Unified Trigger Configuration

Version 1.0

P. Toale

December 13, 2005

This document describes the current approach to unifying the configuration aspect of DAQ triggering across components. This scheme has been developed to support triggering in any DAQ component and to allow for a single set of trigger definitions for the DAQ as a whole.

1 A Trigger

A Trigger is simply an encapsolation of an algorithm. It takes inputs, performs some computation, and occasionally produces output. The computation will, in general, require some number of configurable parameters, such as a threshold. In addition, each trigger can suggest a set of readout instructions for the EventBuilder. These consist of a set of readout elements, specifing both a detector element and a time window. While both the number and type of parameters and readouts can vary from one algorithm to another, there are three quantities that are common to all, and server to identify the trigger:

- `triggerType`: Maps to the name of a java class representing the trigger (`triggerName`), which must exist in `icecube.daq.trigger.algorithm`.
- `triggerConfigId`: Identifies the set of values for each parameter and readout of the trigger.
- `sourceId`: Specifies the DAQ component running the trigger. Valid sourceId's can be found in SourceIdRegistry in the DAQ triggerUtil project.

The complete specification of the configuration of a trigger is captured in the following xml schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="activeTriggers" type="ActiveTriggersType"/>
    <xs:complexType name="ActiveTriggersType">
        <xs:sequence>
            <xs:element name="triggerConfig" type="TriggerConfigType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TriggerConfigType">
        <xs:sequence>
            <xs:element name="triggerType" type="xs:int"/>
            <xs:element name="triggerConfigId" type="xs:int"/>
            <xs:element name="sourceId" type="xs:int"/>
            <xs:element name="triggerName" type="xs:string"/>
            <xs:element name="parameterConfig" type="ParameterConfigType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="readoutConfig" type="ReadoutConfigType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ParameterConfigType">
        <xs:sequence>
            <xs:element name="parameterName" type="xs:string"/>
            <xs:element name="parameterValue" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ReadoutConfigType">
        <xs:sequence>
            <xs:element name="readoutType" type="xs:nonNegativeInteger"/>
            <xs:element name="timeOffset" type="xs:int"/>
            <xs:element name="timeMinus" type="xs:int"/>
            <xs:element name="timePlus" type="xs:int"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

This structure allows for one or more active triggers, each enclosed in a `<triggerConfig>` element. Each trigger can contain zero or more parameters (`<paramererConfig>`) consisting of a `parameterName`, `parameterValue` pair. The readouts (`<readoutConfig>`), of which there can zero or more, include a `readoutType`, `timeOffset`, `timeMinus`, and `timePlus`. The readoutType's include Global, Inice, Icetop, String, and Module (these are specified in IReadoutRequestElement in the DAQ triggerUtil project). The times allow for an offset from the current trigger time and a time window about that offset (all times are signed quantities in units of nanoseconds).

2 Database Organization

The trigger configuration database is arranged in nine tables as shown in Figure 1. These tables hold configuration data for all triggers regardless of DAQ component. The following list provides details about each of the tables:

TriggerConfiguration This table sits at the top of the configuration scheme. It maps a single configuration id onto a set of trigger id's, one for each

