BEA Tuxedo®

Using BEA Tuxedo ATMI on Windows

Version 10.0
Document Released: September 28, 2007

Contents

3. Using the Visual C++.Net IDE To Develop BEA Tuxedo ATMI Applications

CHAPTER 1

Using BEA Tuxedo ATMI on Windows Server 2003

The following sections describe some basic differences *between* using BEA Tuxedo ATMI on a UNIX server system *and* using BEA Tuxedo ATMI on a Windows Server 2003 system:

Windows Considerations

Keep in mind the following general considerations when using BEA Tuxedo ATMI on a Windows system:

- When you specify pathnames, use a back slash (\), not a forward slash (/) to delimit file and directory names. Use a drive letter (such as `C:`) for all fully qualified paths.
- You do not need to specify the `.exe` suffix for executable files. The suffix is always implied for BEA Tuxedo ATMI for Windows.

- Filenames follow Windows naming conventions. For instance, names ending in `.dll` identify dynamically-linked libraries; names ending in `.lib` identify statically linked and imported libraries; names ending in `.cmd` identify command scripts.
- All BEA Tuxedo system executables, command scripts, and dynamically-linked libraries are located in `%TUXDIR%\bin`. Statically-linked libraries are located in `%TUXDIR%\lib`.

Configuration Issues

Keep in mind the following configuration issues when setting up the BEA Tuxedo ATMI software on a Windows Server 2003 system:

- Server names are case sensitive.
- The names of Windows Server 2003 machines specified in BEA Tuxedo ATMI application files should always be spelled in uppercase.
- The `OPENINFO` string in the BEA Tuxedo configuration file (`UBBCONFIG`) must be presented in the following format:

  `OPENINFO="`*`resource managers`*`:`*`resource`*`(`*`s`*`)"`

  For example:

  ```
  OPENINFO="TUXEDO\SQL:APPDIR1\bankdl3;bankdb;readwrite"
  ```

  Note: The first separator in the preceding string is a colon; subsequent separators are semicolons.

Specifying Machine Type and User ID Numbers

In the `MACHINES` section of the configuration file, include the following three entries:

- `TYPE="WinNT"`
- `UID=0`
- `GID=0`

Note: These entries require different settings on a UNIX system.

Whenever you create a configuration file for an environment with both UNIX and Windows systems, include these entries in the `MACHINES` section for every Windows Server 2003 node in your configuration.

Using Network Drives

For reliability purposes, BEA recommend that you do not use network drives. If, however, you try to start a BEA Tuxedo ATMI application on a Windows Server 2003 machine in which the `TUXCONFIG` file resides on a network drive, you must set the following permissions:

- The network drive must be connected as `Administrator`.
- The administrator must use the same password on the local and remote systems.
- In the `tuxipc` service startup options, the `ENTRY` option must have `Administrator` set for `Log On As This Account`. The password must be the same as the administrator's so that the `tuxipc` service has full administrator access rights.

Allocating and Releasing Memory Buffers

When allocating and releasing memory buffers on a Windows Server 2003 system, make sure each memory buffer is released from the same heap in which it was allocated. If it is not, a segmentation fault occurs.

For example, a memory buffer that is allocated using `Falloc()` must be released using `Ffree()`. If a memory buffer allocated with `malloc()` is freed using `Ffree()`, a segmentation fault occurs. The `free()` routine must be used, in the latter case, to free the memory buffer.

For more information about `Falloc, Falloc32(3fml)` and `Ffree, Ffree32(3fml)`, see *BEA Tuxedo ATMI FML Function Reference*. For more information about `malloc()` and `free()`, see the documentation delivered with your operating system.

Using the bankapp Driver

The `bankapp` program is a small example application bundled with BEA Tuxedo ATMI for Windows Server 2003. Besides demonstrating the operation of BEA Tuxedo ATMI and providing an example of BEA Tuxedo ATMI application code, the `%APPDIR%\UBB` file generated by the `bankapp` driver (`driver.exe` is located in `%TUXDIR%\APPS\bankapp\2003\driver`) can act as a template for configurations for any new applications.

Starting BEA Tuxedo ATMI Applications Automatically

When BEA Tuxedo ATMI is installed on a Windows Server 2003 server system, it may be useful to configure the machine to start a BEA Tuxedo ATMI application automatically when booting

up your system using the `srvany.exe` utility program provided in *Microsoft's Resource Kit for Windows 2003*. For configuration instructions, see `srvany.wri` and `rktools.hlp`.

To ensure proper operation of BEA Tuxedo ATMI programs that start automatically when booting up, you must set the BEA Tuxedo system environment variables `%TUXDIR%` and `%NLSPATH%`. Set these variables using the conventional Windows 2003 method or by using the BEA Administration program, as explained in "Setting and Modifying Environment Variables" on page 2-4.

See Also

For information about installing BEA Tuxedo ATMI on a Windows Server 2003 system, see *Installing the BEA Tuxedo System*. For more information about the BEA Tuxedo system, see the following documents:

- *Setting Up a BEA Tuxedo Application*
- *Administering a BEA Tuxedo Application at Run Time*
- *Using the BEA Tuxedo ATMI Workstation Component*
- *BEA Tuxedo Command Reference*
- *BEA Tuxedo ATMI C Function Reference*
- *BEA Tuxedo ATMI COBOL Function Reference*
- *BEA Tuxedo ATMI FML Function Reference*
- *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*

CHAPTER 2

Configuring BEA Tuxedo ATMI for Windows Server 2003

The following sections describe how to configure BEA Tuxedo ATMI on a Windows Server 2003 system:

- Introducing the BEA Administration Program
- Invoking the BEA Administration Program
- Accessing Other Machines on a Network
- Setting and Modifying Environment Variables
- Directing BEA Tuxedo Messages to the Windows Server 2003 Event Log
- Configuring tlisten Processes to Start Automatically
- Configuring IPC Resources to Maximize System Performance
- Reviewing the Windows Server 2003 Registry Content

Introducing the BEA Administration Program

In addition to the BEA Tuxedo Administration Console, the BEA Tuxedo 7.1 or later software for Windows provides a BEA Administration program and two Windows services (Tlisten and BEA procMGR) for configuring the BEA Tuxedo system on a Windows Server 2003 system. Only if the installation included BEA Tuxedo server components will these additional administration tools be installed on the Windows Server 2003 system.

