

---

# 实时数据库用户指南

(RDB user guide)

---

适用：

RDB2025.3(内部版本 5122)及以后版本

Process View 5.0.8.1 及以后版本

重庆唐码软件有限公司

2025 年 6 月修订

---

# 目录

1 概述 .....	1
1.1 系统组成 .....	1
1.1.1 系统结构图 .....	2
1.1.2 实时数据库 .....	3
1.1.3 生产数据网关 .....	3
1.1.4 图形客户端软件 .....	4
1.1.4 管理维护工具 .....	4
1.1.5 后台控制任务 .....	4
1.1.6 客户端数据访问接口 .....	4
1.2 系统平台 .....	4
1.3 硬件系统 .....	5
1.3.1PC 服务器硬件要求 .....	5
1.3.2 嵌入式 ARM 平台 .....	5
2.实时数据库 .....	7
2.1 设计规格 .....	7
2.2 数据类型 .....	8
2.3 标签属性 .....	8
2.3.1 基本属性 .....	9
2.3.2 压缩和例外属性 .....	10
2.3.3 越限事件属性 .....	11
2.4 压缩原理 .....	11
2.5 例外偏差过滤 .....	13
2.6 实时库安装 .....	13
2.6.1 规划和配置 .....	13
2.6.2 Window 安装 .....	16
2.6.3Linux 系统下安装 .....	17
2.6.4 初始默认账号 .....	18
2.7 管理维护 .....	18
2.7.1Windows 系统下安装 .....	18
2.7.2Linux 系统下安装 .....	18
2.7.3 维护 .....	19
2.7.4 标签导入 CSV 文件格式 .....	19
2.7.5 使用 WEB 页面管理维护 .....	20
3.生产数据网关 .....	22
3.1 安装 .....	23
3.2OPCDA 驱动 .....	23
3.3.1OPCDA 参数配置 .....	23
3.3.2OPCDA 标签表 .....	24
3.4MODBUS 驱动 .....	24
3.4.1MODBUS 参数配置 For Windows .....	24
3.4.2MODBUS 参数配置 For Linux .....	26
3.4.3MODBUS 标签表 .....	27
3.5OPCUA 驱动 .....	29
3.5.1OPCUA 参数配置 .....	30
3.5.2OPCUA 标签表 .....	31
3.6 预处理配置 .....	33
3.7 计算标签配置 .....	34
3.8ARM 平台嵌入式 Linux 系统数据采集终端 .....	35
4.图形客户端软件(Process View) .....	36
4.1 图形组态 .....	37

---

4.2 扩展自己的 html 代码 .....	38
5.实时库客户端接口 .....	41
5.1 例子工程 for C#.....	41
5.2 例子工程 for Java .....	41
5.3Javascript 接口 .....	42
5.4 自己根据协议实现 .....	42
5.5OPC DA 实时数据服务.....	42
5.6OPC UA 实时数据服务.....	42
6.工程配置技巧 .....	43
6.1 标签的压缩方式选择 .....	43
7.嵌入版实时数据库 .....	44
7.1 嵌入版硬件平台 .....	44
7.2 规格和售价 .....	45
附录 1: 历史数据规模估算 .....	46
附录 2: 趋势压缩精度的配置原则 .....	47
附录 3: 在线体验云实时库 .....	48
附录 4: 第三方应用提交数据建议 .....	49
附录 5: 编写第三方协议驱动 .....	50
附录 6: 实时库高可用冗余配置 .....	51
6.1 高可用原理 .....	51
6.2 配置双机热备冗余 .....	52
6.3 使用虚拟路由器 .....	52
6.4 数据网关主备双库连接 .....	52
附录 7: 实时库存储空间配置策略 .....	54

---

# 1 概述

实时数据库，也叫过程数据库或者时序数据库，记录测点一个时间过程数据，因此数据记录具有时间性(时标)，有效性(数据质量)，数据性(简单数值或者 json 对象)，记录按照时间递增在历史中存储。

TOM 实时数据库系统（以下简称 TOM 实时库）是重庆唐码软件有限公司开发的基于 C/S 和 B/S 结构的工厂实时数据集成、应用的高可靠高可用企业数据平台。TOM 实时库系统以数据原形的方式长期在线储存工厂所有的生产数据，并满足快速、高效地进行数据采集、存储和显示的要求。实时库数据访问协议基于 websocket 传输层，使用 JSON 编码，并提供使用 C/C++实现此协议的 windows/Linux(包括国产 aarch64 架构 CPU 平台)多双平台客户端 API 接口库用于企业数据的挖掘分析。

- 1) 分布式部署，一个冗余中心站和多个冗余子站，子站数不限。
- 2) 大容量，单节点百万级标签容量。
- 3) 高可用，实时库支持双机主从热备冗余。现场数据网关支持双机热备部署，支持整体切换和设备级切换，双机主备冗余只需要一个授权。
- 4) 高并发，单节点不限制并发连接数(仅受系统资源限值，一般主流服务器大于 2 万并发连接)，很好地支持移动客户端。
- 5) 海量存储，单节点最高 800TB，节点数不限，可配置循环重用历史存储空间。
- 6) 可恢复 redo 日志，保证掉电不丢缓存数据。
- 7) 优化 B+树索引，高效的历史数据检索查询。
- 8) 可配置有损压缩和无损压缩的历史归档。
- 9) 支持 Pubsub 订阅推送，可模糊匹配精准匹配。网络报文实时高速压缩，提高传输效率。
- 10) 跨平台，同时支持 windows 64 位和 Linux64 位系统平台和 ARM-aarch64 平台。
- 11) 支持 http/https 和 websocetk 协议，方便 web 客户端 javaScript 访问实时库，提供 ipv6 访问支持。
- 12) 基于 html5 跨平台的 web 实时工况图发布。
- 13) 可配置加密通信，安全 websocket 基于 TLS1.2 规范。
- 14) 支持镜像数据单向转发，适合电站 SIS 系统数据从生产网单向穿越网闸到管理网。现场数据网关同样支持过网闸的 gapudp 和 gaptcp 协议。
- 15) 端口复用，智能协议识别，HTTP/HTTPS 协议，WS/WSS 协议，所有协议和接口均走一个 TCP 接口，便于防火墙管理。
- 16) 支持公有云和私有云部署。
- 17) 提供基于 ARM 的嵌入式实时库和数据网关，支持无人职守，野外部署。
- 18) 提供 OPCDA 和 OPCUA 服务,供第三方应用订阅快照数据。

TOM 实时库已用于电厂、钢厂、化工厂、汽车充电站系统、风电远程监测系统等需要海量数据存储和分析的行业。

## 1.1 系统组成

作为一个企业生产数据平台，整个系统以实时数据库为核心，分为实时数据库，数据网关、控制任务，图形组态，管理工具，对外标准服务(OPCUA)几大部分。

### 1.1.1 系统结构图

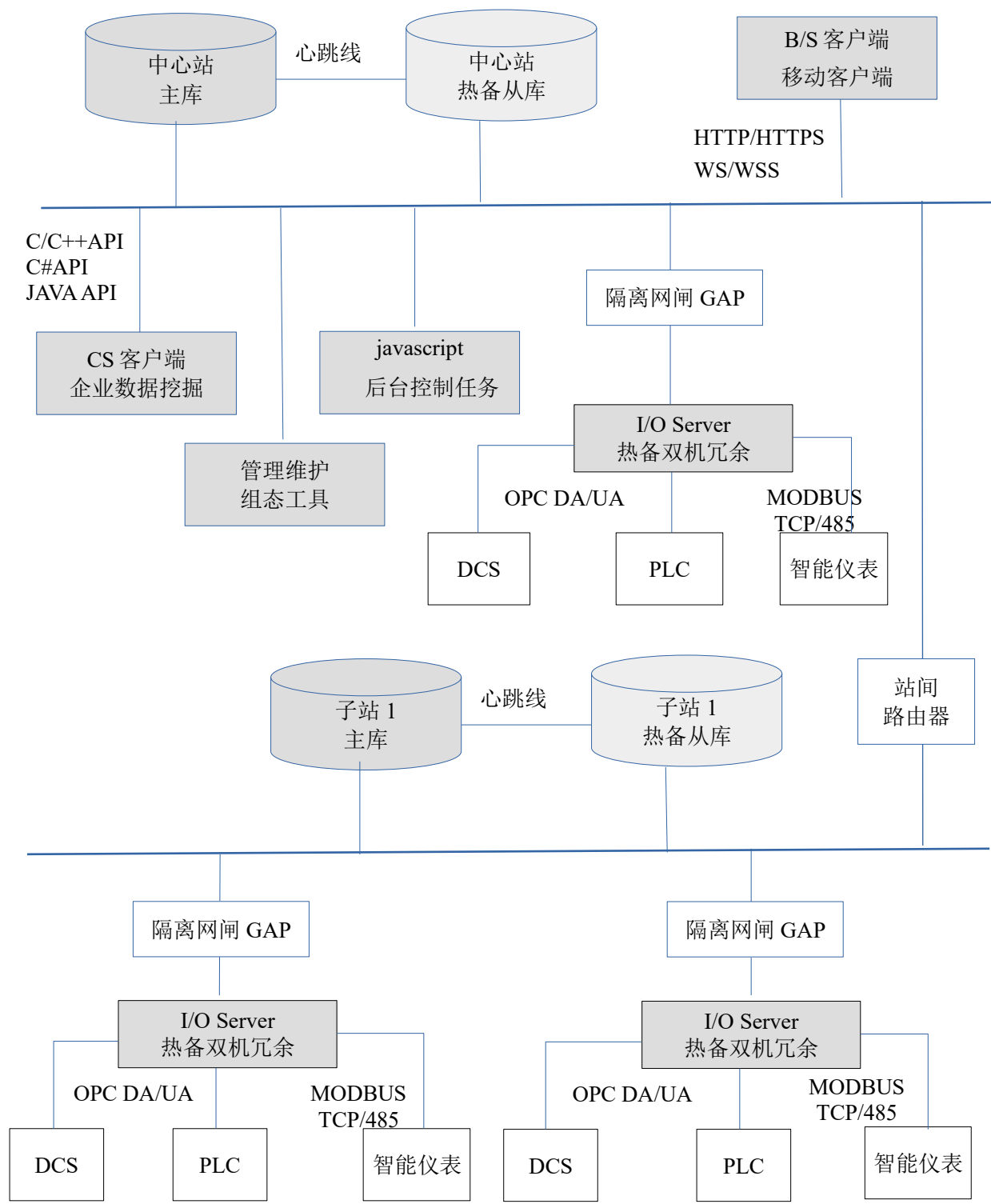


图 1-1 实时数据库典型应用系统结构图

- 1) 主库和从库运行 rdbsrv(widows)或者 rdbux64(Linux64)，用一个 TCP 端口实现 API 协议，HTTP/HTTPS 协议，WS/WSS 协议。轻松部署到云端，包括阿里云，腾讯云。主备库自动从对端同步离线数据。

- 2) I/O 运行数据网关软件 IOserver，支持 windows/Linux PC 平台，ARM linux 平台，支持热备双机冗余，支持主备整体切换和设备级切换。
- 3) 后台控制任务运行 rdbctrl 软件，支持 windows/Linux 平台，使用 javascript 脚本编写后台控制逻辑。
- 4) 管理维护使用图形界面的 rdbman 工具，支持 windows/Linux 平台。
- 5) 图形组态使用 processview 软件，包括组态(自动 web 发布用生成 html 页面)。C/B 运行版，WEB 版使用相同的 html5 方案，支持国产信创平台。
- 6) 客户端可以通过中心站访问子站，只需在标签名签名加上子站名和冒号即可。或者参见实时库数据交互协议 rdb\_data\_exchange\_protocol.pdf。

### 1.1.2 实时数据库

后台服务软件，windows 系统下为 rdbsrv.exe，Linux 环境为 rdbux64，均为 64 位软件。支持故障自动恢复，索引自动修复，索引重建，再恶劣的环境也不惧。

提供 C/C++,JAVA, .NetAPI 接口，支持 HTTP/HTTPS，WS/WSS 直接访问。

另外还提供 ARM 硬件环境的嵌入式 Linux 操作系统版的实时库，软硬件一体，限 2000 标签点，适合小型无人值守环境。

内置 HTTP/HTTPS 发布，使用 html5+svg+websocket+javascript 技术的 WEB 图形发布无需使用 IIS，Apache 等第三方 WEB 发布软件。

### 1.1.3 生产数据网关

为了方便驱动开发和生产使用，RDB5 提供了两个数据网关，分别用于不同用途。数据网关提供上行（数据采集）和下行（设备控制输出）基本功能外，还提供数据采集值工程值转换，数据预处理。

- 1) 前台界面版驱动调试用网关 rdbdac\_wx，是 windows 7 及其以后系统图形界面版。主要用于调试系统集成商自己开发的设备驱动，不具备双机热备冗余功能和多实时库数据提交功能。不建议生产环境使用。
- 2) 多进程版 IOserver 数据网关，驱动以进程方式存在并通过内部消息总线交换数据，后台服务模式，故障隔离，某个驱动的崩溃不会影响其他驱动。IOserver 支持双机热备冗余和驱动设备级故障迁移，支持多库提交数据，每个库均支持主备库故障切换。是推荐使用的生产数据网关。支持 WEB 管理维护。日志输出到专用日志服务器 logsrv 中，使用浏览器接入查看日志。

目前支持的数据协议有 Modbus RTU for TCP/RS485 , OPCDA2.0(windows 版)，OPCUA(新标准的 OPC 统一架构，Windows，Linux(含国产系统和 CPU))。

网关支持数据的采集(从设备上传)和控制输出(下传到设备)。因此客户端可以直接向实时库用发送控制命令，实时库经网关下传到设备。

另外还有嵌入式 ARM 平台的网关，功能同 Linux 版，适合无人值守远端现场，可通过外接 4G 路由器上传数据到云端和接收云端的控制命令。

---

### 1.1.4 图形客户端软件

图形部分为 Processview，包括工况图形组态软件，C/S 客户端和 B/S 客户端均采用 Html5 方案实现，支持 Window，Linux(含国产信创平台)。

### 1.1.4 管理维护工具

Rdbman 用于管理维护实时数据库，图形界面程序，提供 windows 和 Linux gkt2.0 版。用于标签维护，数据查询。

另外提供 WEB 版全功能管理维护界面，客户端只需要一个支持 HTML5 的浏览器即可完成实时库的管理维护。

### 1.1.5 后台控制任务

rdbctrl 以后台服务方式运行，管理和执行定时任务和周期性任务。任务采用类 javascript 脚本编写，脚本提供快照读，快照写，写设备等接口。主要有以下几种用途：

- 1) 利用控制任务可实现比如定时开启或停止设备，比如晚上开启路灯，早上关闭。
- 2) 可实现设备之间的联动，比如根据隧道瓦斯浓度调节排风扇速度。
- 3) 实现应用级的复杂计算标签，将多个实际标签定时计算后通过写快照方式写入到实时库的一个计算标签。
- 4) 实现设备之间的数据传输，将一个设备的数据写入到另一个设备，或者将几个设备的数据计算后写入到另一个设备。

