

SLIK-DAC Control

(Simple Language Independent Kit for Creating OPC Data Access Clients)
 "Quality OPC Server Implementation Solutions for Software Developers"



Contents

OPC AND SLIK-DAC..... 2
 TUTORIAL A—DEVELOP A SIMPLE CLIENT..... 3
 TUTORIAL B—IMPLEMENT ADVANCED FEATURES..... 4
 PRODUCT SUMMARY SHEET..... 7



Introduction

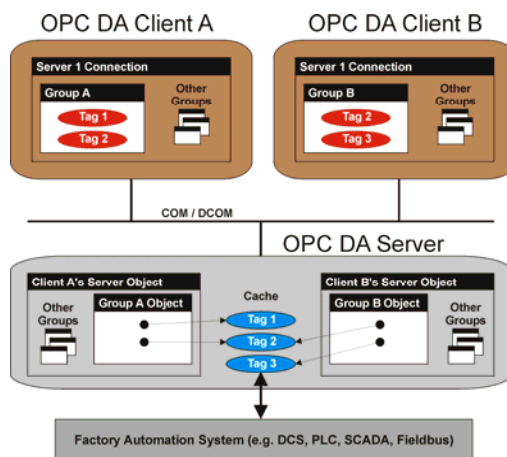
The Simple Language Independent Kit (SLIK) for creating OPC Data Access Clients (DAC) provides a level of simplicity and flexibility not available before in an OPC client software development toolkit. SLIK-DAC is an ActiveX and .NET Windows Form Control used to quickly add OPC client DA interfaces to your Visual Basic, C#, or VB.NET application. It is a sister product to our SLIK-DA toolkit, which is used to develop OPC DA servers.

This document provides two tutorials to demonstrate just how easy it is to develop an OPC client using SLIK-DAC. The first tutorial will show you how to create a simple OPC client that uses the basic functionality of SLIK-DAC. The second tutorial illustrates some advanced features that are relevant to a typical commercial OPC client. While the language used for the sample code is C#, the techniques you learn here can be applied to clients written in any language supported by SLIK-DAC.

To complement these tutorials, the evaluation version of SLIK-DAC includes comprehensive examples of OPC client source code. A sample client exercising the full range of SLIK-DAC functionality is provided with source code in C#, Visual

Free Evaluation Software

A free 30 day full-featured evaluation version of SLIK-DAC is available to download from our website. The URL is <http://www.OPCexperts.com/SLIKDAC>. Then click on Free Evaluation. You can build your own OPC client with the evaluation software.



This diagram illustrates a logical view of the runtime interaction between an OPC DA server and two OPC DA clients.

Servers can simultaneously manage connections with one or more clients. While clients create groups and populate the groups with item references, servers manage the groups and item references.

Typically, items are managed by a server as a cache or real-time database. How the cache is monitored and updated is server implementation specific. How the clients are updated is specified by the OPC DA standard.

OPC and SLIK-DAC

Introduction to OPC

OPC (or Ole for Process Control) is a set of publicly available technical standards for enabling the interoperation of factory automation software applications. The OPC standards are developed by factory automation equipment suppliers and end users through an organization called the OPC Foundation.

Presently, OPC standards exist for the following types of factory automation software interoperation:

- Data Access (DA): real-time data
- Alarms & Events: real-time alarm and event data
- Historical Data Access (HDA): Historical data
- Security: Secure server data access

The OPC standards require software applications to adopt a role for data interchange. Two roles are defined: a server and a client. A server is a producer of data in the exchange. Examples include servers for PLCs and DCSs. A client is a consumer of data in the exchange. Examples include SCADA/HMI software and Northern Dynamic's OPC Gateway product. SLIK-DAC is used by software developers and system integrators to implement DA clients.

Overview of OPC Data Access

As illustrated on the previous page, an OPC Data Access (DA) client connects to an OPC DA server and manages the data interchange using a number of objects, specifically the server, group, and item objects.

Server Object — This object represents the server to which a client has requested a connection. The object is also a container for all the group objects created by the client.

