**OPC客户端开发过程整理**

2014年06月26日 11:38:23 [greston](https://me.csdn.net/greston) 阅读数：19207更多

个人分类： [C/C++](https://blog.csdn.net/greston/article/category/558153)

2013-09-02 17:01:44|  分类： [window其它](http://dgj0600.blog.163.com/blog/#m=0&t=1&c=fks_084066082084083069080095086095080082081069084082085065) |  标签：[opc客户端](http://dgj0600.blog.163.com/blog/#m=0&t=3&c=opc%E5%AE%A2%E6%88%B7%E7%AB%AF)  [opc开发](http://dgj0600.blog.163.com/blog/#m=0&t=3&c=opc%E5%BC%80%E5%8F%91)   |举报 |字号 订阅

OPC开发过程:

1.注册OPC接口组件

2.通过COM接口创建OPCServerList的接口指针

CoCreateInstance只能遍历本机，CoCreateInstanceEx可遍历远程机器OPC

CLSID\_OPCServerList={0x13486D51, 0x4821, 0x11D2, { 0xA4, 0x94, 0x3C, 0xB3, 0x06, 0xC1, 0x00, 0x00 } };

IOPCServerList \*pIOPCServerList = NULL;

CoCreateInstance(CLSID\_OPCServerList, NULL, CLSCTX\_SERVER, IID\_IOPCServerList, (void \*\*)&pIOPCServerList);

3.枚举OPC服务列表

通过IOPCServerList接口EnumClassesOfCategories方法,示例:

unsigned long c;

IEnumGUID \*pIEnumGUID;

CLSID catid = CATID\_OPCDAServer20;

pIOPCServerList->EnumClassesOfCategories(1, &catid, 1, &catid, &pIEnumGUID);

while(S\_OK == pIEnumGUID->Next(1, &clsid, &c))

{

   char buf1[256]={0}, buf2[256]={0};

   LPOLESTR progID, pOPCName;

   pIOPCServerList->GetClassDetails(clsid, &progID, &pOPCName);

   WideCharToMultiByte(CP\_ACP, 0, progID, -1, buf1, 256, NULL, NULL);

   WideCharToMultiByte(CP\_ACP, 0, pOPCName, -1, buf2, 256, NULL, NULL);

   printf("OPC服务器: %s, %s\n", buf1, buf2);

}

3.连接OPC服务器

通过progID获取OPC服务器CLSID，利用CLSIDFromProgID()函数。

CoCreateInstanceEx启动远程OPC服务器，COSERVERINFO结构中的远程主机名称填IP和计算机全名称都可以，代码示例：

COSERVERINFO tCoServerInfo;

MULTI\_QI m\_arrMultiQI; // array of interfaces we will query for each server

m\_arrMultiQI.pIID = &IID\_IOPCServer;

IOPCServer \*m\_pIServer;

CoCreateInstanceEx (

clsid, // CLSID

NULL, // No aggregation

CLSCTX\_REMOTE\_SERVER, // connect to remote servers

&tCoServerInfo, // remote machine name

sizeof (m\_arrMultiQI) / sizeof (MULTI\_QI), // number of IIDS to query

&m\_arrMultiQI);

m\_pIServer = (IOPCServer \*)m\_arrMultiQI.pItf;

4.断开OPC服务器连接

调用COM组件的Release接口，示例代码：

m\_pIServer->Release();

5.遍历指定OPC服务上的Group

连接上指定OPC服务器后，可取得服务器状态,调用GetStatus()，结果存放在OPCSERVERSTATUS结构中。

遍历GROUP:

a.通过IOPCServer接口的CreateGroupEnumerator()创建IEnumUnknown以Group对象方式遍历Group,通过IEnumUnknown获取Group存取对像接口IOPCGroupStateMgt,

调用GetState返回此Group信息,示例代码:

IEnumUnknown \*pEnum = NULL;

hr = m\_pIServer->CreateGroupEnumerator (

eScope, IID\_IEnumUnknown, (IUnknown\*\*)&pEnum);

IUnknown \*pIUnknown;

ULONG uFetched;

IOPCGroupStateMgt \*pIGroupStateMgt = NULL;

// Loop over enumerated groups (call enemerator's next member to

// reference next group and return its IUnknown interface):

while ((hr = pEnum->Next (1, &pIUnknown, &uFetched)) == S\_OK)

{

// Get group state management interface of current group:

hr = pIUnknown->QueryInterface (IID\_IOPCGroupStateMgt, (void\*\*)&pIGroupStateMgt);

// Declare some variable needed to get the group name

// from the server:

OPCHANDLE hClient;

OPCHANDLE hServer;

WCHAR \*pszName;

// Get group state (we are only interested in the group name):

hr = pIGroupStateMgt->GetState (

&m\_dwUpdateRate,

&m\_bActive,

&pszName,

&m\_lBias,

&m\_fDeadband,

&m\_dwLanguageID,

&hClient,

&hServer);

}

b.通过IOPCServer接口的CreateGroupEnumerator()创建IEnumString以Group名方式遍历Group，示例代码：

IEnumString \*pEnum = NULL;

hr = m\_pIServer->CreateGroupEnumerator (

eScope, IID\_IEnumString, (IUnknown\*\*)&pEnum);

LPOLESTR pGroupName;

ULONG uFetched;

// Loop over enumerated groups (call enemerator's next member to

// reference next group and return its name string):

while ((hr = pEnum->Next (1, &pGroupName, &uFetched)) == S\_OK)

{

}

6.遍历服务器Item项

通过OPCServer获取IOPCBrowseServerAddressSpace接口，调用BrowseOPCItemIDs方法,示例代码：

IOPCBrowseServerAddressSpace \*pIOPCBrowseServerAddressSpace=NULL;

pIOPCServer->QueryInterface(IID\_IOPCBrowseServerAddressSpace, (void \*\*)&pIOPCBrowseServerAddressSpace);

LPENUMSTRING pPENUMSTRING;

hr = pIOPCBrowseServerAddressSpace->BrowseOPCItemIDs(OPC\_BRANCH, L"", VT\_EMPTY, 0, &pPENUMSTRING);

IOPCItemProperties \*pIOPCItemProperties=NULL;

pIOPCServer->QueryInterface(IID\_IOPCItemProperties, (void \*\*)&pIOPCItemProperties);

ULONG count=0;

LPOLESTR pItemName, pItemID;

while(pPENUMSTRING->Next(1, &pItemName, &count) == S\_OK)

{

char bufName[256]={0};

