.

Application Setup Utility

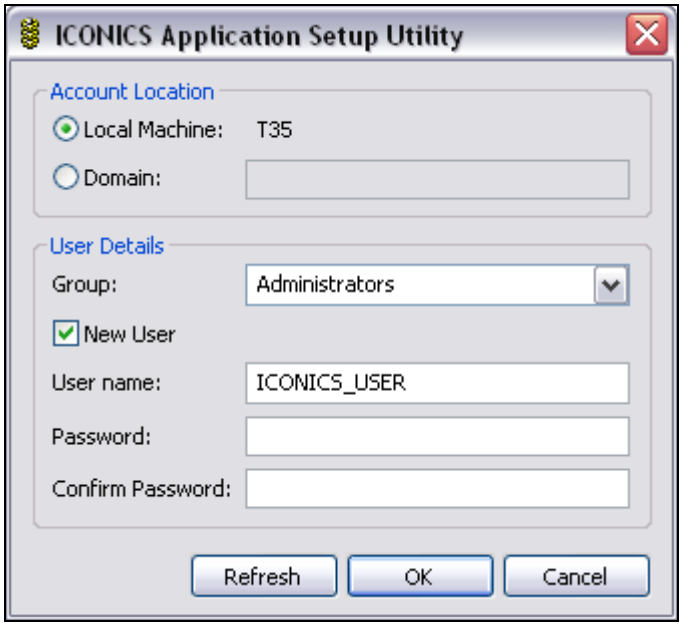
Getting Started

Introduction

The Application Setup Utility creates a new user and group and sets DCOM permissions for all applications in the ICONICS GENESIS32 application suite. The purpose of the application is to create a predictable working security state that will allow the vast majority of GENESIS32 users to successfully run their applications right out of the box. After SETUP runs, GENESIS32 security is appropriately configured for unattended installations, Windows Terminal Server deployments, WebHMI applications, and GENESIS32 applications running in redundant failover configurations.

ICONICS GENESIS32 uses APPSETUPUTILITY during installation to provision DCOM settings, as an optional configuration. If you enable the check box to run APPSETUPUTILITY the program runs twice during SETUP. The first time the utility runs it opens a dialog box briefly and retrieves the user, group, and permission settings it finds in its INI file, APPSETUPUTILITY.INI.

The Application Setup Utility runs in a dialog box during installation to retrieve user settings.



The second time the utility runs it does so with the –NOGUI option as a command line utility without displaying a dialog box. It creates the user, group, and password settings for DCOM you supply in the dialog box. The user is given Launch and Activation as well as Access permissions under DCOM. It also specifies that the GENESIS32 applications that require DCOM run as a service and if necessary restart as a service under NT Service Recovery. Only the GenBroker service is turned on automatically by APPSETUPUTILITY and requires a system restart to start the first time. The other additional services are left in the manual state and require that you turn them on as required.

Developers of GENESIS32 application-based solutions can use APPSETUPUTILITY as a simple method for assigning default DCOM settings directly from a script or a program, creating any number of users, groups, and

DCOM security permissions for them. As an option the utility can specify that one or more GENESIS32 application record its events into the Windows Application Event log.

Settings for DCOM are stored in the Windows Registry for the local system and for systems that are part of a workgroup. APPSETUPUTILITY uses Windows NT Authentication, so when a domain is specified DCOM settings are a associated with a users account in the Active Directory. DCOM settings are applied based on group membership, as they would be if you assigned security in each of the ICONICS GENESIS32 applications individually.

This help file details how APPSETUPUTILITY operates, where it stores its settings, how you can modify those settings, and all of the various options that you can apply should you choose to use APPSETUPUTILTY as part of your application.

Contents

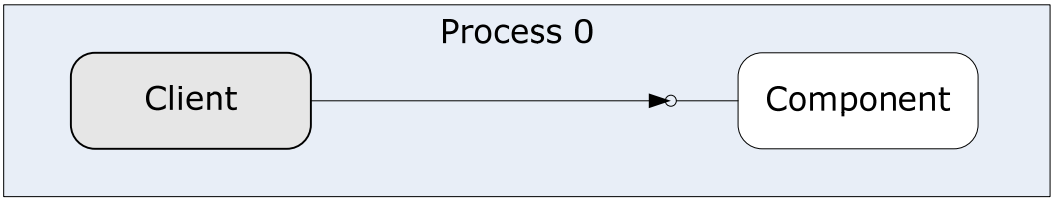
DCOM Security Settings

About DCOM

Microsoft Distribute COM (DCOM) is an extension of the Microsoft Component Object Model (COM) for inter-application communications. DCOM abstracts the location of an object so that it can request data from a DCOM server in another location using a client server model. Although DCOM has been surpassed by other programming models such as Microsoft .NET Framework, during the decade in which DCOM was widely implemented it found support from a a great number of vendors in a wide varieties of applications. Therefore, DCOM will continue to be used for many years to come.

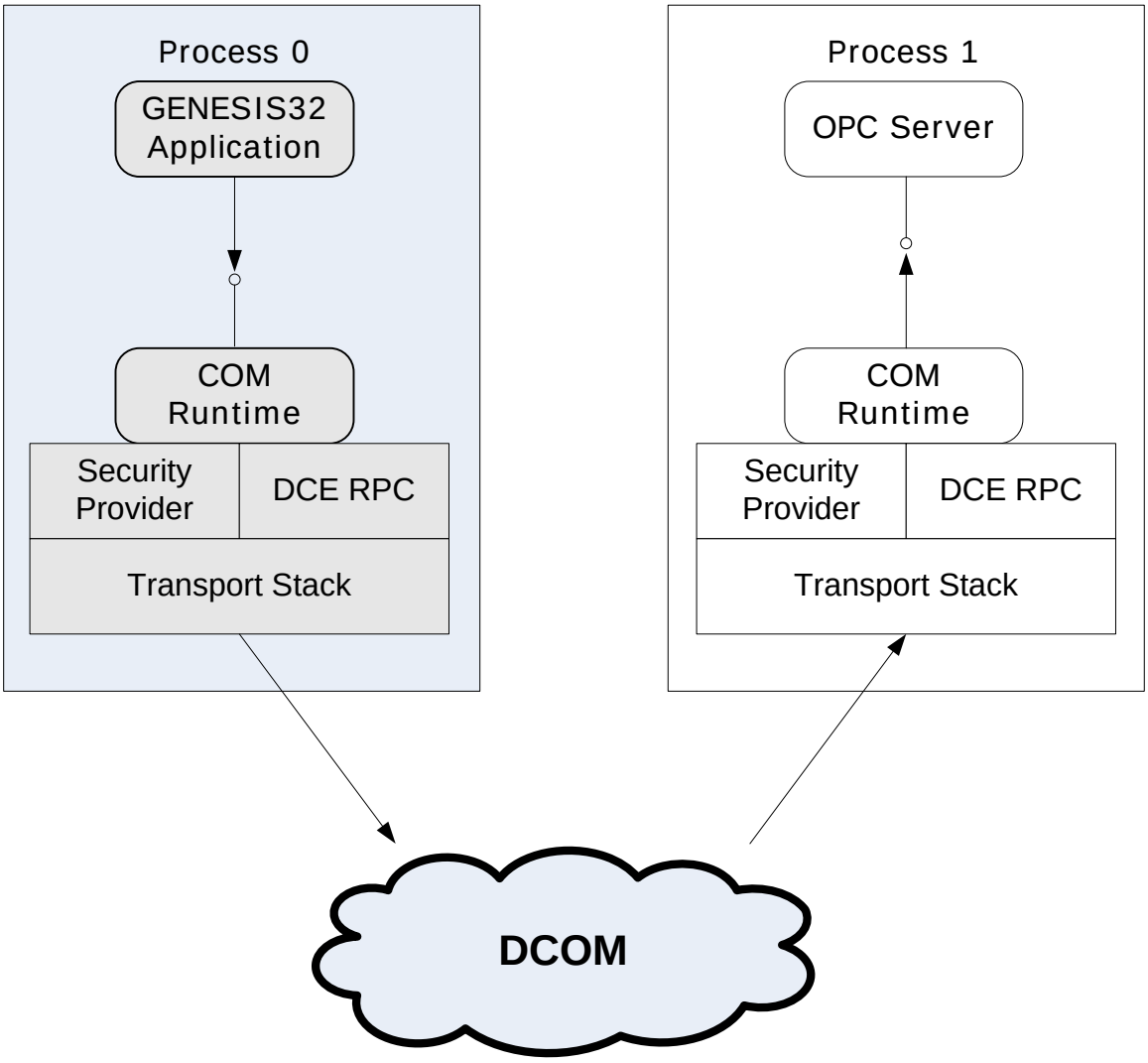
In DCOM a client calls the methods of a component. Both client and components can be written in any of several programming languages provided such as Visual Basic, C++, or Java. If the client and the component are part of the same process, perhaps on the same system, communications is direct without the use of an intermediary, as shown below. The situation below would apply to GENESIS32 and its data source (an OPC Server, for example) installed on a standalone system.