Group Object—This object is a container for organizing one or more items. Clients create and manage groups in a manner that suits their application.

Item Object—Items are the real-time data references. Servers present a list of items that are available to a client. The client chooses which items to use, how to organize them and how to use them. A client can choose to read, write, or receive unsolicited updates for an item. Each item has three values associated with it: a value, a timestamp, and a quality indicator.

About SLIK-DAC

SLIKDAC is an ActiveX and Windows Form Control for creating OPC Data Access client applications. The product can be used to create a new OPC DA client application or to add OPC DA client capability to existing Visual Basic, C#, or VB.NET applications.

Benefits of Using the SLIK-DAC Control

Cut Time from Your Project Schedule.

The SLIK-DAC toolkit will be familiar to programmers that have worked with the OPC Foundation's Automation Wrapper. Now program in Visual Basic, C#, or VB.NET using a familiar object model.

Manage Software Maintenance Costs.

Update your client to the latest version of the OPC specification by updating your toolkit. Our OPC experts take care of the OPC details. An affordable software maintenance program is available.

Reduce Support and Integration Costs for Your Server.

SLIK-DAC provides full support for COM-Call Tracing, a built-in OPC protocol analyzer solution. Quickly diagnose OPC client/server communication problems.

Add OPC Security.

SLIK-DAC now provides an easy-to-use set of methods to logon and logoff any OPC server supporting the OPC Private Security Interface.

Rely on OPC Expert Support.

Northern Dynamic has provided OPC client and server technical support for 7 years to a wide variety of international companies. Get our SLIK-DAC toolkit and our OPC experts behind your next client implementation.



Tutorial A —Develop a Simple Client

Pre-Requisites

If you have not already done so, please download the SLIK-DAC evaluation software from: <http://www.OPCexperts.com/SLIKDAC>. This document assumes that you have a basic understanding of C# and Visual Studio .NET.

SLIK-DAC adheres to the object model described in *OPC Data Access Automation Specification 2.02*. So, before diving into developing applications with SLIK-DAC, please take a moment to take a look at that document, especially Section 2.2, "OPC Automation Server Object Model."

Tutorial Overview

The instructions in the following tutorial sections illustrate how quickly you can develop an OPC framework for your own client using SLIK-DAC ActiveX and Visual Basic.

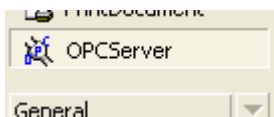
Follow the steps to create a simple OPC Data Access client that simulates tags/points/items from an automation system.

As an overview, here is what you will be doing.

1. Adding an instance of SLIK-DAC to your C# project
2. Connecting to an OPC server
3. Creating an OPCGroup object
4. Adding OPCItem objects to an OPCGroup.
5. Reading a group of OPCItems

Add an Instance of SLIK-DAC to Your C# Project .

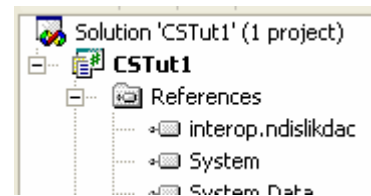
1. Ensure SLIK-DAC is present in your toolbox:
 - a. Right-click on the Toolbox and choose Add/Remove Items...
 - b. Click the .NET Framework Components tab and select OPCServer.
 - c. Click OK. The OPCServer component should appear in your toolbox:



2. Drag the OPCServer component onto one of your forms. The following icon should appear on the form:



1. You can click this object and modify its properties. Most importantly, you may wish to rename the instance of the object; in this tutorial the name `m_opcServer` is used. At this point, the namespace `Interop.NDISLIKDAC` should be displayed in your project's References list:



2. Note: It is recommended that you include the following line at the top of any C# source file that refers to SLIK-DAC objects:

```
using NDI.SLIKDAC.Interop;
```

Connect to an OPC Server

You can connect to an OPC server by using the `OPCServer.Connect` method. Assuming your instance of SLIK-DAC is named `m_opcServer`, the connection would be established as follows:

```
try {
m_opcServer.Connect( "My.OPCServer.1" );
} catch ( Exception e ) {
    MessageBox.Show(
        "Cannot connect to OPC server.",
        "Connection Error",
        MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);
}
```