WideCharToMultiByte(CP\_ACP, 0, pItemName, -1, bufName, 256, NULL, NULL);

printf("ITEM\_NAME: %s\n", bufName);

pIOPCBrowseServerAddressSpace->GetItemID(pItemName, &pItemID);

WideCharToMultiByte(CP\_ACP, 0, pItemID, -1, bufName, 256, NULL, NULL);

printf("ITEM\_ID: %s\n", bufName);

DWORD dwCount;

DWORD \*dwPropertyIDs;

LPWSTR \*ppDescriptions;

VARTYPE \*ppvtDataTypes;

VARIANT \*ppvData;

HRESULT \*ppErrors;

pIOPCItemProperties->QueryAvailableProperties(pItemID, &dwCount, &dwPropertyIDs, &ppDescriptions, &ppvtDataTypes);

pIOPCItemProperties->GetItemProperties(pItemID, dwCount, dwPropertyIDs, &ppvData, &ppErrors);

}

7.数据项ITEM读取

通过组Group取得ITEM项存取接口(IOPCSyncIO,IID\_IOPCAsyncIO等),调用Write和Read方法。

8.总结

逻辑过程:

连接OPC服务器->得到服务器数据项->建立自定义组->将需要的服务器数据项加入到组中->取得项的存取接口->读写数据

**使用C++访问OPC Server的简单方法**

2017年08月04日 14:25:27 [luckyone906](https://me.csdn.net/u011555996) 阅读数：562更多

个人分类： [c#技术](https://blog.csdn.net/u011555996/article/category/6957700) [c++技术](https://blog.csdn.net/u011555996/article/category/6957701)

OPC的文档网上很多，我在这里要介绍的主题是使用C++通过自动化接口来访问OPC Server，写这篇文章的目的是我在网上没有搜索到这方面的文档，如果我有这方面的需要，我想在网上一定也有其他朋友有这个需要，希望能对这些朋友有一些帮助。  
使用C++来访问OPC Server, 相对于使用自定义接口来说，自动化接口要简单很多，因为这和Visual Basic使用的是同一个接口，使用过Visual Basic来访问OPC Server的朋友一定能有这个体会。首先是准备好开发环境，一般测试是在模拟环境中进行，这样比较保险，可以使用一些免费的模拟OPC Server。我这里准备的是Matrikon的模拟服务器，模拟器安装以后。编程环境是VC++ 6.0,使用200X和2010也都大同小异。  
为了演示简单，新建一个Win32控制台工程agOPC，新建agOPC.cpp源文件并加到工程里。  
// --------------------------------- agOPC.cpp -----------------------------------------------  
//在agOPC.cpp开头添加如下一行  
#import 'C:Program FilesMatrikonOPCCommonOPCAuto.dll' no\_namespace  
//这是通过OPCAuto.dll里所包含的类型库信息产生C++能访问的头文件，此时在工程的Debug文件夹下产生OPCAuto.tlh和OPCAuto.tli两个文件。  
//添加需要的头文件  
#pragma warning( disable : 4786 ) // 为了避免vector报出的C4786警告  
#include <comdef.h> // 使用到了\_bstr\_t，\_variant\_t，\_com\_error都在这个文件里定义  
#include <iostream>  
#include <vector>  
using namespace std;  
//声明全局变量  
typedef str uct OLEInit {  
OLEInit() { CoInitialize( NULL ); }  
~OLEInit() { CoUninitialize(); }  
} OLEInit;  
OLEInit oleInit; // 必须在最前面定义，因为在使用COM之前必须初始化COM库,否则程序会崩溃  
// 由于是全局变量oleInit的构造函数在所有对象的构造函数调用之前调用，  
// 析构函数在所有对象的析构函数调用之后调用  
IOPCAutoServerPtr opcSvr; // 这些智能指针类型在OPCAuto.tlh中定义  
IOPCGroupsPtr opcGrps;  
IOPCGroupPtr opcGrp;  
vector<OPCItemPtr> opcItms; // 使用vector来保存三个测试Item。  
//连接到OPC Server, 我所使用的参数是'Matrikon.OPC.Simulation.1'  
void agOPCConn( const char \*opcSvrName ) {  
HRESULT hr;  
hr = opcSvr.CreateInstance( \_\_uuidof( OPCServer ) );  
if( FAILED( hr ) ) {  
cerr << 'OPCServer CreateInstance failed, hr = ' << hr<< endl;  
exit(1);  
}  
opcSvr->Connect( opcSvrName );  
}  
//断开和OPC Server的连接  
void agOPCDisc() {  
opcGrps-&gt;RemoveAll(); // 删除所有的组, 这个演示实例只有一个组  
opcSvr->Disconnect(); // 断开和OPC Server的连接  
}  
//创建一个组  
void agOPCCreateGroup() {  
// OPCGroups是特殊的属性，执行的时候会调用OPCAuto.tlh中的IOPCGroupsPtr GetOPCGroups();  
opcGrps = opcSvr->OPCGroups;  
opcGrp = opcGrps->Add( \_variant\_t( 'group1' ) ); // 组名随意取  
}  
//在组里添加三个不同类型的测试Item, 类型可以从Item的名字可以看出  
void agOPCAddItems() {  
OPCItemPtr opcItm;  
opcItm = opcGrp->OPCItems->AddItem( \_bstr\_t( 'Bucket Brigade.Int4' ), 1 );  
opcItms.push\_back( opcItm );  
opcItm = opcGrp->OPCItems->AddItem( \_bstr\_t( 'Bucket Brigade.Int2' ) , 1);  
opcItms.push\_back( opcItm );  
opcItm = opcGrp->OPCItems->AddItem( \_bstr\_t( 'Bucket Brigade.String' ) , 1);  
opcItms.push\_back( opcItm );  
}  
//用来显示读取的Item的值  
void agDumpVariant(VARIANT \*v)  
{  
switch(v->vt)  
{  
case VT\_I2:  
printf('value(VT\_I2) = %d ', v->iVal );  
break;  
case VT\_I4:  
printf(' value(VT\_I4) = %ld ', v->lVal );  
break;  
case VT\_BSTR:  
printf(' value(VT\_BSTR) = %ls ', v->bstrVal );  
break;  
default:  
printf(' value(unknown type:%d) ', v->vt );  
break;  
}  
}  
//同步读取三个Item的值，同步在很多情况下都是简单有效的选择方案，其实读取的异步方式在C++中可以建立一个工作线程来执行同步读的操作，等有新的Item值的时候再通过某种线程间通信的方式告诉主线程“数据改变”的事件  
void agOPCReadItems() {  
\_variant\_t quality;  
\_variant\_t timestamp;  
SAFEARRAY \*pServerHandles;  
SAFEARRAY \*pValues;  
SAFEARRAY \*pErrors;  
SAFEARRAYBOUND rgsabound[ 1 ];  
long dim[ 1 ];  
long svrHdl;  
vector<\_variant\_t> values;  
vector<long> errs;  
int i;  
\_variant\_t value;  
long err;  
// VC数组索引从0开始，而在OPCAuto.dll需要中从1开始，所以是rgsabound[ 0 ].cElements = 4，而给pServerHandles赋值的时候应该给索引是1，2，3相应的赋值Server Handle  
rgsabound[ 0 ].cElements = 4;  
rgsabound[ 0 ].lLbound = 0;  
pServerHandles = SafeArrayCreate( VT\_I4, 1, rgsabound ); //构建一个1维数组，类型是VT\_I4  
for( i = 0; i < opcItms.size(); i++ ) {  
svrHdl = opcItms[i]->ServerHandle;  
dim[ 0 ] = i + 1;  
// 给数组的每个元素赋值，对应的索引值是1， 2， 3  
SafeArrayPutElement( pServerHandles, dim, &svrHdl );  
}  
opcGrp->SyncRead( OPCDevice,  
3, // 读取的Item数目  
&pServerHandles, // 输入的服务器端句柄数组  
&pValues, // 输出的Item值数组  
&pErrors, // 输出的Item错误状态数组  
&quality, // 读取的值的状态  
&amp;timestamp ); // 读取的事件戳  
for( i = 1; i <= opcItms.size(); i++ ) {  
dim[ 0 ] = i;  
SafeArrayGetElement( pValues, dim, &value ); // 读取Item值在value中  
SafeArrayGetElement( pErrors, dim, &err ); // 读取错误状态值在err中  
values.push\_back( value );  
errs.push\_back( err );  
}  
for( i = 0; i < values.size(); i++ ) {  
agDumpVariant( &values[ i ] ); // 显示读取的Item值  
cout << ', err = ' << errs[ i ] << endl;  
}  
SafeArrayDestroy( pServerHandles );  
SafeArrayDestroy( pValues );  
SafeArrayDestroy( pErrors );  
}  
// 写入3个Item的值，为了演示实例简单，参数传递3个对应的Item值  
void agOPCWriteItems( vector<\_variant\_t> values) {  
\_variant\_t quality;  
\_variant\_t timestamp;  
SAFEARRAY \*pServerHandles;  
SAFEARRAY \*pValues;  
SAFEARRAY \*pErrors;  
longdim[ 1 ];  
long svrHdl;  
int i;  
SAFEARRAYBOUND rgsabound[ 1 ];  
rgsabound[ 0 ].cElements = values.size() + 1;  
rgsabound[ 0 ].lLbound = 0;  
pServerHandles = SafeArrayCreate( VT\_I4, 1, rgsabound );  
pValues = SafeArrayCreate(VT\_VARIANT, 1, rgsabound);  
for( i = 0; i < values.size(); i++ ) {  
svrHdl = opcItms[i]->ServerHandle;  
dim[ 0 ] = i + 1;  
SafeArrayPutElement( pServerHandles, dim, &svrHdl );  
SafeArrayPutElement( pValues, dim, &values[i] );  
}  
opcGrp->SyncWrite( 3, &pServerHandles, &pValues,& pErrors );  
SafeArrayDestroy( pServerHandles );  
SafeArrayDestroy( pValues );  
SafeArrayDestroy( pErrors );  
}  
//main主程序  
int main()  
{  
try  
{  
agOPCConn( 'Matrikon.OPC.Simulation.1' );  
agOPCCreateGroup();  
agOPCAddItems();  
// 第一次写和读  
vector<\_variant\_t> values;  
values.push\_back( ( long )156 );  
values.push\_back( ( short )11 );  
values.push\_back( 'opc' );  
agOPCWriteItems( values );  
agOPCReadItems();  
cout << '---------------------------------------' << endl;  
// 第二次写和读  
vector<\_variant\_t> values1;  
values1.push\_back( ( long )123456 );  
values1.push\_back( ( short )666 );  
values1.push\_back( 'hello' );  
agOPCWriteItems( values1 );  
agOPCReadItems();  
}  
catch ( \_com\_error &e ) {  
// 应该在上面的子函数里面捕捉异常，但为了演示简单，在主函数里面捕捉异常  
\_bstr\_t bstrSource( e.Source( ) );  
\_bstr\_t bstrDescription( e.Description( ) );  
cout << 'Code = ' << e.Error() << endl;  
cout << 'Code meaning = ' << e.ErrorMessage() <<; endl;  
cout << 'Source = ' << ( LPCTSTR ) bstrSource << endl;  
cout << 'Description = ' << ( LPCTSTR ) bstrDescription<< endl;  
}  
return 0;  
}

