NTADS2

Version 3.0.0.0

May 2013

Nick Tucker

www.infinites.com

Contents

Legal

NTADS2 was written and is maintained by Nick Tucker. NTADS is released into the public domain following conditions set forth in the terms of the Mozilla Public License, v. 2.0. You can obtain a copy of this license agreement at **http://mozilla.org/MPL/2.0**

This Source Code is free for commercial and noncommercial work as long as it remains in its unedited form and credit is given in portion to Nick Tucker. There are no warranties either expressed or implied. Use of this library constitutes agreement to this.

If you have changes or wish additions be made to the library, please contact me and I will include them in the next release. Credit will be given.

Many thanks to Peter Baars for his testing.

Overview

In 2002, I needed to be able to program against Active Directory, but I could find very little in the way of example code or help specific to the Delphi IDE. Many hours were spent on the Microsoft site looking at the calls; example code; porting it to Delphi; trying this, that and the other.

The result was the original release of the component. This version still had much of the "testing" code within it which made it very hard to use. A complete rewrite and modernization was in order.

NTADS2 is written with Delphi XE2 and will work with Delphi 2009 and upwards. I have no plans to port the code backwards for the previous versions.

An assumption is made during this document that a) you understand Delphi, and b) you understand Active Directory.

Throughout the component, the path to the object in AD is required. This must be in the format:

Provider+path

For example,

LDAP://DC=MyDomain,DC=Net

WinNT://MyServer/Administrators

Installation

Installation is very typical of any Delphi component installation and should not present any issues. Please be sure to remove any prior version before installing this one. The ActiveDS_TLB of the original version is not compatible with NTADS2. Please make sure all traces of the older version are removed.

1. Copy the ZIP contents to the folder of your choice.
2. Add this folder to the Library search path.
3. Open **NTADS2-All.groupproj**.
4. Right click on the runtime BPL, **NicksADS2R**.
5. Select Build.
6. Right click on the design time BPL, **NicksADS2D**.
7. Select Build.
8. Select Install.

The main component will install along with a number of ActiveX controls imported from the Type Library.

There are a number of samples in the **Demo** folder.

Usage

As with all Delphi projects it's all about the "Uses" clause. Make sure **NTADS2** and **ActiveDS_TLB2** are present.

NTADS2 will automatically be added when you drop the **TNTADS2** component on a form, but the **ActiveDS_TLB2** will not.

Types and Records

tObjType

Each object within AD has a specific object type. These are exposed with the tObjType type:

tObjTypes = *(objUnknown, objContainer, objOrganizationalUnit, objComputer, objGroup, objUser,objPrinter,objPrintQueue, objService, objFileService, objFileShare,objBuiltinDomain, objtrustedDomain, objLostandFound,objsecret,objDomainPolicy,objDFSConfiguration, objnTFRSSettings, objFileLinkTracking,ObjNameSpace,objinfrastructureUpdate,objRidManager,objrpcContainer, objSamServer, objDNSZone, objFSecurityPrincipal,objGPolicyContainer, objdomainDNS);*

TInfo

TInfo exposes the **IADsADSystemInfo** structure and returns the following:

Property	Description
PDCEmulator	Name of the DC in the current Domain that holds the PDCe FSMO role
Schema	Name of the DC in the current Domain that holds the Schema FSMO role
Domain	Distinguished name of the current Domain
SDomain	Short, or NETBIOS, name of the current Domain
Forest	Distinguished name of the current Forest
Machine	Name of the machine where this code is running
Native	Indicates the mode of the current Forest
Site	Name of the Site where the machine is that is running this code
Username	Username of the current user

rObjData

rObjData contains information regarding an object found during the **EnumerateContainer** method and is returned via the **OnEnum** event

Property	Description
Name	String, name of the object
Class_	String representation of the object, “user,” “group,” “computer” etc
GUID	GUID of the object
ADSPath	Distinguished path of the object
Parent	The Parent object
Schema	Schema name of the object. It can could be null.
Index	Index of the class type (tOBJType)

Logging in

You optionally specify login credentials for all the "Open" methods to use. This is accessed via the **NTADS2.Logon**. Specify the user name, password, and set **logon** to true.

Methods

Opening objects

Openobject(ADSPath:String; ClassType: TGuid) : IUnknown

OpenObject is the base method for all of the Openxxx methods and will use the Logon parameters if Logon property is set to true and a username and password have been specified.