### 1.1.6 客户端数据访问接口

直接提供实时库数据交互协议(rdb\_data\_exchange\_protocol.pdf)，基于 WS/WSS 通道使用 JSON 编码的应用层数据通信协议，方便 C/C++，DotNet，Java，C#，javascript 直接访问实时库。

并提供了一个 windows 和 Linux 平台 C/C++版的实现该协议的动态库 rdbapi。提供 C/C++接口 API 使用例子。Java JNA 模式可直接使用 C 接口 rdbapi。

提供了一个 C#使用 rdbapi 的例子。

提供了一个 java 使用的 jni 接口库(jniwsapix64.dll/libjniwsapix64.so)，封装了协议通信层，客户端可直接使用这个接口库向实时库发送消息，具体参见 rdbapi/javaws 中使用例子。

其他平台建议直接使用 rdb\_data\_exchange\_protocol.pdf 协议访问实时库。

对于快照数据，提供了 OPC DA 和 OPC UA Server，第三方软件可以使用 OPC 协议订阅实时库标签点快照数据。

## 1.2 系统平台

windows 系统：服务器为 windows server 2012 及以后的 X86-64 服务器版。客户端软件要求 windows7 及以后的 x86/x64 平台。

Linux(X86\_64)系统：服务器为核心 3.10 版级以后的 X86-64 服务器版。服务端建议使用 centos 7 和 ubuntu server 1604。Linux 图形客户端建议使用 ubuntu1604desktop 版，安装 gtk2.0。

Linux(aarch64)国产系统：aarch64 架构 CPU，银河麒麟 V10 系统；内核 4.4，glibc 2.23，libstdc++.so.6.0.21(编译器 G++ 5.40)，相当于 ubuntu16.04 aarch64 系统平台环境。

### 1.3 硬件系统

客户端软件对硬件系统无特殊要求。服务器要求 CPU 核心和内存资源根据标签数规模不同而不同，下表为各种标签规模下最小要求和推荐要求。

#### 1.3.1PC 服务器硬件要求

表 1-1 实时库服务器硬件配置表

标签规模	最小配置		推荐配置	
	CPU	内存	CPU	内存
< 2 万	2 核	4G	4 核	8G
5 万	4 核	8G	8 核	16G
10 万	6 核	8G	8 核	24G
20 万	8 核	12G	8 核	32G
50 万	8 核	16G	12 核	48G
100 万	8 核	24G	16 核	64G

注：大于等于 5 万标签请使用品牌服务器并配置带回写缓存的 RAID 存储系统。存储空间根据标签规模计算，参考附录 7 实时库存储空间配置策略。

#### 1.3.2 嵌入式 ARM 平台

IOServer 数据采集网关软件全功能支持 ARM 平台嵌入式 Linux 系统。具体硬件平台性能指标请参阅具体的硬件规格文档。

表 1-2 嵌入式 ARM 硬件平台主要参数表

序号	指标	GT6757
1	CPU	ARMV8 aarch64 A35 双核 800MHz
2	内存	128MB
3	存储	8G EMMC
4	网络	10/100M 2 个
5	485 口	4 个 RS232/485 复用
6	看门狗	硬件看门狗
7	加密芯片	有



8	可安装软件	IOServer , 限 2000 点 MODBUS 转 OPCUA 网关, 限 2000 点 Logsrv 本地日志服务
---	-------	---



图 1-2 GT6757 嵌入式 ARM 硬件平台实物照片

## 2.实时数据库

TOM 实时库是高可靠性和高性能实时库系统，先进的压缩和存储格式，极大节约磁盘空间，延长数据的在线年限，高效的优化 B+树索引机制保证了历史数据的检索效率。

这里的实时数据库指后台服务程序，windows 平台为后台 service 服务程序，Linux 平台为 systemd 管理的后台守护(Daemon 进程)。

### 2.1 设计规格

表 2 -1TOM 实时库规格表

项目	指标	说明
运行平台	Windows server 2012 及以上服务器操作系统 X64 版本 Linux 内核 3.10 及以上 X64 版本	支持 x86_64 平台，Aarch64 平台(含国产 CPU 和国产 Linux 系统)
高可用	双机热备冗余，自动同步	实时库和数据网关
分布式大型应用	一个中心站 + N 个子站	N 不限制
最大标签数	N * 100 万	N 为节点数(一个中心站或者子站为一个节点)
最大历史容量	单节点 800TB	可配置循环重用空间
最大并发连接客户数	不限制(单节点大于 50000 客户并发连接)	最终并发数和系统配置以及系统资源有关
实时数据提交	大于 100 万记录/秒	
实时数据读取	大于 100 万记录/秒	
实时数据订阅推送	通配和精准订阅，最大订阅标签数不限	
历史插入和补录	大于 50 万记录/秒	
历史数据读取	大于 100 万记录/秒	
历史数据删除	大于 100 万记录/秒	
压缩方式	例外偏差 + 趋势有损压缩 + 无损压缩	例外偏差和线性分段压缩可单点可配置
可恢复 redo 日志	支持，可配置 redo 日志目录	
安全通信	可配置使用基于 TLS1.2 规范的加密通道。	
索引方式	优化 B+树页面索引	
字符串和对象类型	支持，每条记录最大 1000 字节	
SOE 事件	可配置后台生成	支持订阅和表达式查询。
自定义后台任务	支持	使用 javascript 脚本编写后台任务
第三方应用支持	提供客户端 API 供第三方案程序使用提供 websocket 协议供 web 客户端访问	支持多线程连接。

是否内置 HTTP/HTTPS 发布	是，支持超大文件下载	支持 WEB 图形发布，无需第三方 HTTP/HTTPS 服务器
是否内置 WS/WSS 服务	是，支持压缩扩展	无需第三方组件
管理工具	提供管理工具管理实时库	WEB 版和前台界面版

## 2.2 数据类型

表 2-2 标签数据类型表

类型	长度	说明
Digital	1 字节整数	存储开关量
Int32	4 字节整数	带符号
Float32	4 字节浮点数	IEEE754 格式，32 位
Int64	8 字节整数	带符号
Float64	8 字节浮点数	IEEE754 格式，64 位
String	<1000 字节的字符串	0 结束的字符串,用于存储格式串参数等信息。
Object	<1000 字节,对象类型,二进制数据	对象类型，二进制字节流，用于存储手工标签或预置标签，比如机组每日的计划负荷等信息。

在定义标签时，在满足标签值精度范围的条件下，尽量使用短字节数的类型，这样可节约磁盘空间。

String 和 Object 不能做有损压缩，只有无损压缩，因此适合用于不频繁存储参数配置、计算结果。不适合用于保存频繁插入和追加记录的现场实时数据(如果磁盘空间足够大也可以)。

## 2.3 标签属性

标签属性包括了基本属性和压缩属性，压缩属性的配置直接影响到最终历史数据的规模。

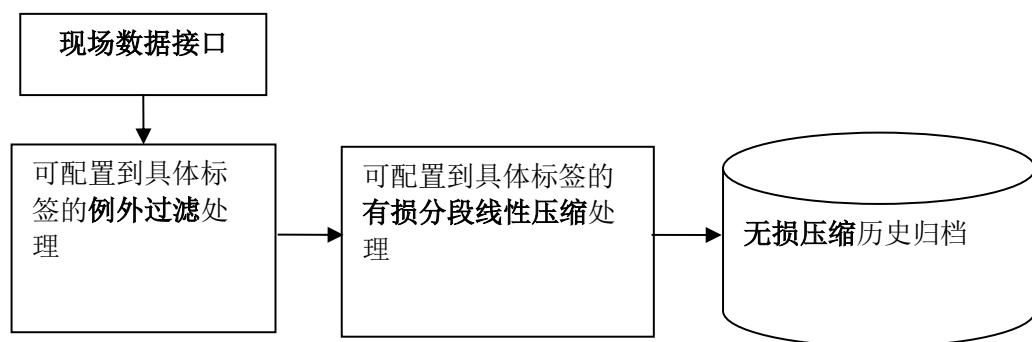


图 2-1 数据处理流程图

结合上面的数据流程图理解标签的例外和压缩属性。压缩原理请参见 2.4。

## 2.3.1 基本属性

表 2-3 标签基本属性表

属性	类型	默认值	说明
TAGID (标签 ID)	无符号 4 字节 整数	自动	实时库内部使用的不重复的标签 ID，作为索引使用，客户端不能更改，但可以通过接口获得这个 ID。
Name (标签名)	字符串		下横线_或大小写字母或汉字开头,后跟数字或大小写字母或汉字的 0 结束的字符串，汉字为 GB2312 编码。作为客户端访问实时库的标签索引，全局不重复。不能含有西文括号和西文单双引号这 4 个字符，另外西文冒号用于分割子站也不能使用。
Description (标签描述)	字符串		用于标签的简单说明性文字，小于 80 字符。汉字为 GB2312 编码。
EngUnit (工程单位)	字符串		工程单位，比如 kw。小于 16 字符。汉字为 GB2312 编码
PointType 数据类型	1 字节整数		标签的数据类型 Digital，Float32 等类型
Class (标签分类)	1 字节整数	0	标签分类: DEC 现场设备标签 DEFINE 定义标签，可写未来数据(rdb2021.4 版开始启用)。 PRESET 预置标签(从 rdb1911 开始，PRESET 标签只有可落地持久化的快照，无历史，可以当作全局变量使用) CURVE 曲线标签，暂未定义用途
Step	1 字节整数	OFF	读插值和绘图数据时是否使用梯形插值算法， rdb2021.10 开始启用 OFF(0) 不使用梯形插值，默认使用线性插值 ON(1) 使用梯形插值，Digital 强制为梯形插值
Archiving (归档存盘标志)	1 字节整数	ON	是否归档存盘 ON(1) 表示存盘 OFF(0)表示不存盘
DisplayDigits (显示精度)	1 字节整数	3	显示用精度： 范围 [-20 , 10] >0 表示精度(小数位数) <0 表示有效位数 实时库并不使用该参数，仅供客户方使用。

## 2.3.2 压缩和例外属性

使用的和 PI 旋转门相似的分段线性压缩 方式，简称压缩和例外。

表 2-4 标签压缩和例外属性

属性	类型	默认值	说明
CompType (压缩方式)	1 字节整数	percent	No (0)不压缩 Percent (1)百分比精度 Abs (2) 绝对值精度 Timer (3) 按照最大压缩周期间隔存储 Extimer (4)按照最大压缩周期间隔存储,并存储区间的最大值和最小值记录。
CompDev (压缩精度)	4 字节浮点数	0.1	这个值的含义和 COMPTYPE 相关 当 COMPTYPE = percent 时表示 0.1%（即千分之一） 当 COMPTYPE = abs 时表示 0.1 绝对值。具体解释见第 2.4。
CompMin (最小压缩周期)	2 字节整数	0(秒)	单位是秒 当两条记录之间的时间差小于 COMPMIN 时，新来的记录被抛弃。该配置具有降噪作用。具体解释见第 2.4。
CompMax (最大压缩周期)	2 字节整数	3600(秒)	单位是秒:范围[30,3600] 当长时间没有后续数据到来时(超过 COMPMAX)或两条记录之间时差大于 COMPMAX 强制提交到历史服务。具体解释见第 2.4。
ExcDev (例外偏差)	4 字节浮点数	0	例外偏差值。一般填写 COMPVAL 的 1/2 即可。具体解释见第 2.5。
ExcMin (最小例外偏差周期)	2 字节整数	0(秒)	单位是秒。 当压缩后的两个记录值时标之差小于该值时，滤掉。该配置具有降噪作用。具体解释见第 2.5 章。
ExcMax (最大例外偏差周期)	2 字节整数	600(秒)	单位是秒，时差大于等于该值时或者当被例外滤掉的记录值长时间没有后续数据到来时(超过 EXCMAX 秒)，强制提交这个记录到压缩处理。具体解释见第 2.5。 如果要关闭例外处理，将 EXMIN 和 EXCMAX 同时置 0 即可。

注：压缩属性配置只对 int32,float32,int64,float64 起作用，digital 标签内部采用默认配置(最大压缩周期为 8 小时。)

### 2.3.3 越限事件属性

实时库支持后台越限事件生成，可为每个标签配置越限事件定义，当发生越限和越限消除时会根据配置产生相应的 SOE 事件并写入库中，如果需要对越限事件做自定义描述，可以使用 rdbman 工具编辑和导入标签扩展属性。

表 2-5 越限属性表

属性	类型	默认值	说明
Alarm (报警方式)	4 字节整数	0	报警设置: 按位设置; bit0: 低限或者开关状态变低; bit1: 高限或者开关状态变高; bit2: 低低限; bit3: 高高限 例如配置 15 表示全报警, 3 表示高低限报警。0 表示不报警
Alarm_LL (低低限值)	4 字节浮点数	0	配置 Alarm bit2=1 且快照小于此值是报警
Alarm_L (低限值)	4 字节浮点数	0	配置 Alarm bit0=1 且快照小于此值是报警
Alarm_H (高限值)	4 字节浮点数	0	配置 Alarm bit1=1 且快照大于此值是报警
Alarm_HH (高高限值)	4 字节浮点数	0	配置 Alarm bit3=1 且快照大于此值是报警

### 2.4 压缩原理

TOM 实时库采用趋势压缩，理解压缩原理有助于正确配置压缩参数，趋势压缩的原理图如下，截取某一段连续时间的压缩过程。

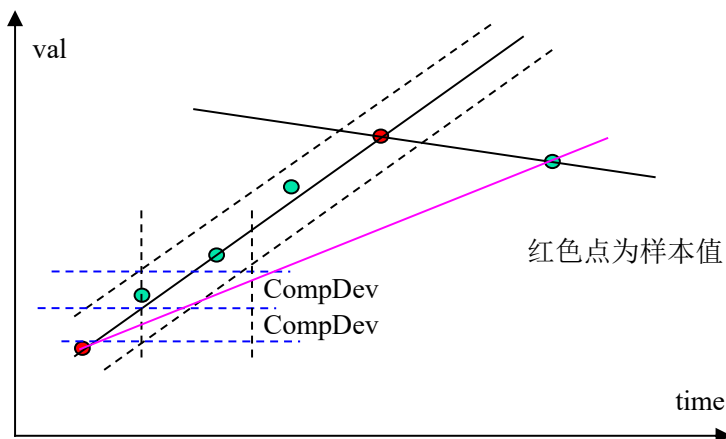


图 2-2 趋势压缩原理图

---

工程值作为时间的函数  $f(t) = val$ 。当计算的  $f(t)$  值和该点的实际值偏差小于指定的压缩偏差 **CompDev** 时，该点可以不存盘。**CompDev** 在绝对偏差压缩方式时表示的是绝对值偏差，在百分比偏差压缩模式时是偏差百分数。

3 各点及以上才能开始趋势压缩。

**CompDev**（压缩精度）

压缩精度，用于限制精度，默认 0.1（千分之一）。当实时记录数据落在两条黑色虚线的范围里时，这些值在满足精度要求的情况下可以被计算插值还原，因此可以不记录这些值(图中的 P2,P3,P4 点)。当 P6 点到来时，从 P1 点到 P6 点的连线会使 P2-P5 之间有一点会落在精度范围外，这是需要将 P5 点作为样本值提交给历史服务，然后将 P5 点作为 P1 点重新开始新的压缩。

