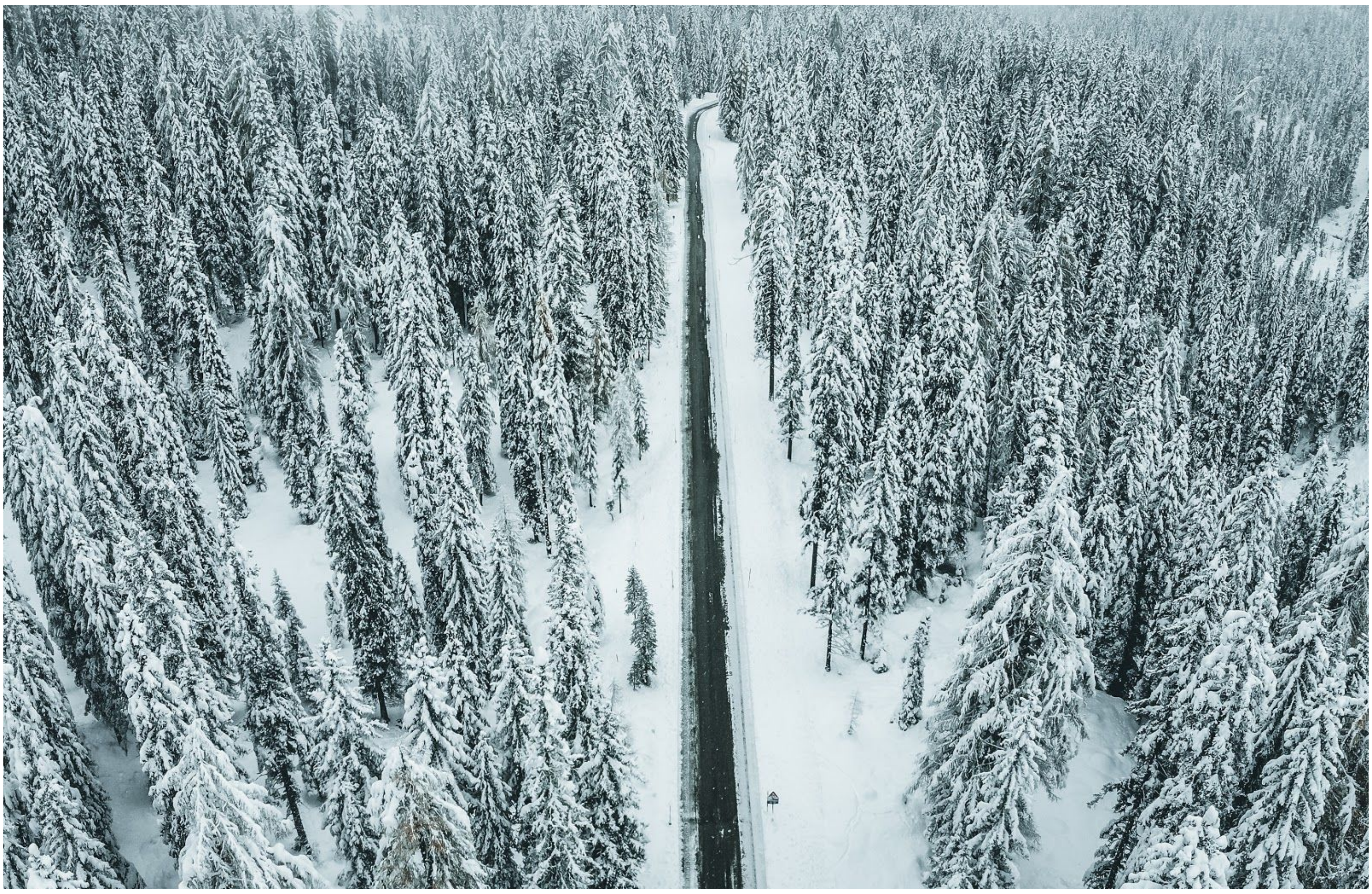
Development Guide

MOJA FLINT IMPLEMENTATION

Version 1.0 ● 30 March 2020



www.moja.global

Contents

Introduction **6**

Requirements **7**

- Hardware Requirements 7
- Software Requirements 8

Environment Preparation **11**

- Hardware Preparation 11
- Operating System Preparation 12
- Tools Installation 16
- Libraries Installation 30
- Environmental Variables Setup 31

Module Systems Development **33**

- Introduction 33
- Scope 33
- Steps 33
  - Step 1: Develop Database Tables 34
  - Step 2: Develop Entity Objects 35
  - Step 3: Develop Modules 36

Annex A : Hardware Audits Reference **38**

Annex B : HW Configurations Reference **41**

Annex C : OS Audits Reference **42**

Annex D : OS Configurations Reference **45**

Annex E: FLINT Concepts **48**

Annex F : Forest Module Database Tables **53**

Appendix 1 : Basic Dependencies **69**

Appendix 2 : Core Dependencies **73**

Appendix 3: Environmental Variables **75**

Appendix 4: Keys For Accessing BIOS settings 76

Abbreviations 77

References 78

Introduction

The FLINT is an open-source C++ platform that provides tools to integrate multiple data types (including remote sensing) with FLINT-compatible modules to produce spatially-explicit calculations of greenhouse gas (GHG) emissions and other variables.

This document provides a step by step guide for setting up the Moja FLINT development environment followed by steps for developing modules for it.

Requirements

This chapter highlights the hardware and software requirements for building Moja FLINT Libraries:

Hardware Requirements

a)	Recommended Hardware Specifications	
01.	Processor	Intel Core i7, Minimum, with Virtualization Support
02.	RAM	16GB, Minimum
03.	Hard Drive	1TB, Minimum

Refer to the following guides to audit how your hardware stacks against these recommendations:

- Annex A1 : Check Processor Capacity
- Annex A2 : Check RAM Capacity
- Annex A3 : Check Hard Drive Capacity
- Annex A4 : Check Support for Virtualization

Software Requirements

a)	Recommended Operating Systems	
01.	Windows 10 Pro	Latest Version
02.	Windows 10 Enterprise	Latest Version

Refer to the following guides to audit how your Operating System stacks against these recommendations:

- Annex C1 : Check Windows Version Edition
- Annex C2 : Check Windows Version Build Number

Please note that the information on the latest build of Windows 10 can be obtained from this page:

- https://support.microsoft.com/en-us/help/4464619/windows-10-update-history

b) Required Tools

01.	CMake	Latest Version
02.	Docker	Latest Version
03.	GIT	Latest Version
04.	Notepad++	Latest Version
05.	PostgreSQL	Latest Version
06.	Python 3	Latest Version
07.	TortoiseGit	Latest Version
08.	QGIS	Latest Version
09.	SQLiteStudio	Latest Version
10.	Visual Studio	2019 - Community
11.	Windows SDK	8.1

c) Required Libraries

01.	Boost	Version 1.63.0
02.	Eigen	Version 3.3.3
03.	Moja	Latest Version
04.	OpenSSL	Version 1.1.0
05.	POCO	Version 1.7.7
06.	SQLite Amalgamation	Version 3170000
07.	Turtle	Version 1.3.0

Environment Preparation

This chapter highlights how to prepare the environment for building Moja Base Libraries.

Hardware Preparation

a)	**Firmware Virtualization**
01.	Check if the Firmware Virtualization is enabled
02.	Enable it if not

The following guides have been added as references for carrying out the above task:

- Annex A5 : Check Firmware Virtualization Enablement Status
- Annex B1 : Enable Firmware Virtualization

Operating System Preparation

a)	Windows Version
01.	Check if the Windows version is the latest
02.	Update it if not

The following guides have been added as references for carrying out the above task:

- Annex C1 : Check Windows Version Edition
- Annex C2 : Check Windows Version Build Number
- Annex D1 : Update to the latest version of Windows 10

Please note that the information on the latest build of Windows 10 can be obtained from this page:

- https://support.microsoft.com/en-us/help/4464619/windows-10-update-history

b)	Administrative Rights
01.	Check if the logged in user account has administrative rights
02.	Switch to an account that has administrative rights if not

The following guide has been added as a reference for carrying out the above task:

- Annex C4 : Check whether a user account has administrative privileges

c) Account Password

01.	Check if the logged in user account has a password
02.	Set it if not

The following guides have been added as a references for carrying out the above task:

- Annex C5 : Check whether a user account has a password
- Annex D2: Add a password to a user account

d) Windows Hyper-v Features

01.	Check whether Windows Hyper-V Features have been turned on
02.	Turn them on if not

The following guides have been added as a references for carrying out the above task:

- Annex C6 : Check whether Windows Hyper-V Features have been turned on
- Annex D3 : Turn on Windows Hyper-V Features

e)	**Port 445**
01.	Check whether Port 445 is open for TCP connections
02.	Open it if not

The following guides have been added as a references for carrying out the above task:

- Annex C7 : Check if port 445 is open for TCP connections
- Annex D4 : Open port 445 for TCP connections

Tools Installation

CMake

a)	**Pre Installation**
01.	Go to https://cmake.org/download/
02.	Download the latest CMake Binary Distribution for Windows

b)	**Steps**
⚠	The following installation steps were written with reference to CMake 3.17.0
01.	Right click the CMake installer and select **Install**
02.	Click **Next** to confirm that you want to proceed with the installation
03.	Acknowledge the license terms and Click **Next**
04.	Optionally select the second option to the add **CMake to the system PATH for all users**
05.	Optionally check the last option to create a **CMake Desktop Icon**
06.	Click **Next** to proceed
07.	Leave the install path unchanged to install CMake in the default location
08.	Click **Next** to proceed
09.	Click **Install** to begin the installation
10.	Click **Finish** to exit the installation