COM messaging in a single process



In most instances GENESIS32 applications are installed on one system and the data source such as an OPC Server or database is installed and running on another system. In this instance the client and the component are running in separate processes. A COM message and the data returned from the component cannot be direct and transparent, and requires mediation. A COM Runtime module with its own security provider and a Remote Procedure Call module creates DCOM network packets that are in the DCOM protocol standard. Wire level transport can be over the network, or across an intranet or the Internet.

COM messaging between different systems.



There are several implications that arise out of the architecture shown above. As shown the communication is one way from the GENESIS32 application to the OPC Server. Security from client to server applies to DCOM communication for transport in that direction. When the OPC Server returns information and the traffic is reversed the OPC Server takes the role of the client and must be validated by the GENESIS32 application in turn. So DCOM's security mechanism is unidirectional and requires two separate authentications. Also, since DCOM transport is over a network protocol it will work with different transports.

A DCOM connection is inherently transient and requires a management system to ensure transport. DCOM can manage connections from components to one client, and DCOM creates a reference count when a connection is made to the client. When the client breaks the connection the reference count is decremented and when that count is zero the component is then free to respond to other requests for service. DCOM uses a PING mechanism to determine if the connection is still live, and after three PINGs over 6 minutes without a response considers the connection broken and decrements the reference count.

Note: A detailed article on DCOM may be found at: http://msdn2.microsoft.com/en-us/library/ms809340.aspx.

The GenBroker application was developed to keep track of the pool of components that clients connect to. DCOM brokers are used to measure the load on servers and to detect when a server is added or removed. Request for services can be routed to a server with the lowest load, and once the connection is made the broker's role in the communications is finished.

DCOM Security Settings

One of DCOM's core functions is to manage the security of communication between client applications and components. GENESIS32 SETUP installs the components you specify, and components that require DCOM communications are registered. Those components show up in the Components Services utility as services that you can run. Security is determined by Windows NT authentication in domains, or by local user accounts for standalone systems or systems that are part of workgroups.

The main utility of opening the Components Services utility is to determine if the components that are listed in the AppSsetupUtility.INI file are installed and that the correct user, group, and assigned permissions are allowed. Whatever user ID you put into the dialog box should be an administrator with Access and Launch & Activate permissions. To see if the application is actually running as a service or to turn that service on you would use the Services Control Panel to perform those tasks.

You can determine if a GENESIS32 application was installed and properly registered by opening Component Services.

To view DCOM settings for an application:

1. Click **Start**, type **DCOMCNFG** in the **Run** text box and press the **Enter** key.

   Or, click **Start**, **Control Panel**, **Administrative Tools**, and select the **Component Services** command.

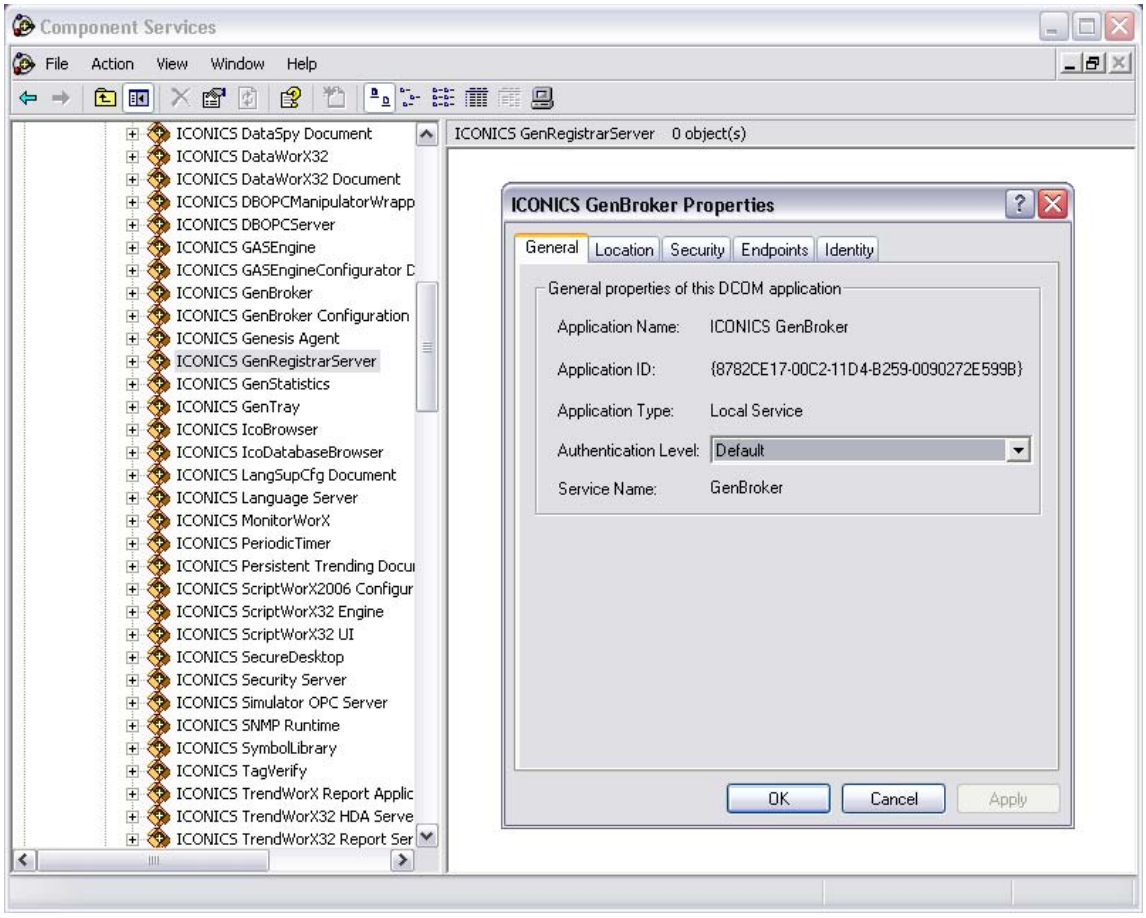
2. Navigate to the following node in the tree control: **Component Services\Computers\My Computer\DCOM Config** node.

3. **Scroll down** the list to view the ICONICS GENESIS32 DCOM registrations.

4. Right click on the application of interest and select the **Properties** command from the context menu.

   The General tab of the Properties dialog box for a DCOM application and the Component Services utility is shown below.

   The Component Services utility



On the General tab is displayed the following pieces of information: the name of the application, the unique DCOM registration ID, the authentication used by DCOM for packets, and the name used in the Services control panel identifying the application's service. When you are on a local system examining a local application the path to the executable is displayed as well.