**CompType**（压缩方式）

压缩模式，默认 **percent**（即百分比偏差）。

如果配置的是百分比压缩精度，则偏差按百分比计算。

如果配置的是绝对精度，则偏差按绝对值计算。

该配置决定压缩偏差计算和后面的例外偏差计算的方式。

**CompMin**（最小压缩周期）

单位是秒，默认值是 0 秒。

假如上图中的  $P2.ltime - P1.ltime < COMPMIN$ ，则 P2 点直接抛弃，不进入压缩缓冲区。两个连续的待压缩点之间时间差小于 **CompMin** 都会抛弃，比如  $P4.ltime - P3.ltime < CompMin$ ，则直接抛弃 P4 点。这个配置具有降噪作用。另外的意义在于，比如对于某些标签点，实际需求不需要密集的存储(比如累计电量，要求不小于 5 分钟存储一点)，这样可以把最小压缩偏差设置为 300 以降低磁盘占用。还有比如环境温度等缓慢变化的量也可以把最小压缩周期设置为 1 分钟或更大。

**CompMac**（最大压缩周期）

单位是秒，默认值是 3600 秒。

1) 最大压缩周期是解决长时间不来新数据的情况下，数据记录提交到历史服务的问题，用上图说明：

在 P5 作为样本值提交到历史服务后，用 P5 做压缩起点，P6 是最新的数据(snapshot)，待处理(在没有新数据到来时，不能知道 P6 是否可以滤掉)，但是自 P6 到达后，长时间（服务器时间）没有新数据到来（假设时间超过了 **CompMax**），这时需要强制将 P6 作为样本值提交到历史服务，用 P6 作为起点重新压缩。

因为实时库自己不产生数据，如果不限最大压缩周期，假如后面一直没有新数据来，则 P6 这个数据永远不会被提交到历史服务。

2) 数据的斜率一直不变(包含数据一直不变，但时标在变的情况)，在最大压缩周期到来时强制转换为样本值提交历史存盘。

---

## 2.5 例外偏差过滤

例外偏差过滤简称例外，是对现场数据接口提交来的数据做初步过滤，过滤后提交到压缩服务。对于具备例外处理的功能现场数据接口可以关闭实时库提供的例外处理。

ExcDev（例外偏差）

用于限制两个记录值之间的差值。默认值为 0，一般配置压缩精度 CompDev 的 1/2 即可。当新来的数据值时标差大于 ExcMin 且偏差大于等于 ExcDev 时不能滤掉。

ExcMin（最小例外周期）

单位是秒，默认值 0 秒，当两个连续记录时间差小于 ExcMin 时滤掉。

ExcMax（最大例外周期）

单位是秒，默认值 600 秒。当连续两个记录值的时间差大于等于 ExcMax 时或者 ExcMax 时间没有来新数据时，不能滤掉，需要强制提交记录给压缩处理。

ExcDev、ExcMin、ExcMax 之间关系

ExcDev、ExcMin、ExcMax 是配合使用的，当两条记录时间差在 ExcMin 和 ExcMax 之间时才计算偏差，根据偏差结果决定是否过滤。否则按上述 ExcMin 和 ExcMax 的过滤方式处理。

取消例外偏差配置

如果不需要例外偏差过滤，只需将 ExcMin 和 ExcMax 同时配置为 0 即可。

## 2.6 实时库安装

### 2.6.1 规划和配置

安装前规划实时库，估计标签规模，选择足够大的存储空间。是否需要双机热备冗余实现高可用等。在 windows 下为 rdbsrv.ini 文件和 httpsrv.ini，linux 下为 rdbux64.conf 文件和 httpsrv.conf 文件，配置方法完全一样。

从 rdb2023.4（内部版本 5105）开始支持 redo 功能，断电后可在下次启动时先从 redo 目录里恢复缓存数据。Redo 采用追加顺序写盘，速度很快，如果应用规模比较大，选一个和数据存储不在一起物理盘来存储 redo 日志。按照每 1 万标签点 500MB(约 2 倍安全系数) 准备 redo 空间。

用纯文本编辑器打开配置文件，编辑配置项。如下实例，分号或井号后面的为注释。

windows 版下的 rdbsrv.ini

```
;real database config file
```

```
[log]
```

```
logurl = udp://127.0.0.1:999/rdbsrv?level=dbg
```



; 日志输出级别, 可选 err, wrn, msg, mor, dbg

[database]

path = d:/rdb5 ;数据库存放目录

redo\_path = c:/rdb5\_redo ;redo 日志存放目录

name = demo ;数据库名称, 自己根据需要取名。

maindb = yes ;[yes,no]主库还是从库, yes 表示主库, no 表示从库。如果没有配置心跳线且配置为 no 时则表示是一个单纯的备份库。

heartline = ;192.168.1.39:911/heartid ;主备热备冗余心跳配置, 高可用才配置

;stationid= rdb\_center\_station; 空表示无子站库, rdb\_center\_station 表示中心站, 其他字符串为子站。

;center\_master = 192.168.1.81:921 ;中心站主机, 自己是子站才需要这个参数

;center\_slave = 192.168.1.39:921 ;中心站从机, 自己是子站才需要这个参数

writethrough = off ;[on,off]存储时写通, on 表示开启写通(不使用系统文件缓存);off 表示使用系统缓存,

;设置为 off 能提高大规模标签时存储系统的写入性能。on 仅用于环境条件很差经常断电的且没有 UPS 的场景。从 2023.4(ver5105)开始到 2025.1(ver5121)之间, 该配置作废, 始终为 writeback 模式。从 2025.3(ver5121)开始, 这个配置仅用于数据页面, 索引页面强制 writethrough 模式。

ws\_ctrlout = on ;WS 服务开启控制输出 on/off

wss\_ctrlout = on ;WSS 服务开启控制输出 on/off

license = ;在线授权方式的授权文件全路径文件名, 比如 c:/ca/dbcert\_gc1.dbcer 不填写表示使用 USB KEY 授权

makesoe = yes ;[yes,no]产生标签超限 SOE, 不配置默认表示关闭此功能。

dbreusesize = 500 ; 单位 GB, 重用空间大小。当实时库空间增长到该配置时, 开始循环重用空间, 默认值 0 表示不重用。

tagreusesize = 10 ; 单位 MB, 标签所占空间重用大小, 默认值 10, 当 dbreusesize 达到条件且该标签所占空间大于本设定值时开始重用磁盘空间。

hotdata = yes ;历史数据保温, yes/no, 用于私有云或者公有云服务器, 能大幅提高冷数据的读取效率, 默认为 no, 该功能从 rdb2023.5(内部版本 5106)开始提供。

[netsrv]

port = 921 ;服务端口

ipv6 = :: ;bind 的 ipv6 地址, ::表示所有

MaxConnectionsPerIP = 64 ;每 IP 最大连接数, 默认 64;0 表示不限制; 如果配置, 不小于 16

---

`MaxSubscribeSessions = 200` ;最大订阅连结数。默认值 200, 可配置范围 50-5000

`ca_public =` ; TLS1.2 服务器证书全路径文件, 含 RSA 公钥, 空表示没有证书, 无法开启 HTTPS 和 WSS 服务。

`ca_private =` ; TLS1.2 服务器私钥全路径文件, 含 RSA 私钥, 空表示没有证书, 无法开启 HTTPS 和 WSS 服务。

`httproot = c:/rdb_http` ; http/https 的 html 文档根目录

`[mirror_out]` ; 转发到镜像服务器

`ip =` ; 镜像服务器 IP

`port = 921` ; 镜像服务器端口

`protocol = off` ; tcp/udp/off 镜像服务器协议, 不配或者 off 表示不开启, 开启后可将本库收到的数据镜像一份输出到配置的镜像服务器。

`user =` ; 写镜像服务器账号

`pswd =` ; 写镜像服务器密码

`[mirror_in]` ; 本机做为镜像服务器

`protocol = off` ; tcp/udp/off, 本机接收镜的像输入协议, udp/off, 不配或者 off 表示不开启。使用[netsrv]中配置的 port 作为端口。配置为 tcp 则数据网关可使用 gaptcp 穿越单向网闸向此库提交数据。配置为 udp 则数据网关可使用 gapudp 协议穿越单向网闸向此库提交数据

Linux 系统下的 rdbux64.conf, 注释使用#, 和 windows 下的配置完全一样, 只是文件目录格式不同。

[database]小节的 path 目录无需创建, 实时库系统启动时会自动创建。

writethrough 为磁盘写通配置, on 表示写通直接到磁盘, off 表示使用系统缓存。如果服务器配置有 UPS 不间断电源, 配置 off。没有 UPS 不间断电源时, 配置 on, 配置 on 时, 需要服务器配置带电池和回写缓存的 raid 卡以便为大规模标签应用提高写入效率。一般几万点应用即使使用普通台式兼容电脑, 怎么配置都能满足 IO 需求。上 10 万点标签推荐使用 HP, IBM 和 DELL 的专业服务器, 配置带电池和回写缓存的 raid 卡, 并开启 RAID 卡的回写缓存。

httpsrv.ini 和 httpsrv.conf 文件为 windows 下和 linux 下提供 http 服务和 websocket 服务的配置文件, 是存储的 html body 类型的 mime 配置, 请勿修改。

httproot 为 http 发布的根目录。

ca\_public 为服务器 https(wss)证书,(x509 v3 DER 格式),必填, 如果是自签发的证书, 要使用 SAN 扩展, 否则 Google 的 Chrome 浏览器不认。

ca\_private 为服务器证书的私有 KEY(PEM 格式), 要保护好, 最好将所在目录设置为只有 root 用户可以访问。

所有的默认端口都是 921，如果没有证书则不会开启 HTTPS 和 WSS 服务。如果提供了错误的 HTTPS 证书，则整个 RDB 的服务会启动失败，RDB 启动也会失败，日志里面将会有记录失败原因。

注意当电力系统的生产网(内网)和管理网(外网)之间有单向隔离网闸时，可以使用内外网各部署一个台实时库的方式，或者更经济的方式(rdb2020.6 版开始支持)，只在外网部署一台实时库，开启 [mirror\_in]，在数据网关 rdbdac\_ux 部署在内网，在 rdbdac\_ux 的 url 里使用 gapudp 或者 gaptcp 协议连接实时库。

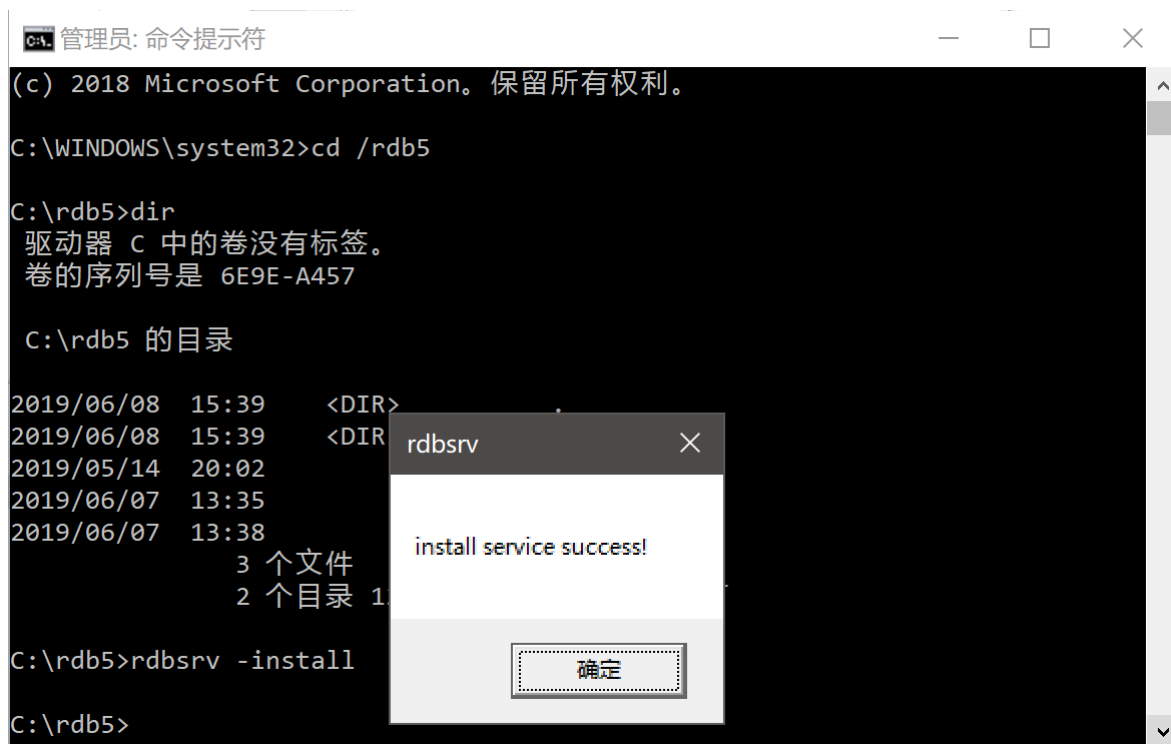
## 2.6.2 Window 安装

windows 版为运行在后台的 Service 服务程序，确定服务器安装的是 64 位版的 windows server2008 或 windows server 2012，使用管理员登录系统，将 rdbsrv.exe 个 rdbsrv.ini 和 httpsrv.ini 复制到服务器，比如复制到 C:/rdb5 目录下。

在管理员权限的命令行窗口，当前目录在 c:/rdb5 目录下，运行如下命令：

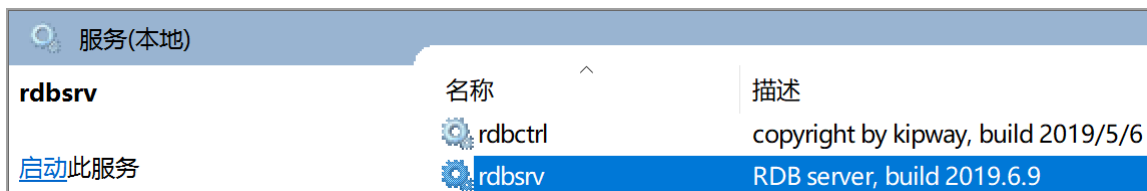
```
rdbsrv -install
```

注册为 Windows 后台服务。



这时在 Windows 的服务中能看到 rdvsrv 服务，检查配置无误后启动该服务。

首次安装后，需要手工点击启动按钮启动服务，以后会随系统的系统自动启动。如果启动不成功，请到你配置的日志目录下查看日志。如果没有日志，可能 rdbsrv.ini 中日志目录都配置错了，检查 rdbsrv.ini 的配置。



如果你的系统有软件防火墙，请在防火墙中例外实时库使用的 TCP 服务端口，在 `rdbsrv.ini` 中的最后一个配置项，默认是 921，如果有开启镜像转发，还需要在防火墙中例外 921 的 UDP 端口。

如果一切顺利，实时库就安装并启动好了，可以使用客户端工具 `rdbman` 导入标签，或者使用你自己的程序使用 `rdbapi` 接口的标签管理接口写入标签。标签写入并配置好后，即可写入数据了和对外提供数据服务了。

