Vicdas2.5 系统

使用手册

2022 年 2 月

www.vicdas.com

目录

1. 概述

感谢您使用 Vicdas 系统的相关产品，在安装和使用本产品之前，请认真阅读本用户手册。

1.1 软件功能

Vicdas 系统是工业 SCADA 系统和企业生产信息化系统的解决方案，为工业领域提供生产时序数据存储、查询、监视及统计分析等功能。

Vicdas 系统具有以下功能：

1. VDB 实时数据库
   实时数据存储与查询；实时数据在线计算引擎；实时报警等功能。
2. VDB 时序数据库/历史库
   历史库数据存储与查询，数据聚合查询。
3. VDC 数据采集服务
   V2.0 中提供基于 OPCDA 协议数据采集器，采集工业实时数据，其他协议后续完善。
4. 组态工具
   Vicdas 系统离线配置工具，实现采集器管理；数据点组态；报警组态；趋势组态；历史数据维护等。
5. 图形组态工具
   Vicdas 离线流程图组态工具，实现图形绘制、测点动态特性设定等。
6. 服务管理器
   用于配置及启停 VDB 实时历史数据和 VDC 采集服务的工具。
7. 企业信息 WEB 系统
   提供企业信息系统管理功能，为业务定制开发部分提供了基本信息与开发基础，如提供用户角色管理(RBAC)、数据访问组件；提供在 PC 端与移动端查看生产实时数据流程图、实时/历史趋势、报警信息功能。
8. 数据访问 API
   提供 Restful、SignalR、Socket 三种接口。每类接口可实现读取系统配置数据、实时数据读写、报警信息读取、历史数据读写及聚合计算。

Windows 版具备全部功能，Linux 版实已现部分功能，功能清单如下：

功能	Windows	Linux	备注
VDB 实时数据库	√	√	
VDB 历史数据库	√	√	
VDC 数据采集	√	√	Linux 版不支持 OPC DA 协议
组态工具	√	√	
图形组态工具	√		
服务管理器	√		
企业信息 WEB 系统	√		
数据访问 API	√	√	SignalR 接口版本差异：windows 版为 ASP.NET 版；Linux 版为 ASP.NET Core 版。

1.2 许可协议

数据点在 800 点以内的实时历史数据库允许任何人使用，包括商业行为；提供 800 次将数据写入设备中；数据点超过 800 点或者多次写入设备数据请购买许可。

使用本软件需遵守下列条款：

1. 本软件不允许修改、自由重新发布。

如果您要在产品中使用本软件，希望您能在您的产品中加入相关的书面承认，但并不是必须的。

2. 本软件的运行环境为 Windows OS、SQL Server(或 Mysql)、Kendo UI 等，使用本软件前，使用者需自行购买相关软件（或许可）。

2. Windows 版软件安装

2.1 网络部署

2.1.1 性能最佳网络部署

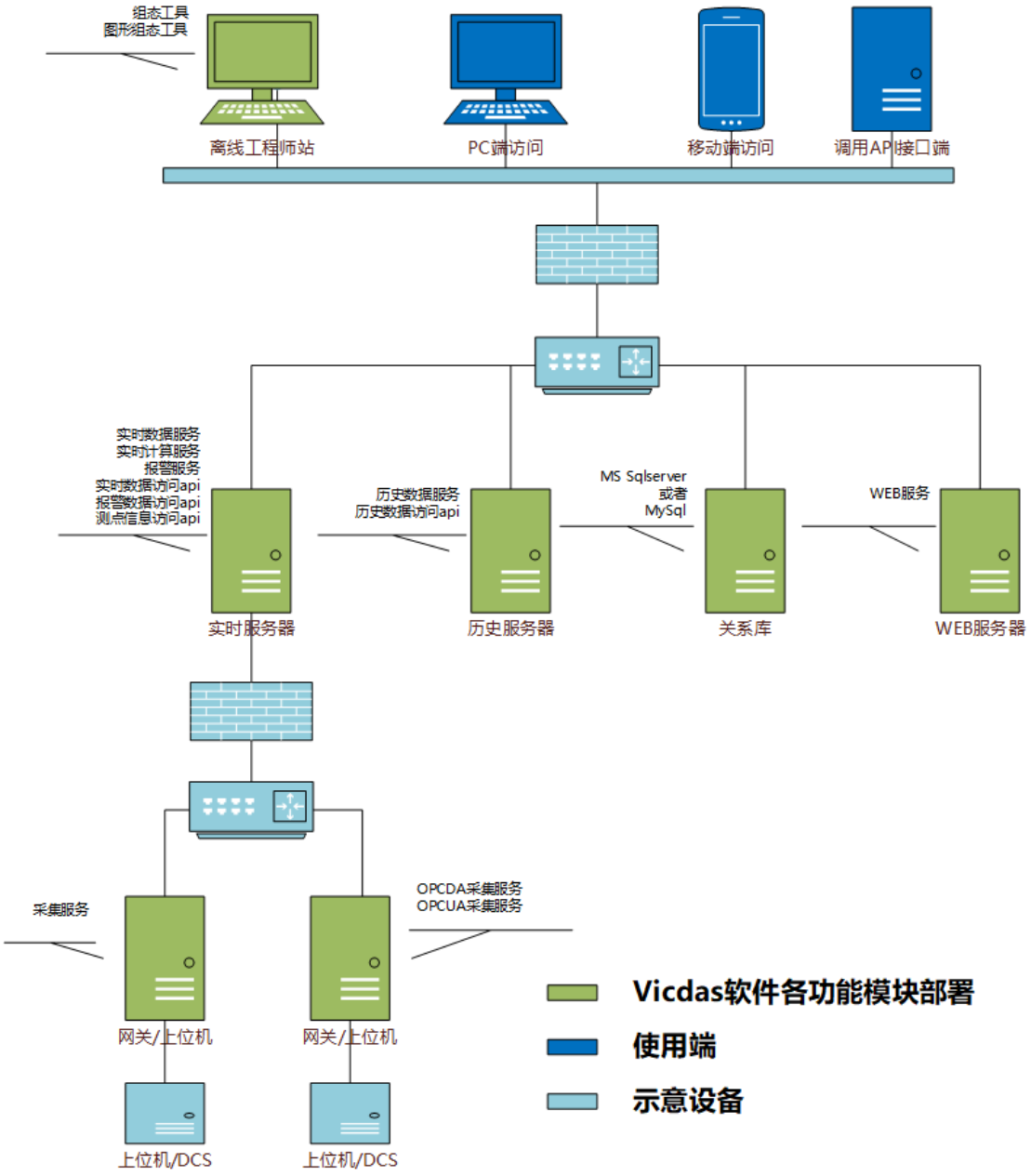


图 2-1 典型网络部署

2.1.2 最小网络部署

实时服务、历史服务、数据采集服务、WEB 服务器、工程师站、关系数据库

部署在同一台计算机中。

2.2 运行环境

2.2.1 数据服务器

硬件推荐配置：

- CPU： i5-3.2GHz
- 内存： 8G(4 万点以下)或 16G(10 万点以下)
- 硬盘： 2T

软件要求：

Windows OS 64 位（推荐 Sever 版）

Framework 4.7.1 及以上

2.2.2 WEB 服务器

硬件推荐配置：

- CPU： i5-3.2GHz
- 内存： 8G
- 硬盘： 512G

软件要求：

Windows OS 64 位（推荐 Sever 版）

IIS7.5 及以上

.NET Framework 4.7.1 及以上

Kendo UI 许可

2.2.3 采集服务器

硬件推荐配置：

- CPU： i5-3.2GHz
- 内存： 4G(32 位系统)/8G(64 位系统)
- 硬盘： 512G

软件要求：

OPC DA:Windows OS 32/64 位（推荐 Sever 版）

OPC UA:Windows OS 64 位（推荐 Sever 版）

.NET Framework 4.7.1 及以上

正确配置 OPC 相关环境，确保与 OPC 可通讯。可使用 OPC Client.exe 测

试与 OPC DA 的通讯。

2.2.4 工程师站

硬件推荐配置：

- CPU： i5-3.2GHz
- 内存： 8G
- 硬盘： 512G

软件要求：

Windows OS 64 位（推荐 Win10）

.NET Framework 4.7.1 及以上

2.2.5 关系库

关系库部署在 WEB 服务器、数据服务器或其他服务器皆可。关系库可选用 MS Sqlserver 或 MY Sql 任一数据库。

MS Sqlserver 要求 Sql2008 R2 及以上

My Sqlserver 要求 5.7 及以上，推荐 8.0。如 Mysql 只能采用其他版，对应的字符集：uft8mb4 和排序规则 utf8mb4_general_ci 改成可使用的字符集及排序规则即可。

2.3 安装方法

2.3.1 自动安装

点击 Vicdas2Setup.exe，启动安装程序，执行界面如下图：

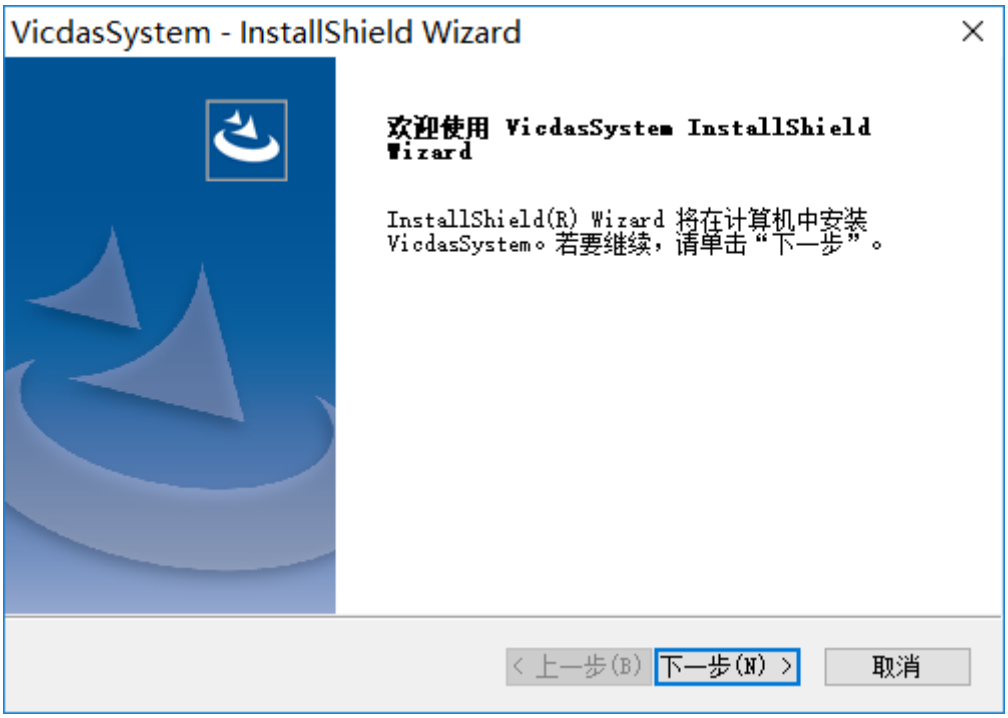


图 2-2 安装界面

逐步执行“下一步”安装。

系统默认安装路径 C:\Vicdas，安装内容为全部安装。如自定义安装，在安装类型界面中选择“定制”安装后，可选择安装路径与安装内容，如下图：

图 2-3 安装类型

2.3.2 手动安装

在自动安装后，具有软件运行程序，当准备重新安装或安装到其他计算机中也可手动安装。（下面所述命令运行在以管理员身份打开的 DOS 窗口中。）

- VDB（实时、历史数据服务）

1. 如系统已安装此服务，请停止服务运行，再执行卸载服务命令：sc delete VicdasVDBSVR；
2. 将已安装的程序目录中 VDB 文件夹复制到目标路径；
3. 执行注册服务命令：sc create VicdasVDBSVR start= auto binpath= 文件路径\VDBService.exe，注意 start= 与 binpath= 后面有空格。

- VDC（数据采集服务）

1. 如系统已安装此服务，请停止服务运行，再执行卸载服务命令：sc delete VicdasDataCollector；
2. 将已安装的程序目录中 VDC 文件夹复制到目标路径；
3. 执行注册服务命令：sc create VicdasDataCollector start= auto binpath= 文件路径\VDCService.exe，注意 start= 与 binpath= 后面有空格

- WEB(企业门户)

1. 程序目录中 WEB 文件夹为门户网站程序，部署在 IIS 中，应用程序池需对应 CLR4.0 集成，PC 端起始页：/Default/Login；移动端起始页：/Default/MLogin。

- CnfgUtility(组态工具)

1. 程序目录中 CnfgUtility 文件夹可直接复制移动，复制到目标路径后，可直接运行 VConfigurationUtility.exe。

- 关系数据库

1. MSSQL/MYSQL 复制、备份、恢复请遵循数据库要求。

2.4 关系库安装

先安装 MYSQL 或者 MSSQL 数据库，配置可通过 IP 访问并启动。运行组态工具--安装目录\CnfgUtility\VConfigurationUtility.exe，在登录界面输入用户名：vadmin，密码：当前日期，如 20180909，点击登录或回车，等待数据库连接测试结果后，弹出界面如下图：

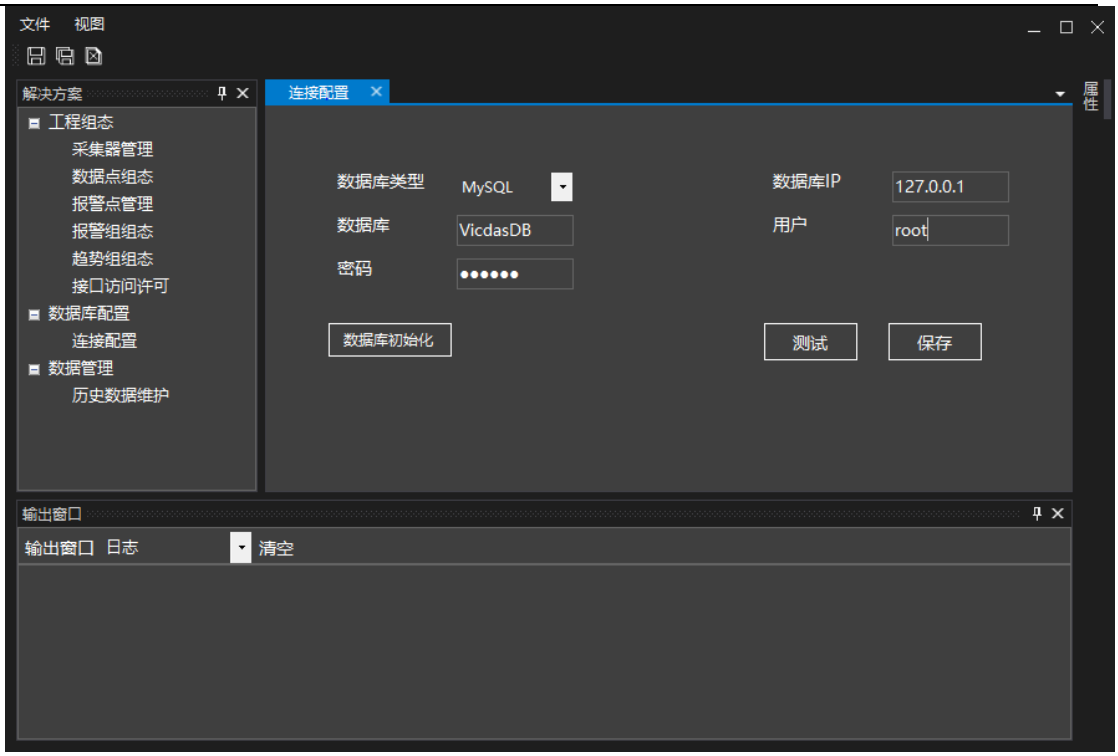


图 2-4 组态数据库连接配置

进入数据库配置-连接配置界面，选择好数据库类型，填写数据库 IP、填写数据库名称，尽量采用 VicdasDB，输入用户名与密码后，点击“数据库初始化”，系统会弹出两次数据库初始化确认。

确认后等待初始化执行，执行完毕后，系统在日志输出窗口中给出执行结果。执行成功后，点击保存及重启。重启成功后，登陆界面未直接打开“连接配置”界面，说明数据库初始化、连接配置成功。如还是需要连接配置，则需确认数据库连接信息是否正确。

2.5 软件卸载

卸载前请停止：VicdasVDBSVR、VicdasOPCDACollector、VicdasMODBUSCollector;

当安装时执行方法为自动安装，再次运行 Vicdas2Setup.exe，开始执行程序卸载。

当安装方式为手动安装，请以手动方式删除。

2.6 软件参数配置

2.6.1 数据服务

安装目录\VDB\VSvrManager.exe，在登录界面输入用户名：vadmin，密码：当前日期，如 20180909，点击登录或回车，弹出服务管理界面，在左侧树形列表中点击“实时服务配置”，界面如下图：

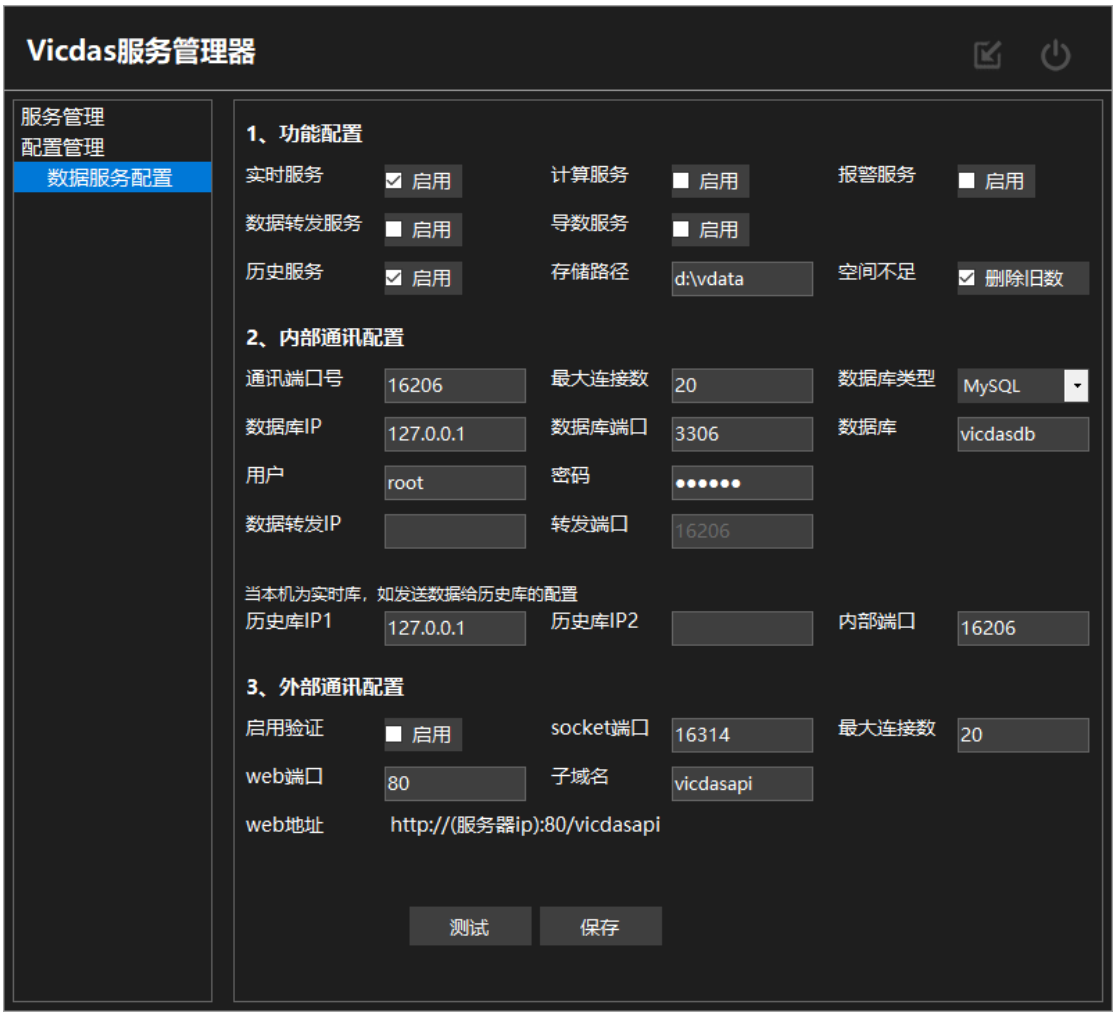


图 2-5 数据服务配置

1. 功能配置

实时服务：是否启用实时数据处理，提供查询接口。

历史服务：是否启用历史存储，提供查询接口。

计算服务：是否启用实时在线计算功能，如启用需提供实时计算脚本。

空间不足是否删除历史数据：当存储空间不足时，是否删去最旧的历史数据，用来存储新的数据。

存储路径：历史数据存放目录。

报警服务：是否启动报警服务。

数据转发服务：将本系统 VDB 中实时数据发送给另一系统 VDB 中，即镜像功能。

说明：实时服务、历史服务至少启用一个服务，也可以同时启动。当一套系统中，将实时服务、历史服务分别部署在两台电脑中，实时服务需配置历史服务的服务器 IP 及内部通讯端口。

2. 内部通讯配置

端口号：内部通讯使用。

最大连接数：内部通讯链接数量最大值。

最大连接数：采集器与服务的链接数量最大值。

数据库类型：选择 MYSql 或 MSSql。

数据库 IP：关系库的 IP 地址

数据库：关系库名称，建议采用 VicdasDB，不要修改。

用户、密码：系统所使用关系库的登录信息。

服务器 IP1、IP2、端口：当 1、功能配置中，本服务器配置为实时服务，则会有一台服务器配置为历史服务，处填写历史服务器 IP 和内容通讯端口。

说明：系统通讯支持双链路，如实现双链路，需要服务器两个以上网卡，在填写 IP 填写两个 IP，。

3. 外部通讯配置

启用验证：如开启此功能，系统对外开放的 Restful、SignalR、Socket 三种接口，在访问时需带有许可号才能访问系统。许可号生成见工程组态中“接口访问许可”章节，将生成的许可号发给客户端。

Socket 端口号、最大连接数：外部 Socket 访问端口号及连接数。

Web 端口及域名：生成 Restful、SignalR 访问地址。域名如需配置到二三级，可在域名中添加“/”，如 data/vicdasapi，Restful、SignalR 的地址都为界面中“web 地址”所示。如 http://192.168.0.2/vicdasapi。如启用 1 功能配置中启用“实时服务”，“web 地址”为实时数据 api 地址；如启用“历史服务”，“web 地址”为历史数据 api 地址；如“实时与历史”服务都启用，实时数据 api 与历史数据 api 都使用“web 地址”这一个地址。

说明：当数据服务与 WEB 服务部署在一台电脑中，此处“web 地址”最好不与 WEB 服务域名相同。

点击测试按钮会提示：服务器与数据库是否连接正确。

点击保存按钮，完成实时历史数据服务器配置。

2.6.2 采集服务

启动安装目录\VDC\VSvrManager.exe，在登录界面输入用户名：vadmin，密码：当前日期，如 20180909，点击登录或回车，弹出服务管理界面，在左侧树形列表中点击“采集服务配置”，界面如下图：

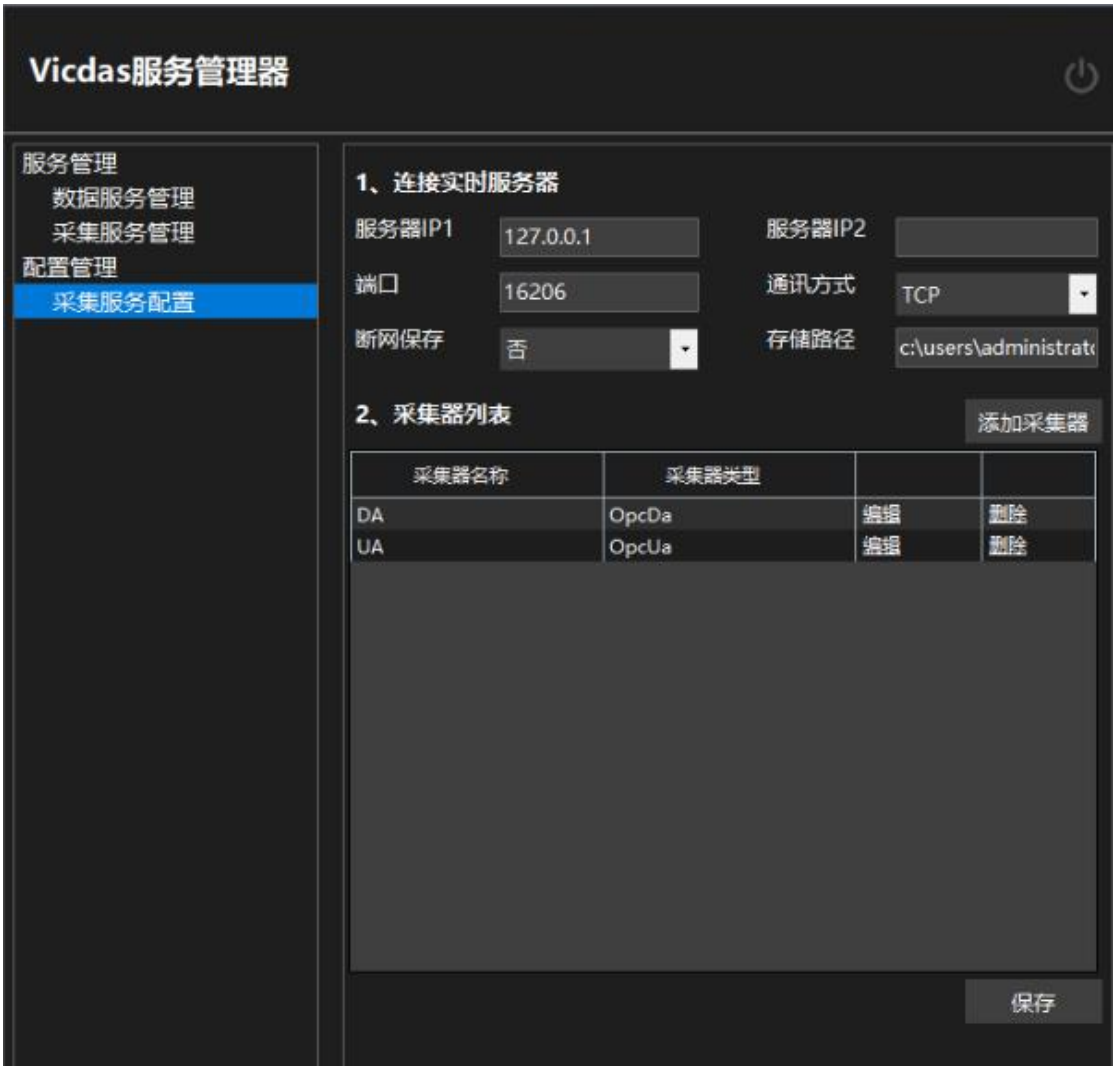


图 2-6 数据采集服务配置

1. 连接实时服务器

服务器 IP、端口：实时数据服务器的 IP 地址，支持双链路，至少填写一组 IP 与端口。填写内容与 2.6.1 中“服务器 IP、端口”的值一致。

断网保存：当 OPC 采集器与实时历史服务器断开连接时，从 OPC 采集的数据是否缓存到本地，缓存数据有效周期为当前日。

存储路径：数据断网保存的路径。

通讯方式：通常选择 TCP 进行通讯。当采集器与 VDB 服务通讯时，通过网闸进行隔离，此时选择 UDP 通讯。当选择 UDP 通讯时，需手动配置测点文件，并将测点文件命名为“points.csv”，并放置 VDC 服务所在目录中，如“安装目录\VDC”，points.csv 文件格式需要：点名、采集器、类型、IO 地址、最大值、最小值、点 ID，公 7 例，如下图所示：

