Overview of Fault System Dag Markup Language (fdagml)

Syntax

The Fault System DAG is defined in XML using a syntax defined by the following *tags*:

- **<dag>** The root node - all subsequent definitions are contained in this scope.
- **<range>** Establish a looping relationship over enclosed scopes.
- **<mp>** Identify a monitor point to be used by the fault system.
- **<mp_ref>** Tag to reference a predefined monitor point in possibly multiple places (e.g. input based monitor points for baseline flagging definitions).
- **<gather>** Define a collection of monitor points to be referenced later.
- **<gather_ref>** Tag to reference a gathered collection.
- **<transient_preds>** Identify monitor points (or gathers) to be selectively turned on or off based on predetermined criteria.
- **<attrs_scope>** Define scopes of common attributes.
- **<bf_output>** Define the final blanking flagging output structure using all previous tags. This is the final output from processing a fdagml file.

In turn, each of these tags contains multiple *attributes* that define the behavior and/or identity of the tag. It should be noted that the only data contained in these tags is other tags.

<range>

The <range> tag has four possible attributes: `var`, `first`, `last` and `end`. The `var` attribute specifies a variable name to be referenced within contained scopes using parentheses. The range being looped over is specified by the `first` and `last` attributes. For example:

```
<range var="bandNoVar" first="1" last="4">
  <range var="inputNoVar" first="1" last="15">
      <!-- Add the downconverter state MP -->
      <mp canon="Sldc.Band(bandNoVar).Input(inputNoVar).state"/>
  </range>
</range>
```

This forms a double loop over bands 1-4 and all inputs to add the monitor point specified by the <mp> tag. The `last` attribute should be replaced with an `end` attribute if it references another `range` attribute variable from an enclosing scope rather than a literal number.

<mp> and <mp_ref>

The `<mp>` tag defines a monitor point to be monitored by the fault system. The canonical name of the monitor point is specified via the `canon` attribute:

```
<mp canon="Control.Subarray1.scriptState"/> <!—Fully Qualified -->
<mp canon=”Ovro(ovroAnt).Cryo.plate4kTemp"/> <!-- Inside range -->
```

Canonical names can also be created using the `canon_last` attribute enclosed inside another scope using the `canon_begin` or `canon_add` attributes. Both **<attrs_scope>** and **<gather>** tags (see below) support `canon_begin` and `canon_add`. The values of `canon_begin + canon_add + canon_last` form the fully qualified canonical name.

The `alarm`, `alarm_after` and `effect` attributes specify the behavior of the fault system when a monitor point's range is in error. `alarm` is either `“true”` of `“false”`. `alarm_after` specifies a time in seconds (also must be in quotes). Finally, `effect` must be either `“blank”` or `“flag”`.

Hierarchically embedded `<mp>` tags define root faults with the root faults defined by the innermost scopes and masked faults defined by the outermost scopes. For example, consider the following:

```
<mp canon_last="Receivers.rxState" alarm=”false” effect="flag">
    <mp canon_last="LO.loState" alarm=”false” effect="blank">
        <mp canon_last="LO.yigState" alarm=”false” effect="blank"/>
    </mp>
 </mp>
```

In this example, loState and yigState are progressively prioritized root faults that will cause an rxState fault to be masked if *either monitor point is in error*. This is important to note as it implies that *there is no way* to specify that *both* predecessor monitor points must be in error in order to mask the rxState!

One must also pay close attention to the `alarm` tags on individual monitor points within dependency chains as any monitor point marked with `alarm=”true”` will sound the alarm and appear as an alarm fault on the faults page, regardless of being masked by root faults. This is more of a gui issue than anything as the 'root faults page' actually contains both alarms and root faults. Basically, alarms and root faults are monitored and reported independently.

<mp_ref> tags are used to reference a monitor point previously added with the <mp> tag. This enables one to reuse references to specific monitor points. Note that mp_ref tags do not support the `alarm` or `effect` attributes, as these can only be defined once when the monitor point is introduced with the `<mp>` tag.

<gather> and <gather_ref>

These nodes define a collection of mps that can be referenced independent of a specific monitor point name. Inside carma.fdagml they are used to separately collect antenna based monitor points and correlator input based monitor points (note the distinction between antennas and inputs). This in turn simplifies the blanking flagging output definitions and allows these collections to be used in multiple places with <gather_ref>. The primary attribute for both is `name`, which must specify a unique name or, in the case of `<gather_ref>`, a predefined name. These tags also support `canon_begin` and `canon_add` attributes described above.

<transient_preds>

This tag defines `<mp>` and `<gather>` definitions that are conditionally used by the fault system based on conditions established dynamically by the observer. Currently only two conditions are supported via the `enableIf` attribute: `"driveBlankingOn"` and `"slCorrRfOn"`. This is effectively how dynamic dag changes are implemented.

<attrs_scope>

This tag allows one to define large blocks of elements with identical properties or behavior. Supported attributes are `alarm_default="true"|"false"`, `alarm_after_default="10.0"`, `canon_begin="MonitorSystemContainer"` and `canon_add="MonitorSubsystemContainer"`. Any tag contained within an attribute scope will inherit the attributes of that scope.

<bf_output>

Aside from stand-alone alarm monitor points, the <bf_output> tag defines the end blanking flagging output of the fault system. Each bf_output tag is internally associated with a band and baseline specific monitor point to specify blanking and flagging on that band-baseline combination. The association is defined with the 'band', 'input_a' and 'input_b' attributes that are then internally mapped to the blanking output monitor point.

<bf_output> tags may contain <mp>, <mp_ref> and/or <gather> and <gather_ref> tags, which themselves may contain <transient_preds> and <attrs_scope> tags and <mp> tags.

Additional Shorthands

Since monitor system antenna subsystems do not have regular names (e.g they are named Ovro and Bima rather than Carma) Tom introduced an additional shorthand to easily loop over antennas. The phrase `(antCommon:carmaAntNo)`, where carmaAntNo is a variable defined inside a `<range>` tag will be expanded to 'OvroX' or 'BimaX' depending on the value of `carmaAntNo` at the time.

Brief XML Reference

<tag attribute="value"/> defines a stand-alone XML *element* named 'tag' with a single *attribute* named 'attribute' set to 'value'. Tag and attributes may be arbitrarily named. Values must be enclosed in quotations even if they represent a numeric value. XML elements can contain *data* using the following syntax: <tag attribute="value">DATA</tag>. Note that all elements must be terminated via an explicit </tag> if not of the form <tag attr="val"/>. The only data contained by elements in the Fault Dag Markup Language are other elements.