The first argument to the `Connect` method is the ProgID of the OPC server with which you wish to communicate. By default, the OPC server is assumed to reside on the same machine as the client. However, you may specify the machine on which the OPC server resides as the second argument. For example:

(Continued on page 4)

Tutorial A —Develop a Simple Server (cont'd)

```
m_opcServer.Connect( "My.OPCServer.1",
" MyMachine" );
```

Note that you should catch any exceptions thrown by Connect in case the connection cannot be established.

Create an OPCGroup Object

In order to access the data on an OPC server, an OPC Automation client must create at least one OPCGroup object. Assuming you store a reference to an OPCGroup in a member variable named m_opcGroup, the creation code looks like this:

```
private OPCGroup m_opcGroup;

...

m_opcGroup = m_opcServer.OPCGroups.Add
( sName );
```

In this case, sName is a string denoting the desired name of the newly-created group. However, you may omit the name parameter if you wish, in which case a name will be generated automatically.

Note that it is possible to instantiate more than one OPCGroup. Your client can create additional groups simply by calling OPCGroups.Add, as shown above.

Add OPCItem Objects to an OPCGroup

Suppose the OPC server's namespace consists of the following items:

```
Tank.Pressure
Tank.Temperature
Tank.Humidity
```

In order to read or write these items you must associate them with an OPCGroup object. This is achieved by first instantiating an OPCItem object for every item of interest (you do not have to use every item in the namespace.) In this example, we will use Tank.Pressure and Tank.Temperature, but we will ignore Tank.Humidity. First, obtain a reference to the OPCItems collection within an OPCGroup:

```
OPCItems opcItems = m_opcGroup.OPCItems;
```

Next, pick a "client handle" for each item. This can be any unsigned integer value, but must be unique across all items used by the client. In this example we will let 1 be the client handle for Tank.Pressure, and 2 will be the client handle for Tank.Temperature. Then the OPCItem

objects are created like this:

```
OPCItem tankPressure = opcItems.AddItem
( "Tank.Pressure", 1 );
```

```
OPCItem tankHumidity = opcItems.AddItem
( "Tank.Humidity", 2 );
```

Read a Group of OPCItems

One way to read items is to call the SyncRead method of the appropriate OPCGroup. This method performs a synchronous read—that is, the method call blocks until the data has been read.

```
int[] ServerHandles;

object[] Values;

int[] Qualities;

System.DateTime[] TimeStamps;

int[] Errors;

OPCItems opcItems = m_opcGroup.OPCItems;
int NumItems = opcItems.Count;
int[] ServerHandles = new int[NumItems];

for( int i = 0 ; i < NumItems ; i++ )
{
    ServerHandles[i] = opcItems
[i+1].ServerHandle;
}

m_opcGroup.SyncRead(
    Source, NumItems, ServerHandles,
    out Values, out Errors, out Qualities,
    out TimeStamps );
```

The data values are returned in the Values array. The Values, Errors, Qualities and TimeStamps arrays are "out" parameters.

Each ServerHandle is a unique identifier that is automatically assigned to an OPCItem when you call OPCGroup.AddItem. The ServerHandles you pass to OPCGroup.SyncRead determine which items are read.

Notice how the ServerHandles array is 0-based, like all C# arrays. However, the OPC DA specification dictates that all OPC DA collections are 1-based. Therefore, within the for loop above, element *i* of ServerHandles corresponds to element *i+1* of m_opcGroup.OPCItems.

Tutorial B — Implement Advanced Features

Overview

This tutorial illustrates more advanced features of SLIK-DAC by building on the material covered in Tutorial A. As an overview, here is what you will be doing:

1. Subscribing to a group
2. Setting a group's update rate
3. Responding to DataChange events
4. Asynchronously reading a group
5. Enumerating OPC servers
6. Enabling COM call tracing

Note: part of this tutorial involves creating callback functions that will be invoked by an OPC server. The OPC server and OPC client are assumed to reside on the same machine. However, if you ever create a client that accesses a remote OPC server, you may need to adjust your DCOM security settings using the Windows utility `dcomcnfg.exe`—otherwise the server may not have sufficient permission to invoke the client's callbacks. Please see the document *DCOM Security – Overview and Setup Guidelines* for a more detailed discussion of DCOM security.

