🚶 Walkthrough: Implementing a plugin data parser

Implementing a native plugin data parser can be easy. Below is a walkthrough for implementing a simple effect, `Fruity Balance`.

> **Note**
>
> Prequisites
>
> The steps ahead assume that you have an understanding of how binary data types (C's integral types, in this context) work along with a basic understanding of Python itself.

1. Note the parameters exposed by the plugin

Also take note of the **order** in which they occur. Here its **Balance** followed by **Volume**.

2. Prepare a test FLP

Create an empty new FLP, add a **Fruity Balance** to one of the insert slots.

Save this FLP as `fruity-balance.flp`.

3. Getting the plugin data

Since this is an **empty** FLP, with no other plugins loaded, you can simply access the plugin data by,

```python
import pyflp
from pyflp.plugin import PluginID

# Parse the FLP file into a project
project = pyflp.parse("fruity-balance.flp")

# Collect all the events as a dict of ID to event
events = project.events_asdict()

# Get the first plugin data event - the Fruity Balance one itself
plugin_data_event = events[PluginID.Data][0]

# Get the raw data and convert it to a tuple of 8-bit unsigned integers
data = tuple(bytearray(plugin_data_event._struct))
print(data)
```

4. Observe and analyze the output

▶️ You will get an output like this:

```
(0, 0, 0, 0, 256, 0, 0, 0)
```

That's a total of **8 bytes** worth of data for **2 knobs**.

FL Studio *generally* uses 4 bytes for most type of data, so let's assume each knob takes **4 bytes**.

Now compare it with the **positions** of the knobs in Fruity Balance.

‼ Suddenly the data above makes sense.

How? Let me explain.

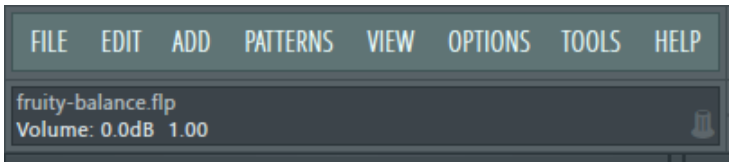
- **Balance** knob is at 12 o' clock
- **Volume** knob is somewhere at 80% of its maximum.

Now convert the 8-bit integer tuple into a two 32-bit integer tuple. We get the values `0` and `256` respectively. So, now we know, that **Balance** is 0 (because its centred) and **Volume** is at 256. Also, since we didn't modify them at all, these are the **default** values.

😅 Success! We cracked it!

5. Exercise: Find out the minimum and maximums (optional, but recommended)

By rotating the knobs to their extremes and following steps 3-4 again, you can find out the minimum and maximums of each knob.

> **Hint**
>
> One very important place for finding out the extremes is the hint panel.
>
> 
>

6. Writing the plugin code

ℹ️ All plugins are implemented in the `pyflp.plugin` module.

> **Note**
>
> Take care to follow the naming conventions as shown below.

Begin with writing the code for the plugin event:

```
class FruityBalanceEvent(StructEventBase):
    STRUCT = c.Struct("pan" / c.Int32ul, "volume" / c.Int32ul).compile()
```

> **Note**
>
> What is `c.Struct`?
>
> PyFLP uses the `construct` library to define and binary structures. Its a fairly simple to understand declarative binary parser creator.
>
> > **Tip**
> >
> > Call `construct.Struct.compile()` to get a faster version of the "Struct". Check <https://construct.readthedocs.io/en/latest/compilation.html> for more information.

Now create a model for the event we just created in the same module:

```
class FruityBalance(_PluginBase[FruityBalanceEvent]):
    pan = _PluginDataProp[int]()
    volume = _PluginDataProp[int]()
```

You don't need to worry about `_PluginBase` and `_PluginDataProp`. They are implementation-level details, you don't *generally* need to worry about.

Derive our newly create `FruityBalance` from `_IPlugin` and implement it:

> **Important**
>
> Don't forget to do this. Otherwise the event will not be parsed.

```
class FruityBalance(_PluginBase[FruityBalanceEvent], _IPlugin):
    INTERNAL_NAME = "Fruity Balance"
    pan = _PluginDataProp[int]()
    volume = _PluginDataProp[int]()
```

> **Note**
>
> Use FLPEdit to find out `INTERNAL_NAME` of a plugin.

🎉 And that's basically it. The implementation is complete! Now all we need to do is glue `FruityBalanceEvent` and `FruityBalance` to the effect slot's `pyflp.mixer.Slot.plugin` attribute.

7. Glue the implementation to `pyflp.mixer.Slot`:

Import our newly created classes in `pyflp.mixer` and add an entry to `pyflp.mixer.Slot.plugin` like so:

```
plugin = PluginProp(
     {
     FruityBalanceEvent: FruityBalance,
     ...
     }
 )
```