

Observatory Sciences Ltd



LabVIEW Native Channel Access for EPICS

LANCE-01

Issue: 1.0

Date: 28.04.2010

OSL Author(s):

A. Greer
Name Date Signature



**LabVIEW Native Channel Access for
EPICS**

Doc: LANCE-01

Issue: Issue 1.0

Date: 28.04.2010

Page: 2 of 15

CHANGE RECORD

ISSUE	DATE	SECTION/PARA. AFFECTED	REASON/INITIATION DOCUMENTS/REMARKS
1.0	28.04.2010	All	First issue



TABLE OF CONTENTS

1	Introduction.....	4
1.1	Scope	4
1.2	Intended Audience.....	4
1.3	Glossary, Definitions and Conventions.....	4
1.4	Abbreviations and Acronyms	4
2	Related Documents	5
2.1	Reference Documents	5
3	Overview.....	6
4	Installation	7
4.1	Prerequisites.....	7
4.2	Installation Instructions	7
4.3	Demo Application	7
5	Description	9
5.1	Calnit	9
5.2	CaGet	9
5.3	CaPut.....	10
5.4	CaMonitorOn	10
5.5	CaMonitor	11
5.6	CaMonitorOff	11
5.7	CaClose.....	12
6	Limitations	13
7	Future Development	14



1 Introduction

1.1 Scope

This document describes the use of LabVIEW Native Channel Access for EPICS (LANCE) VIs allowing integration of LabVIEW applications into an EPICS environment without the use of externally compiled code or any additional LabVIEW modules.

1.2 Intended Audience

The intended audience of this document is software architects, software managers, designers and potential integrators of LabVIEW and EPICS systems.

1.3 Glossary, Definitions and Conventions

Definitions of terms used for each subsystem are provided at the beginning of the subsystem requirements chapter.

1.4 Abbreviations and Acronyms

API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
EPICS	Experimental Physics and Industrial Control System
GUI	Graphical User Interface
LANCE	LabVIEW Native Channel Access for EPICS
PV	Process Variable
SW	Software
TBD	To Be Defined
TCS	Telescope Control System



2 Related Documents

2.1 Reference Documents

	Title	Reference	Issue
RD1	Channel Access Protocol Specification http://epics.cosylab.com/cosyjava/JCA-Common/Documentation/Caproto.html	n/a	n/a



3 Overview

The LANCE VIs have been developed to provide easy integration between LabVIEW applications and EPICS applications. The following goals were considered when developing the set of VIs:

- 1) The VIs should work on LabVIEW for Windows and Linux
- 2) The VIs should not require any external code or libraries
- 3) The public interface (public VIs) should be kept as simple as possible
- 4) As much implementation as possible should be hidden from the public interface
- 5) The “caget”, “caput” and “camonitor” capabilities should be provided

To achieve the first goal the VIs were developed on a LabVIEW for Linux development system. The channel access protocol was taken from RD1 and all necessary structures developed into LabVIEW strict type definitions. The communication is performed using the LabVIEW tcp and udp nodes and event notification (required for channel access monitoring) is performed using LabVIEW notifier nodes. Using these basic LabVIEW nodes ensures (2) is met. The remainder of the goals have been met by using standard LabVIEW programming practices where possible and the result is a small set of public VIs that should feel comfortable to use by a LabVIEW developer or an EPICS developer.



4 Installation

This section details installation of the LANCE VIs.

4.1 Prerequisites

The software has been developed and tested on Windows XP and Linux RHEL 5. The software should work on any platform supported by LabVIEW version 2009.

4.2 Installation Instructions

To use the LANCE VIs in an application simply unpack the tar file downloaded from the Observatory Sciences Ltd website

(http://www.observatorysciences.co.uk/labview_downloads.php). Once unpacked the public VIs can be found at:

```
<unpacked_location>/channel_access/vi/public
```

Once located from within a project file they can be dragged into any other open VI. Use of LabVIEW projects is beyond the scope of this document.

4.3 Demo Application

There is a demonstration VI that can “caput”, “caget” and “camonitor” a single value. This can be found at

```
<unpacked_location>/channel_access/vi/demo/ChannelAccessDemo.vi
```

The VI can be run to test that LANCE is communicating correctly. The figure below shows the demo application running and monitoring a PV from a TCS application. To perform an operation on a particular PV enter the name of the PV into the “PV Name” text entry box. Once entered the value can be retrieved by clicking on the “CaGet” button. The returned value is placed into the “Current Value” text indicator. If continuous monitoring of the PV is required then click on the “CaMon” button. Whenever the value is updated the new value is placed into the “Monitored Value” text indicator. To stop monitoring the value click on the “MonOff” button. To set a new value enter the required value into the “New Value” text entry box and then click on the “CaPut” button. The new value request is sent to the PV. To stop the demo VI running click on the “STOP” button located at the bottom right of the screen.