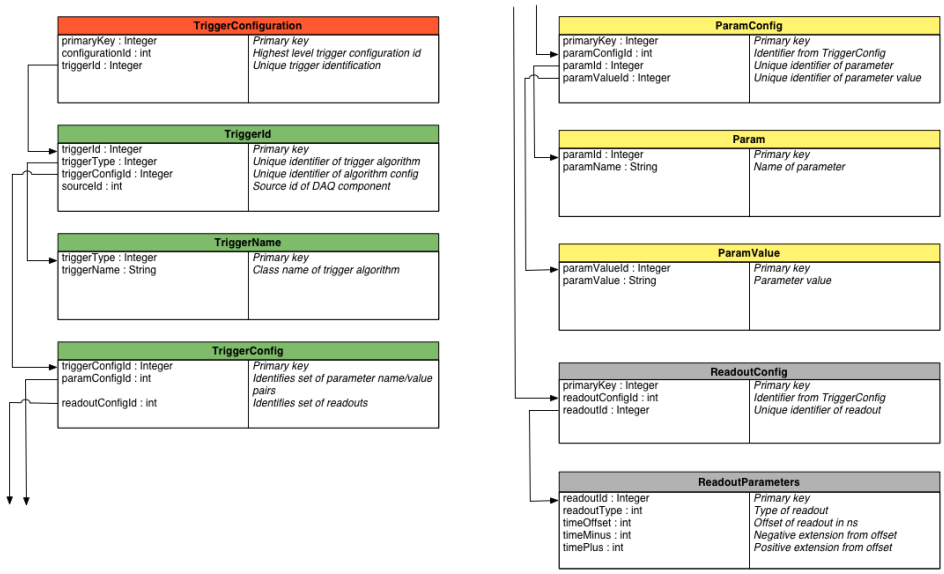

Figure 1: Configuration Tables.

active trigger in the system. The triggerId's are foreign keys into the TriggerId table.

```
CREATE TABLE TRIGGERCONFIGURATIONBEAN (
        PrimaryKey INTEGER NOT NULL,
        ConfigurationId INTEGER,
        TriggerId INTEGER,
        CONSTRAINT PK_TRIGGERCONFIGURATIONBEAN PRIMARY KEY (PrimaryKey)
);
```

TriggerId Each trigger is uniquely identified by three quantities: triggerType, triggerConfigId, and sourceId. This table maps these onto a single quantity, triggerId. The triggerType is a foreign key to TriggerName, and triggerConfigId is a key in TriggerConfig.

```
CREATE TABLE TRIGGERIDBEAN (
        TriggerId INTEGER NOT NULL,
        TriggerType INTEGER,
        TriggerConfigId INTEGER,
        SourceId INTEGER,
        CONSTRAINT PK_TRIGGGERIDBEAN PRIMARY KEY (TriggerId),
        CONSTRAINT FK_TRIGGERNAMEBEAN FOREIGN KEY (TriggerType)
                REFERENCES TriggerNameBean (TriggerType),
        CONSTRAINT FK_TRIGGERCONFIGBEAN FOREIGN KEY (TriggerConfigId)
                REFERENCES TriggerConfigBean (TriggerConfigId)
);
```

TriggerName This table lists all known triggers by name and associates a unique identifier, triggerType, with each. This name must correspond to class name in the package: icecube.daq.trigger.algorithm.

```
CREATE TABLE TRIGGERNAMEBEAN (
        TriggerType INTEGER NOT NULL,
        TriggerName VARCHAR(256),
        CONSTRAINT PK_TRIGGERNAMEBEAN PRIMARY KEY (TriggerType)
);
```

TriggerConfig The triggerConfigId identifies a specific configuration of a trigger. It maps to a unique pair of paramConfigId and readoutConfigId, which are columns in ParamConfig and ReadoutParameters, respectively.

```
CREATE TABLE TRIGGERCONFIGBEAN (
        TriggerConfigId INTEGER NOT NULL,
        ParamConfigId INTEGER,
        ReadoutConfigId INTEGER,
        CONSTRAINT PK_TRIGGERCONFIGBEAN PRIMARY KEY (TriggerConfigId)
);
```

ParamConfig For each paramConfigId there can be many pairs of paramId and paramValueId, one for each parameter of the trigger algorithm. These are foreign keys in the Param and ParamValue tables, respectively.

```
CREATE TABLE PARAMCONFIGBEAN (
        PrimaryKey INTEGER NOT NULL,
        ParamConfigId INTEGER,
        ParamId INTEGER,
        ParamValueId INTEGER,
        CONSTRAINT PK_PARAMCONFIGBEAN PRIMARY KEY (PrimaryKey),
        CONSTRAINT FK_PARAMBEAN FOREIGN KEY (ParamId)
                REFERENCES ParamBean (ParamId),
        CONSTRAINT FK_PARAMVALUEBEAN FOREIGN KEY (ParamValueId)
                REFERENCES ParamValueBean (ParamValueId)
);
```

Param This table lists all known parameters by name and associates a unique identifier, paramId, with each.

```
CREATE TABLE PARAMBEAN (
        ParamId INTEGER NOT NULL,
        ParamName VARCHAR(256),
        CONSTRAINT PK_PARAMBEAN PRIMARY KEY (ParamId)
);
```