Docker

a)	**Pre Installation**
01.	Complete the hardware preparation instructions as described earlier
02.	Complete the operating system preparation instructions as described earlier
03.	Go to https://www.docker.com/products/docker-desktop
04.	Download the Docker Desktop installer for Windows [1]

b)	**Steps**
⚠	The following installation steps were written with reference to Docker Desktop 2.2.0.3
01.	Right click the Docker Desktop installer and select **Run as administrator**
02.	Wait for Docker Desktop to download the required packages
03.	Leave the first checkbox checked to add a Docker Desktop shortcut to your desktop
04.	Click **OK** to proceed
05.	Wait for Docker Desktop to unpack its files and install
06.	Click **Close** to exit the installation

[1] This step will ask you to sign-in into your **docker hub** account. If you don't have one, please sign-up up for free here: https://hub.docker.com/signup

c) Configuration

⚠	The following configuration steps were written with reference to Docker Desktop 2.2.0.3
01.	Double click the **Docker Desktop Shortcut** to start it
02.	Wait for Docker Desktop to notify you that its up and running and then proceed to the next step
03.	Go to the System Tray [2] and click the Docker Desktop icon [3]
04.	Select **Settings** on the pop-up menu
05.	Select the **Resources** menu on the Settings Window
06.	Click the **Advanced** Resource subcategory if not currently selected
07.	Increase the **Memory** available to Docker to at least 4GB
08.	Click the **File Sharing** Resource subcategory
09.	Select drive **C:\** as the local drive you want to be available to your containers
10.	Click **Apply & Restart** to save the changes
11.	Close the settings window after Docker Desktop successfully restarts

[2] The System Tray is another name given to the Notification Area found at the right-side of the Windows Taskbar.
[3] If you cannot see the Docker Desktop icon at first glance, try looking for it in the pop-up drawer of the System Tray

Git

a)	**Pre Installation**
01.	Go to https://git-scm.com/downloads
02.	Download the latest Git binary release for Windows

b)	**Steps**
⚠	The following installation steps were written with reference to Git 2.26.0
01.	Right click the Docker Desktop installer and select **Run as administrator**
02.	Click **Install** to acknowledge the license terms and carry out the installation

c)	**Configuration** [4]
⚠	The following configuration steps were written with reference to Git 2.26.0
01.	Open the Windows 10 search tool
02.	Search for **Windows PowerShell**, open it and use it to execute the commands that follow (i) git config --global user.name "<your_user_name>" [5] (set your global username) (ii) git config --global user.email "<your_email@someservice.com>" [6] (set your global email)

[4] These steps assume that you have a Github Account. If you don’t, please sign up here: https://github.com/join
[5] Replace the content in the angular bracket, including the bracket itself, with your github username
[6] Replace the content in the angular bracket, including the bracket itself, with your github email

Notepad++

a)	**Pre Installation**
01.	Go to https://notepad-plus-plus.org/downloads/
02.	Download the latest Notepad++ release for Windows

b)	**Steps**
⚠	The following installation steps were written with reference to Notepad++ 7.8.5
01.	Right click the Notepad++ installer and select **Run as administrator**
02.	Leave **English** as the selected language and click **OK**
03.	Click **Next** to confirm that you want to proceed with the installation
04.	Acknowledge the license terms and Click **Next**
05.	Leave the install path unchanged to install Notepad++ in the default location; click **Next**
06.	Leave the selected features unchanged to install the default components; click **Next**
07.	Click **Install** to carry out the installation
08.	Click **Finish** to complete the installation

PostgreSQL

(a)	**Pre Installation**
01.	Go to https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
02.	Download the latest PostgreSQL release for Windows

(b)	**Steps**
⚠	The following installation steps were written with reference to PostgreSQL 12.2
01.	Right click the PostgreSQL installer and select **Run as administrator**
02.	Click **Next** to confirm that you want to proceed with the installation
03.	Leave the install path unchanged to install PostgreSQL in the default location; click **Next**
04.	Leave the selected features unchanged to install the default components; click **Next**
05.	Leave the data location unchanged to store PostgreSQL data in the default location; click **Next**
06.	Enter and repeat your preferred password; click **Next**
07.	Leave the suggested port number unchanged to allow PostgreSQL to use its default port; click **Next**
08.	Leave the default locale selected to allow PostgreSQL to use it; click **Next**
09.	Click **Next** to confirm that you want to use the selected settings for the installation
10.	Click **Next** to carry out the installation; click **Finish** and proceed to the post installation tasks

(c) Post Installation

⚠	The following configuration steps will be carried out in **Stack Builder**
01.	Select the recently installed PostgreSQL server as the target for the Stack Builder configuration
02.	Click **Next**
03.	Expand the **Spatial Extensions** category
04.	Select **PostGIS 3.0 Bundle for PostgreSQL 12**
05.	Click **Next**
06.	Leave the install path unchanged to install the bundle in the default location; click **Next**
07.	Wait for the download of the installation files then click **Next** to start the installation
08.	Acknowledge the license terms
09.	Leave the selected features unchanged to install the default component (PostGIS); click **Next**
10.	Leave the install path unchanged to install the PostGIS bundle in the default location; click **Next**
11.	Click **Yes** to register GDAL_DATA environmental variable
12.	Click **Yes** to register POSTIGIS_GDAL_ENABLED_DRIVERS environmental variable
13.	Click **Yes** to register POSTIGISL_ENABLE_OUTPUTDB_RASTERS environmental variable
14.	Click **Close** to exit the installer
15.	Click **Finish** to exit the Stack Builder

Python 3

(a)	**Pre Installation**
01.	Go to https://www.python.org/downloads/
02.	Download the latest Python release for Windows

(b)	**Steps**
⚠	The following installation steps were written with reference to Python 3.8.2
01.	Right click the Python installer and select **Run as administrator**
02.	Select **Customize installation** to custom-install Python
03.	Leave the default Optional Features selected and click **Next**
04.	Leave the default Advanced Options selected
05.	Change the install location to **C:\Python38**
06.	Click **Install**
07.	Disable the 260 characters max path length limit
08.	Click **Close** to exit the installer

TortoiseGit

(a)	**Pre Installation**
01.	Install GIT As previously described
02.	Go to https://tortoisegit.org/download/
03.	Download the latest TortoiseGit release for Windows

(b)	**Steps**
⚠	The following installation steps were written with reference to TortoiseGit 2.10.0.2
01.	Right click the TortoiseGit installer and select **Install**
02.	Click **Next** to confirm that you want to proceed with the installation
03.	Click **Next** to acknowledge the license terms and proceed with the installation
04.	Leave the default SSH Client selected; click **Next**
05.	Leave the default features and install path unchanged; click **Next**
06.	Click **Install** to begin the installation
07.	Leave the **Run first wizard** option checked
08.	Click **Finish** and proceed to the post installation tasks

(c) Post Installation

⚠	The following configuration steps were written with reference to TortoiseGit 2.10.0.2
01.	Leave English as the default selected language and click **Next**
02.	Click **Next** to confirm that you want to proceed with the configuration
03.	TortoiseGit should automatically detect the Git executable; [7] click **Next**
04.	TortoiseGit should automatically detect your username and email; [8] click **Next**
05.	Leave the authentication and credential store with the default settings
06.	Click **Finish** to exit the installer

[7] This will only happen if you installed Git prior to installing TortoiseGit. Otherwise, you will have to pause, install Git and then update its path yourself.
[8] This will only happen if you carried out the post installation steps after installing Git as previously described. Otherwise, you will have to manually carry out this step.

QGIS

(a)	**Pre Installation**
01.	Go to https://qgis.org/en/site/forusers/download.html
02.	Download the latest QGIS Standalone Installer for Windows - from OSGeo4W packages

(b)	**Steps**
⚠	The following installation steps were written with reference to QGIS 3.12.1 'București'
01.	Right click the QGIS installer and select **Run as administrator**
02.	Click **Next** to confirm that you want to proceed with the installation
03.	Acknowledge the license terms
04.	Leave the install path unchanged to install QGIS in the default location; click **Next**
05.	Leave the selected features unchanged to install the default components; click **Install**
06.	Click **Finish** to complete the installation

SQLiteStudio

(a)	**Pre Installation**
01.	Go to https://github.com/pawelsalawa/sqlitestudio/releases
02.	Download the latest binary release of SQLiteStudio for Windows

(b)	**Steps**
⚠	The following installation steps were written with reference to SQLiteStudio 3.2.1
01.	Right click the SQLiteStudio installer and select **Run as administrator**
02.	Click **Next** to confirm that you want to proceed with the installation
03.	Leave the install path unchanged to install SQLiteStudio in the default location
04.	Leave the other settings as they are
05.	Click **Next**
06.	Leave the selected components unchanged - to install Qt and SQLiteStudio; click **Next**
07.	Click **Install** to carry out the installation
08.	Uncheck the **Run SQLiteStudio** option and Click **Finish** to complete the installation