A	B	C	D	E	F	G
点名	采集器	类型	IO地址	最大值	最小值	点ID
Mp1	Modbus	UShort	1;400001	30000	0	728
Mp10	Modbus	UShort	1;400010	30000	0	729
Mp11	Modbus	UInt	1;400011	9.88E+08	0	730
Mp12	Modbus2	Long	1;400015	9.88E+10	0	731

图 2-7 points.csv 格式

生成“points.csv”方式有两种方法：

方法 1：

从 4.5“数据点组态”界面中，右键，导出数据，获得测点文件，对测点排序，删除不需测点，删除不需要列，将文件另存为“points.csv”即可。

方法 2：

先选择 TCP 通讯，此时系统会自动生成“points.csv”文件，然后将 TCP 修改为 UDP 即可。注意，如目录中已存在“points.csv”将不会执行更新生成功能，如测点发生变化，需手动删除“points.csv”文件，设置 TCP 通讯，重新生成。

2. 采集器列表

用来添加剂管理采集器，系统提供 OPC DA、OPC UA、ModbusTCP 采集器。

- OPC DA

图 2-8 opc da 采集配置

采集器名称：采集器对应组态工具中配置的采集器集合中一个采集器，此处配置对应的采集器名称。

OPCIP：提供 OPC DA 通讯的服务器 IP。

OPC 服务：OPCDA 服务器提供的 OPC 服务名称。

采集周期：从 OPC 采集数据的周期。

数据访问：访问 OPC 数据的方式，订阅、异步、同步，建议采用异步或订阅。

采集死区：OPC 数据采集死区，单位百分比。

数据超量程：OPC DA 采集的数据超过组态中最大值、最小值的处理方式。

点击确定按钮，完成 OPC DA 采集器配置。

注意：当打开 OPC DA 配置界面，出现下图红线框中内容，先关闭窗口，然后手动对 安装目录\VDC\OPCDAAuto.dll 进行注册。

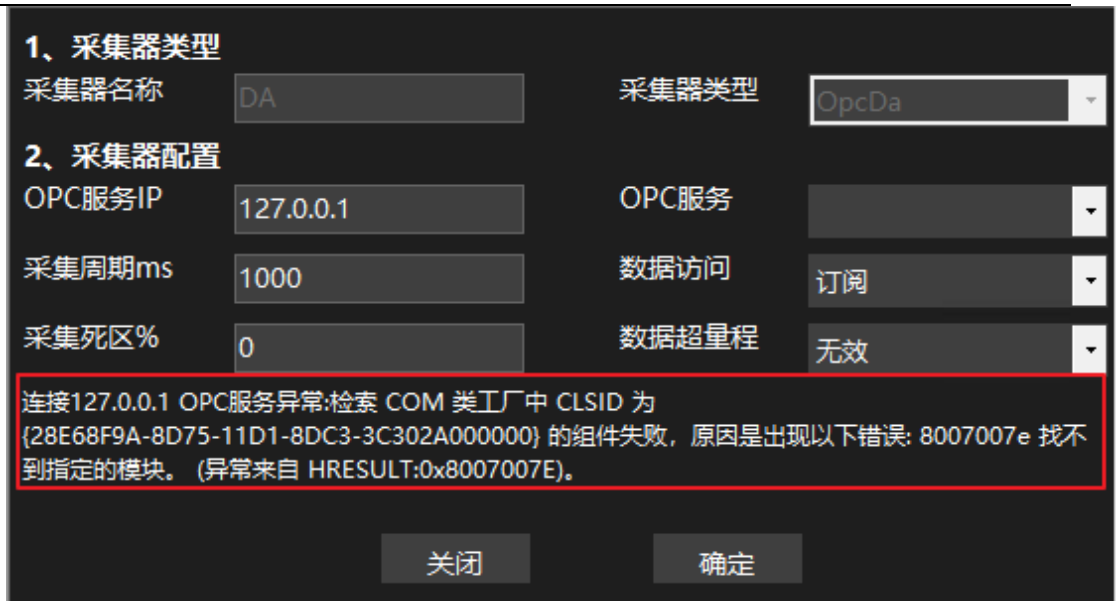


图 2-9 opc da 异常

a) 以管理员身份运行 dos 窗口，

b) 输入 regsvr32 安装目录\VDC\OPCDAAuto.dll

c) 回车

如注册提示出现成功，再次打开 OPC DA 采集器配置界面是，不会出现红框内容。

➢ OPC UA

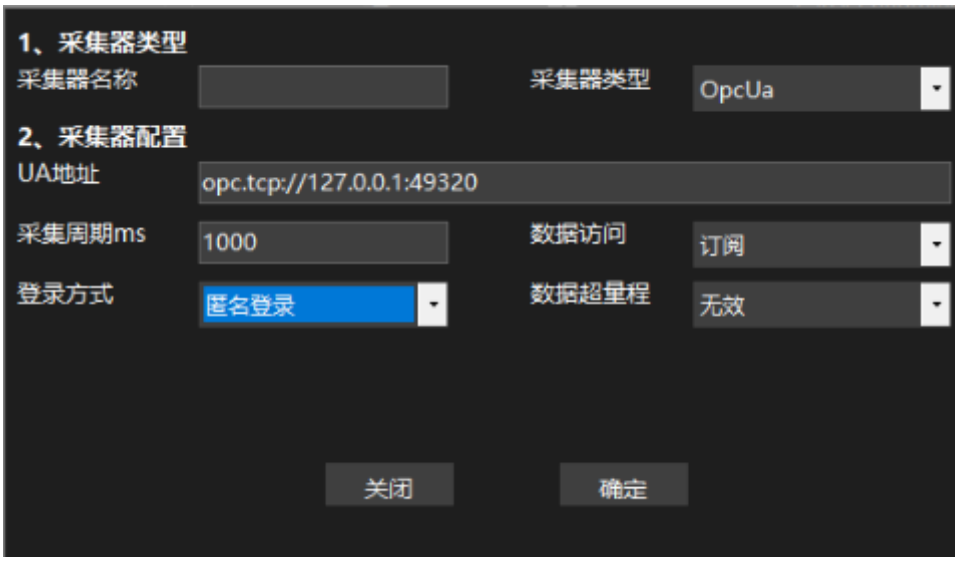


图 2-10 opc ua 采集配置

采集器名称：采集器对应组态工具中配置的采集器集合中一个采集器，此处配置对应的采集器名称。

UA 地址：OPC UA 访问地址。

采集周期：从 OPC UA 采集数据的周期。

OPC 服务：OPCDA 服务器提供的 OPC 服务名称。

数据访问：访问 OPC 数据的方式，订阅、异步、同步，建议采用异步或订阅。

数据超量程：OPC UA 采集的数据超过组态中最大值、最小值的处理方式。

登陆方式：OPC UA 的登陆方式有匿名登录、用户名登录及证书登陆。当选择用户名登录或证书登陆，请根据显示内容填写相关信息。建议先选择匿名登录、OPCUA 服务端也配置匿名访问，判断数据能否采集，采集成功后，如需修改登陆方式再修改。

点击确定按钮，完成 OPC UA 采集器配置。

- ModbusTCP

图 2-11 ModbusTCP 采集配置

采集器名称：采集器对应组态工具中配置的采集器集合中一个采集器，此处配置对应的采集器名称。

MODBUS IP1，MODBUS IP2：MODBUS 的 IP 地址。

MODBUS 端口号：MODBUS 端口号，默认值 502。

采集周期：从 MODBUS 采集数据的周期。

数据访问：访问 MODBUS 数据的方式：异步、同步，点数量少时采用异步，点多采用异步。

数据超量程：MODBUS 采集的数据超过组态中最大值、最小值的处理方式。

格式转换：从 MODBUS 获取字节数组转换为所需格式。

字符串反转：字节数组转成字符串后，字符串反向排序。

点击确定按钮，完成 ModbusTCP 采集器配置。

➢ ModbusRTU

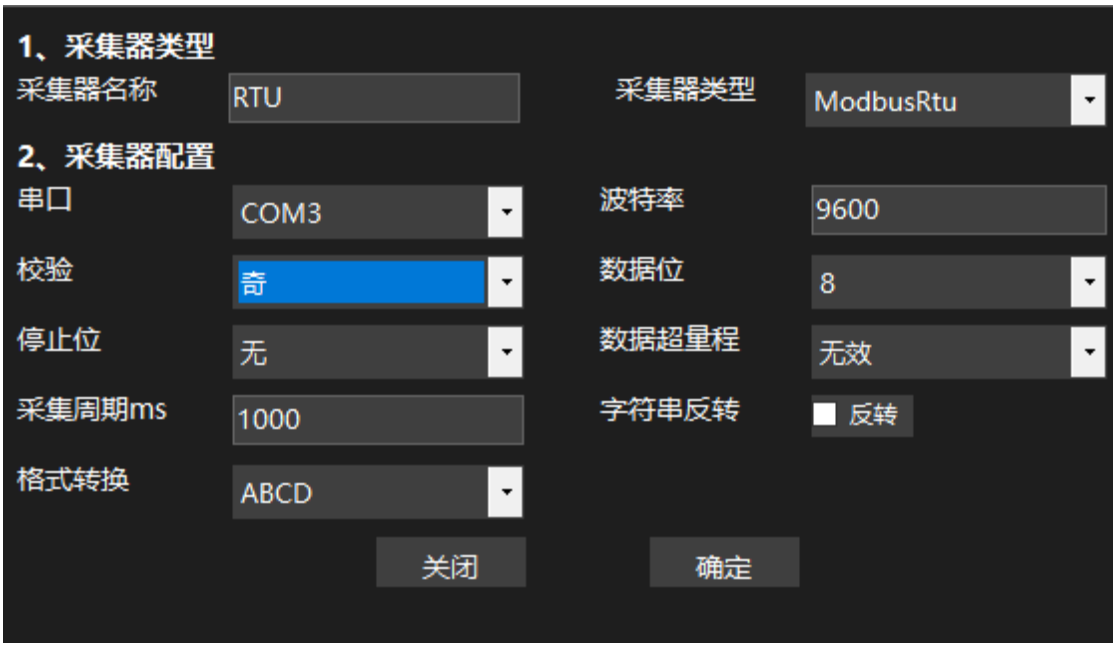


图 2-12 ModbusRTU 采集配置

采集器名称：采集器对应组态工具中配置的采集器集合中一个采集器，此处配置对应的采集器名称。

串口：MODBUS 的串口号。

波特率：通讯中传输速率。

验位：检错方式。

数据位：通讯中实际数据位。

停止位：单个包的最后一位。

采集周期：从 MODBUS 采集数据的周期。

数据超量程：MODBUS 采集的数据超过组态中最大值、最小值的处理方式。

格式转换：从 MODBUS 获取字节数组转换为所需格式。

字符串反转：字节数组转成字符串后，字符串反向排序。

点击确定按钮，完成 ModbusRTU 采集器配置。

2.6.3 WEB 系统

因为企业门户还会增加企业自身业务功能，故此处不提供配置程序，需手动修改文件。

打开安装目录\WEB\Web.config 文件，修改文件中节点：

DefaultPWD 节点的 Value 值：系统创建用户及重置用户的系统登录密码，节点位置 Configuration/appSettings/add[key="DefaultPWD"]。

VDBAnonymous 节点的 Value 值:系统是否允许匿名访问(不需登陆系统),时、历史数据的趋势图、报警、流程图及自定义测点趋势查询界面,地址可在组态工具中查询。true 为允许,false 不允许。节点位置 Configuration/appSettings/add[key="VDBAnonymous"]。

WebCommRT 节点的 Value 值:实时数据服务提供的 WEB 地址,配置内容为 2.6.1 中“WEB 地址”的值,节点位置 Configuration/ appSettings/ add[key="WebCommRT"]。

WebCommHIS 节点的 Value 值:历史服务提供的 WEB 地址,配置内容为 2.6.1 中“WEB 地址”的值,节点位置 Configuration/ appSettings/ add[key="WebCommHIS"]。

ESwordConn 节点的 connectionString 值:系统的关系库的链接配置字符串,内容填写“Data Source=数据库 IP;Initial Catalog=VicdasDB;User Id=数据库用户;Password=数据库用户密码;” ;providerName 根据已安装关系库类型填写“System.Data.SqlClient”或“MySql.Data.MySqlClient”。

示例:

```
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="VDBAnonymous" value="true" />
  <add key="DefaultPWD" value="1" />
  <!--Web通讯-->
  <add key="WebCommRT" value="http://127.0.0.1/vicdasapi" />
  <add key="WebCommHIS" value="http://127.0.0.1/vicdasapi" />
</appSettings>
<connectionStrings>
  <add name="ESwordConn" connectionString="Data Source=.;Initial Catalog=VicdasDB;User ID=sa;Password=123456;" providerName="System.Data.SqlClient" />
  <!--<add name="ESwordConn" connectionString="Data Source=127.0.0.1;Initial Catalog=VicdasDB;User Id=root;Password=123456;" providerName="MySql.Data.MySqlClient" />-->
</connectionStrings>
```

2.6.4 组态工具

安装目录\CnfgUtility\VConfigurationUtility.exe,在登录界面输入用户名:vadmin,密码:当前日期,如 20180909,点击登录或回车,弹出界面如下图:

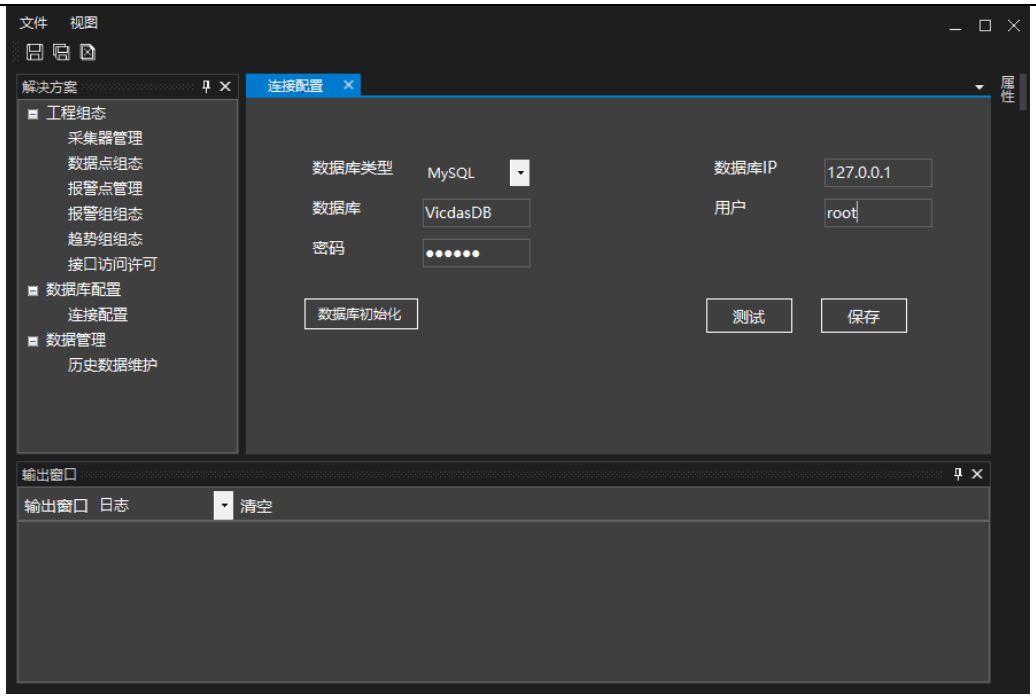


图 2-13 组态工具

数据库初始化已在 2.2.5 节完成，如果修改 IP 或其他登陆关系库信息，在界面中录入数据库连接信息。

数据库 IP：关系库的 IP 地址

数据库：关系库名称

用户、密码：系统所使用关系库的登录信息。

点击测试按钮会提示：服务器是否连接正确。

点击保存：保存数据库配置信息。

图形组态工具，数据库配置如上。

2.7 软件运行

系统运行参数配置完毕后进行工程组态，然后依次启动实时历史数据服务、OPC 数据采集服务，系统开始运行。

启动服务方式有两种：

1、 运行“安装目录\VDB\VSvrManager.exe”和“安装目录\OPCDA\VSvrManager.exe”，在服务管理中启动“实时服务”和“OPCDA 采集服务”。

2、 打开 Windows 系统的“服务”界面，启动“VicdasVDBSVR”、“VicdasOPCDACollector”服务。

2.8 注意事项

2.8.1 OPCDA

当使用 OPC 远程访问方式，需要对操作系统进行配置（包括 DCOM、本地策略和防火墙），请按照相关技术要求逐一配置。

DA 采集器所在服务器中需安装 OpcEnum 服务并运行，如未安装，请到 OPC 基金会官网或百度下载 OpcEnum 服务安装方式

DA 采集器所在服务器一定要注册 OPCDAAuto.dll，用安装包安装时会自动注册，但因 DCOM 原因，如出现 OPC 检测异常等现象："**检索 COM 类工厂中 CLSID 为 {28E68F9A-8D75-11D1-8DC3-3C302A000000} 的组件时失败**"，请按照 2.6.2 中"**注意**"说明进行 OPCDAAuto.dll 重新注册。

2.8.2 IIS

在安装 IIS 时，建议选择 IIS 所有功能，**不能采用默认选择**，否则 WEB 页面不能正常展示，下图为 IIS 选择的示例：

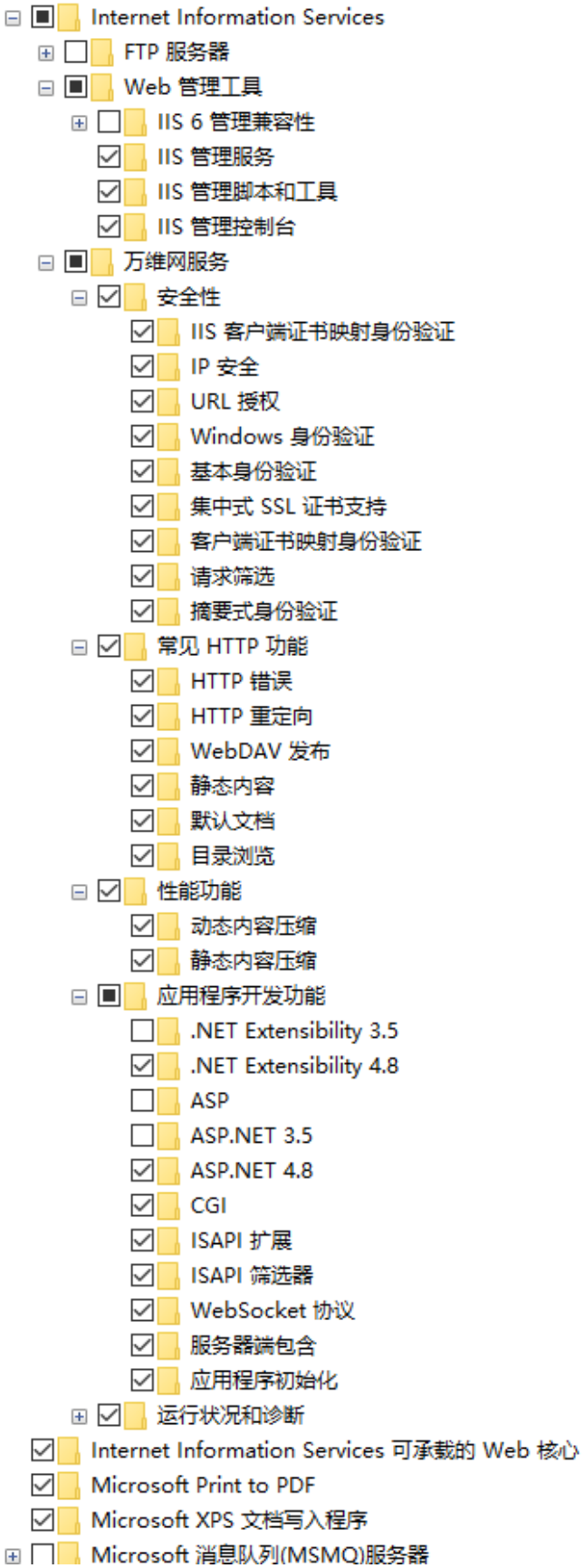


图 2-14 IIS 功能选择示例

2.8.3 数据点组态

数据点组态执行保存，如组态工具提示异常“File XXX not found(**OS error 12 – Permission denied**)”,解决办法如下：

关闭组态工具，删除组态工具的快捷方式，找到组态工具的执行文件，如“安装目录\CnfgUtility\VConfigurationUtility.exe”，双击打开组态工具即可。或者在执行文件上右键，发送到桌面快捷方式，重新生成快捷方式。

3. Linux 版软件安装

3.1 网络部署

网络部署同 2.1

3.2 运行环境

数据服务(VDB)、采集服务(VDC)部署在 Linux 系统中，组态工具部署在 windows 系统中。

产品支持 Linux 的版本与.Net6 保持一致。参见微软网址：https://docs.microsoft.com/zh-cn/dotnet/core/install

3.2.1 数据服务器

硬件推荐配置：

- CPU： i5-3.2GHz
- 内存： 8G(4 万点以下)或 16G(10 万点以下)
- 硬盘： 2T

软件要求：

Ubuntu 20.04

dotnet 6.0 运行时

3.2.2 采集服务器

硬件推荐配置：

- CPU： i5-3.2GHz
- 内存： 4G(32 位系统)/8G(64 位系统)
- 硬盘： 512G

软件要求：

Ubuntu 20.04

dotnet 6.0 运行时

OPC DA 不支持 Linux 系统，如需采集 OPC DA，请安装 windows 版。

3.2.3 工程师站

硬件推荐配置：

- CPU： i5-3.2GHz

- 内存： 8G
- 硬盘： 512G

软件要求：

Windows OS 64 位（推荐 Win10）

.NET Framework 4.7.1 及以上

3.2.4 关系库

关系库部署在 VDB 数据服务器或者其他服务器皆可。关系库可选用 MS Sqlserver 或 MY Sql 任一数据库。

MS Sqlserver 要求 Sql2008 R2 及以上，2014 及前版本需安装 KB3135244 ，是 Sqlserver 支持 TLS 1.2，推挤使用 sql2016 及以上或 sql2017forlinux 版 My Sqlserver 要求 5.7 及以上，推荐 8.0。如 Mysql 只能采用其他版，对应的字符集：uft8mb4 和排序规则 utf8mb4_general_ci 改成可使用的字符集及排序规则即可。

3.3 安装与运行

3.3.1 Linux 运行环境

在 Linux 上安装 dotnet 6.0 运行时，安装方式参考微软网站：https://docs.microsoft.com/zh-cn/dotnet/core/install/ ，安装完毕后，运行 dotnet --list-runtimes，显示 aspnetcore6.0 和 netcore6.0 为安装完成，如下图：

```
~/桌面$ dotnet --list-runtimes
Microsoft.AspNetCore.App 6.0.2 [/usr/share/dotnet/shared/Microsoft.AspNetCore.App]
Microsoft.NETCore.App 6.0.2 [/usr/share/dotnet/shared/Microsoft.NETCore.App]
```

图 3-1dotnet6 运行时环境

如采用离线安装形式，安装完毕.net6 后，**需在/usr/bin/目录中建立软连接文件 dotnet，连接离线安装的 dotnet 执行文件**，完成可运行，

/usr/bin/dotnet --list-runtimes

下图为自动安装生成的/usr/bin/dotnet 文件信息

```
[root@localhost vdc]# ls -l /usr/bin/dotnet
lrwxrwxrwx. 1 root root 24 2月  26 09:23 /usr/bin/dotnet -> /usr/share/dotnet/dotnet
[root@localhost vdc]# ls -l /usr/share/dotnet/dotnet
-rwxr-xr-x. 1 root root 141760 1月  15 04:18 /usr/share/dotnet/dotnet
```