**关于OPC Client 编写**

2013年12月05日 19:57:39 [changbaolong](https://me.csdn.net/changbaolong) 阅读数：15282 标签： [OPC](http://so.csdn.net/so/search/s.do?q=OPC&t=blog) [客户端](http://so.csdn.net/so/search/s.do?q=客户端&t=blog) 更多

个人分类： [OPC](https://blog.csdn.net/changbaolong/article/category/1787477)

    昨天又有人问我 OPC Client 编写，实际是他们不了解OPC 客户端的工作原理，要想写客户端程序，必须知道OPC对象， OPC逻辑对象模型包括３类对象：OPC server对象、OPC group对象、OPC item对象，每类对象都包括一系列接口。

    OPC Server对象

    主要功能为：1、创建和管理OPC Group对象；

                2、管理服务器内部的状态信息；

    OPC Group对象

    主要功能为：1、管理OPC Group对象的内部状态信息；

                2、创建和管理Items对象。

                3、OPC服务器内部的实时数据存取服务（同步与异步方式）。

    OPC组中有以下几个主要属性：Name ：组的名字 ；Active：组的激活状态标志 ；Update Rate OPC：服务器向客户程序提交数据变化的刷新速率；Percent Dead band：数据死区，即能引起数据变化的最小数值百分比。

    OPC ITEM 是非COM对象，在OPC标准中用来描述实时数据，是客户端不可见的对象。代表了与服务器中的数据的连接，它并不是数据源，而仅仅是与数据源的连接。每个项都有以下主要属性： Active项的激活状态、Value项的数值、类型为VARIANT、Quality项的品质，代表数值的可信度，类型为SHORT、TimeStamp时间戳，代表数据的存取时间。

