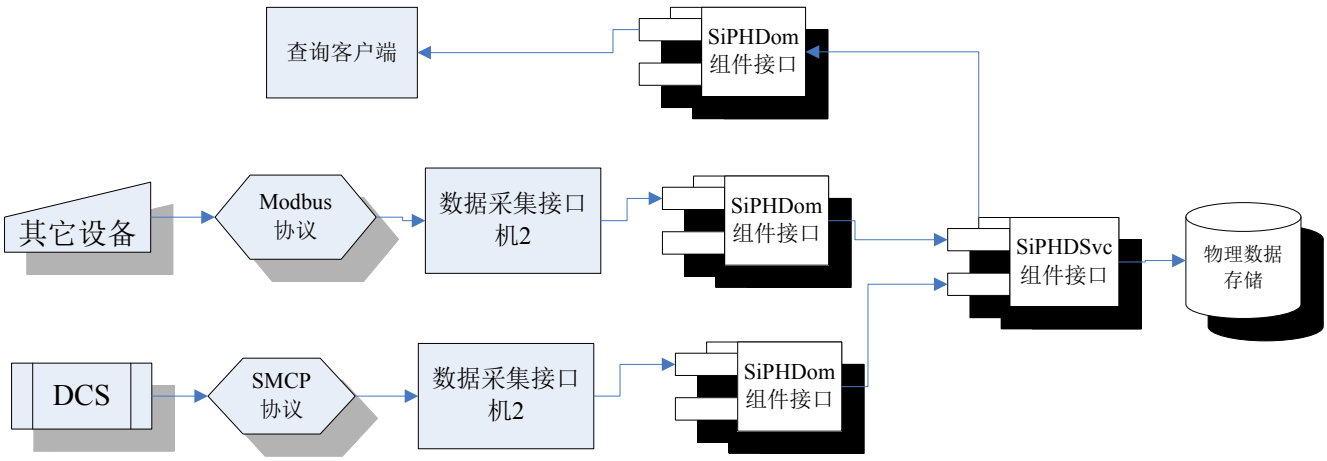
版本：1.0

# SIPHD 客户端开发流程(Java 版)

## 1、概述

SIPHD 实时历史数据库的服务器与客户端的通信机制如下图所示：



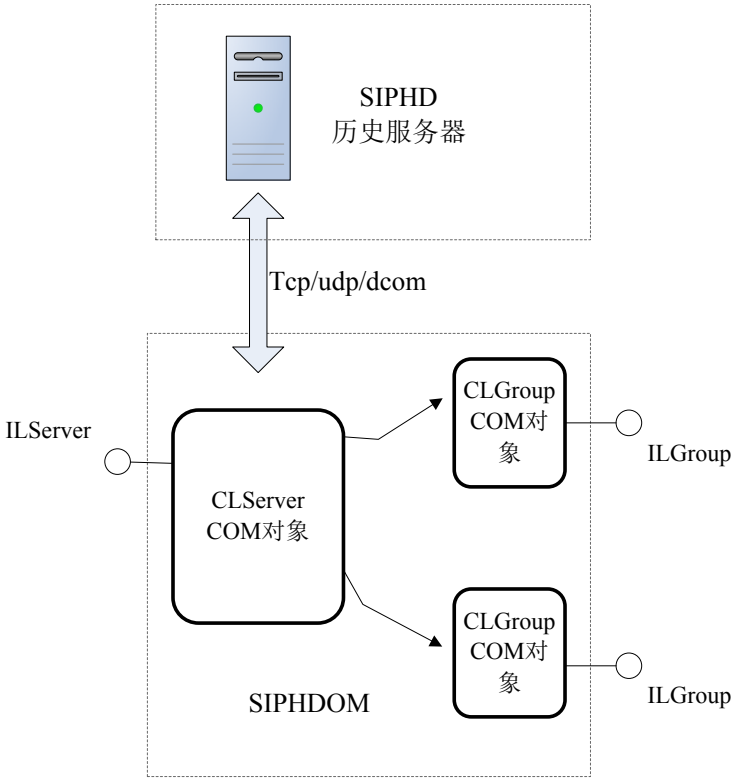

SIPHDOM 组件是 SIPHD 实时历史数据库提供的用于客户端通信的进程内 COM 组件，所有的客户端程序（如接口机，趋势，报表等）都通过该组件来间接访问历史服务器，该组件提供的主要功能包括：

