How to generate InkAtlas files for Cyberpunk 2077 Modding

Two tools are required:

- The Wolvenkit CLI tool (dotnet tool)
  - `dotnet tool install -g wolvenkit.cli`
- The inkatlas generator tool (python tool)
  - Github Link
  - By default, there are no arguments that can be passed in. It is supplied it's arguments during execution of the script.
  - I have converted it (and it's dependencies) into an `.exe` file that can be run from within CRA and allows for arguments to be passed in. This `.exe` is embedded within CRA's assembly and extracted to a temporary location upon execution.

The inkatlas tool generates three files from the PNG's within a folder:

- `filename.inkatlas.json`
- `filename.png`
- `filename_1080.png`

The output file path **MUST** be within this structure for the tool to work:

`../source/raw/`

This is because the tool expects to be run against a WolvenKit project. We can bypass this by "mimicking" what a project structure would look like.

Once we have the raw files, we can generate the `.archive` file using the WolvenKit CLI tool:

- Generate `.inkatlas`:
  - `cp77tools convert deserialize "X:\Files\Downloads\generate inkatlas script\dist\output\source\raw\test.inkatlas.json"`
- Generate `.xbm` texture file:
  - `cp77tools import -p "X:\Files\Downloads\generate inkatlas script\dist\output\source\raw"`

Create a directory structure for the faux mod: `<radio name>\base\icon\`

- Generate final `.archive` file:
  - `cp77tools pack -p "X:\Files\Downloads\generate inkatlas script\dist\output\source\raw\<radio name>"`
  - For this to work, we need to make sure our directory structure within the root folder is as follows:
    - `<radio name>\base\icon\<..files generated above..>`
    - The `radio name` will become the name of the final `.archive` file. For instance, if the folder structure is `awesome station\base\icon\..`, the output file would be `awesome station.archive`.

Sequence of commands example

▶ Expand to see commands

```
X:\Files\Downloads\generate inkatlas script\generate_test>mkdir source

X:\Files\Downloads\generate inkatlas script\generate_test>generate_inkatlas.exe

INKATLAS GENERATOR INPUT:
    Enter the path to the folder containing your individual icon PNG images: .
    Enter the path to output the raw inkatlas files for you to import in
Wolvenkit: .\source
    Enter the name for your new inkatlas file (without extension): awesome icon
Data has been saved to .\source\raw\awesome icon.inkatlas.json
Combined image has been saved to .\source\raw\awesome icon.png
Combined image has been saved to .\source\raw\awesome icon_1080.png

X:\Files\Downloads\generate inkatlas script\generate_test>cp77tools convert
deserialize ".\source\raw\awesome icon.inkatlas.json"
[ 0: Information ] - Found 1 files to process.
[ 0: Success     ] - Imported awesome icon.inkatlas.json to
X:\Files\Downloads\generate inkatlas script\generate_test\source\raw\awesome
icon.inkatlas.
[ 0: Success     ] - Converted X:\Files\Downloads\generate inkatlas
script\generate_test\source\raw\awesome icon.inkatlas.json to CR2W
[ 0: Information ] - Elapsed time: 4200ms.

X:\Files\Downloads\generate inkatlas script\generate_test>cp77tools import -p
".\source\raw"
[ 0: Warning     ] - Image dimension (width and/or height) is an odd number.
Texture might not work as expected.
[ 0: Information ] - Imported 2/2 file(s)

X:\Files\Downloads\generate inkatlas script\generate_test>mkdir ..\base\icon

X:\Files\Downloads\generate inkatlas script\generate_test>mkdir base\icon

X:\Files\Downloads\generate inkatlas script\generate_test>mkdir "awesome
mod\base\icon"

X:\Files\Downloads\generate inkatlas script\generate_test>cp ".\source\raw\awesome
icon.xbm" "base\icon\awesome icon.xbm
"
'cp' is not recognized as an internal or external command,
operable program or batch file.

X:\Files\Downloads\generate inkatlas script\generate_test>copy
".\source\raw\awesome icon.xbm" "base\icon\awesome icon.x
bm"
The system cannot find the path specified.
        0 file(s) copied.
```

```
X:\Files\Downloads\generate inkatlas script\generate_test>copy
".\source\raw\awesome icon.xbm" "awesome mod\base\icon\aw
esome icon.xbm"
        1 file(s) copied.

X:\Files\Downloads\generate inkatlas script\generate_test>copy
".\source\raw\awesome icon.inkatlas" "awesome mod\base\ic
on\awesome icon.inkatlas"
        1 file(s) copied.

X:\Files\Downloads\generate inkatlas script\generate_test>cp77tools pack -p
"awesome mod"
[ 0: Success     ] - Finished packing X:\Files\Downloads\generate inkatlas
script\generate_test\awesome mod.archive.
```