    你不管用什么开发语言只要了解上面几个对象，就会写程序了

    下面举个VC的例子

    HRESULT  r1;  
 CLSID  clsid;  
 LONG  TimeBias = 0;  
 FLOAT  PercentDeadband = 0.0;  
 DWORD  RevisedUpdateRate;  
 LPWSTR  ErrorStr;  
 char  str[100];  
 CString  szErrorText;

 m\_pItemResult = NULL;

 客户端程序必须对DCOM进行初始化设置，以保证OPC服务器端回调函数不会被堵塞。   
 r1 = CoInitialize(NULL);  
 if (r1 != S\_OK)  
 { if (r1 == S\_FALSE)  
  { MessageBox("COM Library already initialized",  
              "Error CoInitialize()", MB\_OK+MB\_ICONEXCLAMATION);  
  }  
  else  
  { szErrorText.Format("Initialisation of COM Library failed. Error Code= %4x", r1);  
   MessageBox(szErrorText,"Error CoInitialize()", MB\_OK+MB\_ICONERROR);  
   SendMessage(WM\_CLOSE);  
   return;  
  }  
 }  
   
 通过OPC服务器的ProgID查询注册表中相关CLSID。每个COM服务器都有一个字符串型的ProgID，通过ProgID可以得到全球惟一的CLSID，使用CLSIDFromProgID( )函数实现ProgID到CLSID的转换。 r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);  
 if (r1 != S\_OK)  
 { MessageBox("Retrival of CLSID failed",  
       "Error CLSIDFromProgID()", MB\_OK+MB\_ICONERROR);  
  CoUninitialize();  
  SendMessage(WM\_CLOSE);  
  return;  
 }

 连接OPC服务器，查询对象的IID\_IOPCServer接口。在连接OPC服务器前，OPC客户端需要事先指定计算机名和OPC数据访问服务器名，建立连接后，创建OPC组并添加OPC数据项。

 r1 = CoCreateInstance (clsid, NULL, CLSCTX\_LOCAL\_SERVER ,IID\_IOPCServer, (void\*\*)&m\_pIOPCServer);  
 if (r1 != S\_OK)  
 { MessageBox("Creation of IOPCServer-Object failed",  
             "Error CoCreateInstance()", MB\_OK+MB\_ICONERROR);  
  m\_pIOPCServer = NULL;  
  CoUninitialize();  
  SendMessage(WM\_CLOSE);  
  return;  
 }

 创建OPC组，查询IOPCItemMgt接口。IOPCServer接口的AddGroup()[方法](http://www.studa.cn/)可以创建一个有指定名称和属性的OPC组。

r1=m\_pIOPCServer->AddGroup(L"grp1",   // [in] group name  
      TRUE,     // [in] active  
      500,     // [in] request this Update Rate from Server  
      1,      // [in] Client handle  
      &TimeBias,    // [in] no time interval to system UTC time  
      &PercentDeadband,   // [in] no deadband, so all data changes are reported   
      LOCALE\_ID,    // [in] Server uses English language for text values  
      &m\_GrpSrvHandle,  // [out] Server handle to identify this group in later calls  
      &RevisedUpdateRate,  // [out] the answer form the Server to the requested update rate  
         IID\_IOPCItemMgt,  // [in] requested interface type of the group object  
      (LPUNKNOWN\*)&m\_pIOPCItemMgt); // [out] pointer to the requested interface  
   
 if (r1 == OPC\_S\_UNSUPPORTEDRATE)  
 {   
  szErrorText.Format ("Revised Update Rate %d is different from Requested Update Rate 500",RevisedUpdateRate );  
  AfxMessageBox(szErrorText);  
 }  
 else  
  if (FAILED(r1)){  
   MessageBox("Can't add Group to Server!", "Error AddGroup()", MB\_OK+MB\_ICONERROR);  
   m\_pIOPCServer->Release();  
   m\_pIOPCServer = NULL;  
   CoUninitialize();  
   SendMessage(WM\_CLOSE);  
   return;  
  }  
添加OPC数据项。使用IOPCItemMgt接口的AddItem()方法可以添加具有特殊属性的指定数量的数据项。

// define an item table with one item as in-paramter for AddItem  
 m\_Items[0].szAccessPath    = L"";

 m\_Items[0].szItemID     = szItemID;  // 影响数据类型  
 m\_Items[0].bActive     = TRUE;     
 m\_Items[0].hClient     = 1;  
 m\_Items[0].dwBlobSize    = 0;  
 m\_Items[0].pBlob     = NULL;  
 m\_Items[0].vtRequestedDataType = 0;      
             // defined by the item itself

 r1 = m\_pIOPCItemMgt->AddItems(1,    // [in] add one item  
       m\_Items,    // [in] see above  
       &m\_pItemResult,   // [out] array with additional information about the item  
       &m\_pErrors);   // [out] tells which of the items was successfully added.  
             // For any item which failed it provides a reason

程序退出时

OPC连接断开，释放接口指针。当程序退出或停止服务器时，依次删除Item(RemoveItems)、Group(RemoveGroups)，释放资源。

void CAsynOPCDlg：：OnStop( ) 停止服务器。

h1=m\_IOPCItemMgt->RemoveItems(1，phServer，&pErrors)；删除Items。

h1=m\_IOPCServer->RemoveGroup(m\_GrpSrvHandle，TRUE)；删除Group。

m\_IOPCServer->Release( )；释放服务器。

m\_IOPCServer=NULL；

CoUninitialize( )；关闭COM库。

转载：[http://blog.cechina.cn/zhangfenglu/252638/message.aspx#](http://blog.cechina.cn/zhangfenglu/252638/message.aspx)