图 3-2/usr/bin/dotnet 文件

3.3.2 组态工具

下载并解压 vicdas-linux.zip，得到文件内容如下图：

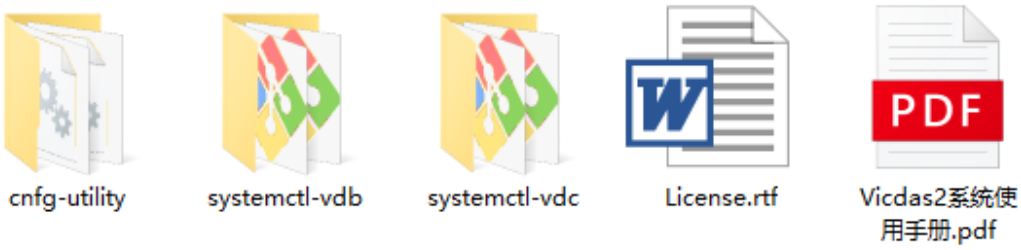


图 3-3 安装包清单

cnfg-utility：组态工具、配置 VDB 数据服务和 VDC 采集服务启动的工具。复制到 windows 系统上(系统需安装.NET Framework 4.7.1 及以上)，进入文件夹，双击 exe 程序可直接运行。

VConfigurationUtility.exe：组态工具，使用方案见 4

VCnfgJsonTool.exe：配置 VDB、VDC 启动工具，配置完成点击保存后，在目录中生成 JsonToolFile/vdb 和 JsonToolFile/vdc 文件夹，分别生成 VDB 服务和 VDC 服务启动时所需的配置文件 appsettings.json

3.3.3 VDB 数据服务

systemctl-vdb：VDB 数据服务程序。将 systemctl-vdb 文件夹复制到 Linux 中。

以下操作以管理员权限运行。

进入到 systemctl-vdb 目录，可看到 vicdas-vdb-install.sh 和 vicdas-vdb-uninstall.sh 文件，在当前目录下运行 sudo sh ./vicdas-vdb-install.sh 命令，完成“**vicdas-vdb**”服务配置及设置开机启动。

运行 VCnfgJsonTool.exe(其功能参见 2.6.1)，将生成 VDB 的 appsettings.json 放置 systemctl-vdb/vdb 目录下。如有计算脚本，将脚本放入 systemctl-vdb/vdb/CalcScript 下，参见 6

运行 sudo systemctl start vicdas-vdb 命令，完成启动。

3.3.4 VDC 采集服务

systemctl-vdc：VDC 采集服务程序。将 systemctl-vdc 文件夹复制到 Linux 中。

以下操作以管理员权限运行。

进入到 systemctl-vdc 目录，可看到 vicdas-vdc-install.sh 和 vicdas-vdc-uninstall.sh 文件，在当前目录下运行 sudo sh ./vicdas-vdc-install.sh 命令，完成“**vicdas-vdc**”服务配置及设置开机启动。

运行 VCnfgJsonTool.exe(其功能参见 2.6.1)，将生成 VDC 的 appsettings.json 放置 systemctl-vdb/vdc 目录下。

运行 sudo systemctl start vicdas-vdc 命令，完成启动。

3.3.5 其他

License.rtf：软件使用许可协议

Vicdas2 系统使用手册.pfd：使用手册

4. 工程组态

在软件完成安装与参数配置后，进行工程组态，组态完毕后，即可启动实时/历史数据服务、数据采集服务，系统开始运行。

组态是用户通过工具对系统进行配置、设定，构建出系统的数据采集与监视功能。Vicdas 系统使用 VConfigurationUtility.exe 作为组态工具，完成系统数据采集、存储、计算、导出等功能的配置。

启动“安装目录\CnfgUtility\VConfigurationUtility.exe”，弹出登录界面，如下图：

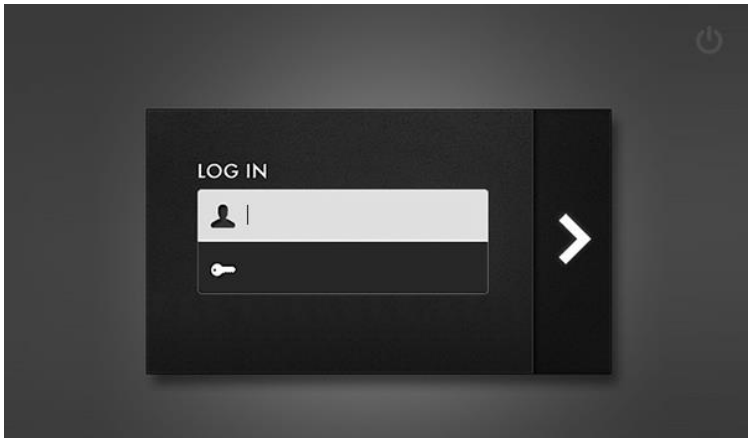


图 4-1 登录界面

在登录界面输入用户名：vadmin，密码：当前日期，如 20180909，点击登录或回车，弹出组态界面。

4.1 操作界面介绍

4.1.1 界面划分

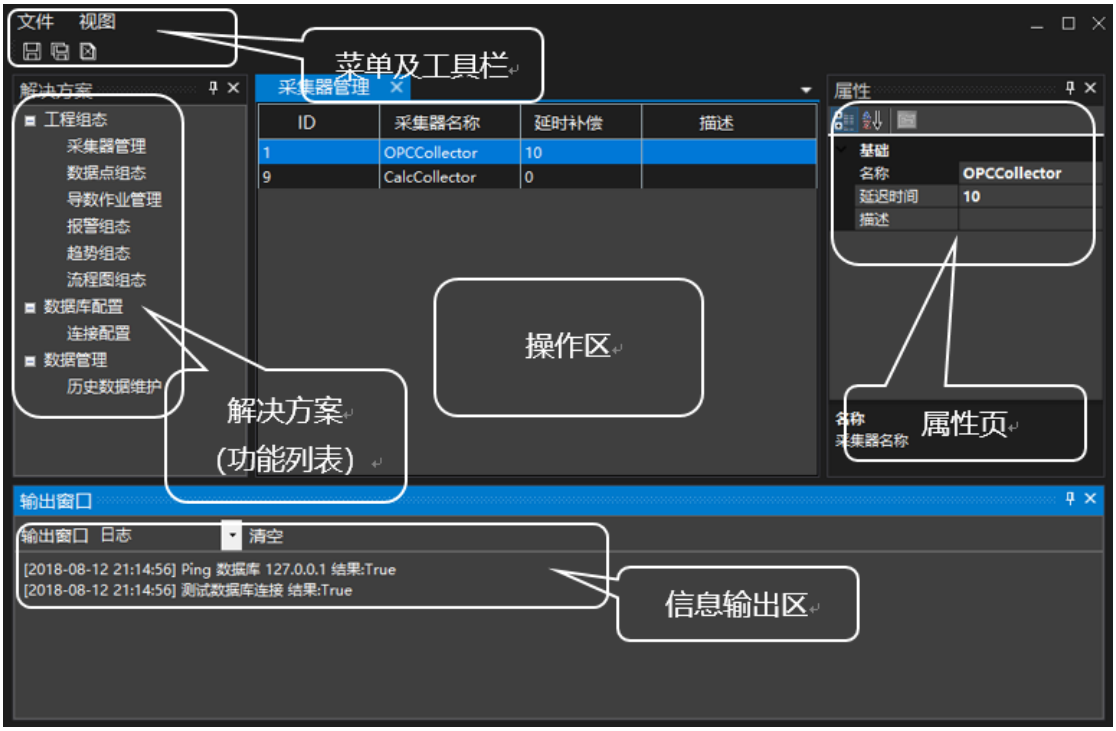


图 4-2 操作界面

图 4-2 操作界面是组态软件整体操作界面。在默认的布局情况下，可以划分为五个功能视图区域：

1. 菜单及工具栏：位于顶部，包括：标题栏、主菜单和工具栏等子功能区。
2. 解决方案视图：位于中间左侧，用于显示功能列表。
3. 操作区：位于中央，主要的组态界面。
4. 属性页：位于中间右侧，属性页，编辑各种属性值。
5. 信息输出视图：位于底部，信息输出视图区。

以上区域的位置是默认情况下的布局，在使用时，用户可以根据自己的风格自行布局调整。

4.1.2 基本操作

菜单及工具栏：提供对操作区的多个 Tab 窗口的保存、全部保存及关闭功能。

解决方案：双击解决方案的某一树形节点，在操作区打开该功能的窗口。

操作区：在操作区的各个窗口中，右键单击，可弹出相关操作菜单，如增加、删除等。

属性页：在操作区的窗体中单击某一数据行，其属性信息在属性页中显示，

并可在属性页中对其属性内容进行设置。

信息输出视图：显示在操作区、属性页中的执行操作的结果信息。输出信息分为错误和日志两类。

4.2 组态流程

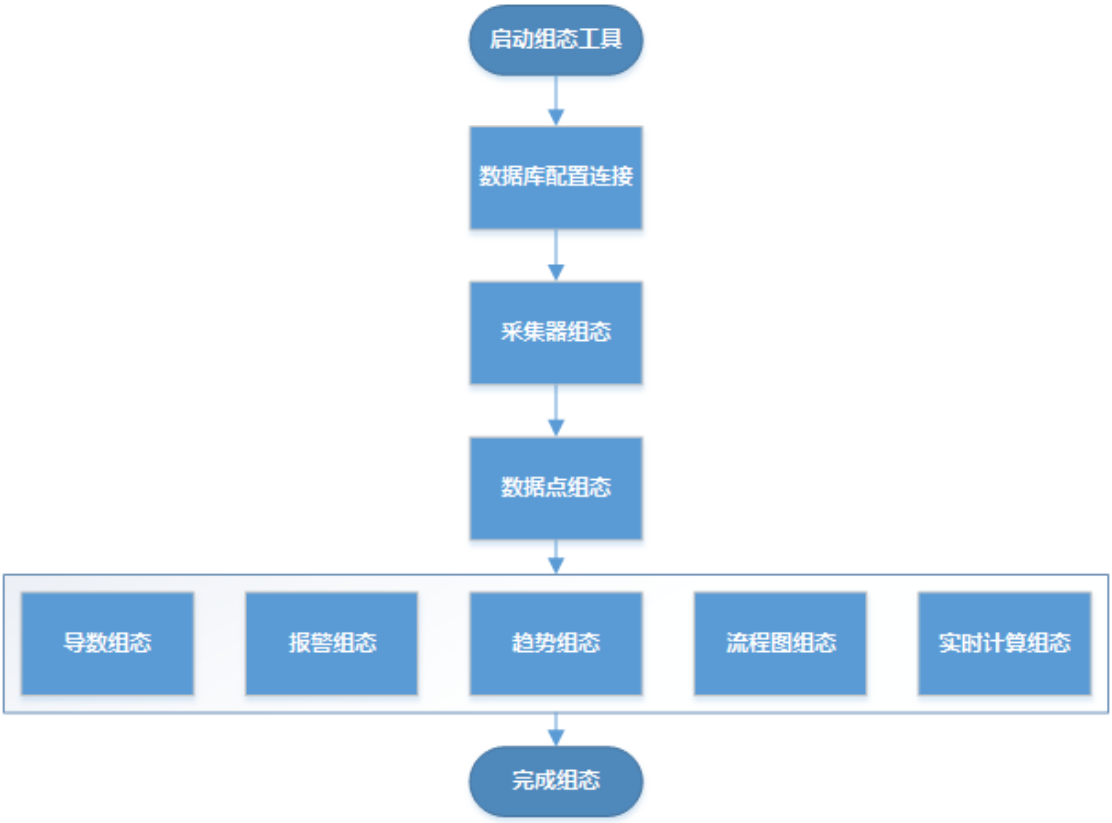


图 4-3 组态流程

图 4-3 组态流程所示是组态工作时的主要流程。请按照流程顺序进行组态。中间矩形中并列的 5 项组态内容是可选组态内容。每个组态内容的具体操作请参考相关章节。

4.3 数据库连接配置

请参见 2.6.4

4.4 采集器组态

采集器：包含两层含义，其一表示采集数据的程序，比如针对 OPC 协议所作的 OPC 采集器;其二表示点的物理集合,实时库中点一定从属于某个采集器。采集器包含的属性有采集器名称、延时补偿、描述等。

双击解决方案中的“采集器管理”节点，弹出采集器管理窗口，如下图：

图 4-4 采集器管理

在窗口中右键弹出菜单，如下图：

图 4-5 采集器菜单

新建：创建新的采集器，并显示在窗口中。

- 名称：采集器名称。
- 延迟时间：用于补充从采集器获取数据到实时库获取数据的网络传输时间，减小时间戳的误差。
- 描述：采集器的描述信息。

保存：将采集器编辑的内容进行保存。

删除：删除所选择的采集器。

全部删除：删除所有采集器。

点击窗口的一个采集器，在属性页中显示该采集器的信息，在属性页编辑采集器信息。

4.5 数据点组态

数据点又称测点，简称点，表示控制系统中某个参数，例如表示锅炉的压力。概念上点的数据集合构成了实时历史数据库。

4.5.1 数据点属性

- 点名：数据点名称；
- 类型：数据点的数据类型，类型分为：Int(4字节)、UInt(4字节) 、Short(2字节) 、 UShort(2字节) 、 Long(8字节) 、 ULong(8字

节）、 Float(4字节) 、 Double(8字节) 、Bool(1字节)、 Byte(1字节) 、 SByte(1字节) 、 Char(2字节) 、Bytes(64字节) 、 String(64字节) 、Date(8字节)；

- 采集器：数据点所属采集器的名称；
- IO地址：数据点所在某协议采集器对应点的IO地址；
- 执行周期：设定点的存储周期、报警周期等，单位毫秒；
- 存储方式：数据点存储入历史库方式；
- 存储死区：值的变化百分比小于该值不存储当前值；
- 最大/小值：数据的量程上下限，超过时，数据无效；
- 点描述：数据点说明信息。
- 布尔、字符、字符串、数组、时间：存储方式不能选择压缩存储；

4.5.2 数据点配置

数据点组态界面如下图：

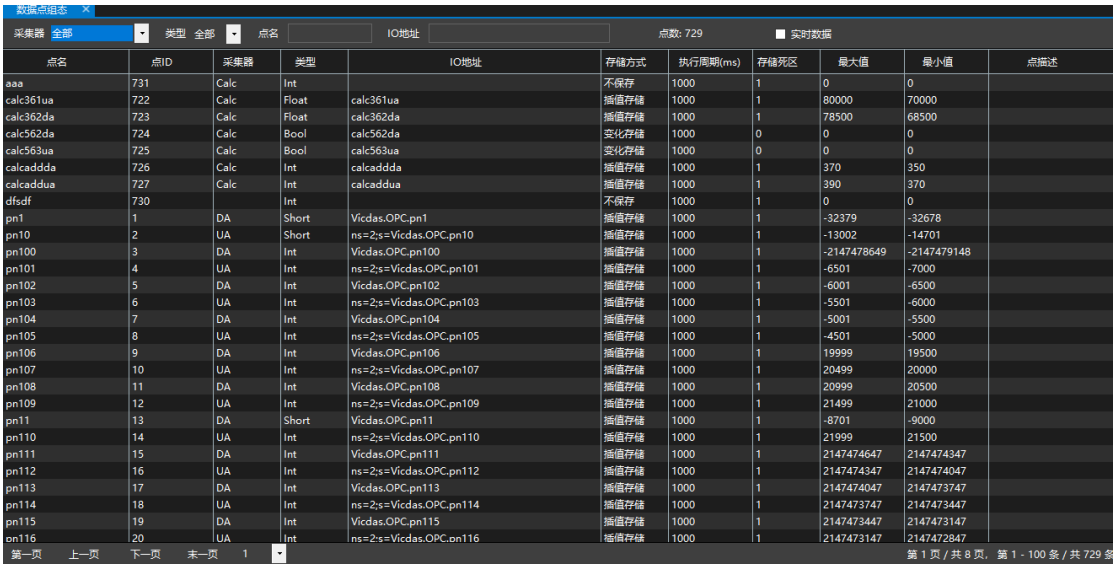


图 4-6 数据点管理

通过窗口头部的采集器、类型、点名的筛选，窗体中显示符合条件的点集合。在列表选中一个点，该点属性显示在属性页中，对点进行编辑。

在窗口中右键弹出菜单，如下图：

图 4-7 数据点菜单

新建：添加一个数据点；

保存：保存所有数据点信息；

删除：删除当前所选数据点；

列表删除：删除点列表中所有点；

全部删除：删除系统中所有的点；

数据导出：将当前所有点导出到 CSV 文件中；

数据导入：将 CSV 文件中的点导入到系统中。

提示：批量添加点时，采用先用 CSV 文件编写点信息，然后再导入数据库中，如不知某一属性如何填写时，可在窗口中新建一个点，然后执行数据导出，然后参考导出文件编写点信息。点类型不同编辑属性有所差别，CSV 文件中不需填写全部属性，属性编写参考 4.5.1 中“数据点根据类型不同，属性变化说明”部分。

CSV 文件采用“,”分割。

4.5.3 数据点 IO 上行地址

IO 上行地址是获取设备数据的对外输出的地址，采集服务根据数据点的 IO 上行地址获取数据。

- OPC DA/UA

在 OPC DA 和 UA 中，IO 值为 Item 的地址

- MODBUS TCP/RTU

在 MODBUS 点 IO 地址配置格式用“;”进行分割，寄存器地址从 1 开始，说明如下：

非字符串类型：站号;寄存器地址

字符串类型：站号;寄存器地址;字符串字节长度

示例：IO 地址： 1;400341 站号 1，寄存器地址 400341

数据采集部分，实现功能码为：0x01、0x02、0x3、0x04

地址	范围
输出线圈[功能码(十进制)：01]	000001-065536
输入线圈[功能码(十进制)：02]	100001-165536
内部寄存器[功能码(十进制)：04]	300001-365536
保持寄存器[功能码(十进制)：03]	400001-465536

4.5.4 数据点 IO 下行地址

IO 下行地址是写入设备数据的输入的地址，用于实现对设备控制，采集服务根据数据点的 IO 下行地址将数据发送给设备。

下行数据目前不支持数据类型：字符串、时间、数组

- OPC DA/UA

在 OPC DA 和 UA 中，IO 值为 Item 的地址

- MODBUS TCP/RTU

在 MODBUS 点 IO 地址配置格式用“;”进行分割，寄存器地址范围为 1-9999，说明如下：

地址：站号;寄存器地,

示例：IO 地址： 1; 341 站号 1，寄存器地址 341

4.5.5 数据点实时数据

在数据点组态界面，勾选“实时数据”后，在数据点列表中出现数据点的实时值和值状态两列，用来显示实时数据。

显示实时数据的前提，需配置与 VDB 实时服务的连接，在左侧功能列表中中选择“实时服务连接”，打开如下界面：

图 4-8 实时服务连接

WEB IP 即 VDB 实时服务所在服务器的 IP，端口及子域名参考 2.6.1， 点击测试及保存按钮，完成配置。

4.6 实时导数作业

系统定期的将测点实时、测点统计数据导入到关系库中，为企业生产经营分析报表提供生产数据。

4.6.1 导数原理

生产报表一般用于记录某个测点在某一时刻的数值，格式比较简单。如下面的例子：

时间	温度	压力	电压
12:00	20	300	380
13:00	22	310	375

其中的温度、压力、电压都对应系统中的数据点；时间列则是指采样的时刻。

展示的生产报表可能对应着下面的数据库表：

字段名	类型	描述
Time	Datatime	采样时刻
Temp	Float	实时温度
Pressure	Float	实时压力
Voltage	Float	实时电压

可以使用 sql 语句（假设数据库表名为 RT_Report）：

Insert RT_Report (time,temp,pressure,voltage) values(‘2018-09-09 12:00:00’, 20, 300, 380)，向关系库中插入一条记录。

开发报表要做的是根据这张表中的数据生成最终的报表；导数作业要做的是通过 sql 语句向数据库表中插数据；导数作业管理则用来指定填哪些数据（字段关联哪个数据点）和如何填（哪些时刻应该插入数据，插入数据时使用哪个 sql 语句等）。

如果报表形式比较复杂，insert 语句不能满足需求，所以需要存储过程来完成插入操作。对于上面的例子，对应的存储过程是：

Create proc usp_ RT_Report @NowDate datetime, @Curtemp float, @Curpres float, @Curvol float

As

Insert RT_Report (time, temp, pressure, voltage)

values(@NowDate, @ Curtemp, @ Curpres, @ Curvol)

Go

在导数作业程序中，用于插入记录的 sql 语句必须是一个存储过程，并且存储过程具有一个名为@NowDate 的参数，其类型为 datatime，用于记录数据采样的时间（即使这个时间不插入到数据库中）。

编制一个导数作业过程如下：

1. 建立报表所需的数据库表；
2. 编写向数据库表中插数据用的存储过程；
3. 在导数作业管理中建立存储过程的参数和实时库测点间的关联关系；
4. 导数作业服务读入报表定义信息，并按要求导出数据；

4.6.2 导数组态

双击解决方案中的“实时导数作业”节点，弹出导数作业管理窗口，如下图：

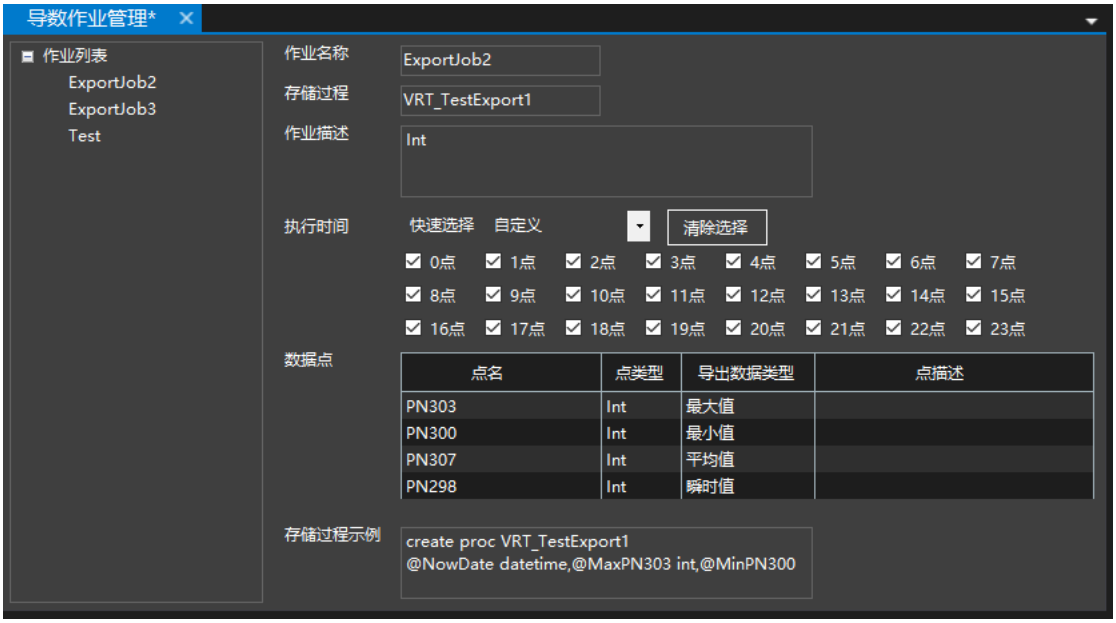


图 4-9 导数作业管理

在窗口左侧作业列表树中右键弹出菜单，如下图：

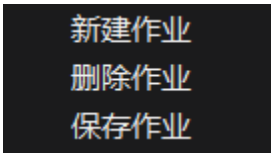


图 4-10 导数作业管理菜单

新建作业：添加新的导数作业；

删除作业：删除作业列表中选择的作业；

保存作业：新建作业、编辑作业后，进行数据保存。

点击作业列表树上节点，窗口右侧显示导数作业信息，导数作业内容说明如下：

存储过程：导数作业定时运执行的存储过程名称；

执行时间：每天执行导数的时间；

数据点：本次作业导出的统计数据、实时数据的集合，导出数据点数据类型为，最大(小)值：上次导数时间到本次导出时，采集到该数据点的最大(小)值；平均值：上次导数时间到本次导出时，采集到该数据点的平均值；瞬时值：执行导数时，该数据点的当前实时值；方差：上次导数时间到本次导出时，采集到该数据点的方差；求和：上次导数时间到本次导出时，按采集周期累计求和；增量：上次导数时间到本次导出时，采集到该数据点的差值；变位记数：上次导数时间到本次导出时，采集到 Bool 类型数据点的变化次数。在点集合中右键弹出导数点编辑菜单，如下图：

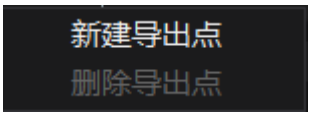


图 4-11 导数点编辑菜单

在导数点列表中，选中某一个点，在属性页中可编辑该点的导出数据类型，布尔、字符、数组、时间的数据点的导出数据类型只能为瞬时值。

编辑所有导数作业后可在导数列表中右键，执行保存。

存储过程示例：系统给出执行导数存储过程的定义主体信息，方便用户编写存储过程。

在 2.6.1 中配置启用导数服务，再次启动实时历史数据服务时，系统启动导数服务。

4.7 报警点组态

测点中有些点的值超过一定限值是需要产生报警，这些点称为报警点，报警点是测点的一个子集。

4.7.1 报警点属性

模拟量报警点

- 高4报警值~低4报警值：系统提供8个报警级别，超过对应值，根据对

应报警设置产生报警级别；

- 高4报警设置~低4报警设置：配置当前值超过报警值时产生报警颜色；
- 高4报警说明~低4报警说明：当发生报警时，给与报警说明信息；
- 模拟量报警死区：当数据回落时，值小于低级别报警值与报警死区之和时，报警等级不发生变化；

开关量报警点

- 开关量报警值：当值为True或False时发生报警；
- 开关量报警设置：配置当前值为报警值时产生报警颜色；
- 开关量报警说明：当发生报警时，给与报警说明信息；