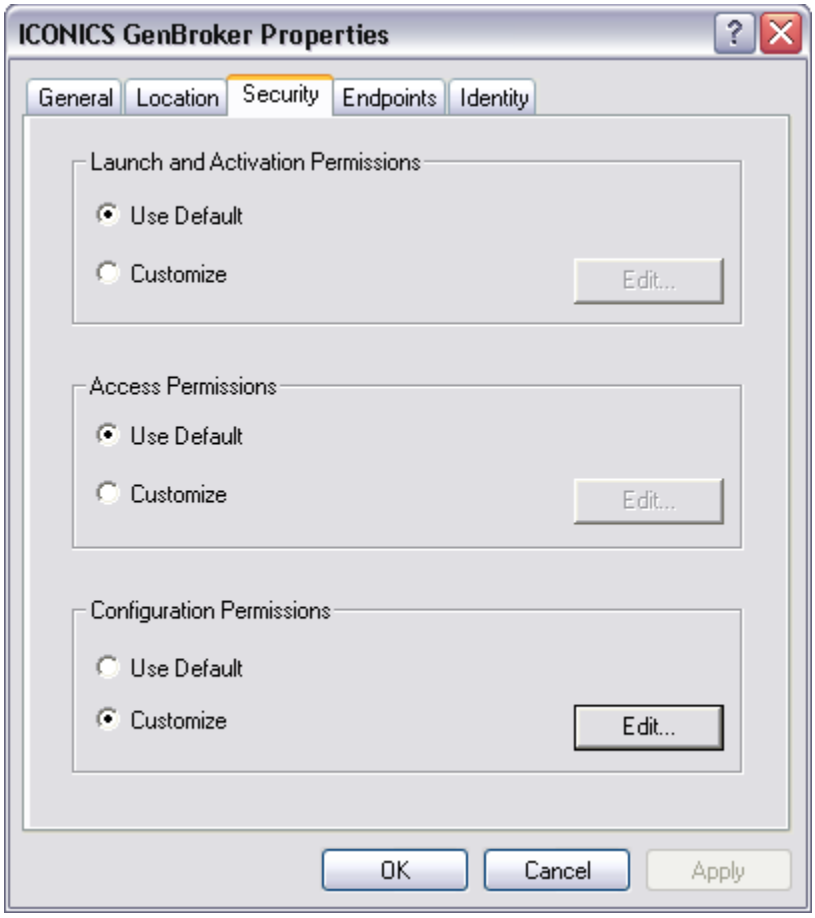
The most important tab from the standpoint of getting ICONICS GENESIS32 applications to function correctly are the settings found on the Security and the Identity tabs of the Properties dialog box, as shown below. The Security tab contains the settings users that are authenticated. The Security properties for a GENESIS32 application depending on DCOM should be set to:

Use Default for Launch and Activation Permissions

Use Default for Access Permissions

Customize for Configuration Permissions

The Security tab for the DCOM Configuration Properties dialog box for GenBroker.

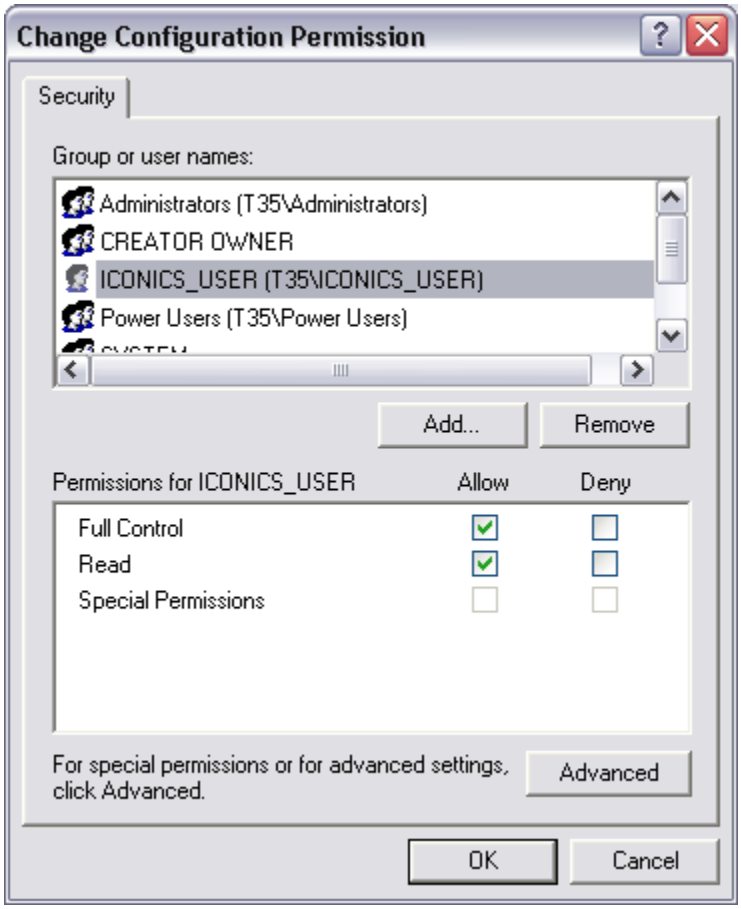


To check that the user is added correctly to DCOM settings:

1. Click on the **Edit** button in the Configuration Permissions section on the Security tab of the DCOM Configuration Properties dialog box, as shown above.
2. Scroll if necessary and **highlight the user ID name** that you created.

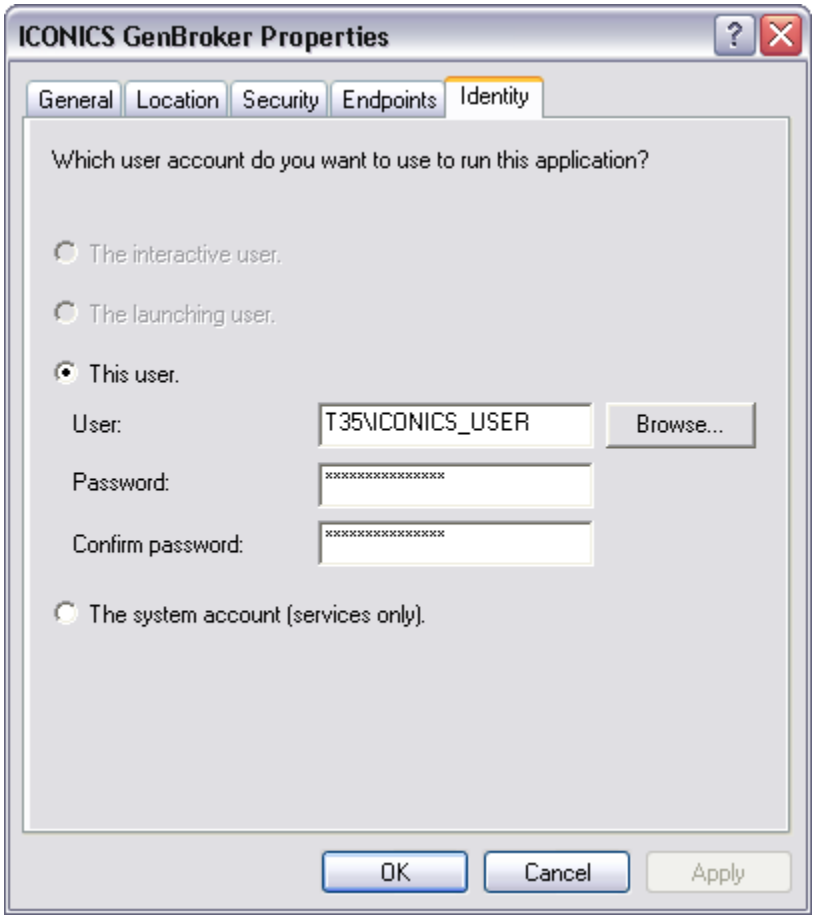
   As shown in the figure below the user created was ICONICS_USER.
3. Check that the user ID has **Full Control** and **Read** permissions enabled.
4. Click the **OK** button to close the dialog box when you are finished.

To view user permissions, open the Change Configuration Permissions dialog box.



You can use the Identity tab of the DCOM Properties dialog box to change the password for the user ID, as well as to determine the current user ID that is logged on to the system. As you can see in the figure below the ICONICS_USER account is the current account.

To change the password associated with a User ID, open the Identity tab of the DCOM Properties dialog box.



Services

An operating system service is a program or routine that provides support for other programs, particularly at a low level for communication with hardware. Many of ICONICS GENESIS32 programs are installed as services that you can run when required. The goal of APPSETUPUTILITY is to setup the programs you specified as services, set the User ID, group, password, and privileges and prepare the system to run your GENESIS32 applications.

As configured, APPSETUPUTILITY only configures GenBroker as a service that is Automatically On. All of the other applications that were specified has the correct user and permissions, but are set to manual. Therefore, you will need to **turn on any service that you require** to run to support your application. You can use the Services Control Panel to turn a service on, or do so at the command line.