**OPC及OPC服务器的设计与实现**

2004年09月20日 10:14:00 [freeboy20sui](https://me.csdn.net/freeboy20sui) 阅读数：12023 标签： [服务器](http://so.csdn.net/so/search/s.do?q=服务器&t=blog) [microsoft](http://so.csdn.net/so/search/s.do?q=microsoft&t=blog) [服务器软件](http://so.csdn.net/so/search/s.do?q=服务器软件&t=blog) [制造](http://so.csdn.net/so/search/s.do?q=制造&t=blog) [null](http://so.csdn.net/so/search/s.do?q=null&t=blog) [工作](http://so.csdn.net/so/search/s.do?q=工作&t=blog)

come from：

<http://www.chint.net/bbs/dispbbs.asp?Boardid=4&ID=50915&replyID=315788&skin=1>

**OPC及OPC服务器的设计与实现**  
OPC及OPC服务器的设计与实现  
                         \_\_\_\_\_\_\_OPCAnd Design&Implementation of OPC Server  
引言   
   OPC（OLE for  process  control）是一个工业标准。它是由一些世界上占领领先地位的自动化系统和硬件、软件公司和微软合作而建立的、满足开放性的和互操作性接口标准。OPC标准的建立基于微软的COM技术规范，并由OPC基金会这个国际组织管理。OPC为不同的厂商的硬件设备、软件和系统定义了公共的接口，使得过程控制和工厂自动化中的不同的系统、设备和软件之间能够互相连接、通信、操作。  
一般在控制系统中，包含过程画面的软件（如SCADA软件）在与现场设备通信的时常    
  采用如下的方法：  
  (1)采用标准通信协议，如profibus、modbus、HART、CAN总线等，因为监控管理软件与现场的设备采用同样的通信协议，所以，监控管理软件不需要对现场设备的驱动程序。  
  (2)对于没有采用标准通信协议的设备，监控管理软件需要在设备制造商的配合下为这些设备开发驱动程序。这种方式虽然执行效率比较高，但兼容性差，软件供应者必须对没一种接入的设备开发驱动程序，设备制造商也必须提供设备的通信协议。  
  （3）通过OPC这个开放性的协议和过程控制或其他系统软件进行通信。这种方式的优点在于：不管硬件设备是否使用标准的通信协议，制造商只需要提供一套OPC服务器，就可以支持大部分的监控等软件，也不需要将自己的通信协议细节提供给软件商。  
   OPC服务器的设计和实现  
   目前的小型DCS控制系统使用了OPC服务器后，实现了对主流的监控软件（一般都有OPC接口）的兼容性，监控软件等OPC客户程序可以很方便的访问和设置DCS中的数据。  
OPC服务器需要实现的主要功能就是根据控制系统的组态信息，实现OPC服务器对象、组对象等，并通过OPC驱动程序实现与DCS系统中主控器的通信，获得现场数据或设置现场数据项等。  
   OPC服务器软件主要分为OPC服务器对象模块、服务器界面模块和OPC驱动程序模块，三个模块通过同一块主内存数据区来共享数据，通过线程的同步和互斥等技术的使用，可以解决共享数据的保护问题。下面简单介绍一下各个模块的功能：  
  （1） 服务器界面模块  
 服务器界面程序主要根据组态信息完成OPC Group对象和OPC Item对象的添加，并采用树型结构浏览查看内存数据区中的数据项（TAG），如数字量信号输入卡的输入信号等。  
  （2） OPC驱动程序模块  
 OPC驱动程序主要通过TCP/IP协议与多台主控制器的通信来实现OPC服务器的驱动部分（通过对驱动程序的替换，可以实现对其他产品的OPC服务器）。  
   （3）OPC服务器对象模块  
     OPC服务器对象模块是OPC服务器程序与OPC客户程序的交互部分，主要依靠OPC基金会的数据存取规范来实现。一般需要实现OPCserver、OPCGroup、EnumOPCItemAttributes等对象，其中OPCServer对象需要实现IOPCCommon、IOPCServer、IOPCItemProperties、IconnectionPointContaniner等接口的方法；OPCGroup对象需要实现IOPCSyncIO、IOPCGroupStateMgt、       
IOPCAsyncIO2、IOPCItemgt、IconnectionPointContainer等接口方法；EnumOPCItenAttributes需要实现IEnumOPCItemAttributes接口的方法。接口方法的原型参照OPCDa.idl中的定义。  
   在对OPC服务器和客户端的开发中使用IDE的是  Microsoft  VisualC++,其中OPCServer对象、OPCGroup对象等COM组件的定义和实现运用了微软的ATL（Active Template Library活动摸板库）技术。  
   下面的这一段代码示范了OPC服务器IOPCServer接口GetStarus()方法的访问（客户端可以通过GetStarus方法获得服务器开始工作的时间、更新、名称等信息）。  
OPC客户程序：  
{………….  
  CoInitialize(NULL);   //初始化COM环境  
   ……..             //略  
  IOPCServer       \*pSvr=null;   //定义IOPCSERVER接口指针  
  HRESULT  hr=CoCreateInstance(CLSID-OPCServer,NULL,CLSCTX-ALL,  
                                  IID-IOPCServer,(void\*\*)&pSvr);  
   //备注：OPCDa2.0可以通过IOPCServer调用OPCServerList获得OPCserver的CLSID  
   OPCSERVERSTATUS\*pServerStaus;   //定义OPCServer状态的结构  
  PSvr->GetStaus(&pServerStaus);       //获得OPCserver返回的状态  
  …………  
  pSvr->Relese();                     //释放IOPCServer接口指针  
  Couninitialize();                     //结束COM环境  
  }  
  OPC服务器程序：  
 Class ATL-NO-VTABLE COPCserver:  
      Public CcomObjectRootEx<CcomSingleThreadModel>,  
      Public CComCoClass<CmeSvr,&CLSID-MeSvr,  
      Public IOPCServer……//定义OPCServer  
  {………  
  STDMETHOD(GetStatus)(OPCSERVERSTATUS\*\*ppServerStatus)  
 {  
   if (PPServerStatus==NULL)  
    return E-INVALIDARG;   //错误的调用  
  OPCSERVERSTATUS\*ppServerStdus;  //定义OPCServer状态的结构  
  PServerStatus=(OPCSERVERSTATUS\*)pM->Alloc(OPCSERVERSTATUS)); //分配内存  
  If(pServerStatus)  
{  
   pServerStatus->szVendorInfo=VendouInf;  //制造商信息  
  pServerStatus->ftStartTime=svrStartTime;  //OPCserver开始工作时间  
  CoFileTimeNow(&ServerStatus->ftCurrentTime;  //当前时间  
  PServerstatus->ftLastUpdateTime=m-LastUpdate;  //最近更新时间  
  PServerstatus->dwServerStatus=OPC-STATUS-RUNNING;  //OPCserver状态  
  PServerstatus->deGroupCount=0;  //OPCserver中Group个数  
  PServerstatus->dwBandWidth=0;   //带宽  
  PServerstatus->wMajorVersion=0;  //版本信息  
  PServerstatus->wMinorVersion=0;  //版本信息  
  PServerstatus->wBuildNumber=0;  //版本信息  
  PserverStatus->wReserved=11;     //保留位  
  \*ppServerstatus=pServerStatus;    //向OPCCClient返回OPCServer状态  
  returen S-OK;   //OPCClient调用成功  
                    }  
  return E-OUTOFMEMORY;   //内存不够  
        }  
           ………. //略  
                 }  
   通过上面这个例子，可以了解到OPC客户程序OPC服务器程序的调用过程，即OPC服务器程序定义了COPCServer这个COM组件，继承了IOPCserver接口并实现了该接口的GetStatus()方法。OPC客户程序获得IOPCserver接口指针后，通过对GetStatus()方法的调用获得OPC服务器的工作状态，类似地，通过对OPC规范中定义的其他的必要借口（Required Interface）的实现，就可以：即插即用“的将设备集成到各种符合OPC规范的系统或应用中去。  
  硬件设备通过OPC服务器可以规范地、以独立于协议的方式与客户进行通信，极大地提高了控制系统的互连和互操作性。同时COM、OPC规范都在不断的发展壮大之中，通过这些技术的掌握和运用，我们可以实现开放性好、兼容性强、配置方便的分布式控制系统

