include even if repeated capabilities are not applicable for instrument
    - Can pass in VI_NULL or an empty string if specific instrument has only one channel
- Selector-style (specify the element with a mode switch)
  - Special SetActive function used to set the active repeated capability identifier
    - All subsequent function/attribute calls use active repcap identifier
  - Useful if repcap identifier is complex and used repeatedly in a sequence of calls
- Collection-style (specify the element as a member of a collection)
  - Most common technique in IVI-COM drivers
  - Much simpler than other repcap styles when nesting is involved
  - Works a lot like standard COM collections
    - But w/o the nice VB for-each syntax

3 Ways to Expose Repeated Capabilities

- Parameter-style (pass element to every call)

```
AgM950x_FanTraySpeed(vi, “Tray1”, &speed);
```

- Selector-style (specify the element with a mode switch)

```
AgM950x_FanTraySpeedSelect(vi, “Tray1”);
AgM950x_FanTraySpeed(vi, &speed);
```

- Collection-style (element indexes into a collection)
  - Available (and preferred) with IVI-COM and IVI .NET (and preferred)

```
Int32 speed = myChassis.FanTray[“Tray1”].FanTraySpeed
```

Repeated Capability Attributes and Functions

Technique	Attributes	Functions
Parameter	<Capability> Count <Capability> Name (COM only)	Get <Capability> Name (C only)
Selector	<Capability> Count Active <Capability> <Capability> Name (COM only)	Get <Capability> Name (C only) SetActive <Capability>
Collection*	<Capability>s.Item <Capability>s.Count <Capability>s.Name**	*Not supported*

* IVI-COM collection attributes are placed in a collection interface with a name ending in <Capability> followed by an 's'.

** IVI-COM collections are 1-based

“Trace” Repeated Capability Example

- Class specification defines a “Trace” repeated capability

Technique	Attributes	Functions
Parameter	**TraceCount** **TraceName (COM only)**	**GetTraceName (C only)**
Selector	**TraceCount** **ActiveTrace** **TraceName (COM only)**	**GetTraceName (C only)** **SetActiveTrace**
Collection*	**Traces.Item** **Traces.Count** **Traces.Name****	***Not supported***

Selector-Style Repeated Capabilities

```
‘ Repcap identifier only specified once => convenient for complex identifiers

Dim specan as New AgilentPSA

specan.ActiveTrace = “Trace1,Trace2,Trace3”

specan.Bandwidth = 4E6
specan.Frequency = 3E9
specan.Span = 2E10
```

Repeated Capabilities Using Collections

- For each syntax not supported for IVI-COM collections
  - IVI-COM collections are not “real” COM collections
  - COM collections require IDispatch and IVI-COM interfaces are intentionally not IDispatch-based

```
Dim specan as New AgilentPSA
Dim trace as IAgilentPSATrace

Set trace = specan.Traces.Item(“Trace1”)

trace.Bandwidth = 4E6
trace.Frequency = 3E9
trace.Span = 2E10
```

Repeated Capabilities and IVI-C Attributes

- All IVI-C attribute accessors accept a repcap identifier as a parameter
  - Can pass VI_NULL or empty string if repeated capabilities do not apply to the attribute being read/written

```
agpsa_SetAttributeViReal64 (ViSession Vi, ViConstString RepCapIdentifier,
                            ViAttr AttributeID,
                            ViReal64 AttributeValue);
```

```
ViSession vi;
ViStatus viStatus = agpsa_init("GPIB::10", VI_FALSE, VI_FALSE, &vi);

viStatus = agpsa_SetAttributeViReal64(vi, "Trace1", AGPSA_ATTR_BANDWIDTH, 3E6);
viStatus = agpsa_SetAttributeViReal64(vi, "Trace1", AGPSA_ATTR_SPAN, 2E9);
```

Comparing IVI-COM and IVI-C

IVI-COM

- Collection interfaces indicate what functionality applies to repeated capabilities.

```
myNA.Window[“a1”].Trace[“S11”].Start=23;
```

IVI-C

- Need to know which attributes apply to a repeated capability and which apply to the driver as a whole.
- Nested repeated capabilities use an IVI-defined string-based syntax.

```
Acme12_WindowTraceStart(vi,“a1:S11”,23)
```

Repeated Capability Access Pitfalls

```
// Wrong – must indicate which trigger repeated capability
ag34401_SetAttributeViReal64(session, VI_NULL,
        AG34401_ATTR_TRIGGER_LEVEL, 0.45);

// Wrong – Range applies to the whole driver, not to Channel1
ag34401_SetAttributeViReal64(session, "Channel1",
        AG34401_ATTR_RANGE, 100);

// Wrong – Enabled is a property of output repeated capability,
// not the trigger repeated capability
ag34401_SetAttributeViBoolean(session, "Out1:Trig1",
        AG34401_ATTR_OUTPUT_ENABLED, VI_TRUE);
```

Strings are not checked until runtime

Selecting Multiple Capabilities At Once

- Parameter used to specify repeated capability instances is known as a *repeated capability selector*
  - Same rules apply for all 3 repeated capability techniques
- *Simple repeated capability selector*
  - Single, non-nested repcap instance
  - May be a physical or virtual identifier
  - Example: “chan1”
- *Repeated capability ranges*
  - Lower bound to upper bound
  - Example: “1-3”, “8-10”
- *Repeated capability lists*
  - Simple comma-separated list
  - Example: “1, 4, 7, 9”
  - Combined Example: “1-3, 6, 8, 10-12”

- This demonstration shows
  - Selector style repeated capability in IVI-C
- Repeated Capability CVI Demo

Power supply with multiple outputs
Repeated capability allows controlling each output