Cape Software is pleased to offer its Rockwell Automation Compute Module (RACM) Software Development Kit (SDK) (i.e., RACM Cape SDK). RACM Cape SDK is a cross-platform, multi-language, remotable wrapper for Rockwell Automation’s ControlLogix® Compute Module.

The RACM SDK wraps the low-level C language API provided by Rockwell Automation with a server process that allows users to implement multiple clients, written in any of the supported languages across both the Windows and Linux operating systems.

Cape’s core technology is written in portable C++ code, which provides a common platform for the server process that runs on the Compute Module. Any functionality to be developed by the user can be added to client programs, which can be in the form of normal applications or services.

RACM Cape SDK provides a common C++ client object, which is then made available to the four (4) supported languages. The SDK user can program in any of the supported languages to build their respective target application. This provides the exact same behavior in each of the supported languages.

RACM Cape SDK benefits

- **Access**
  Use other languages:
  - Python
  - Perl
  - .NET
  - C++

- **Power**
  Run multiple applications on the Compute Module as separate apps.

- **Robust**
  Separating the functionality on the Compute Module in different client applications improves robustness.

- **Flexibility**
  Develop programs on your local development box without installing development software on the target Compute Module.

- **Easier integration** into an automated build and continuous integration frameworks.
RACM Cape SDK Supported Languages

RACM Cape SDK allows users to implement multiple clients written in four (4) supported languages across both the Windows and Linux operating systems. In each language, the end user will instantiate a client object which will communicate with the Compute Module. While the typically use case is expected to have the developed program communicate with the local Compute Module, this is not restricted by the SDK. The user program should instantiate one client object for each simultaneous connection to a Compute Module.

The following are examples for the various supported languages:

**C++**

In the C++ language, the SDK user will instantiate an RACM Client object, which has methods similar to the functions exposed by the native Rockwell API. This object has methods that match up with the methods of the native API very closely.

```cpp
int main(int argc, char *argv[]) {
  int result = 0;
  RACM_Client cli;
  char dispString[10];
  cli.init(argc, argv);
  result = cli.Connect("racm_cape1bcm");
  result = cli.SetED(1, 0);
  result = cli.GetED(1, &ledstate);
  print("%d", ledstate);
  result = cli.SetED(1, 1);
  result = cli.GetED(1, &ledstate);
  print("%d", ledstate);
  result = cli.SetDisplay("--");
  result = cli.GetDisplay(&dispString);
  print("%s", dispString);
  result = cli.SetDisplay("Cape");
  result = cli.GetDisplay(&dispString);
  print("%s", dispString);
  cli.Disconnect();
  return 0;
}
```

**Python**

RACM Cape SDK’s Python implementation provides a Python module that has similar methods to the API. However, the data types are more natural Python data types. Complex return values are returned from the methods by either dedicated objects or dictionary types. Instead of using return values to indicate success or failure, the Python implementation will throw a RuntimeException with the (failing) return value embedded in the exception.

```python
def main():
    client = RACM_Client()
    result = client.Connect("racm_cape1bcm")
    result = client.SetED(1, 0)
    result = client.GetED(1)
    print(result)
    result = client.SetED(1, 1)
    result = client.GetED(1)
    print(result)
    result = client.SetDisplay("--")
    result = client.GetDisplay()
    print(result)
    result = client.SetDisplay("Cape")
    result = client.GetDisplay()
    print(result)
    client.Disconnect()
```

**.NET**

The .NET language binding is implanted as a .NET assembly. As with the other languages, the assembly handles the communication to the server and uses .NET style data within the structures needed by the API.

**Perl**

RACM Cape SDK’s Perl interface is implemented by a Perl Module.

For more information, please contact us at:

**Cape Software Inc.**
17325 Park Row Drive
Houston, TX 77084

**Wood Intelligent Operations**
17325 Park Row
Houston, TX 77084

**Contact Details:**
Sales or Info: 1.281.600.3637
Tech Support: 1.281.362.1950
www.capesoftware.com or www.woodplc.com/vplink/
Email: VPLink@woodplc.com

Cape Software Inc., a wholly-owned Wood company
Cape Software is an experienced provider of products and engineering services that assist with factory acceptance testing (FAT), application logic validation and documentation, automated SIS proof testing and operator training.

Cape Software offers VP Link that is used to simulate the field devices during the testing phase of process control systems. Our VP Link software is routinely used for both control systems testing and operator training. When VP Link is connected to a Distributed Control System (DCS), Safety Instrumented System (SIS), or Programmable Logic Controller (PLC), the logic will react as though it is on a live process. Please contact us to find out more.