LabVIEW Native Channel Access for EPICS

Doc: LANCE-01
Issue: Issue 1.0
Date: 28.04.2010
Page: 8 of 15

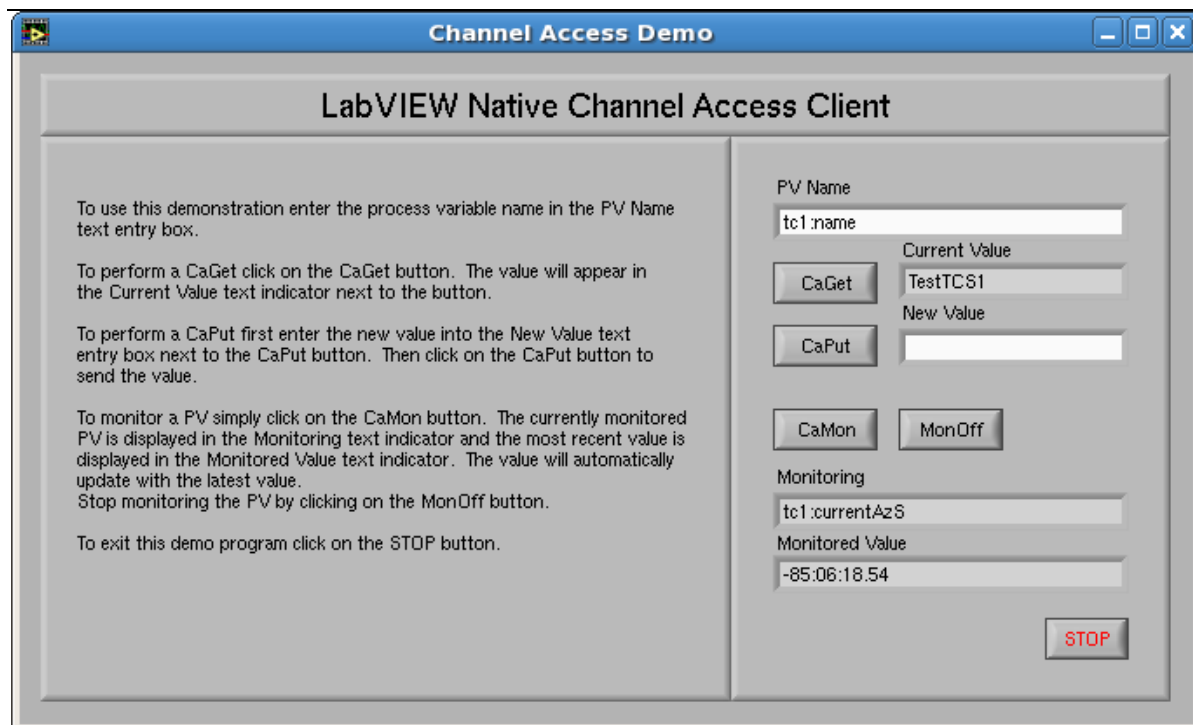


Figure 1 Channel Access Demonstration VI



5 Description

The following section describes in detail each public LANCE VI. There are two restrictions placed on any application using the LANCE VIs. The first, is that the “Calnit” VI is executed before any other LANCE VIs. The second is that the “CaClose” VI is executed when shutting down a running application. Failure to adhere to these restrictions could result in cloned VIs continuing to run and possibly incorrect monitored values being returned (as a result of Virtual Circuits not getting destroyed).

5.1 Calnit



Figure 2 Calnit VI

Inputs: Standard error in
Outputs: Standard error out
Notes: This VI must be executed before any other LANCE VIs.

5.2 CaGet

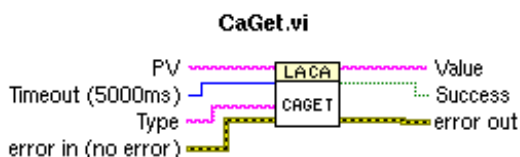


Figure 3 CaGet VI

Inputs: PV. The name of the PV to get the value of.
Timeout (5000ms). The time to wait before giving up on getting the value. Set this to -1 to wait forever.
Type. This VI is polymorphic and can accept string, int32 or double types. The value is not used, only the type is important.
Standard error in

Outputs: Value. The value retrieved. The type of this output will be the same as the input “Type” parameter.
Success. If this is true the value is valid, if this is false the VI has failed to get the value.
Standard error out

Notes: This can be used to obtain the value of any PV.



5.3 CaPut

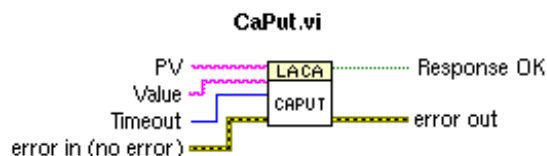


Figure 4 CaPut VI

- Inputs:**
- PV. The name of the PV to get the value of.
 - Value. The value to write to the PV. This VI is polymorphic and can accept values of the type string, int32 or double.
 - Timeout (5000ms). The time to wait before giving up on setting the value. Set this to -1 to wait forever.
- Outputs:**
- Response OK. This is true if the write completed OK. If the write failed (for example if the PV was not found) then this will be false.
 - Standard error out
- Notes:**
- Use this VI to write a value into any PV.