Unpacking an `.archive` and getting back a `.png` file

First, unbundle the `.archive` file:

```
cp77tools unbundle -p "X:\Files\Downloads\generate inkatlas script\dist\vwave
icon.archive" -o "X:\Files\Downloads\generate inkatlas script\dist\output"
```

Then, we can convert the `.xbm` file to a `.png`:

```
cp77tools export --uext png -p "X:\Files\Downloads\generate inkatlas
script\dist\output\base\icon\vwave.xbm" -o "X:\Files\Downloads\generate inkatlas
script\dist\output\base\icon"
```

Detailed Steps and Process

We should always operate within the apps working directory. Temporary directories can be created as an exception under `%temp%`. This implementation only supports 1 (ONE) .png file for importing as a station's icon.

Create Icon (`.archive`) Mod

Let's assume we have a root folder: `%localappdata%\RadioExt-Helper\working`:

▶ Directory Tree

```
%localappdata%/RadioExt-Helper/working
```

1. Import the `.png` file to the station's imported directory: `.\imported\<stationName>\<GUID>.png`
    - The directory should be created if it doesn't exist.
    - The imported `.png` should be associated with the station by adding a new **read-only** key to the `CustomData` dictionary:
        - CustomData["iconGuid"] = `<GUID>`

- We can read back the custom data and get the `.png` GUID from it so we can load the original `.png` file into a picture box for previewing now (and on subsequent launches of CRA).
- Import primarily happens using the drag-and-drop feature on the picture box in the station editor. But, should also be able to import using the `Tools` menu item.
- **Final `.png` image path:** `%localappdata%\RadioExt-Helper\working\imported\<stationName>\<GUID>.png`

▶ Directory Tree After Step 1

Assuming `<stationName>` is `station1`:

```
%localappdata%/RadioExt-Helper/working
└── imported/
    └── station1/
        └── 6e00cac9-05d1-4750-a9ee-370f37cebfc2.png
```

2. Copy the `.png` from step 1 to the faux project directory: `.\tools\<atlasName>\source\raw`
   - This directory should be created if it doesn't exist!
   - Execute the command to generate the `.inkatlas.json` file using `generate-inkatlas.exe`.
   - **Final `.inkatlas.json` path:** `%localappdata%\RadioExt-Helper\working\tools\<atlasName>\source\raw\<atlasName>.inkatlas.json`

▶ Directory Tree After Step 2

Assuming `<stationName>` is `station1` and `atlasName` is `station_1_atlas`:

```
%localappdata%/RadioExt-Helper/working/
├── imported/
│   └── station1/
│       └── 6e00cac9-05d1-4750-a9ee-370f37cebfc2.png
└── tools/
    └── station_1_atlas/
        └── source/
            └── raw/
                ├── station_1_atlas.png
                ├── station_1_atlas_1080.png
                └── station_1_atlas.inkatlas.json
```

3. Convert the `.png` file into an `.inkatlas.json` file using `WolvenKit-CLI.exe`:
   - `convert deserialize` command with the `.inkatlas.json` path from step 2.

▶ Directory Tree After Step 3

Assuming `<stationName>` is `station1` and `atlasName` is `station_1_atlas`:

```
%localappdata%/RadioExt-Helper/working/
├── imported/
```

```
│   └── station1/
│       └── 6e00cac9-05d1-4750-a9ee-370f37cebfc2.png
└── tools/
    └── station_1_atlas/
        └── source/
            └── raw/
                ├── station_1_atlas.png
                ├── station_1_atlas_1080.png
                ├── station_1_atlas.inkatlas.json
                └── station_1_atlas.inkatlas
```

4. Import the project files to WolvenKit CLI as `.xbm` files:
    - `import -p` command with the project path: `%localappdata%\RadioExt-Helper\working\tools\<atlasName>\source\raw\`

▶ Directory Tree After Step 4

Assuming `<stationName>` is `station1` and `atlasName` is `station_1_atlas`:

```
%localappdata%/RadioExt-Helper/working/
├── imported/
│   └── station1/
│       └── 6e00cac9-05d1-4750-a9ee-370f37cebfc2.png
└── tools/
    └── station_1_atlas/
        └── source/
            └── raw/
                ├── station_1_atlas.png
                ├── station_1_atlas_1080.png
                ├── station_1_atlas.inkatlas.json
                ├── station_1_atlas.inkatlas
                ├── station_1_atlas.xbm
                └── station_1_atlas_1080.xbm
```

5. Create the archive folder structure and copy `.inkatlas` and `.xbm` files to the `icon` folder:
    - The directories should be created if they do not exist.