- ✧ 标签点信息的查询
- ✧ 实时数据的存储、查询
- ✧ 历史数据的查询

SIPHDOM 组件内部结构示意图如下图所示，对外提供两种类型接口：

- ✧ ILServer 接口，历史服务器访问基本接口，提供与历史服务器的基本访问接口及服务器的连接、断开、标签点信息的查询等功能；
- ✧ ILGroup 接口，标签点组访问接口，提供对特定标签点集合的实时数据存储和查询等功能。

通过 ILServer 接口的标签点组管理功能，可以对标签点进行编组访问(例如对同一设备的标签点加入到同一个标签点组中)。




# 2.标签点类型及实时值数据格式

SIPHD历史数据库目前支持的标签点类型有：

- ✧ 模拟量
- ✧ 开关量
- ✧ 数字量

SIPHD历史数据库在网络上传输的实时数据结构为ILValue，其结构为：

```
ILValue
{
    __int64 m_tTime;          ///数据时间戳
    short m_nMillisecond;     ///毫秒值
    short m_nQAStatus;        ///质量位
    short m_nTVStatus;        ///状态位
    VARIANT m_Value;          ///数值
};
```

其中，质量位占2位物理空间，状态位有14位物理空间，状态位的数值含义定义如下表(见TVStatus枚举定义)：

| 值 | 数据状态含义 |
| --- | --- |
| 0 | TVStatus::tvsNoData，本时间点无数据 |

| | |
|---|---|
| 1 | TVStatus::tvsArchive，归档数据 |
| 2 | TVStatus::tvsInner，插值数据 |
| 3 | TVStatus::tvsOuter，外插数据 |
| 4 | TVStatus::tvsSnap，超时强存数据 |
| 5 | TVStatus::tvsStop，采集停止点数据 |
| 6 | TVStatus::tvsStart，采集起始点数据 |
| 7 | TVStatus::tvsCalc，计算点数据 |
| 8 | TVStatus::tvsDisCarded，丢弃数据 |

质量位枚举值：

| 值 | 数据状态含义 |
|---|---|
| 0 | qasGood，正常数据 |
| 1 | qasGood，坏点数据 |
| 2 | qasUnCertain，质量状态不确定 |

实时值存入 m_Value 时应视标签点的类型进行分别赋值：

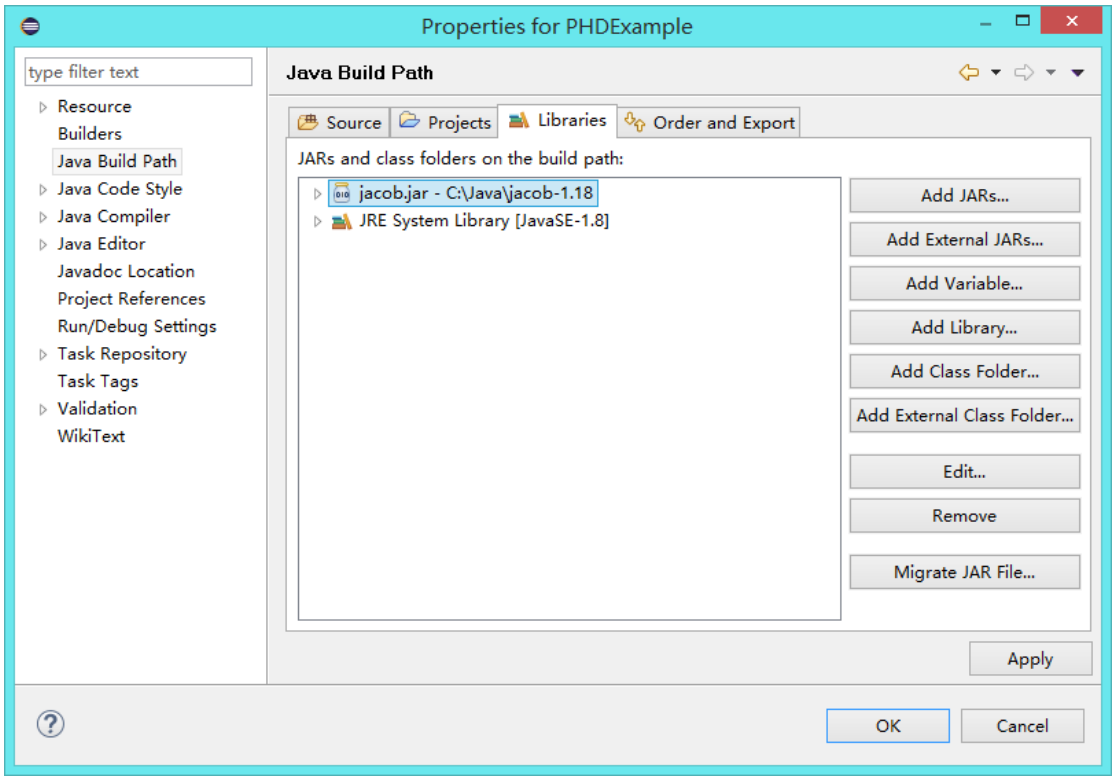
| 标签点类型 | 数值类型 |
|---|---|
| 模拟量点 | m_Value 数据类型 VT_R4; |
| 开关量点 | m_Value 数据类型应为 VT_BOOL; |
| 数字量点 | m_Value 数据类型 VT_I4; |