数据点根据类型不同，属性变化说明：

- 模拟量类型数据点：不具有开关量报警值、开关量报警设置、开关量报警说明；
- 布尔型类型数据点：不具有高 4~低 4 报警值、高 4~低 4 报警设置、高 4~低 4 报警说明；
- 字符、字符串、数组、时间：不具有报警相关属性；

4.7.2 报警点配置

报警点组态界面如下图：

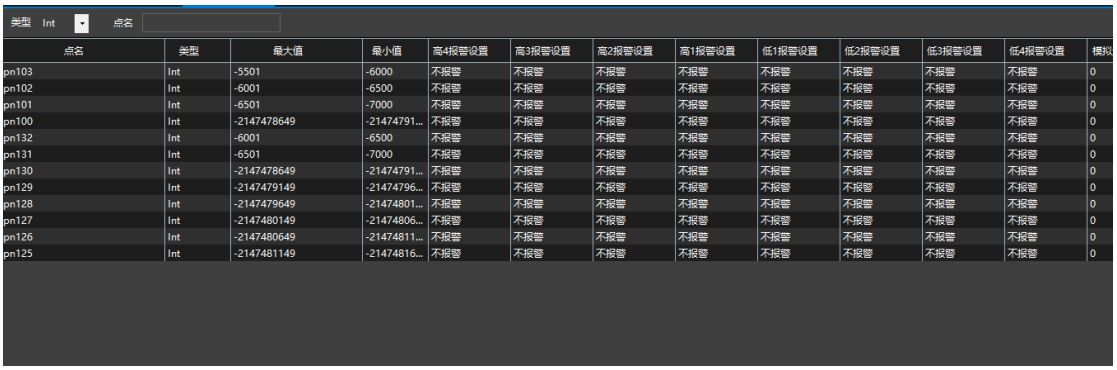


图 4-12 报警点点管理

通过窗口头部的类型、点名的筛选，窗体中显示符合条件的报警点点集合。在列表选中一个点，该点属性显示在属性页中，对点进行编辑。

在窗口中右键弹出菜单，如下图：

图 4-13 报警点菜单

新建：添加一个数据点；

保存：保存所有数据点信息；

删除：删除当前所选数据点；

列表删除：删除点列表中所有点；

全部删除：删除系统中所有的点；

报警点导出：将当前所有点导出到 CSV 文件中；

报警点导入：将 CSV 文件中的点导入到系统中。

提示：批量添加点时，采用先用 CSV 文件编写点信息，然后再导入数据库中，如不知某一属性如何填写时，可在窗口中新建一个点，然后执行数据导出，然后参考导出文件编写点信息。点类型不同编辑属性有所差别，CSV 文件中不需填写全部属性，属性编写参考 4.7.1 中“数据点根据类型不同，属性变化说明”部分。

CSV 文件采用“,”分割。

4.8 报警组组态

4.8.1 报警组组态

双击解决方案中的“报警组态”节点，弹出报警组态窗口，如下图：

图 4-14 报警组态

在窗口左侧报警列表树中右键弹出菜单，如下图：

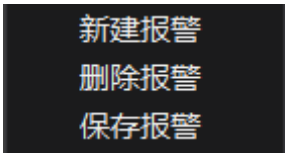


图 4-15 报警组态菜单

新建报警：添加新的报警界面；

删除报警：删除报警列表中选择的报警；

保存报警：新建报警、编辑报警后，进行数据保存。

在报警列表中选择某一报警时，在属性页中显示此报警信息：

页面名称：报警页面显示标题文字；

背景色：报警页面的背景颜色；

边框颜色：报警页面中报警表格的边框颜色；

字体颜色(大小)：报警页面中报警点文字的颜色(大小)；

描述：此报警的说明信息。

在窗口右侧报警点列表中右键弹出报警点编辑菜单，如下图：

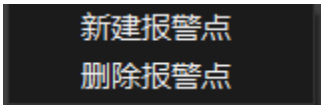


图 4-16 报警点编辑菜单

能添加到报警中的数据点需在数据点组态中完成报警信息配置，报警等级选择、报警限制设定。推荐填写报警信息说明，以便在报警时给与更多说明。

报警点排序：在报警页面中每个报警点显示的排列顺序。

在 2.6.1 中配置启用报警服务，再次启动实时历史数据服务时，系统启动报

警服务。

4.8.2 报警界面配置

在窗口头部显示的 Vicdas 相对地址，描述了此报警页面配置到 Vicdas 企业门户的网页路径(提供了 PC 端与移动端的路径)。

用 Vicdas 门户管理员登录，进入菜单管理界面，添加此报警的 PC 端页面和移动端页面，并将此页面访问权限分配给相关用户角色，即可查看报警实时信息。

WEB 配置时：区域“vdb”，控制器“alarm”。

4.9 趋势组组态

4.9.1 趋势组组态

双击解决方案中的“趋势组组态”节点，弹出趋势组组态窗口，如下图：

图 4-17 趋势组组态

在窗口左侧趋势组列表树中右键弹出菜单，如下图：

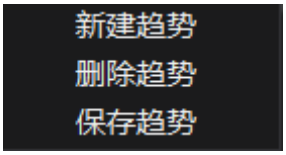


图 4-18 趋势组组态菜单

新建趋势：添加新的趋势界面；

删除趋势：删除趋势列表中选择的趋势；

保存趋势：新建趋势、编辑趋势后，进行数据保存。

在窗口右侧趋势点列表中右键弹出趋势点编辑菜单，如下图：

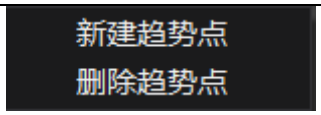


图 4-19 趋势点编辑菜单

报警点线颜色：在趋势页面中每个数据点的趋势曲线的颜色。

4.9.2 趋势界面配置

在窗口头部显示的 Vicdas 相对地址，描述了此趋势页面配置到 Vicdas 企业门户的网页路径(提供了 PC 端与移动端的路径)。

用 Vicdas 门户管理员登录，进入菜单管理界面，添加此趋势的 PC 端页面和移动端页面，并将此页面访问权限分配给相关用户角色，即可查看实时趋势曲线。

WEB 配置时：区域“vdb”，控制器“trend”。

4.10 历史数据维护

因某种原因需对数据点的历史数据进行补录或删除，通过历史数据维护功能，对数据进行维护工作，使其为完整的数据集合。

4.10.1 维护要求

- 数据维护需遵循：

1. 维护数据的时间戳要小于当前日；
2. 一次性补录数据的文件大小要小于 60M；
3. 尽量避免在访问历史数据繁忙期间修改历史数据。

- 数据文件格式如下

增加历史数据格式，点名、时间、值用英文","分隔，(.fff)表示可选：

点名,add,HH:mm:ss(.fff),值

删除历史数据格式，点名、时间、值用英文","分隔，(.fff)表示可选：

点名,del,开始时间 HH:mm:ss(.fff),结束时间 HH:mm:ss(.fff)

数据文件示例：

文件名：PNData.txt

文件内容如下：

PN110,add,11:23:34,100

PN110,add,11:23:38,101

PN110,add,11:23:41,102

PN111,add,11:23:34,200
PN111,add,11:23:38,201
PN111,add,11:23:41,202
PN111,del,11:20:00,11:21:00
PN113,del,10:10:00,10:15:00

4.10.2 维护流程

双击解决方案中的“历史数据维护”节点，历史数据维护窗口，如下图：

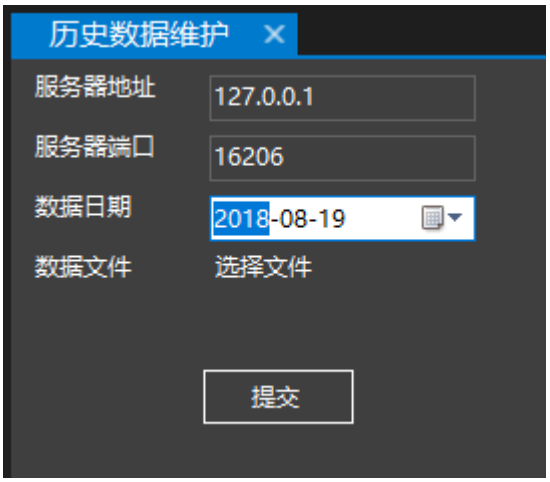


图 4-20 历史数据维护窗口

1. 填写服务器地址：实时历史数据服务地址；
2. 填写服务器端口：系统默认 16206，如在 2.6.1 中修改端口，填写修改后端口。如在 2.6.1 中服务器 1、2 都已配置，任选一个服务器 ip 和端口即可，需保证组态工具所在的计算机 IP 与配置服务器 IP 可相互通讯；
3. 选择数据维护日期；
4. 选择数据文件；
5. 点击“提交”按钮，等待系统数据维护，如系统中数据点量较多如 5 万点，维护时间较长，请耐心等待；
6. 系统数据维护结束时提示完成数据维护。

5. 图形组态

图形组态是用户通过工具对系统进行绘制、动态特性配置构建出系统流程图实时监视功能。Vicdas 系统使用 VGraph.exe 作为图形组态组态工具。

流程图组态分为两个部分：1、图形编辑：绘制流程图图形文件，配置图形动态特性；2、解决方案：用来管理流程图组态文件上传、下载、uri 地址。

流程图组态组态过程为：先在“图形编辑”中创建图形文件，并进行绘制编辑及保存，然后在“解决方案”中创建一个流程图，在流程图中上传已编制完毕的图形文件。3、在 WEB 企业门户的菜单管理中配置流程图菜单，并将访问菜单权限分配给相关角色及用户。

推荐流程图编辑过程：先用 SVG 图形编辑专业软件（如 Adobe Illustrator）绘制界面，保存为 svg 文件，将此文件设置为为流程图背景图片，然后再绘制需要进行动态显示的图形，再配置动态特性，编辑完毕后，保存流程图，上传发布。

启动“安装目录\VicdasGraph\VGraph.exe”，弹出登录界面，如下图：

图 5-1 登录界面

在登录界面输入用户名：vadmin，密码：当前日期，如 20180909，点击登录或回车，弹出组态界面。

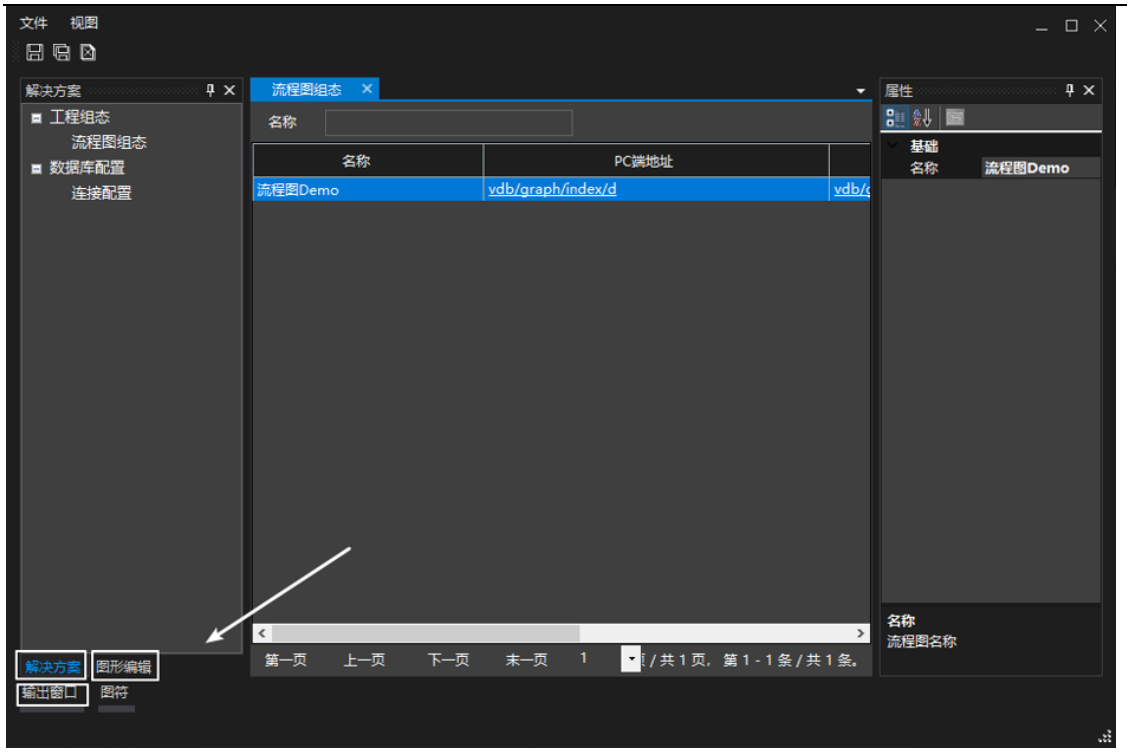


图 5-2 操作界面

注意:

1. 系统登录时，会检测数据库连接，如仅使用“图形编辑”功能，可不配置数据库连接，不影响功能使用。
2. “输出窗口”未像“工程组态”工具默认为显示，是为了考虑绘图时尽可能获取界面尺寸，在“解决方案”操作及部分“图形编辑”时，注意“输出窗口”给出的提示信息，能够给予很大帮助。

5.1 图形编辑

点击界面中“图形编辑”，操作界面左侧切换到界面如下图:

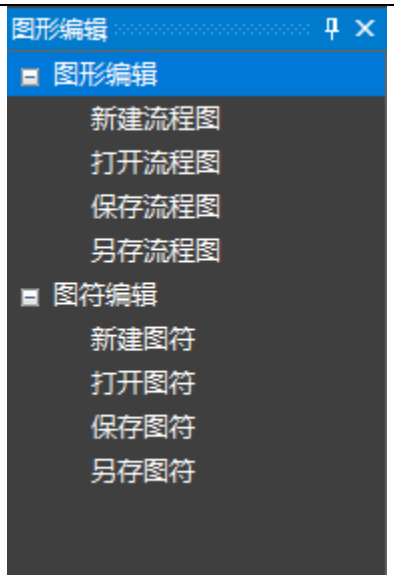


图 5-3 图形编辑菜单

双击“图形编辑”子项和“图符编辑”子项，操作区显示操作界面。当鼠标点自己“图形编辑”与“图符编辑”节点后，鼠标右键，页面显示出其子项菜单。

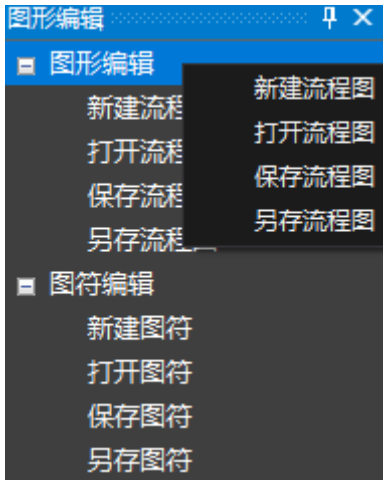


图 5-4 图形菜单

在流程图编辑界面，绘制流程图后，点击图形，在操作界面右侧的属性栏中显示该图形的属性列表，可对该图形属性进行编辑，示例如下图：

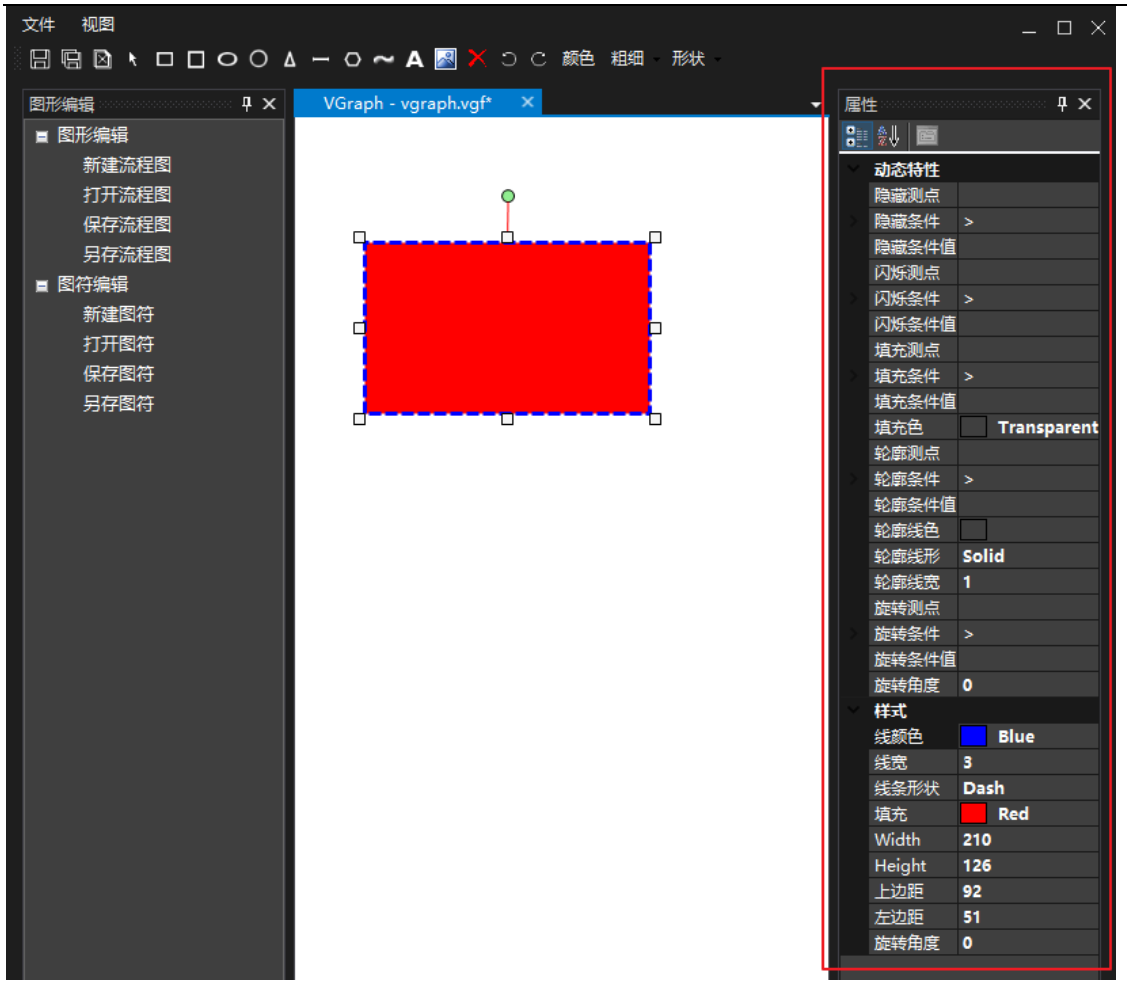


图 5-5 图形属性

5.1.1 系统图形

系统支持的形状有：矩形、正方形、椭圆、圆、三角形、直线、多边形、曲线、文字、图片及图符。

正方形、矩形绘制、椭圆、圆、三角形、直线、图片：在操作区左键点击并按住左键，滑动鼠标到适当位置，放开鼠标，完成矩形绘制。

多边形：在操作区左键点击鼠标，移动鼠标再次点击、再次移动点击，以此类推，绘制多边形的几个点需要点击几次鼠标，绘制完毕后，点击鼠标右键，结束编辑。

曲线：操作区左键点击并按住左键，滑动鼠标到适当位置，放开鼠标，界面显示一条默认曲线，然后选择图形调整点，点击并按住左键，滑动鼠标到适当位置，放开鼠标，完成曲线绘制。