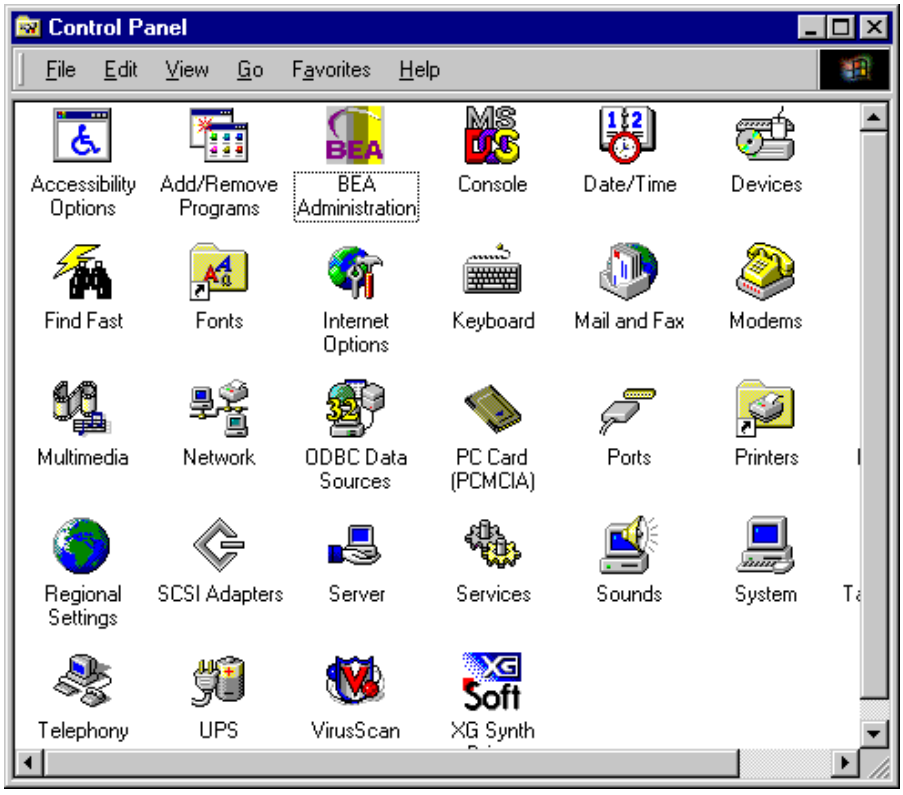
Note: The BEA Tuxedo Administration Console offers extensive online help: instructions for all the administrative tasks that can be performed through the console, plus reference information for all configuration tool folders. For information about how to use the BEA Tuxedo Administration Console, see *BEA Tuxedo Administration Console Online Help*.

Invoking the BEA Administration Program

With BEA Tuxedo release 7.1 or later software installed on your Windows Server 2003 system, perform the following steps to access the BEA Administration program:

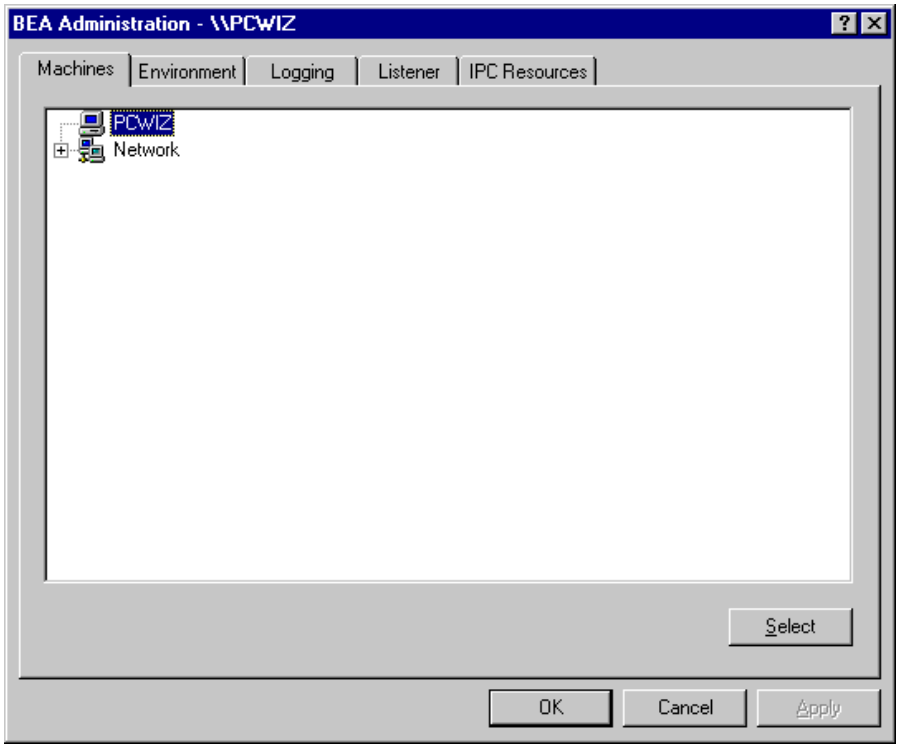
1. Choose Start → Settings → Control Panel to launch the Control Panel.

Figure 2-1 Microsoft Windows Control Panel



2. In the Control Panel, double-click the BEA Administration icon to launch the BEA Administration window.

Figure 2-2 BEA Administration Window with Machines Page Displayed



You can use the BEA Administration window to perform the following tasks:

- Access other machines on the network on which the BEA Tuxedo system is installed
- Set and modify environment variables for the BEA Tuxedo system
- Direct BEA Tuxedo system messages to the Windows Server 2003 Event Log
- Configure BEA Tuxedo tlisten(1) processes to start automatically
- Tune interprocess communication (IPC) resources to maximize BEA Tuxedo system performance

Accessing Other Machines on a Network

The Machines page of the BEA Administration window enables you, as the BEA Tuxedo system administrator, to access any machine (where you have login privileges) on the Microsoft Windows Network running Microsoft Windows Server 2003. You can then set environment variables remotely; determine the location of BEA Tuxedo event logging; add, remove, stop, or start `tlisten` services; and tune IPC resources.

To access a remote machine, locate and click the machine's icon on the network tree. If you know the name of a remote machine, but do not know its work group, perform the following steps to select it:

1. In the lower right-hand corner of the Machines page, click Select to display the Enter Machine Name dialog box.

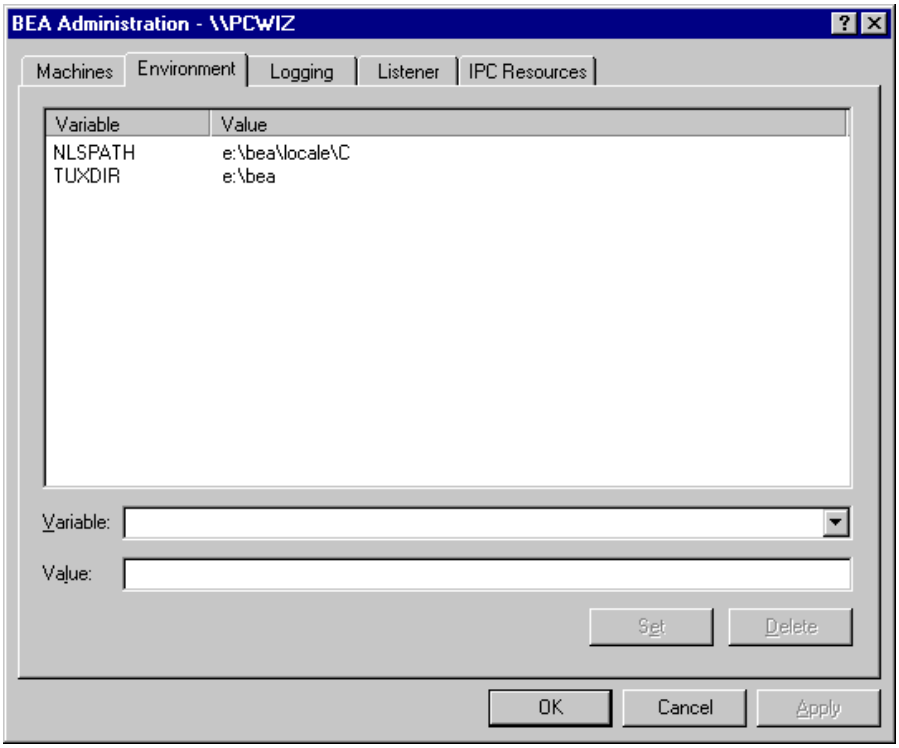
2. In the Enter Machine Name dialog box, enter the name of the remote machine (for example, `\\intrepid`) and click OK.