# 3、数据采集客户端开发流程

数据采集客户端是指定时将实时数据存储到数据库的客户端程序。
其开发的基本流程为：

## 3.1 向 Java 项目中添加 Jacob 库

向新建的的Java项目中添加Jacob库，如下图所示；




## 3.2 连接服务器

数据库通过接口 ILServer::Connect()(详细说明见 3.1.1)接口与指定服务器建立连接。

例如：

```
ILServer pServer = new ActiveXComponent("SiPHDOM.LServer");
Dispatch.call(pServer,"Connect",0, “192.162.1.10”, “test”, “adm”, “123”, 1666);
```

## 3.3 获取标签点信息

查询服务器数据库中的已创建标签点信息主要通过以下接口：

✧ CLServer::QueryTags()，SQL查询指定条件的标签点名称，例如，ID > 2 && ID < 1000等。

采集接口机的标签点信息可以通过上述两接口从服务器在线获取，亦可以从本地点表文件中获取，此时应保证文件中的标签点信息与数据库一致。

接口机标签点信息获取完成以后，通过ILServer接口创建若干个标签点组，并把标签点加入到相应的标签点组中。

代码示例：

```
String szCmd = "ID>0";
String[] tags = new String[0];
tags = Dispatch.call(pServer,    "QueryTags",szCmd);
```

## 3.5 创建标签点组

通过ILServer接口创建若干个标签点组，并把标签点加入到相应的标签点组中；
代码示例：

```
Dispatch pGroup= Dispatch.call(pServer,    "NewGroup",
“Read”).toDispatch();

iID1=Dispatch.call(pGroup, "NewItem","tag001").toInt();
iID2=Dispatch.call(pGroup, "NewItem","tag002").toInt();
```

## 3.6 实时采集设备数据，并实时递交至 SIPHDOM

对现场设备数据进行预处理成TripleItem数据格式，通过ILGroup:: SetTriple()接口对各标签点进行赋值，并定时通过IGroup::Update()接口递交至历史服务器。
代码示例：

```
long rt = Dispatch.call(pGroup, "GetIndex",0).toInt();
//java代码
Date dt=new Date();
long tt=Date2TimeT(dt);
double fValue=88.8f;
Dispatch.call(pGroup, "SetTripleAt",0, tt, 0, 0, 1, fValue);
fValue=77.7f;
Dispatch.call(pGroup, "SetTripleAt",1, tt, 0, 0, 1, fValue);
Dispatch.call(pGroup, "Update",new Variant(true));
```