ParamValue This table lists all known parameter values as a String. These should be formatted such that code like, Integer.parseInt(paramValue), produces the proper numeric value.

```
CREATE TABLE PARAMVALUEBEAN (
        ParamValueId INTEGER NOT NULL,
        ParamValue VARCHAR(256),
        CONSTRAINT PK_PARAMVALUEBEAN PRIMARY KEY (ParamValueId)
);
```

ReadoutConfig For each readoutConfigId there can be many readoutIds, one for each readout of the trigger. These are foreign keys in the ReadoutParameters table.

```
CREATE TABLE READOUTCONFIGBEAN (
        PrimaryKey INTEGER NOT NULL,
        ReadoutConfigId INTEGER,
        ReadoutId INTEGER,
        CONSTRAINT PK_READOUTCONFIGBEAN PRIMARY KEY (PrimaryKey),
        CONSTRAINT FK_READOUTPARAMETERSBEAN FOREIGN KEY (ReadoutId)
                REFERENCES ReadoutParametersBean (ReadoutId),
);
```

ReadoutParameters This table holds all of the readout configuration data. There will be one row with a given readoutId for each defined readout request element. Each element has a readoutType, timeOffset, timeMinus, and timePlus. The readoutType is matched to the constants defined in the IReadoutRequestElement interface. The timeOffset is the offset, in nanoseconds, from a specific TriggerRequest. The timeMinus and timePlus define the readout window relative to the timeOffset.

```
CREATE TABLE READOUTPARAMETERSBEAN (
        ReadoutId INTEGER NOT NULL,
        ReadoutType INTEGER,
        TimeOffset INTEGER,
        TimeMinus INTEGER,
        TimePlus INTEGER,
        CONSTRAINT PK_READOUTPARAMETERSBEAN PRIMARY KEY (ReadoutId)
);
```

3 Configuration System

The trigger configuration consists of a source of configuration information, a configuration object, and a trigger builder, as shown in Figure 2. The source can either be valid xml files or the DAQ database, and configuration data can be transfered between the two.

The configuration object can be created from either data source. The trigger configuration object is generated, with the help of JAXB, from the xml schema discussed above.

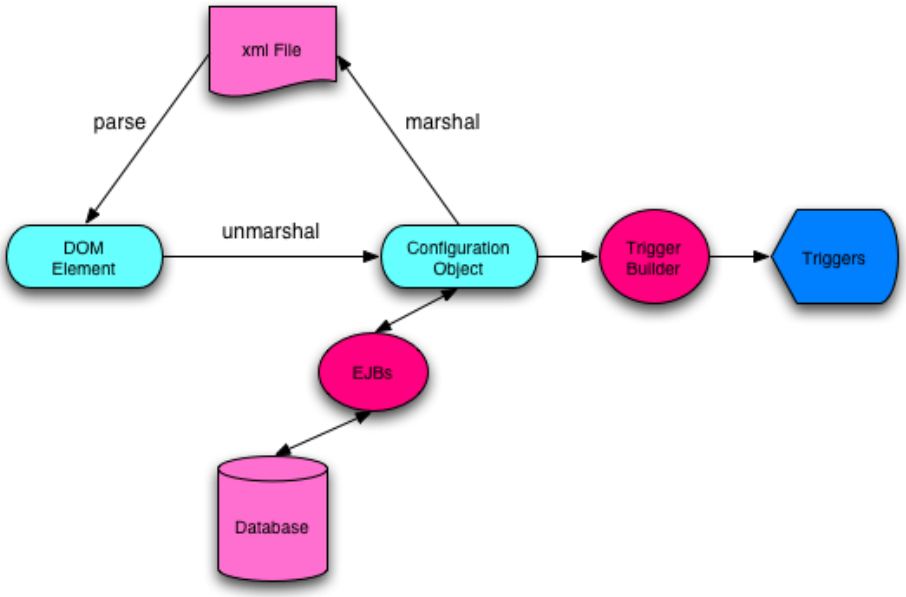

Figure 2: Configuration System.

The xml schema shown above, with the help of JAXB, descibes a trigger configuration object through several auto-generated interfaces and implementing classes. The configuration object is used by a trigger builder to create and configure the actual trigger objects.