5.4 CaMonitorOn

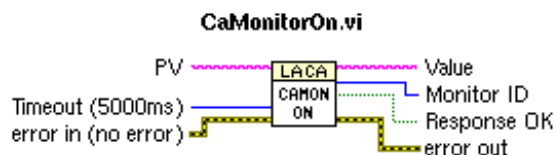


Figure 5 CaMonitorOn VI

- Inputs:**
- PV. The name of the PV to start monitoring.
 - Timeout (5000ms). The time to wait before giving up starting the monitor. Set this to -1 to wait forever.
- Outputs:**
- Value. This returns the initial value of the PV that is to be monitored.
 - Monitor ID. An ID unique to each monitor. Use this to wait for monitor events.
 - Response OK. This is true if the monitor was setup OK. If it failed (for example if the PV was not found) then this will be false.
 - Standard error out
- Notes:**
- Call this VI to setup a monitor. If successful then the monitor ID passed out can be used by the "CaMonitor" VI (see below) to wait for monitor events.



5.5 CaMonitor

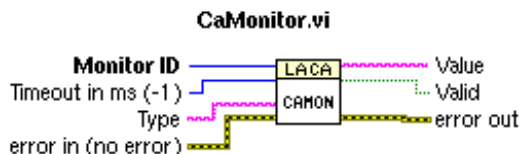


Figure 6 CaMonitor VI

- Inputs:**
- Monitor ID. This must be supplied. Use the “CaMonitorOn” VI to get a valid monitor ID for a PV.
 - Timeout in ms (-1). The time to wait before giving up on getting the value. Set this to -1 to wait forever.
 - Type. This VI is polymorphic and can accept string, int32 or double types. The value is not used, only the type is important.
 - Standard error in
- Outputs:**
- Value. The value retrieved. The type of this output will be the same as the input “Type” parameter.
 - Valid. If this is true the value is valid, if this is false the VI has failed to get the value.
 - Standard error out
- Notes:**
- This VI will wait for a monitor event to be received (or for the timeout). The valid flag should always be checked before using the value to be sure that a monitor has been received. If this VI is waiting and the monitor is destroyed then it will immediately return with the valid value set to false.

5.6 CaMonitorOff

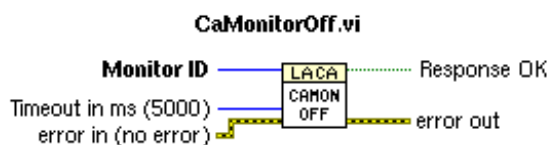


Figure 7 CaMonitorOff VI

- Inputs:**
- Monitor ID. This must be supplied. Use the “CaMonitorOn” VI to get a valid monitor ID for a PV.
 - Timeout (-1ms). The time to wait before giving up on stopping the monitor. Set this to -1 to wait forever.
 - Standard error in
- Outputs:**
- Response OK. If this is true the monitor destruction has completed OK.
 - Standard error out
- Notes:**
- This VI will stop a monitor on a PV. Any “CaMonitor” VIs immediately start triggering with their valid flags set to false.



5.7 CaClose

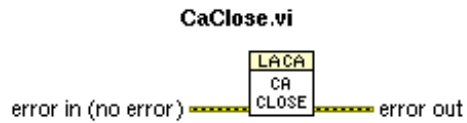


Figure 8 CaClose VI

Inputs: Standard error in

Outputs: Standard error out

Notes: This VI should be executed before closing down any application that is using the LANCE VIs.



6 Limitations

This version of LANCE is very much a work in progress. I hope that it can be of use to anyone wanting to integrate LabVIEW and EPICS applications and eventually it can grow into a much more stable VI set offering full channel access capabilities.

I believe the current LANCE version has the following limitations. With each point are my comments on how to overcome the obstacle.

- 1) It was developed on LabVIEW 2009 and I don't believe it can be converted into earlier versions from the current VI set. However, the VIs could be recreated in earlier versions as the nodes used have been around for a while.
- 2) If the IOC is restarted monitors do not reconnect automatically. Right now you just have to restart the LabVIEW application.
- 3) The types are limited to string, double and int32.
- 4) Currently a few things (port numbers) are hardcoded.
- 5) This is just a client. It would be nice to be able to offer a full channel access server.



7 Future Development

If there is sufficient interest from the community the following features could be added in a future release:

- 1) Automatic reconnection if an IOC is rebooted.
- 2) Puts and Gets of arrays of values.
- 3) Allow configuration of EPICS settings from within the LabVIEW environment (or through the use of VIs).
- 4) Retrieve all information from PVs, not just the value. This would be useful for LabVIEW displays as well as other applications.



LabVIEW Native Channel Access for EPICS

Doc: LANCE-01
Issue: Issue 1.0
Date: 28.04.2010
Page: 15 of 15

__oOo__