## 2.6.3 Linux 系统下安装

Linux 系统支持 `centos server7.5` 和 `ubuntu server 1604`，在 `rdbsrv` 目录的 `linux64` 目录中。`centos` 比较保守，内核版本和 `glibc` 版本比较低，可以在 `suse` 或者 `ubuntu` 上使用，反之则不行。

在 linux 上使用的是 `systemd` 守护进程，在各个发行版上的安装方式完全一样。

将 `rdbux64`，`rdbux64.conf`，`httpsrv.conf` 三个文件复制到 `/usr/sbin` 下，如下命令(假设在 windows10 下安装了 `putty` 或者安装了 `linux` 子系统，`linux` 目标系统在 192.168.1.230 上)：

```
scp rdbux64 root@192.168.1.230:/usr/sbin
scp rdbux64.conf root@192.168.1.230:/usr/sbin
scp httpsrv.conf root@192.168.1.230:/usr/sbin
```

使用 `ssh` 到目标系统上，为 `rdbux64` 加上可执行权限：

```
chmod +x /usr/sbin/rdbux64
```

将 `rdb5.service` 文件复制到 `/etc/systemd/system` 下：

```
scp rdb5.service root@192.168.1.230:/etc/systemd/system
```

将 `rdb_http` 复制到目标系统 `/home` 下：

```
scp -r rdb_http root@192.168.1.230:/home
```

检查配置后，在 `root` 权限下执行命令：

```
systemctl enable rdb5
```

```
systemctl daemon-reload
```

如果没有提示错误，则安装 `systemd` 服务成功。

安装成功后，重启服务器后 `rdbux64` 进程会自动启动，或者使用 `systemctl` 命令启动 `rdb5` 服务。

启动：`systemctl start rdb5`

停止：`systemctl stop rdb5`

状态: `sudo rdbux64 -status`

版本: `sudo rdbux64 -ver`

## 2.6.4 初始默认账号

实时库安装完成后，只有一个全部权限的管理账号 `admin`

账号: `admin`

密码: `admin`

利用管理工具 `rdbman` 或 WEB 版 `dbman` 可以定义添加自定义用户角色和账号，并修改默认账号 `admin` 的密码。

自 2023.4 版(内部版本 5105)开始，默认生成 4 个账号：

账号	密码	说明
<code>admin</code>	<code>admin</code>	默认的管理账号
<code>ioserver1</code>	<code>ioserver1</code>	默认的 <code>ioserver</code> 数据提交账号
<code>operator1</code>	<code>operator1</code>	默认的操作员账号
<code>reader1</code>	<code>reader1</code>	默认的数据只读账号

## 2.7 管理维护

提供了 `rdbman` 用于管理实时库，是一个图形客户端软件，分为 windows 版和 Linux 版，界面操作和功能完全一样。

主要功能分为账号管理，标签管理，数据查询。可以同时连接多个 TOM 实时库。

### 2.7.1 Windows 系统下安装

绿色软件，无附加依赖包，拷贝即可使用，将 `rdbman.exe` 和 `rdbapi.dll` 复制到磁盘即可，保证 `rdbman.exe` 和 `rdbapi.dll` 在相同目录。然后运行 `rdbman.exe` 即可。

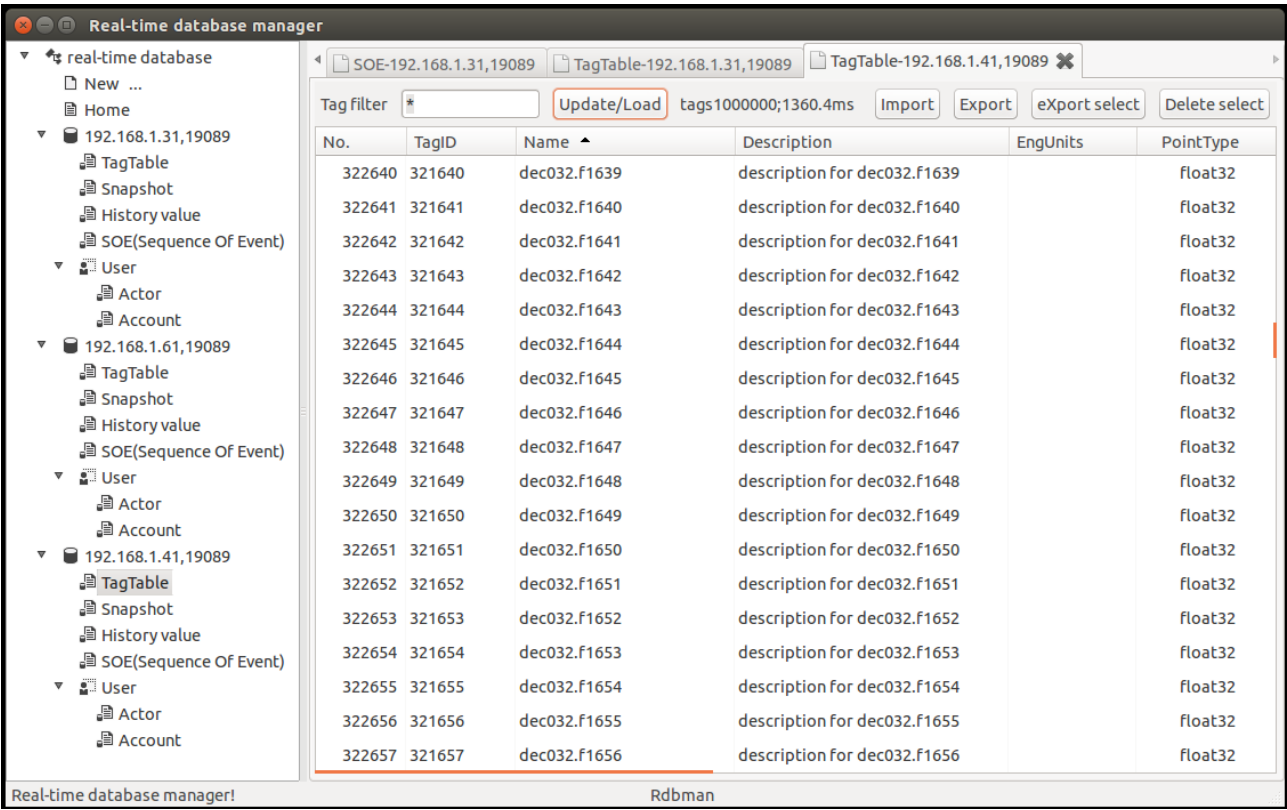
windows 系统下为 X86 版，字符集为 Unicode 编码，支持 windows7 及以后的服务器和桌面版系统。

### 2.7.2 Linux 系统下安装

Linux 下为基于 GTK 图形界面的应用程序，将 `rdbman` 文件和 `librdbapix64.so` 文件拷贝到系统，比如拷贝到 `/home/rdbman` 目录，保证 `rdbman` 文件和 `librdbapix64.so` 在同一个目录即可。为 `rdbman` 加上可执行属性。然后运行 `rdbman` 即可。在 Linux 系统为 X64 版，字符 Unicode 编码，如果要显示中文，请为系统添加中文显示字符集。

### 2.7.3 维护

可对进行账号管理、标签管理、数据查询等操作。可连接多个实时库。标签管理采用导入导出方式，导出文件问 csv 格式，可用 Excel 编辑。下面为 Ubuntu16.04 桌面版下的截图。



### 2.7.4 标签导入 CSV 文件格式

使用 TOM 实时库提供的例子标签 CSV 文件，其中的各属性列按照下表填写。

表 2-6 标签 csv 文件填写规范表

csv 列名	属性	可选值	说明
Name	标签名	下横线_或大小写字母或汉字开头,后跟数字或大小写字符或汉字的 0 结束的字符串，汉字为 GB2312 编码。	下横线_或大小写字母或汉字开头,后跟数字或大小写字符或汉字的 0 结束的字符串，汉字为 GB2312 编码。作为客户端访问实时库的标签索引，全局不重复。不能含有西文括号和西文单双引号这 4 个字符
Description	描述	0 结束的任意字符串	小于 80 字符
EngUnits	单位	0 结束的任意字符串	小于 16 字符
PointType	标签类型	Digital Int32	不分大小写

		Float32 Int64 Float64 String Object	
CompType	压缩方式	No Percent abs Timer Extimer	不分大小写,Timer 和 Extimer 在
ComDev	压缩精度	浮点数	比如 0.1
ComMin	最小压缩周期	整数	单位秒，默认 0
ComMax	最大压缩周期	整数	单位秒，默认 3600
ExcDev	例外偏差精度	浮点数	比如 0.05
ExcMin	最小例外周期	整数	单位秒
ExcMax	最大例外周期	整数	单位秒
Class	标签分类	DEC DEFINE PRESET CURVE PLAN	不分大小写
Step	梯形插值	ON OFF	OFF 不使用梯形插值，默认使用线性插值 ON 使用梯形插值，Digital 强制为梯形插值
Archiving	归档存盘标志	ON OFF	ON 存盘 OFF 不存盘
Alarm	越限报警配置	整数	具体见 2.3.3
Alarm_LL	低低限值	float	
Alarm_L	低限值	float	
Alarm_H	高限值	float	
Alarm_HH	高高限值	float	

## 2.7.5 使用 WEB 页面管理维护

从 rdb2022.3 开始，提供了全功能的 WEB 页面管理维护 dbman，需要配置实时库的[netsrv]小节中的 httproot 目录。下面以 Windows 版的配置为例说明。

[netsrv]

port = 921 ;服务端口

---

`ca_root =` ; TLS1.2 服务器根证书全路径文件, 可以不填写。

`ca_public =` ; TLS1.2 服务器证书全路径文件, 含 RSA 公钥, 空表示没有证书, 无法开启 HTTPS 和 WSS 服务。

`ca_private =` ; TLS1.2 服务器私钥全路径文件, 含 RSA 私钥, 空表示没有证书, 无法开启 HTTPS 和 WSS 服务。

`httproot = c:/rdb_http` ; http/https 的 html 文档根目录

将软件包中 `rdbsrv/rdb_http` 拷贝到配置的 `httproot` 目录位置, 且检查子目录 `dbman` 存在。假如服务器的 IP 地址为 192.168.1.214, 在客户端浏览器输入 url

`http://192.168.1.214:921/dbman/index.html`

登录账号为安装后默认的账号, 参见 2.6.4 初始默认账号。



### 3.生产数据网关

生产系统的数据可以使用 rdbapi 提交到实时数据库中。在实际应用中可以使用 C/C++，C#等编写应用程序使用调用 rdbapi 接口函数提交数据。

为了方便驱动开发和生产使用，RDB5 提供了两个数据网关，分别用于不同用途。数据网关提供上行（数据采集）和下行（设备控制输出）基本功能外，还提供数据采集值工程值转换，数据预处理。

- 1) 前台界面版驱动调试用网关 rdbdac\_wx，是 windows 7 及其以后系统图形界面版。主要用于调试系统集成商自己开发的设备驱动，不具备双机热备冗余功能和多实时库数据提交功能。不建议生产环境使用。
- 2) 多进程版 IOserver 数据网关，驱动以进程方式存在并通过内部消息总线交换数据，后台服务模式，故障隔离，某个驱动的崩溃不会影响其他驱动。IOserver 支持双机热备冗余和驱动设备级故障迁移，支持多库提交数据，每个库均支持主备库故障切换。是推荐使用的生产数据网关。支持 WEB 管理维护。日志输出到专用日志服务器 logsrv 中，使用浏览器接入查看日志。

以上两种数据网关使用完全相同的驱动程序，目前系统提供常用的三种设备驱动，并开放设备驱动规范，系统集成商可以自己开发专有设备驱动，使用 rdbdac\_wx 调试完成后再 IOserver 中使用：

- 1) OPCUA 新规范跨平台的 OPC 统一接口(dac\_opcua.dll/dac\_opcua.so, dac\_opcuax.dll/dac\_opcuax.so)
- 2) MODBUS-RTU 基于 TCP/RS485 的 modbus rtu 协议(dac\_modrtu.dll/dac\_modrtu.so)。
- 3) OPC DA 2.0 老标准基于 COM/DCOM 模式的 OPC DA 协议(dac\_opcdrv.dll)

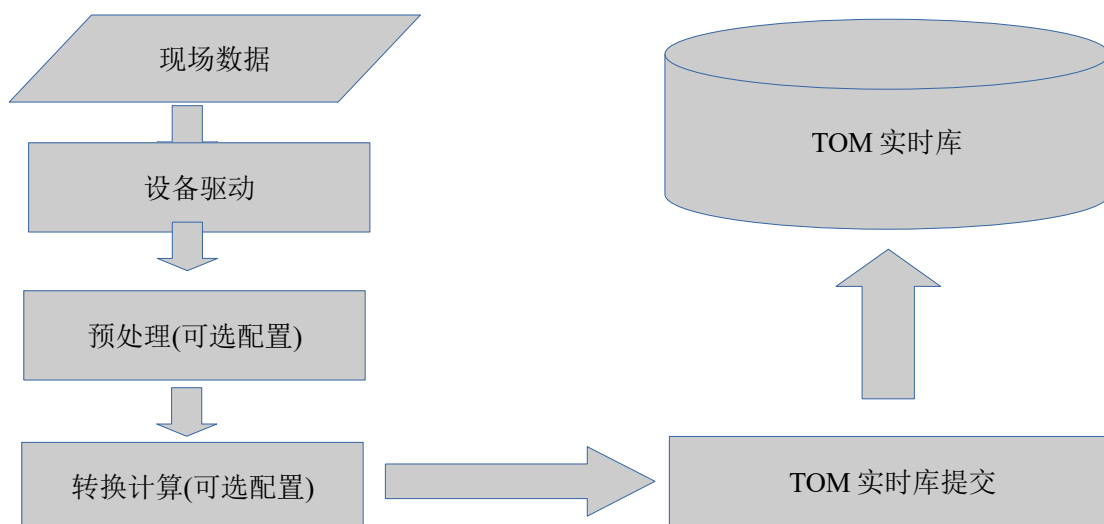


图 3-1 数据采集流程图

另外，网关除了采集数据上传数据的基本功能外，还提供了三个附加功能：

- 1) 标签值预处理，用于过滤和修正错误的值
- 2) 计算标签,用于采集值和工程值得转换，双向转换。
- 3) SOE 事件生成,用于产生越限和变位事件

如果工程应用有以上附件功能需求，可以按照 rdbdrv sdk 目录下提供的驱动规范编写设备驱动程序，然后挂在数据网关下运行。

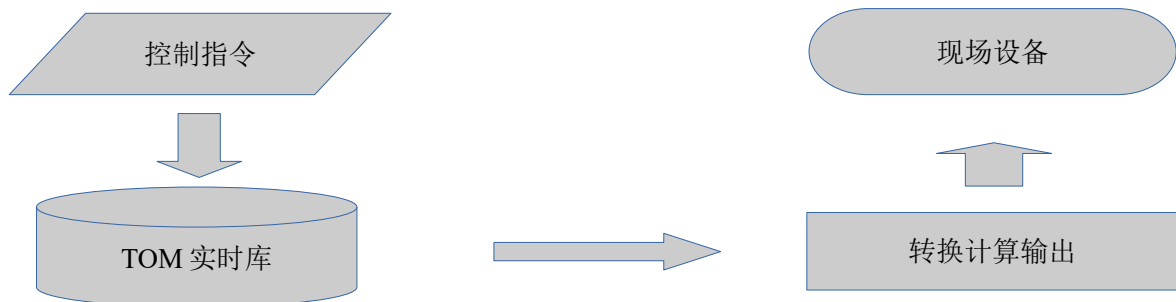


图 3-2 控制输出流程图

### 3.1 安装

后台版的 IO Server 安装和使用请阅读《ioserver 网关使用说明.pdf》，前台版的调试用 rdbdac\_wx 是绿色软件，拷贝即可使用。

具体的配置参见软件包里的例子工程。

### 3.2 OPCDA 驱动

OPC 驱动支持 OPC DA2.0 同步和异步方式，驱动程序名为 dac\_opcdrv.dll 需要本系统提供的 ezopc\_c.dll 支持。如果系统中没有安装 OPC DA 的 COM 代理库，需要安装 OPC 基金会提供的 COM 代理库，或者在本系统提供的 rdb\_opcgw/opcproxy 中安装从 OPC 基金会提供的 COM 代理库中提取出来的只含 OPC DA 的代理库。