OPC UA book https://max.book118.com/html/2017/0622/117300344.shtm

**OPC客户端开发几个境界**

2004年09月14日 15:29:00 [freeboy20sui](https://me.csdn.net/freeboy20sui) 阅读数：7445 标签： [vb](http://so.csdn.net/so/search/s.do?q=vb&t=blog) [interface](http://so.csdn.net/so/search/s.do?q=interface&t=blog) [delphi](http://so.csdn.net/so/search/s.do?q=delphi&t=blog) [编程](http://so.csdn.net/so/search/s.do?q=编程&t=blog) [开发工具](http://so.csdn.net/so/search/s.do?q=开发工具&t=blog) [文档](http://so.csdn.net/so/search/s.do?q=文档&t=blog)

OPC我就不介绍了,就是那么回事.呵呵.

开发OPC说难也不难看你想掌握到什么程度.下面写一下有浅入深的几个境界.

1. 超简单的:? 用组态软件去实现^\_^，随便找一个组态软件，按F1找到本组态软件OPC相关的帮助，加入服务器就可以了。特点：简单实用，方便易用。缺点：组态要钱，灵活性极低。时间：最多半天。推荐

2. 很简单的：利用OPC? Active ，找一个控件，加入您的工程（VB的或VC的）就可以了。特点：简单不实用，方便但没用。缺点：这种组件不多，灵活性低。时间：找控件n天，应用一回就可以了。

3. 简单的：利用快速开发包。用客户端的快速开发包。特点：简单实用，快速方便，灵活性高。缺点：要钱，而且对开发包有依赖性。时间：看掌握的程度。

4.通用简单的：利用基金会提供的OPCDAauto.dll，用自动化接口编程，利用VB等工具编程。这是基金会的的标准，网上有很多的源码（有VB也有delphi）。推荐使用这种方法。有以下优点：a。是OPC标准，没有任何依赖性（当然要依赖OPC）b。有较好的源码可以参考，并且有相关的标准文档。C。不用其他费用。d。就掌握难度来说，只要有一点的编程基础就可以掌握。并且源码可以直接应用到系统开发中。

**????? 相关资料：**

[Kepware](http://www.kepware.com/)? 的网站有vb ，VC的源码。

[http://www.kepware.com](http://www.kepware.com/)  
[www.kepwarecn.com](http://www.kepwarecn.com/)

<http://www.faweb.net/us/opc/opclabs.html#OPC_Layer>  
OPC cliet相关的代码,简单的应用. vb 简单实用

**OPC基金会有相关的英文文档**

Data Access Automation Interface Standard Version 2.01

书籍：《OPC应用程序入门》－－－－很多人都说这本书很烂，其实就是很烂总比看英文好一点。

5. 通用有点难的：应用VC custom 开发接口。比用自动化接口难度增大，要求对COM有一定的了解，并且能应用VC开发工具。在Vc下熟练的可以使用。优点是可以不用opcdaauto。dll了。

6. 变态的：自己实现Automation Interface ，呵呵，这是最变态的做法，kepware就是这样做的，他们的kepdaauto.dll就是这个例子，可以直接取代opcdaauto.dll真是巨变态啊。

?????? 总体来说OPC Cliet的客户端开发难度不是太大，尤其是不需要了解COM相关的知识。只是大家了解的资料不多。国内没有相关的专业网站，只有工控OPC论坛可以看看。有空可以来我blog看看。我会写相关的文章来给大家的。希望大家来捧场啊 ^\_^

<http://blog.csdn.net/freeboy20sui>

?

原

**OPC技术学习总结**

2015年12月03日 16:50:58 [xwt\_coding](https://me.csdn.net/u013120247) 阅读数：12063 标签： [OPC](http://so.csdn.net/so/search/s.do?q=OPC&t=blog) [JeasyOPC](http://so.csdn.net/so/search/s.do?q=JeasyOPC&t=blog) [Openscadautgard](http://so.csdn.net/so/search/s.do?q=Openscadautgard&t=blog) [OPC工作原理](http://so.csdn.net/so/search/s.do?q=OPC工作原理&t=blog) [OPC的java实现API](http://so.csdn.net/so/search/s.do?q=OPC的java实现API&t=blog) 更多

个人分类： [OPC](https://blog.csdn.net/u013120247/article/category/6001013)

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/u013120247/article/details/50163147

前段时间公司项目需要用到OPC方面的技术，所以就学习了解了一下，并做个总结，和大家分享分享。

**1. 什么是OPC**

**1.1 名词解释**

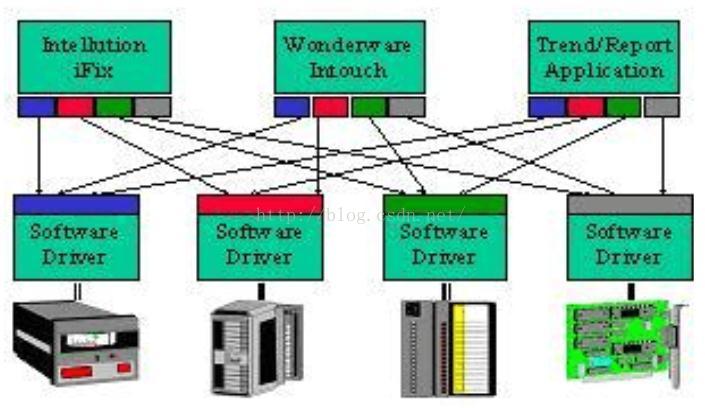
Ø  OPC——OLE for Process Contrl，微软公司对象链接和嵌入技术在过程控制方面的应用，是一系列接口、方法和属性的标准集。它基于OLE、COM、DCOM技术、XML，采用客户端/服务器（Client/Server）结构，将通讯协议与设备/应用隔离的技术。微软的OLE、COM定义了各种不同的软件如何交互使用和分享数据，从而使得OPC能够提供通用的接口规范，用于各种过程控制设备之间的通讯，不论过程中采用何种应用软件和硬件设备。