## 3.7 连接属性设置

通过对CLServer的RetryNum和TimeOut属性的修改，可以设置SIPHDOM与历史服务器的故障重次数及连接超时时间。

## 3.8 通讯故障恢复功能

当接口机与服务器通讯发生故障后(调用 ILServer 和 ILGroup 接口时返回异常)，SIPHDOM 将周期进行网络状态的轮询与自动恢复，故障恢复后 SIPHDOM 自动恢复与服务器的连接。

## 3.9 与服务器断开连接

完成数据采集后，通过 ILServer::DisConnect()接口与服务器断开连接。
代码示例：

```
pServer.DisConnect();
```

# 4. 查询客户开发流程

查询客户端二次开发流程及接口说明：

## 4.1 向 C#项目中添加 SiPHDOM 引用

同3.1；

## 4.2 连接服务器

同3.2；

## 4.3 获取标签点信息

同3.3；

## 4.4 创建标签点组

同 4.4；

## 4.5 实时数据查询

按照一定的刷新周期调用ILGroup::Update(false)接口来实现标签组本地实时值缓存与服务器的同步，然后通过ILGroup:: GetValue ()接口来读取组内所有标签点的当前值。

代码示例：

```
Dispatch.call(pGroup, "Update",new Variant(false));

  //依次读实时数据
foreach(int i=0; i < nCount;i++)
{
Dispatch pValue=Dispatch.call(pGroup, "GetValue",iIndex).toDispatch();

    if(pValue!=null)
    {
        long tt=(long)Dispatch.get(pValue, "Time").toInt();
        short uQaulity = Dispatch.get(pValue, "Quality").toShort();
        short uStat = Dispatch.get(pValue, "Stat").toShort();
```

```
                double fValue = Dispatch.get(pValue, "Value").toDouble();
                fValue = fValue;
            }
        }
```

## 4.6 历史时刻值查询

通过设置Group组对象的Time时间来设置历史时刻，通过调用ILGroup::Update(false)接口来实现历史时刻值的服务器查询，然后通过ILGroup:: GetValue ()接口来读取组内所有标签点的查询结果值；

示例代码：

```
Date dt=new Date(115,9,28,11,38,30);
String szTime = dt.toString();
Dispatch.put(pGroup, "Time", dt);
Dispatch.call(pGroup, "Update",new Variant(false));

  //依次读实时数据
for(int i=0; i < nCount;i++)
{
    ILValue pValue = pGroup.GetValue(0);
    if(pValue!=null)
    {
        long tt=(long)Dispatch.get(pValue, "Time").toInt();
        short uQaulity = Dispatch.get(pValue, "Quality").toShort();
        short uStat = Dispatch.get(pValue, "Stat").toShort();
        double fValue = Dispatch.get(pValue, "Value").toDouble();
        fValue = fValue;
    }
}
```

## 4.6 历史时段值查询

通过ILServer:: GetHistValue ()接口可以同时查询标签点的某段时间的历史数据。

示例代码：

```
Date dtStart=new Date(115,8,28,11,39,0);
Date dtEnd = new Date(115,8,28,11,44,0);

Dispatch pList=Dispatch.call(pServer, "GetHistValue",m_iID1, dtStart,
dtEnd).toDispatch();

//依次读历史数据
int iCount=Dispatch.get(pList, "Count").toInt();
```

```
for(int i=0;i<iCount;i++)
{
    Dispatch pValue=Dispatch.call(m_pGroup, "GetValue",i).toDispatch();
    long tt=(long)Dispatch.get(pValue, "Time").toInt();
    short uQaulity = Dispatch.get(pValue, "Quality").toShort();
      short uStat = Dispatch.get(pValue, "Stat").toShort();
      double fValue = Dispatch.get(pValue, "Value").toDouble();
      fValue = fValue;

}
```

### 4.7 与服务器断开连接

完成查询后，通过 ILServer::DisConnect()接口与服务器断开连接。

# 5 ILServer 接口说明

## 5.1 ILServer::Connect

```
HRESULT Connect(
        [in] LConnectionType eType,
        [in] BSTR bstrHost,
        [in] BSTR bstrDBName,
        [in] BSTR bstrUserName,
        [in] BSTR bstrPassword,
        [in] VARIANT vParam);
```