#### 3.3.1 OPCDA 参数配置

TOM 实时库提供了 OPC DA2.0 客户端的驱动程序例子。其配置文件如下：

```
; real-time database v5.0 OPC 设备驱动配置

[opc]

progid = tom.opctst_w32.1 ;也可以填写 clsid，格式为注册表格式{8AF72085-716B-4963-
B5FF-14AADD3FB5}

ipaddr =                ;OPC 服务器所在机器名或 IP 地址，本机不填写。

syncread = no           ;是否同步读,yes 表示采用同步。

uselocaltime = no       ;是否采用本地时标，当 syncread=yes 是有效。
```

;当有些 opc server 长时间不推送不变位的开关量时，syncread 和 uselocaltime 设置为 yes 可能会解决问题。

[tag]

tagfile = opc1tag.csv ;标签文件

当在 OPC Server 在局域网其他机器上不在本机时，因 OPC 是基于 windows DCOM 协议的，需要配置 DCOM 以便双方具备访问权限。具体可参照其他资料，如果不同，一般都是防火墙或者 windows 登录账号权限问题。使用第三方的 OPC client 调试通局域网访问 OPC Server 后本系统就可运行了。

### 3.3.2OPCDA 标签表

其中的标签表格式是每个驱动程序自己定义的，不做任何要求，但是都一般都包含实时库标签名，标签数据类型，是否可写，工程单位，描述。标签表的配置参见例子，其中 A1 单元格为文件格式标识字符串“dacopcv5”，不能更改否则 dac\_opcdrv.dll 驱动不认。下图是 OPC 驱动的标签表格式。

表 3-1 OPCDA 标签表属性

rdbtagname	实时库标签名
opc_tagname	opc server 中的标签名
datatype	数据类型，按照 2.7.4 的 PointType 要求填写，即实时库的数据类型。
access	访问方式,r 表示读，rw 表示读写，带有 w 的表示可以写，会被 rdbdac_ux 注册到实时库当作可 device 写方式的标签，即控制标签，其他客户端如果有写 device 权限，即控制权限，就可向这个设备发送下传控制数据。
Unit	工程单位。
Description	描述

## 3.4MODBUS 驱动

本系统提供的 MODBUS 驱动协议层支持 RTU 协议，通讯层支持 TCP 网络和 RS485 总线通讯。

MODBUS 的串行和 TCP 报文依据 <http://www.modbus.org/specs.php> 中提供的协议文档：

- Modbus\_Application\_Protocol\_V1\_1b3.pdf (应用层协议定义)
- Modbus\_Messaging\_Implementation\_Guide\_V1\_0b.pdf (TCP/IP 通讯层定义)
- Modbus\_over\_serial\_line\_V1\_02.pdf (RS485/422/232 串行定义)

### 3.4.1MODBUS 参数配置 For Windows

Modbus 在 windows 和 Linux 下配置稍有差别，主要是 COM 口的差别。

例子配置如下：

;modbus config.ini

;mod1cfg.ini

---

[modbus]

comargs = 127.0.0.1,502 ;通讯参数, 自动识别

;TCP 配置 192.168.1.41,502

;格式: ip 地址,端口

;例子: 192.168.1.41,502

;-----

;RS232/485/422 配置格式:

;格式: 通讯口,波特率,数据位,校验方式,停止位

;例子: com1,9600,8,even,1

;COM 口: 为 com1, com2, com3, 到从 com255 之一。

;波特率: 2400, 4800, 9600, 19200, 38400,57600 之一。

;校验方式: none, odd, even, mark, space 之一。

;停止位: 1, 1.5, 2 之一

;-----

tagfile = mod1tag.csv ; 标签配置表

interval = 1000 ;[100,86400000]轮询间隔,单位毫秒,大于等于 100,小于 86400000

slavetimeout = 200 ;[100,1000],Slave 设备响应超时,单位毫秒。

;不包括通讯时间,大于等于 100,小于 1000,一般设置 200

logcompkg = no ;[yes,no] 是否输出通讯报文到日志,yes 要输出, no 不输出。

frmregs = 120 ; [16,120], 默认值 120, 轮询报文每次通讯最大寄存器数。01,02 命令自动 16 倍。

blockpoll = true ; [true,false], 默认值 false。

; true 表示按照块轮询寄存器(配置可以不连续, 会读取到不需要的寄存器)

; false 表示每次严格按照配置读连续的寄存器。

大部分参照例子配置即可。TCP 和 RS485 的 MODBUS 设备只有通讯参数不一样，其他都一样。

通讯参数用字符串表示，中间用西文逗号分开，TCP 只有 IP 地址和端口格式。RS485 总线 MODBUS 标准推荐的是 9600 波特率，8 位数据位，偶校验，1 位停止位。

logcompkg 一般配置为 no，只有在需要调试 MODBUS 设备通讯故障时需要开启并配置为 yes。

### 3.4.2 MODBUS 参数配置 For Linux

例子配置如下：

```
;modbus config.ini
```

```
[modbus]
```

```
;comargs = 192.168.1.8,502 ;通讯参数，自动识别
```

```
;TCP 配置 192.168.1.41,502
```

```
;格式: ip 地址,端口
```

```
;例子: 192.168.1.41,502
```

```
;-----
```

```
comargs = ttyS0,9600,8,even,1 ;通讯参数，自动识别
```

```
;RS232/485/422 配置格式:
```

```
;格式: 通讯口,波特率,数据位,校验方式,停止位
```

```
;例子: ttyS1,9600,8,even,1
```

```
;COM 口: 为 ttyS1, ttyS2, ttyS3, 到从 ttyS1 到 ttyS32 之一。
```

```
;波特率: 2400, 4800, 9600, 19200, 38400,57600 之一。
```

```
;校验方式: none, odd, even 之一。
```

```
;停止位: 1, 2 之一
```

```
;-----
```

```
tagfile = mod_fh_tag.csv ; 标签配置表
```

```
interval = 1000 ;[100,86400000]轮询间隔,单位毫秒,大于等于 100,小于 86400000
```

```
slavetimeout = 200 ;[100,5000],Slave 设备响应超时,单位毫秒.
```

```
;不包括通讯时间,大于等于 100,小于 1000,一般设置 200
```

`logcompkg = no` ;[yes,no] 是否输出通讯报文到日志,yes 要输出, no 不输出。

`frmregs = 120` ; [16,120], 默认值 120, 轮询报文每次通讯最大寄存器数. 01,02 命令自动 16 倍。

`blockpoll = true` ; [true,false], 默认值 false.

    ; true 表示按照块轮询寄存器(配置可以不连续, 会读取到不需要的寄存器)

    ; false 表示每次严格按照配置读连续的寄存器。

注意: 如果使用了 USB 串口转换器, 那么 COM 口可能为 ttyUSB0,ttyUSB1 等。在 Linux 中查看串口使用如下命令

```
dmesg | grep tty*
```

3.4.3MODBUS 标签表

按照 MODBUS 的规范, 数据按照寄存器组织, 有 1bit 寄存器和 16bit 的寄存器, 16bit 也叫 WORD 或中文的“字”, 大于一个 WORD 的数据采用连续的多个 WORD 寄存器存储,字节顺序约定为大头格式 (big-Endian), 即高字节在前(低地址)。

寄存器地址约定, 采用 MODBUS MASTER 协议地址, 使用 6 位 10 进制数表示地址:

表 3-2 MODBUS 寄存器地址定义表

寄存器地址	说明
1-65536	Coils 输入: 对应 MODBUS 命令 0x01 地址 0-0xFFFF 输出: 对应 MODBUS 命令 0x05/0x0F 地址 0-0xFFFF
100001-165536	Discrete Inputs 输入: 对应 MODBUS 命令 0x02 地址 0-0xFFFF
300001-365536	Input Registers 输入: 对应 MODBUS 命令 0x04 地址 0-0xFFFF
400001-465536	Holding Registers 输入: 对应 MODBUS 命令 0x03 地址 0-0xFFFF 输出: 对应 MODBUS 命令 0x06/0x10 地址 0-0xFFFF

为了便于配置, 本系统约定 MODBUS 的数据类型如下: 配置是也可以填写括号里的类型, 比如 i2 等价于 int16

表 3-3 MODBUS 数据类型定义表

MODBUS 数据类型	描述
-------------	----

bit	1 位, MODBUS MASTER 协议寄存器地址为 1-65536 和 100000-165536 使用。
i2(int16)	16 位带符号整数,占 MODBUS MASTER 协议寄存器地址为 300000-365536 和 400000-465536 一个地址
ui2(uint16)	16 位无符号整数,占 MODBUS MASTER 协议寄存器地址为 300000-365536 和 400000-465536 一个地址
i4(int32)	32 位带符号整数,占 MODBUS MASTER 协议寄存器地址为 300000-365536 和 400000-465536 连续两个地址
ui4(uint32)	32 位无符号整数,占 MODBUS MASTER 协议寄存器地址为 300000-365536 和 400000-465536 连续两个地址
f4(float)	32 位 IEEE754 浮点数,占 MODBUS MASTER 协议寄存器地址为 300000-365536 和 400000-465536 连续两个地址
f8(double)	64 位 IEEE754 浮点数,占 MODBUS MASTER 协议寄存器地址为 300000-365536 和 400000-465536 连续四个地址
i8(int64)	64 位带符号整数,占 MODBUS MASTER 协议寄存器地址为 300000-365536 和 400000-465536 连续四个地址
ui8(uint64)	64 位无符号整数,占 MODBUS MASTER 协议寄存器地址为 300000-365536 和 400000-465536 连续四个地址

标签表的配置参见例子, 其中 A1 单元格为文件格式标识字符串“modbustags”, 不能更改,否则 dac\_modrtu.dll 驱动不认。

表 3-4 MODBUS 标签表属性定义:

UnitID	设备地址, 总线上唯一。
RegAddr	寄存器地址, 6 位 10 进制表示法。
DataType	MODBUS 设备的数据类型。
RdbTagName	实时库表签名
RdbDT	实时库数据类型
EngUnits	工程单位
Description	描述
byteorder	4 字节和 8 字节数据的字节顺序(从 2020.7 版开始两字节的数据也支持位置交换), 不填写为默认模式, 填写 inverse 表示 big-endian 模式(即 modbus_slave 工具中的 float_inverse, long_inverse 等)

	<p>float 为例:</p> <p>默认模式: LH LL HH HL 或 Modbus Slave( CD AB )</p> <p>inverse 模式: HH HL LH LL 或 Modbus Slave( AB CD)</p> <p>从 2021.11 版开始, 增加下面 6 种模式:</p> <p>WORD 两种模式:</p> <p>12 : LH; 小头 WORD, 等于原 inverse 模式</p> <p>21 : HL; 大头 WORD, 原默认模式</p> <p>双 WORD,4 种模式: 适合 int32, float32</p> <p>1234 : LL LH HL HH; 4 字节</p> <p>2143 : LH LL HH HL; 原默认模式或 Modbus Slave( CD AB )</p> <p>4321 : HH HL LH LL; 原 inverse 模式或 Modbus Slave( AB CD)</p> <p>3412 : HL HH LL LH;</p> <p>四 WORD,4 种模式: 适合 int64, float64</p> <p>1234: DWL(LL LH HL HH) DWH(LL LH HL HH), 即字节顺序为“12345678”;</p> <p>2143 : DWL(LH LL HH HL) DWH(LH LL HH HL) , 即字节顺序为“21436587”;</p> <p>原默认模式或 Modbus Slave( GH EF CD AB )</p> <p>4321: DWH(HH HL LH LL) DWL(HH HL LH LL) , 即字节顺序为“87654321”;</p> <p>原 inverse 模式或 Modbus Slave( AB CD EF GH)</p> <p>3412: DWH(HL HH LL LH) DWL(HL HH LL LH) , 即字节顺序为 “78563412”;</p>
--	--

MODBUS 的例子标签表如下图:

Linux 和 windows 版的标签表是完全一样的。

从 2020.7 版开始, modbus 驱动支持驱动级别的位分解。寄存器后面用小数点分开后跟 bit 位, 0 表示 bit0, 比如 300050.0 表示 bit0 位, 300050.3 表示 bit3 位, modbus 类型填写 bit, 实时库类型填写 digital, 如下

1,300050.0,bit,mod1.bit5000,digital,,300050

1,300050.1,bit,mod1.bit5001,digital,,300050

经过以上配置后, rdbdac\_ux 会自动在 MODBUS 的数据类型和实时库数据类型之间转换。但是, 很多 MODBUS 设备采用定点数据约点小数点方式表示浮点数。比如 MODBUS 使用 16 位整数 (即一个字) 值 12345, 来表示小数 123.45, 这是通过以上标签表就不能正确转换, 还需要配合计算表达式来转换, 具体参见《ioserver 网关使用说明.pdf》中的采集值工程值转换。

### 3.5 OPCUA 驱动

OPC UA 适用于现场设备, 控制系统, 制造执行系统和企业资源规划系统等应用领域的制造软件。这些系统旨在交换信息, 并为工业过程使用命令和控制。OPC UA 是一种独立于平台的标准, 各种系统和



---

设备可以通过各种类型的网络在客户端和服务端之间发送消息进行通信。具体参见 OPC 官方网站 (<https://opcfoundation.org/>) 的介绍。

目前提供了两个 OPCUA 驱动, `dac_opcua.so`(不支持证书)和 `dac_opcuax.so`(支持证书), 下面使用支持证书的 `dac_opcuax.so` 作为例子说明, 具体可参见《OPCUA 驱动使用说明.pdf》

### 3.5.1 OPCUA 参数配置

下面以 linux 系统 `opcuax1.ini` 为例:

**#opc ua 客户端配置**

**[opcua]**

**#server 的连接点 URL**

**endpointUrl = opc.tcp://127.0.0.1:53530/OPCUA/SimulationServer**

**#通讯层安全策略, 填写, None, Basic256Sha256, Basic256, Basic128Rsa15, 其中 Basic256, Basic128Rsa15 已经弃用。**

**securityPolicy = Basic256Sha256**

**#客户端自己的应用标识, 只有使用加密安全策略时采用, 必须设置和证书里使用者可选名称中 URL 参数相同**

**applicationUri = urn:kipway.client.application**

**#客户端自己的证书文件名, 只有使用加密安全策略时采用, 必须加入 opcua server 的信任证书列表, 或者从 opcua server 那里获取应用端证书**

**certFilename = /etc/opcuacert/kipway\_uaclient\_cert.der**

**#客户端私钥文件名, 妥善保管, 防止泄露, 只有使用加密安全策略时采用。**

**keyFilename = /etc/opcuacert/kipway\_uaclient\_key.der**

**#消息加密方式, 填写 0-3; 0: INVALID 无效; 1: NONE(无); 2: SIGN(签名); 3: SIGNANDENCRYPT(签名和加密)**

**#签名防止被篡改, 加密防止被偷窥, 配置 2, 3 必须要有证书。**

**msgSecurityMode = 3**

**#会话层用户验证-用户名; 不配置表示匿名**

username =

#会话层用户验证-用户密码

password =

[tag]

#标签表 csv 文件,自动识别文件内容编码

tagFilename = opcua1tags.csv

注:

- 1) 证书和私钥文件配置文件名为绝对路径文件名。
- 2) 标签表文件文件名放在 config 目录下,不带路径。可以是 utf8 编码,也可以是 gbk 编码,但是不要混合编码,否则汉字会出现乱码。
- 3) applicationUri 必须和证书里“使用者可选名称”的 URL 相同。
- 4) 运行 ioserver 的机器的 IP 必须在证书“使用者可选名称”的 IP 列表中。如果 IP 条件无法满足要求。还有两个方式:从 OPCUA server 哪里获取客户端应用证书或者自己制作自签名证书。注意此时要将证书里“使用者可选名称”中 URL 字段内的应用端标识 Uri 配置到 applicationUri 字段。
- 5) 将证书加入到 OPCUA server 的信任列表中。
- 6) dac\_opcuax 驱动是固定内置为信任对端证书的,因此无需配置 OPCUA server 的证书,也不需配置对端的证书信任列表,工程人员施工配置时,可以人工审核 OPCServer URL 指向的服务端证书是否可信任。
- 7) msgSecurityMode 填写的是数字,没有证书时填写 1,有证书时填写 0, 2, 3 均可。

如果连接失败,查看日志,检查错误原因,一般是 OPCUAServer 不支持指定的 securityPolicy, msgSecurityMode, 或者证书里的 applicationUri 和配置里不相同,或者客端 IP 不在客户端证书“使用者可选名称”的 IP 列表中。修改参数重新测试,或者从 OPCUA server 管理员处获取准确的安全策略和消息加密模式。

## 3.5.2OPCUA 标签表

标签表和 OPCDA 差不多,多一列 nsindex(命名空间索引,是一个 0-65535 的值),opc 标签名直接使用 nodeid 配置。

表 3-5 OPCUA 标签表字段属性定义

rdbtagname	实时库标签名
nsindex	opcua server 中标签的命名空间索引(namespaceIndex),从 OPCUA server 那里获取
nodeid	opcua server 标签的 nodeid(node identifier)。默认为字符串,从 rdb2020.8 开始支持数字,"i="开头为数字型,"s="开头为字符串型。例如下面三个都是支持的:

	i=10032 s=m1.intval2 m1.intval2
datatype	<p>数据类型:</p> <p>Digital 开关量</p> <p>Int32 4 字节整数</p> <p>Float32 单精度浮点型, 4 字节</p> <p>Int64 8 字节整数</p> <p>Float64 双精度浮点型, 8 字节</p> <p>String 字符串型</p> <p>-----</p> <p>下面是 rdb2022.7 开始支持分别配置实时库类型和 opcua 类型; 用于某些 opcua server 下行控制数据类型必须相同的场景</p> <p>格式</p> <p>实时库类型:opcua 类型</p> <p>中间是西文冒号分开, 前面是实时库类型, 后面是 opcua 类型。</p> <p>如果实时库类型和 opucua 类型完全相同, 只填写实时库类型, 不要冒号和 opcua 类型(兼容 rdb2022.7 以前版本)。</p> <p>opcua 特有类型定义如下:</p> <p>char 1 字节整数, 自动转换为实时库 int32</p> <p>byte 1 字节无符号整数, 自动转换为实时库 int32</p> <p>short 2 字节整数, 自动转换为实时库 int32</p> <p>word 2 字节无符号整数, 自动转换为实时库 int32</p> <p>dword 4 字节无符号整数, 自动转换为实时库 int32</p> <p>uint64 8 字节无符号整数, 自动转换为实时库 int64</p> <p>例子 1: 实时库 digital, opcua 是 char</p> <p>digital:char</p> <p>例子 2: 实时库是 float32, opcua 是 word</p> <p>float32:word</p> <p>例子 3: 实时库是 int32, opcua 是 short</p> <p>int32:short</p> <p>注意: 如果是只读标签, 不用配置 opcua 类型, IO Server 会自动转换, 按照以前版本配置即可。</p> <p>只有 rw 标签, 需要向下控制输出的, 才需要精确配置 opcua 类型, 以便不支持自动转换的 opcua server 产生 0x80740000(类型不匹配)错误。</p>
access	访问方式,r 表示读, rw 表示读写, 带有 w 的表示可以写, 会被 rdbdac_ux 注册到实时库当作可 device 写方式的标签, 即控制标签, 其他客户端如果有写 device 权限, 即控制权限, 就可向这个设备发送下传控制数据。

Unit	工程单位。
Description	描述

具体参见 /rdbdac\_ux/opcua1tag.csv 中的配置。注意 csv 文件的第一行和第二行不要改动。

### 3.6 预处理配置

预处理标签是对采集的数据进行初步过滤和处理，目前定义有以下两种预处理模式：

- 1) 小于阈值替换设定值或丢弃(del)
- 2) 大于阈值替换预定值或丢弃(del)

预处理标签采用 csv 配置表，格式如例子中的 predotags.csv

如下：

	A	B	C
1	predotags		
2	tagname	dotype	
3	opc21.r_i32	down:0=0;up:100=300	
4	opc21.r_f32	down:30=0;up:100=100	
5			

A1 单元格为标识字符串"predotags",不能更改，否则 rdbdac\_ux 不认。

tagname,dotype

opc1.r\_i32,down:0=0;up:100=300

opc1.r\_f32,down:30=0;up:100=100

opc1.r\_testi32,down:0=del; up:30000=del

第一行为标识行

第二行为表头

第三行开始为标签预处理配置

第一列为标签名，实时库标签名。

第二列为配置,预处理规则之间用分号隔开如下：

down:0=0;up:100=300

至少要有一列

---

规则:

**dwon** 表示下限规则

**up** 表示上限规则

格式:

规则名:阈值=设定值

比如:

**down:0=0**

表示小于 0 是设置为 0

**up:100=100**

表示大于 100 时设置为 100;

如果标签 **opc1.r\_f32** 只需要处理小于 0, 可如下配置

**opc1.r\_f32,down:0=0**

如果标签 **opc1.r\_if32** 只需要处理大于 100, 可如下配置

**opc1.r\_i32,up:100=100**

注意其中的逗号等号冒号不要写成中文符号了。

如果要配置小于 0 作废,大于 30000 也作废

**down:0=del; up:30000=del**

配置好预处理配置表后, 需要挂在 **rdbdac\_ux.ini** 上:

**[other]**

**predotags = predotags.csv** ;预处理标签配置,无可以不填写

在 **IOServer** 中, 则默认固定文件名为 **predotags.csv**, 具体参见《**ioserver** 使用说明》

## 3.7 计算标签配置

计算标签用于将采集值转换为工程值, 应用于以下场景:

- 拆分状态量: 采集一个 16 位整数表达的 16 个状态量的采集值, 需要将其转换为对应的 16 个工程量, 存储与事实库中的 16 个 **digital** 类型标签。
- 采集值工程值之间转换, 比如采集值是整数表达的小数, 需要除以 100 变成小数。
- 量程转换, 采集值对应工程值区间值。

具体参见《**ioserver** 使用说明.pdf》

---

### 3.8 ARM 平台嵌入式 Linux 系统数据采集终端

目前提供两种 ARM Linux 数据采集终端 GT6502, GT66575, GT6658 三种, 运行 ARM 版全功能 IO Server, 功能和用法和服务端版完全相同。可直接将现场 modbus 设备的数据采集后写入实时数据库, 如果有公网, 可以直接写入云实时库。高可靠性, 带硬件看门狗, 适合铁塔、油井等野外无人值守环境。

## 4.图形客户端软件(Process View)

Process View 图形组态软件是一个高可定制的平台软件，将生产现场用图形的方式直观表达出来。分为组态部分和运行部分，运行部分有分为 C/S 方式和 B/S 的 WEB 方式，其中 WEB 方式的图形界面和 C/S 表现出来的完全一样。

软件位于 Processview 目录主要包括两大部分：

- 组态工具软件 PVEdit，运行在 windows 平台，直接输出工况图和图表混排的 html 页面。
- 运行软件，C/S 架构的 PVRun 和 B/S 架构都是完全相同的 HTML5 方案，运行在 Windows，Linux（含国产 Linux 系统和 aarch64 平台）。

采用图形和动作逻辑分离的方式，图形的变化和表现形式采用动作触发，任何一个图形均可为其添加支持的动作逻辑，从而在运行时表现出动态属性。

动作分为直接连接和脚本两种方式，90%的应用均可使用直接连接，简单快捷，高级应用可采用脚本动作。动作的触发分为定时事件触发和事件触发方式。

自 ProcessView 5.0.8.1 版开始，已经完全采用 html5(svg+js+html+css+websocket)方案，C/S 运行端和 WEB 运行端完全一样。因此从该版本起，不再支持 OCX 控件和 PV 插件，标准控件采用 WEB 版的控件，同时不再支持原 C/S 版的子图，不再支持原 C/S 版的 CScript 脚本。

图形分为基本图形(线类、面类、文字类、图片类、矢量图类)、子图、标准控件、svg 的外部对象 (foreignobject)。基本图形可打包成子图，子图可直接编辑其内部基本图形属性和动作而不用解开。外部对象可以扩展监控视频，第三方图表，在 svg 中实现混排。

为方便使用，提供了对其、均分等自动排列工具，提供组合将一批图形元素的相对位置固定，且组合可缩放。提供界面图形元素的公共属性批量修改功能，使整个图形组态工作更加高效快捷。

画面的渲染包括画面的显示，动作的解析执行两大主要部分。

表 4-1 动作支持一览表

动作	支持图形	WEB	C/S	表达式
输出显示	标签值	V	V	数值,显示结果
颜色	除位图外的所有图形	V	V	数值，按照定义分段，显示对应段定义的颜色
填充色	静态文本，标签值，矩形，椭圆，封闭多边形	V	V	数值，按照定义分段，显示对应段定义的颜色
显示/隐藏	所有图形	V	V	结果不为 0 显示
闪烁	所有图形	V	V	结果不为 0 闪烁
移动	所有图形	X	V	用脚本驱动，仅 C/S 版支持。
百分比填充	封闭图形，位图	V	V	按照结果值所占量程比例填充
旋转	所有图形	V	V	WEB 支持连续旋转

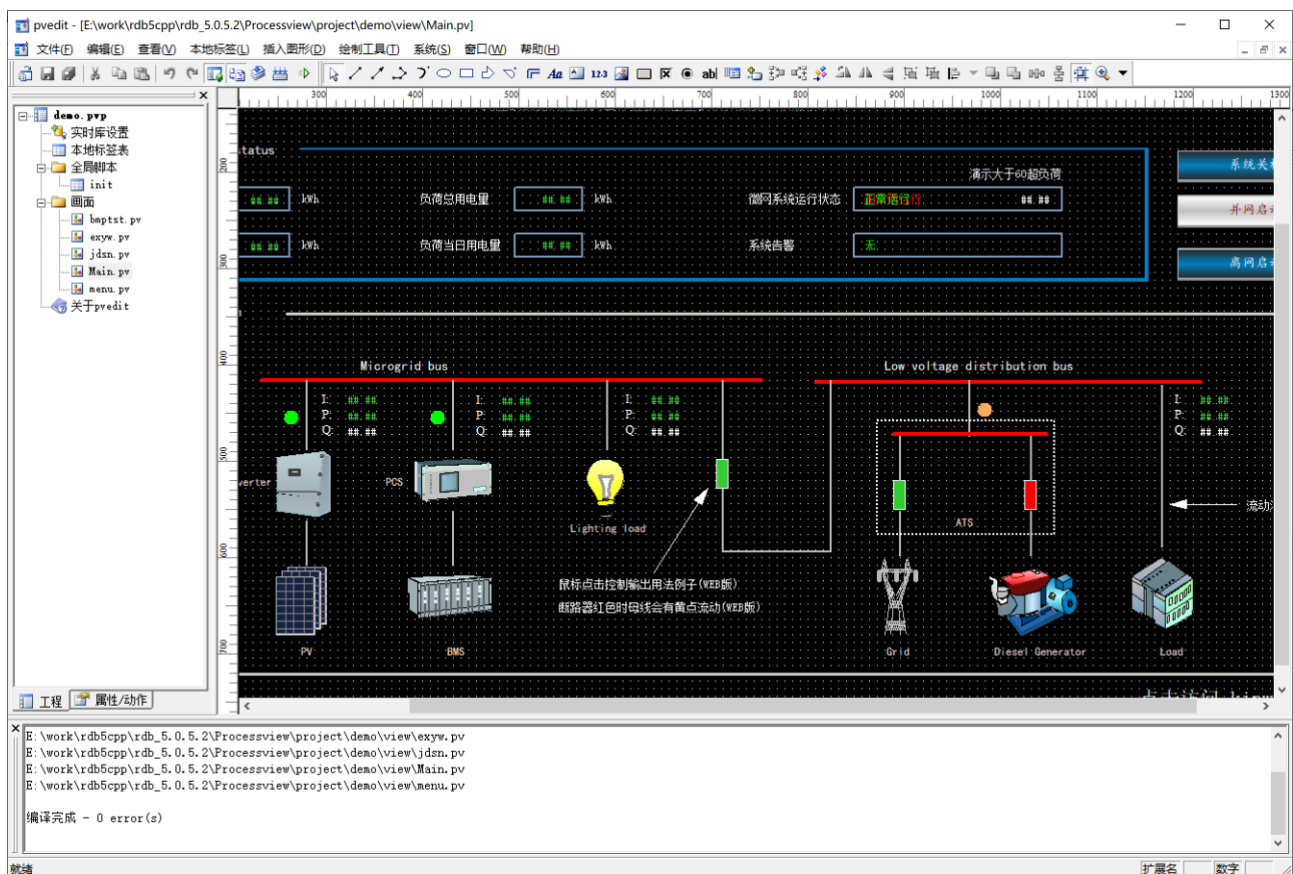
动作	支持图形	WEB	C/S	表达式
仪表指针	线，箭头	V	V	按照值对应量程旋转角度
流动	线，多线，箭头，管道	V	V	结果不为 0 流动,等于 0 则隐藏
HTML-Link	静态文本，填充矩形，填充椭圆，位图	V	V	和 Html 的 a 标签一样
鼠标事件	几乎所有的图形	V	V	使用 javascript 编写脚本。