Ø  OLE——Object Linking & Embeding，对象链接与嵌入，也就是现在的微软的ActiveX。

**1.2  传统技术与OPC技术**

**1.2.1传统的过程控制系统结构**

传统的过程控制系统是一对一的系统，任何一种HMI等上位监控软件或其它应用软件，在使用某种硬件设备时都需要开发专用的驱动程序。



系统构建完成后的最终结果是：

Ø  1种软件要使用N类硬件设备需要开发N个驱动程序；

Ø  M类软件要使用N类硬件设备需要开发M\*N个驱动程序；

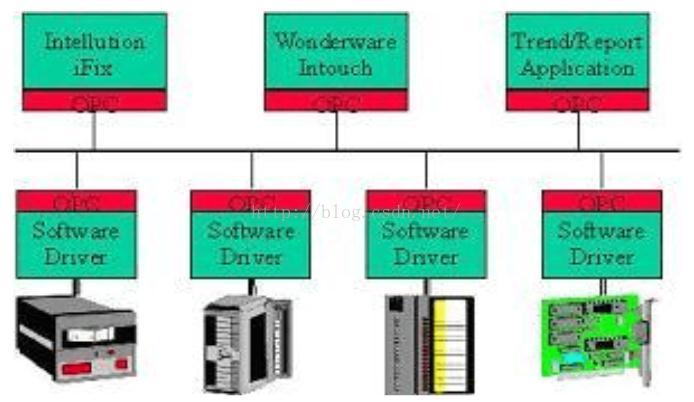
Ø  每增加1个新的应用软件需要另外开发N个硬件设备的驱动程序；

Ø  每增加1个新的硬件设备需要为M个软件开发新的设备驱动程序。

在上图所示的例子中，针对4种控制设备所完成的3个应用系统一共需要开发3\*4=12种驱动程序。新增应用软件或者硬件设备带来的只会是驱动程序种类的迅速增长。

**1.2.2基于OPC技术的过程控制系统结构**

基于OPC技术的过程控制系统则可以完美地解决传统方式种的上述问题。



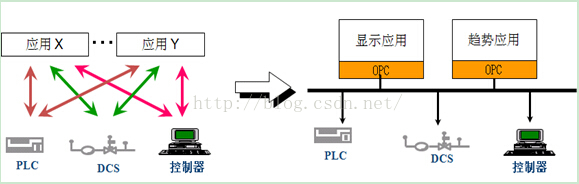
任何一种设备只需要提供一种驱动就可以供任何软件系统使用，系统构建完成后的最终结果是：

Ø  M类软件要使用N类硬件设备只需要开发N个驱动。

Ø  每增加1个新的应用软件不需要另外开发硬件设备的驱动程序；

Ø  每增加1个新的硬件设备只需要为开发1个新设备的驱动程序；

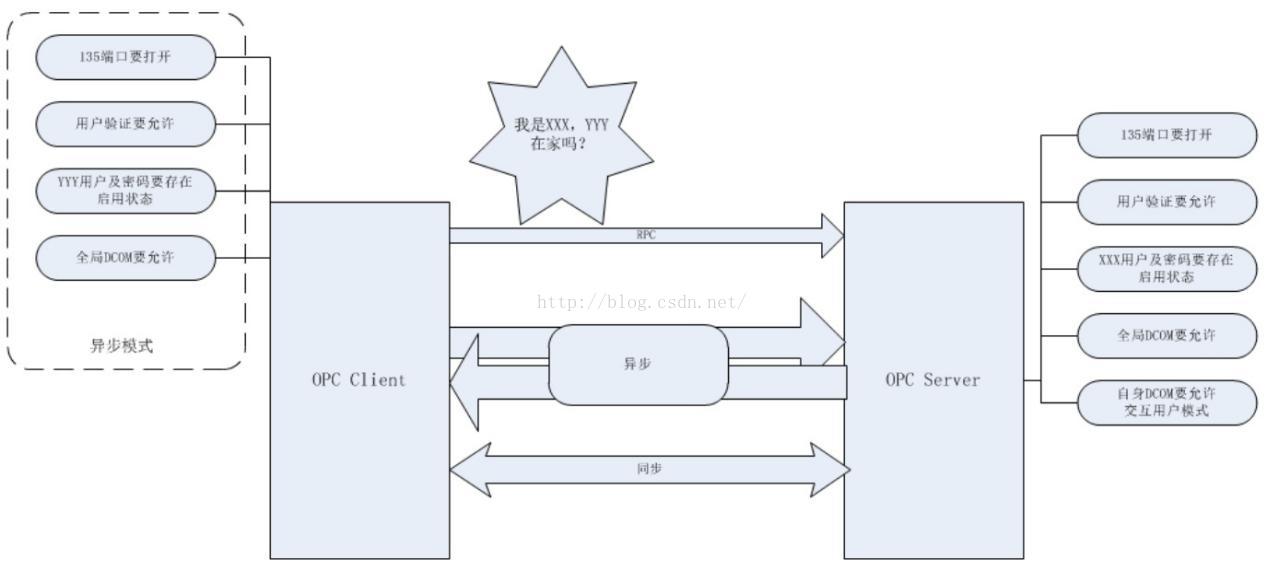
在上图所示的例子中，针对4种控制设备所完成的3个应用系统一共仅需要开发4种驱动程序。新增应用软件或者硬件设备可以轻松的扩展您的系统。



**1.3 工作原理**



详细原理图：



**1.4  OPC服务器所兼容的设备分类**

Ø  硬件设备：DCS，PLC，扫描仪，电子设备…

Ø  应用软件：HMI，历史数据库，震动检测器…

Ø  通讯协议：DDE，Modbus，ODBC，GenCS…

Ø  操作系统：Windows，UNIX，VMS，Macintosh，嵌入式OS…

即OPC服务器可以从上面任何一种形式的数据产生实体中读取数据。这个实体可以是霍尼韦尔PHD的Application，也可以是Modbus的通讯协议，还可以是一种PLC，也就是一种Device（硬件设备），或者是一种Platform。