Visual Studio

(a)	**Pre Installation**
02.	Go to https://my.visualstudio.com/Downloads [9]
03.	Type **Visual Studio Community 2019** in the **search downloads** field
04.	Press **Enter** and wait for Microsoft to retrieve the desired Visual Studio versions
05.	**Download** the installer of the latest version of Visual Studio Community 2019

(b)	**Steps**
⚠	The following installation steps were written with reference to Visual Studio Community 2019 (V 16.5)
01.	Right click the Visual Studio installer and select **Run as administrator**
02.	Click **Continue** and wait for the installer to prepare for the installation
03.	Check the **Desktop development with C++** option under the **Workloads** tab
04.	Check the **GitHub extension for Visual Studio** option under the **Individual components** tab [10]
05.	Click **Install** in the bottom right corner of the screen
06.	Wait for the installation to complete then restart the computer

[9] This will require you to sign in. You can sign in with any of your Microsoft accounts credentials e.g Skype credentials. You can alternatively sign in with your Github credentials

[10] The **GitHub extension for Visual Studio** option is located under the **Code tools** category

Windows SDK

(a)	**Pre Installation**
01.	Go to https://developer.microsoft.com/en-us/windows/downloads/sdk-archive/
02.	Download Windows 8.1 SDK installer

(b)	**Steps**
⚠	The following installation steps were written with reference to Windows 8.1 SDK
01.	Right click the Windows SDK installer and select **Run as administrator**
02.	Leave the first option selected to install the SDK in the current computer
03.	Leave the install path unchanged to install the SDK in the default location
04.	Click **Next**
05.	Leave the default option selected in the CEIP program participation request and click **Next**
06.	Acknowledge the license terms
07.	Leave the selected features unchanged to install the default component; click **Install**
08.	Click **Close** to complete the installation

Libraries Installation

Adding the libraries that the Moja FLINT Implementations depend upon into the local environment is a prerequisite for all development efforts. There are two ways you can go about doing it:c i) Build the libraries manually into the local environment or ii) build the libraries into the local environment via the vcpkg tool.

When installing third party libraries we recommend using the vcpkg tool to get it right the first time round. When installing the Moja FLINT Library, we recommend that you build it manually.

💡 **Important**

- See **Moja Base Libraries Build Guide** for instructions on how to build the third party libraries
- See **Moja FLINT Libraries Build Guide** for instructions on how to build the Moja libraries

Environmental Variables Setup

a)	**Add new system variables**
01.	Open the Windows 10 search tool
02.	Search for **Edit the system environment variables** and open it
03.	At the bottom-right corner of the open Advanced Tab, Click **Environment Variables**
04.	Under the System Variables section, add the following variables
(i)	Variable Name: BOOST_ROOT Variable Value:<path to Boost install dir> [11]
(ii)	Variable Name: BOOST_INCLUDEDIR Variable Value:<path to Boost include dir> [12]
(iii)	Variable Name: POCO_BASE Variable Value:<path to POCO library root> [13]

[11] e.g C:\Development\boost
[12] e.g C:\Development\boost\lib\boost-1_63\lib64-msvc-14.0>\lib\lib64-msvc-14.0
[13] e.g. C:\Development\poco-1.7.7-all

b)	**Update the Path variable**
01.	Open the Windows 10 search tool
02.	Search for **Edit the system environment variables** and open it
03.	At the bottom-right corner of the open Advanced Tab, Click **Environment Variables**
04.	Under the System Variables section, look for the PATH variable
05.	Append the following lines to it separated by semicolons

```
<path to your Git install dir>\cmd
<path to your CMake install dir>\bin
<path to your TortoiseGit install dir>\bin
%BOOST_INCLUDEDIR%
%POCO_BASE%\bin64
```

Module Systems Development

Introduction

This chapter goes through the process of developing and attaching a Module System to a FLINT Implementation Framework. It does so by examining what it takes to implement the classic Forest Module System that is commonly found attached in several FLINT Implementations. In doing so, it aims at revealing the typical steps that are required to develop and attach other Module Systems to a FLINT Implementation Framework.

💡 **Important**

It is important that the readers go through Annex E: FLINT Concepts to familiarize themselves with the Basic Concepts of the FLINT Framework such as Modules & Module Systems before moving on. This will bring greater clarity, not only to the introductory content of this chapter, but also to the steps that follow

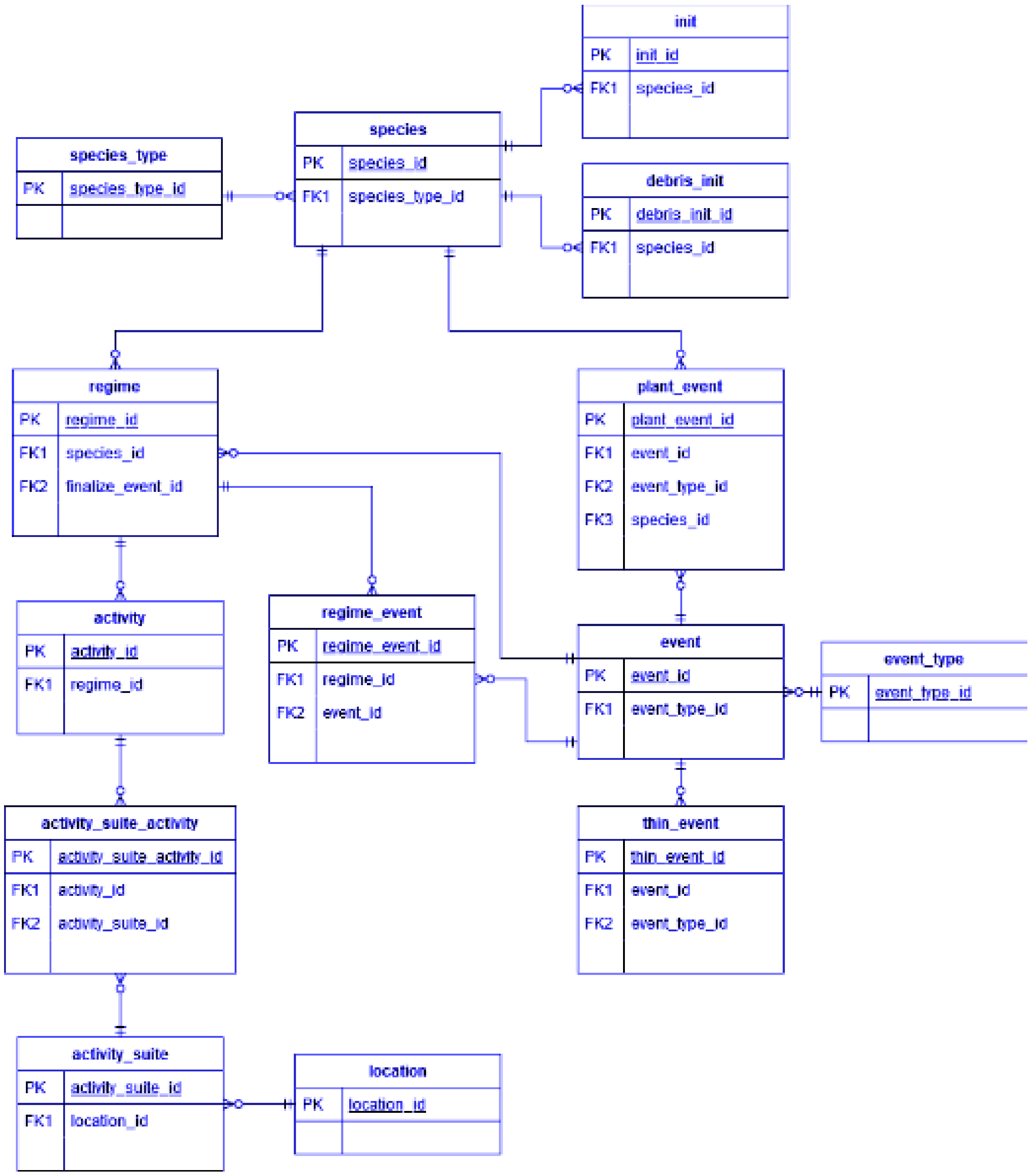
Scope

For brevity, we will work with a subset of the modules found in the classic Forest Module System. In particular we will consider only the functionality that relate to the implementation of a Module System that can be used to estimate emissions and removals from Forest Plantations.

Steps

Step 1: Develop Database Tables

The journey of a module system implementation typically starts with the design and implementation of its database schema. The diagram below outlines the key tables that are typically involved in the implementation of a Forest Module System. A detailed description of these tables has been included for your examination in Annex F : Forest Module Database Tables.


init
PK init_id
FK1 species_id
species
PK species_id
FK1 species_type_id
species_type
PK species_type_id
debris_init
PK debris_init_id
FK1 species_id
regime
PK regime_id
FK1 species_id
FK2 finalize_event_id
plant_event
PK plant_event_id
FK1 event_id
FK2 event_type_id
FK3 species_id
activity
PK activity_id
FK1 regime_id
regime_event
PK regime_event_id
FK1 regime_id
FK2 event_id
event
PK event_id
FK1 event_type_id
event_type
PK event_type_id
activity_suite_activity
PK activity_suite_activity_id
FK1 activity_id
FK2 activity_suite_id
thin_event
PK thin_event_id
FK1 event_id
FK2 event_type_id
activity_suite
PK activity_suite_id
FK1 location_id
location
PK location_id

Step 2: Develop Entity Objects

The journey of the module system implementation continues with the mapping of data stored in the database tables (and configuration files) back to c++ objects. This is to allow for convenient interchange and modification of data during run-time and / or writing back of the modified data back to the database.