图符：图符绘制需先创建及编辑图符，保存图符，完成图符编辑；然后打开

图形编辑界面，点击“图符”列表，

图 5-6 图符列表

右键导入已有图符

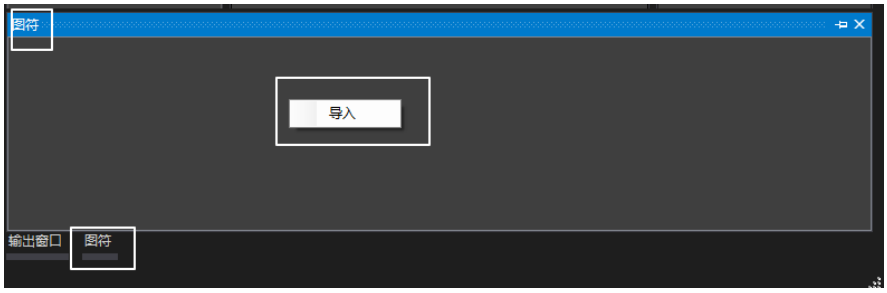


图 5-7 导入图符

在图符列表中左键点击按住图符，移动鼠标到编辑区，放开鼠标，完成图符绘制，如下图：

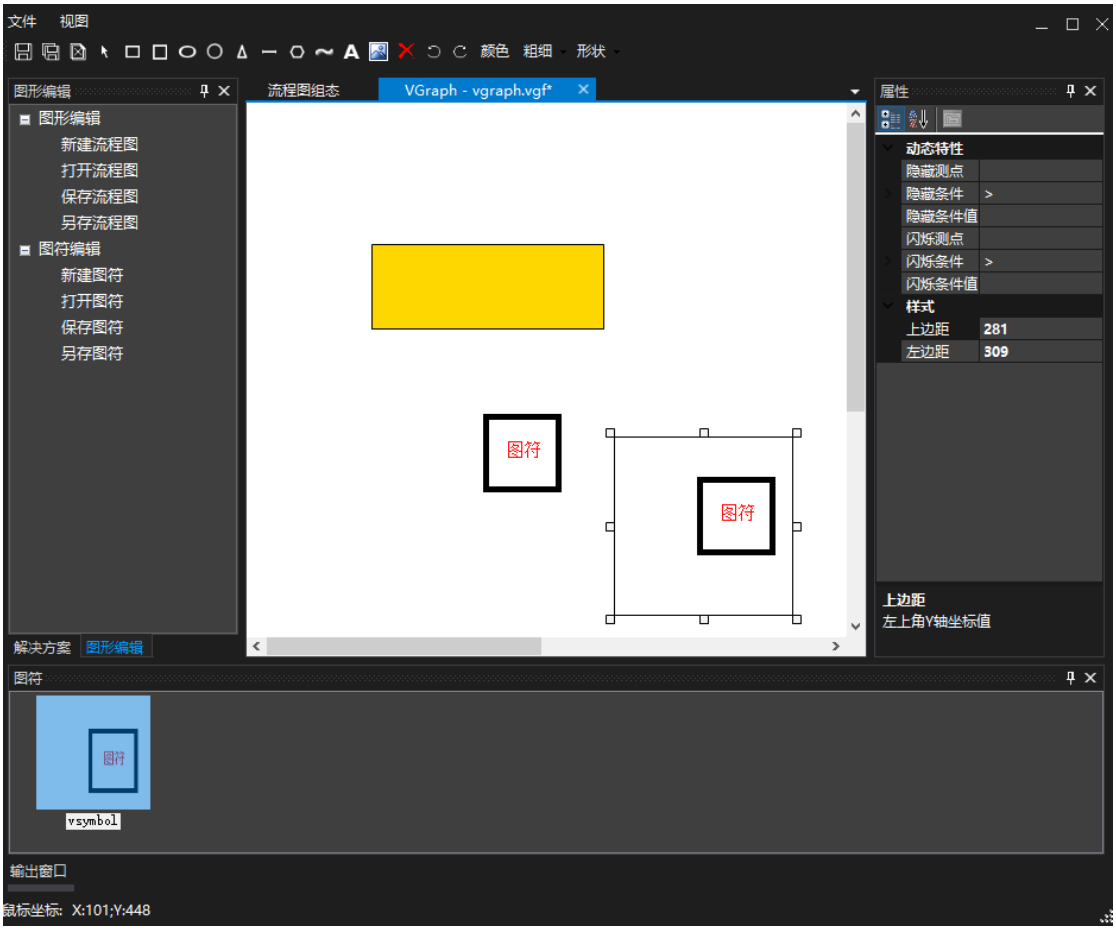


图 5-8 图符绘制

5.1.2 动态特性

图形动态显示主要有以下动作：

- 发生条件：

条件点：测点的点名，点的实时值作为判断参数值

条件：数据点值与条件值比较方式，“>=”、“<=”、“>”、“<”、“==”、“!=”6 中条件比较方式；当条件点为非数字型，条件值只能配置“==”、“!=”。

条件值：条件比较目标值，当“条件点的实时值”“比较条件”“条件值”组合值为 true 时，系统执行动态特性。

- 动态特性：

闪烁：关联数据点符合条件时，图形在界面上发生闪动；不配置，系统默认不闪烁

隐藏：关联数据点符合条件时，图形不显示；不配置，系统默认显示。

字体：关联数据点符合条件时，字体大小、颜色从系统配置值变为字体的动态特性配置值。

显示：关联数据点符合条件时，图形显示在界面中；不配置，系统默认显示。

边框：关联数据点符合条件时，图形边框颜色、边框线性、宽度变化。

填充：关联数据点符合条件时，图形填充颜色变化。

旋转：关联数据点符合条件时，图形以图形中心转动配置角度。

实时值：文本显示为实时显示测点的当前值。

5.1.3 流程图属性

点击流程图任何一处，属性栏显示属性，如下图：

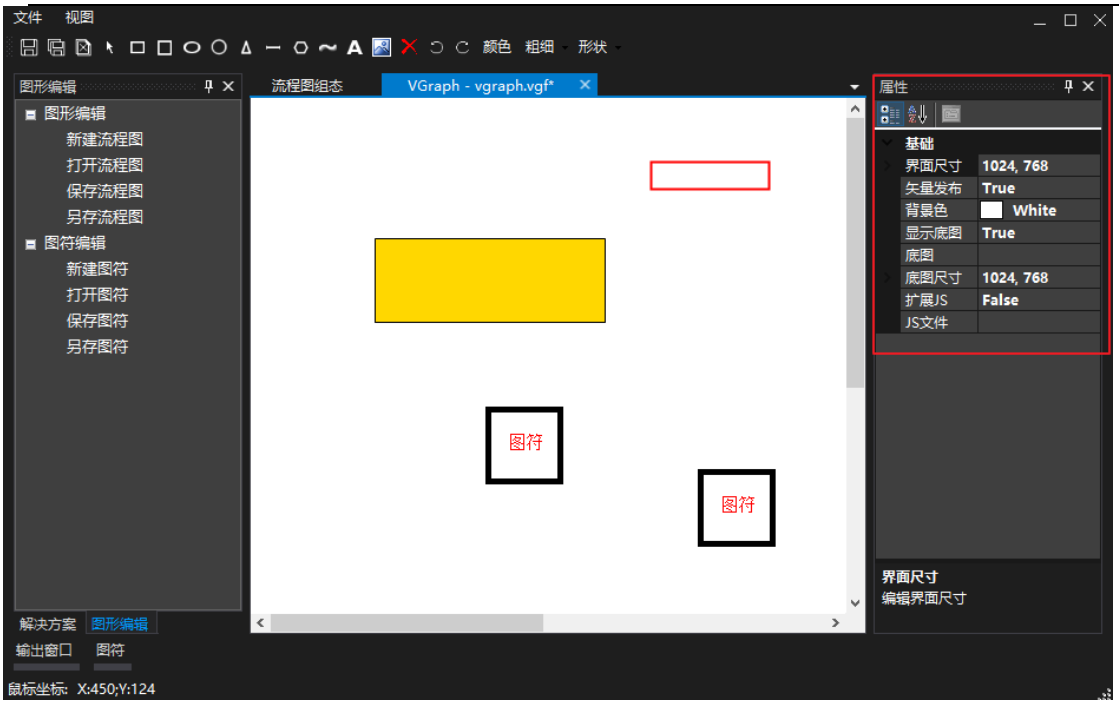


图 5-9 流程图属性

界面尺寸：流程图默认大小，发布到 WEB 页面时默认值按照此尺寸尽心绘制。

矢量发布：当选择为事项发布时，流程图 WEB 页面显示大小按照浏览器界面大小进行矢量缩放。

背景色：流程图背景色，也是 WEB 页面的背景色。

底图：可选择 svg 文件或者位图作为流程图底图。

扩展 JS：用户自定义 JS 文件，会随流程图一起发布到 WEB 端，此 JS 文件只对本流程图有效。

5.1.4 快捷键

ctrl+c：复制

ctrl+v：粘贴

ctrl+z：后退

ctrl+y：后退

ctrl+方向键：快速移动

alt+方向键：较慢移动

delete：删除

5.2 流程图组态

5.2.1 数据库连接配置

请参见 2.6.4

5.2.2 流程图组态

双击解决方案中“流程图组态”，显示如下：

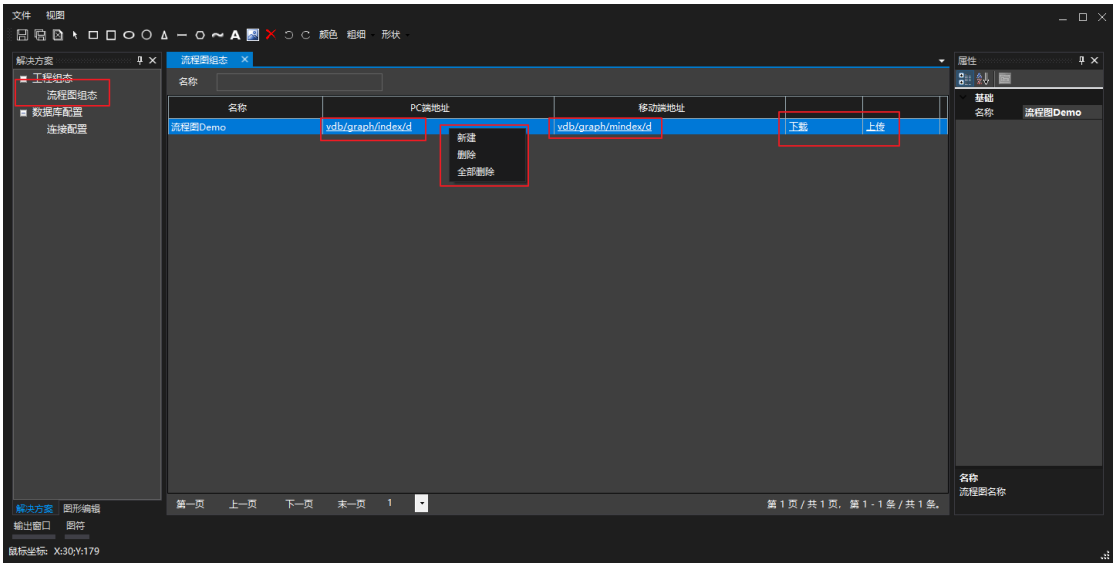


图 5-10 流程图组态

在操作区右键，显示操作菜单，进行创建与删除流程图操作。

选择一个流程图，点击“上传”按钮，选择已绘制完成的图形，完成上传工作。如图形存在问题或上传成功，信息显示在“输出窗口”中。

点击“PC 端地址”或“移动端地址”，系统将地址复制到内存中，信息显示在“输出窗口”中。如下图：

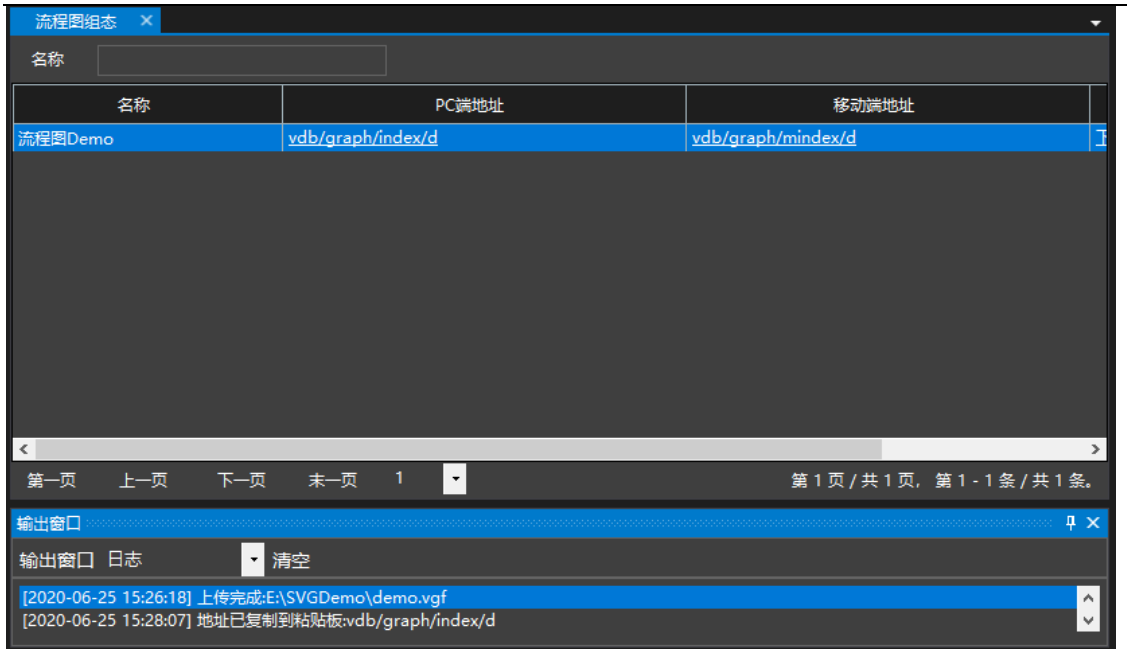


图 5-11 流程图组态信息

点击“下载”，将此流程图的图形文件从服务端下载到本地。

5.2.3 流程图界面配置

在窗口头部显示的 Vicdas 相对地址，描述了此趋势页面配置到 Vicdas 企业门户的网页路径(提供了 PC 端与移动端的路径)。

用 Vicdas 门户管理员登录，进入菜单管理界面，添加此流程图的 PC 端页面和移动端页面，并将此页面访问权限分配给相关用户角色，即可查看实时流程图。

WEB 配置时：区域“vdb”，控制器“graph”。

6. 实时计算组态

系统提供 javascript 脚本引擎和脚本运行接口实现实时数据在线计算，用户通过自定义 js 脚本，对实时数据进行二次计算及统计分析。

6.1 JavaScript 脚本编写

6.1.1 脚本目录

js 公共类库存放目录：系统安装目录\VDB\CalcScript\Lib，无公共库此目录为空，路径示例 C:\VicdasSystem\VDB\CalcScript\Lib

js 脚本存放目录：系统安装目录\VDB\CalcScrip，系统启用计算服务，需保证有计算脚本，路径示例如 C:\VicdasSystem\VDB\CalcScript。脚本执行顺序，按照脚本名称排序执行。

6.1.2 编写说明

1. 脚本编写遵循 javascript 语法规则，版本为 ES5 及以下；
2. 每一个作为公共类库的 js 文件编写完毕后放置脚本库目录，无公共库 js 文件此目录为空；
3. 每一个 js 脚本文件，编写完毕后放置脚本；
4. 每一个 js 脚本文件需实现 1 个接口函数和 1 个变量赋值：CalcMain()和 CalcPeriod。 CalcMain()函数，用于定义脚本运行入口，从此方法开始执行；CalcPeriod 属性表示此脚本计算周期，单位毫秒，值要求为 500 的整数倍的值，如 1000,1500 等，如果不设置此属性，系统默认 500 毫秒，每隔这个周期，系统执行一次脚本。
5. js 脚本作为独立运行，脚本间互不影响，每个脚本内对象作用域仅在一次执行内有效，多次执行同一脚本，每次执行无影响。
6. 如脚本之间、每次执行之间需要存在数据传递，可以使用中间点实现：1、从组态工具组中建立一个中间点采集器；2 根据需求组所需测点，设置测点属于中间点采集器，数据存储不保存。3、在脚本中通过 6.2 的接口将值读写到测点中。实现各个脚本之间调用。

6.2 实时库 JavaScript 接口

6.2.1 测点读取

接口： vicdasApi. getValue(point)
输入参数：point 可以为点名称，也可以为点 ID
返回值：

```
{
    value: 测点值,
    state: 测点状态,
    alarmlevel: 报警等级,
    alarmtime: 报警时间
}
```

6.2.2 批量测点读取

接口： vicdasApi. getValues(point[])
输入参数：point 可以为点名称，也可以为点 ID，
返回值：

```
[{
    value: 测点值,
    state: 测点状态,
    alarmlevel: 报警等级,
    alarmtime: 报警时间
},
{
    value: 测点值,
    state: 测点状态,
    alarmlevel: 报警等级,
    alarmtime: 报警时间
}]
```

6.2.3 测点写入

接口： vicdasApi. setValue(point, value)
输入参数：point 可以为点名称，也可以为点 ID；value 为测点值
返回值：bool 类型，true 执行成功，false 执行失败。

6.2.4 批量测点写入

接口： vicdasApi. setValues(vals)

输入参数：vals 为字符串，是对象{ pn:'', value: }集合序列化后的字符串，如“[{ pn:'pn1', value:34},{ pn:34, value:15}]”

pn 可以为点名称，也可以为点 ID；value 为测点值

返回值：bool 类型，true 执行成功，false 执行失败。

6.2.5 日志写入

接口：vicdasApi. log(log)

输入参数：log 可以为数值型、字符型也可为对象。日志写入数据服务的日志中。

返回值：无

说明：除调试等需求，如无特殊原因，不建议脚本中调用日记接口。尤其在正式正产运行中。

6.3 脚本示例

定义公共库文件：jslib.js，内容如下

```
function ExecAdd(a, b)
{
    return a + b;
}

function ExecDecrease(a, b)
{
    return a - b;
}
```

定义算法脚本：demoscript.js，内容如下

```
//脚本执行周期
CalcPeriod = 500;

//每个脚本的计算入口
function CalcMain()
{
    try
    {
        var p1 = vicdasApi.getValue("pn1");//点名称
        var p2 = vicdasApi.getValue(2);//点 ID
```

```
            var p1abs = getABS(p1.value);//本脚本的方法
            var calc = ExecAdd(p1abs, b.value);//公共库的方法
            vicdasApi.setValue("pn752", calc);//点名称
            vicdasApi.setValue(753, calc);//点 ID
        }
        catch (e)
        {
            //vicdasApi.log(e);   日志
        }
    }

    function getABS(obj)
    {
        return Math.abs(obj);
    }
```

jslib.js 放置安装目录\VDB\CalcScript\Lib 目录中，demoscript.js 放置安装目录\VDB\CalcScript 中，在 2.6.1 中配置启用计算服务，再次启动实时历史数据服务时，系统启动在线实时计算服务。

7. 企业门户

Vicdas 系统企业门户部分为企业生产信息化系统提供了用于系统权限控制的用户管理、角色管理、菜单管理；统提供用于查看实时历史数据的流程图、报警及趋势界面；系统提供二次开发使用的人员管理、组织机构管理、数据字典管理、密码管理、ORM 及工作流引擎类库等功能。

门户登录网址，如系统自动安装门户地址：

PC 端：http://服务器 ip/vicdas 或 http://服务器 ip/vicdas/default/login；如 http://127.0.0.1/vicdas

移动端：http://服务器 ip /vicdas/default/mlogin 注：基础信息及其子页面不支持移动端。

如手动安装门户的地址请咨询安装人员。

系统初始提供最高权限管理员账户：vadmin，初始密码：1。

7.1 二次开发支持简述

1. 人员管理、组织机构及数据字典是方便企业生产信息系统业务快速开发提供数据管理界面。配置完毕的数据，开发人员可自行访问数据库或调用 VBLL.dll、VModel.dll 类库使用。
2. 系统提供轻量级 ORM 类库 ESword.dll，支持快速操作数据库。

```
using System;
using ESword.ORM;
namespace VModel
{

    系统内置值与 C#的 Enum 的描述一致

    [Serializable]
    [TableName("vw_role")]
    public class RoleModel : BaseModel
    {
        [TableColumn(TableColumnType.IdentityPrimaryKey)]
        public int RoleID
        { set; get; }

         [TableColumn(TableColumnType.CommonColumn, 30)]
        public string RoleName
        { set; get; }

        [TableColumn(TableColumnType.CommonColumn, 200)]
```

```
        public string RoleDes
        { set; get; }

        [TableColumn(TableColumnType.CommonColumn)]
        public int RoleType
        { get; set; }

        public string RoleTypeName
        { get; set; }
    }
}
```

RoleModel 的静态方法和实力化对象的方法完成 RoleModel 到表 vw_role 的数据操作.

7.2 系统权限控制

7.2.1 菜单管理

菜单管理用于维护系统中菜单列表，vadmin 用户登录系统后，在基础信息列表中选择菜单管理，弹出界面如下图：

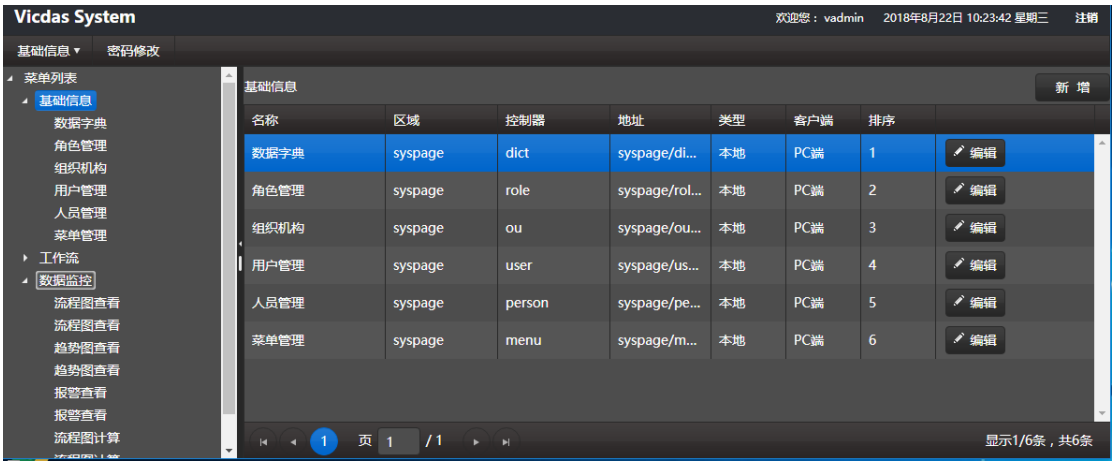


图 7-1 菜单管理

界面左侧菜单列表树为系统中所有菜单，“菜单列表”的第一级子节点为系统一级菜单，一级菜单第一级子节点为二级菜单，依次类推；在界面左侧选择节点，界面右侧显示内容为其子菜单。

界面中基础信息及子项，密码管理属于系统必须菜单，不可删除，其他菜单可删除。

菜单属性：区域、控制器对应 asp.net mvc 的区域与控制器，自开发页面请填写相关数据，如无区域、控制器或界面为第三方页面，区域、控制器也可不填。

注：系统已使用 syspage、vdb 两个区域

7.2.2 角色管理

角色是系统中具备一定页面访问权限的用户组，角色管理用于配置角色信息及可访问页面的集合。在基础信息列表中选择角色管理，弹出界面如下图：

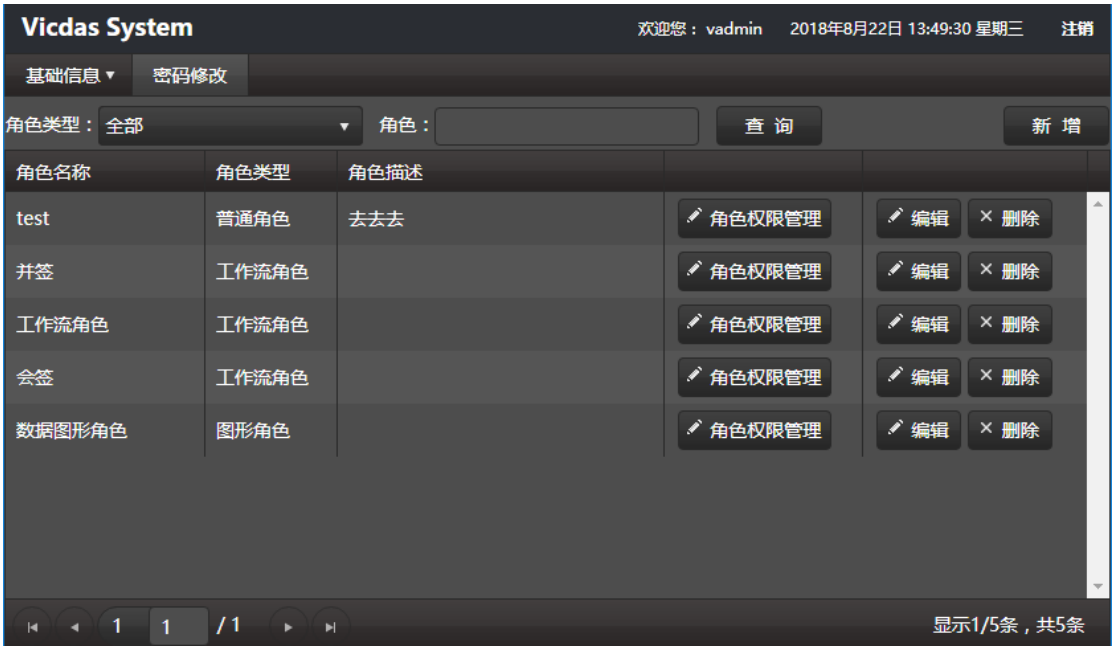


图 7-2 角色管理

角色类型：仅为方便查询的标识。工作流页面配置工作流角色；实时历史数据查看页面配置图形角色；其他角色选择普通角色即可。

新建角色并开始分配可访问页面，点击“角色权限管理”，弹出界面如下：

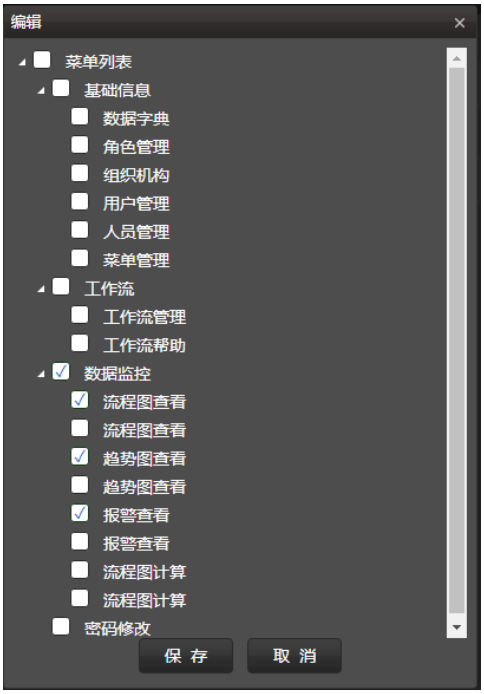


图 7-3 角色菜单

列表内容为“菜单管理”中配置的菜单列表。

选择角色可访问的页面，点击保存按钮，页面访问权限分配完毕。

7.2.3 用户管理

用户管理用于管理访问系统的账户。

图 7-4 用户管理

密码重置：当用户忘记密码时，管理员可初始化其密码，密码值为 XX 配置内容

角色设置：将用户分配到某一个角色中时，用户即可访问角色中配置的页面。

7.2.4 配置示例

实时历史数据查看需系统管理员将查看页面在系统中配置才可查看，同理，业务二次开发页面也需系统管理员将页面在系统中配置才可查看，以流程图为例说说明配置过程。

以系统管理员 vadmin 账户登录，选择“基础信息”下的“菜单管理”，系统转向菜单管理界面

点击界面左侧列表根节点“菜单列表”，在界面右侧点击“新增”按钮，填出新增界面，如下图填写

图 7-5 新增菜单

客户端：表面此菜单在不同的客户端是否显示；

点击“保存”按钮，完成新建一级菜单；

在组态工具中，将趋势、报警、流程图组态完毕，每一个组态界面头部都会给出 PC 端地址和移动端地址，见 4.8、4.9 复制地址信息，如 PC 端：vdb/trend/group/dd

点击界面左侧列表根节点中新增加菜单“数据监控”，在界面右侧点击”新增”按钮，填出新增界面，如下图填写

区域和控制器：如界面是业务二次开发，请根据界面在项目中开发位置自行填写；如界面是 vicdas 的实时历史数据查看页面必须按照“区域：vdb，控制器：trend”填写。

