# RemoteDecition 远程决策系统（GKQE 2.0）架构设计文档

# 架构概览

**RemoteDecition以下简称”RD”**

RD

WebService

RD Socket

RD Database

阿里云CDN

RD

静态服务

RD

管理后台

Client 1

Client 2

Client N

Mac、Win、iOS、Android、其它客户端

Client 1

Client 2

Client N

Web端、小程序等

RD Web Socket

研发、维护、产品、其它业务部门等

# **远程决策（RemoteDecition）工作原理**

* 采用TCP全双工通讯，增量更新方式，确保“远程决策数据”能够在短时间内同步更新到接入端。使接入端本地缓存的“决策数据”与服务端的“决策数据”保持实时同步。
* 接入端向本地“决策数据”递交请求，由接入端预制“策略权重SDK”计算，并最终获得相应的“策略”，最大程度保证“策略”的即时生效性，接入端根据“策略”开关指定功能或进行不同的界面渲染展示。

# 详细设计

## WebService接口服务组成

WebServic Core (基于Golang)

RD Webservice

长连接服务

（基于Golang net）

接口服务（基于iris 框架）

RD 静态资源服务

对外业务（HTTP GET）

监听端口 8080

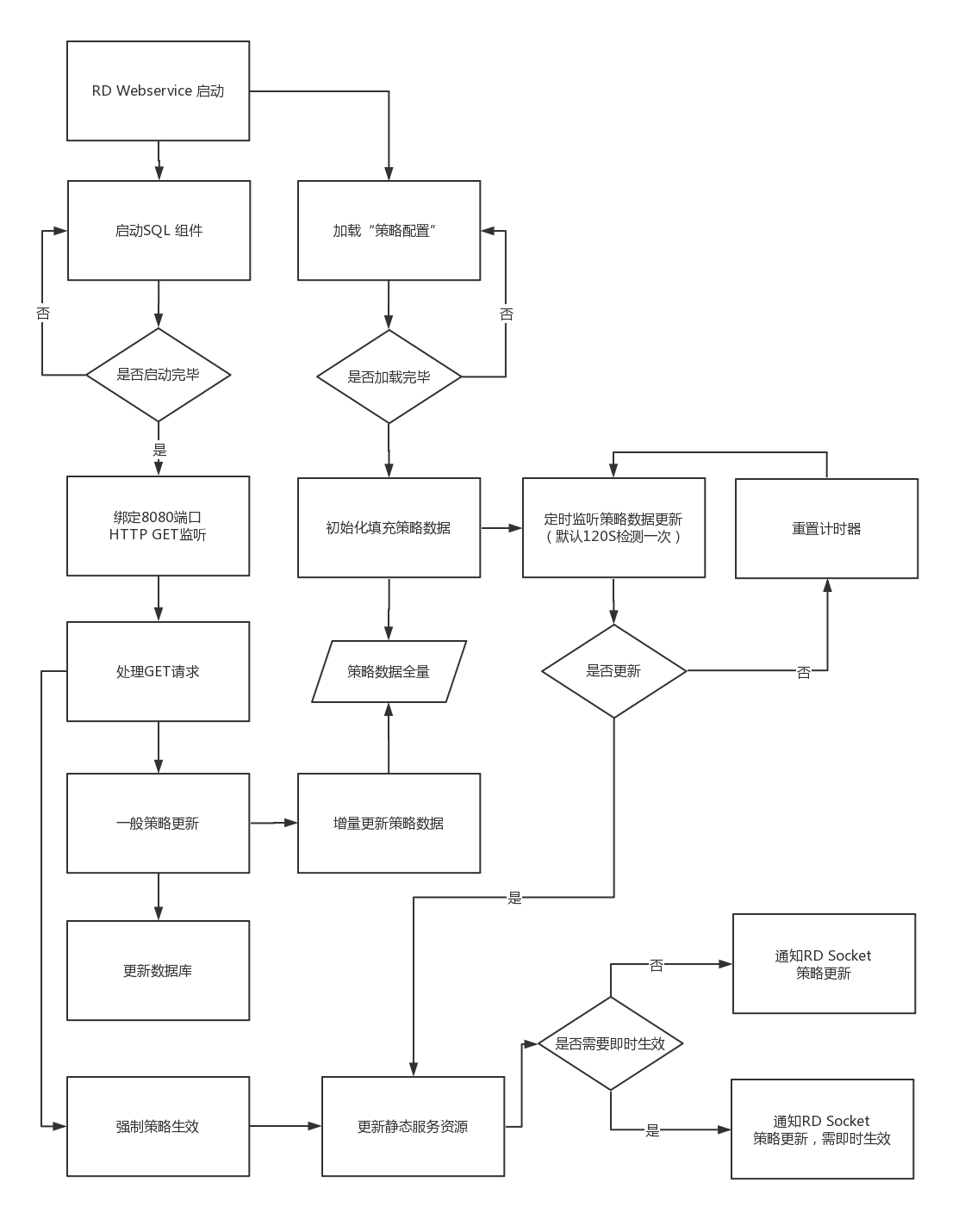
对接RD 信令服务（TCP）

数据库

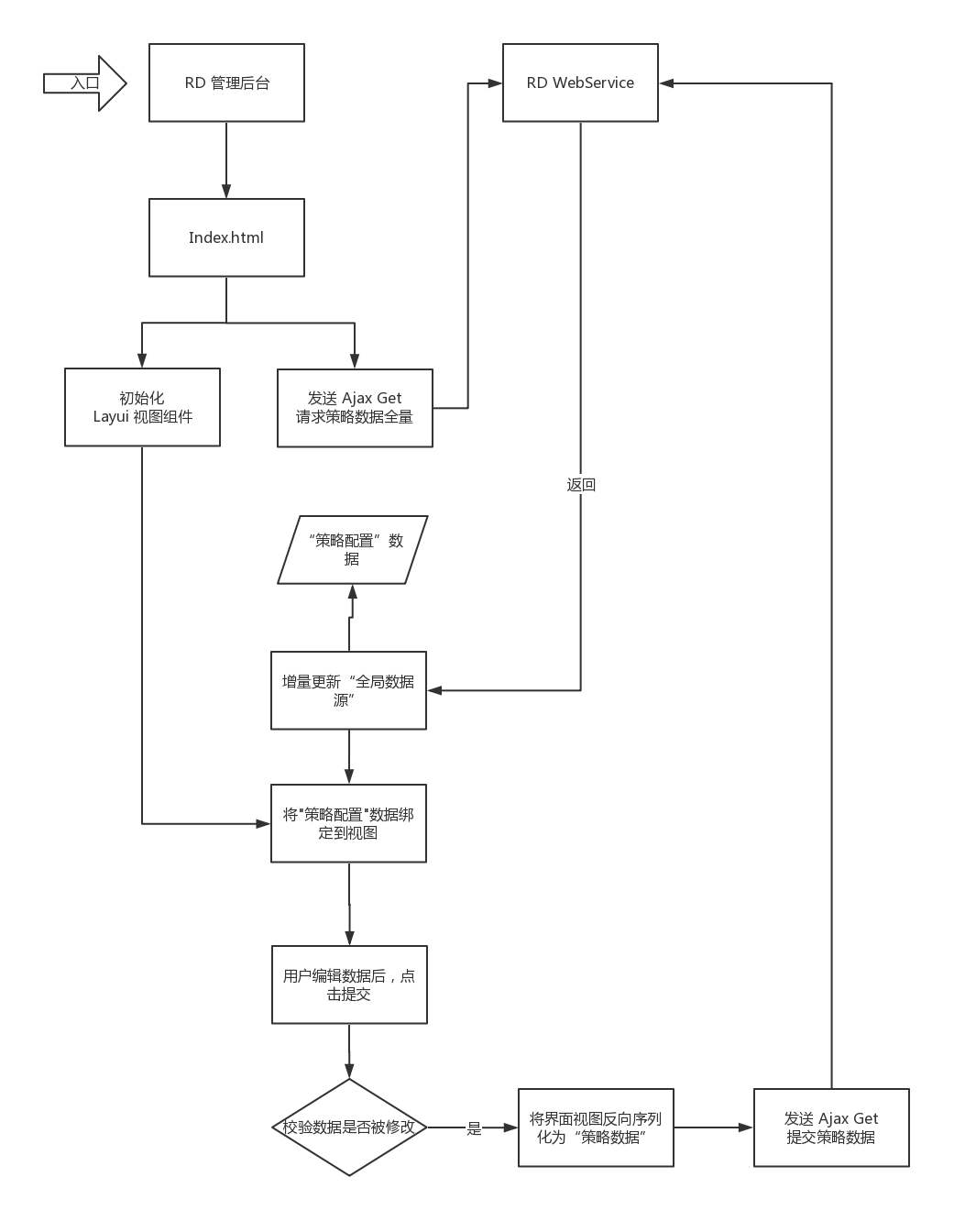
**WebService接口服务主要由4大块组成，分别为：**

* 数据持久化模型（目前为直连数据库）
* 对外接口服务。采用golang iris框架，提供动态数据服务与静态资源服务，对外公开端口为8080
* 数据同步服务，使用golang源生net相关支持，与RD Socket服务建立TCP通讯。目的是通过RD Socket服务中转，推送实时“策略增量更新数据”到接入端。