To do this, entity classes that logically represent the database tables are defined. Data objects are then created from them during run-time. Below is an example of such a class: TreeSpecies, as defined in a typical Forest Module System:

```
class SLEEK_FSR_API TreeSpecies : public flint::IFlintData {

  Public:

   int species_id;
   std::string name;

   double carbon_frac_stem;
   double carbon_frac_branch;
   ...

}

class SLEEK_FSR_API PlantationSpecies : public TreeSpecies {


   void configure(DynamicObject config, ..) override {
      species_id = config["species_id"];
      name = config["name"].extract<const std::string>();
      carbon_frac_stem = config["carbon_frac_stem"];
      carbon_frac_branch = config["carbon_frac_branch"];
      ...

   }

   DynamicObject exportObject() const override {
      DynamicObject object;

      object["name"] = name;

      object["carbon_frac_stem"] = carbon_frac_stem;
      object["carbon_frac_branch"] = carbon_frac_branch;
      ...
   }

}
```

- The Tree species' class encapsulates the characteristics of a tree species as defined in the data repository. This includes all the species carbon partitions, growth properties, allocations and increments defined in **Annex F2 : SPECIES** that relate to both Natural Forest and Plantation Forest Species

- Any property that applies specifically to natural forests or plantation forests is provided in the child classes - aptly named plantation or natural forest species - that extend this class

Step 3: Develop Modules

Now that we have access to state variables and data, the journey of the module system implementation moves to the actual implement of the modules. Modules typically listen to notifications from the FLINT core system (Unit Controller) and perform calculations; then they return information that specifies what updates need to be performed on state variables and carbon pools.

Below is an example of a Forest Module System Growth Module followed by a description of the functionality carried out in its key methods:

```
class SLEEK_FSR_API PlantationGrowthModule : public moja::flint::ModuleBase {

  public:

   //Constructors / Destructors

   void configure(const DynamicObject& config) override {
     ...
   }

   void subscribe(NotificationCenter& notificationCenter) override {
     ...
   }

   void onLocalDomainInit() override {
     ...
   }

   void onTimingInit() override {
     ...
   }

   void onTimingStep() override {
     ...
   }

   ...
}
```

- configure() is typically called by the FLINT core system (Unit Controller) to provide the module the opportunity to bootstrap itself at the start of a simulation. Data and variable initialization are thus typically performed within this method. The PlantationGrowthModule usually does nothing at this stage.

- subscribe() allows the module to express interest in one or more events that occur during a simulation, and receive notifications only when those events occur. The PlantationGrowthModule for example is interested in local domain initialization, timing init and time step notifications only.

- onLocalDomainInit() handles local domain initialization events. For the PlantationGrowthModule this includes initializing the pool and variable values that will be used during the simulation.

- onTimingInit() handles timing initialization events. For the PlantationGrowthModule this includes checking whether a plantation exists, obtaining the age of the plantation and then getting the details of the tree species in that plantation.

- onTimingStep() handles timing step events. For the PlantationGrowthModule this includes includes updating the tree properties and submitting moves

Annex A : Hardware Audits Reference

A1 : Check Processor Capacity

01.	Open the Windows 10 search tool
02.	Search for the **System Information** tool and open it
03.	Select **System Summary** menu on the **System Information** window
04.	Look for the **Processor** specification on the right pane

A2 : Check RAM Capacity

01.	Open the Windows 10 search tool
02.	Search for the **System Information** tool and open it
03.	Select the **System Summary** menu on the **System Information** window
04.	Look for the **Physical Memory** specifications on the right pane

A3 : Check Hard Drive Capacity

01.	Open the Windows 10 search tool
02.	Search for the **System Information** tool and open it
02.	Expand the **Components** category in the **System Information** window
03.	Expand the **Storage** subcategory under the **Components** category
04.	Click the Disks subcategory under the **Storage** subcategory
05.	Look for the **Size** specifications under the disk descriptions [14]

A4 : Check Support for Virtualization

01.	Open the Windows 10 search tool
02.	Search for the **Task Manager** tool and open it
03.	Open the **Performance** tab on the opened window [15]
04.	Look for a line that says **“Virtualization: (En/Dis)abled”** on the bottom-right side of the opened tab

[14] Watch out for multiple disk descriptions with different sizes when multiple Hard Drives are present
[15] You might have to click on **More details** to see this tab the very first time you open Task Manager

A5 : Check Firmware Virtualization Enablement Status

01.	Open the Windows 10 search tool
02.	Search for the **Task Manager** tool and open it
03.	Open the **Performance** tab on the opened window [16]
04.	Look for a line that says **“Virtualization: Enabled”** on the bottom-right side of the opened tab

[16] You might have to click on **More details** to see this tab the very first time you open Task Manager

Annex B : HW Configurations Reference

B1 : Enable Firmware Virtualization

01.	Restart the PC
02.	Press the key required to enter BIOs (See Appendix 4: Keys For Accessing BIOS settings)
03.	Navigate to either the **Advanced**, **Security** or the **Systems Configurations** tab
04.	Select **Virtualization** or **Virtualization Technology** and then press the **Enter** key [17]
05.	Select **Enabled** and then press the **Enter** key
06.	Press the **F10** key then select **Yes** and press the **Enter** key to save the changes and **Reboot** [18]

[17] On some Lenovo PCs, the Virtualization option will be found buried one level deeper under a **CPU Setup** option
[18] On some Sony PCs, you will need to navigate to a dedicated **Exit** tab to save changes and exit

Annex C : OS Audits Reference

C1 : Check the Windows Version Edition

01.	Open the Windows 10 search tool
02.	Search for the **System Information** tool and open it
03.	Select the **System Summary** menu on the **System Information** window
04.	Look for the **OS Name** specification on the right pane

C2 : Check the Windows Version Build Number

01.	Open the Windows 10 search tool
02.	Search for the **System Information** tool and open it
03.	Select the **System Summary** menu on the **System Information** window
04.	Look for the **Version** specification on the right pane

C3 : Check for the latest Windows Operating System

01.	Open https://en.wikipedia.org/wiki/List_of_Microsoft_Windows_versions
02.	Look for the latest Windows **Version, Edition** and **Build Number**

C4 : Check whether a user account has administrative privileges

01.	Open the Windows 10 search tool
02.	Search for the **Manage your account** tool and open it
03.	Look for the word "**Administrator**" underneath the **account** name

C5 : Check whether a user account has a password

01.	Open the Windows 10 search tool
02.	Search for the **Manage your account** tool and open it
03.	Click the **Sign-in options** on the left pane of the opened window
04.	Scroll down to the **Password** section on the right pane of the opened window
05.	Look for a statement that says "**Sign in with your account’s password**" underneath it

C6 : Check whether Windows Hyper-V features are turned on

01.	Open the Windows 10 search tool
02.	Search for the **Turn Windows features on or off** tool and open it
03.	Locate the Hyper-V section and find out if it’s checked

C7 : Check if port 445 is open for TCP connections

01.	Open the Windows 10 search tool
02.	Search for the **Windows Defender Firewall** tool and open it
03.	Click **Advanced settings** on the left pane of the **Windows Defender Firewall** window
04.	Click the **Inbound Rules** category on the left pane of the newly popped up window
05.	Locate the **Local Port** column on the newly opened **Inbound Rules** table
06.	Scroll down this **Local Port** column and see whether there's a TCP entry for port 445
07.	Click the **Outbound Rules** category on the left pane of the newly popped up window
08.	Locate the **Remote Port** column on the newly opened **Outbound Rules** table
09.	Scroll down this **Remote Port** column and see whether there's a TCP entry for port 445

Annex D : OS Configurations Reference

D1 : Update to the latest version of Windows 10

01.	Go to https://www.microsoft.com/en-us/software-download/windows10
02.	Click the **Update now** button to download the **Windows 10 Update Assistant**
03.	Right click the downloaded **Windows 10 Update Assistant** and select **Run as administrator**
04.	Click **Update Now** on the newly opened window
05.	Click **Next** after the PC is ascertained as being compatible with the update
06.	Click **Minimise** to optionally have the update run in the background
07.	Click **Restart now** to restart your PC when the update is complete

D2 : Add a password to a user account

01.	Open the Windows 10 search tool
02.	Search for the **Manage your account** tool and open it
03.	Click the **Sign-in options** on the left pane of the opened window
04.	Scroll down to the **Password** section on the right pane of the opened window
05.	Click the **Add** button underneath it
06.	Enter the password and password hint details and click **Next**
07.	Click **Finish**

D3 : Turn on Windows Hyper-V features

01.	Open the Windows 10 search tool
02.	Search for the **Turn Windows features on or off** tool and open it
03.	Locate the Hyper-V section
04.	Check it and click **OK**
05.	Click **Restart now** to finish installing the requested changes

D4 : Open port 445 for TCP connections

01.	Open the Windows 10 search tool
02.	Search for the **Windows Defender Firewall** tool and open it
03.	Click **Advanced settings** on the left pane of the **Windows Defender Firewall** window
04.	Click the **Inbound Rules** category on the leftmost pane of the newly popped up window
05.	Click the **New Rule** option on the rightmost pane of the newly popped up window
06.	Select **Port** as the type of rule to be created
07.	Click **Next**
08.	Select **TCP** as the protocol of the rule to created
09.	Select **Specific local ports** and enter **445** as the port that the rule should be apply to
10.	Click **Next**
11.	Select **Allow the connection** as the action to take when a connection matches the conditions
12.	Check **Domain, Private and Public** to have the rule apply to each of these profiles
13.	Click **Next**
14.	Enter **Docker** as the name of the rule and click **Finish**
15.	Repeat Steps 04 to 14 for **Outbound Rules**

Annex E: FLINT Concepts

E1 :Simulation Units

A Simulation Unit is the basic unit for simulation by the FLINT. A Simulation Unit can represent a spatial area, such as a pixel or forest stand, or it can represent an emissions source, such as livestock. Where the Simulation Unit refers to a geographically referenced area, it is known as a Land Unit.