At this point, all subsequent actions performed on other pages (Environment, Logging, Listener, and IPC Resources) in the BEA Administration window will take place on the selected machine (`intrepid` in this example).

Setting and Modifying Environment Variables

The Environment page enables you to view, set, or modify BEA Tuxedo environment variables on your Windows Server 2003 system.

Figure 2-3 BEA Administration Window with Environment Page Displayed



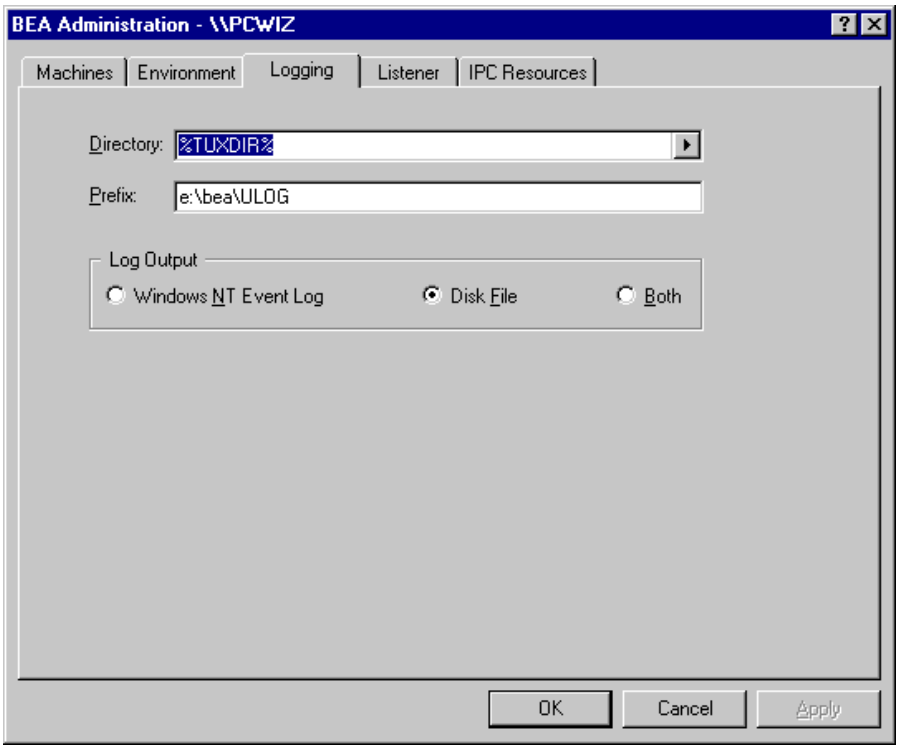
To add, modify, or delete environment variables using the Environment page, follow these steps.

1. To add a variable, enter its name in the Variable field and its value in the Value field, and then click Set.
2. To modify a variable, select the variable, enter its new value in the Value field, and then click Set.
3. To delete a variable, select the variable you want to delete and then click Delete.
4. Click OK or Apply to write your changes to the Windows Server 2003 Registry.

Directing BEA Tuxedo Messages to the Windows Server 2003 Event Log

The Logging page enables you to direct BEA Tuxedo system messages to the Event Log on your Windows Server 2003 system, to the traditional user log (ULOG) disk file, or both.

Figure 2-4 BEA Administration Window with Logging Page Displayed



You may select the Event Log option, the traditional user log (ULOG) Disk File option, or both. No setup is required for the Event Log.

Setting Up a ULOG

If you want to set up the traditional user log (ULOG) messages, perform the following steps to select a storage directory:

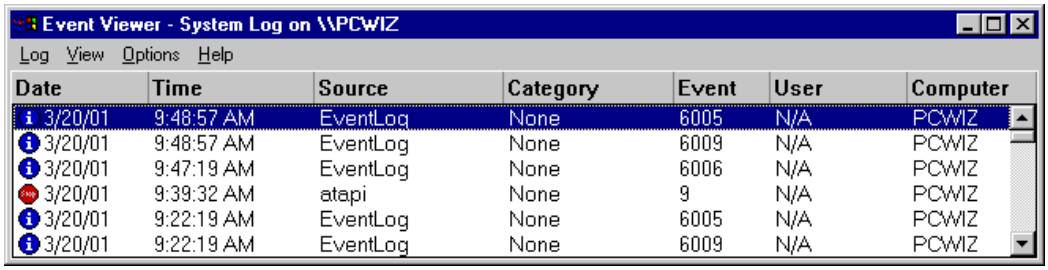
1. On the Logging page, click the arrow to the right of the Directory field to display a drop-down list of directories.
2. Select the name of the directory in which you want ULOG messages to be stored.
3. In the Prefix field, type a prefix for the name of the log file. The default prefix is ULOG, and the default file name is ULOG.*mmddyy*, where *mmddyy* is the month, day and year that the log file was created.
4. To save your selections to the Windows Server 2003 Registry, click OK or Apply.

Viewing Windows Server 2003 Event Log Entries

To view Windows Server 2003 Event Log entries, follow these steps:

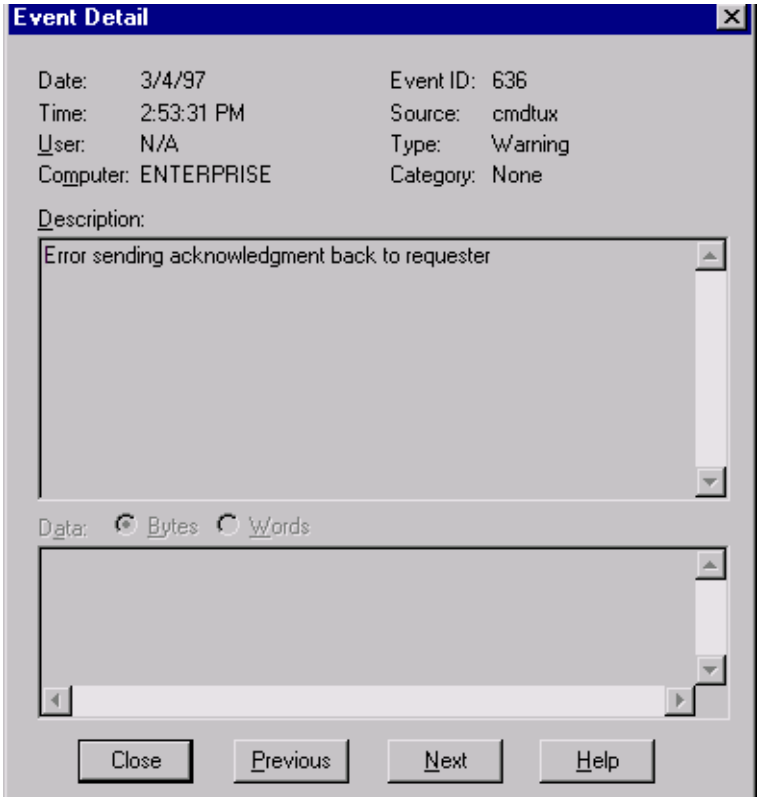
1. From the Windows Server 2003 desktop, select Start → Programs → Administrative Tools → Event Viewer to display a list of all the events that have occurred since the application booted.