Subscribe to a Group

Rather than repeatedly calling `OPCGroup.SyncRead` to check if a data value has changed, you can let the server notify the client automatically. Every time an `OPCItem`'s value changes, SLIK-DAC can fire a "DataChange" event to which your client may respond; these events are fired on a per-group basis. However, your client will only be notified of events for `OPCGroup` objects that are "subscribed." To subscribe to a particular group, use the following code:

```
m_opcGroup.IsSubscribed = true;
```

where `m_opcGroup` is the `OPCGroup` object to which you wish to subscribe. If you ever want to stop receiving DataChange events for this group, you can set `IsSubscribed` to false.

Set a Group's Update Rate

You set a group's "update rate" in order to control the

minimum period between DataChange events for items within that group—this prevents the client from being overwhelmed in case the values change frequently. The default value of this property is 1000 ms (1 second), but you may modify it at run time.

The code for setting the update rate is simply:

```
m_opcGroup.UpdateRate = n;
```

where *n* is an integer value > 0.

Note that the value you specify may not be supported by the OPC server, in which case the server will use the closest rate that it does support.

Respond to DataChange Events

First, the method to handle DataChange events must be defined. For the purposes of this example, we will name the method `m_opcGroup_DataChange`:

```
private void m_opcGroup_DataChange (
    int TransactionID,
    int NumItems,
    int[] ClientHandles,
    object[] ItemValues,
    int[] Qualities,
    System.DateTime[] TimeStamps )
{
    // do something with the changed data
    values...
}
```

The `ClientHandles` array identifies which `OPCItems` have changed (see Step 4 of Tutorial A.) The `ItemValues` array contains the latest `OPCItem` values; the other parameters are described in section 4.4.6.1 of the *OPC DA Automation Specification*.

Next, the DataChange event must be associated with the method defined above:

```
m_opcGroup.DataChange +=
    new OPCGroup.DataChangeEventHandler
    ( m_opcGroup_DataChange );
```

Now, whenever a DataChange event occurs for any item

(Continued on page 6)

Tutorial B — Implement Advanced Features (cont'd)

in the group `m_opcGroup`, the method `m_opcGroup_DataChange` will be called.

Asynchronously Read a Group

The method `OPCGroup.SyncRead` blocks until the read operation is complete (see Step 4 of Tutorial A). However, this may not always be desirable—particularly if a read operation is expected to take a long time—in which case an asynchronous operation may be more appropriate. This is where `OPCGroup.AsyncRead` can be helpful: in order to read data, the client can call `OPCGroup.AsyncRead`, which returns immediately. Later, when the read operation is complete, a callback is invoked on the client and the client can process the data at that time.

The client is notified that a read operation is complete via an `AsyncReadComplete` event. For a group referenced by the variable `m_opcGroup`, the method to handle an `AsyncReadComplete` event could be defined as follows:

```
private void m_opcGroup_ASyncReadComplete (
    int TransactionID,
    int NumItems,
    int[] ClientHandles,
    object[] ItemValues,
    int[] Qualities,
    DateTime[] TimeStamps,
    int[] Errors )
{
    // do something with the item values...
}
```

Then the `AsyncReadComplete` event can be associated with the above method:

```
m_opcGroup.AsyncReadComplete +=
    new OPCGroup.AsyncReadCompleteEventHandler
    ( m_opcGroup_ASyncReadComplete );
```

Enumerate OPC Servers

You can obtain a list of OPC servers residing on a given machine by calling `OPCServer.GetOPCServers`. For example, if your instance of SLIK-DAC is given by `m_opcServer`, you may use the following code:

```
string[] allServers;
allServers = m_opcServer.GetOPCServers();
```