Within the databases and data-layers underpinning the FLINT, there are attributed values that describe the characteristics of each Simulation Unit. For example, for a Land Unit there may be information on the unit's area, land type, age of vegetation, species and carbon pools. The overall framework of FLINT manages the processing of Simulation Units over time. While Simulation Units are the basis of all simulations run in FLINT, they are rarely used for reporting purposes.

E2 : Pools

A pool is a reservoir within which something can be stored. The most obvious example is a carbon pool which is a reservoir into which carbon can be stored. Within FLINT, each pool is attributed a value, for example tonnes of carbon, and at each time step the FLINT can move stores from one pool to another using operations (discussed below). The fluxes between pools, including the atmosphere, are used to calculate changes in carbon stock and resulting emissions and removals.

Pools are fully configurable depending on the modules used or data available (e.g. harvested wood products). Since the emissions estimates will be reported to the UNFCCC, all carbon pools will need to be aggregated into the IPCC pool types (aboveground biomass, belowground biomass, deadwood, litter, soil (mineral and organic)) before they are reported. This can be done at the end of the FLINT run. The FLINT allows the developers to define other pool types in addition to carbon, such as water (for models with water balance models) and money (for economics modules).

E3 : Operations

An operation is a process within the FLINT that moves carbon stock between pools. Operations are defined within modules, and can reflect processes, such as growth, or events such as harvests, or fires, whether natural or human induced. For example, an operation reflecting a harvest event, moves plant material to products and debris pools, while a wildfire moves plant material to debris and atmospheric pools. The amount of stock moved during an operation is referred to as a 'flux'. Operations allow the FLINT to track changes in carbon stock through time, including fluxes into and out of pools.

FLINT uses operations to update values for Simulation Units and to record flux values in a flux table. For example, an operation reflecting plant growth can be applied to aboveground biomass pools to estimate the growth flux over time. FLINT has been designed to ensure the conservation of mass (and area) throughout the calculation process. This means that FLINT will only transfer carbon stock from one pool to another through known and valid pathways, ensuring the system is balanced so that the sum of all the fluxes is equal to the sum of the stock changes.

The FLINT tracks each operation and the resulting fluxes, and depending on which pools the flux relates to, they are classified in the flux table and the resulting carbon stock tables. For example, the output will be classified as ‘stem biomass’. The resulting information can then be used to calculate different characteristics of the Simulation Unit. For example, by summing all stock changes in aboveground, belowground biomass pools and net litter turnover, the Net Primary Production (NPP) can be calculated. Similarly, the sum of all fluxes from the dead organic matter and Soil Carbon pools to the atmosphere pools that are the result of natural decay processes, may be considered heterotrophic respiration (Rh). By subtracting Rh from NPP, Net Ecosystem Productivity (NEP) can be calculated.

In addition to heterotrophic respiration, other disturbances can cause loss of organic matter to the atmosphere. These disturbances also need to be tracked by disturbance type. NEP minus losses from disturbances is equal to Net Biome Production (NBP).

E4 : Processes

Processes continuously occur on every time-step (see below) of a simulation once a variable status is met. This includes processes that increase carbon stock in biomass, such as plant growth, as well as processes that decrease carbon stock in biomass, such as decomposition and turnover. When the variable of tree status is set to ‘True’, for example, the tree growth will occur. This is a ‘process’. Processes are coded for the FLINT as a module (see Modules description).

E5: Events

Events are operations that occur intermittently (rather than for every time step in a simulation) resulting in the movement of carbon from one pool to another. Events include natural and anthropogenic events including fire, harvesting, ploughing, and fertiliser application. Events are coded for the FLINT as a module (see Modules description).

E6: Timing

Estimating carbon stock change has an inherent temporal component, as it requires tracking the interactions between carbon pools through time. To track the interactions between carbon pools, it is necessary for them to be on a comparable temporal scale. Tracking changes through multiple pools, from multiple processes and events through time should be performed at timeframes that are appropriate for the pools that are being modelled. For example, an empirical forest growth module may provide annual growth data while the debris module may operate monthly and the soil carbon module daily. The temporal scale in FLINT is referred to as a time-step.

Time-steps are lengths of time over which operations are reported. It is only at the end of a time-step that carbon can be moved from one pool to another. For example, for carbon to move from pool A to B to C, it will take two time-steps [A>B][B>C]. Time-steps are used to reduce the processing requirements of the model. The finer the step (i.e. the shorter it is), the more processing is required for simulations. Because of time-steps, rather than continuous changes, there is a certain amount of 'graininess' in the output data of FLINT (Figure 1).

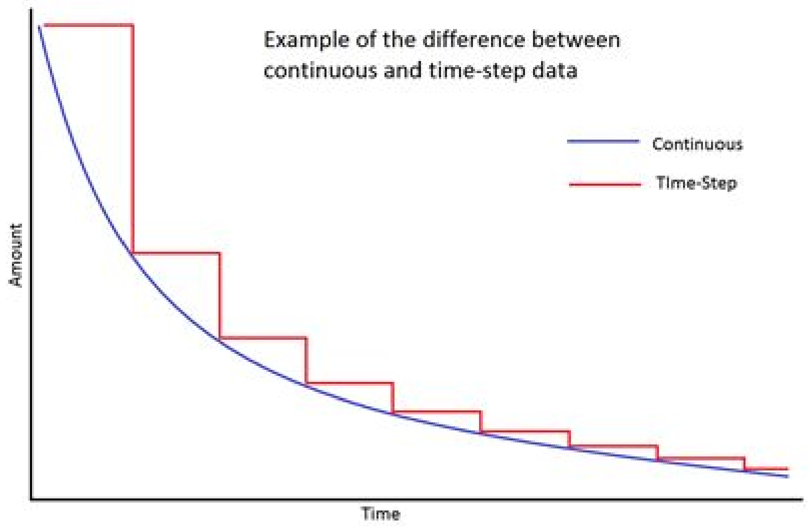


Figure 1: Stylised example of the differences between continuous and time-step data for a decay curve. The difference between the two is referred to as 'graininess', where the larger the time-step (i.e. the more time it covers) the more graininess the model has.

The ability of FLINT to control the timing and flow of inputs and outputs from processes and events that operate at different time steps without adjusting the modules themselves is one of its key features. To achieve this, modules are run at the time-step that they have been built and calibrated for. This minimises the graininess of the results without exacerbating error. Without this, simply running an annual model at daily time steps can lead to significant errors.

During a simulation, the FLINT will run at the finest time step of any input module or the output module. For example, if the soil module runs at a daily time-step, then the FLINT will run daily. The other modules will run at their native temporal resolution (e.g. annual for forest growth). The FLINT will handle the interpolation of low temporal resolution data to high resolution data by proportionally allocating module output. For example, if running at a daily time-step the annual module will have the total results divided between the number of days in the year. The allocation does not need to be linear and it can take other forms, for example exponential. The operations in the FLINT will ensure that any sub-time step information is reported as such in the flux summary tables and that FLINT calculates the appropriate summary statistics.

E7 : Modules

All operations, processes and events are managed through modules. A module is a self-contained set of operations that determine the state of, or change in, variables across a specified period for a single Simulation Unit in direct response to notifications from the FLINT core system (Unit Controller). Notifications can be triggered from events, or time-steps . For example, the empirical forest growth module includes all the operations required to simulate biomass accumulation in forests.

Each module reads (or is provided with) information about the current state variables and the data required to update the state variables, such as climate data or information about events to simulate. Each module then performs the required calculations and returns information about the updates to apply to each of the state variables and carbon pools. For state variables, such as age, it is possible for modules to return the updated value, but for all carbon pools and other fluxes the module returns an array (sparse matrix) of the proposed operations. This array will include the information about the source pool, the sink pool, and the amount (tC ha-1 per time step) of the flux. Module-specific metadata regarding units and time step size are also required. This information is all returned to the Unit Controller.

In terms of the FLINT programming framework, there are utility and calculation modules. The utility modules are generic utility modules, for example, for producing output tables. While calculation modules are often regionally or nationally specific. By using modules and separating the generic and specific, FLINT's framework remains highly flexible and new modules can be 'plugged in' as they are developed.

