Spark HOLLAND

TECHNICAL NOTE OCTOBER 2024

Empowering Laboratory Instruments Control:

Python integration on Linux for Spark Devices

Better Sample Care.

› visit us at sparkholland.com

In the era of cross-platform development facilitated by .NET Core, the compatibility of .NET application with Linux and MacOS has become a reality. In this technical note, we illustrate a significant achievement: the seamless integration of the Spark Holland instrument control library (ICI) into a Python application running on Linux. By showcasing this successful integration, we aim to inspire developers to explore the versatile capabilities of the ICI library, enabling support for Spark Holland devices across a spectrum of programming languages and operating systems beyond the confines of Windows.

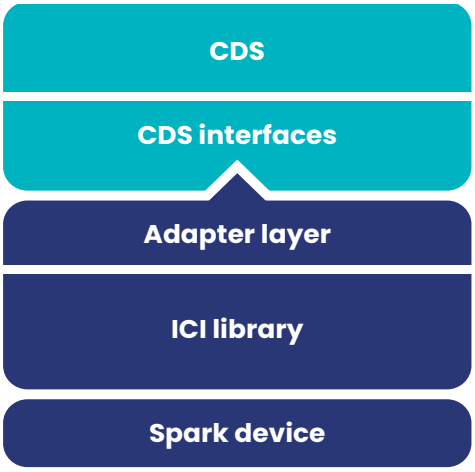


Figure 1. Simplified architectural overview.

```
public class AdaptedController: ICdsController
{
  // Constructor
  public AdaptedController()
  {
      // Instatiate IciController
      _iciController = MainProvider.CreateController();
  }

  private IIciController _iciController;
  private IciConfiguration _iciConfigurationProvider;

  // ICdsController functions

  // AdaptedController: Connects and initializes a device
  public void AdaptedController.Initialize()
  {
    _iciController.Connect()
  }

  // AdaptedController: Start a device run
  public void AdaptedController.StartRun()
  {
    _iciController.StartActualRun()
  }

  // etc.
}
```

Listing 1. Adapting IciController to be used in a CDS.

Background

In order to streamline integration into analytical workflows, Spark Holland has developed the Instrument Control Interface (ICI) software library. This library includes essential functionalities such as communication protocols and reporting mechanisms, simplifying driver development.

Originally developed for in-house use, the ICI library now extends its benefits to OEM partners, ensuring Spark Holland device support in their chromatography data systems (CDS). As most data systems operate on .NET platforms within Windows environments, the ICI library was originally targeting this ecosystem.

However, expanding its compatibility to other operating systems such as Linux opens avenues for wider adoption and integration across diverse software environments.

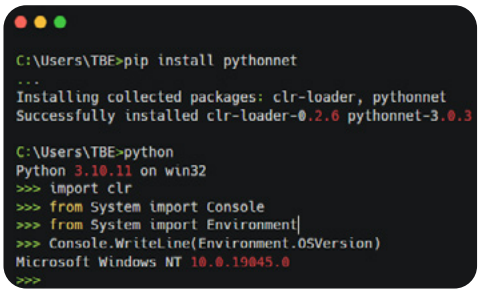
Technical details

Figure 1 illustrates the architectural overview of the ICI library when integrated into a Chromatography Data System (CDS) device driver. The CDS application communicates with the device driver through a dedicated interface layer tailored to the specifics of the CDS.

Meanwhile, the ICI library exposes a set of interfaces, facilitating direct interaction with the Spark device. To bridge the gap between the CDS and the ICI interfaces, a lightweight 'adaptation layer' is used. This layer translates CDS calls via ICI into commands that are compatible with the device firmware. Notably, should a different CDS be utilized, only the adapter layer requires redevelopment, ensuring flexibility and ease of integration.

Listing 1 provides a practical demonstration of the adapter layer in action. Here, the AdaptedController object implements the necessary CDS controller interface, ICdsController, by orchestrating calls to the matching functions to the IciController object. This instance of

IciController is obtained via the Ici subsystem, showcasing the integral role of the adapter layer in facilitating communication between the CDS and ICI.



Listing 2. Installing and using PythonNet using Windows 10.

In many instances, CDS systems are developed using .NET languages. However, the utilization of .NET wrappers facilitates the accessibility of .NET libraries across various programming languages, including (unmanaged) C++, by leveraging Component Object Model (COM), an older standard for application interface interaction. Furthermore, through tools like PythonNET on Windows, developers can instantiate .NET objects and invoke their functions within the Python language. This is shown in Listing 2.

Furthermore, .NET ran, until recently, only on the Windows platform. With the introduction of .NET Core it was possible to use the same compiled libraries on other platforms than Windows, such as Linux.

Implementation

To ensure the library's compatibility with Linux, the first and foremost requirement was to target .NET Core or .NET 6+ instead of the .NET Framework. While the .NET Framework is exclusive to Windows, .NET Core and subsequent versions like .NET 6+ are cross-platform, supporting Linux and macOS in addition to Windows. This is illustrated in Listing 3, where netstandard2.0 is used as the target; netstandard2.0 is compatible with .NET Core and .NET 6+.

Listing 3. Adding netstandard2.0 as target.

Another requirement was ensuring that any referenced assemblies were also compatible. Any P/Invoke calls, which are used to call native functions from unmanaged libraries, or calls to the Win32 API, had to be removed or replaced with cross-platform alternatives.

In addition to cross-platform compatibility, the data types used in the .NET assembly needed to be compatible with those in Python. Therefore, common types such as strings, integers, and arrays, which can be easily marshaled between .NET and Python, were utilized. Types that are difficult to handle, such as DateTime, were avoided to prevent compatibility issues.

Benefits and Use Cases

By targeting .NET Core or .NET 6+, the ICI library can operate on multiple operating systems, including. This cross-platform support broadens the potential user base and allows integration into diverse environments. The use of the ICI library abstracts the communication protocols required to interface with Spark Holland devices. This simplification enables developers to focus on higher-level application logic and functionality rather than dealing with low-level device communication details.

Third parties (for example OEM customers) can benefit from this cross-platform approach by integrating Spark Holland devices into their products. Developing device drivers and control software that work across different operating systems ensures broader market reach.

Conclusion

In summary, integrating the Spark Holland library (ICI) with Python on Linux using .NET Core/.NET 6+ represents a practical and efficient solution for controlling laboratory instruments.

For third-party developers and customers, this solution offers an efficient way to integrate Spark Holland devices into their applications.

Future Work

Moving forward, there are several opportunities for further exploration and development in this area. One avenue is to extend the compatibility of the ICI library with more programming languages, and with MacOS.

We also aim to improve documentation and provide SDK's to assist developers in leveraging the full potential of the ICI library.

References

Python.NET homepage,
https://pythonnet.github.io/

Python.NET GitHub home,
https://github.com/pythonnet/pythonnet

Install .NET on Linux,
https://learn.microsoft.com/en-us/dotnet/core/install/linux-scripted-manual#manual-install