**描述：** 与指定服务器建立连接；

| 参数 | 说明 |
|---|---|
| eType | 指定 siphdDom 与远方服务器的通信协议，<br>✧ CT_TCP：TCP 协议<br>✧ CT_DCOM：DCOM 方式<br>✧ CT_UDP：UDP 协议 |
| bstrDBName | PHD 服务器名字或 IP |
| bstrDBName | SIS 数据库名称 |
| bstrUserName | 该数据库的登录用户名 |
| bstrPassword | 数据库登录密码 |

| | |
|---|---|
| vParam | 连接参数，在 TCP 或 UDP 协议下指与服务器通信的端口号，默认为 1666。 |

返回值：

| 返回值 | 说明 |
|---|---|
| S_OK | 连接成功 |
| 其它 | 连接失败 |

## 5.2 ILServer::DisConnect

```
HRESULT DisConnect();
```

描述：

与服务器断开连接；

返回值：

| 返回值 | 说明 |
|---|---|
| S_OK | 断开成功 |
| 其它 | 断开失败 |

## 5.3 ILServer:: NewGroup

```
HRESULT NewGroup(
[in] BSTR bstrGrpName,
[out,retval] ILGroup ** ppVal);
```

描述： 新建标签点组.

| 参数 | 说明 |
|---|---|
| bstrGrpName | 标签组名称 |
| ILGroup | 新生成标签组对象的访问指针 |

返回值：

| 返回值 | 说明 |
|---|---|
| S_OK | 创建成功 |
| 其它 | 创建失败 |

## 5.4 ILServer:: RemoveGroup

```
HRESULT RemoveGroup (
[in] BSTR bstrGrpName);
```

**描述：**

删除指定标签点组.

| 参数 | 说明 |
| --- | --- |
| bstrGrpName | 标签组名称 |

**返回值：**

| 返回值 | 说明 |
| --- | --- |
| S_OK | 删除成功 |
| 其它 | 删除失败 |

## 5.5 ILServer:: GetHistValue

```
HRESULT GetHistValue([in] VARIANT ID,
[in] DATE Start,
[in] DATE End,
[out,retval]VARIANT* pts);
```

**描述：**

查询指定ID的一段历史时间内的所有归档历史数据.

| 参数 | 说明 |
| --- | --- |
| ID | 标签点 ID |
| Start | 起始时间 |
| End | 结束时间 |
| pts | 查询结果返回值，为 IValueList 对象 |

**返回值：**

| 返回值 | 说明 |
| --- | --- |
| S_OK | 递交成功 |
| 其它 | 递交失败 |

## 5.6 ILServer:: GetGroup

```
HRESULT GetGroup(
[in] BSTR bstrGrpName,
[out,retval] ILGroup ** ppVal);
```

**描述：**

查询指令标签组的访问指针.

| 参数 | 说明 |
| --- | --- |
| bstrGrpName | 待查询标签组名 |
| ppVal | 标签组对象访问指针 |

返回值：

| 返回值 | 说明 |
| --- | --- |
| S_OK | 查询成功 |
| 其它 | 查询成功 |

## 5.7 ILServer:: QueryTags

```
HRESULT QueryTag(
[in] BSTR bstrCmd,
[out,retval] VARIANT *pNames);
```

**描述：**

查询标签点信息.

| 参数 | 说明 |
| --- | --- |
| bstrCmd | 查询 SQL 语句 |
| pNames | 查询结果数据,字符串数组 |

**返回值：**

| 返回值 | 说明 |
| --- | --- |
| S_OK | 查询成功 |
| 其它 | 查询失败 |

# 6 ILGroup 接口说明

## 6.1 ILGroup::NewItem

```
HRESULT NewItem(
[in] BSTR bstrTag,
[out, retval] long *pnID);
```

**描述：** 向组内添加新标签点，若添加成功(数据库内存在该点)，则返回该标签点的ID。

| 参数 | 说明 |
|---|---|
| bstrTag | 标签点名称 |
| pnID | 查询成功时带回该标签点的 ID 值 |

返回值：

| 返回值 | 说明 |
|---|---|
| S_OK | 添加成功，数据中存在指定标签名 |
| 其它 | 添加成功，该标签点名不存在或通讯故障 |

## 6.2 ILGroup::NewItems

```
HRESULT NewItems(
[in] VARIANT Tags,
[out, retval] VARIANT* IDs);
```

**描述：**向标签点组同时添加多个标签点

| 参数 | 说明 |
|---|---|
| Tags | 标签点名称数组(VT_BSTR 类型的 SAFEArray 结构) |
| IDs | VT_UINT 类型的 SafeArray 结构；<br>若点存在则返回其 ID,若点不存在返回 ID=0xFFFFFFFF |