Modules are described by the functionality that they provide, such as the Forest Module or Crop Module. While there may be an overarching module system, such as the Forest Module, this may be made up of multiple modules, with each completing a specific purpose; (See Table Below):

Module System	Module
Forest Module	Forest Growth Module
	Forest Management Module
	Plantation Events Module
	Plantation Growth Module
	Plantation Management Module
	Forest Debris Module

A key module for all FLINT implementations is the Build Land Unit Module (BLUM). The BLUM defines the rules of developing the sequence of events and processes using both spatial data as well as database data. The rules within BLUM are configured based on the input data logic, and are typically restricted through source code to the key design decisions of a FLINT Implementation. In simple terms, the BLUM can be considered as a series of ‘if, then’ statements. For example, if land cover changes from X to Z, then run Y events, or if tree status equals true, then apply the Tree Growth module. The logic of these statements should be hard coded into the BLUM based on the decisions within the FLINT Implementation Handbook. As with most modules, the BLUM consists of multiple modules, each completing a specific computational process (See Table Below):

Module System	**Module**
Build Land Unit	Build land unit fsr module
	Build land unit spin-up module
	Build land unit transitions module

E8 : FLINT Implementation

A FLINT Implementation is a unique combination of datasets and modules that are attached to the FLINT. For the application of the FLINT within the SLEEK program for example, it is known as SLEEK-FLINT.

In summary, the FLINT tracks all ‘fluxes’ for a ‘simulation area’ through time. These fluxes are determined by processes and events (collectively known as ‘operations’). The type and magnitude of events and fluxes are determined by the user and the input data.

Annex F : Forest Module Database Tables

F1 : SPECIES_TYPE

Different forest species are usually grouped into two categories, natural forests and plantation forests. The natural forests represent the endemic forest types that are left to ‘natural processes’, while the plantation forests are endemic of introduced species, typically managed for timber. This table provides a way to identify if species type refers to a plantation or natural forest.

Columns

Column Name	Description
species_type_id	The primary key of the species type record
species_type	The name of the species type

F2 : SPECIES

Where a Forest Module is concerned, a species typically refers to a forest type, be is a monoculture or diverse forest type. That is, a species can refer to a forest type category, such as dry forest, a species group, such as Eucalyptus, or a species, such as Eucalyptus globulus. This table provides a way to detail the growth and partitioning for a species.

Columns

Column Name	Description
species_id	Unique ID for species (or forest Type)
species_type_id	Unique ID for species type (Plantation of natural forest)
carbon_frac_stem	Carbon fraction of stem wood
carbon_frac_branch	Carbon fraction of branch wood
carbon_frac_bark	Carbon fraction of bark
carbon_frac_leaf	Carbon fraction of leaves
carbon_frac_coarse_root	Carbon fraction of coarse roots
carbon_frac_fine_root	Carbon fraction of fine roots
turnover_frac_branch	Turnover fraction of branches
turnover_frac_bark	Turnover fraction of bark
turnover_frac_leaf	Turnover fraction of leaves
turnover_frac_coarse_root	Turnover fraction of coarse roots
turnover_frac_fine_root	Turnover fraction of fine roots
resistant_frac_stem	Resistant fraction of stem wood
resistant_frac_branch	Resistant fraction of branch wood
resistant_frac_bark	Resistant fraction of bark
resistant_frac_leaf	Resistant fraction of leaves
resistant_frac_coarse_root	Resistant fraction of coarse roots
resistant_frac_fine_root	rResistant fraction of fine roots
breakdown_frac_ decomposable_standing_ deadwood	Breakdown fraction of decomposable standing deadwood

breakdown_frac_ decomposable_downed_ deadwood	Breakdown fraction of decomposable downed deadwood
breakdown_ frac_decomposable_bark_litter	Breakdown fraction of decomposable bark litter
breakdown_ frac_decomposable_leaf_litter	Breakdown fraction of decomposable leaf litter
breakdown_ frac_decomposable_coarse_dead_ro ot	Breakdown fraction of decomposable coarse dead roots
breakdown_ Frac_decomposable_fine_dead_ root	Breakdown fraction of decomposable fine dead roots
breakdown_ Frac_resistant_standing_ deadwood	Breakdown fraction of resistant standing deadwood
breakdown_ Frac_resistant_downed_ deadwood	Breakdown fraction of resistant downed deadwood
breakdown_ frac_resistant_bark_litter	Breakdown fraction of resistant bark litter
breakdown_ frac_resistant_leaf_litter	Breakdown faction of resistant leaf litter
breakdown_ frac_resistant_coarse_dead_root	Breakdown faction of resistant coarse dead roots
breakdown_ frac_resistant_fine_dead_root	Breakdown fraction of resistant fine dead roots
atmospheric_ frac_decomposable_standing_ deadwood_breakdown	Fraction of biomass that breaks down to atmosphere from decomposable standing deadwood
atmospheric_ frac_decomposable_downed_ deadwood_breakdown	Fraction of biomass that breaks down to atmosphere from decomposable downed deadwood
atmospheric_ frac_decomposable_bark_litter_ breakdown	Fraction of biomass that breaks down to atmosphere from bark litter
atmospheric_ frac_decomposable_leaf_litter_ breakdown	Fraction of biomass that breaks down to atmosphere from leaf litter
atmospheric_ Frac_decomposable_coarse_dead_ root_breakdown	Fraction of biomass that breaks down to atmosphere from coarse dead roots

atmospheric_ frac_decomposable_fine_dead_ root_breakdown	Fraction of biomass that breaks down to atmosphere from fine dead roots
atmospheric_ frac_resistant_standing_ deadwood_breakdown	Fraction of biomass that breaks down to atmosphere from resistant standing deadwood
atmospheric_ frac_resistant_downed_ deadwood_breakdown	Fraction of biomass that breaks down to atmosphere from resistant downed deadwood
atmospheric_ frac_resistant_bark_litter_ breakdown	Fraction of biomass that breaks down to atmosphere from resistant bark litter
atmospheric_ frac_resistant_ leaf_litter_breakdown	Fraction of biomass that breaks down to atmosphere from resistant leaf litter
atmospheric_ frac_resistant_coarse_dead_ root_breakdown	Fraction of biomass that breaks down to atmosphere from resistant coarse dead roots
atmospheric_ frac_resistant_fine_dead_ root_breakdown	Fraction of biomass that breaks down to atmosphere from fine dead roots
biomass_equation_id	Equations ID
reference_age	Reference age
alpha	Growth Equation coefficient
beta	Growth Equation coefficient
beta0	Growth Equation coefficient
beta1	Growth Equation coefficient
beta2	Growth Equation coefficient
a1	Growth Equation coefficient
a2	Growth Equation coefficient
a3	Growth Equation coefficient
b1	Growth Equation coefficient
b2	Growth Equation coefficient
b3	Growth Equation coefficient

si	Growth Equation coefficient
root_to_shoot	Root to Shoot Ratio
bark_frac	Fraction of stem and bark mass that is bark
coarse_root_frac	Fraction of roots that are coarse roots
fine_root_frac	Fraction of roots that are fine roots

F3 : INIT

At the start of a Run, users may wish to have an initial biomass value. This table provides a way to include an initial biomass value at the start of the Run. Alternatively, plantation plant dates may be used to give an initial biomass value.

Columns

Column Name	Description
init_id	The primary key of a plantation forest initial values record
species_id	Unique ID for species (or forest Type)
stem_carbon_mass	Initial biomass in stem wood
branch_carbon_mass	Initial biomass in branch mass
bark_carbon_mass	Initial biomass in bark mass
leaf_carbon_mass	Initial biomass in leaf mass
coarse_root_carbon_mass	Initial biomass in coarse roots
fine_root_carbon_mass	Initial biomass in fine roots
age_init	Initial age of the forest

F4 : DEBRIS_INIT

Debris pool is the Dead Organic Matter Pool. An initial debris pool biomass can be assigned at the start of a simulation. This table provides a way to populate the initial dead organic matter. If no value is included, the initial Dead Organic Matter is typically taken to be zero.

Columns

Column Name	Description
debris_init_id	The primary key of a plantation forest initial debris record
species_id	Unique ID for a plantation species
decomposable_standing_deadwood_carbon_mass	Initial biomass of decomposable standing deadwood, in tonnes carbon per hectare
decomposable_downed_deadwood_carbon_mass	Initial biomass of decomposable standing deadwood, in tonnes carbon per hectare
decomposable_bark_litter_carbon_mass	Initial biomass of decomposable bark litter, in tonnes carbon per hectare
decomposable_leaf_litter_carbon_mass	Initial biomass of decomposable leaf litter, in tonnes carbon per hectare
decomposable_coarse_dead_root_carbon_mass	Initial biomass of decomposable coarse dead wood from roots, in tonnes carbon per hectare
decomposable_fine_dead_root_carbon_mass	Initial biomass of decomposable fine dead wood from roots, in tonnes carbon per hectare
resistant_standing_deadwood_carbon_mass	Initial biomass of resistant standing deadwood, in tonnes carbon per hectare
resistant_downed_deadwood_carbon_mass	Initial biomass of decomposable standing deadwood, in tonnes carbon per hectare
resistant_bark_litter_carbon_mass	Initial biomass of resistant bark litter, in tonnes carbon per hectare
resistant_leaf_litter_carbon_mass	Initial biomass of resistant leaf litter, in tonnes carbon per hectare
resistant_coarse_dead_root_carbon_mass	Initial biomass of resistant coarse dead wood from roots, in tonnes carbon per hectare
resistant_fine_dead_root_carbon_mass	Initial biomass of resistant fine dead wood from roots, in tonnes carbon per hectare

F5 : EVENT_TYPE

Events types FLINT are either Plant (1) or Thin (2). This table is typically utilized as a reference integrity table. It provides a way to restrict users from creating invalid event types.