Variable	Description
ADSPath	Path to the Container, LDAP://CN=Users,DC=MyDomain,DC=Net
ClassType	As specified in ActiveDS_TLB2, such as IAsdsUser, IAdsGroup, IAdsContainer etc.

Return type is *IUnknown,* and this can be typecast to what is required.

Var User : IAdsUser;

User := OpenObect('LDAP://CN=User1,DC=MyDomain,DC=Net',IAdsUser) as IAdsUser;

OpenContainer(ADSPath : String) : IADSContainer

ADSPath = Path to the Container object to open must be in the format *Provider+Path*.

OpenUser(ADSPath : String) : IADSUser

ADSPath = Path to the User object to open must be in the format *Provider+Path*.

OpenGroup(ADSPath : String) : IADSGroup

ADSPath = Path to the Group object to open must be in the format *Provider+Path*.

OpenIADS(ADSPath : String) : IADS

ADSPath = Path to the IADS object to open must be in the format *Provider+Path*.

Helpers

There are a number of methods included to make some of the basic functionality easier to handle and reduce the amount of recreating code.

GetInfo

Fills in the **TInfo** type structure (See the *Types and Records* section).

LoadEnvironmentStrings : tStrings;

Loads the environmental strings into a *tStrings* object.

GetClassIndex(Class_ : String) : tObjTypes;

Converts a string-based class_ variable to its tObjType.

'User' will return objUser, 'group' will return objGroup and so on.

GetObjectClass

Given the distinguished name of an object, the class type will be returned, 'user,' 'group,' 'computer,' etc.

Function GetClass(ADSPath:String):String

locateobjects(ojbClass,Filter:string) : tStringlist;

A handy method to locate objects of a particular class type.

objClass = the class type to look for, 'user,' or 'group,' for example.

Filter = the name or partial name with an "*" to search for.

LocateObjects('User','Nic');*

ObjectExists

Very simply determines if an object exists given the distinguished name of the object.

Function ObjectExists(ADSPath:String):Boolean;

CorrectDomain(Domain:String) : String;

Takes a domain in the format **MyDomain.Net** and converts it to **DC=MyDomain,DC=Net**.

ToDotname(Path:string) : String;

The exact reverse of **CorrectDomain**, takes a domain in the format **DC=MyDomain,DC=Net** and converts it to **MyDomain.Net**.

NameTranslate(InName:String;FromType,ToType:Integer) : String;

Takes various name formats and converts it to a different format. The table below shows the principal conversion types that can be converted to and from.

Constant name	Description
ADS_NAME_TYPE_NT4	NT Format: MyDomain\thisuser
ADS_NAME_TYPE_1779	LDAP: Distinguished Names
ADS_NAME_TYPE_CANONICAL	Mydomain.com\thisuser
ADS_NAME_TYPE_USER_PRINCIPAL_NAME	thisuser@mydomain.com

To convert an NT style name to an LDAP style would be:

NameTranslate('ISS\Nick', ADS_NAME_TYPE_NT4, ADS_NAME_TYPE_1779);

The object that is being searched for must exist, if not an exception will be raised.

GetServerlist(ADSPath:string) : tStringlist;

Simply returns a list of servers given the container to search in. Results will be returned in the fully distinguished name.

LocateDCinSite(ADSite,Domain:String) : String;

Currently not working.

GetDomains(path:String;ShortName:boolean) : tStringList;

Returns a list of all domains in the current forest.

pathtolist(path:string;var l : tstringlist);

Very similar to the concept of the commatext of a tStringlist but does not suffer from having spaces in the elements. Takes a path such as **CN=Users,DC=MyDomain,DC=Net** and returns a tstringlist.

Object manipulation

DeleteObject

There are two ways to delete and object, one being an overloaded function of the other;

Function DeleteObject(ADSPathtoCont,OType,Name:string):Boolean;
Function DeleteObject(Cont:IadsContainer;OType,Name:string):Boolean;

Variable	Description
ADSPathtoCont	Path to the container, LDAP://CN=Users,DC=MyDomain,DC=Net
Cont	An already opened container
OType	ObjectType, 'user,' 'Group,' 'Computer,' etc.
Name	Name of the object to delete.

A path to a container can be given in the first method, or if a Container is already open, the second method can be used. Obviously, the object to delete must be present in the target container.

