Interoperating With Two VISA Implementations on the Same Computer

Accessing multiple instruments from VISA is a simple matter when the VISA implementation you are using supports the interfaces and hardware you need to access. There may be times, however, when two different pieces of hardware you need to access are not both supported by a single VISA implementation. This article describes how Agilent and National Instruments VISA implementations support interoperability with each other's hardware, how to set up a program to access either Agilent or NI VISA, and how to dynamically load both Agilent and NI VISA to allow a single program to use both VISA implementations. Source code is provided to simplify the dynamic loading and calling of VISA functions in both C and C++ in the 32-bit Windows environment.

Interoperability Using a Single VISA Implementation.

Agilent and National Instruments have worked together to solve some of these interoperability problems within their respective VISA implementations. With this approach, a single VISA implementation can be used to access hardware from each vendor. Some examples of this are:

- **GPIB-VXI** – GPIB-VXI functionality for a VXI Command Module is implemented in a vendor-specific GPIB-VXI DLL. This DLL has a publicly defined interface that both Agilent and NI VISA understand. Agilent and NI each supply a custom DLL for their GPIB-VXI command modules. When a GPIB-VXI interface is configured, VISA queries the VXI command module for its '*IDN' string and using a common _gpibvxi.ini file, VISA can determine which GPIB-VXI DLL to use. This technique allows both Agilent and NI VISA to interoperate with each other's VXI Command Modules. It even allows a mixture of Agilent and NI VXI Command Modules in the same system.

- **Agilent TULIP Drivers** – By using specially designed TULIP[1] drivers, Agilent VISA and SICL can access NI hardware through NI's NI-488 and NI-VXI libraries. NI interfaces that can be accessed through NI-488 and NI-VXI can be configured in Agilent IO Libraries to allow access to devices on these interfaces[2].

- **NI Passport** – National Instruments provides a 'Passport' layer beneath their VISA implementation and a 'VISA Library Passport for Tulip' DLL which allows NI VISA to be configured to use Agilent hardware[2].

Interoperability Using More Than One VISA Implementation.

On Microsoft Windows, the VISA specification dictates that the VISA library be named *visa32.dll* and that it be installed in the Windows system directory (e.g. *\windows\system32*). This makes it impossible to have two different VISA implementations installed on a single computer at the same time. To allow for better VISA interoperability, Agilent VISA is implemented in a DLL named *agvisa32.dll* and Agilent's *visa32.dll* is simply a shell that redirects VISA calls to *agvisa32.dll*. Normally when you the install Agilent IO Libraries, both *visa32.dll* and *agvisa32.dll* are installed in the Windows system directory. There is, however, an installation option that allows Agilent VISA to be installed as secondary (side-by-side). When installed as secondary, Agilent's *visa32.dll* is not installed; only *agvisa32.dll* is installed. This will prevent the overwriting of NI's *visa32.dll* if it had previously been installed. In addition, when installed side-by-side, Agilent's standard VISA support files (e.g. *visa.h*, *visa32.lib*, etc.) are not placed in their standard locations so NI's support files will not be overwritten.

---

[1] TULIP (The User-Land Interface Protocol) is an Agilent designed API for low-level drivers used by Agilent's IO Libraries.

[2] Refer to the *'Using Agilent and National Instruments Hardware and Software in the Same System'* article in the *'Troubleshooting'* section of the Agilent *IO Libraries Readme* for configuration details and limitations.

Copies of all of the Agilent's standard VISA support files as well as Agilent's visa32.dll are installed under the *agvisa* subdirectory of the VISA framework path so these files are available to programmers even when Agilent VISA is installed as secondary (side-by-side). The default framework path for Agilent VISA is *'c:\program files\VISA\win95'* for Windows 9x and *'c:\program files\VISA\winnt'* for Windows NT, 2000 and XP.

If you have NI VISA installed and Agilent VISA installed as secondary (side-by-side), there are several steps you can take to have specific VISA programs use whichever VISA implementation you desire.

Statically Linked VISA Programs

When a program is built it is normally statically linked with the DLL's it requires. This is accomplished by linking the program with the *.lib* file that corresponds to the DLL. For example most VISA programs are linked with *visa32.lib* so that they will use *visa32.dll* when they are run. If a program is linked with *agvisa32.lib*, it will use *agvisa32.dll* when it runs. When Windows loads a program and prepares it for execution, it searches for statically linked DLL's in specific locations. When it finds a DLL the program requires, it then maps the DLL into the program's memory space before executing the program. The search order for statically linked DLLs is documented in the 'LoadLibrary' function reference in the MSDN Library. Referring to that documentation, the normal DLL search order is:

1. The directory from which the application loaded.
2. The current directory.
   Windows XP: If **HKLM\System\CurrentControlSet\Control\SessionManager\SafeDllSearchMode** is 1, the current directory is the last directory searched. The default value is 0.
3. The Windows system directory. Use the **GetSystemDirectory** function to get the path of this directory.
   Windows NT/2000/XP: The name of this directory is System32.
4. **Windows NT/2000/XP:** The 16-bit Windows system directory. There is no function that obtains the path of this directory, but it is searched. The name of this directory is System.
5. The Windows directory. Use the **GetWindowsDirectory** function to get the path of this directory.
6. The directories that are listed in the PATH environment variable.

