University of Sarajevo
Faculty of Electrical Engineering
Department of Computer Science and Informatics

NatId Setup Guide

Artificial intelligence

Authors: Benjamin Bandić, Nedim Bečić
Professor: Izudin Džafić

Installing the natID Framework

Follow these steps carefully to set up the natID framework on your system.

Download the natID Repository

The first step is to obtain the natID source code.

1. **Open your web browser** and navigate to the natID GitHub repository (e.g., `https://github.com/idzafic/natID/`).
2. **Clone or Download:**
   - **Option A: Using Git (Recommended)** If you have Git installed, this is the most flexible method.
     a) Open your terminal (macOS/Linux) or Git Bash/Command Prompt (Windows).
     b) Navigate to a directory where you want to temporarily download the repository (it will be moved later). For example:
     ```
     cd ~/Downloads
     ```
     c) Execute the following command to clone the repository:
     ```
     git clone [https://github.com/idzafic/natID/]
     ```
   - **Option B: Downloading as a ZIP File**
     a) On the GitHub repository page, click the green "Code" button.
     b) Select "Download ZIP".
     c) Save the `natID-main.zip` file (or similar name) to your `Downloads` folder.
     d) Once downloaded, extract the contents of the ZIP file. This will create a folder named `natID-main` (or similar).

Set Up in the HOME Folder

natID expects to be located in your user's `HOME` directory, along with two essential subfolders: `Work` and `other_bin`.

1. **Locate your HOME Directory:**
   - **macOS/Linux:** Your HOME directory is typically `/home/yourusername` or `~`. You can verify this by opening a terminal and typing `echo $HOME`.
   - **Windows:** Your HOME directory is usually `C:\Users\YourUsername`. To confirm or set the `HOME` environment variable:
     - **Check Existing `HOME` Variable:**
       i. Open the Command Prompt or PowerShell.
       ii. Type `echo %HOME%` and press Enter. If a path is displayed, this is your current `HOME` directory.
     - **Define/Set `HOME` Variable (if not defined or incorrect):**
       i. Right-click on the "Start" button and select "System" (or go to `Settings > System > About`).
       ii. Click on "Advanced system settings" on the right-hand side.
       iii. In the "System Properties" window, click the "Environment Variables..." button.
       iv. Under "User variables for YourUsername", click "New...".
       v. For "Variable name", enter `HOME`.
       vi. For "Variable value", enter the full path to your user profile directory, typically `C:\Users\YourUsername` (replace `YourUsername` with your actual Windows username).
       vii. Click "OK" on all open windows to save the changes.

viii. **Important:** You may need to restart your terminal or IDE for the new environment variable to take effect.

- **Move the natID Repository:**
  - Move the `natID-main` folder (or whatever you named the cloned/extracted repository) directly into your `HOME` directory.
  - Rename this folder to `natID` for simplicity.
  - So, on macOS/Linux, you should have `~/natID`. On Windows, `C:\Users\YourUsername\natID`.
- **Create Essential Subfolders:**
  - Inside your `HOME` directory, create two new empty folders:
    - `Work`
    - `other_bin`
  - After this step, your `HOME` directory structure should look something like this (paths are illustrative):

```
Your HOME directory
 natID/ (containing the framework source code)
 Work/ (empty for now, will contain demo projects)
 other_bin/ (empty for now, will contain release binaries)
```

Download Release Binaries

The `other_bin` folder needs pre-compiled binaries from the natID releases.

1. **Open your web browser** and go to the "Releases" section of the natID GitHub repository.
2. **Identify and Download:** Look for files named `other_bin_*` (e.g., `other_bin_macOS.zip`, `other_bin_linux.tar.gz`, `other_bin_windows.zip`). Download the package appropriate for your operating system.
3. **Extract into `other_bin`:**
   - Once downloaded, extract the contents of this archive directly into the `other_bin` folder you created in your `HOME` directory.
   - **Do not create a subfolder within `other_bin`** — the contents (DLLs, .so files, dylibs, etc.) should be directly inside `~/other_bin` or `C:\Users\YourUsername\other_bin`.

```
Your HOME directory
 natID/
 Work/
 other_bin/
     some_library.dll (Windows)
     libsome_library.so (Linux)
     libsome_library.dylib (macOS)
     ... (other binaries)
```

4. **macOS Specific: Remove Quarantine Attribute for Binaries (Crucial for Gatekeeper)**
   - When downloading files from the internet, macOS adds a "quarantine" attribute. Gatekeeper might prevent these unsigned or unnotarized binaries from being loaded by your applications, leading to "damaged" file errors or crashes.
   - To allow the system to load these libraries, you need to remove this attribute.
     a) Open your Terminal application.
     b) Navigate to your `other_bin` directory:

```
cd ~/other_bin
```

     c) Execute the following command to remove the quarantine attribute from all dynamic libraries (`.dylib` files) and other common executable types (like `.so` for shared objects, and generic executables).

```
xattr -d com.apple.quarantine *.dylib *.so *
```

d) **Note:** This command uses a wildcard ('*') and targets common library extensions. If you encounter further issues with specific files, you may need to apply `xattr -d com.apple.quarantine [filename]` individually, or more broadly to any nested folders that contain executables.

   General tip - exercise caution when removing quarantine attributes from files you do not fully trust(in this case it is okay as this is a trusted source).