Figure 2-5 Event Viewer Window



2. On the menu bar (across the top of the Event Viewer window), click Log to display a drop-down Log menu.
3. From the drop-down Log menu, choose Application. As a result of this selection, only application-specific events are listed in the Event Viewer window.
4. Double-click the entry for the event for which you want more information. The Windows Server 2003 Event Detail window appears, with information about the event you have specified.

Figure 2-6 Event Detail Window



This window provides the following information:

- Date—date on which the event occurred.
- Time—time at which the event occurred.
- User—person using the software at the time the event occurred.
- Computer—machine on which the event occurred.
- Event ID—number under which the message is listed in *System Messages*.
- Source—section of *System Messages* in which the event is described and an appropriate action is recommended.

- Type—purpose of the message; specifically, whether the message is intended to provide information, a warning, or notification of an error.
- Description—brief summary of the event.

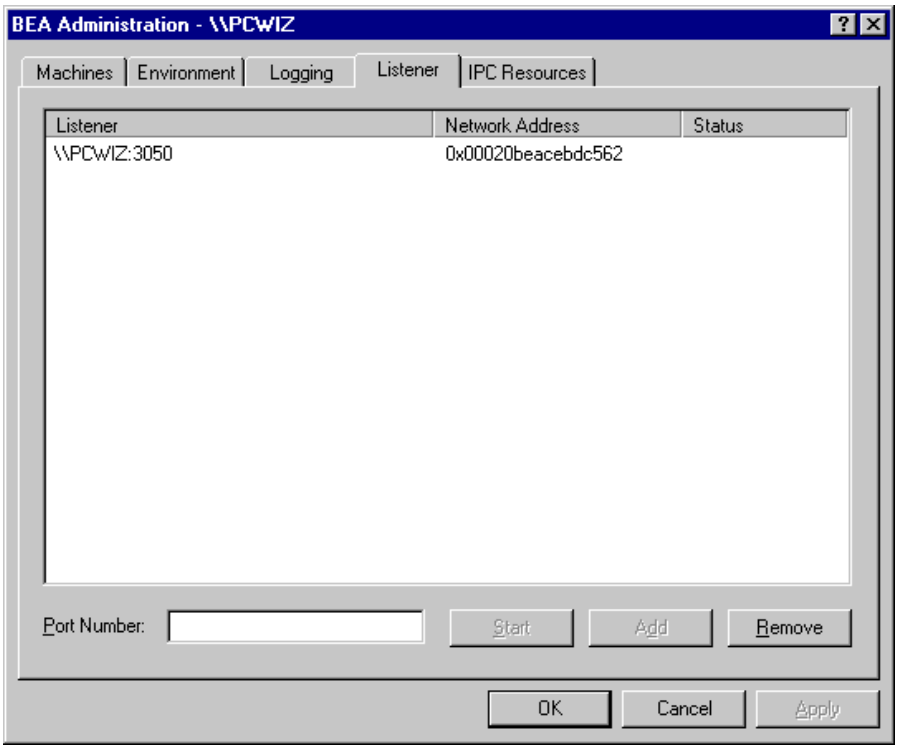
Viewing ULOG Entries

A ULOG is a text file. You can use any text editor to open a ULOG and view its contents.

Configuring tlisten Processes to Start Automatically

The Listener page enables you to view, create, or modify Tlisten services on your Windows Server 2003 system.

Figure 2-7 BEA Administration Window with Listener Page Displayed



During the BEA Tuxedo installation, the installer program installed a Tlisten service on your Windows Server 2003 system. Each time you boot your system, the Tlisten service starts a `tlisten` process on port 3050 of your machine. The password associated with the `tlisten` process is the one you entered during the installation.

A `tlisten` process must be started on each machine of a networked BEA Tuxedo application before the BEA Tuxedo system and application servers can boot. You use the `tlisten` process to perform administrative actions across multiple machines. To learn more about `tlisten` processes, see `tlisten(1)` in *BEA Tuxedo Command Reference*.

To add, remove, stop, or start Tlisten services using the Listener page, follow these steps.

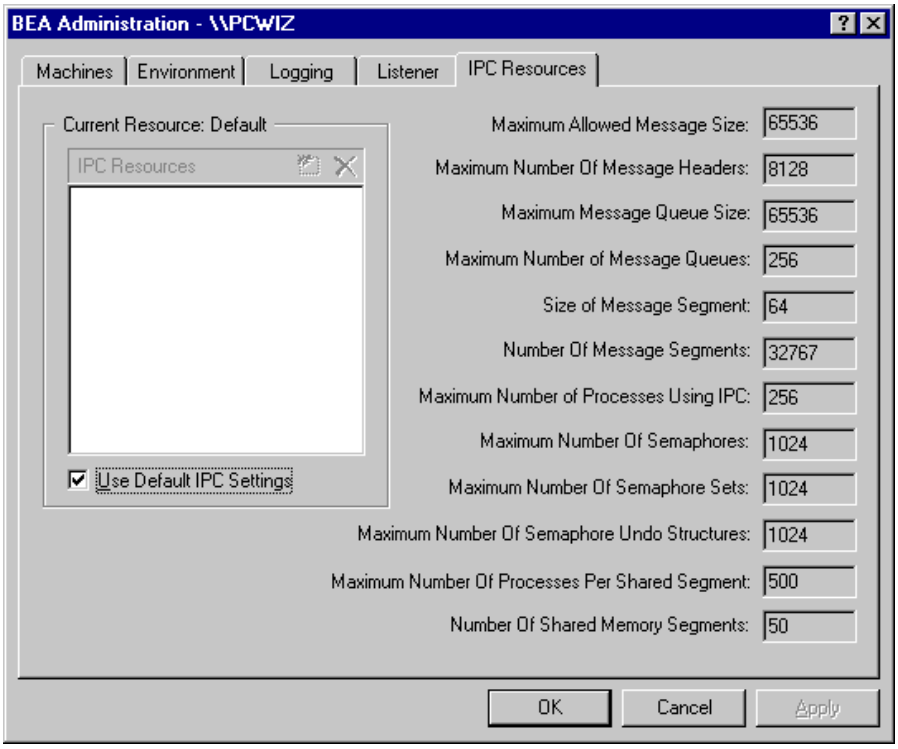
1. To add a Tlisten service, enter a port number in the Port Number field and then click Add. (Generally, you need one `tlisten` process for each BEA Tuxedo application running on your machine.)
2. To remove a Tlisten service, select the Tlisten service you want to delete and then click Remove.
3. To stop a Tlisten service that has been started, select the Tlisten service you want to stop and then click Stop.
4. To start a Tlisten service that has been stopped, select the Tlisten service you want to start and then click Start.
5. Click OK or Apply to write your changes to the Windows Server 2003 Registry.