▶ Directory Tree After Step 5

Assuming `<stationName>` is `station1` and `atlasName` is `station_1_atlas`:

```
%localappdata%/RadioExt-Helper/working/
├── imported/
│   └── station1/
│       └── 6e00cac9-05d1-4750-a9ee-370f37cebfc2.png
└── tools/
    └── station_1_atlas/
        ├── source/
        │   └── raw/
```

```
        │       ├── station_1_atlas.png
        │       ├── station_1_atlas_1080.png
        │       ├── station_1_atlas.inkatlas.json
        │       ├── station_1_atlas.inkatlas
        │       ├── station_1.atlas.xbm
        │       └── station_1.atlas_1080.xbm
        └── archive/
            └── base/
                └── icon/
                    ├── station_1_atlas.inkatlas
                    ├── station_1.atlas.xbm
                    └── station_1.atlas_1080.xbm
```

6. Pack the contents of the faux project folder to create an `.archive` mod file:
    - The output folder should be a location outside the input folder.
    - For example, `pack -p archive -o station_1_atlas`

▶ Directory Tree After Step 6

Assuming `<stationName>` is `station1` and `atlasName` is `station_1_atlas`:

```
%localappdata%/RadioExt-Helper/working/
├── imported/
│   └── station1/
│       └── 6e00cac9-05d1-4750-a9ee-370f37cebfc2.png
└── tools/
    └── station_1_atlas/
        ├── source/
        │   └── raw/
        │       ├── station_1_atlas.png
        │       ├── station_1_atlas_1080.png
        │       ├── station_1_atlas.inkatlas.json
        │       ├── station_1_atlas.inkatlas
        │       ├── station_1.atlas.xbm
        │       └── station_1.atlas_1080.xbm
        ├── archive/
        │   └── base/
        │       └── icon/
        │           ├── station_1_atlas.inkatlas
        │           ├── station_1.atlas.xbm
        │           └── station_1.atlas_1080.xbm
        └── station_1_atlas.archive
```

7. Associate the final `.archive` file to the staging directory's `icons` folder:
    - We can use the `LoadIconFromFile` method to associate this final `.archive` file with a station.
    - Staging `.archive` Path: `<stagingFolder>\icons`

▶ Directory Tree After Step 7

Assuming `<stationName>` is `station1` and `atlasName` is `station_1_atlas`:

```
%localappdata%/RadioExt-Helper/working/
├── imported/
│   └── station1/
│       └── 6e00cac9-05d1-4750-a9ee-370f37cebfc2.png
└── tools/
    └── station_1_atlas/
        ├── source/
        │   └── raw/
        │       ├── station_1_atlas.png
        │       ├── station_1_atlas_1080.png
        │       ├── station_1_atlas.inkatlas.json
        │       ├── station_1_atlas.inkatlas
        │       ├── station_1.atlas.xbm
        │       └── station_1.atlas_1080.xbm
        ├── archive/
        │   └── base/
        │       └── icon/
        │           ├── station_1_atlas.inkatlas
        │           ├── station_1.atlas.xbm
        │           └── station_1.atlas_1080.xbm
        └── station_1_atlas.archive
stagingFolder/
└── icons/
    └── station_1_atlas.archive
```

Assuming the values from the walkthrough above, the station's `CustomData` should contain these read-only keys:

- CustomData["iconFile"] = `station_1_atlas.archive`
- CustomData["iconFileHash"] = `<SHA256 hash of station_1_atlas.archive>`
- CustomData["iconGuid"] = `6e00cac9-05d1-4750-a9ee-370f37cebfc2`

Extract Raw PNGs from (`.archive`) Mod

This is mainly used when importing a station from a `.zip` or `.rar` file downloaded from Nexus Mods. Using this process, we can unpack the `.archive` file that most custom station mods have and display a preview of the icon within CRA.

This also leads to the possibility of displaying the icon's from the game as well!