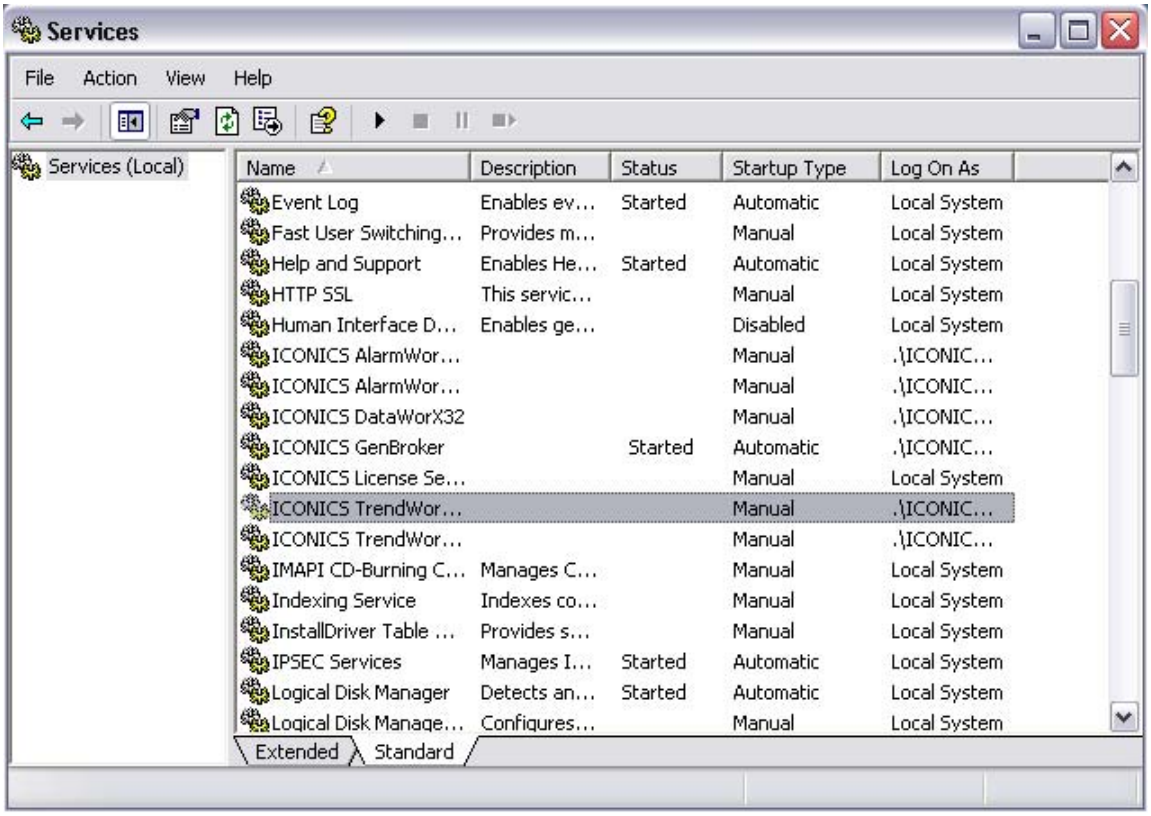
To examine a service or change its state:

1. Click **Start**, **Run**, then type **SERVICES.MSC** into the Run text box, then press the **Enter** key.

   Or click **Start**, select **Control Panel**, double click on **Administrative Tools**, and then double click on the **Services** Control Panel icon.

2. Check that GenBroker is Started and runs as an Automatic Startup Type.

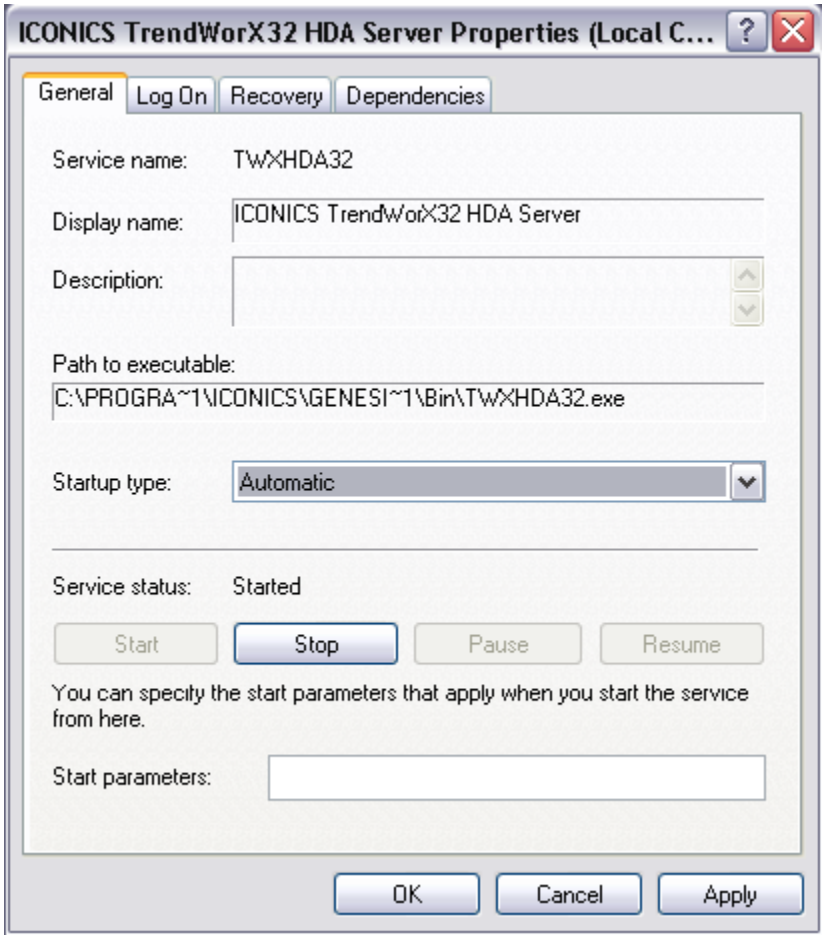
   After APPSETUPUTILITY runs GenBroker should have an Automatic Startup Type and after your first system reboot should show a Status of Started.



3. **Double click** on the application that you want to run as a service.
4. On the General tab of the Services Properties dialog box change the **Startup type** to **Automatic** and click the **Start** button.

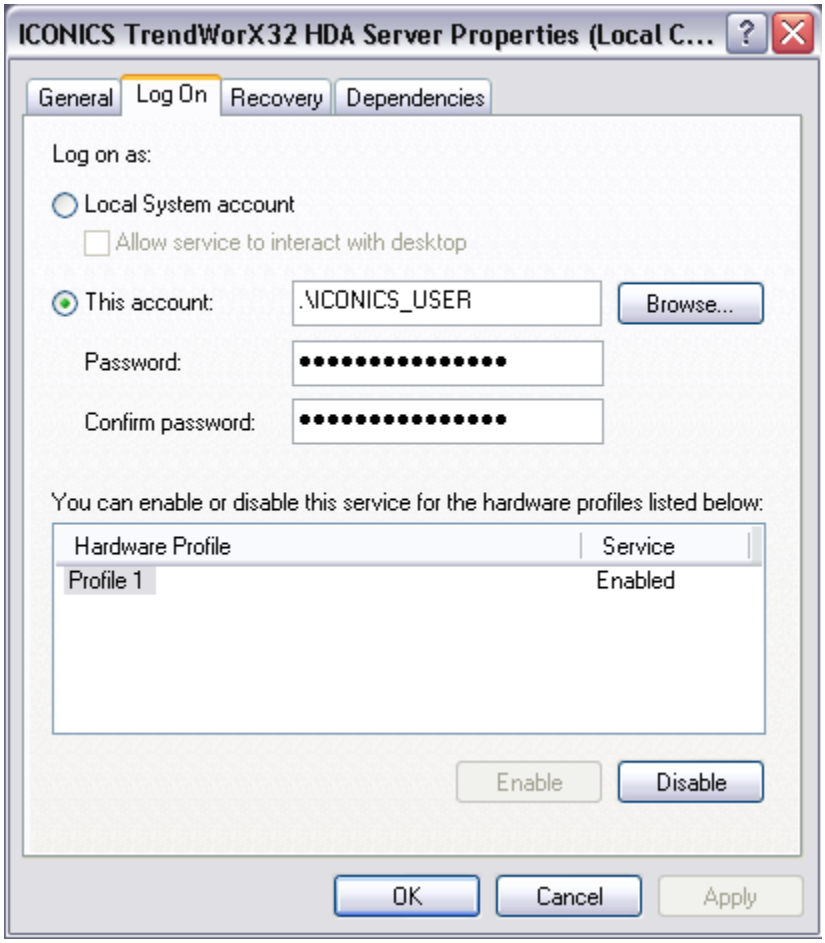
   Depending upon the version of Windows you have installed GENESIS32 on you may be required to restart Windows in order to enforce your selections.