Besides viewing and controlling Tlisten services using the Listener page, you can click Start → Programs → Administrative Tools → Services to launch the Services window and then view and control Tlisten services via the Services window.

Configuring IPC Resources to Maximize System Performance

The IPC Resources page enables you to configure the interprocess communication (IPC) resources on your Windows Server 2003 system to maximize BEA Tuxedo performance.

Figure 2-8 BEA Administration Window with IPC Resources Page Displayed



During the BEA Tuxedo installation, the installer program installed a BEA ProcMGR service on your Windows Server 2003 system. Each time you boot your system, the BEA ProcMGR service configures the IPC resources on your machine to whatever values you set on the IPC Resources page. The values shown in the preceding display are the default IPC values set by the installer program.

On most machines, BEA ProcMGR runs as installed; however, you can use the IPC Resources page to tune the IPC resources and maximize performance. To determine the minimum IPC Resource values required for a BEA Tuxedo application, see the following table and "Checking IPC Requirements" in *Installing the BEA Tuxedo System*. The following table maps the names of the IPC Resources on a Windows Server 2003 system to the traditional names on a UNIX system.

Table 2-1 IPC Resource Name Mappings Between Windows and UNIX Systems

Windows Server 2003 Name	Traditional UNIX Name
Maximum Allowed Message Size	MSGMAX
Maximum Number of Message Headers	No matching name
Maximum Message Queue Size	MSGMNB
Maximum Number of Message Queues	MSGMNI
Size of Message Segment	MSGSSZ
Number of Message Segments	MSGSEG
Maximum Number of Processes Using IPC	NPROC
Maximum Number of Semaphores	SEMMNS
Maximum Number of Semaphore Sets	SEMMNI
Maximum Number of Semaphore Undo Structures	SEMMNU
Maximum Number of Processes Per Shared Segment	No matching name
Number of Shared Memory Segments	SHMMNI

To modify IPC Resource values using the IPC Resources page, follow these steps:

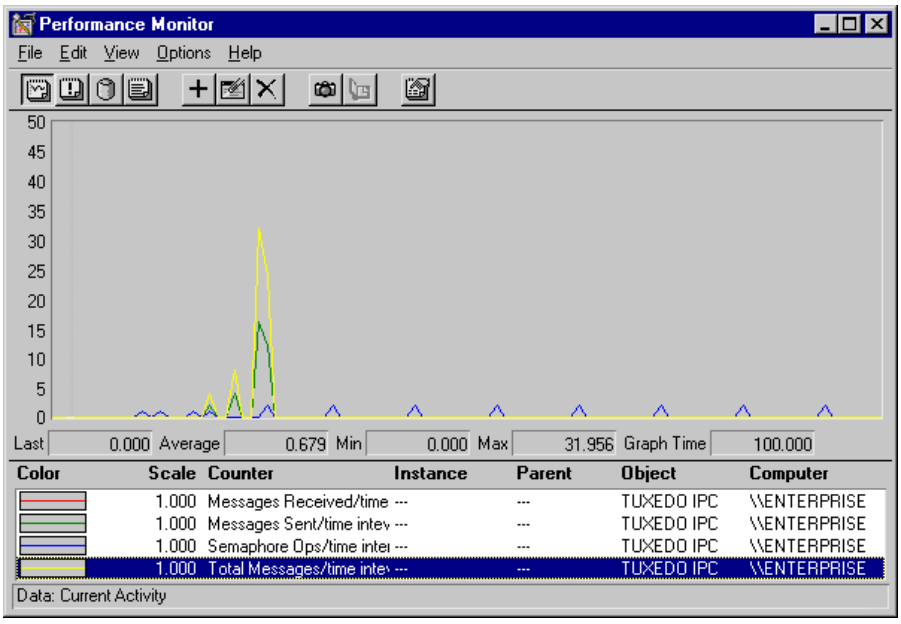
1. In the "Current Resource: Default box," click the Use Default IPC Settings check box to clear it. An insert box appears in the Current Resource: Default box.
2. Click the insert box, enter the name of your Windows Server 2003 machine, and press Enter.
3. Click the fields next to the IPC resources you want to change and enter the desired values.
4. Click OK or Apply to write your changes to the Windows Server 2003 Registry.
5. Stop and restart the BEA ProcMGR service to put your changes into effect: click Start → Programs → Administrative Tools → Services to launch the Services window and then stop and restart the BEA ProcMGR via the Services window.

When interpreting the `Maximum Number of Processes Using IPC` parameter on the IPC Resources page, keep the following information in mind:

- You must count any multicontexted BEA Tuxedo ATMI client multiple times. Your total should match the number of application associations (contexts) that can be outstanding concurrently.
- You must count any multicontexted BEA Tuxedo ATMI server multiple times. Your total should match the number of contexts calculated by adding 1 to the value of `MAXDISPATCHTHREADS` in the target application's configuration (`UBBCONFIG`) file, where 1 represents the main dispatcher thread.

You can view the performance of a running BEA Tuxedo application on the Windows Server 2003 Performance Monitor. Choose Start → Programs → Administration Tools → Performance Monitor to launch the Performance Monitor window.

Figure 2-9 Performance Monitor



Reviewing the Windows Server 2003 Registry Content

The Windows Server 2003 Registry is the repository for all hardware, software, and application configuration settings for the Windows Server 2003 system. During the BEA Tuxedo installation,

the installer program writes general installation information as well as IPC Resource values to the Registry. The structure of the Registry relevant to BEA Tuxedo is as follows.

```
HKEY_LOCAL_MACHINE\Software\BEA Systems\Tuxedo\9.1\...
```

- `Developer`
- `Environment`
- `Security`

To view this structure, choose Start → Run to launch the Run dialog box, enter `regEdt32`, and click OK to launch the Registry Editor window.

Developer Key

The Developer key stores product information, including the major and minor version numbers of the release, and user and company names.

Environment Key

The Environment key stores the locations referenced by the BEA Tuxedo environment variables set on your Windows Server 2003 system. It also stores other values such as IPC resource settings.

Security Key

The Security key holds the access permissions for BEA Tuxedo processes and services. The following permissions are mandatory:

- Any user who runs `tlisten(1)` must have read access permissions.
- The account under which the BEA ProcMGR service is running must have read access permissions.

BEA recommends that the `Administrator` have full control permissions.

CHAPTER 3

Using the Visual C++.Net IDE To Develop BEA Tuxedo ATMI Applications

The following sections describe how to develop ATMI application clients and/or servers in the Microsoft Visual C++ Integrated Development Environment (`msdev`) on a Windows XP, or Windows Server 2003 system:

Before You Start

Because BEA Tuxedo ATMI client but not server components can be installed on Windows XP, you can use the Microsoft Visual C++ IDE on these platforms to develop only ATMI application

clients. In contrast, because both BEA Tuxedo ATMI client and server components can be installed on Windows Server 2003, you can use the Microsoft Visual C++.Net IDE on the Windows Server 2003 platform to develop both ATMI application clients and servers.