注 1: V 支持; X 不支持。

注 2: pvedit 组态界面里还保留又原 C/S 版的内容,已经不起作用,后面会删除。插件扩展采用插入 WEB 块模式实现。

## 4.1 图形组态

下图为组态编辑软件界面,分为绘制区,编译输出区,属性区,工具条和菜单区。



使用 150% 页面放大可以看得更清楚,图形组态部分提供了一个例子工程,可以用 pvedit 打开这个 demo.pvp 工程仔细查看各种动作、脚本的用法,对照 pvruntime 运行部分看运行效果。组态完成后,需要完全编译一次,无错后方可发布。



---

## 4.2 扩展自己的 html 代码

有三种方式可以在组态生成的 html 页面中增加 html,svg,javascript 元素以扩展功能。

### 1. 公用的集成商自己的扩展

加入到 `html_head.html`, `html_svgstart.html`, `html_svgend.html`, 可以参考默认的实现。

### 2. 页面专属的 javascript 脚本

将脚本写入和画面相同, 后缀为 `.js` 的文件, 编译生成时会自动引入。参考例子工程的 `index.js`。

### 3. 画面专属的图表, 其他 svg 图形嵌入, 视频监控

插入 `foreignobject` 对象, 在 `pvedit` 画面界面菜单“插入图形”->“插入 WEB 块”菜单项, 在画面放置位置单击完成插入, 点击该图形, 左边属性页编辑内容。

参见 `index` 页面, 里插入的中国地图 `svg` 和画面导航下拉菜单。

下面是预置的通用 `web` 扩展(方式 1), 实现控制输出和历史趋势。

通过将扩展的 `html` 和 `js` 代码写在下面三个可配置文件里, 由 `pvedit` 在编译时插入到生成的 `html` 文件中实现。

`html_head.html` 被插入到 `<head>` `</head>` 之间

`html_svgstart.html` 被插入 `body` 里 `<svg>` 之前

`html_svgend.html` 被插入 `body` 里 `<svg>` 之后

依赖:

jquery 3.2.1

bootstrap 3.3.7

chartjs 2.9.4

已经写在 `html_head.html` 里自动插入。

浏览器支持 ES6 `async/await`

任何应用厂家都可以自己扩展, 这里实现一个扩展控制输出和历史趋势的 `web` 扩展共参考。任何应用厂家均可

免费使用本扩展和对本扩展做任何形式的修改而不需通知我们(包含对 `rdbview.js` 和 `rdb5api.js` 的使用和修改)

扩展配置: 参见 `ctrldemo.pv/ctrldemo.html`

`selfext` 在图形组态图形属性"自定义扩展"属性中定义扩展, 字符串, 不能有西文单引号(因为需要存入 `svg` 图形标签的 `selfext` 属性中),

图形元素不要求必须有一个直接连接动作绑定一个实时库标签。

---

这里实现的扩展采用 json 对象或对象数组配置。

#### 1.不弹窗直接输出

```
{"evt":"onclick","opt":"ctrl","tag":"ctrl.dbl0.pv","val":120.5}
```

输出 val 值

#### 2.不弹窗直接取反

```
{"evt":"onclick","opt":"ctrlnot","tag":"ctrl.k2.pv"}
```

当前快照为 0 输出 1，否则输出 0

#### 3.不弹窗增量输出

```
{"evt":"onclick","opt":"ctrladd","tag":"ctrl.dbl0.pv","val":2.5}
```

输出当前快照+val 之和，val 可以为负值

#### 4.弹窗确认输出

```
{"evt":"onclick","opt":"ctrlc","tag":"ctrl.dbl0.pv","val":100.5,"optdes":"行程设定值","title":"参数设置","txtbtok":"确认并输出"}
```

其中 optdes,title,txtbtok 可选，不填采用默认值

#### 5.弹窗取反输出

```
{"evt":"onclick","opt":"ctrlnotc","tag":"ctrl.k2.pv","title":"断路器操作","txtbtok":"确认并输出","txt0":"分闸","txt1":"合闸"}
```

其中 title,txtbtok, txt0, txt1 可选，不填采用默认值

#### 6.弹窗增量输出

```
{"evt":"onclick","opt":"ctrladdc","tag":"ctrl.dbl0.pv","val":2.5,"optdes":"行程设定值","title":"参数设置","txtbtok":"确认并输出"}
```

其中 optdes,title,txtbtok 可选，不填采用默认值

#### 7.弹窗输入值输出

```
{"evt":"onclick","opt":"ctrlp","tag":"ctrl.dbl0.pv","val":100.5,"optdes":"行程设定值","title":"参数设置","txtbtok":"确认并输出"}
```

其中 val,optdes,title,txtbtok 可选，不填采用默认值

输出操作者手工填写的值，val 作为默认值，不填会采用当前快照作为默认值，快照也没有会采用 0 值。

#### 8.历史趋势

```
{
```

---

```
"evt": "onclick",
"opt": "chart",
"title": "机组负荷趋势",
"tags": [
    {
        "tag": "d0.f01.pv",
        "lab": "1 号机组负荷"
    },
    {
        "tag": "d0.f02.pv",
        "lab": "2 号机组负荷"
    }
],
"args": {
    "timestart": "2025-5-10T08:12:32.100+08:00",
    "timearea": 7200,
    "type": 0,
    "xlab": "Time",
    "ylab": "Value"
}
}
```

**title** 可选，不填表示无趋势标题。

**tags** 为标签数组：建议不超过 5 个标签。

**args** 可选：**timestart** 格式 ISO 带时区时标，不填写默认值为当前两小时前。其余为上面默认值。

**timearea** 为窗口时间区域，单位秒；**type** 0 表示带质量，1 表示不带质量。

---

## 5.实时库客户端接口

rdbapi 是《rdb\_data\_exchange\_protocol.pdf》协议的一个 C/C++实现。应用程序访问 TOM 实时库可以直接使用 rdb\_data\_exchange\_protocol，也可以使用实时库系统提供的接口 API。系统自带的其他软件也是使用这个 API 接口，包括提供的 rdbman 管理工具以及前端数据网关 IOServer。

接口 API 提供 windows x86/x64 版和 Linux x64 版。windows 版采用标准 DLL 动态库封装，Linux 平台采用标准 SO 动态库封装，两个平台下的导出 C 接口完全一致，参数和用法也完全相同。

rdbdef.h 类型和数据定义

rdbapi.h 接口定义

这两个文件在 windows 和 Linux 下通用。具体可以看 rdbdef.h 中的注释或 rdbapi.chm 帮助文件。

windows 平台接口库，windows7 及以后系统版本：

rdbapi.dll X86 版，

rdbapix64.dll X64 版

Linux 平台：

librdbapix64.so x64 版，GCC 4.8.5 下编译

rdbapi 支持数据的提交和控制命令的下传，因此 TOM 实时库也可以当成生产系统的 DCS 系统使用。

### 5.1 例子工程 for C#

rdbapi 是标准的动态库，因此 C#完全可以使用，为了便于应用程序使用，提供了 rdbapi.cs 文件，从 C 接口直接翻译而来,用法和 C 接口完全一致。

在 C#目录下提供了 csapitst 工程，C#的开发工程师可以参考。

### 5.2 例子工程 for Java

JAVA 使用 rdbapi 有两种方法：

- 使用 JNA (java native access) 包装一下可以直接访问 C 标准动态 rdbapi 里的 C 函数。具体访问：[GitHub - java-native-access/jna: Java Native Access](https://github.com/java-native-access/jna) 用法不在本文讨论之中。
- 使用 rdbapi/javaws，一个用 C++开发的 jni 应用通信接口，提供了登录实时库，命令请求等接口，可以按照《rdb\_data\_exchange\_protocol.pdf》向实时库发送 json 消息和接收返回结果，并提供一个纯 java 实现的 json 解析类供选用。

第二种方法的 javaws 库在 src 目录提供了使用例子 tstwsapi.java

---

## 5.3 Javascript 接口

Html5 中的 Javascript 脚本直接使用《rdb\_data\_exchange\_protocol.pdf》协议连接实时库，具体用法可以参看 WEB 版管理工具 rdb\_http/dbman 里的 JS 脚本。

## 5.4 自己根据协议实现

实时库接口是居于 websocket 通道的 JSON 编码消息交互协议，《rdb\_data\_exchange\_protocol.pdf》中由详细描述核例子。如果需要，可以自己用熟悉的语言实现 API，比如直接用 Java 或者 C#实现。

## 5.5 OPC DA 实时数据服务

只适合 windows 系统，安装 rdb\_opcgw 后台服务，支持 opc da 的客户端可以直接订阅标签的实时数据。

## 5.6 OPC UA 实时数据服务

支持 windows 和 Linux 系统，参阅《rdb\_opcua\_server 使用说明.pdf》安装 rdb\_opcuasrv 后台服务，支持 OPC UA 的客户端可以直接订阅标签的实时数据。

## 6.工程配置技巧

### 6.1 标签的压缩方式选择

实时数据库最关键的技术并不在实时，而在海量历史数据的存储和检索，根据标签数据的应用特点合理的选择压缩(存储)模式可以极大的节约存储空间，减少数据量，加快历史数据的读取。

在实时库 file version 5.0.3.3 以及之后，增加了按照最大压缩周期间隔存储(Timer)和增强的周期间隔存储(Extimer)两种模式。这两种模式比较符合传统的组态软件的数据存储方式。特别适合缓慢变化的数据，并且不需要过度关注变化细节的场景，比如，水位，大气温度这样的数据，每 5 分钟存储一个记录就满足应用要求了。

表 6-1 标签压缩方式

压缩方式	名称	原理	适用场景
Percent	百分比精度趋势压缩	见 2.4，使用百分比精度判定是否在趋势内。一般配置 0.1 表示千分之 1 精度。	适合在满足百分比精度要求的前提下记录数据变化的全过程。不确定数值大小，按照百分比确定偏差。
Abs	绝对值精度趋势压缩	见 2.4，使用绝对值偏差判定是否在趋势内。比如大气温度可以设置 0.2 度绝对偏差。	适合在满足偏差精度要求的前提下记录数据变化的全过程。确定数据在某个范围，按照绝对值确定偏差。
Timer	周期存储	按照最大压缩周期时间间隔存储数据记录。一个最大压缩周期存储 1 条记录。	缓慢变化的量，或者波动不大的量。比如，水位，大气温度。或者应用上只需要固定间隔存储的量。
Extimer	增强周期存储	按照最大压缩周期时间间隔存储数据记录，同时存储周期内的最大值和最小值记录。一个最大压缩周期内存储最多 3 条记录。	缓慢变化的量，或者波动不大的量。比如，水位，大气温度。或者应用上只需要固定间隔存储的量。

合理的运用周期压缩存储(Timer 或 Extimer)，可以极大的节约存储空间。比如按照 5 分钟增强周期存储(Extimer)，5 分钟内只需存储 3 条记录。如果每秒收到一条实时数据，实际只存储了 1/100 的记录，折算压缩倍速就是 100 倍。

通常，如果不清楚数据的应用需求，也不太确定数据的大小，可以按照默认配置 Percent 模式精度设置 0.1 表示千分之一，或者 0.05 表示万分之五。

## 7. 嵌入版实时数据库

嵌入版实时库是一个运行在 ARM Linux 平台的全功能实时数据库服务端，功能和 PC 服务器版相同，规模限制在 2000 标签点。内置 EMMC 5G 预分配存储空间循环使用，2000 点正常配置压缩可存储最近 3-5 年数据。

访问接口协议级兼容配套的 PC 版 rdbapi，可以使用 PC 版的管理工具和接口库访问嵌入版实时库，支持嵌入版的数据网关接入。

### 7.1 嵌入版硬件平台

GT6657/6658 嵌入式工业计算机，采用先进的高性能 4 核心 A9 处理器，主频高达 1.4GHz。内存采用 32 位高速 DDR3 内存，能提供高速数据处理能力。支持超宽压交直流供电输入。电源输入电路采用独有的专用电路设计，能抵抗 GB/T 17626.5-2008 标准中最高等级 4 级(4KV)8/20uS 的雷击测试。多重电源保护，抗静电、过流、防反接等保护能有效保证野外等恶劣环境下的可靠运行。

集成 2 个 100M/10M 高速自适应网卡，网卡采用双级防雷防静电保护，能抵抗 2KV 雷击。其中 6658 支持 4G 无线网络。

2 路 RS485 电路采用全电气隔离设计，RS485 电路采用三级防雷防静电保护，支持 GB/T 17626.5-2008 标准中 10/700uS 测试的最高等级 4KV 防护。稳定的硬件设计能保证系统长时间正常运行，支持纯硬件定时看门狗，适合无人值守 7X24 小时运行的应用环境。

操作系统为 Linux 系统。



表 8-1 嵌入版实时库硬件平台主要指标:

CPU	ARM V7 架构, 4 核 A9,主频最高为 1.4GHz
内存	512MByte DDR3 高性能内存
存储	MLC eMMC 板载为 8G Byte eMMC
网口	2 个 100M/10M 以太网接口 支持 AUTO MDI/MDIX 双级抗雷防护 支持 GB/T 17626.5-2008 标准中 10/700uS 测试的 3 级 2KV 防护 ±15kV Human Body Model ±15kV IEC1000-4-2 Air Discharge
RS485 接口	4 个全隔离 RS485 接口 (支持收发数据指示灯) RS485 采用三级防护 支持 GB/T 17626.5-2008 标准中 10/700uS 测试的最高等级 4KV 防护 ±15kV Human Body Model ±15kV IEC1000-4-2 Air Discharge
电源接口	支持电源接口 标准 5.08mm 间距 3PIN 欧式端子接口 输入电压: 交流 AC 9~24V 直流 DC 9~36V 电源防护 GB/T 17626.5-2008 标准 4 级(4KV)8/20uS 雷击测试 防反接保护 过压保护 抗脉冲群保护 抗静电: ±15kV Human Body Model ±15kV IEC1000-4-2 Air Discharge
尺寸	124mm×113mm×27mm(L×W×H) 含挂耳
功耗	主板最大功耗≤5W
整机重量	230g
GT6657 环境参数	工作温度: -40~85℃ 储运温度: -40~85℃ 工作相对湿度: 20%~90%无凝露 储运相对湿度: 15%~95%无凝露

## 7.2 规格和售价

因硬件平台可能发生变化, 具体价格请联系我们。



---

## 附录 1：历史数据规模估算

按 1 万标签计算，每个标签每秒产生一个历史数据记录。

一天的记录数： $86400 \times 10000 = 864000000$  记录(8 亿 6 千 4 百万)