Columns

Column Name	Description
event_type_id	The primary key of a plantation forest event record
event_type	The name of each event

F6 : EVENT

An event is the movement of carbon from one pool to another at a single point in time. Where forest modules are concerned, there are usually only two event types; Planting and Thinning. These event types are applied to both plantations and natural forests. The Planting moves carbon from the atmosphere to the living biomass pools, while Thinning moves carbon from the living biomass pools to product pools or dead matter pools. The Thinning event is used to define Pruning, Thinning, and Final Harvests. These movements are usually presented as a fraction of carbon from Pool A to Pool B. This table provides a way to include all events valid for plantations (both Thin and Plant events).

Columns

Column Name	Description
event_id	The primary key of a plantation forest event record
event_type_id	Unique ID for the event type
name	Event Type name

F7 : PLANT_EVENT

Planting events typically move carbon from the atmosphere to the living biomass pools. This table provides a way of detailing the specific effects of each plant event, including the species, age that is planted, and the initial planting mass.

Columns

Column Name	Description
event_id	The primary key of a plantation forest plantation event record
event_type_id	Unique ID for the event type
species_id	Unique ID for species (or forest Type)
age	The planted age of a tree.
stem_mass	Planted stem mass
branch_mass	Planted branch mass
bark_mass	Planted bark mass
leaf_mass	Planted leaf mass
coarse_root_mass	Planted coarse root mass
fine_root_mass	Planted fine root mass

F8 : THIN_EVENT

Thinning events typically move carbon from the living biomass pools to product pools and dead matter pools. In the FLINT context, where Forest Modules are involved, thinning events are used to define Pruning, Thinning, and Final Harvests. These movements are then presented as a fraction of carbon from Pool A to Pool B. This table provides a way to detail all of the Thin events that are relevant for Plantation forests, and the associated fractions of forest biomass that move from one pool to another. The Thin events typically include a fraction of biomass affected, and then the fraction of biomass that moves from one pool to another. A Thin event where 50% of the biomass was affected, and 50% of stem wood went to biofuel, would mean that 25% (50% of 50%) of stem wood would go to biofuel.

Columns

Column Name	Description
event_id	The primary key of a plantation forest thin event record
event_type_id	Unique ID for the event type
frac_affected	The fraction of biomass that was affected by the thin event (0-100%)
frac_stem_to_biofuel	The fraction of stem wood from the affected biomass that is moved to biofuel.
frac_stem_to_paper	The fraction of stem wood from the affected biomass that is moved to paper.
frac_stem_to_sawnwood	The fraction of stem wood from the affected biomass that is moved to sawnwood.
frac_stem_to_poles_posts	The fraction of stem wood from the affected biomass that is moved to poles and posts.
frac_stem_to_woodpanel	The fraction of stem wood from the affected biomass that is moved to wood panel.
frac_stem_to_standing_deadwood	The fraction of stem wood from the affected biomass that is moved to standing deadwood.
frac_stem_to_downed_deadwood	The fraction of stem wood from the affected biomass that is moved to down dead wood.
frac_branch_to_biofuel	The fraction of branch wood from the affected biomass that is moved to biofuel.
frac_branch_to_paper	The fraction of branch wood from the affected biomass that is moved to paper.
frac_branch_to_sawnwood	The fraction of branch wood from the affected biomass that is moved to sawnwood.

frac_branch_to_poles_posts	The fraction of branch wood from the affected biomass that is moved to poles and posts.
frac_branch_to_standing_deadwood	The fraction of branch wood from the affected biomass that is moved to standing deadwood.
frac_branch_to_downed_deadwood	The fraction of branch wood from the affected biomass that is moved to down dead wood.
frac_bark_to_biofuel	The fraction of bark from the affected biomass that is moved to biofuel
frac_bark_to_bark_litter	The fraction of bark from the affected biomass that is moved to litter
frac_leaf_to_biofuel	The fraction of leaf from the affected biomass that is moved to biofuel
frac_leaf_to_leaf_litter	The fraction of leaf from the affected biomass that is moved to litter
Frac_coarse_root_to_dead_ coarse_root	The fraction of coarse roots from the affected biomass that is moved dead coarse roots
frac_fine_root_to_dead_fine_root	The fraction of fine roots from the affected biomass that is moved dead fine roots
frac_bark_litter_to_biofuel	The fraction of bark litter from the affected biomass that is moved biofuel
frac_leaf_litter_to_biofuel	The fraction of leaf litter from the affected biomass that is moved biofuel
frac_standing_deadwood_to_biofuel	The fraction of standing dead wood from the affected biomass that is moved biofuel
Frac_standing_deadwood_to_ downed_deadwood	The fraction of standing dead wood from the affected biomass that is moved downed deadwood
frac_standing_deadwood_to_paper	The fraction of standing dead wood from the affected biomass that is moved paper
Frac_standing_deadwood_to_ sawnwood	The fraction of standing dead wood from the affected biomass that is moved sawnwood
Frac_standing_deadwood_to_ woodpanel	The fraction of standing dead wood from the affected biomass that is moved wood panel
frac_downed_deadwood_to_biofuel	The fraction of downed dead wood from the affected biomass that is moved biofuel
frac_dead_coarse_root_to_biofuel	Fraction of dead coarse roots that move to biofuel

F9 : REGIME

Regimes are a typically a standard sequence of events. These sequences could include a single event, such as planting, or a series of events, such as planting and then fertiliser application. Any one regime can includeinclude one or more Activities.

Where forest plantations are involved, FLINT Implementations should prescribe a unique management regime for each unique collection of tree species (Cypress, Eucalyptus, Pine), grown for a specific purpose (Sawn Timber & Plywood, Pulp, Poles), within a specific management system (Non-Resident cultivation, Total Ban) and within a specified time period (years or year ranges).

This table typically identifies the regime name, species and final events to a regime.

Columns

Column Name	Description
regime_id	The primary key of a plantation forest regime record
name	The descriptive name of the plantation forest regime
species_id	The tree species that was or is still under the plantation forest management regime. Foreign key to fp_species
finalize_event_id	The finalise_event _id refers to the event that will occur when there is a transition (land use change).

F10 : REGIME_EVENT

Regimes are a typically a standard sequence of events. These sequences could include a single event, such as planting, or a series of events, such as planting and then fertiliser application. Any one regime can include one or more Activities. This table provides a way to identify the event type and age (year and day) for a regime. There can be multiple events associated with one regime_id, and each event typically has a unique regime_event_id.

Columns

Column Name	Description
regime_event_id	The primary key of a plantation forest regime event record
year	The year from planting that the event occurs
day	The day in the year from panting that the event occurs
event_id	The unique ID for the event that occurs
regime_id	The unique ID for a regime

F11 : ACTIVITY

A management regime, defined otherwise is as a bundle of human activities that serve one or more land-use purposes. Different management regimes usually have a different set of events. Regimes are thus typically linked to Activities. This table provides a way to capture forest plantations management regimes activities, and the relative frequencies (likelihood of occurrence) of the activities occurring within the activity suite. The effect being, where there are multiple activities that relate to an activity suite, the relative frequency is used to determine which activity is applied to a simulation unit (pixel).

Columns

Column Name	Description
activity_id	The primary key of a plantation forest activity record
frequency	The rate at which an activity occurs, over a particular period of time, for a given management regime
regime_id	The management regime that applies to the activity.

F112: ACTIVITY SUITE

The Activity Suite defines the valid group of activities for a given location, year ranges, and land cover. Under an activity suite, there can be one or more activities. The previous_land_cover and reason_for_change are included to allow for attribution. This table also identifies which activity suites are valid for which years.

Columns

Column Name	Description
activity_suite_id	The primary key of a plantation forest activity suite record
location_id	The ID that represents a specific geographical area (location)
year_valid_low	The year before which the specific Activity Suite is not valid
year_valid_high	The year after which the specific Activity Suite is not valid
previous_land_cover	The prior land cover for which the Activity Suite is valid. This is not used, but is a place-holder for future improvement.
reason_for_change	The attributed reason for change for which the Activity Suite is valid. This is not used, but is a place-holder for future improvement.

Appendix 1 : Basic Dependencies

Dependency	About
bash-completion	Programmable completion for the bash shell. This package extends bash's standard completion behavior to achieve complex command lines with just a few keystrokes. It was conceived to produce programmable completion routines for the most common Linux/UNIX commands, reducing the amount of typing sysadmins and programmers need to do on a daily basis.
build-essential	Informational list of build-essential packages. This package contains an informational list of packages which are considered essential for building Debian packages. It also depends on the packages on that list, to make it easy to have the build-essential packages installed.
doxygen	Documentation generation tool. Doxygen is a documentation system for C, C++, Java, Objective-C, Python, IDL and to some extent PHP, C#, and D. It can generate an on-line class browser (in HTML) and/or an off-line reference manual (in LaTeX) from a set of documented source files.
doxygen-latex	Doxygen dependency package. Adds dependencies for all LaTeX packages required to build documents using the default stylesheet.
git	Fast, scalable, distributed version control system. This package provides the git main components with minimal dependencies.
gdb	GNU Debugger. GDB is a source-level debugger, capable of breaking programs at any specific line, displaying variable values, and determining where errors occurred. Currently, gdb supports C, C++, D, Objective-C, Fortran, Java, OpenCL C, Pascal, assembly, Modula-2, Go, and Ada. A must-have for any serious programmer.
graphviz	Open source graph visualization software.

	Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. This package contains graph visualization command-line tools.
libcurl4-gnutls-dev	Development files and documentation for libcurl (GnuTLS flavour). libcurl is an easy-to-use client-side URL transfer library, supporting DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP. libcurl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, Kerberos), file transfer resume, http proxy tunneling and more.
libeigen3-dev	Lightweight C++ template library for linear algebra. Eigen 3 is a lightweight C++ template library for vector and matrix math, a.k.a. linear algebra. Unlike most other linear algebra libraries, Eigen 3 focuses on the simple mathematical needs of applications.
libgeos-dev	Geometry engine for GIS. GEOS provides a spatial object model and fundamental geometric functions. It implements the geometry model defined in the OpenGIS Consortium Simple Features Specification for SQL.
libhdf4-alt-dev	Hierarchical Data Format development files (without NetCDF). HDF is a multi-object file format for storing and transferring graphical and numerical data mainly used in scientific computing. HDF supports several different data models, including multidimensional arrays, raster images, and tables. Each defines a specific aggregate data type and provides an API for reading, writing, and organizing the data and metadata.
libhdf5-serial-dev	Packages providing libhdf5-serial-dev This is a virtual package.
libnetcdf-dev	Creation, access, and sharing of scientific data. NetCDF (network Common Data Form) is a set of interfaces for array-oriented data access and a freely distributed collection of data access libraries for C, Fortran, C++, Java, and other languages. The netCDF libraries support a machine-independent format for representing scientific data. Together, the interfaces, libraries, and format support the creation, access, and sharing of scientific data.
libpoppler-dev	PDF rendering library. Poppler is a PDF rendering library based on Xpdf PDF viewer.
libpq-dev	Header files for libpq5 (PostgreSQL library).

	Header files and static library for compiling C programs to link with the libpq library in order to communicate with a PostgreSQL database backend.
libproj-dev	Cartographic projection library. Proj and invproj perform respective forward and inverse transformation of cartographic data to or from Cartesian data with a wide range of selectable projection functions (over 100 projections).
libspatialite-dev	Geospatial extension for SQLite. The SpatiaLite extension enables SQLite to support spatial (geometry) data in a way conformant to OpenGis specifications, with both WKT and WKB formats.
libssl-dev	Secure Sockets Layer toolkit. This package is part of the OpenSSL project's implementation of the SSL and TLS cryptographic protocols for secure communication over the Internet. It contains development libraries, header files, and manpages for libssl and libcrypto.
libxml2-dev	Development files for the GNOME XML library. XML is a metalanguage to let you design your own markup language. A regular markup language defines a way to describe information in a certain class of documents (eg HTML). XML lets you define your own customized markup languages for many classes of documents. It can do this because it's written in SGML, the international standard metalanguage for markup languages.
nasm	General-purpose x86 assembler. Netwide Assembler: NASM will currently output flat-form binary files, a.out, COFF and ELF Unix object files, and Microsoft 16-bit DOS and Win32 object files.
openssl	Secure Sockets Layer toolkit - cryptographic utility. This package is part of the OpenSSL project's implementation of the SSL and TLS cryptographic protocols for secure communication over the Internet. It contains the general-purpose command line binary /usr/bin/openssl, useful for cryptographic operations.
postgis	Geographic objects support for PostgreSQL. PostGIS adds support for geographic objects to the PostgreSQL object-relational database. In effect, PostGIS "spatially enables" the PostgreSQL server, allowing it to be used as a backend spatial database for geographic information systems (GIS).
postgresql-client-10	Front-end programs for PostgreSQL

	This metapackage always depends on the currently supported database client package for PostgreSQL.
python3-dev	Header files and a static library for Python (default). Header files, a static library and development tools for building Python modules, extending the Python interpreter or embedding Python in applications.
python3-numpy	Fast array facility to the Python 3 language. Numpy contains a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code, and useful linear algebra, Fourier transform, and random number capabilities.
python3-pip	Python package installer. pip is the Python package installer. It integrates with virtualenv, doesn't do partial installs, can save package state for replaying, can install from non-egg sources, and can install from version control repositories.
software-properties-common	Manages the repositories that you install software from (common). This software provides an abstraction of the used apt repositories. It allows you to easily manage your distribution and independent software vendor software sources.
sqlite3	Command line interface for SQLite 3. SQLite is a C library that implements an SQL database engine. Programs that link with the SQLite library can have SQL database access without running a separate RDBMS process.
wget	Retrieves files from the web. Wget is a network utility to retrieve files from the web using HTTP(S) and FTP, the two most widely used internet protocols. It works non-interactively, so it will work in the background, after having logged off. The program supports recursive retrieval of web-authoring pages as well as FTP sites.

Appendix 2 : Core Dependencies

Dependency	About
boost	Free, peer-reviewed, portable C++ source libraries. boost libraries are a collection of C++ libraries that provide support for standard tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing..
cmake	Build process manager. CMake is an open-source, cross-platform family of tools designed to build, test and package software.
fmt	A modern formatting library. {fmt} is an open-source formatting library for C++ that can be used as a safe and fast alternative to (s)printf and iostreams
gdal	Raster and Vector translation library. GDAL is a translator library for raster and vector geospatial data formats that is released under an X/MIT style Open Source License by the Open Source Geospatial Foundation.
poco	C++ libraries for building network- and internet-based applications. The POrtable COmponents (POCO) C++ Libraries are a set of cross-platform C++ libraries for developing computer network-centric, portable applications in C++.
rabbitmq-c	This is a C-language AMQP client library for use with v2.0+ of the RabbitMQ broker. RabbitMQ is an open-source message-broker software that originally implemented the Advanced Message Queuing Protocol (AMQP) and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol (STOMP), Message Queuing Telemetry Transport (MQTT), and other protocols.
SimpleAmqpClient	C++ wrapper around the rabbitmq-c C library SimpleAmqpClient is an easy-to-use C++ wrapper around the rabbitmq-c C library
sqlite	Database engine.

	SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.
turtle	Mock object library. Turtle is a C++ mock object library based on Boost with a focus on usability, simplicity and flexibility.
zipper	C++ wrapper around minizip compression library. Zipper is a reliable, simple and flexible compression library that supports all kinds of inputs and outputs. Moreover it allows the compression of files into memory instead of being restricted to file compression only, and using data from memory instead of just files as well.

Appendix 3: Environmental Variables

Variable	About
CURL_CA_BUNDLE	CURL_CA_BUNDLE is an environmental variable used to specify a custom Certificate Authority (CA) certificate path.
GDAL_DATA	GDAL_DATA is an environment variable used to specify location of supporting files used by GDAL libraries as well as GDAL and OGR utilities.
GDAL_HTTP_MERGE_CONSECUTIVE_RANGES	GDAL_HTTP_MERGE_CONSECUTIVE_RANGES is an environment variable used to specify if ranges of a single ReadMultiRange request that are consecutive should be merged into a single request
GDAL_HTTP_MULTIPLEX	GDAL_HTTP_MULTIPLEX is an environment variable used to specify if multiplexing can be used to download multiple ranges in parallel, during ReadMultiRange requests that can be emitted by the GeoTIFF driver
GDAL_HTTP_VERSION	GDAL_HTTP_VERSION is an environment variable used to specify which HTTP version to use
LANG	LANG is an environment variable used to specify a locale
LC_ALL	LC_ALL is an environment variable used to override all LC_xxx environmental variables. LC_xxx environment variables e.g. LC_CTYPE, LC_NUMERIC, LC_TIME, LC_COLLATE, LC_MONETARY, LC_MESSAGES, and so on, are the environment variables meant to override LANG and affect a single locale category only
LD_LIBRARY_PATH	LD_LIBRARY_PATH is an environment variable used to specify a list of directories in which to search for Executable and Linkable Format (ELF) libraries at execution time
PATH	PATH is an environment variable used to specify to the shell which directories to search for executable files (i.e., ready-to-run programs) in response to commands issued by a user
PYTHONPATH	PYTHONPATH is an environment variable used to specify additional directories where python will look for modules and packages

Appendix 4: Keys For Accessing BIOS settings

Manufacturer	F1	F2	F3	F6	F10	F11	F12	ESC	INS	DEL
Acer	A	**C**								**C**
Asus		**C**							A	A
DELL	A	**C**	A				A			A
HP	A	A		A	**C**	A	A	**C**		
Lenovo	**C**	**C**								
Sony	A	**C**	**C**							
Toshiba	A	C						A		

Where C = Most Common and A = Alternative

Abbreviations

Abbreviation	Meaning
CEIP	Customer Experience Improvement Program
CPU	Central Processing unit
FLINT	Full Lands Integration Tool
HW	Hardware
OS	Operating System
RAM	Random Access Memory
TCP	Transmission Control Protocol
URL	Uniform Resource Locator

References

Basic Dependencies:

Ubuntu Packages	https://packages.ubuntu.com/

Core Dependencies:

Boost C++ Libraries	https://www.boost.org/
CMake	https://cmake.org/
fmtlib/fmt	https://github.com/fmtlib/fmt
GDAL	https://gdal.org/
POCO	https://pocoproject.org/
RabbitMQ C	https://github.com/alanxz/rabbitmq-c
SimpleAmqpClient	https://github.com/alanxz/SimpleAmqpClient
SQLite	https://www.sqlite.org/index.html
Turtle	http://turtle.sourceforge.net/
Zipper	https://github.com/sebastiandev/zipper