Using Development Tools

BEA Tuxedo ATMI integrates into the Microsoft Visual C++.Net IDE (`msdev`) and emulates the functionality of `msdev` when integration is not possible. This integration makes it easier for you to develop BEA Tuxedo ATMI applications for 16-bit and 32-bit Windows operating systems.

The coding required to create BEA Tuxedo ATMI service requests, send requests, set up conversational connections, and get replies is fundamentally the same in both UNIX and Windows Server 2003 environments. `BuildTuxedo` and `TUXDEV` are tools that help you in your development environment.

- `BuildTuxedo` is a single tool, tightly integrated with the `msdev` build environment, that developers can use instead of the `buildserver`, `buildclient`, and `buildclt` commands. (Of course, these commands are still available for those who prefer to use them.) The `BuildTuxedo` system operates seamlessly on all currently supported Windows environments.
- `TUXDEV` allows you to create, edit, and compile multiple 16/32-bit FML tables and multiple 16/32-bit VIEW files. It also uses Multiple Document Interface (MDI) architecture so that you can use multiple views of these file types simultaneously.

Using the buildserver and buildclient Commands

Both the `buildclient()` and `buildserver()` commands are available to Windows Server 2003 but not to Windows XP. Only the `buildclient()` command is available to Windows XP, meaning that only Tuxedo application clients can be built on a Windows XP system. For more information about the build commands, see `buildclient(1)` and `buildserver(1)` in *BEA Tuxedo Command Reference*.

There are differences between how the options to the `buildclient()` and `buildserver()` commands work on non-integrated development environments and on integrated development environments, as indicated in the following table.

Table 3-1 Using the buildserver and buildclient Commands

To . . .	In a Non-Integrated Development Environment, Use This Option . . .	In an Integrated Development Environment . . .
Turn on verbose mode	`-v`	All options are displayed on tabs by default. (The `-v` option is unnecessary and unsupported.)
Specify an output file	`-o` (*`output_filename`*)	1. Select Settings from the `msdev` Build menu. 2. Select the Link tab in the Project Settings dialog box. 3. Specify the name of your output file by using the edit control.
Specify the first file to be linked	`-f`	1. Select Settings from the `msdev` Build menu. 2. Select the Link tab in the Project Settings dialog box. 3. Specify the first file to be linked by using the edit control.
Specify the last file to be linked	`-l`	1. Select Project from the `msdev` menu. 2. Select the Link tab in the Project Settings dialog box. 3. Specify the last file to be linked by using the edit control.
Specify a resource manager	`-r`	1. Access the `BuildTuxedo` GUI. 2. Access the `BuildTuxedo` Test window and select the Resources tab. 3. In the Tuxedo Resource Manager field, enter the name of the resource manager.
Specify services that will be available on a server	`-s`	1. Access the `BuildTuxedo` GUI. 2. Access the `BuildTuxedo` Test window and select the Services tab. 3. In the Service Names field, enter the name of each service on a separate line.
Use the COBOL compiler	`-c`	COBOL is unavailable.

To modify the build environment in an integrated development environment, follow these steps:

1. Choose Settings from the `msdev` Project menu.
2. Select the C/C++ or Link tab.

Note: CC and CFLAGS are no longer needed.

To specify the library and include paths in an integrated development environment, follow these steps:

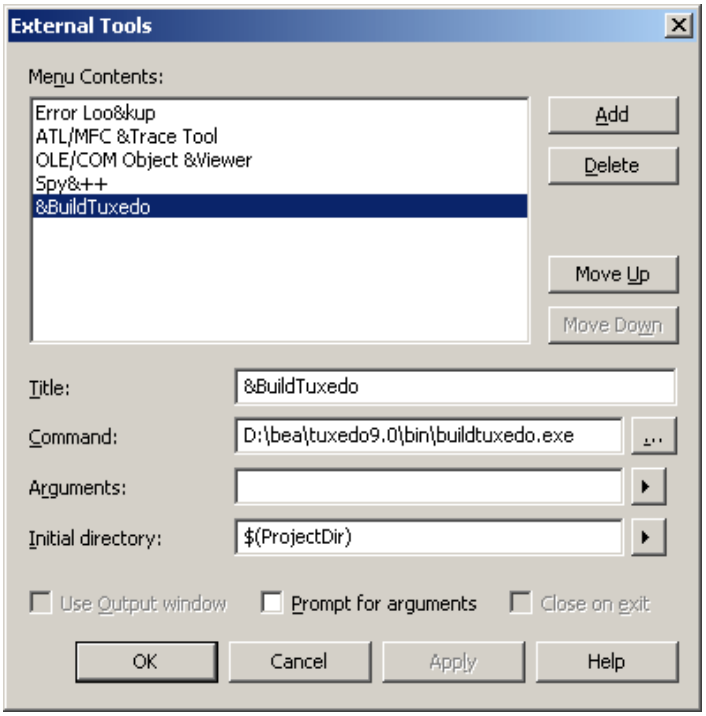
1. From the MSDEV Tools menu, choose Options to display the Options dialog box.
2. In the Options dialog box, select the Directories tab.

Adding BuildTuxedo to the MSDEV Tools Menu

To add `BuildTuxedo` to the MSDEV Tools menu, follow these steps:

1. In the Microsoft Visual C++.Net IDE (`msdev`) graphical user interface (GUI), choose Tools → External Tools. The External Tools window displays.
2. In the External Tools window, click Add to clear the Title field.

Figure 3-1 External Tools Window



3. In the Title field of the External Tools page, type `&BuildTuxedo` in the empty field.

   Note: To enable a letter key as a *hot key*, put an ampersand (&) before the appropriate letter.

4. In the Command field of the External Tools page, provide a pathname by doing one of the following:

   – Type the full pathname of `BuildTuxedo` (`%TUXDIR%\bin\BuildTuxedo`).

   – To the right of the Command field, click the ellipsis button  , browse to the desired directory, select `buildtuxedo.exe`, and click Open. The selected pathname appears in the External Tools window.

5. In the Initial directory field of the External Tools page, click the right arrow button to display a pop-up menu. In the pop-up menu, choose Project Directory (`$(ProjectDir)`).

   Note: To modify an item, highlight it and type over it. To move any item, highlight it and select the up or down arrow at the top of the Menu Contents box.

6. Click Apply to save the changes and click OK to close the External Tools window.

`BuildTuxedo` is now part of the MSDEV Tools menu.

Creating BEA Tuxedo ATMI Project Files

`BuildTuxedo` maintains a separate project file in the current directory for each BEA Tuxedo ATMI application using it. When `BuildTuxedo` begins, it searches for a valid project file in the current directory. If one is found, the various dialog controls are set to the values stored in the file and the dialog is displayed. The title bar displays the following information:

`BuildTuxedo` *`project_name`*

Because your `BuildTuxedo` project is closely associated with the `msdev` project in the current directory, `BuildTuxedo` also searches for two other files:

- A valid `msdev` makefile (*`filename`*`.mak`)
- An `msdev` project file (*`filename`*`.mdp`, *`filename`*`.dsw`, or *`filename`*`.dsp`)

If `BuildTuxedo` cannot find either of these files, it displays a warning and/or fails to activate. If the directory contains multiple `BuildTuxedo` project files, or multiple `msdev` project files or make files, menu items that contain appropriate target names are added to the System menu.

