Object Names

Object names are only to be used in the UI, for the user to be able to identify objects in lists[1]. Kst internals should not use object names for identification. So, for example, equations keep a broken down version of the equation, with the data objects held as *ptr*s. When populating the equation dialog, the names are re-inserted. This way, object names can change, and everything still works.

Object names need only be unique to avoid ambiguity with the user. So there can be a primitive (eg, vector), a data object (eg, a psd), a relation (eg, a curve), and a view object (eg, a plot) all named `GYRO1`. But there can't be two primitives named `GYRO1` (eg, a vector and a scalar) because that would be ambiguous in an equation.

Object names can be in one of two states:

1. auto generated
2. manually set.

If auto generated, then the name can change if its status changes. If manually set, the name becomes static unless changed manually.

Autogenerated names should not have the type embedded in them (eg, no C- for curves, or V2- for vectors, ***unless*** there would be a collision, in which case a -<typeID><int> will be appended to the name. If types are needed by the UI (eg, in the data manager) they should be explicitly provided separate from the object name.

Automatic DataVector names:

- If there is only one primitive of <fieldname> use:
  <Field name>
  `eg. GYRO1`

- If there are more than 1 GYRO1 primitives, but they are from different data sources use:
  <Data Source>/<Field Name>
  `eg, /data/flightdir/GYRO1`

- If there are more than 1 GYRO1 primitives and they are from the same data source, but have different lengths or sample properties:
  <Field Name>(<start frame>:<num frames>[:<framesPerSample>[:a]])
  `eg, GYRO1(100:1000)` or, if needed `GYRO1(100:1000:2:a)`

- If there are more than 1 GYRO1 primitives, and some are from different data sources, and some have different lengths use:

1A possible exception to this is in .kst files, but I think we should use a different, static, not user visible unique ID here.

<Data Source>/<Field Name>(<start frame>:<num frames>[:<framesPerSample>[:a]])
`eg, /data/flightdir/GYRO1(100:1000)`

Remaining collisions are resolved by appending a “-V<int>” (eg, “-V1”) to the field.

Automatic curve names:

If there is only one curve using <yvector> as the y vector use:
<yvectorname>
eg, `GYRO1`

If there are more than one curve using <yvector> as the Y vector use:
“<yvectorname> vs <xvectorname>”
eg, `GYRO1 vs INDEX`

If there would be a name collision with another relation, append a -C<int>.
eg, `GYRO1 vs INDEX-C2`

Automatic Image names:

Same rules as curves, except use -I as a typeId.

Data object names:

When it makes sense, use the primary vector name.
If this name would give a collision with another data object, append -<TYPEID><int>.

Data object	**TypeId**	**Eg**
Spectrum	P	`GYRO1-P1`
Histogram	H	`GYRO1-H1`
Spectrogram	S	`GYRO1-S1`

Plugin object names:

Use the same rules as data object names, but each plugin might want to provide a typeID. A low pass filter with a name clash might be
eg `GYRO1-LP1`.

How to chose the uniqueness index:

In names which have to resort to -<typeID><int>, the only requirement on <int> is that it is different than any other object with the same prefix. Rather than insisting that the index start at 1 for the first example, and increase, as GYRO-P1, GYRO-P2, GYRO-P3, in parallel with BOLO-C1, BOLO-C2, BOLO-C3 ..., each object can get a unique integer at creation time which it keeps forever, ready to be used if a name collision ever appears, and which it can never lose or change. So, for the above example, you might have GRYO-C1, GYRO-C2, BOLO-

C3, GYRO-C4, BOLO-C5, BOLO-C6 if that were the order of creation. If the suffix is not needed, it is not shown.