- **Link your program with visa32.dll or agvisa32.dll:**

  If you are developing a VISA program to run in a dual-VISA environment with NI VISA installed as primary and Agilent VISA installed as secondary (side-by-side), you can determine which VISA the program will use by linking with the appropriate *.lib* file. To use NI VISA, link the program with *visa32.dll* or to use Agilent VISA, link the program with *agvisa32.dll*. There is a situation, however, where linking with *agvisa23.lib* may not work as expected. If your program uses an additional DLL such as a Plug&Play driver that has been linked with *visa32.lib*, then even though your program will use *agvisa32.dll*, the code in the additional DLL will use *visa32.dll*. To have your program and this additional DLL both use *agvisa32.dll* you can use the technique described in the following paragraph.

- **Link your program with visa32.dll and make use of the Windows search order:**

  If you want to use an existing VISA program that has already been linked with *visa32.lib* or if your program uses a pre-existing DLL that requires *visa32.dll*, you can use the DLL search order described above to have Windows load Agilent's VISA implementation. If you copy the Agilent version of *visa32.dll* to the same directory as the program you are executing. Windows will find and load the Agilent *visa32.dll* rather than the NI version from the Windows system directory.

The above techniques will work for programs that use a single VISA implementation. If you want to call both Agilent and NI VISA from within the same program, you will have to dynamically load both VISA's.

Dynamically Linked VISA Programs

Instead of linking your program with *visa32.lib* or *agvisa32.lib*, you can use the Win32 function 'LoadLibrary' dynamically load the VISA library of your choice and the 'GetProcAddress' function to get a pointer to each of the VISA functions that you wish to call. When statically linking, you cannot link with both *visa32.lib* and *agvisa32.lib* since the entry point names (functions) are identical in both DLL's and the Windows loader can't determine which library to use when your program calls a VISA function. When dynamically linking, however, you can call 'LoadLibrary' to load one or both VISA libraries and then call 'GetProcAddress' on these loaded libraries to get function pointers to the VISA functions you wish to call. This makes it very easy to use both VISA libraries in a single program – something that you cannot do with a statically linked program.

Using the source code supplied with this article can simplify the task of dynamically loading VISA in your C or C++ program.

Here are some pointers to keep in mind when you want to use two different VISA implementations in the same program:

1. Refer to the main.cpp and main.c programs below for examples of calling VISA functions through the DynamicVisa class (C++) or struct (C).

2. When installing dual VISA's, first install NI VISA and then install the Agilent IO Libraries. The Agilent IO Libraries install program will detect the presence of NI VISA and default to the side-by-side VISA installation. If you need them, remember that copies of the Agilent VISA support files are installed in the VISA framework path in the *agbin*, *include* and *lib* directories.

3. In the C implementation, you must be sure to call the 'DynamicVisaInitialize' function before attempting to call through the VISA function pointers in the DynamicVisa structure. If this structure is not initialized, your program will crash when you attempt to call through an uninitialized pointer.

4. When using session or object values returned from VISA calls in one VISA implementation, always use the returned values in calls to the same VISA implementation. For example:

```
ViSession agtDrm, agtVi;
ViSession priDrm, priVi;
ViStatus  status;
//
status = agtVisa.viOpenDefaultRM(&agtDrm);
status = priVisa.viOpenDefaultRM(&priDrm);
//
// Note that using 'priDrm' in the following call instead of
// 'agtDrm' would most likely fail or cause unpredictable
// results because 'priDrm' was returned from the a priVisa
// call but it is being used in an agtVisa call.
//
status = agtVisa.viOpen(agtDrm,"ASRL1::INSTR",VI_NULL,VI_NULL,&agtVi);
```

- The following files C++ are supplied with this article:

C++ File Name	Description
DualVisa.h	header file – definition of DualVisa class
DualVisa.cpp	Implementation of DualVisa class
main.cpp	Sample C++ program showing how to call two VISA's from same program