CreateUser

Similar to **DeleteObject**, there are two overloaded functions that can be used to create a user. Either specify the path to the Container to create the user, or pass an already opened Container to create the user in.

Function CreateUser(ADSPathtoCont,Name,Pre2K,Password : String):Boolean;
Function CreateUser(Cont:IADSContainer;Name,Pre2K,Password : String):Boolean;

Variable	Description
ADSPathtoCont	Path to the container, LDAP://CN=Users,DC=MyDomain,DC=Net
Cont	An already opened container
Name	Name of the user object to create
Pre2K	Set the same as Name, used for backwards compatibility.
Password	Password for the user

CreateGroup

There are two overloaded methods to create a group:

Function CreateGroup(ADSPathtoCont,Name,Pre2K,
Description:String;GType:tGroup;Security:Boolean):Boolean;

Function CreateGroup(Cont:IADSContainer;Name,Pre2K,
Description:String;GType:tGroup;Security:Boolean):Boolean;

Variable	Description
ADSPathtoCont	Path to the container, LDAP://CN=Users,DC=MyDomain,DC=Net
Cont	An already opened container
Name	Name of the group object to create
Pre2K	Set the same as Name, used for backwards compatibility.
Description	Description of the group
GType	One of: gtGlobal, gtDomain or gtUniversal
Security	If true, a security group will be created. If false, a distribution group will be created

AddToGroup

There are two overloaded methods to add an object to a group:

Function AddToGroup(ADSPathtoGroup,Name:string):Boolean;
Function AddToGroup(Group:iadsGroup;Name:string):Boolean;

Variable	Description
ADSPathtoGroup	Path to the Group, LDAP://CN=G1,CN=Users,DC=MyDomain,DC=Net
Group	An already opened Group
Name	Name of the object to add to the group. Fully distinguished name.

RemovefromGroup

There are two overloaded functions to remove an object from a group:

Function RemoveFromGroup(ADSPathtogroup,Name:string):Boolean;
Function RemoveFromGroup(Group:iadsGroup;Name:string):Boolean;

Variable	Description
ADSPathtoGroup	Path to the group, LDAP://CN=G1,CN=Users,DC=MyDomain,DC=Net
Group	An already opened Group
Name	Name of the object to remove from the group. Fully distinguished name.

MoveObject

Moves an object from one Container to another.

Open the container where you want the object to be moved to, and specify the distinguished name of the object you wish to move.

Function MoveObject(cont:iadscontainer;Name:String):Boolean;

ChangePassword

Changes the password of a user object.

Function ChangePassword(ADSPath,Old,New:String):Boolean;

Domain Admins or those with elevated privileges do not need to supply the old password; this can be left as a blank.

Properties

GetPropertList

Active Directory in its basic format is a database. Each object type has a schema, or set of fields, that can be accessed. Some of these fields are mandatory, some are optional.

Function GetPropertyList(ADSPath : String;Mandatory:Boolean):tStringList;

GetPropertyList will return the properties of the object described in the **ADSPath** variable. Set Mandatory to return either the mandatory properties or the optional ones.

GetPropertyValue

Once you have a property list or you know the property you need information on, this function will return the value of the specified property.

Function GetPropertyValue(Cont:IADS;Name:string):OleVariant

Description := getPropertyValue('LDAP//CN=Nick,DC=MyDomain,DC=Net','description');

It is often the case that even though the property is mandatory it can in fact have no value assigned. In such cases an exception is raised indicating that the value is not in the current cache. Make sure to trap these in an exception block.

The result is in the form of an OleVariant so be sure to understand the type being returned and how to handle it.

Setting a property

There is no method called out specifically to set properties, but this is very easy to accomplish. To set the "description" for a user object is as simple as:

```
User := Openuser('LDAP//CN=Nick,DC=MyDomain,DC=Net);
User.set('description','Author of NTADS!');
user.SetInfo;
```

The same concept applies for all object types and properties. Remember to call **SetInfo** to actually perform the write operation.

Searching and Enumerating

Events

Various events are used during searching and enumerating objects and Containers.