类型：本项目中的页面选择本地，非本项目中选择远程

菜单地址：填写从组态工具中复制的 PC 端地址；本地的页面填写相对地址，非本项目填写全路径

客户端：选择 PC 端

点击“保存”按钮，完成菜单配置。如配置移动端，类型选择移动端，菜单地址如是实时历史数据页面请填写对应移动端地址。

选择”基础信息”下的”角色管理”，添加或者编辑一个角色，如“数据图形角色”，并在角色权限管理中勾选新添加的”数据监控”，”流程图查看”两个菜单，点击保存。

选择”基础信息”下的”用户管理”， 添加或者编辑一个用户如张三，点击角色设置，填出窗口如下：

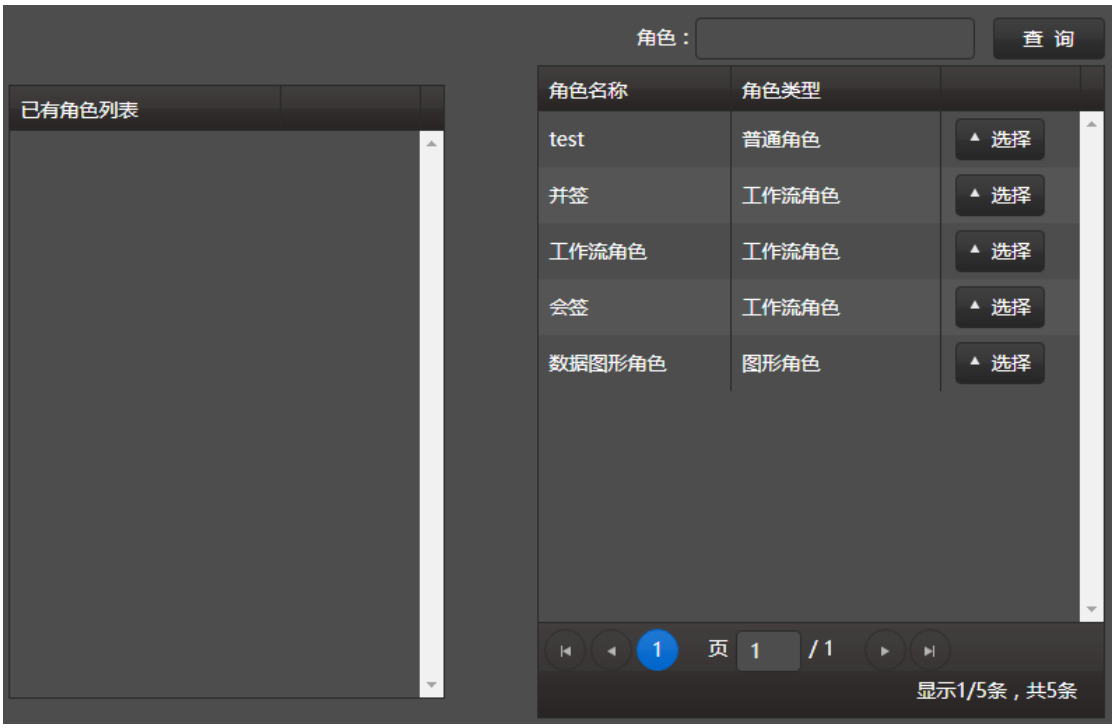


图 7-6 角色列表

选择上一步骤中的角色“数据图形角色”，添加到已有角色列表，关闭界面；退出系统，以用户 zhangsan 登录后，在菜单栏中可看刚添加菜单。

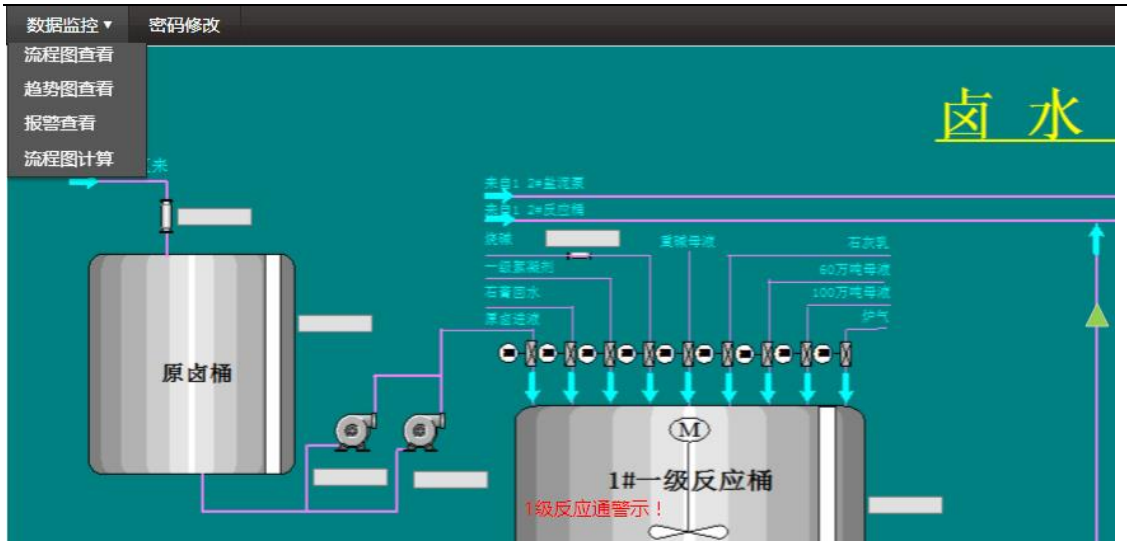


图 7-7 添加菜单示例

7.3 实时、历史数据

Vicdas 系统提供查看的趋势图、报警界面、流程图，查看的界面按照 4.8、4.9 中进行组态，并按 7.2 配置对应界面，即可查看。

7.3.1 流程图

点击某一个流程图菜单，进入界面，示例如下图：

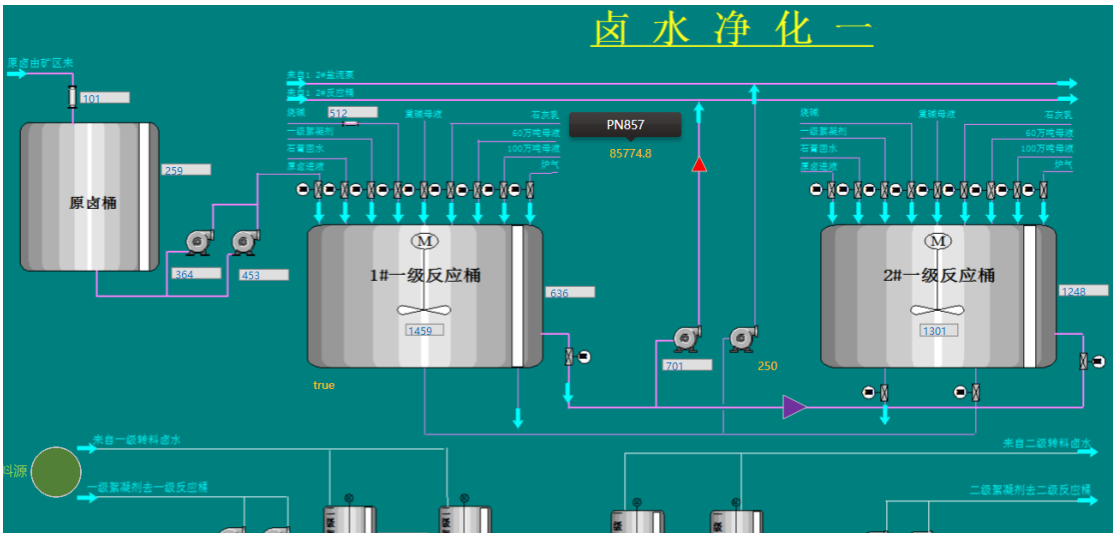


图 7-8 流程图示例

点击数据实时变化的数据点时，弹出查询点列表，如下图：

图 7-9 查询点列表

点击“添加”按钮，将此点添加到列表中，其次点击其它点并添加到列表中，然后点击“历史趋势查询”或“实时趋势查看”，进入历史、实时趋势界面，趋势曲线为列表中的点。

7.3.2 趋势图

点击某一个趋势图菜单，进入界面，示例如下图：

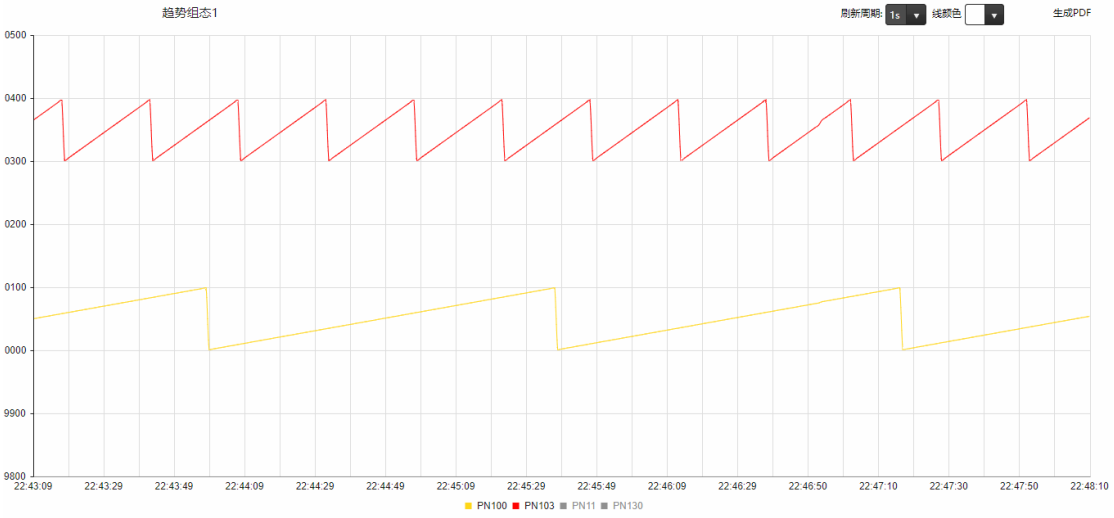


图 7-10 趋势图示例

界面显示趋势图组态中所配置点的实时数据趋势。

点击趋势线，弹出线颜色选择界面，选择颜色，点击“确定”后线颜色发生变化；

点击图下发点集合列表，可设置某条线是否显示。

7.3.3 报警

点击某一个报警菜单，进入界面，示例如下图：

报警组态11

点名	点值	等级	报警值	报警信息	确认
PN312	31300	H2	31260	31260	报警确认
PN301	30115	L3	30120	低3报警	报警确认
PN1354	true	置1报警	True	True红色报警	报警确认
PN1356	true				
PN1353	true	置1报警	True	True紫色报警	报警确认
PN1	115				

图 7-11 报警信息示例

界面显示报警组态中所配置点的实时报警信息：报警等级、报警颜色、报警值、报警描述等，当发生点值越限时系统发出报警，产生颜色变化与点信息闪动。

点击“报警确认”按钮，此次报警信息被确认，点信息不再闪动，等再次发生越限时产生报警，再次闪动。

7.3.4 自定义查询

提供自定义实时、历史趋势查询界面，一次最多支持 10 个测点查询，查询地址为：http://XXXX/VDB/Trend/Customize

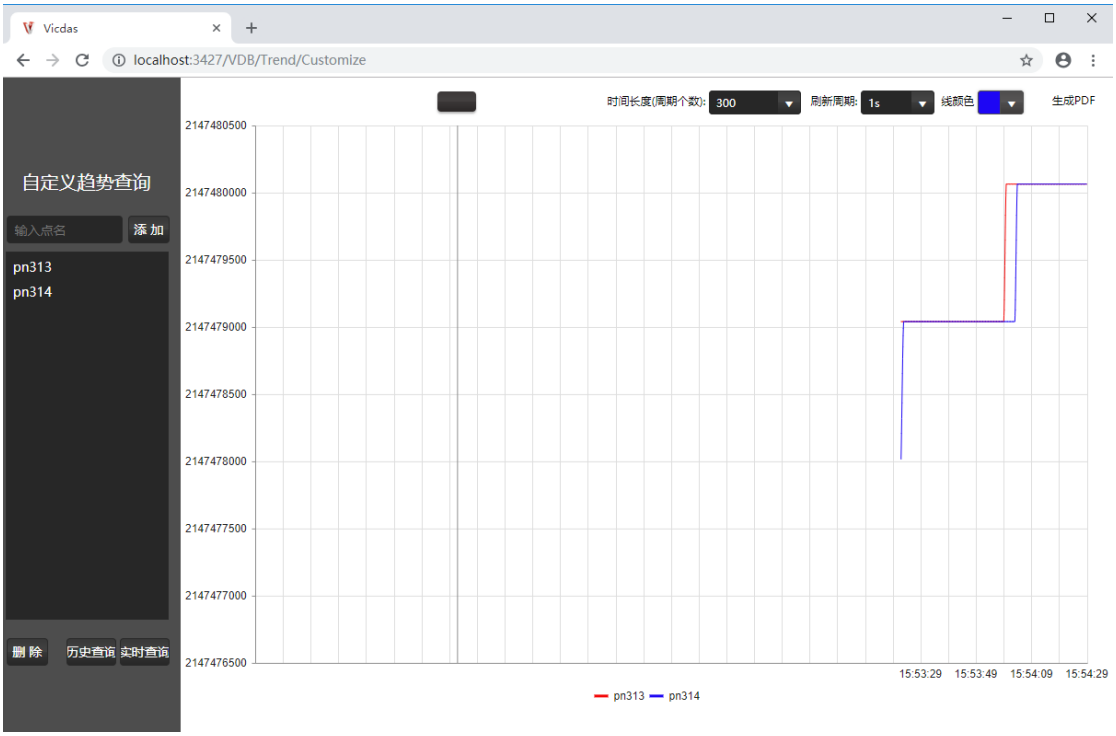
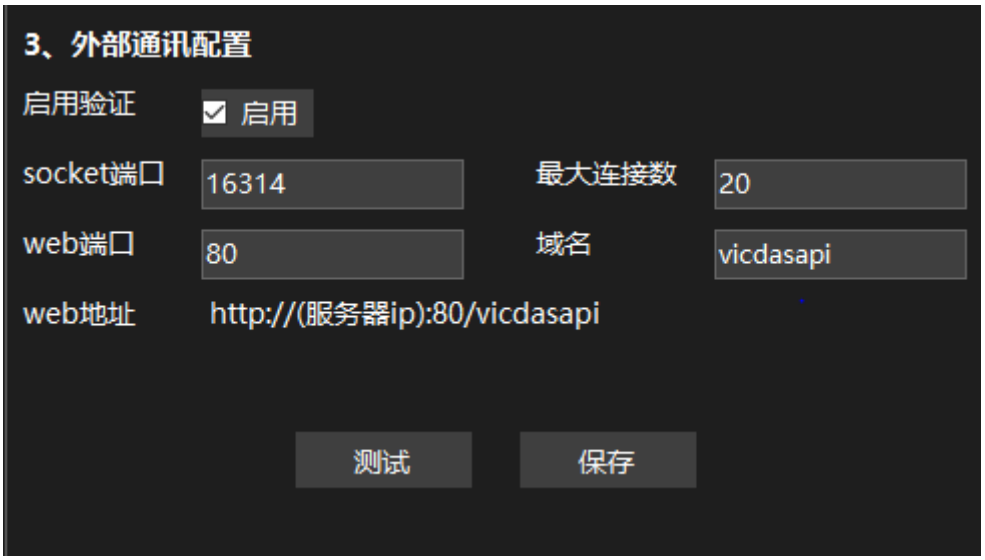


图 7-12 自定义查询

8. 数据访问 API

系统提供 Restful、SignalR、Socket 三种接口。接口访问地址参见数据服务的配置，数据服务配置工具的截图示例：



Restful 接口的访问地址为图片中“web 地址”，如：http://127.0.0.1/vicdasapi

SignalR 接口的地址(路由地址)为图片中“web 地址”如：http://127.0.0.1/vicdasapi

Socket 访问 IP 为服务器 IP，端口为图片中“socket 端口”，如 16314

Restful、SignalR、Socket 三种接口实现的功能相同，功能说明参见 Restful 接口中说明。

当数据服务勾选“启用验证”时，需给接口调用者提供许可，许可生成见组态工具的“接口访问许可”配置。

定义：

1. WEB 地址

   本章以下内容所使用为截图(数据服务配置的外部通讯配置)中 web 地址，地址字符串不区分大小写。本章节为显示清晰，地址字符串中使用大驼峰命名。

2. 时间格式

   输入、输出时间为 UTC 时间，世界时返回 1970 年 1 月 1 日到指定日期的毫秒数，单位毫秒。Restful、SignalR 接口输入时间参数也可以为自然时间格式，为“yyyy-MM-dd HH:mm:ss[.fff]”，自然日格式时间为当地时间。

3. 取样点及时间

设定在某一时间范围内的取样点个数，系统约定：开始时间、结束时间分别对应一个取样点，所以要求如使用取样点个数时，取样个数>=3。每个取样点对应时刻计算方式：时间范围/(个数-1)，获取对应取样点时刻的历史值，第一个点和最后一个点分别对应开始时间与结束时间。

8.1 Restful 接口

8.1.1 数据说明

1. 请求参数

   请求参数格式为 json 数据，请在 http 请求的 headers 添加 key：“Content-Type”，value：“application/json。

2. 返回数据

   返回数据格式为 json 数据，所有 api 接口返回数据都具有的属性为:

   code：执行结果代码，见 8.5

   msg：执行结果信息

   utc：系统接收到请求时间

   在后续描述中，这三个返回数据称为“通用数据”

3. 数据订阅

   系统提供订阅功能查询相关数据，订阅具有有效期，在最后一次访问后，2 小时内不再访问，订阅号失效。

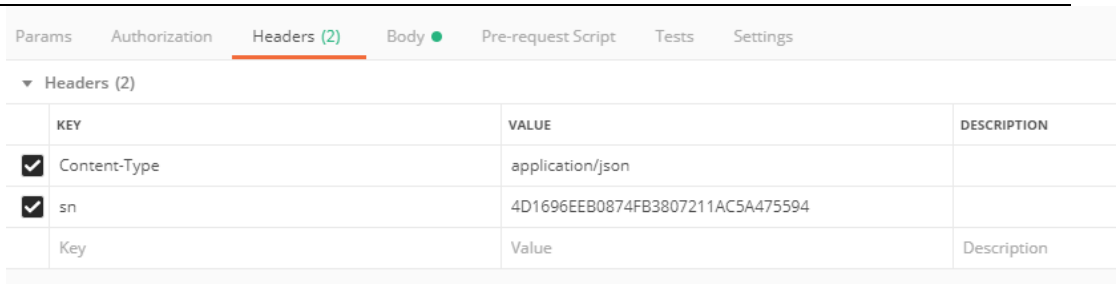
4. 示例数据格式

   示例中数据格式为 json，有时实例中“文字”代表字符，也可能代表数字，此时作为属性值，是否需要加引号，请根据 json 格式要求填写，如数字不需要引号、文字需要引号。

   示例中 json 数据仅为示例，实际编写参数时，请遵循 json 格式要求。

8.1.2 访问许可

服务端启用许可验证时，将许可添加到 http 请求的 headers 中，key 为“sn”，如图：



8.1.3 配置接口

8.1.3.1 测点列表

功能：获取系统中所有测点

地址：WEB 地址/CnfgRest/Points

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"id":"点ID","pn":"点名称","pt":"点类型","save":"数据存储方式","des":"点描述" }，{"id":"点ID","pn":"点名称","pt":"点类型","save":"数据存储方式","des":"点描述" }...]

}

8.1.3.2 报警点列表

功能：获取系统中所有报警点

地址：WEB 地址/CnfgRest/AlarmPoints

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"id":"点ID","pn":"点名称","pt":"点类型"}，{"id":"点ID","pn":"点名称","pt":"点类型" }...]

}

8.1.4 实时数据接口

8.1.4.1 测点(ID)值

功能：通过测点 ID 获取测点实时值

地址：WEB 地址/RTRest/ValueByID/点 ID

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态"}]

}

8.1.4.2 测点(名称)值

功能：通过测点名称获取测点实时值

地址：WEB 地址/RTRest/ValueByName/点名称

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态"}]

}

8.1.4.3 测点(ID_值及报警

功能：通过测点 ID 获取测点实时值、报警值

地址：WEB 地址/RTRest/ValueWithAlarmByID/点 ID

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"}]

}

8.1.4.4 测点(名称)值及报警

功能：通过测点名称获取测点实时值、报警值

地址：WEB 地址/RTRest/ValueWithAlarmByName/点名称

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"}]

}

8.1.4.5 批量测点(ID)值

功能：发送点 ID 集合，获取测点的当前值

地址：WEB 地址/RTRest/ValuesByID

HTTP Method：post

请求参数：

```
{
    "list":[id1,id2...]
}
```

返回结果：

```
{
    通用数据,
    "data":[{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态"},{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态"}...]
}
```

8.1.4.6 批量测点(名称)值

功能：发送点名称集合，获取测点的当前值

地址：WEB 地址/RTRest/ValuesByName

HTTP Method：post

请求参数：

```
{
    "list":["名称 1","名称 2"...]
}
```

返回结果：

```
{
    通用数据,
    "data":[{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态"},{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态"}...]
}
```

8.1.4.7 批量测点(ID)值及报警

功能：发送点 ID 集合，获取测点的当前值、报警值

地址：WEB 地址/RTRest/ValuesWithAlarmByID

HTTP Method：post

请求参数：

{
"list":[id1,id2...]
}

返回结果：

{
通用数据,
"data":[{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"},{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"}...]
}

8.1.4.8 批量测点(名称)值及报警

功能：发送点名称集合，获取测点的当前值、报警值

地址：WEB 地址/RTRest/ValuesWithAlarmByName

HTTP Method：post

请求参数：

{
"list":["名称 1","名称 2"...]
}

返回结果：

{
通用数据,
"data":[{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"},{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"}...]
}

8.1.4.9 测点(ID)值订阅

功能：发送点 ID 集合，返回订阅号，订阅号用来查询订阅点集合的实时信息。

地址：WEB 地址/RTRest/SubsValuesByID

HTTP Method：post

请求参数：

{
"list":[id1,id2...]
}

返回结果：

```
{
    通用数据,
    "data": "订阅号"
}
```

8.1.4.10 测点(名称)值订阅

功能：发送点名称集合，返回订阅号，订阅号用来查询订阅点集合的实时信息。

地址：WEB 地址/RTRest/SubsValuesByName

HTTP Method：post

请求参数：

```
{
    "list":["名称 1","名称 2"...]
}
```

返回结果：

```
{
    通用数据,
    "data": "订阅号"
}
```

8.1.4.11 查询订阅测点(ID)值

功能：发送订阅号获取订阅测点的 ID、当前值。

地址：WEB 地址/RTRest/ValuesWithIDBySubs/订阅号

HTTP Method：get

返回结果：

```
{
    通用数据,
    "data":[{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态"},{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态"}...]
}
```

8.1.4.12 查询订阅测点(名称)值

功能：发送订阅号获取订阅测点的名称、当前值。

地址：WEB 地址/RTRest/ValuesWithNameBySubs/订阅号

HTTP Method：get

返回结果：

```
{
```

通用数据,

"data":[{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态"},{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态"}...]

}

8.1.4.13 查询订阅测点(ID)值及报警

功能：发送订阅号获取订阅测点的 ID、当前值、报警值

地址：WEB 地址/RTRest/ValuesWithIDAlarmBySubs/订阅号

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"},{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"}...]

}

8.1.4.14 查询订阅测点(名称)值及报警

功能：发送订阅号获取订阅测点的名称、当前值、报警值

地址：WEB 地址/RTRest/ValuesWithNameAlarmBySubs/订阅号

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"},{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"}...]

}

8.1.4.15 测点(ID)值变化订阅

功能：发送点 ID 集合，返回订阅号，订阅号用来查询订阅点集合的值变化；在获取变化值后，如点值无变化，再次获取，则返回空值；如长时间未获取变化值，值发生多次变化，查询时系统提供的值为最后一次变化值。

地址：WEB 地址/RTRest/SubsValuesChangeByID

HTTP Method：post

请求参数：

{

"list":[id1,id2...]

}

返回结果：

{

通用数据,

"data": "订阅号"

}

8.1.4.16 测点(名称)值变化订阅

功能：发送点名称集合，返回订阅号，订阅号用来查询订阅点集合的值变化；在获取变化值后，如点值无变化，再次获取，则返回空值；如长时间未获取变化值，值发生多次变化，查询时系统提供的值为最后一次变化值。

地址：WEB 地址/RTRest/SubsValuesChangeByName

HTTP Method：post

请求参数：

{

"list":["名称 1","名称 2"...]

}

返回结果：

{

通用数据,

"data": "订阅号"

}

8.1.4.17 查询订阅测点(ID)变化值

功能：发送订阅号获取订阅测点的 ID、当前值。在获取变化值后，如点值无变化，再次获取，则返回空值；如长时间未获取变化值，值发生多次变化，查询时系统提供的值为最后一次变化值。

地址：WEB 地址/RTRest/ValuesChangeWithIDBySubs/订阅号

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态"},{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态"}...]

}

8.1.4.18 查询订阅测点(名称)变化值

功能：发送订阅号获取订阅测点的名称、当前值。在获取变化值后，如点值无变化，再次获取，则返回空值；如长时间未获取变化值，值发生多次变化，查询时系统提供的值为最后一次变化值。

地址：WEB 地址/RTRest/ValuesChangeWithNameBySubs/订阅号

HTTP Method：get

返回结果：

{

通用数据,

"data":[{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态"},{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态"}...]

}

8.1.4.19 点值强制

功能：输入点名称(或点 ID)、点值，强制该点的值为输入值，

地址：WEB 地址/RTRest/Force/名称(或点 ID)? val=value

HTTP Method：get

返回结果：

{

通用数据

}

8.1.4.20 取消测点值强制

功能：取消点名称(或点 ID)的强制值，恢复正常值

地址：WEB 地址/RTRest/Unforce/名称(或点 ID)

HTTP Method：get

返回结果：

{

通用数据

}

8.1.4.21 取消所有测点值强制

功能：取消所有已强制的测点，恢复到正常值

地址：WEB 地址/RTRest/UnforceAll

HTTP Method：get

返回结果：

{

通用数据

}

8.1.4.22 添加测点(ID)实时值

功能：向系统增加测点实时值。一次添加请求，一个测点只能添加一个值。

地址：WEB 地址/RTRest/InsertRTDataByID

HTTP Method：post

请求参数，增加实时值为集合：

[

{"p":"点 ID","v":"点值" ,"s":"数据状态"},{"p":"点 ID ","v":"点值","s":"数据状态"}...

]

返回结果：

{

通用数据,

}

8.1.4.23 添加测点(名称)实时值

功能：向系统增加测点实时值。一次添加请求，一个测点只能添加一个值。

地址：WEB 地址/RTRest/InsertRTDataByName

HTTP Method：post

请求参数，增加实时值为集合：

[

{"p":"点名称","v":"点值" ,"s":"数据状态"},{"p":"点名称","v":"点值","s":"数据状态"}...

]

返回结果：

{

通用数据,

}

8.1.4.24 设备数据写入

功能：输入点名称(或点 ID)、点值，将数据写入到设备中，实现对设备控制。

地址：WEB 地址/RTRest/WriteDevData

HTTP Method：post

设备数据写入需具备

1. 访问许可：

必须提供具有控制权限的许可。http 请求的 headers 中，许可的 key 为“sn”

2. 采集服务 VDC 与数据服务 VDB 通讯方式

必须采用 TCP 通讯方式。

请求参数：

{"p":"点名称/点 ID","v":"点值"}

返回结果：

{

通用数据,

}

8.1.5 报警接口

8.1.5.1 报警点(ID)值变化订阅

功能：发送点 ID 集合，返回订阅号，订阅号用来查询订阅点集合的报警值变化；在获取变化报警值后，如点报警值无变化，再次获取，则返回空值；如长时间未获取报警值，值发生多次变化，查询时系统提供的值为最后一次变化值。

地址：WEB 地址/AlarmRest/SubsAlarmChangeByID

HTTP Method：post

请求参数：

{

"list":[id1,id2...]