In this case, each string in the array `allServers` is the ProgID of an OPC server on the local machine. However, you also have the option of specifying a remote machine. For instance, the following will return a list of servers on the remote machine named “MyMachine01”:

```
allServers = m_opcServer.GetOPCServers
( "MyMachine01" );
```

Enable COM Call Tracing

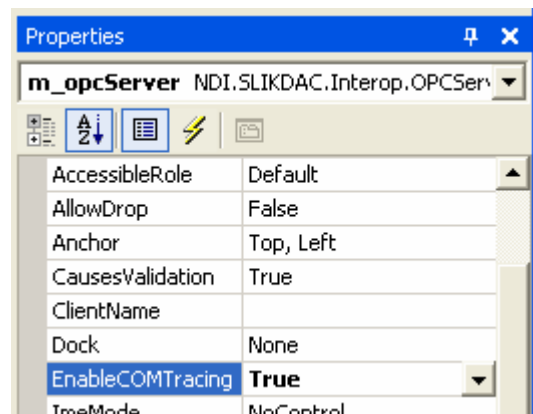
COM call tracing allows you to produce a complete trace log of OPC DA method calls. You may enable this feature at run time. For example:

```
m_opcServer.EnableCOMTracing = true;
```

Alternatively, you can set the `OPCServer` object’s `EnableCOMTracing` property at design time. First, click the SLIK-DAC icon on your form:



Then, toggle the value of `EnableCOMTracing` in the Properties pane:



When enabled, the trace log file is written to “\Program Files\Common Files\Northern Dynamic\NDITraceHook\NDITraceHook.log.” Appendix A of the *SLIK-DAC Reference Manual* describes this feature in detail.

Product Summary Sheet

System Requirements

Operating Systems — Windows 98, Windows NT 4.0 (SP5 or later), Windows 2000, Windows XP, Windows Server, Windows Vista.

OPC Servers Supported — All that are compliant with OPC Data Access 1.0a, 2.0, and 3.0 standards.

Development Environment — OLE automation enabled development tools like Visual Basic and Delphi.

Product Licensing

A SLIK-DAC license is required for each development computer using the ActiveX control to create OPC clients. The OPC client you develop will only run if the executable is generated on a SLIK-DAC licensed computer. Re-distributables can be used in any number of products and no royalty fees are required. Annual developer support contracts are available.

Product Support

Northern Dynamic's dedicated support team is available via telephone and e-mail.

Your new license purchase includes 30 days of unlimited product support via email and telephone. The 30 days start from the date of first contact. The 30 day support period expires if not used within 180 days from the date of purchase.

Northern Dynamic's OPC experts are ready to assist with the development of your OPC server. Comprehensive on-site support and best-practices training are available. Contact a sales representative today for more information.

Proven Interoperability

SLIK-DAC offers guaranteed compliance with the OPC Data Access standard. The reference implementation clients included with SLIK-DAC are certified compliant with the OPC Foundation's Data Access Compliance Test Software.

Ordering Information

You can try SLIK-DAC for free by downloading an evaluation version. When you are ready to purchase, contact Northern Dynamic to obtain your license. Your evaluation software can be converted to a licensed version immediately.

Contact Northern Dynamic for pricing information and a quotation.

telephone	+1 (519) 725-2071
	+1 (888) 265-7345 (toll free in NA)
fax	+1 (519) 725-2072
email	solutions@nordyn.com
website	http://www.OPCexperts.com

About Northern Dynamic

Our team of OPC experts can help you embed OPC technology into your software products, supply robust integration solutions, and provide third party support during your integration efforts.

Visit our website at www.OPCexperts.com to learn more about our OPC solutions.

- OPC Server Toolkits · OPC Gateway · OPC Consulting
- OPC Software Development · OPC Technical Support



Northern Dynamic Inc.
 Suite 3—295 Hagey Blvd
 Waterloo Research & Technology Park
 Waterloo, Ontario
 Canada N2L 6R5
 Tel : +1 (519) 725-2071
 Fax : +1 (519) 725-2072
 Email: solutions@nordyn.com