A Service Properties dialog box.

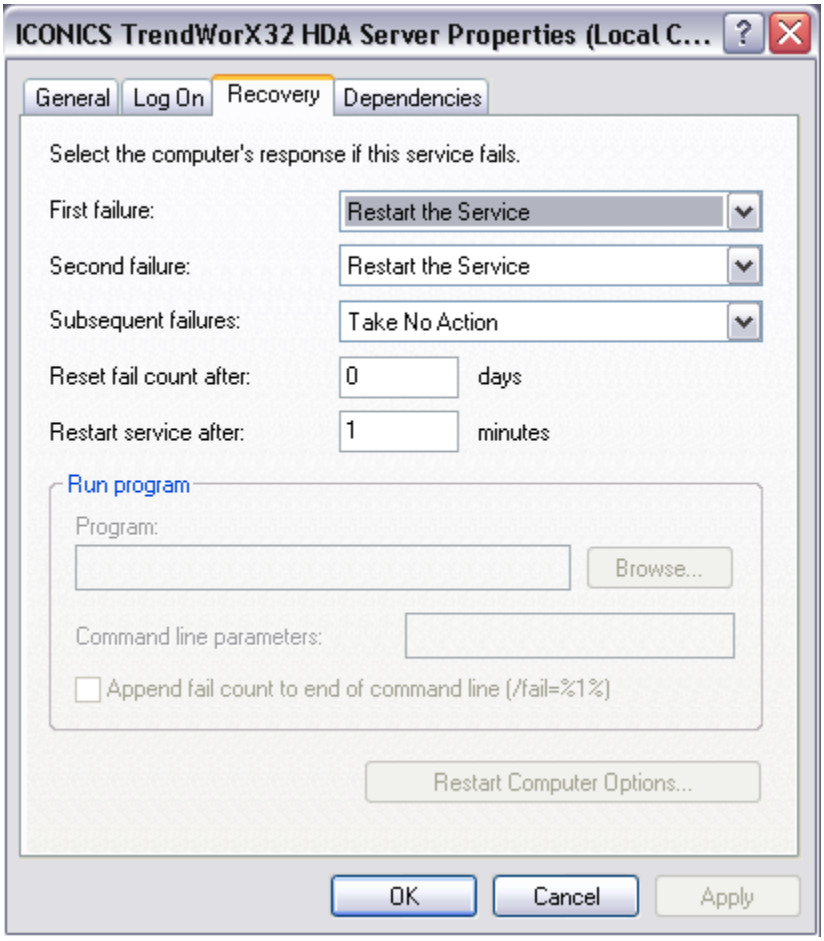


5. Click on the **Log On** tab (see below) to determine the User ID used for the service, it should match the User ID found in the Component Services dialog box.

   The User ID currently logged on is displayed in the Log On tab.

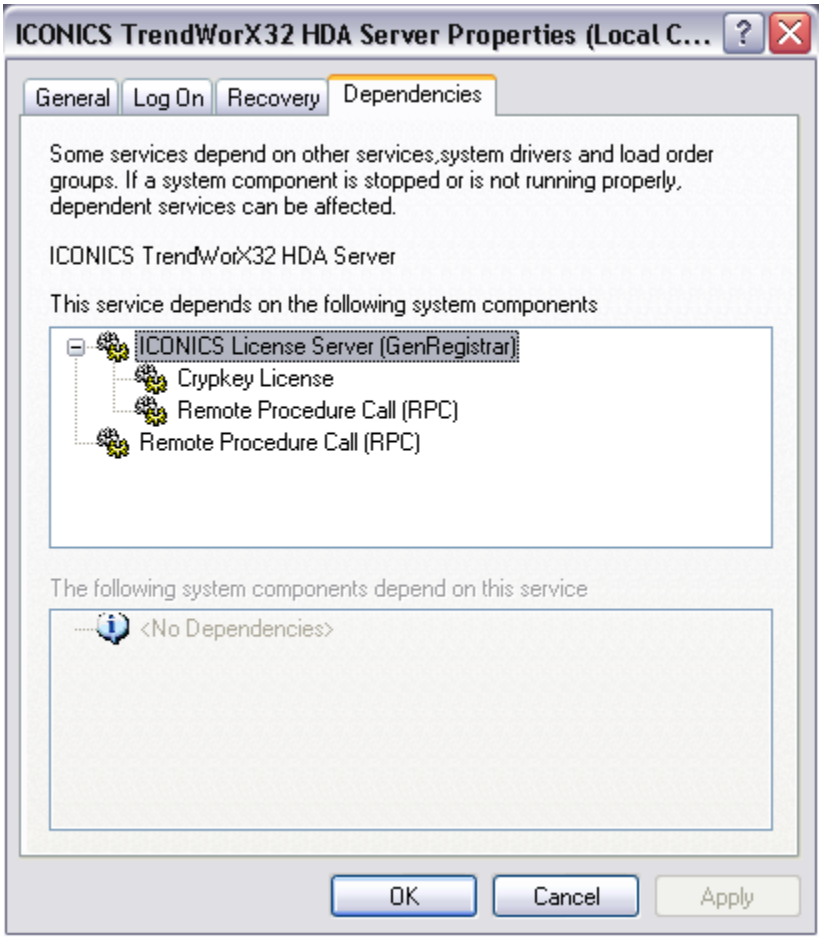


6. Click on the **Recover** tab to check that APPSETUPUTILITY has correctly set up the recovery behavior for your service. As shown in the figure below the First failure and Second failure should be set to **Restart the Service**, while Subsequent failures should be set to **Take No Action**.



7. APPSETUPUTILITY also configures the services that a service is dependent on. If a service doesn't run correctly check the Dependencies tab to see if it matches what you see in the `Module` section for that application in the APPSETUPUTILITY.INI file.

   To check that dependencies are set correctly click on the Dependencies tab and confirm that the listed dependencies match the listing in the APPSETUPUTILITY.INI file.



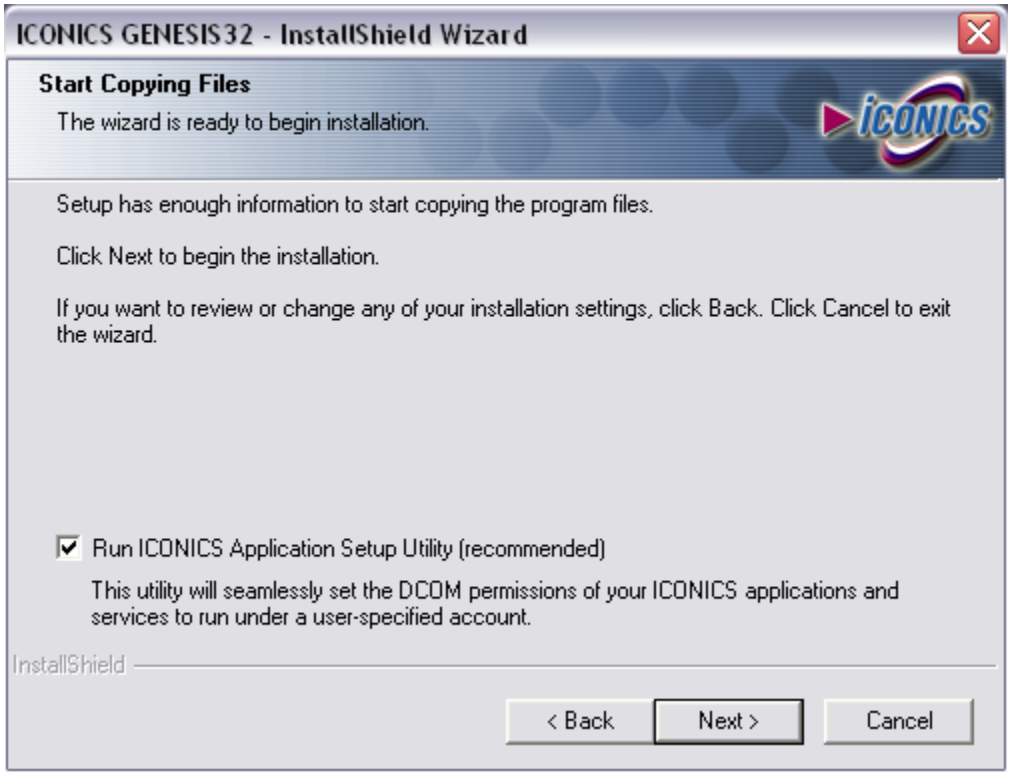
8. Click OK to close the Services dialog box when you are through.