**2.常用的OPC服务器**

**2.1 Matrikon公司的OPCServer Simulator**

简单，单文件注册后，各种数据类型，各种点类型（只读点，只写点，读写点）都有了，更好的一点是，如果想做大规模的测试，10万点，那么只需要在它的Random下添加任意名称即可，OPCServer会自动给你建出对应的测点，还是随机数变化。

**2.1.1配置时的几个关键概念**

Alias Group(Group) ——对Items的分类管理，自定义的。类似于Java中的包。

Alias (Item) ——一个Item代表与服务器进行通信的PLC或其它硬件设备上的一个地址。与Kepware中的一样。

配置顺序如下（在Alias Configuration下）：

New AliasGroup/Alias --> Inset Alias

**2.2 Kepware公司的OPCServer**

一个商业软件，需要花费金钱购买License的，但Kepware公司的OPCServer需要花钱的不是OPCServer本身，而是采集驱动。不过Kepware公司的OPCServer提供了不花钱的仿真驱动，用来测试上非常好的。

缺点就是太庞大，不简单，学习成本较高。但为了能满足OPC这类产品的各种测试，它真的是最佳选择。可以去它的官方网站下载。

**2.2.1配置时的几个关键概念**

Chanel——通道是一个从PC到一个或多个外部设备之间的传播媒介。一个通道可以用来代表一个串行端口（一个安装在个人电脑上的卡或是以太网接口）。

Device——设备代表了与服务器进行通信的PLC或其他硬件。它受限于Channel所选择的设备驱动程序（Device driver）。

Group——对tags的分类管理，自定义的。类似于Java中的包。

Tag——一个Tag代表与服务器进行通信的PLC或其它硬件设备上的一个地址。服务器允许动态标签（客户端自定义创建）和用户定义的静态标签（服务端管理人员创建的标签）。动态标签是直接进入了OPC客户端和指定设备存取数据。静态标签在服务器被创建的且支持标签扩展，他们可以从OPC客户浏览,支持标签浏览。

配置顺序如下：

New Channel -->New Device --> New Group/Tag --> New Tag

**3. 常用的OPC客户端**

**3.1** [**Matrikon公司的OPCClient**](http://sns.ca800.com/)

标准，稳定，单文件，通讯过程的信息还比较丰富，我是比较喜欢用它在创建OPC组时定义同步方式或异步方式来验证远程计算机的OPC配置是否正确。缺点就是大了点，2M多。

**3.2 Kepware公司的OPCClient**

功能很强大，标准，稳定，日志信息很丰富，最推荐的功能是支持对OPCServer中点名的条件过滤，支持点表的导入导出，支持自动导入OPCServer的所有点，根据点表识别好点坏点，按照列排序，用它可以弥补很多国产组态软件不能在线遍历OPCServer点表的功能、不能过滤OPCServer点的功能、不能识别OPCServer中好点坏点的功能等。缺点就是非单文件。

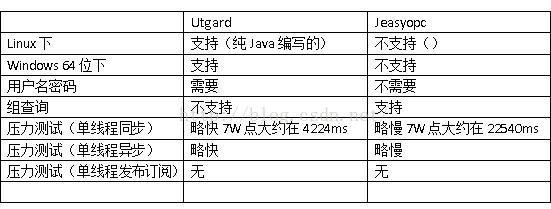
**4. OPC Client和OPC Server配置的安全策略**

下表是对 OPC Client 和 OPC Server 所在操作系统之间的安全策略的配置对应说明：



**5. Java实现OPC客户端**

有两种开源包，分别是JeasyOpc和utgard（Openscada），他们的区别如下：



经对比分析，我们采用Utgard（Openscada）开源技术，Openscada可以很好的实现与OPC服务器的链接及读写数据等功能，还可以随时获取和修改OPC的Server和Item的属性状态信息。所以它能够很好的满足我们的需求，建议使用Openscada。

**5.1 Openscada链接配置方法**

OPC数据存取服务器由三个对象组成：服务器对象(Server)、组对象(Group)和项对象(Item)。

按说明配置好OPC Server  与 OPC Client 所在电脑的组件服务配置和防火墙设置（windows7直接关闭就行了）注意一定要把本机希望链接OPC服务的用户或用户组添加到DCOM配置列表中，否则链接会失败。

如果其他都配置好了，运行程序还是连接不上的话，首先常看防火墙是否配置（或关闭）。

OPC对象——服务器（OPC  Server）、组（OPC  Group）、项（OPC  Item）。

Openscada开源项目：

         ConnectionInformation中：

                   Host——本地主机/网络主机IP （示例：localhost（默认）、127.0.0.1）

                   Domain——域（默认为localhost）

                   User——用户名

                   Password——用户登录密码

                   Clsid——应用在注册表中相对应的CLSID值

                   Grogid——应用在注册表中对应的程序名称

                   【Clsid 和 Grogid 作用相同，只要设置一个就可以了，如果两个都设置了，程序会优先选择Clsid。我们建议使用Clsid，因为使用Grogid时，Openscada的内部处理还是会通过JISystem.getClsidFromProgId( progId )方法将其转换为Clsid，并且还需要进行服务器上用户的权限的高级配置才可以使用。查找Clsid和Grogid的方法：打开注册表（regedit）—》在HKEY\_CLASSES\_ROOT下找到相应的OPC服务器名称，该名称就是Grogid，在其目录下的CLSID的值（右边区域）就是Clsid】

**5.2 Openscada远程链接时常见的问题及解决方法**

（1）org.jinterop.dcom.common.JIException: Message not found for errorCode:0xC0000034

原因：未启动RemoteRegistry和Windows Management Instrumentation服务。

解决方法：打开控制面板，点击【管理工具】—>>【服务】，启动RemoteRegistry和Windows ManagementInstrumentation服务。

（2）org.jinterop.dcom.common.JIException:Access is denied, please check whether the [domain-username-password] arecorrect. Also, if not already done please check the GETTING STARTED and FAQsections in readme.htm. They provide information on how to correctly configurethe Windows machine for DCOM access, so as to avoid such exceptions.  [0x00000005]

原因：首先检查错误提示的配置信息是否有误，如果都正确，则原因可能是你访问的当前用户没有该访问权限。

解决方法：

1、打开注册列表，

选择HKEY\_CLASSES\_ROOT\CLSID\{76A64158-CB41-11D1-8B02-00600806D9B6}

2、右键点击[权限]>>【高级】>>[所有者]>>添加opc用户到权限项目中，点击应用，确定。