............................................................................................

**告警数据结构FireData**：

type FireData struct {

    Labels    map[string]string `json:"labels"`

    Mid       string            `json:"mid"` // 配置 ID，如果db压力过大，可以直接把配置传递过来

    Entity    string            `json:"entity"`

    Timestamp int64             `json:"timestamp"`

    Value     interface{}       `json:"value"` // float64 的科学计数法

    Severity  int64             `json:"severity"`

    Idc       string            `json:"idc"`

    StrValue  string            `json:"str\_value"` // 可能是用户自定义的字符串报警

    Model     Model             `json:"model"`

    Rule      Rule              `json:"rule"`

}

FireData子数据结构Model：

type Model struct {

    // 上面是通用的参数

    Id               bson.ObjectId           `bson:"\_id,omitempty" json:"id"`                      // ID

    Name             string                  `bson:"name" json:"name"`                             // 规则名

    Matcher          Matcher                 `bson:"matcher" json:"matcher"`                       // 自主配置用这个

    MatcherId        string                  `bson:"matcher\_id" bson:"matcher\_id"`                 // 数据match部分的配置id（这里暂时是 prometheus.rule\_name，api需要注意，这里是严格一对一的）

    Tag              []string                `bson:"tag" json:"tag"`                               // 规则标签

    IsSafety         bool                    `bson:"is\_safety" json:"is\_safety"`                   // 是否是保底规，默认false

    Status           int64                   `bson:"status" json:"status"`                         // 当前 model 状态，0为正常，1为禁用

    PromEvalInterval int64                   `bson:"prom\_eval\_interval" json:"prom\_eval\_interval"` // rule 拉取数据的周期

    Entity           []string                `bson:"entity" json:"entity"`                         // 采用哪些tag来生成issue的唯一键，聚合用

    Info             []string                `bson:"info" json:"info"`                             // 用哪些tag生成 info.entity，运维侧默认是 instance，租户侧五花八门，默认用所有label

    DisplayName      []string                `bson:"display\_name" json:"display\_name"`             // 采用哪些tag来生成issue展示名字

    StrategyId       []string                `bson:"strategy" json:"strategy"`                     // 策略信息，这里留坑一个策略信息，有报警策略，就不适用自定义报警

    CustomStrategy   map[int64]EventStrategy `bson:"custom\_strategy" json:"custom\_strategy"`       // 自定义简单报警策略，简单报警无需模板

    SilenceTime      int64                   `bson:"silence\_time" json:"silence\_time"`             // 静默时间/重复通知间隔

    RepeatTimes      int64                   `bson:"repeat\_times" json:"repeat\_times"`             // 重复通知次数                   //重复次数，默认0

    RecoverNotify    int64                   `bson:"recover\_notify" json:"recover\_notify"`         // 恢复是否通知，0否 1是，默认 0

    Owner            string                  `bson:"owner" json:"owner"`                           // 拥有者，运维侧为项目，租户侧为用户

    Ctime            int64                   `bson:"ctime" json:"ctime"`

    Mtime            int64                   `bson:"mtime" json:"mtime"`

    // 留坑的参数

    // 类型留坑

    // 1 代表jyw的prometheus数据报警

    // 10 代表从api创建的报警，不需要进行聚合之类的操作

    Type int64 `bson:"type" json:"type"` // 这就用上这个参数了

    // 分享留坑，大概率用不上

    ShareProject string `bson:"share\_project" json:"share\_project"`

    // 当前运维监控需要的参数

    Project    string   `bson:"project" json:"project"` // 就是jyw的group

    Idc        string   `bson:"idc" json:"idc"`

    AlarmTopic string   `bson:"alarm\_topic" json:"alarm\_topic"` // prometheus.rules 的唯一键

    AlarmType  string   `bson:"alarm\_type" json:"alarm\_type"`   // jyw 前端页面的报警类型（主机、进程、链路、端口、拨测等）

    Relations  []string `bson:"relations" json:"relations"`     // 此告警配置被关联到了相关性配置

}

FireData子数据结构Rule：

type Rule struct {

    Metric           string `bson:"metric" json:"metric"` // 目标指标

    AlarmDescription string `bson:"alarm\_description" json:"alarm\_description"`

    AlarmName        string `bson:"alarm\_name" json:"alarm\_name"`

    Expr             string `bson:"expr" json:"expr"`

    For              int64  `bson:"for" json:"for"` // 持续时间

    Severity         int64  `bson:"severity" json:"severity"`

}

..............................................................................