}

返回结果：

{

通用数据,

"data": "订阅号"

}

8.1.5.2 报警点(名称)值变化订阅

功能：发送点名称集合，返回订阅号，订阅号用来查询订阅点集合的报警值变化；在获取变化报警值后，如点报警值无变化，再次获取，则返回空值；如长时间未获取报警值，值发生多次变化，查询时系统提供的值为最后一次变化值。

地址：WEB 地址/AlarmRest/SubsAlarmChangeByName

HTTP Method：post

请求参数：

{
"list":["名称 1","名称 2"...]
}
返回结果：
{
通用数据,
"data": "订阅号"
}

8.1.5.3 查询订阅测点(ID)变化报警值

功能：发送订阅号获取订阅测点的 ID、报警值。在获取变化报警值后，如点报警值无变化，再次获取，则返回空值；如长时间未获取变化值，报警值发生多次变化，查询时系统提供的值为最后一次变化值。

地址：WEB 地址/AlarmRest/AlarmChangeWithIDBySubs/订阅号

HTTP Method：get

返回结果：

{
通用数据,
"data":[{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"},{"p":"点ID","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"}...]
}

8.1.5.4 查询订阅测点(名称)变化报警值

功能：发送订阅号获取订阅测点的名称、报警值。在获取变化报警值后，如点报警值无变化，再次获取，则返回空值；如长时间未获取变化值，报警值发生多次变化，查询时系统提供的值为最后一次变化值。

地址：WEB 地址/AlarmRest/AlarmChangeWithNameBySubs/订阅号

HTTP Method：get

返回结果：

{
通用数据,
"data":[{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"},{"p":"名称","v":"点值","t":"此值在系统生效时间","s":"数据状态","al":"报警等级","at":"报警时间"}...]
}

8.1.5.5 获取各报警等级的报警点总数

功能：获取系统中当前各报警等级的报警点总数。

地址：WEB 地址/AlarmRest/AlarmLevelCount

HTTP Method：get

返回结果：

```
{
    通用数据,
    "data": {
        "1": count,
        "2": count,
        "3": count,
        "4": count,
        "5": count,
        "6": count,
        "7": count,
        "8": count,
        "9": count,
        "10": count
    }
}
```

8.1.6 历史数据接口

8.1.6.1 测点(ID)时刻历史值

功能：获取批量测点(ID)，在某些时刻的历史值。返回值中，测点历史值列表 list 与 times 一一对应

地址：WEB 地址/HISRest/MomentValueByID

HTTP Method：post

请求参数：

```
{
    "list":[id1,id2...],
    "times":[time1, time12...]
}
```

返回结果：

```
{
    通用数据,
```

"times":[time1, time12...],
"data":[{"p":"点ID","list":[{"v":"点值" ,"s":"数据状态"},{"v":"点值" ,"s":"数据状态"}]},{"p":"点ID","list":[{"v":"点值" ,"s":"数据状态"},{"v":"点值" ,"s":"数据状态"}]}...]
}

8.1.6.2 测点(名称)时刻历史值

功能：获取批量测点(名称)，在某些时刻的历史值。返回值中，测点历史值列表 list 与 times 一一对应

地址：WEB 地址/HISRest/MomentValueByName

HTTP Method：post

请求参数：

{
"list":["名称 1","名称 2"...],
"times":[time1, time12...]
}

返回结果：

{
通用数据,
"times":[time1, time12...],
"data":[{"p":"点名称","list":[{"v":"点值" ,"s":"数据状态"},{"v":"点值" ,"s":"数据状态"}]},{"p":"点名称","list":[{"v":"点值" ,"s":"数据状态"},{"v":"点值" ,"s":"数据状态"}]}...]
}

8.1.6.3 测点(ID)历史插值

功能：在开始时间到结束时间范围内，批量测点(ID)获取取样个数对应时刻的历史值。

取样个数 X 测点数<=5000000;从实时数据存储为历史数据,需一定时间，查询近几分钟内历史数据，无结果；返回值中，测点历史值列表 list 与 times 一一对应

地址：WEB 地址/HISRest/InterpolationValueByID

HTTP Method：post

请求参数：

{
"starttime":"开始时间",
"endtime":"结束时间",
"count":取样数,

"list":[id1,id2...]

}

返回结果：

{

通用数据,

"times":[time1, time12...],

"data":[{"p":"点ID","list":[{"v":"点值" ,"s":"数据状态"},{"v":"点值" ,"s":"数据状态"}]},{"p":"点ID","list":[{"v":"点值" ,"s":"数据状态"},{"v":"点值" ,"s":"数据状态"}]}...]

}

8.1.6.4 测点(名称)历史插值

功能：在开始时间到结束时间范围内，批量测点(名称)获取取样个数对应时刻的历史值。

取样个数 X 测点数<=5000000;从实时数据存储为历史数据,需一定时间，查询近几分钟内历史数据，无结果；返回值中，测点历史值列表 list 与 times 一一对应

地址：WEB 地址/HISRest/InterpolationValueByName

HTTP Method：post

请求参数：

{

"starttime":"开始时间",

"endtime":"结束时间",

"count":取样数,

"list":["名称 1","名称 2"...]

}

返回结果：

{

通用数据,

"times":[time1, time12...],

"data":[{"p":"点名称","list":[{"v":"点值" ,"s":"数据状态"},{"v":"点值" ,"s":"数据状态"}]},{"p":"点名称","list":[{"v":"点值" ,"s":"数据状态"},{"v":"点值" ,"s":"数据状态"}]}...]

}

8.1.6.5 测点(ID)历史原始值

功能：在开始时间到结束时间范围内，批量测点(ID)获取在系统存储的原始值。

当获取数据超过 5000000 条时，系统停止继续查询，返回已查到的原始数据。

地址：WEB 地址/HISRest/RawValueByID

HTTP Method：post

请求参数：

```
{
    "starttime":"开始时间",
    "endtime":"结束时间",
    "list":[id1,id2...]
}
```

返回结果：

```
{
    通用数据,
    "data":[{"p":"点ID","list":[{"v":"点值", "t":"时间","s":"数据状态"},{"v":"点值" , "t":"时间","s":"数据状态"}]},{"p":"点ID","list":[{"v":"点值" , "t":"时间","s":"数据状态"},{"v":"点值", "t":"时间","s":"数据状态"}]}...]
}
```

8.1.6.6 测点(名称)历史原始值

功能：在开始时间到结束时间范围内，批量测点(名称获取在系统存储的原始值。

当获取数据超过 5000000 条时，系统停止继续查询，返回已查到的原始数据。

地址：WEB 地址/HISRest/RawValueByName

HTTP Method：post

请求参数：

```
{
    "starttime":"开始时间",
    "endtime":"结束时间",
    "list":["名称 1","名称 2"...]
}
```

返回结果：

```
{
    通用数据,
    "data":[{"p":"点名称","list":[{"v":"点值", "t":"时间" ,"s":"数据状态"},{"v":"点值", "t":"时间" ,"s":"数据状态"}]},{"p":"点名称","list":[{"v":"点值", "t":"时间" ,"s":"数据状
```

态"},{"v":"点值", "t":"时间" ,"s":"数据状态"}]}...]
}

8.1.6.7 测点(ID)聚合查询

功能：在开始时间到结束时间范围内，批量测点(ID)获取在系统的聚合值。聚合计算功能列表如下

- 最大值

code：max

适用数据类型：Int、Uint、Short、Ushort、Long、Ulong、Float、Double、Byte、SByte

功能说明：时间范围内的最大值。

请求参数：

- 最小值

code：min

适用数据类型：Int、Uint、Short、Ushort、Long、Ulong、Float、Double、Byte、SByte

功能说明：时间范围内的最小值。

- 求和

code：sum

适用数据类型：Int、Uint、Short、Ushort、Long、Ulong、Float、Double、Byte、SByte

功能说明：时间范围内的数据之和，当取样数(count)>2 时，按照取样点时刻值对应数据求和，如不填取样数(count)或取样数(count)<=2 时，计算时取系统存储的原始值求和。

- 平均值

code：avg

适用数据类型：Int、Uint、Short、Ushort、Long、Ulong、Float、Double、Byte、SByte

功能说明：时间范围内的平均值。当取样数(count)>2 时，按照取样点时刻值对应数据求平均值，如不填取样数(count)或取样数(count)<=2 时，计算时取系统存储的原始值求平均值。

- 方差

code：variance

适用数据类型：Int、Uint、Short、Ushort、Long、Ulong、Float、Double、

Byte、SByte

功能说明：时间范围内的方差。当取样数(count)>2 时，按照取样点时刻值对应数据求方差，如不填取样数(count)或取样数(count)<=2 时，计算时取系统存储的原始值方差。

- 增量

code：increment

适用数据类型：Int、Uint、Short、Ushort、Long、Ulong、Float、Double、Byte、SByte

功能说明：结束时刻的值与开始时刻的差值。

- 变位记数

code：changecount

适用数据类型：Bool

功能说明：时间范围数据变化次数。

请求参数

地址：WEB 地址/HISRest/AggregateByID

HTTP Method：post

请求参数：

```
{
    "starttime":"开始时间",
    "endtime":"结束时间",
    "func":"聚合 code",
    "count":"取样个数",//求和、平均值、方差三项选填，其他计算不填
    "list":[id1,id2...]
}
```

返回结果：

```
{
    通用数据,
    "data":[{"p":"点ID","v":"计算值"},{"p":"点ID","v":"计算值"}...]
}
```

8.1.6.8 测点(名称)聚合查询

功能：在开始时间到结束时间范围内，批量测点(名称)获取在系统的聚合值。聚合计算功能列表见 8.1.6.7

请求参数

地址：WEB 地址/HISRest/AggregateByName

HTTP Method：post

请求参数：

```
{
    "starttime":"开始时间",
    "endtime":"结束时间",
    "func":"聚合 code",
    "count":"取样个数",//求和、平均值、方差三项选填，其他计算不填
    "list":["名称 1","名称 2"...]
}
```

返回结果：

```
{
    通用数据,
    "data":[{"p":"点名称","v":"计算值"},{"p":"点名称","v":"计算值"}...]
}
```

8.1.6.9 测点(ID)持续时间值

功能：获取批量测点(ID)，在请求时间段内，指定测点的历史值(范围值)持续时间长度。

使用此功能时建议：当点类型为模拟量时，尽量使用范围值如 ge、le、between，而少用 equal。当点类型为 Bool 时，用 equal。如对模拟量要求使用 equal 方式，建议模拟量应该是整数，并且值变化频率相对较低。

本功能支持的数据类型为模拟量和开关量，不支持 Char、String、Bytes、Date 类型。

地址：WEB 地址/HISRest/DurationByID

HTTP Method：post

请求参数：

```
{
    "starttime":"开始时间",
    "endtime":"结束时间",
    "list":[
            {"p":"点 ID","v1":值 1,"v2":值 2,"cmp":"between"},
            {"p":"点 ID","v1":值 1,"v2":值 2,"cmp":"ge"},
            ...
```

]

}

参数说明：

list：各个测点的符合查询历史值(范围值)和值对比条件的集合。

cmp：内容来自 8.4.6

当 cmp 不是 between 时，查询条件为：v1 与 cmp 的组合，如 v1 为 10，cmp 为"ge"，含义为：在"starttime":"开始时间"和"endtime":"查询"点"的历史值大于等于 10 的时间范围。此时 v2 值可不填。当点类型为 Bool 时，v1 填写：true、1、false、0；当类型为数字量时，v1 填写数字。当 cmp 为 between 时，点类型只能为数字量，v2 必填，查询范围为大于等于 v1 且小于等于 v2,即 v1<=点值<=v2。

返回结果：

```
{
    通用数据,
    "data":[{"p":"点ID","time": 持续时长(毫秒), "count": 时间段数, "detail": [
                        {"from": 1段开始1段开始, "to": 1段结束时间},
                        {"from": 2段开始1段开始, "to": 2段结束时间},
                        ...
                        ] },
    {"p":"点ID","time": 持续时长(毫秒), "count": 时间段数, "detail": [
                        {"from": 1段开始1段开始, "to": 1段结束时间},
                        {"from": 2段开始1段开始, "to": 2段结束时间},
                        ...
                        ] }
                        ...]
}
```

8.1.6.10 测点(名称)持续时间值

功能：获取批量测点(名称)，在请求时间段内，指定测点的历史值(范围值)持续时间长度。

使用此功能时建议：当点类型为模拟量时，尽量使用范围值如 ge、le、between，而少用 equal。当点类型为 Bool 时，用 equal。

本功能支持的数据类型为模拟量和开关量，不支持 Char、String、Bytes、Date 类型。

地址：WEB 地址/HISRest/DurationByName

HTTP Method：post

请求参数：

```
{
    "starttime":"开始时间",
    "endtime":"结束时间",
    "list":[
            {"p":"点名称","v1":值 1,"v2":值 2,"cmp":"between"},
            {"p":"点名称","v1":值 1, "cmp":"ge"},
            …
        ]
}
```

参数说明：

list：各个测点的符合查询历史值(范围值)和值对比条件的集合。

cmp：内容来自 8.4.6

当 cmp 不是 between 时，查询条件为：v1 与 cmp 的组合，如 v1 为 10，cmp 为"ge"，含义为：在"starttime":"开始时间"和"endtime":"结束时间","查询"点"的历史值大于等于 10 的时间范围。此时 v2 值可不填。当点类型为 Bool 时，v1 填写：true、1、false、0；当类型为数字量时，v1 填写数字。当 cmp 为 between 时，点类型只能为数字量，v2 必填，查询范围为大于等于 v1 且小于等于 v2。

返回结果：

```
{
    通用数据,
    "data":[{"p":"点名称","time": 持续时长(毫秒), "count": 时间段数, "detail": [
                    {"from": 1段开始1段开始, "to": 1段结束时间},
                    {"from": 2段开始1段开始, "to": 2段结束时间},
                    …
                    ] },
    {"p":"点名称","time": 持续时长(毫秒), "count": 时间段数, "detail": [
                    {"from": 1段开始1段开始, "to": 1段结束时间},
                    {"from": 2段开始1段开始, "to": 2段结束时间},
                    …
                    ] }
                    …]
}
```

8.1.6.11 添加测点(ID)历史值

功能：向系统增加测点历史值。

添加测点历史数据的时间范围只能为当前天；添加时，如不写时间"t"属性或者其值为空时，默认为提交请求时的系统时间，也可自定义当前天任意时刻。

地址：WEB 地址/HISRest/InsertHISDataByID

HTTP Method：post

请求参数，增加历史值为集合：

[

{"p":"点 ID","v":"点值","t":"历史时间(可选项)" ,"s":"数据状态"},{"p":"点 ID ","v":"点值","t":"历史时间(可选项)","s":"数据状态"}...

]

返回结果：

{

通用数据,

}

8.1.6.12 添加测点(名称) 历史值

功能：向系统增加测点历史值。

添加测点历史数据的时间范围只能为当前天；添加时，如不写时间"t"属性或者其值为空时，默认为提交请求时的系统时间，也可自定义当前天任意时刻。

地址：WEB 地址/HISRest/InsertHISDataByName

HTTP Method：post

请求参数，增加历史值为集合：

[

{"p":"点名称","v":"点值","t":"历史时间(可选项)","s":"数据状态"},{"p":"点名称","v":"点值","t":"历史时间(可选项)","s":"数据状态"}...

]

返回结果：

{

通用数据,

}

8.2 SignalR 接口

8.2.1 SignalR 技术说明

8.2.1.1 Windows 版 SignalR

采用 ASP.NET SignalR，~~非 ASP.NET Core SignalR~~，调用本接口的客户端请注意选择对应框架。**接口采用 Hub 连接模式。**

支持 JavaScript 客户端、.NET C#客户端，参见：

https://docs.microsoft.com/zh-cn/aspnet/signalr/overview/getting-started/tutorial-getting-started-with-signalr

支持 Java 客户端、C++客户端，参见：

https://github.com/aspnet/SignalR/tree/master/clients

8.2.1.2 Linux 版 SignalR

采用 ASP.NET Core SignalR，~~非 ASP.NET SignalR~~，调用本接口的客户端请注意选择对应框架。**接口采用 Hub 连接模式。**

https://docs.microsoft.com/zh-cn/aspnet/core/signalr/introduction?view=aspnetcore-6.0

8.2.2 数据说明

本节内容 Hub 的方法所需输入参数与返回值的格式与 8.1 章节一致，**区别为 8.1 节采用 json 对象，本节采用是 json 对应的字符串**，如 JavaScript 调用：

输入参数：JSON.stringify(json)

返回值：json = jQuery.parseJSON(r)

8.2.3 访问许可

服务端启用许可验证时，需在指定查询字符串参数中录入{"sn"："许可号"}，如 JavaScript 代码：

```
var conn = $.hubConnection("http://127.0.0.1/vicdasapi");
conn.qs = { "sn": "许可号" };
```

如 C#代码：

```
var querystringData = new Dictionary<string, string>();
querystringData.Add("sn", "4D1696EEB0874FB3807211AC5A475594");
var hubConnection = new HubConnection("http://127.0.0.1/vicdasapi", querystringData);
```

注：上述代码为 Widows 版示例

8.2.4 Hub 列表

配置接口 Hub：CnfgHub

实时数据接口 Hub：RTHub

报警接口 Hub：AlarmHub

历史数据接口：HISHub

JavaScript 代码示例，其他语言接口请参见 8.2.1 提供开发说明的连接：

```
var conn = $.hubConnection("http://127.0.0.1/vicdasapi");
conn.qs = { "sn": "4D1696EEB0874FB3807211AC5A475594" };//服务端未启用许可，可不填写；许可号从服务器管理员获取
var hubProxy = conn.createHubProxy('HISHub');
conn.start().done(function ()
{
     console.log("success");
}).fail(function (f)
{
    console.log(f);
 });

 Var sp = new Object();
 sp.startTime = 开始时间;
 sp.endTime = 结束时间;
 设置sp的其他参数
 hubProxy.invoke('interpolationvaluebyid', JSON.stringify(sp)).done(function (r)
 {
     var obj = jQuery.parseJSON(r);
     if (obj.code == 1)
     {
         obj 为返回数据，执行相关数据处理
     }
 }).fail(function (error)
 {
        console.log(error);
 });
```

注：上述代码为 Widows 版示例

8.2.5 配置接口

8.2.5.1 测点列表

功能及参数：8.1.3.1

方法：string Points()

8.2.5.2 报警点列表

功能及参数：8.1.3.2

方法：string AlarmPoints()

8.2.6 实时数据接口

8.2.6.1 测点(ID)值

功能及参数：8.1.4.1

方法：string ValueByID(string id)

8.2.6.2 测点(名称)值

功能及参数： 8.1.4.2

方法：string ValueByName(string name)

8.2.6.3 测点(ID_值及报警

功能及参数：8.1.4.3

方法：string ValueWithAlarmByID(string id)

8.2.6.4 测点(名称)值及报警

功能及参数： 8.1.4.4

方法：string ValueWithAlarmByName(string name)

8.2.6.5 批量测点(ID)值

功能及参数： 8.1.4.5

方法：string ValuesByID(string data)

8.2.6.6 批量测点(名称)值

功能及参数： 8.1.4.6

方法：string ValuesByName(string data)

8.2.6.7 批量测点(ID)值及报警

功能及参数： 8.1.4.7

方法：string ValuesWithAlarmByID(string data)

8.2.6.8 批量测点(名称)值及报警

功能及参数： 8.1.4.8

方法：string ValuesWithAlarmByName(string data)

8.2.6.9 测点(ID)值订阅

功能及参数： 8.1.4.9

方法：string SubsValuesByID(string data)

8.2.6.10 测点(名称)值订阅

功能及参数： 8.1.4.10

方法：string SubsValuesByName(string data)

8.2.6.11 查询订阅测点(ID)值

功能及参数： 8.1.4.11

方法： string ValuesWithIDBySubs(string id)

8.2.6.12 查询订阅测点(名称)值

功能及参数： 8.1.4.12

方法：string ValuesWithNameBySubs(string id)

8.2.6.13 查询订阅测点(ID)值及报警

功能及参数： 8.1.4.13

方法：string ValuesWithIDAlarmBySubs(string id)

8.2.6.14 查询订阅测点(名称)值及报警

功能及参数： 8.1.4.14

方法：string ValuesWithNameAlarmBySubs(string id)

8.2.6.15 测点(ID)值变化订阅

功能及参数： 8.1.4.15

方法：string SubsValuesChangeByID(string data)

8.2.6.16 测点(名称)值变化订阅

功能及参数： 8.1.4.16

方法：string SubsValuesChangeByName(string data)

8.2.6.17 查询订阅测点(ID)变化值

功能及参数： 8.1.4.17

方法：string ValuesChangeWithIDBySubs(string id)

8.2.6.18 查询订阅测点(名称)变化值

功能及参数： 8.1.4.18

方法：string ValuesChangeWithNameBySubs(string id)

8.2.6.19 点值强制

功能及参数： 8.1.4.19

方法：string Force(string id, string val)

8.2.6.20 取消测点值强制

功能及参数： 8.1.4.20

方法：string Unforce(string id)

8.2.6.21 取消所有测点值强制

功能及参数： 8.1.4.21

方法：string UnforceAll(string id)

8.2.6.22 添加测点(ID)实时值

功能及参数： 8.1.4.22

方法：string InsertRTDataByID(string data)

8.2.6.23 添加测点(名称)实时值

功能及参数： 8.1.4.23

方法：string InsertRTDataByName(string data)

8.2.6.24 设备数据写入

功能及参数：8.1.4.24

地址：string WriteDevData(string data)

8.2.7 报警接口

8.2.7.1 报警点(ID)值变化订阅

功能及参数： 8.1.5.1

方法：string SubsAlarmChangeByID(string data)

8.2.7.2 报警点(名称)值变化订阅

功能及参数： 8.1.5.2

方法：string SubsAlarmChangeByName(string data)

8.2.7.3 查询订阅测点(ID)变化报警值

功能及参数： 8.1.5.3

方法：string AlarmChangeWithIDBySubs(string id)

8.2.7.4 查询订阅测点(名称)变化报警值

功能及参数： 8.1.5.4

方法：string AlarmChangeWithNameBySubs(string id)

8.2.7.5 获取各报警等级的报警点总数

功能及参数： 8.1.5.5

方法：string AlarmLevelCount()

8.2.8 历史数据接口

8.2.8.1 测点(ID)时刻历史值

功能及参数： 8.1.6.1

方法：string MomentValueByID(string data)

8.2.8.2 测点(名称)时刻历史值

功能及参数： 8.1.6.2

方法：string MomentValueByName(string data)

8.2.8.3 测点(ID)历史插值

功能及参数： 8.1.6.3

方法：string InterpolationValueByID(string data)

8.2.8.4 测点(名称)历史插值

功能及参数： 8.1.6.4

方法：string InterpolationValueByName(string data)

8.2.8.5 测点(ID)历史原始值

功能及参数： 8.1.6.5

方法：string RawValueByID(string data)

8.2.8.6 测点(名称)历史原始值

功能及参数： 8.1.6.6

方法：string RawValueByName(string data)

8.2.8.7 测点(ID)聚合查询

功能及参数： 8.1.6.7

方法：string InterpolationValueByID(string data)

8.2.8.8 测点(名称)聚合查询

功能及参数： 8.1.6.8

方法：string InterpolationValueByName(string data)

8.2.8.9 测点(ID)持续时间值

功能及参数： 8.1.6.9

方法：string DurationByID(string data)

8.2.8.10 测点(名称)持续时间值

功能及参数：8.1.6.10

方法：string DurationByName(string data)

8.2.8.11 添加测点(ID)历史值

功能及参数： 8.1.6.11

方法：string InsertHISDataByID(string data)

8.2.8.12 添加测点(名称) 历史值

功能及参数： 8.1.6.12

方法：string InsertHISDataByName(string data)

8.3 Socket 接口

8.3.1 数据结构

本系统数据采用小端模式。

发送与接收测点实时数据、历史数据使用相同的数据结构字节数组，系统为每个类型测点都定义了数据结构。

如涉及到测点报警值，数据结构增加 16 字节：0-7 字节报警时间、8-11 报警等级、12-15 为 0。

因各语言、操作系统对结构体转化为字节流有差别，发送数据最终以每种类型下“字节数组结构”结构描述为标准结构

8.3.1.1 Int 类型

```
// Int点值结构体,字节对齐用系统自动补齐
//长度16 + 16(报警)
struct Int
{
```

```
        // UTC，毫秒为单位
        long Time;


        // 值
        int Val;


        // 点状态
        byte Status;

        /* 如请求中不含报警信息，
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */

        // 报警时间
        long AlarmTime;


        // 报警等级
        int AlarmLevel;
        /*************************/
    }
```

对应字节数组结构

0-7 字节	8-11字节	12字节	13-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7字节	8-11字节	12字节	13-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.2 UInt 类型

```
    // UInt点值结构体
    //长度16 + 16(报警)
    struct UInt
    {

        // UTC，毫秒为单位
        long Time;


        // 值
        uint Val;


        // 点状态
        byte Status;

        /* 如请求中不含报警信息，
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */
```

```
        // 报警时间
        long AlarmTime;


        // 报警等级
        int AlarmLevel;
        /*************************/
    }
```

对应字节数组结构

0-7 字节	8-11字节	12字节	13-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7字节	8-11字节	12字节	13-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.3 Short 类型

```
    // Short点值结构体
    //长度16 + 16(报警)
    struct Short
    {

        // UTC，毫秒为单位
        long Time;


        // 值
        short Val;


        // 点状态
        byte Status;

        /* 如请求中不含报警信息,
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */

        // 报警时间
        long AlarmTime;


        // 报警等级
        int AlarmLevel;
        /*************************/
    }
```

对应字节数组结构

0-7 字节	8-9字节	10字节	11-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7字节	8-9字节	10字节	11-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.4 UShort 类型

```
// UShort点值结构体
//长度16 + 16(报警)
struct UShort
{

    // UTC，毫秒为单位
    long Time;


    // 值
    ushort Val;


    // 点状态
    byte Status;

    /* 如请求中不含报警信息,
     * 则数据结构中不含有AlarmTime、 AlarmLevel属性
     * */

    // 报警时间
    long AlarmTime;


    // 报警等级
    int AlarmLevel;
    /*************************/
}
```

对应字节数组结构

0-7 字节	8-9字节	10字节	11-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7字节	8-9字节	10字节	11-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.5 Long 类型

```
// Long点值结构体
//长度24 + 16(报警)
struct Long
{

    // UTC，毫秒为单位
    long Time;


    // 值
    long Val;
```

```
        // 点状态
        byte Status;

        /* 如请求中不含报警信息,
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */

        // 报警时间
        long AlarmTime;


        // 报警等级
        int AlarmLevel;
        /*************************/
    }
```