To save the current project file, select OK or Apply. To cancel any changes that you make to the project file or any file maintained by the BEA Tuxedo system, click Cancel or Esc.

Setting Up Your Environment

Before you can build your BEA Tuxedo ATMI application in an IDE, you must set the following fundamental parameters in your environment:

- Build type
- Header file
- Filename of the C or C++ file to be created and maintained by `BuildTuxedo`
- Function names
- Service names

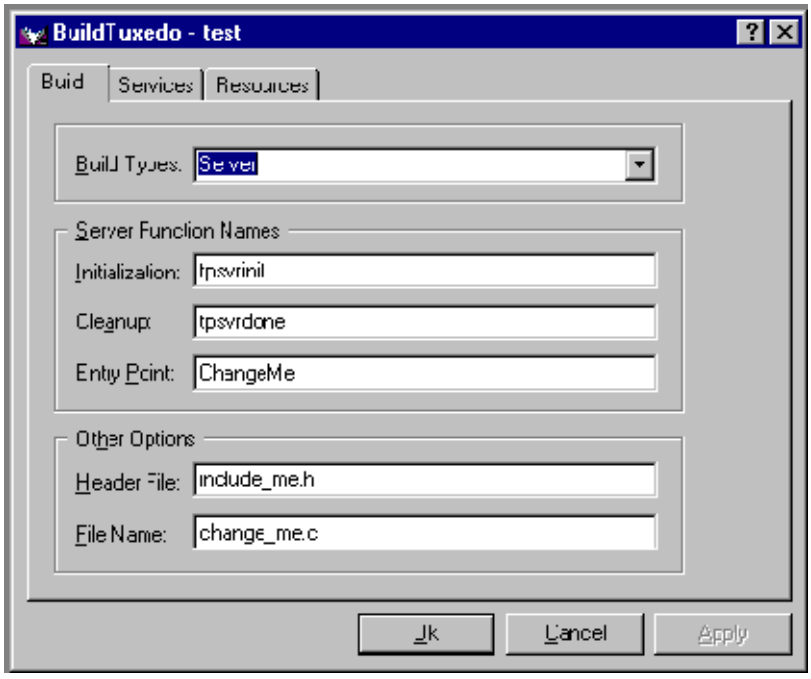
To provide this information, access the `BuildTuxedo` project_name dialog box in the `msdev` GUI.

Specifying the Build Type, Header File, and Filename

To specify the build type, header file, and filename, follow these steps:

1. In the project_name dialog box, click the Build tab to display the Build page.

Figure 3-2 Build Page



2. In the Build Types field of the Build page, click the down arrow and choose one of the following:
   - Server
   - Native client or Workstation client

Note: Windows XP users can select only Workstation Client type.

If You Select . . .	Then, After Making Your Selection . . .
Server	Enter information in the Initialization, Cleanup, and Entry Point fields. Proceed to step 3.
Native Client or Workstation Client	Proceed to step 3.

The Initialization and Cleanup options allow you to override the default init/exit functions by specifying valid function names. The Entry Point option allows you to specify the name of the function generated by `BuildTuxedo`. After specifying its name, you can call the function from anywhere in the application.

3. In the Header File field of the Build page, enter `stdafx.h`.
4. In the File Name field of the Build page, enter the name of the C or C++ file to be generated and maintained by `BuildTuxedo`.

How BuildTuxedo Uses the Header File

In the header file, `BuildTuxedo` adds the necessary pragma statements to build the current BEA Tuxedo ATMI project correctly. `BuildTuxedo` opens the file specified in the Header File field of the Build page, if the file is present; otherwise, it creates a new one. A section starting with:

```
//Begin Tuxedo Section*****DO NOT EDIT*****
```

and ending with:

```
//End Tuxedo Section
```

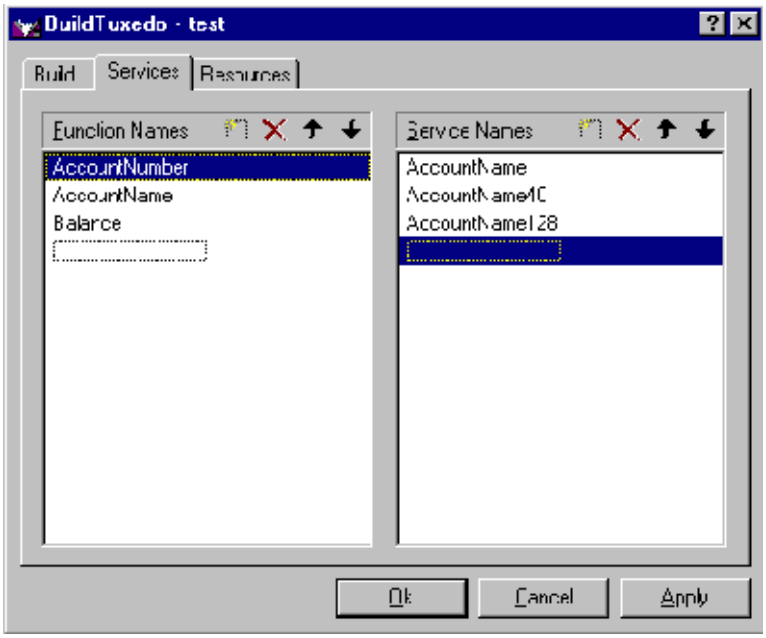
designates the area in the header file in which pragma statements are written. This area is maintained by `BuildTuxedo`. If `BuildTuxedo` does not locate this section in the file, it appends it to the end of the header file. Because all other text within the file remains unchanged, you can specify `stdafx.h` as the header file.

Note: If your project is new, and you select OK or Apply, you must select Files into Project from the MSDEV Insert menu. You then add the file generated by `BuildTuxedo` to the current project. You need to do this only for a new project or when you change the name of the C/C++ output file.

Specifying Function and Service Names

Select the Services tab to display the Services page, in which you specify function and service names.

Figure 3-3 Services Page



In the Services page, two lists are used to maintain the service dispatch table:

- Function Names is a user-maintained list of functions that you can associate with a service.
- Service Names is a user-maintained list of the associated services.

Note: To scroll up and down either list, use the arrow keys.

Generally, the service and the function that performs the service are represented by the same name. For example, function `x` performs service `x`. In some cases, the function may have a different name from the service it performs. For example, in one case, function `abc` performs services `x`, `y`, and `z`. In another case, the service name may not be known until run time.

You must specify any function associated with a service when you build the server. For any function associated with a service, you must specify the service, the appropriate prototype, a C

linkage, a void return, and a single `TPSVCINFO` pointer parameter. To specify a function to which a service name can be mapped, you must add the function to the Function Name list. This information is required for the service dispatch table.

Note: If you are using the `buildserver(1)` command, you can provide this information with the `-s` option. For more information about the `-s` option, see *Programming a BEA Tuxedo ATMI Application Using C* or *Programming a BEA Tuxedo ATMI Application Using COBOL*.