...................................................................................

Model子数据结构EventStrategy

type EventStrategy struct {

    Id        bson.ObjectId `bson:"\_id,omitempty" json:"id"` // ID

    Name      string        `bson:"name" json:"name"`        // 策略名

    Version   int64         `bson:"version" json:"version"`

    Project   string        `bson:"project" json:"project"` // 所属项目 all 表示通用，只有管理员才能设置

    Level     string        `bson:"level" json:"level"`     // 策略等级 level1 level2 level3

    MTime     int64         `bson:"mtime" json:"mtime"`

    IsUpgrade int           `bson:"is\_upgrade" json:"is\_upgrade"` // 是否升级策略

    Policies  []EventPolicy `bson:"policies" json:"policies"`     // 策略集

    // Loopback  int           `bson:"loopback" json:"loopback"`     // 是否开启策略循环

    Owner string `bson:"owner" json:"owner"` // 拥有者，运维侧为项目，租户侧为用户

}

Model子数据结构Matcher：

type Matcher struct {

    Rules []Rule `bson:"rules" json:"rules"`

}

......................................................................................

........................................................................................

EventStrategy子数据结构EventPolicy:

type EventPolicy struct {

    Index      int64       `bson:"index" json:"index"`             // 通知顺序的序号，从0开始。

    AckTimeout int64       `bson:"ack\_timeout" json:"ack\_timeout"` // 确认超时时间，单位为：秒

    Unit       string      `bson:"unit" json:"unit"`               // 确认超时事件单位。仅辅助显示，与ack\_timeout无关。

    TTL        int64       `bson:"ttl" json:"ttl"`                 // 新增通知周期，单位秒，默认 0

    Notice     AlarmNotice `bson:"notice" json:"notice"`           // 告警通知配置，新增部分，后续考虑兼容问题

    WorkConfig AlarmConfig `bson:"work\_config" json:"work\_config"` // 工作时间通知配置

    RestConfig AlarmConfig `bson:"rest\_config" json:"rest\_config"` // 休息时间通知配置

}

...........................................................................................

...............................................................................

EventPolicy子数据结构AlarmNotice:

type AlarmNotice struct {

    Name           string   `json:"name" bson:"name" gorm:"column:name;comment:'模板名称'"`

    Conditions     string   `json:"conditions" bson:"conditions" gorm:"column:conditions;comment:'触发条件（start 告警开始，end 告警恢复）'"`

    StartTime      int64    `json:"start\_time" bson:"start\_time" gorm:"column:start\_time;comment:'开始时间'"` //告警有效期开始时间

    EndTime        int64    `json:"end\_time" bson:"end\_time" gorm:"column:end\_time;comment:'结束时间'"`       //告警有效期结束时间

    Frequency      string   `json:"frequency" bson:"frequency" gorm:"column:frequency;comment:'通知频率'"`

    FrequencyValue string   `json:"frequency\_value" bson:"frequency\_value" gorm:"column:frequency\_value;comment:'通知频率对应值'"`

    Receive        []string `json:"receive" bson:"receive" gorm:"column:receive;comment:'通知对象'"`

    Way            string   `json:"way" bson:"way" gorm:"column:way;comment:'通知方式(短信(sms)，邮件(email)，企业微信(wechat))'"`

    Wechat         string   `json:"wechat" bson:"wechat" gorm:"column:wechat;comment:'企业微信的微信群ID'"`

    Personal       bool     `json:"personal" bson:"personal" gorm:"column:personal;comment:'是否暂停通知个人(yes,no)'"`

    Content        string   `json:"content" bson:"content" gorm:"column:content;comment:'通知模板内容配置'"`

    Creator        string   `json:"creator" bson:"creator" gorm:"column:creator;comment:'创建人'"`

    CreateTime     int64    `json:"create\_time" bson:"create\_time" gorm:"column:create\_time;comment:'创建时间'"`

}

EventPolicy子数据结构AlarmConfig:

type AlarmConfig struct {

    // Time string // 如 all,rest,work,night,daytime 等，后面可以考虑将 rest/work config 做个兼容，存到同一个如上的 Configs 自动里面

    Receivers map[string][]string `bson:"receivers" json:"receivers"` // {"mail": [x, x, x]}

}

...........................................................................................