Using AppSetupUtility

Application Setup During GENESIS32 Installation

The Application Setup Utility runs during the installation of GENESIS32, MMX, and WebHMI if you enable the Run ICONICS Application Setup Utility (recommended) as shown below.

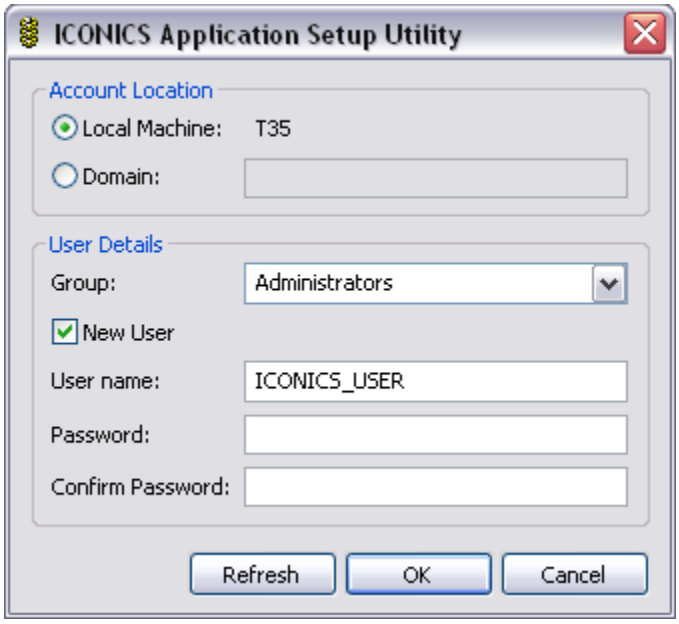
Enable the Run ICONICS Application Setup Utility check box to have the utility set default DCOM settings for your GENESIS32 applications.



The utility reads the `AppSetupUtility.INI` file and populates the dialog box shown below. After you enter a password and click OK the utility does the following things:

- The applications listed as modules are registered as DCOM Components.
- The application's identification CLSID is written into the Windows Registry
- The User ID, group, and password is created if you select the New User check box.
- The group and password for an existing User ID is modified if necessary.
- The User account is given Access as well as Launch and Activate permissions and assigned to each of these DCOM components.
- The applications are registered as services and set to Startup Type Manual.
- GenBroker is set to a service, configured with Startup Type Automatic and after a reboot its state is set to Started.
- Recovery behavior is set to restart the service on the first two service failures

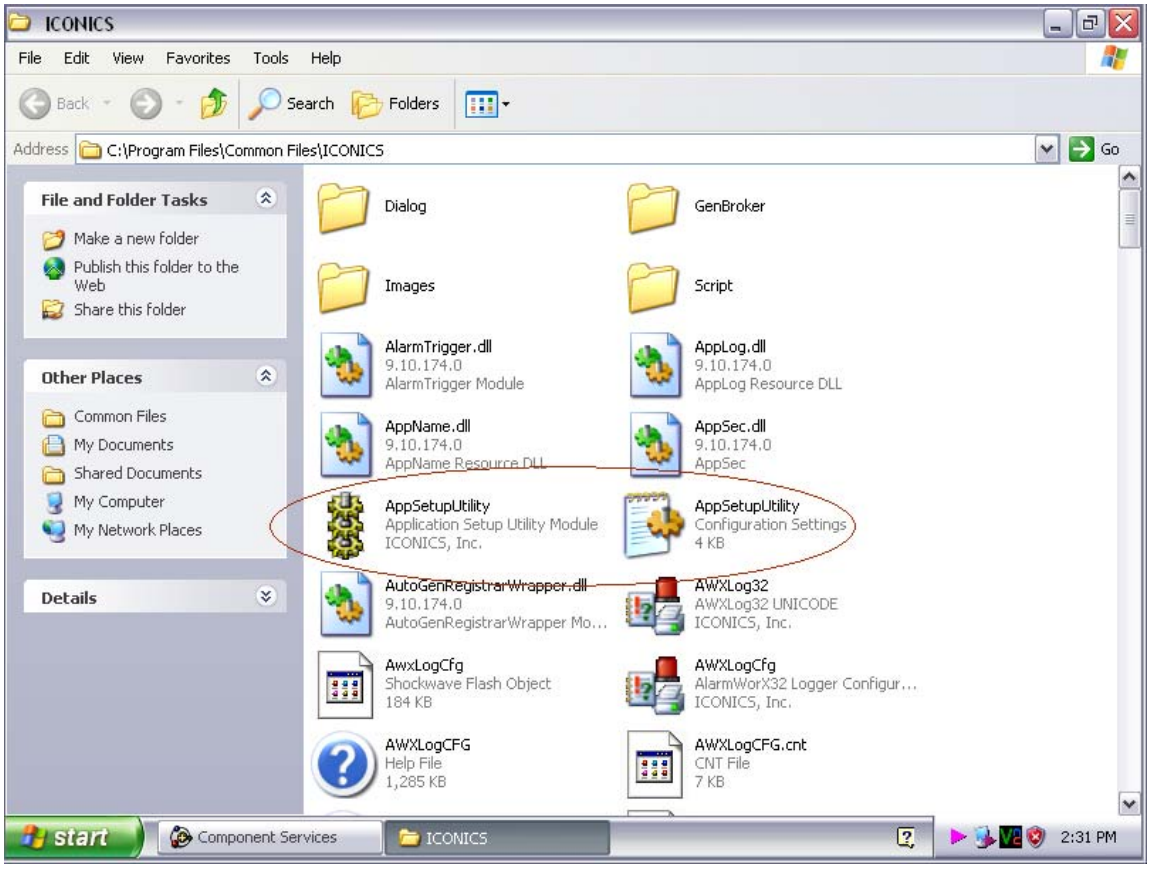
The Application Setup Utility's graphical mode.



Graphical Mode

The Application Setup Utility is installed by GENESIS32 SETUP program into the ICONICS Common Files directory. To run this utility you must switch the current directory to the ICONICS Common Files directory to execute the program. The program requires that its INI file be located in the same directory as the executable file.

The `AppSetupUtility` (left) and its associated INI file are shown in their default folder below.



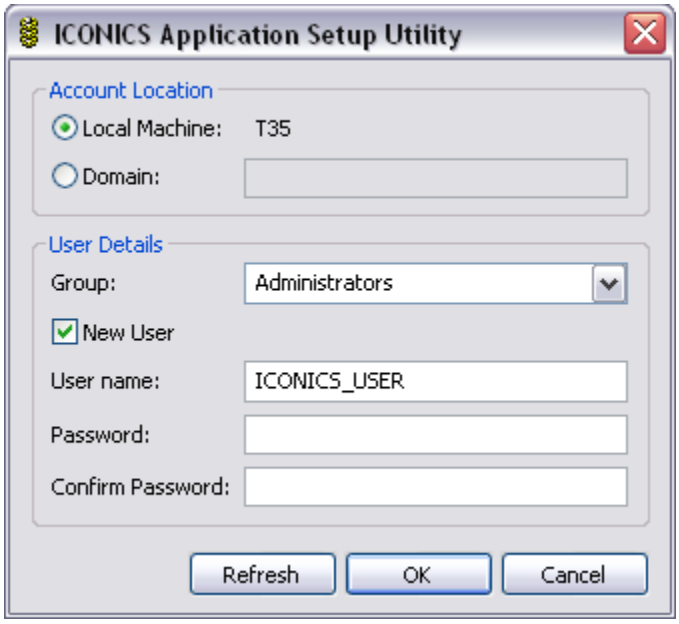
If you double click on the program icon the `AppSetupUtility` dialog box appears on your screen, as it did briefly during GENESIS32 installation.