Prepare the Build Environment

natID uses CMake for its build system, and you'll need an appropriate IDE/compiler for your OS.

1. **Install CMake:**
   - Download and install CMake (the GUI version is recommended for beginners) from the official website: `https://cmake.org/download/`
   - Follow the installation instructions for your operating system. Ensure that CMake is added to your system's PATH during installation (especially on Windows).
2. **Install Recommended Building Tools (IDE/Compiler):**
   - **Windows (Visual Studio 2022 Community Edition):**
     - Download Visual Studio 2022 Community from the official Microsoft website: `https://visualstudio.microsoft.com/vs/community/`
     - During installation, select the "Desktop development with C++" workload. This will install all necessary compilers, SDKs, and tools.
     - Make sure to include CMake support if prompted during installation.
   - **macOS (Xcode):**
     - Open the App Store on your Mac and search for "Xcode".
     - Click "Get" and then "Install" to download and install Xcode. This may take some time due to its size.
     - After installation, open Xcode at least once to accept the license agreement and allow it to install any additional necessary components.
     - You'll also need to install the Xcode Command Line Tools. Open Terminal and run:
       ```
       xcode-select --install
       ```
   - **Linux (Qt Creator or Visual Studio Code with C++ extensions):**
     - **GTK 4.14 or later:** Ensure your system has GTK 4.14 or later installed. On most modern distributions, this will be available through your package manager. For example, on Ubuntu/Debian:
       ```
       sudo apt update
       sudo apt install libgtk-4-dev
       ```
     - **Qt Creator:**
       - Download Qt Creator from the official Qt website: `https://www.qt.io/download` (You might need to create an account).
       - Follow the installation wizard. Ensure you select the appropriate kits for C++ development.
     - **Visual Studio Code (VS Code):**
       - Download VS Code from `https://code.visualstudio.com/`.
       - Install the "C/C++ Extension Pack" by Microsoft from the Extensions view within VS Code. This will provide C++ language support, debugging, and CMake tools integration.
       - Ensure you have a C++ compiler installed, such as `g++` (usually included with `build-essential` on Debian/Ubuntu or `gcc-c++` on Fedora/RHEL).

```
sudo apt install build-essential # For Debian/Ubuntu
sudo dnf install gcc-c++       # For Fedora
```

Exploring Demo Projects

With the framework installed and your environment ready, you can now explore the natID demo projects.

1. **Locate Demo Projects:**
   The demo projects are located in `~/Work/CppProjects` (or `C:\Users\YourUsername\Work\CppProjects`).
2. **Start with Lectures:** It is highly recommended to begin by exploring the projects within the `Lectures` folder. These are likely designed to introduce you to natID's features incrementally.
3. **Opening Projects in Your IDE:** Most modern IDEs can directly open CMake projects, or you can use CMake from the command line to generate project files.
   - **For macOS (Xcode):**
     a) Open your Terminal.
     b) Navigate to a specific demo project folder, for example:
     ```
     cd ~/Work/CppProjects/Lectures/HelloWorld
     ```
     c) Create a build directory and navigate into it:
     ```
     mkdir build
     cd build
     ```
     d) Run CMake to generate the Xcode project files for the parent directory:
     ```
     cmake -G "Xcode" ..
     ```
     e) Once CMake completes, open the generated '.xcodeproj' file in Xcode:
     ```
     open YourProjectName.xcodeproj
     ```
     (Replace `YourProjectName` with the actual project name, which CMake derives from the project's `CMakeLists.txt`.)
   - **For Windows (Visual Studio):**
     a) **Option A: Open directly in Visual Studio (Recommended)**
        - Open Visual Studio.
        - Go to `File > Open > CMake...` and navigate to the root of your desired demo project (e.g., `C:\Users\YourUsername\Work\CppProjects\Lectures\HelloWorld`). Select the `CMakeLists.txt` file. Visual Studio will automatically configure the project.

     b) **Option B: Generate with CMake GUI/CLI**
        - Open CMake GUI, or your command prompt.
        - Navigate to your chosen demo project (e.g., `HelloWorld`).
        - Create a `build` folder inside it.
        - Run CMake, specifying a Visual Studio generator (e.g., `cmake -G "Visual Studio 17 2022" ..`).
        - Open the generated `.sln` solution file in Visual Studio.
   - **For Linux (Qt Creator or Visual Studio Code):**
     a) **With Qt Creator:**
        - Open Qt Creator.
        - Go to `File > Open File or Project...`.
        - Navigate to your desired demo project folder (e.g., `~/Work/CppProjects/Lectures/HelloWorld`) and select the `CMakeLists.txt` file.

  - Qt Creator will guide you through configuring the project (selecting a kit, build directory, etc.).

  b) **With Visual Studio Code:**
  - Open VS Code.
  - Go to `File > Open Folder...` and select the root of your desired demo project folder (e.g., `~/Work/CppProjects/Lectures/HelloWorld`).
  - The CMake Tools extension (if installed) should automatically detect the `CMakeLists.txt` and prompt you to configure and build the project. Look for prompts at the bottom right or in the CMake sidebar.
  - Alternatively, open the Command Palette ('Ctrl+Shift+P' or 'Cmd+Shift+P') and search for "CMake: Configure" to start the process.

4. **Build and Run:** Once the project is loaded and configured in your IDE, build the project and then run it to see the natID applications in action!