The source for main.cpp is shown below. See the comments in DualVisa.h and DualVisa.cpp for more implementation details.

```
//
// main.cpp
//   This is a sample C++ program that illustrates how to dynamically load
//   and use two different VISA implementations in the same program.
//
#include <stdio.h>
#include"dualVisa.h"
//
// Create two static instances of the DynamicVisa class.
// One pointing to Agilent VISA (agvisa32.dll) and the
// other to the pimary VISA (visa32.dll).
//
DynamicVisa agilentVisa("agvisa32.dll");
DynamicVisa primaryVisa("visa32.dll");
//
int main( int argc, char** argv ) {
   ViStatus  statusA  = 0;
   ViStatus  statusP  = 0;
   ViSession agtDrm   = 0;
   ViSession priDrm   = 0;
   ViUInt16  udata16A = 0;
   ViUInt16  udata16P = 0;
   //
   printf( "agilentVisa.MissingFnCount = %d\n", agilentVisa.MissingFnCount() );
   printf( "primaryVisa.MissingFnCount = %d\n", primaryVisa.MissingFnCount() );
   printf( "\n" );
   //
   statusA = agilentVisa.viOpenDefaultRM( &agtDrm );
   statusP = primaryVisa.viOpenDefaultRM( &priDrm );
   printf( "agilentVisa.viOpenDefaultRM:         agtDrm=  0x%08x, status=0x%08x\n", agtDrm, statusA );
   printf( "primaryVisa.viOpenDefaultRM:         priDrm=  0x%08x, status=0x%08x\n", priDrm, statusP );
   printf( "\n" );
   //
   statusA = agilentVisa.viGetAttribute(agtDrm,VI_ATTR_RSRC_MANF_ID,&udata16A);
   statusP = primaryVisa.viGetAttribute(priDrm,VI_ATTR_RSRC_MANF_ID,&udata16P);
   printf( "agilentVisa.viGetAttribute : VI_ATTR_RSRC_MANF_ID=0x%04x, status=0x%08x\n", udata16A, statusA );
   printf( "primaryVisa.viGetAttribute : VI_ATTR_RSRC_MANF_ID=0x%04x, status=0x%08x\n", udata16P, statusP );
   printf( "\n" );
   //
   statusA = agilentVisa.viClose( agtDrm );
   statusP = primaryVisa.viClose( priDrm );
   printf( "primaryVisa.viClose        :         agtDrm=  0x%08x, status=0x%08x\n", agtDrm, statusA );
   printf( "agilentVisa.viClose        :         priDrm=  0x%08x, status=0x%08x\n", priDrm, statusP );
   printf( "\n" );
   return 0;
}
```

The following files C are supplied with this article:

C File name	Description
DualVisaC.h	header file – definition of support functions and VISA function pointer struct.
DualVisaC.c	Implementation of support functions
main.c	Sample C program showing how call two VISA's from the same program

The source for main.c is shown below. See the comments in DualVisaC.h and DualVisaC.c for more implementation details.

```
//
// main.c
//   This is a sample C program that illustrates how to dynamically load
//   and use two different VISA implementations in the same program.
//
#include <stdio.h>
#include "dualVisaC.h"
//
// Create two static instances of the DynamicVisa struct.
// One pointing to Agilent VISA (agvisa32.dll) and the
// other to the pimary VISA (visa32.dll).
//
struct DynamicVisa agilentVisa;
struct DynamicVisa primaryVisa;

int main( int argc, char** argv ) {
   ViStatus  statusA  = 0;
   ViStatus  statusP  = 0;
   ViSession agtDrm   = 0;
   ViSession priDrm   = 0;
   ViUInt16  udata16A = 0;
   ViUInt16  udata16P = 0;
   //
   // Initialize the DynamicVisa structures (load the VISA libraries)
   //
   DynamicVisaInitialize( "agvisa32.dll", &agilentVisa );
   DynamicVisaInitialize( "visa32.dll",   &primaryVisa );
   //
   printf( "agilentVisa.MissingFnCount = %d\n", MissingFnCount(&agilentVisa) );
   printf( "primaryVisa.MissingFnCount = %d\n", MissingFnCount(&primaryVisa) );
   printf( "\n" );
   //
   statusA = agilentVisa.viOpenDefaultRM( &agtDrm );
   statusP = primaryVisa.viOpenDefaultRM( &priDrm );
   printf( "agilentVisa.viOpenDefaultRM:         agtDrm=  0x%08x, status=0x%08x\n", agtDrm, statusA );
   printf( "primaryVisa.viOpenDefaultRM:         priDrm=  0x%08x, status=0x%08x\n", priDrm, statusP );
   printf( "\n" );
   //
   statusA = agilentVisa.viGetAttribute(agtDrm,VI_ATTR_RSRC_MANF_ID,&udata16A);
   statusP = primaryVisa.viGetAttribute(priDrm,VI_ATTR_RSRC_MANF_ID,&udata16P);
   printf( "agilentVisa.viGetAttribute : VI_ATTR_RSRC_MANF_ID=0x%04x, status=0x%08x\n", udata16A, statusA );
   printf( "primaryVisa.viGetAttribute : VI_ATTR_RSRC_MANF_ID=0x%04x, status=0x%08x\n", udata16P, statusP );
   printf( "\n" );
   //
   statusA = agilentVisa.viClose( agtDrm );
   statusP = primaryVisa.viClose( priDrm );
   printf( "agilentVisa.viClose        :         agtDrm=  0x%08x, status=0x%08x\n", agtDrm, statusA );
   printf( "primaryVisa.viClose        :         PriDrm=  0x%08x, status=0x%08x\n", priDrm, statusP );
   printf( "\n" );
   //
   // Uninitialize the DynamicVisa structures (unload the VISA libraries)
   //
   DynamicVisaUninitialize( &agilentVisa );
   DynamicVisaUninitialize( &primaryVisa );
   return 0;
}
```