

How to add charts to PicDat

Adding search keys to PicDat so that they are included in the analysis and create new charts might be very easy and very hard as well. This depends on how the respective data is constructed and whether this is similar to data which is already considered. This piece of text tries to explain which options you have doing so.

In any case you need to go into the python source code. You will find two packages, called `perfstat_mode` and `asup_mode`. As PerfStat files and ASUP files are very different, search keys have to be established for each mode independently. Both modes work with container classes. The docstrings for the classes and the modules containing them give further description. Read them to understand which container is the right one for your concern.

The general rule is that most container modules are holding constant lists (`UPPER_CASE_LETTERS`) at the top which define the search keys in some way. The format is described locally in the code. If a key about the chart you wish to include can be formulated like the keys in one of those lists, you are lucky. Just append it to the respective list. If not, you have to go deeper into the code.

For PerfStat mode

In `perfstat_mode`, there are three different classes holding search keys. As PerfStat data is pretty unhomogenous, the approach of gaining data about the keys is fundamentally different among the three classes. You find the classes inside the modules `per_iteration_container`, `sysstat_container` and `statit_container`.

per iteration container

The `per_iteration_container` has four key lists (and one additional single key). The lists are about the four different objects 'aggregate', 'processor', 'volume', and 'lun'. If you want an additional chart for one of those objects, just append to the respective list. If you want a chart for similar data, but for a different object, you need to create a new list. Furthermore, you need to append at least to following methods: `init` (declare an empty list for the new chart's table), `process_per_iteration_keys` (call `process_object_type` for your object), `rework_per_iteration_data` (flat your tables if it is not empty), and `get_labels` (append appropriate information to identifiers, units and `is_histo`). Pay attention that the order in the `flat_tables` list from `rework_per_iteration_data` must be equivalent to the orders of the lists in `get_labels`.

sysstat container

The `sysstat_container` has three key lists. The `sysstat` blocks in PerfStats are table-like structures; PicDat reads some of the columns. The considered columns are categorized after the units belonging to them. All considered columns having either the unit '%' or 'MB/s' or no unit at all. Each of the three key lists in `sysstat_container` is about one of these three categories. The column names are defined in the keys. Values for the same unit can be visualized in the same chart, so each key list is collecting data for exactly one chart. Some of the keys are tuples. This is because the `sysstat` table headers are spread over two lines and sometimes several key words are necessary to determine a column. To better understand why the keys have the format they have, just look into a PerfStat's `sysstat` block. If you want to evaluate more columns from a `sysstat` block, try if it makes sense to append to one of the existing key lists which will add a graph line to the respective chart. Otherwise, a more extensive adaption of PicDat is necessary.

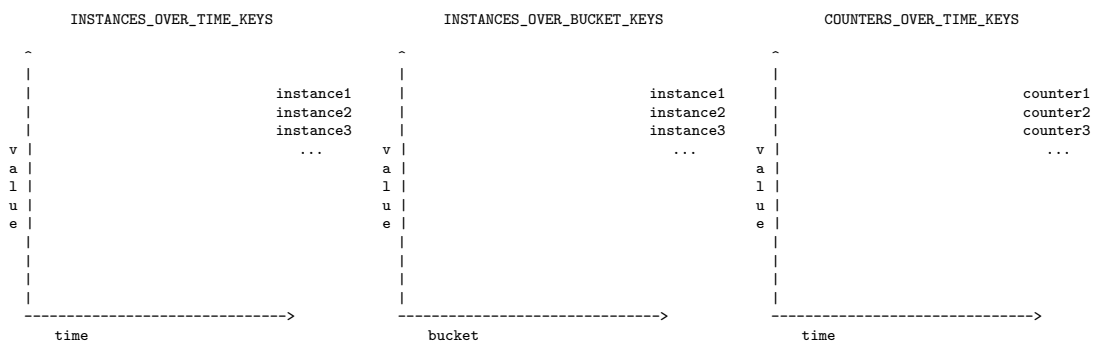
statit container

The `statit_container` is responsible for collecting data for a single chart from a specific PerfStat block. This is not really flexible; the container does not even have any kind of key list. So there is no true chance to easily add a new search key to this container.

For ASUP mode

Data from ASUP is much more homogenous than PerfStats, so life is somewhat easier with them. As the package `asup_mode` provides support for two different kinds of input data – XML and JSON – there are two container classes. Their names are, unsurprisingly, `XmlContainer` inside the `xml_container` module and `JsonContainer` inside the `json_container` module. Both modules have again several lists with search keys. Fortunately, the key lists' structures in both modules are exactly the same. Only the content differs a bit. So, the following applies to all container modules.

The key lists `INSTANCES_OVER_TIME_KEYS`, `INSTANCES_OVER_BUCKET_KEYS`, and `COUNTERS_OVER_TIME_KEYS` are each for a different kind of charts. The difference is about the charts' x-axis and whether the single graph lines in one chart are about different instances or different counters. Again, read the list descriptions inside the code. Additionally, the following drawing might help your imagination:



If one of the lists fits your needs, append your key and your chart should be drawn. Otherwise, you will need to reimplement most of the methods.

When you add keys to the `json_container`, be aware that you probably **also need to change the `config.yml` for the `convert_ccma_to_json` script!**. Otherwise, PicDat will search for your new objects and counters in your JSON files, but has no chance to find anything, because you didn't ask Trafero to convert this information at all.

Finally, it is to mention, that there is a third container module named `hdf5_container` inside the `asup_mode` package. It is deprecated, but it still works. You can manipulate its keys in the same way as them in the other two `asup` container modules.