Event name	Description
OnEnum	Called each time the Enumerator finds an object: (sender : tObject ; OBJ : rObjData; Count : int64;Index:Integer;var Abort, Keep : boolean)
OnEnumStart	Called when the enumerator starts: (sender : tObject;Index:Integer)
OnEnumStop	Called when the enumerator stops: (sender : tObject;Index:Integer)
OnSearch	Called when a search starts: (sender : tObject ; Data,OBJClass : String;Count : int64;Index:Integer)
OnSearchStart	Called when a search starts: (sender : tObject;Index:Integer)
OnSearchStop	Called when the search stops: (sender : tObject;Index:Integer)
OnGroupEnum	Called when an item is found in a group: (sender : tObject ; Objectname,Groupname : String; ObjClass,index:Integer; var Abort:boolean)
OnGroupStart	Called when a group enumeration starts: (sender : tObject;Index:Integer)

OnGroupStop	Called when a group enumeration starts: (sender : tObject;Index:Integer)

A common theme amongst these enumerators is the Index. This value is completely up to the programmer and is used **OnSearch, OnEnum and OnGroupEnum** events. When the enumeration is started a number is specified via the Index parameter. Then during the enum events the index can be used optionally in some sort of selection code such as a case statement.

For example calling *ExecuteSearch(1);* will then call the **OnSearch** event each time it finds a match for the search criteria. During the **OnSearch** event you can process the index value via a *Case..Of* operation;

Case Index of
0 : <Perform some function>
1: <Perform some other function>
End

This way a single event can actually perform multiple functions depending on what index number was passed to it.

ExecuteSearch

ExecuteSearch has two overloaded methods.

Function ExecuteSearch(Var ReturnList:tStringlist;Index:Integer):boolean;
Function ExecuteSearch(Index:Integer):boolean;

The first version can return the results in a string list as well as via the **OnSearch** event. Make sure the string list is created before calling, or an exception will be raised.

The second version will only return values via the **OnSearch** event.

In both cases enter the index value will be used in the event.

When searching, search parameters must be specified in order to search against. Use **NTADS2.Search** to specify what to search for.

Name	Description
ObjectCatagory	The object category to search for, such as ‘user,’ ’group,’ ‘computer,’ etc.
SearchPath	The starting container
SearchSubs	Search sub containers
SearchFilter	Optionally specifies a filter, CN=Nick*

The **OnSeach** event has the following properties:
(sender : tObject ; Data,OBJClass : String;Count : int64;Index:Integer)

Name	Description
Sender	The normal RTTI Sender object
Data	A string containing what was found.
OBJClass	String representation of the class, such as 'user,' 'group,' 'computer,' etc.
Count	The number of objects found.
Index	The Index supplied when the ExecuteSearch was started.

EnumerateContainer

Simply stated, this method returns the objects contained within a Container.

Procedure EnumerateContainer(Cont: IADsContainer;Index:Integer;Var ReturnList:tStringlist);

Name	Description
Cont	An already opened Container
Index	Index number to be used within the OnEnum event.
ReturnList	Results of the enumeration. Can be discarded if data is kept via the OnEnum event.

The **OnEnum** has the following properties:
(sender : tObject ; OBJ : rObjData; Count : int64;Index:Integer;var Abort, Keep : boolean)

Name	Description
Sender	The normal RTTI Sender object
OBJ	Data on the object found. See the rOBJData record in **Types and Records**
Count	The number of objects found.
Index	The Index supplied when the ExecuteSearch was started.
Abort	Set to true to abort the operation.
Keep	If true the data will be kept in the ReturnList. Set to false to discard it.

EnumUser

Returns the Group membership of the user specified in the **ADSPath**.

Function EnumUser(ADSPath:string) : tStringlist;

GroupMembers

Enumerates the group specified in **ADSPath**.

Function GroupMembers(ADSPath: String;Index:Integer;Recurse: Boolean): Boolean;

All data is returned via the **OnGroupenum** event with the following properties:
(sender : tObject ; Objectname,Groupname : String; ObjClass,index:Integer; var Abort:boolean)

Name	Description
Sender	The normal RTTI Sender object
ObjectName	Name of the object within the group
GroupName	Name of the group. If Recurse was set to true all nested groups will also be searched. Groupname will then show the name of the nested Group.
ObjClass	Object class of the group member
Index	Index supplied when the enumeration was started
Abort	Set to true to abort the operation

Support

Support is via email or the Infinite solutions web site.

www.infinites.com

Support@infinites.com

Please check often for updates and new sample programs.