对应字节数组结构

0-7 字节	8-15字节	16字节	17-23字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7 字节	8-15字节	16字节	17-23字节	24-31字节	32-35字节	36-39字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.6 ULong 类型

```
    // ULong点值结构体
    //长度24 + 16(报警)
    struct ULong
    {

        // UTC，毫秒为单位
        long Time;


        // 值
        ulong Val;


        // 点状态
        byte Status;

        /* 如请求中不含报警信息,
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */

        // 报警时间
        long AlarmTime;


        // 报警等级
        int AlarmLevel;
        /*************************/
    }
```

对应字节数组结构

0-7 字节	8-15字节	16字节	17-23字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7 字节	8-15字节	16字节	17-23字节	24-31字节	32-35字节	36-39字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.7 Float 类型

```
// Float点值结构体
//长度16 + 16(报警)
struct Float
{

    // UTC，毫秒为单位
    long Time;


    // 值
    float Val;


    // 点状态
    byte Status;

    /* 如请求中不含报警信息,
     * 则数据结构中不含有AlarmTime、 AlarmLevel属性
     * */

    // 报警时间
    long AlarmTime;


    // 报警等级
    int AlarmLevel;
    /**************************/
}
```

对应字节数组结构

0-7 字节	8-11字节	12字节	13-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7字节	8-11字节	12字节	13-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.8 Double 类型

```
// Long点值结构体
//长度24 + 16(报警)
struct Double
{
```

```
        // UTC，毫秒为单位
        long Time;


        // 值
        double Val;


        // 点状态
        byte Status;

        /* 如请求中不含报警信息，
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */

        // 报警时间
        long AlarmTime;


        // 报警等级
        int AlarmLevel;
        /**************************/
    }
```

对应字节数组结构

0-7 字节	8-15字节	16字节	17-23字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7 字节	8-15字节	16字节	17-23字节	24-31字节	32-35字节	36-39字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.9 Bool 类型

```
    // Bool点值结构体
    //长度16 + 16(报警)
    struct Bool
    {

        // UTC，毫秒为单位
        long Time;

        // 值
        bool Val;

        // 点状态
        byte Status;

        /* 如请求中不含报警信息，
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */

        // 报警时间
        long AlarmTime;
```

```
    // 报警等级
    int AlarmLevel;
    /*************************/
}
```

对应字节数组结构

0-7 字节	8字节	9字节	10-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7 字节	8字节	9字节	10-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.10 Byte 类型

```
// Byte点值结构体
//长度16 + 16(报警)
struct Byte
{

    // UTC，毫秒为单位
    long Time;


    // 值
    byte Val;


    // 点状态
    byte Status;

    /* 如请求中不含报警信息,
     * 则数据结构中不含有AlarmTime、 AlarmLevel属性
     * */

    // 报警时间
    long AlarmTime;


    // 报警等级
    int AlarmLevel;
    /*************************/

}
```

对应字节数组结构

0-7 字节	8字节	9字节	10-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7 字节	8字节	9字节	10-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.11 SByte 类型

```
// SByte点值结构体
//长度16 + 16(报警)
struct SByte
{

    // UTC，毫秒为单位
    long Time;


    // 值
    sbyte Val;


    // 点状态
    byte Status;

    /* 如请求中不含报警信息,
     * 则数据结构中不含有AlarmTime、 AlarmLevel属性
     * */

    // 报警时间
    long AlarmTime;


    // 报警等级
    int AlarmLevel;
    /*************************/
}
```

对应字节数组结构

0-7 字节	8字节	9字节	10-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7 字节	8字节	9字节	10-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.12 Char 类型

```
// Char点值结构体
//长度16 + 16(报警)
struct Char
{

    // UTC，毫秒为单位
    long Time;

    // 值,1个汉字、字符，utf8编码

    byte[2] Val;
```

```
        // 点状态
        byte Status;

        /* 如请求中不含报警信息,
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */

        // 报警时间
        long AlarmTime;


        // 报警等级
        int AlarmLevel;
        /*************************/

    }
```

对应字节数组结构

0-7 字节	8-9字节	10字节	11-15字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7字节	8-9字节	10字节	11-15字节	16-23字节	24-27字节	28-31字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.13 Date 类型

```
    // Time点值结构体
    //长度24 + 16(报警)
    struct Date
    {

        // UTC, 毫秒为单位
        long Time;


        // 值,UTC

        long Val;

        // 点状态
        byte Status;

        /* 如请求中不含报警信息,
         * 则数据结构中不含有AlarmTime、 AlarmLevel属性
         * */

        // 报警时间
        long AlarmTime;


        // 报警等级
        int AlarmLevel;
        /*************************/
```

```
}
```

对应字节数组结构

0-7 字节	8-15字节	16字节	17-23字节
Time	Val	Status	占位

对应含有报警字节数组结构

0-7 字节	8-15字节	16字节	17-23字节	24-31字节	32-35字节	36-39字节
Time	Val	Status	占位	AlarmTime	AlarmLevel	占位

8.3.1.14 Bytes 类型

```
// Bytes点值结构体
//长度80 + 16(报警)
struct Bytes
{

    // UTC，毫秒为单位
    long Time;

    // 点状态
    byte Status;

    // 数据字节长度

    byte Length;

    // 值,长度64

    byte[64] Bytes;

    /* 如请求中不含报警信息,
     * 则数据结构中不含有AlarmTime、 AlarmLevel属性
     * */

    // 报警时间
    long AlarmTime;


    // 报警等级
    int AlarmLevel;
    /*****************************/

}
```

0-7 Time	8 Status	9 Length	16到*16+(Length-1)* 字节流	*16+Length* 到79 占位

对应字节数组结构

0-7 字节	8字节	9字节	16到*16+(Length-1)*字节	*16+Length* 到79字节
Time	Status	Length	Bytes	占位

对应含有报警字节数组结构

0-7 字节	8字节	9字节	16到*16+(Length-1)*字节	*16+Length* 到79字节	80-87字节	88-91字节	92-95字节
Time	Status	Length	Bytes	占位	AlarmTime	AlarmLevel	占位

8.3.1.15 String 类型

```
// String点值结构体
//长度80 + 16(报警)
struct String
{

    // UTC，毫秒为单位
    long Time;

    // 点状态
    byte Status;

    // 数据字节长度

    byte Length;

    // 值,长度64,utf8字节流

    byte[64] Str;

    /* 如请求中不含报警信息,
     * 则数据结构中不含有AlarmTime、 AlarmLevel属性
     * */

    // 报警时间
    long AlarmTime;


    // 报警等级
    int AlarmLevel;
    /*************************/
}
```

对应字节数组结构

0-7 字节	8字节	9字节	16到*16+(Length-1)*字节	*16+Length* 到79字节
Time	Status	Length	Bytes	占位

对应含有报警字节数组结构

0-7 字节	8字节	9字节	16到16+(Length-1)字节	16+Length 到79字节	80-87字节	88-91字节	92-95字节
Time	Status	Length	Bytes	占位	AlarmTime	AlarmLevel	占位

8.3.2 通讯协议

系统规定了通讯协议用于收发数据，如不按照协议发送数据，系统会断开连接或不予反馈，后续请求不会执行。

本节内容请求协议所需输入参数与返回值的格式与 8.1 章节一致，**区别为 8.1 节采用 json 对象，本节采用是对应的字节数据组。**

Socket 通讯系统不注明情况下，使用的是 tcp。

为实现网闸隔离发送数据，8.3.5.11 添加实时数据接口和 8.3.7.6 添加测点历史数据接口，也提供 UDP 协议。

8.3.2.1 客户端请求协议

Head 长度共：64	请求号	Int32　　请求号间后面章节描述	4
	Body 长度	Int32	4
	客户端时间	Long64　　UTC	8
	许可	Byte[32]　　utf8 编码	32
	备用(数据无)	Byte[16]	16
Body	具体请求参数	长度为 head 中 body 描述长度 参数内容见后面章节描述	

定义：

无扩展数据请求：客户端发送的请求只按要求填写 Head 数据，无 Body 数据即 Body 长度为 0。

后面功能章节描述请求协议时，不再描述请求协议头部，只描述请求 Body 结果，如 Body 无数据，则写无扩展数据请求

后面描述中默认 Head 数据已将数据填写完毕，已填写内容如下：

1、　按请求功能填写请求号；

2、　Body 长度：有 Body 则填写 Body 长度值，无 Body 填写 0；

3、　客户端 UTC 时间；

4、　填写许可，如服务端未开启验证可不填写。

8.3.2.2 服务端返回协议

Head 长度共:64	执行结果 Code	Int32　　见 8.5	4
	Body 长度	Int32	4
	系统内置数据	Byte[32]	32
	服务端执行请求时刻	Long64　　UTC	8
	备用(数据无)	Byte[16]	16
Body	具体返回数据	长度为 head 中 body 描述长度 具体返回数据见后面章节描述	

请注意：对返回数据处理时，需先判断 Code 结果，和 Body 长度后，根据

再决定是否有 body 体，是否需要解析 bod。

定义

无扩展数据返回：服务端返回的数据只有 Head，无 Body 数据即 Body 长度为 0。

8.3.2.3 请求号

500001：请求测点列表

500002：请求报警点列表

510001：请求批量测点当前实时值

510002：请求批量测点当前实时值、报警值

510003：注册测点实时数据订阅

510004：请求订阅组实时值

510005：请求订阅组实时值、报警值

510006：注册实时数据值变化订阅

510007：请求订阅组实时数据变化值

510008：强制点值

510009：解除点值

510010：解除所有强制

510011：插入实时数据

520001：注册报警点变化订阅

520002：请求订阅组报警点变化值

520003：请求各报警等级的报警点总数

530001：查询历史时刻值

530002：查询历史数据聚合值

530003：插值查询历史数据

530004：获取时段内系统内存储的历史值

530005：插入历史数据

8.3.3 访问许可

Uft8 编码将许可序列化为字节数组，该数组长度为 32，将数组写在请求头部 16-47 位置处。

8.3.4 配置接口

8.3.4.1 测点列表

功能及参数：8.1.3.1

请求号：500001

客户端请求：无扩展数据请求

服务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：

Body	一个点信息数据长度	Int32	4	不含本身(4)的长度	一个点的描述信息
	测点ID	Int32	4	长度和==一个点信息数据长度	
	测点类型	Int32	4		
	测点存储方式	Int32	4		
	测点名称长度	Int32	4		
	测点名称字节数组	utf8	Byte[]		
	测点描述长度	Int32	4		
	测点描述字节数组	utf8	Byte[]		
	.....				
	一个点信息数据长度	Int32	4	不含本身(4)的长度	一个点的描述信息
	测点ID	Int32	4	长度和==一个点信息数据长度	
	测点类型	Int32	4		
	测点存储方式	Int32	4		
	测点名称长度	Int32	4		
	测点名称字节数组	utf8	Byte[]		
	测点描述长度	Int32	4		
	测点描述字节数组	utf8	Byte[]		

8.3.4.2 报警点列表

功能及参数：8.1.3.2

请求号：500002

客户端请求：无扩展数据请求

服务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：

Body	一个报警点信息数据长度	Int32	4	不含本身(4)的长度	一个报警点的描述信
	报警点ID	Int32	4	长度和==一个报警点信息数据长度	
	报警点类型	Int32	4		

	报警点名称长度	Int32	4		息
	报警点名称字节数组	utf8	Byte[]		
	.....				
	一个报警点信息数据长度	Int32	4	不含本身(4)的长度	
	报警点ID	Int32	4	长度和==一个报警点信息数据长度	一个点的描述信息
	报警点类型	Int32	4		
	报警点名称长度	Int32	4		
	报警点名称字节数组	utf8	Byte[]		

8.3.5 实时数据接口

8.3.5.1 批量测点(ID)值

功能及参数： 8.1.4.5

请求号：510001

客户端请求：

Body	id1	Int32	4
	id2	Int32	4
	…		

服务端返回，执行失败：返回异常点/无扩展数据返回

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	id1	Int32	4
	id1对应的类型结构数据	见8.3.1	
	Id2	Int32	4
	Id2对应的类型结构数据	见8.3.1	
	…		

8.3.5.2 批量测点(ID)值及报警

功能及参数： 8.1.4.7

请求号：510002

客户端请求：

Body	id1	Int32	4
	id2	Int32	4
	…		

服务端返回，执行失败：返回异常点/无扩展数据返回

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	id1	Int32	4
	id1对应的类型结构数据，含报警	见8.3.1	
	Id2	Int32	4
	Id2对应的类型结构数据，含报警	见8.3.1	
	…		

8.3.5.3 测点(ID)值订阅

功能及参数： 8.1.4.9

请求号：510003

客户端请求：

Body	id1	Int32	4
	id2	Int32	4
	…		

服务端返回，执行失败：返回异常点/无扩展数据返回

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	Byte[16]	utf8编码string	16

8.3.5.4 查询订阅测点(ID)值

功能及参数： 8.1.4.11

请求号：510004

客户端请求：

Body	Byte[16]	utf8编码string	16

务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：

Body	id1	Int32	4
	id1对应的类型结构数据	见8.3.1	
	Id2	Int32	4
	Id2对应的类型结构数据	见8.3.1	
	…		

8.3.5.5 查询订阅测点(ID)值及报警

功能及参数： 8.1.4.13

请求号：510005

客户端请求：

Body	Byte[16]	utf8编码string	16

务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：

Body	id1	Int32	4
	id1对应的类型结构数据，含报警	见8.3.1	
	Id2	Int32	4
	Id2对应的类型结构数据，含报警	见8.3.1	
	…		

8.3.5.6 测点(ID)值变化订阅

功能及参数： 8.1.4.15

请求号：510006

客户端请求：

Body	id1	Int32	4
	id2	Int32	4
	…		

服务端返回，执行失败：返回异常点/无扩展数据返回

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	Byte[16]	utf8编码string	16

8.3.5.7 查询订阅测点(ID)变化值

功能及参数： 8.1.4.17

请求号：510007

客户端请求：

Body	Byte[16]	utf8编码string	16

务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：

Body	id1	Int32	4
	id1对应的类型结构数据	见8.3.1	
	Id2	Int32	4
	Id2对应的类型结构数据	见8.3.1	
	…		

8.3.5.8 点值强制

功能及参数： 8.1.4.19

请求号：510008

客户端请求：

Body	id	Int32	4
	value	utf8编码string	

value：对 id 强制的值，转字符串，然后 utf8 编码为字节数组

8.3.5.9 取消测点值强制

功能及参数： 8.1.4.20

请求号：510009

客户端请求：

Body	id	Int32	4

服务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：无扩展数据返回

8.3.5.10 取消所有测点值强制

功能及参数： 8.1.4.21

请求号：510010

客户端请求：无扩展数据请求

服务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：无扩展数据返回

8.3.5.11 添加测点(ID)实时值

功能及参数： 8.1.4.22

请求号：510011

客户端请求：

Body	id1	Int32	4
	id1对应的类型结构数据	见8.3.1	时间值不需填写
	Id2	Int32	4
	Id2对应的类型结构数据	见8.3.1	时间值不需填写
	…		

服务端返回，执行失败：返回异常点/无扩展数据返回

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：无扩展数据返回

本接口也提供 UDP 协议，发送请求与 TCP 请求数据格式相同，但无返回数据。

8.3.5.12 设备数据写入

功能及参数：8.1.4.24

请求号：510012

客户端请求：

Body	id	Int32	4
	id对应的类型结构数据	见8.3.1	时间、状态值不需填写

服务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：无扩展数据返回

8.3.6 报警接口

8.3.6.1 报警点(ID)值变化订阅

功能及参数： 8.1.5.1

请求号：520001

客户端请求：

Body	id1	Int32	4
	id2	Int32	4
	…		

服务端返回，执行失败：返回异常点/无扩展数据返回

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	Byte[16]	utf8编码string	16

8.3.6.2 查询订阅测点(ID)变化报警值

功能及参数： 8.1.5.3

请求号：520002

客户端请求：

Body	Byte[16]	utf8编码string	16

务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：

Body	id1	Int32	4
	id1对应的类型结构数据，含报警	见8.3.1	
	Id2	Int32	4
	Id2对应的类型结构数据，含报警	见8.3.1	
	…		

8.3.6.3 获取各报警等级的报警点总数

功能及参数： 8.1.5.5

请求号：520003

客户端请求：无扩展数据请求

服务端返回，执行失败：无扩展数据返回

服务端返回，执行成功：

Body	LevelID1	Int32	4
	count	Int32	4
	LevelID2	Int32	4
	count	Int32	4
	…		

8.3.7 历史数据接口

8.3.7.1 测点(ID)时刻历史值

功能及参数： 8.1.6.1

请求号：530001

客户端请求：

Body	时间个数	Int32	4
	测点个数	Int32	4
	时刻1	Int64	8
	时刻2	Int64	8
	…		
	id1	Int32	4
	id2	Int32	4
	…		

服务端返回，执行失败：1、无扩展数据返回

服务端返回，执行失败：2、30002

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行失败：2、code 102

Body	时刻6	Int64	8
	时刻8	Int64	8
	…		

服务端返回，执行成功：

Body	id1	Int32	4
	id1 count 历史时刻值的个数		
	id1 hisVal1 对应的类型结构数据	见8.3.1	
	id1 hisVal2 对应的类型结构数据	见8.3.1	4
	…		
	Id2	Int32	4
	Id2 count 历史时刻值的个数		
	Id2 hisVal1 对应的类型结构数据	见8.3.1	
	Id2 hisVal2 对应的类型结构数据	见8.3.1	4
	…		

每个测点下在各个时刻值，与请求中时刻值列表(从小到大排序后)一一对应

8.3.7.2 测点(ID)历史插值

功能及参数： 8.1.6.3

请求号：530003

客户端请求：

Body	开始时间	Int64	8
	结束时间	Int64	8
	取样个数	Int32	4
	测点个数	Int32	4
	id1	Int32	4
	id2	Int32	4
	…		

服务端返回，执行失败：1、无扩展数据返回

服务端返回，执行失败：2、code 30002

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	id1	Int32	4
	id1 count 历史插值的取样个数		
	id1 hisVal1 对应的类型结构数据	见8.3.1	
	id1 hisVal2 对应的类型结构数据	见8.3.1	4
	…		
	Id2	Int32	4
	Id2 count 历史插值的取样个数		
	Id2 hisVal1 对应的类型结构数据	见8.3.1	
	Id2 hisVal2 对应的类型结构数据	见8.3.1	4
	…		

每个测点下在历史时刻值的个数等于取样个数

8.3.7.3 测点(ID)历史原始值

功能及参数： 8.1.6.5

请求号：530004

客户端请求：

Body	开始时间	Int64	8
	技术时间	Int64	8
	测点个数	Int32	4
	id1	Int32	4
	id2	Int32	4
	…		

服务端返回，执行失败：1、无扩展数据返回

服务端返回，执行失败：2、code 30002

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	id1	Int32	4
	id1 count 历史原始值的个数		

	id1　hisVal1 对应的类型结构数据	见8.3.1	
	id1　hisVal2 对应的类型结构数据	见8.3.1	4
	…		
	Id2	Int32	4
	Id2 count 历史原始值的个数		
	Id2　hisVal1 对应的类型结构数据	见8.3.1	
	Id2　hisVal2 对应的类型结构数据	见8.3.1	4
	…		

8.3.7.4 测点(ID)聚合查询

功能及参数：　8.1.6.7

请求号：530002

客户端请求：

Body	开始时间	Int64	8
	结束时间	Int64	8
	聚合id	Int32 见8.4.5	4
	取样个数	Int32	4
	测点个数	Int32	4
	id1	Int32	4
	id2	Int32	4
	…		

请求Body中的数据根据请求的聚合进行填写，哪些值必须填写项，请参见8.1.6.7。无论哪种聚合请求，请求Body的数据结构按照列表中填写，不能删减。

服务端返回，执行失败：1、无扩展数据返回

服务端返回，执行失败：2、code 30002

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	id1	Int32	4
	聚合值	Double	8
	id1	Int32	4
	聚合值	Double	8
	…		

8.3.7.5 测点(名称)持续时间

功能及参数： 8.1.6.11

请求号：530006

客户端请求：

Body	开始时间	Int64	8
	结束时间	Int64	8
	id1	Int32	4
	比较id	Int32	4，见8.4.6
	v1	Double	4
	v2	Double	4
	id1	Int32	4
	比较id	Int32	4，见8.4.6
	v1	Double	4
	v2	Double	4
	…		

请求Body中的数据根据请求的聚合进行填写，哪些值必须填写项，请参见8.1.6.11。当点类型为Bool时，v1得值填写仍然填写Double，系统会判断当v1大于0时，判断值为true，小于等于0时为false。

服务端返回，执行失败：1、无扩展数据返回

服务端返回，执行失败：2、codeC10001、C10004、C10005、C30001、30002

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：

Body	查询点个数	Int32	4
	id1	Int32	4
	id1的段数总数	Int32	4
	id1的总时长，单位ms	Long	8
	id1段数1开始时间	Long	8
	id1段数1结束时间	Long	8
	id1段数2开始时间	Long	8
	id1段数2结束时间	Long	8
	….，id1段数时间范围		
	Id2	Int32	4

	id2的段数总数	Int32	4
	id2的总时长，单位ms	Long	8
	id2段数1开始时间	Long	8
	id2段数1结束时间	Long	8
	id2段数2开始时间	Long	8
	id2段数2结束时间	Long	8
	….，id1段数时间范围		
	id3、id4…….		

8.3.7.6 添加测点(ID)历史值

功能及参数： 8.1.6.11

请求号：530005

客户端请求：

Body	id1	Int32	4
	id1对应的类型结构数据	见8.3.1	时间值可不填写
	Id2	Int32	4
	Id2对应的类型结构数据	见8.3.1	时间值可不填写
	…		

服务端返回，执行失败：1、无扩展数据返回

服务端返回，执行失败：2、code 30002

Body	id7	Int32	4
	id9	Int32	4
	…		

服务端返回，执行成功：无扩展数据返回

本接口也提供 UDP 协议，发送请求与 TCP 请求数据格式相同，但无返回数据。

8.4 系统数据属性值

8.4.1 测点类型

```
enum type
{
    4字节
    Int = 1,

    4字节
```

```
    UInt = 2,

    2字节
    Short = 3,

    2字节
    UShort = 4,

    8字节
    Long = 5,

    8字节
    ULong = 6,

    4字节
    Float = 7,

    8字节
    Double = 8,

    Bool = 9,

    0-255
    Byte = 10,

    -128-127
    SByte = 11,

    2字节UTF8
    Char = 12,

    64字节
    Bytes = 13,

    64字节UTF8
    String = 14,

    UTC毫秒
    Date = 15
}
```

8.4.2 数据存储方式

```
enum type
{
    不保存
    None = 0,

    变化存储
    Change = 1,

    插值存储
    Compress = 2,

    全部存储
    All = 3
```

```
}
```

8.4.3 数据状态

```
enum type
{
    未知
    Unknown = 0,
    修改
    Modify = 1,
    计算值
    CalcValue = 10,
    强制值
    ForceValue = 11,
    优
    QualityGood = 192,
    不确定
    QualityUncertain = 193,
    差
    QualityBad = 194
},
```

上述值为系统内置状态，当通过 api 接口写入时，可自定义数据状态值，最大值为 255

8.4.4 报警等级

```
enum type
{
    不报警
    None = 0,
    高 4 报警
    H4 = 1,
    高 3 报警
    H3 = 2,
    高 2 报警
    H2 = 3,
    高 1 报警
    H1 = 4,
    低 1 报警
    L1 = 5,
    低 2 报警
    L2 = 6,
    低 3 报警
    L3 = 7,
    低 4 报警"
    L4 = 8,
    置 1 报警"
    TRUE = 9,
    置 0 报警
    FALSE = 10
}
```

8.4.5 聚合方法

enum aggregatefunc

```
{
    最大值
    max = 1,
    最小值
    min = 2,
    和
    sum = 3,
    平均值
    avg = 4,
    方差
    variance = 5,
    增量
    increment = 6,
    变位记数
    changecount = 7
}
```

8.4.6 持续时间比较方式

```
enum durationsymbol
{
    相等
    equal = 1,
    大于等于
    ge = 2,
    小于等于
    le = 3,
    范围内
    between = 4
}
```

8.5 返回 Code 值

```
Code： 0;
0Msg = "操作已完成，执行结果失败";

Code： 1;
1Msg = "成功";

C4 = 4;
C4Msg = "超时失败";

Code： 101;
101Msg = "输入参数格式不正确";

Code： 102;
102Msg = "输入时间参数格式不正确";

Code： 103;
103Msg = "开始时间小于结束时间";

Code： 104;
104Msg = "输入数据字节流长度不正度";

C105 = 105;
```

```
C105Msg = "缩小查询时间范围";

Code: 110;
110Msg = "聚合方法名称不正确";

Code: 201;
201Msg = "命令代码错误";

Code: 301;
301Msg = "请求需带许可(参数sn)";

Code: 302;
302Msg = "无效许可";

Code: 303;
303Msg = "无写入数据权限";

Code: 304;
304Msg = "无读取数据权限";

Code: 10001;
10001Msg = "点不存在";

Code: 10002;
10002Msg = "订阅信息不存在";

Code: 10003;
10003Msg = "报警点点不存在";

Code: 10004;
10004Msg = "点类型不符合此聚合查询";

Code: 10005;
10005Msg = "点重复";

Code: 11001;
11001Msg = "趋势组不存在";

Code: 11002;
11002Msg = "报警组不存在";

C18001 = 18001;
C18001Msg = "采集器中不存在测点";

C18101 = 18101;
C18101Msg = "未成功连接采集器";


Code: 30001;
30001Msg = "点时间不在正确范围内";

Code: 30002;
30002Msg = "历史点不存在";

Code: 30003;
30003Msg = "查询结果集行数超出系统限制";
```

Code： 99999;
99999Msg = ″异常信息″;