TOM 实时库采用特殊存储模式，每条记录平均按 5 字节紧凑格式计算。那么一天的字节数为：

$$(864000000 \times 5) / (1024 \times 1024) = 4119.87\text{MB}$$

也就是 1 万点不压缩每天占用 4G 空间。

所以合理配置每个标签的压缩属性很重要。配置压缩后，平均可以压缩到 20-100 倍(配置的压缩属性不同而不同)。

按平均压缩为 50 倍计算，一天占用 80MB 左右的空间。

实际现场的数据压缩率更大，一直不变的开关量最大可以压缩到 3600 倍。一直在精度范围内波动的数据也可最大压缩到 3600 倍。

注：以上估算没考虑历史数据的无损压缩，TOM 实时库历史存储带无损压缩的，根据数据记录字节冗余度，可压缩率在 2 - 32 倍之间。

---

## 附录 2：趋势压缩精度的配置原则

压缩精度配置直接影响历史数据的规模，因此合理配置压缩精度很重要。CompDev 为压缩精度的限值，该值越小精度越高，压缩率越小，历史数据量越大。反之该值越大，则压缩精度越低，压缩率越大，历史数据量越小。

压缩精度配置原则：

1) 满足数据精度要求

比如要求数据精度为 2%，就没有必要配置为 1%

2) 小于数据采集仪表精度

比如数据采集仪表的精度为 0.5%，配置压缩精度为 0.2%就没有意义。

建议按下述原则配置压缩精度(置百分比精度情况下)

数据使用要求精度 < 配置压缩精度值 < 2 倍采集仪表精度

例子：

使用精度：1%

仪表精度：0.2% (千分之 2 的表)

则配置压缩精度值可取范围：1% ~ 0.4% 之间，配置 0.8% 就比配置 0.4% 压缩率高。

---

## 附录 3：在线体验云实时库

云实时库运行在阿里云上一台 ECS 云虚拟机上,模拟运行了 1000 个标签。点击下面的登录按钮登录成功后进入实时库工况画面。浏览器需要支持 Html5, 推荐 Firefox 和 chrome, IE 可能不支持。

系统配置:

系统: centos 7.5

CPU : 1 核

内存: 1G

带框: 1M bits

WEB 实时工况特点:

采用 svg + javascript + websocket 提供跨平台跨系统的基于 html5 的工控组态软件运行界面,支持 chrome 和 firefox 浏览器,android 手机和平板内置浏览器等内置浏览器。

符合 TLS1.2 版协议的 secret websocket 通道 WSS,支持安全密码套件包括 TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256, TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256, TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA, TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA

实时数据库直接支持 WS/WSS 接口,支持 websocket 的 permessage-deflate(适合 chrome,firefox)和 x-webkit-deflate-frame(适合 ios safari)两种压缩扩展,节约网络资源。

实时数据库直接支持 http/https 的 GET 和 HEAD 命令,无需其他 WEB 发布服务器就能实现基于 WS/WSS 接口和前端 javascript+html5 的现代 WEB 交互式应用。

已测试完全兼容的浏览器:

google chrome for windows,linux

Firefox for windows,linux

IOS 10,11 iphone/ipad safari

Android 安卓手机和平板内置浏览器

实时库服务器端口 443, 地址 kipway.net

在浏览器地址栏中输入体验地址:

<https://kipway.net/indexsvg.html>

---

## 附录 4：第三方应用提交数据建议

一般使用本系统提供的 IO Server 或者 rdbdac\_ux 数据网关从其他系统采集数据写入实时库，使用已有的设备驱动，如：OPCDA，OPCUA，MODBUS 驱动，应用系统集成商也可以按照规范添加驱动，驱动规范和 C++ 的 DEMO 在 rdbdrv sdk 目录中。如果不使用本系统提供网关提交数据，要注意以下规则：

对于 DEC 设备标签，同一个标签的时标必须递增，且间隔大于 0.1 秒才能被实时库接受。

从 RDB1911 开始，增加 PRESET 预置标签，由于 PRESET 标签落地持久化最后一个快照，可以相同时标更改值。因此新纪录时标只要大于等于旧纪录就可被实时库接受。

从 RDB1911 开始，可以向实时库提交 0 时标的记录，实时库自动采用当前实时库服务器的时标作为新记录的时标。适合于不同前置机提交时间相关性的记录，或者前置机无实时钟或者无法校正时间的系统。

一般从其他设备采集的数据，如果本身带有时标，建议使用数据自身的时标，比如 OPCDA 带时标的 2.0 异步模式。对于 MODBUS 等不带时标的数据，一般建议 0 时标提交，使用实时库服务器的时标。

因为实时库是严格过程数据库，时序相关，建议实时库服务器定时校时，保证时钟误差小于 0.1 秒。

---

## 附录 5：编写第三方协议驱动

在 rdbdrv sdk 目录中提供了驱动规范和 C++ 的编写例子。

主要特点：

- 支持多线程，同一个驱动可生成多个驱动实例。
- 数据上行提交采用回调模式，包括实时数据，SOE 事件，日志信息。
- 支持数据的下传(向设备发送控制指令)
- 支持动态加载卸载和启动停止。
- 支持驱动自解析配置文件。

驱动规范采用约定导出接口函数的动态连接库封装，所有导出函数采用 C 方式导出。在 windows 下为标准 DLL 动态库文件，在 Linux 下为 SO 动态连接库文件。开发工具推荐使用 C/C++，其他能生成满足此规范的动态库的开发工具也可(比如 windows 下的 C#)。开发时文档结合提供的 DEMO 驱动 simusrv 理解，调试使用前台版图形版的 rdbdac\_wx 加载调试。最终生产运行建议使用后台版的 rdbdac\_ux for win32/Linux64.

注意，windows 编译为 32 位 x86 指令集，Linux 下编译为 64 位 x64 指令集。

# 附录 6：实时库高可用冗余配置

实时库从 RDB1912 开始，在应用层实时库后台服务端实现高可用双机热备冗余。

## 6.1 高可用原理

本系统采用热备冗余实现高可用，两台服务器(主机 master，从机 slave)同时处于工作状态，中间通过心跳线保持联系和同步数据。如下图 附 6-1.

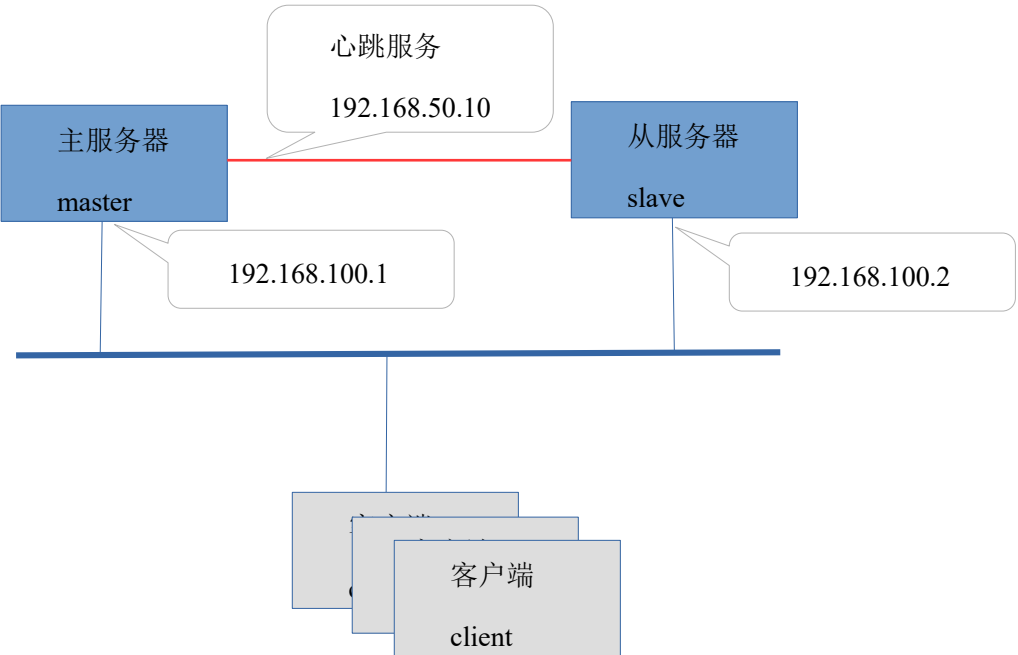


图 附 6-1 服务器冗余

主从服务器是两台完全相同的服务器，拥有双网口(一个数据服务和一个心跳通信)，都具备本地存储，数据存储在各自己的本地硬盘，对外提供相同的数据服务(安装相同的实时库软件)。

主服务器提供心跳服务，心跳服务和数据服务器一般不在同一个网段，不在同一个 VLAN，不在同一个交换机，不使用同一个网口，保证心跳线可靠。

在正常情况下，主备服务器同时工作，客户端可任意连接主或者从服务器获得完全相同的数据提交和读取访问服务，此时提交的数据会在两台服务器落地，读取的数据始终从主服务器读取(如果连接的从服务器，从服务器会将请求服务通过心跳线转发到主服务器)。

任何一台服务器故障宕机，客户端只需重连到另一个服务器即可(如果使用 VRRP 虚拟路由器则对客户终端来讲只有一个 IP，平时 IP 在主机，主机故障后会自动将 IP 漂移到从机，主机恢复后再漂移回主机，linux 可使用 keepalived 软件实现虚拟路由器)。

实时库运行时，每 5 分钟会写一条运行记录到本地系统标签 sysdbman.online 中，当服务器启动时，会自动读取该标签的记录，将维护正常停机或者故障宕机到现在的时间范围写入到同步任务系统标签

sysdbman.datasync 中。后台同步任务线程会自动运行，将离线时间端的数据从心跳线的对端读回来补齐，补齐后会将状态写回 sysdbman.datasync 标签的数据记录，并落地同步成功的日志。

主从服务器之间数据同步和协调完全自动，无需人为干预。状态可以通过两个系统标签来记录。

## 6.2 配置双机热备冗余

配置很简单，在 rdsrv.ini(windows 版)或者 rdbux64.conf(linux 版)中的[database]小结配置如下参数：

```
[database]

maindb = yes    #[yes,no]

heartline = 192.168.1.39:911/heartid # HA heartline
```

其中，主机的 maindb 配置 yes,从机配置 no。

heartline 指定 IP 和端口，中间用:分开，斜杠后面的是一般不超过 16 个字符配对字符串。这个 IP 和端口是主机上的 IP 和端口，从机和主机配置必须完全相同。

如上例：心跳线服务位于主机的 IP 192.168.1.39 上，TCP 端口为 911，配对 ID 为 heartid。所有的 TCP 服务端口建议使用 900-1024 之间的端口，1024 以下的端口被系统保留不会动态分配给连出 socket。

## 6.3 使用虚拟路由器

VRRP (Virtual Router Redundancy Protocol, 虚拟路由冗余协议) 是由 IETF 提出的解决局域网中配置静态网关出现单点失效现象的路由协议。其工作原理是：局域网中多个路由器组成一主多备的冗余结构。同一时间，只有主路由器工作，并定期（默认为 1 秒）广播自己的状态通告（advertisement），当备用路由器在连续三个通告间隔内收不到主路由器的通告或收到优先级为 0 的通告时，会重新进行选举，新选举出的主路由器会使用与之前失效的主路由器相同的 IP 地址与 MAC 地址。虽然在故障切换的过程中会有短时间的网络状况异常，但整个故障切换是透明的。

keepalived 是在 linux 上实现的 VRRP 虚拟路由器，要实现的功能就是对外提供一个固定的虚拟 IP(特别是公网 IP 资源比较宝贵)，映射到内网实时库主从服务器的 IP，主机正常时映射到主机，否则映射到从机，主机恢复时又映射到主机。对客户端来讲是透明，始终只有一个 IP，方便客户端的使用。

实时库的高可用冗余和 VRRP 是没有关系的，也可以不使用 VRRP 路由器，这样客户端就需要配置双 IP，当一个 IP 断开后由客户端自己去切换连接另一个 IP，对于大规模应用特别是用公网 IP 服务的应用来讲不方便的。

## 6.4 数据网关主备双库连接

为配合主备实时库，实现高可用，当在不使用 VRRP 虚拟路由提供高可用对外单一虚拟 IP 时，rdbdac\_ux 数据网关可配置双实时库提交，当主(master)库不通时，自动切换到从(slave)库提交。

---

在 rdbdac\_ux.ini 配置文件中，增加[rdbslave]小节用于指定从(slave)库的连接和登录参数。如下：

[rdbslave] ;备用连接

url = ws://192.168.1.230:921 ;实时库的 IP 地址

user = admin ;连接账号

pass = admin ;账号密码

切换规则：主库连接优先，主库失联时，切换到从库，主库恢复后，切换回主库。切换过程在日志中有输出。

目前仅 rdbdac\_ux for windows86/Linux64 后台服务版带主备连接。rdbdac\_wx 桌面版只用于调试自编写的第三方驱动，无冗余功能。

rdbdac\_ux 从 rdb2020.6 开始也支持的主从双机热备冗余部署。支持整体切换（当其中一个对上连接全部断开或者宕机后，该机上的所有设备会迁移到冗余机器上运行）和单个设备切换(当其中一个设备发生故障会将该设备调度到另一台机器上运行)。具体参见《数据网关 rdbdac\_ux 快速部署指南.pdf》或者本文 3.2 节。



## 附录 7：实时库存储空间配置策略

对于普通应用，带默认压缩的，2 万标签点以下，配置 1TB 存储一般可存储 5-10 年数据。对于大规模不压缩应用，可能几个月就会写满 1TB 磁盘空间。从 2021.10 版开始，实时库配置里增加了存储重用策略，按照配置的空间限值自动删除最旧的历史数据，回收存储空间存储新数据。

在 rdbsrv.ini(linux 版为 rdbux64.conf)配置文件中[database]小节，增加了两个参数：

dbreusesize = 0 ; 实时库重用大小，单位 GB，默认值 0 表示不重用。

tagreusesize = 10 ; 标签重用大小，单位 MB，默认值 10

当实时库增长到 dbreusesize 时才会开始重用。重用是按照标签处理的，当该标签所占空间达到 tagreusesize 时(并且实时库大小达到 dbreusesize 条件成立)才会重用该标签的最旧页面。

实际工程中，一般标签规模是已知的，配置存储时如果预算不是问题，可以从每个标签最多占用空间来计算，然后乘以 1.5 倍作为最终的配置大小。软件包中提供了一个 excel 表格《实时库空间配置.xlsx》用于计算存储配置。可以调整参数计算结果，如果以每个标签 10MB 开始重用计算，常用规模按照下表配置。

tagreusesize (MB)	10	
预留系数	1.5	
标签规模	dbreusesize(GB)	配置空间(GB)
1000	10	15
5000	50	75
10000	100	150
20000	200	300
50000	500	750
100000	1000	1500
200000	2000	3000
500000	5000	7500
1000000	10000	15000

实际执行中，dbreusesize 并不是绝对精确的，而是按照实时库表空间单个文件大小 8GB 对齐的，比如配置 10GB 实际从 16GB 开始重用。

如果磁盘存储固定，可以反向计算每个标签的 tagreusesize 参数值。比如现场配置 1TB 的数据磁盘，标签点数 5000，配置 dbreusesize 的大小为磁盘空间的 2/3 大约 700GB，计算

$$\text{tagreusesize} = \text{dbreusesize} / \text{标签数} = (700 * 1000) \text{MB} / 5000 = 140 \text{MB}$$