1. We start with a path to an `.archive` file. This file (if imported from Nexus Mods via drag-and-drop) should be located within the StagingIconsPath, however it doesn't have to be.
2. Create a temporary directory within the working directory. For example, `%localappdata%/RadioExt-Helper/working/<tempFolderName>`
    - We can use `Path.GetRandomFileName()` to get a random folder name.
    - We'll use this path as a working directory for the export operation.
    - Copy the archive file to this temporary directory.

▶ Directory Tree After Step 2

Assuming `<tempFolderName>` is `4ucfvq25.yug` and the archive name is is `vwave.archive`:

```
%localappdata%/RadioExt-Helper/working/
├── 4ucfvq25.yug/
│   └── vwave.archive
├── imported/
│   └── ...
└── tools/
    └── ...
stagingFolder/
└── icons/
    └── vwave.archive
```

3. Unpack the `.archive` file to this temporary directory.
    - `unbundle -p <archiveFilePath> -o <temporaryDirectoryPath>`
    - In this example, the command would be: `unbundle -p %localappdata%/RadioExt-Helper/working/4ucfvq25.yug/vwave.archive -o %localappdata%/RadioExt-Helper/working/4ucfvq25.yug/`
        - This would "unpack" the archive into it's raw files.

▶ Directory Tree After Step 3

Assuming `<tempFolderName>` is `4ucfvq25.yug` and the archive name is is `vwave.archive`:

```
%localappdata%/RadioExt-Helper/working/
├── 4ucfvq25.yug/
│   ├── vwave.archive
│   └── base/
│       └── icon/
│           ├── vwave.xbm
│           └── vwave.inkatlas
├── imported/
│   └── ...
└── tools/
    └── ...
stagingFolder/
└── icons/
    └── vwave.archive
```

4. Create a station folder for the imported archive:
    - Similar to creating an icon, we will create a folder under the `imported` directory to store the extracted `.png` for the station.

▶ Directory Tree After Step 4

Assuming `<tempFolderName>` is `4ucfvq25.yug`, the archive name is `vwave.archive`, and the station name is `VWave FM`:

```
%localappdata%/RadioExt-Helper/working/
├── 4ucfvq25.yug/
│   ├── vwave.archive
│   └── base/
│       └── icon/
│           ├── vwave.xbm
│           ├── vwave.inkatlas
│           └── vwave.png
├── imported/
│   └── VWave FM/
└── tools/
    └── ...
stagingFolder/
└── icons/
    └── vwave.archive
```

5. Convert the extracted texture file to a `.png` file.
    - `export --uext png -p %localappdata%/RadioExt-Helper/working/4ucfvq25.yug/ -o %localappdata%/RadioExt-Helper/working/4ucfvq25.yug/`
    - We'll copy the extracted `.png` file to the station's imported folder and rename the file with a unique GUID.
    - Then, we'll assign this file to the station's custom data so we can quickly reference it later, if needed:
        - CustomData["iconFile"] = `vwave.archive`
        - CustomData["iconFileHash"] = `<SHA256 hash of vwave.archive>`
        - CustomData["iconGuid"] = `d4557a89-d8bd-41bb-94c1-73545c32e31c`

▶ Directory Tree After Step 5

Assuming `<tempFolderName>` is `4ucfvq25.yug`, the archive name is `vwave.archive`, and the station name is `VWave FM`:

```
%localappdata%/RadioExt-Helper/working/
├── 4ucfvq25.yug/
│   ├── vwave.archive
│   └── base/
│       └── icon/
│           ├── vwave.xbm
│           ├── vwave.inkatlas
│           └── vwave.png
├── imported/
│   └── VWave FM/
│       └── d4557a89-d8bd-41bb-94c1-73545c32e31c.png
└── tools/
    └── ...
stagingFolder/
```

```
└── icons/
    └── vwave.archive
```

Important Paths

1. StagingIconsPath -> will always be in the staging folder under `icons`.
2. WorkingDirectory -> `%localappdata%/RadioExt-Helper/working`
3. Images Import Directory -> `%localappdata%/RadioExt-Helper/working/imported`
    - Create subfolders for each station using the display name of the station.
4. InkAtlasExe -> Embedded in CRA assembly: `RadioExt_Helper.tools.InkAtlas.generate_inkatlas.exe`
5. WolvenKitExe -> Downloaded from URL to a zip file within `%localappdata%/RadioExt-Helper/working`. This zip file is then extracted to a temporary directory; the file `WolvenKit.CLI.exe` **MUST** exist within this temporary directory after extraction.