To return GENESIS32 applications to the same (default) settings the suite had after installation do the following:

1. Launch Notepad and open: `C:\Program Files\Common Files\Iconics\AppSetupUtility.INI`.
2. Examine the INI file and make sure that has the same entries that it had right after installation. The file is reproduced in its entirety in the topic AppSetupUtility INI File.

   If you want to omit or add modules as NT Services and DCOM Servers, edit the Modules section and add the appropriate settings. Settings in the INI file are described in the AppSetupUtility .INI File section.

The Application Setup Utility's graphical mode.



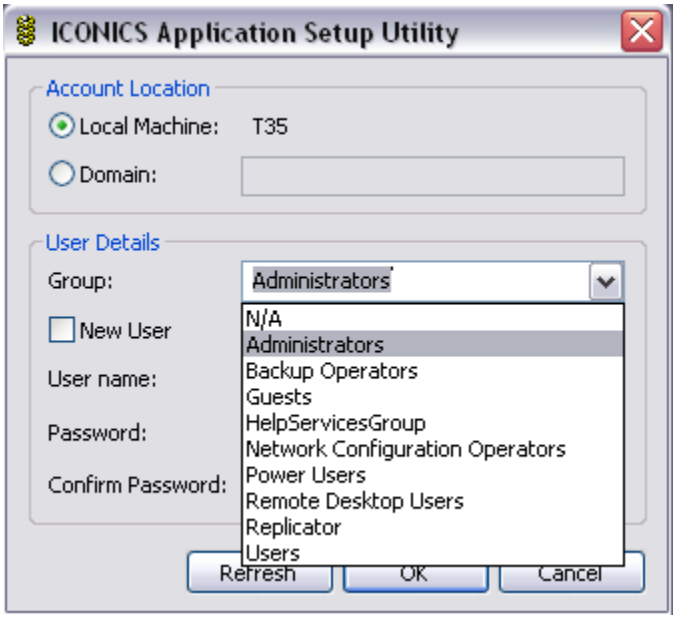
3. For a local account leave the Local Machine radio button selected, or for NT Authorization in a domain click the Domain radio button and enter the Domain Name.

   `AppSetupUtility` reads the INI file's [Options] section for the User Name.

4. The default user group is Administrators, but you can set the user group to any group name you wish by clicking on the Group drop down list, as shown below.

   Keep in mind that the purpose of this utility is to allow users guaranteed access to DCOM components, so be sure that the group name you provide has the appropriate DCOM Access and Launch and Activate privileges. You can use the Component Services utility to examine user and group DCOM permissions.

   The Application Setup Utility populates the Group list with available group names.



5. If you wish to use the default user name disable the New User check box.
6. To create a new user, enter the User Name, make sure the New User check box is enabled, enter a password into the Password and Check Password text boxes, and click the OK button

   When you click OK button the user name is created if New User check box was enabled and that user name is written to the [Options] section of the INI file.

   If the New User check box was disabled and the user name already exists that user name is written to the INI file.

   If the New User check box was disabled and the user name doesn't exist then the user name permissions will be local and not be allowed in a domain.
7. After the utility runs it posts the alert box shown below if it is successful. Click the OK button to acknowledge the alert box.

   Upon completion `AppSetupUtility` posts the following alert box.

CAUTION: Although you can accept the default user name and leave the password blank, ICONICS strongly recommends that you **enter a secure password** in order to avoid the situation where a administrator account can be accessed by any connected user.

`AppSetupUtility` proceeds to set the DCOM permissions so that this user can both Access and Launch components as well as Activate them. The enumerated

modules are either set as NT Services/DCOM Servers or as just DCOM Servers and services are set to Automatic and On by default.

NOTE: In order to have the registered services run after installation you are required to restart the server or manually turn those services on the first time.

The Recovery settings for NT Services as configured by AppSetupUtility is to have the system attempt to restart the service twice in case of failure and if unsuccessful then to not attempt to restart the service the third time.

When the AppSetupUtility runs it creates a user with unlimited capability to access components through DCOM. This is done to ensure that the system works correctly out of the box. For most applications you will need to enforce the security settings that you wish users to have by creating the appropriate user accounts and associated access rights.

Command Line Interface Settings

However, you can also call this application from the command line and customize the operation of the utility with a number of switches

To run AppSetupUtility from the command line:

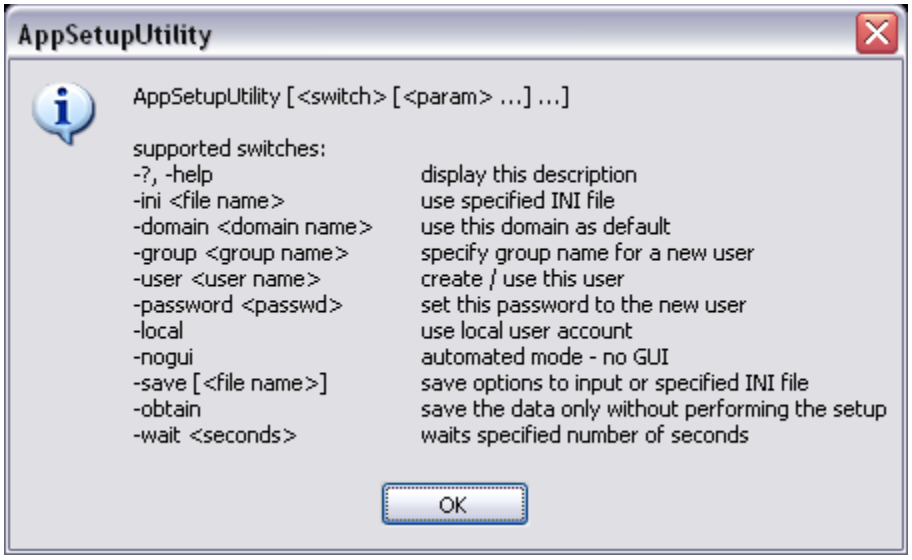
1. Click **Start**, **Run**, enter CMD into the Run dialog box and press the **Enter** key.
2. Use the **change directory** command as follows to switch to the directory containing the AppSetupUtility EXE file:

   ```
   CD C:\Program Files\Common Files\Iconics
   ```

3. If you enter the AppSetupUtility command without any switches at the command line the utility's Graphical Mode dialog box appears.
4. To view help for the AppSetupUtility command line switches, modify the command with the Help switch by entering: AppSetupUtility /?

   The CLI posts the help dialog box shown below.

   Command line help is provided as a dialog box with a list of available switches.



In most instances developers or OEMs will want to use the AppSetupUtility in its silent mode (-NOGUI) so that the application runs from a script or program automatically. You can use the other switches to pass parameters to this utility.

In addition to the explanation provided in the dialog box above, here are some additional comments regarding the program's switches:

- To use a different INI file, enter that file name as a switch. The INI file must be located in the ICONICS Common Files directory.
- In the Graphical Mode the user, group, and password are written to the Active Directory for a specified domain.
- For an existing user the password you enter will overwrite the existing password for that account, if necessary.
- For standalone installations or workgroups use the –LOCAL switch to create local accounts.
- The –GROUP switch is normally set to Administrator during development and then modified to something more restrictive once development has been completed.
- If you specify a user or group that is different than the contents of the AppSetupUtility.INI file you can use the –SAVE switch to write their values back to the INI file.

AppSetupUtility .INI File

The Application Setup Utility stores and retrieves setting from an INI file that is located in the same directory as the APPSETUPUTILITY.EXE file. The default location for both the EXE and the associated INI file is:

```
C:\PROGRAM FILES\ICONICS\COMMON FILES
```

The APPSETUPUTILITY.INI file is organized into sections:

- [Options]. The Options section lists the user name. During SETUP this user name is entered into the APPSETUPUTILITY dialog box. If a different user name is entered by the user, then that user account is created and stored in the INI file Options section. There is no user group setting, the current version of APPSETUPUTILITY always assigns the Administrators group to any user account that is created.