**返回值：**

| 返回值 | 说明 |
|---|---|
| S_OK | 添加成功 |
| 其它 | 添加失败 |

## 6.3 ILGroup::Clear

HRESULT Clear();
**描述：** 清空组内所有标签点
**返回值：**

| 返回值 | 说明 |
| --- | --- |
| S_OK | 清除成功 |
| 其它 | 清除失败 |

## 6.4 ILGroup::Delete

HRESULT Delete(void);
**描述：** 删除当前组对象
**返回值：**

| 返回值 | 说明 |
| --- | --- |
| S_OK | 删除成功 |
| 其它 | 删除失败 |

## 6.5 ILGroup::Update

HRESULT Update(
[in] VARIANT_BOOL bUpdateToServer);
**描述：** 与服务器进行一次实时数据同步，bUpdate为true，表示存储组内实时数据至服务器，false表示从服务器读取实时数据.

| 参数 | 说明 |
| --- | --- |
| bUpdateToServer | true，递交组内实时值至服务器，<br>false,从服务器读取实时值刷新组内实时值 |

返回值：

| 返回值 | 说明 |
| --- | --- |
| S_OK | 同步成功 |
| 其它 | 同步失败 |

## 6.6 ILGroup::SetTriple

HRESULT SetTriple(
[in] ULONG nID,
[in] LONGLONG lTime,

[in] SHORT nMS,
[in] SHORT nQAStatus,
[in] SHORT nTVStatus,
[in] VARIANT vValue);

**描述：** 刷新标签组内指定ID标签点的实时值，nID为标签点在数据库的全局唯一ID，对组内本地实时值缓存进行刷新；

| 参数 | 说明 |
| --- | --- |
| nID | 标签点 ID，数据库内全局唯一 |
| lTime | 数值时间戳 |
| nMS | 毫秒值 |
| nQSStatus | 数值质量位，2 位地址空间 |
| nTVStatus | 数值状态值，14 位地址空间 |
| vValue | 实时值 |

**返回值：**

| 返回值 | 说明 |
| --- | --- |
| S_OK | 刷新成功 |
| 其它 | 刷新失败 |

## 6.7 ILGroup::SetTripleAt

HRESULT SetTripleAt(
[in] ULONG nIndex,
[in] LONGLONG lTime,
[in] SHORT nMS,
[in] SHORT nQAStatus,
[in] SHORT nTVStatus,
[in] VARIANT vValue);

**描述：**刷新组内指定次序标签点的实时值，其中nIndex为标签点在组内缓存中的次序，对组内本地实时值缓存进行刷新。

| 参数 | 说明 |
| --- | --- |
| nIndex | 为标签点在标签组内缓存中的次序 |
| lTime | 数值时间戳 |
| nMS | 毫秒值 |
| nQSStatus | 数值质量位 |
| nTVStatus | 数值状态值 |
| vValue | 实时值 |

返回值：

| 返回值 | 说明 |
| --- | --- |
| S_OK | 刷新成功 |
| 其它 | 刷新失败 |

## 6.8 ILGroup::GetValue

```
HRESULT GetValue(
[in] ULONG nIndex,
[out,retval] ILValue* ppVal);
```

**描述：**取组内指定次序点的实时值，该当前值指组内本地缓存内的实时值。

| 参数 | 说明 |
| --- | --- |
| nIndex | 为标签点在标签组内缓存中的次序 |
| ppVal | 返回数值，为 ILValue 类型 |

返回值：

| 返回值 | 说明 |
| --- | --- |
| S_OK | 刷新成功 |
| 其它 | 刷新失败 |