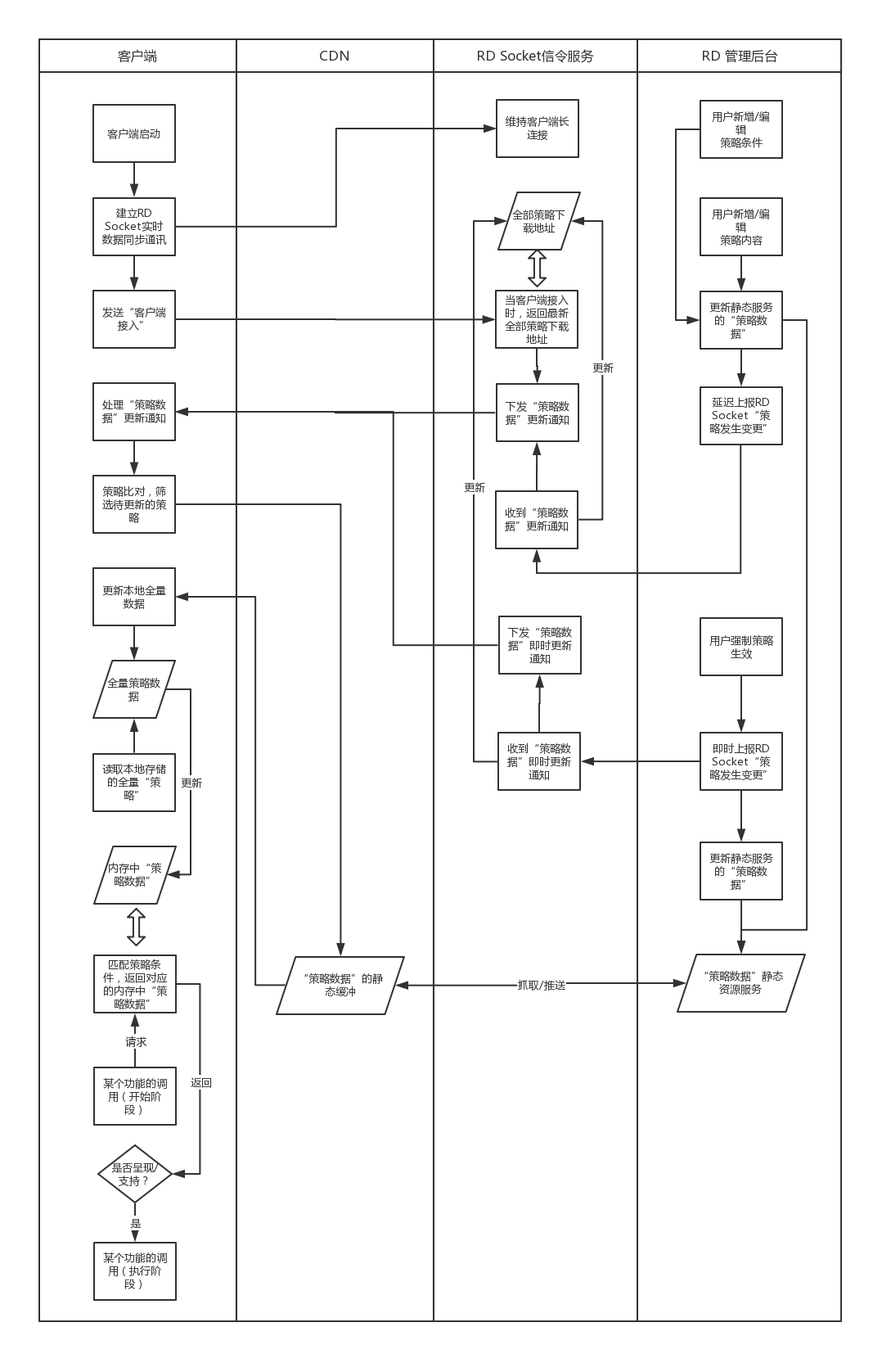
## 接口服务工作流程



## RD管理后台工作流程



## 客户端与RD Socket通讯流程



## 策略匹配与生效原则

**策略匹配工作由客户端RD SDK来完成，其匹配原则如下（其中涉及到的客户端请求参数结构体，以下简称req）：**

* 从本地所有的策略中筛选出”策略类别名称 == req.请求策略名称”的所有策略，作为待匹配列表
* 从待匹配列表中，剔除已过期的策略
* 使用用条件计数法遍历上面的结果列表，当”策略自带的绑定条件的值 == req中的对应的条件的值”时，该策略计数加1，如果”策略自带的绑定条件的值 != req中的对应的条件的值”，则该策略计数直接标记为-1,并且跳过该策略，直接尝试匹配下一条策略。
* 当所有策略都计算出得分后，选得分最高的一条策略作为匹配策略回调给客户端。如果存在相同的分的N个策略，则选取“策略绑定的优先级级数”最高的策略作为最终匹配策略，如果N个策略均无绑定优先级，则默认选择第一条策略作为最终匹配策略。