NOTE: Any line in APPSETUPUTILITY.INI that starts with a semicolon is a comment.

- [Modules]. The Modules section contains an editable list of the applications that use DCOM security settings for authentication.

  If you don't install or use an ICONICS application you can remove that entry from the Modules section. Alternatively, if you want to use a third party OPC server you can add the name of the server(s) to the Modules section and then add the required DCOM a settings section for that server in the NT Services or DCOM Servers section; or create your own section with a commented line and add the settings below that line.

  APPSETUPUTILITY loops through the INI file one module at a time reading the settings for each application in the sections below. Class ID values are written to the Windows Registry.

- [<*ModuleName*>]. Every application that uses APPSETUPUTILITY to set initial DCOM settings has those settings contained in their own section. Depending upon the type of application and how the developers want the application to service or respond to DCOM requests different settings are used. As an example, consider the settings section for AlarmWorX Server which is duplicated below:

```
[Awx32Svr]
ModuleType=3
CLSID={BA1F0A01-D560-11d1-84DF-00608CB8A7E9}
DisplayName=ICONICS AlarmWorX32 Server
Dependencies=RpcSs,GenRegistrar
```

ModuleType identifies how APPSETUPUTILTY will configure the DCOM server.

The value 0 would be used if you wanted to start an application, say for a third party program.

The value 1 indicates that the program is registered as a DCOM Server.

The value 2 indicates that the program is an NT Server.

The value 3 is actually the combination of 2+1 and indicates that the program is both an NT Service and a DCOM Server. That is the case for AlarmWorX32 Server, as is shown in the sample above.

CLSID is the Class Identification number for the application's binary that is stored in the Registry. APPSETUPUTITITY uses the CLSID to find that Registry entry and add the additional values for the NT Service Name and any applications or services that that service is dependent upon. When a clients calls a DCOM component the CLSID is the unique identifier which determines which component's method is called by the client in the DCOM model.

DisplayName is the name of the service as shown in the Services Control Panel.

Dependencies are the services that must be running in order for this service to start and run. Nearly all of the modules that run as an NT Service require the use of Microsoft RPC, which is the Windows component program RPCSS.EXE. This program implements the connection security for DCOM, essentially brokering the connection between the client and the component running on the server. The word "server" here refers to an architectural context and both can be on a local system or on different systems.

The default settings for APPSETUPUTILITY.INI are shown below.

```
;AppSetupUtility.INI

[Options]
user=ICONICS_USER

[Modules]
```

```
AWXLog32=AWXLog32
Awx32Svr=Awx32Svr
DataWorX32=DataWorX32
GenBroker=GenBroker
TWXHDA32=TWXHDA32
TWXLOG32=TWXLOG32
IcoTrendReportHost=IcoTrendReportHost
UDMRuntime=UDMRuntime
DBOPC=DBOPC
GASEngine=GASEngine
GenAgent=GenAgent
LASEngine=LASEngine
MonitorWorX=MonitorWorX
SecureDesktop=SecureDesktop
Security=Security
SimOPC=SimOPC
SNMPRuntime=SNMPRuntime
UTMRuntime=UTMRuntime
SNMPRutimeService=SNMPRutimeService

; if you have MMX installed then remove the following line with

[MMX Modules]
AWXMMX32=AWXMMX32
MMXCall_In=MMXCall_In
MMXFax=MMXFax
MMXMail=MMXMail
MMXMessenger=MMXMessenger
MMXPager=MMXPager
MMXSnapshot=MMXSnapshot
MMXSound=MMXSound
MMXVideo=MMXVideo
OPCEnum=OPCEnum

; NT Services ////////////////////////////////

[AWXLog32]
ModuleType=3
CLSID={8BB8749F-9142-11D2-A541-00104B0D13C3}
DisplayName=ICONICS AlarmWorX32 Logger
;Dependencies=Crypkey License,GenRegistrar,RpcSs
Dependencies=RPCSS,GenRegistrar

[AWXMMX32]
ModuleType=3
CLSID={6DDED201-E48C-11D3-A624-00104B0D13C3}
DisplayName=ICONICS AlarmWorX32 Multimedia Server
Dependencies=RPCSS,GenRegistrar

[Awx32Svr]
ModuleType=3
CLSID={BA1F0A01-D560-11d1-84DF-00608CB8A7E9}
DisplayName=ICONICS AlarmWorX32 Server
Dependencies=RpcSs,GenRegistrar

[DataWorX32]
ModuleType=3
```

```
CLSID={0E8B028E-8569-43F3-AE5E-50072BD2ED1E}
DisplayName=ICONICS DataWorX32
Dependencies=RpcSs,GenRegistrar

[GenBroker]
ModuleType=3
CLSID={DB45904D-1C49-11D4-B273-0090272E599B}
DisplayName=ICONICS GenBroker
Dependencies=+TDI

[TWXHDA32]
ModuleType=3
CLSID={750B7441-D92C-11d1-8F96-00A024C11193}
DisplayName=ICONICS TrendWorX32 HDA Server
Dependencies=RPCSS,GenRegistrar

[TWXLOG32]
ModuleType=3
CLSID={1D48BD20-2259-11D3-8F32-00105A29CD0E}
DisplayName=ICONICS TrendWorX32 SQL Logger
Dependencies=RPCSS,GenRegistrar

[IcoTrendReportHost]
ModuleType=3
CLSID={35865765-66BD-11D3-8F96-00105A29CD0E}
DisplayName=ICONICS Trend Report
Dependencies=RPCSS,GenRegistrar

; DCOM Servers ////////////////////////////////

[UDMRuntime]
ModuleType=1
CLSID={22DFD824-A66D-4C66-B186-7E8ED07C9CAB}

[DBOPC]
ModuleType=1
CLSID={419B4AEC-0CA9-4B2E-9FBE-A7E1AE856C0E}

[GASEngine]
ModuleType=1
CLSID={979D65FE-F995-4503-8440-546FDFFC5061}

[GenAgent]
ModuleType=1
CLSID={9735149F-C752-11D2-B1A4-0090272E599B}

[LASEngine]
ModuleType=1
CLSID={44F19677-636C-4c3a-BAED-0427FCE033CB}

[MonitorWorX]
ModuleType=1
CLSID={591D9326-C524-491F-83D8-BB2D105381F0}

[SecureDesktop]
ModuleType=1
CLSID={9B74623A-9D2E-11D3-A425-0060086848B4}
```

```
[Security]
ModuleType=1
CLSID={542C1680-FDE8-11d0-844C-00608CB8A7E9}

[SimOPC]
ModuleType=1
CLSID={585DED25-936B-11D2-A8DC-00A024C111A3}

[SNMPRuntime]
ModuleType=1
CLSID={5F4E4F3B-0BED-4b81-B95B-B5FF8244AF73}

[UTMRuntime]
ModuleType=1
CLSID={8DE5A1BC-BA79-4DF4-99A5-B4DC63414530}

[SNMPRutimeService]
ModuleType=1
CLSID={326631F7-7C40-4DBE-B690-50BF30D545C4}

[MMXCall_In]
ModuleType=1
CLSID={44378630-544D-11d4-B9D3-00600868297B}

[MMXFax]
ModuleType=1
CLSID={E333AEC8-8050-11D4-B9DE-00600868297B}

[MMXMail]
ModuleType=1
CLSID={293870E0-F5CE-11D3-B9B5-00600868297B}

[MMXMessenger]
ModuleType=1
CLSID={4B45F420-4867-11d4-844C-00104B6FD496}

[MMXPager]
ModuleType=1
CLSID={12E23258-AEA7-11D4-B9E9-00600868297B}

[MMXSnapshot]
ModuleType=1
CLSID={DFF2AD05-737A-11D4-8468-00104B6FD496}

[MMXSound]
ModuleType=1
CLSID={2CBEB6CE-FB4C-11D3-8430-00104B6FD496}

[MMXVideo]
ModuleType=1
CLSID={72008394-67C6-11D4-845D-00104B6FD496}

[OPCEnum]
ModuleType=1
CLSID={13486D51-4821-11D2-A494-3CB306C10000}
```