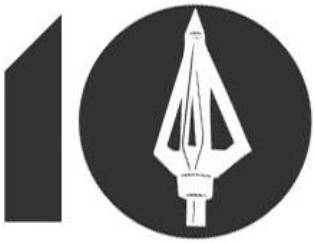

SIMPLE IMAGING. TACTIAL TRIAGE. ZERO CLICKS.

IO DEVELOPER GUIDE

CIPHER TECH SOLUTIONS, INC.
WWW.CIPHERTECHSOLUTIONS.COM
IO@CIPHERTECHSOLUTIONS.COM

Contents

Developer Notes

Despite being a Java project, IO is limited to Windows currently due to our use of various Windows APIs and utilities to extract disk information. Support for other platforms may be added in the future, but is not currently in the development plan.

Creating a Custom Processor

To create and use a custom processor you must implement the IProcessor interface and add it to either the addDefaultProcessors method in ProcessorManager or to the run() method within setupImaging in ProcessController.

You may find TriageProcessorBase to be useful in helping you to create an IProcessor more easily. In the future, we intend to add a configuration file that allows IProcessor classpaths to be specified for loading to enable the addition or removal of processors without modifying portions of the project within Cipher Tech's namespace.
Implementing IProcessor requires the following methods:

process()

This method is how you will receive data from the disk. The byte array given will always be filled, although the size may not be consistent. This method must return as quickly as possible to ensure smooth operation of ION. As such, it is recommended that you implement this method via simply adding the byte array to a bounded queue with non-blocking adds. You can then have another thread or threads pull from this queue and perform your desired processing. Bounding the queue is recommended because otherwise if your processing lags behind the disk read, you will quickly run out of RAM as it begins queuing the entire disk into memory.

initialize()

This method will be called before process is called. You may perform any necessary setup and initialization during it. If you are following the recommended approach for processing, you may wish to launch your processing threads here.

finish()

This method will be called when all data has been read from disk. After this call, no further calls will be made to process. You may handle this signal however you like, although you should not block on this call. waitForExit will be called after this method, and any blocking you wish to do should be done there.

waitForExit()

This method will be called after finish, or, potentially, after cancel. You may wait for your processing to complete here if you would like to ensure your processing is complete before ION terminates. In any event, you should ensure you have closed any open resource handles.

cancel()

This method will be called when the process needs to terminate prematurely for some reason (user request, disk error, disk removal, etc), you should stop your processing as soon as you can and close any open resource handles.

Building IO

Required

- Java 8 or higher.
- 64 bit Windows.
- Git

Recommended (and assumed for these instructions)

- Eclipse
- E(fx)clipse plugin for Eclipse
- SceneBuilder for JavaFX
- Launch4j

Build Steps

1. Use git to checkout the repository from TODO_FILL_THIS_IN_WITH_URL to your local machine.
2. Next, import the project into Eclipse as an E(fx)clipse project. You can still build it without E(fx)clipse, but you will have to generate your own build.xml file, which will not be covered in this guide.
3. In the base directory in Eclipse, right click on build.fxbuild and select "Build Export FX Application". This will generate and run a build.xml file.
    a. If you are generating your own build file, instead of this step, simply run your build.xml file.
4. You should now have an IO.jar in /build/dist. If not, check your console output in Eclipse. You may need to modify your system environment variables for JavaFX to be able to build properly.
    a. Note, this JAR file will only work alongside the libs folder as it will otherwise be lacking dependencies.
5. To build an executable, we recommend Launch4j. Once launch4j is installed, launch it and use it to open the IOexe.cfg.xml file (located in the luanch4j folder within the project) and click the "gear" icon in launch4j.
6. You should now have a IO.exe file in your /build/dist folder. If you do not, check the output of Launch4j for any possible errors.
    a. As with the JAR, this exe will only work when placed alongside the libs folder containing all necessary dependencies. Despite being an exe, it will also still require that Java 8 or higher is installed on the machine to run.

Basic Package Descriptions

- applicationLogic: Contains logic regarding the overall state of the application. This includes the back-end representation of the user configuration, as well a controller to manage the various processing and polling threads.
    - options: The user configurable portion of the application state.
- device: Our representations of readable devices, namely Disks and Volumes currently, along with a few related helper classes for retrieving device information.
- ewf: Our representations of the portions of an Encase6 file.
- logging: Some custom logging utilities.
- processing: The main part of our image creation and processing. The base package includes the manager for a single imaging process, the drive read process, the compression process, and the output process.
    - digests: Contains classes for computing the MD5 and SHA1 hashes that implement our IProcessor interface.
    - triage: Contains classes for performing triaging of the drive as it is imaged by implementing our IProcessor interface.
- ui: Contains all of the GUI related components and controllers.
    - css: Has the JavaFX style sheet in it.
    - fxml: Has all of our fxml files.
    - icons: Contains all of our image files.
- usb: Contains USB write block handling as well as polling for inserted or removed devices.

For more in-depth descriptions of any part of the project, please consult the source code documentation itself. Javadoc is available for all classes and most public methods.

If you have any issues, feel free to contact us via email at io@ciphertechsolutions.com or by submitting an issue to our GitHub page at https://www.ciphertechsolutions.com/open-source/.