To add or edit names on the Function Names list, follow these steps:

1. In the Services folder, choose one of the actions described in the following table.

To	Action
Add an item	1. Click the New icon (broken box with twinkle in the upper left corner). 2. Click Insert, or select a blank area on the list (that is, a dotted lined box). 3. Type the name of the new function.
Modify an existing name	Highlight the name and type over it.
Delete a name	Highlight the name on the list and click the X icon or click Delete.

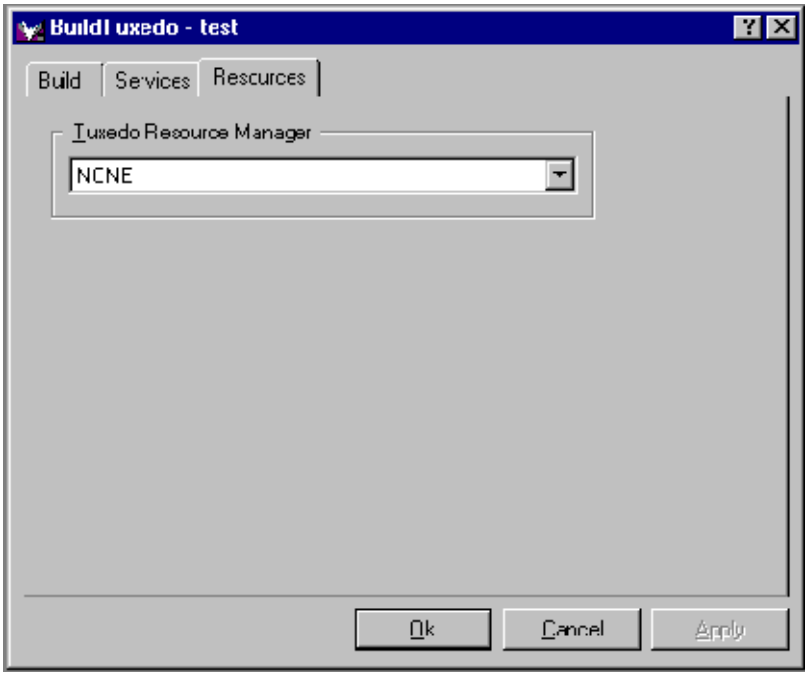
2. When complete, click OK or press Enter to save your changes and exit. (To exit without saving your changes, click Esc or Tab.)

Specifying a Resource Manager

To specify a resource manager, follow these steps:

1. Select the Resources tab in the `BuildTuxedo` window to display the Resources page.

Figure 3-4 Resources Page



2. In the Tuxedo Resource Manager field of the Resources page, type the full pathname of the resource manager for your application.

Note: The Tuxedo Resource Manager field contains a list of BEA Tuxedo resource managers available on the system, as defined in the `%TUXDIR%\udataobj\RM` file. If the file is not present, a default (NONE) is displayed.

Debugging a BEA Tuxedo ATMI Server Application

You should use the following procedure only if you have built the ATMI server application using the Debug configuration for your project.

To debug a server that has not been booted, follow these steps.

1. At any Windows command prompt, type `tmboot -n -d 1 -s` *`servername`* to display the command-line options used by `tmboot(1)` to start *`servername`*.

2. Execute the `tmboot -M` command to boot the BBL. (If necessary, boot additional application servers or machines.)
3. In `msdev`, select Project → Settings.
4. In the Program Arguments field of the Debug page, type the command-line options used in step 1.
5. Start the debugger and start debugging the server application.

   Note: Because BEA Tuxedo libraries are not built with debugging information and source code is not provided, you cannot access the BEA Tuxedo code directly.
6. To end the debugging session, type `tmshutdown` at any Windows command prompt.

   WARNING: Do not stop the server by selecting Debug → Stop; the BEA Tuxedo system may subsequently attempt to restart the server.

Note: To debug a server that is already running, type `msdev -p` *nnn* at any Windows command prompt, replacing *nnn* with the server's process ID (represented by a decimal number).

Developing an ATMI Application Using the Command Line Instead of the Visual C++.Net IDE GUI

If you must develop a BEA Tuxedo ATMI application using the command line instead of the Microsoft Visual C++ IDE GUI, use the `buildserver(1)` and `buildclient(1)` commands. To do this, specify the compiler and link options necessary to build a BEA Tuxedo ATMI application. For information about using these tools, see the following documents:

- *BEA Tuxedo Command Reference*
- *Tutorials for Developing BEA Tuxedo ATMI Applications*
- One of the following language-specific guides:
  - *Programming a BEA Tuxedo ATMI Application Using C*
  - *Programming a BEA Tuxedo ATMI Application Using COBOL*

To build a debug version of your ATMI application using `buildserver` or `buildclient`, you must compile all source files with the `/Zi` and `/Od` options. The `/Zi` option enables debugging; the `/Od` option disables optimization. In addition, you may need to define the `_DEBUG` preprocessor directive. To complete the process, indicate the link option as follows:

```
-l"/link/debug:full /debugtype:both"
```

Using the Tuxdev Application

To install the `Tuxdev` application, follow these steps:

1. In the Microsoft Visual C++.Net IDE (`msdev`) graphical user interface (GUI), choose Tools → External Tools. The External Tools window displays.
2. In the External Tools window, click Add to clear the Title field.
3. In the Title field of the External Tools page, type `&Tuxdev` in the empty field.

   Note: To enable a letter key as a *hot key*, put an ampersand (&) before the appropriate letter.
4. In the Command field of the External Tools page, provide a pathname by doing one of the following:
   - Type the full pathname of `BuildTuxedo` (`%TUXDIR%\bin\tuxdev.exe`).
   - To the right of the Command field, click the ellipsis button  , browse to the desired directory, select `tuxdev.exe`, and click Open. The selected pathname appears in the External Tools window.
5. To change the item displayed on the External Tools menu, change the entry in the Menu Contents field.

   Note: To create a hot key for a tool, put an ampersand (&) before any letter in the tool name. This enables you to invoke the tool at any time simply by typing that letter.
6. In the Initial directory field of the External Tools page, click the right arrow button to display a pop-up menu. In the pop-up menu, choose Project Directory (`$(ProjectDir)`).

   Note: To modify an item, highlight it and type over it. To move any item, highlight it and select the up or down arrow at the top of the Menu Contents box.
7. Click Apply to save the changes and click OK to close the External Tools window.

`Tuxdev` is now part of the MSDEV Tools menu.

Using the BEA Tuxedo ATMI Editors

Two editors are available in this environment: the FML Table Editor and the VIEW Table Editor. The user interface for both editors is similar to a workbook in which you can work on multiple views (that is, documents) simultaneously. You can also edit multiple files of various types at the same time. Both editors are similar to a Microsoft Excel spreadsheet:

- FML Table Editor enables you to create and/or edit an FML field table easily, and to generate user-selectable 16- or 32-bit field header files.
- VIEW Table Editor enables you to create and/or edit a VIEW Table file easily, and to generate user-selectable 16- or 32-bit VIEWS and header files.

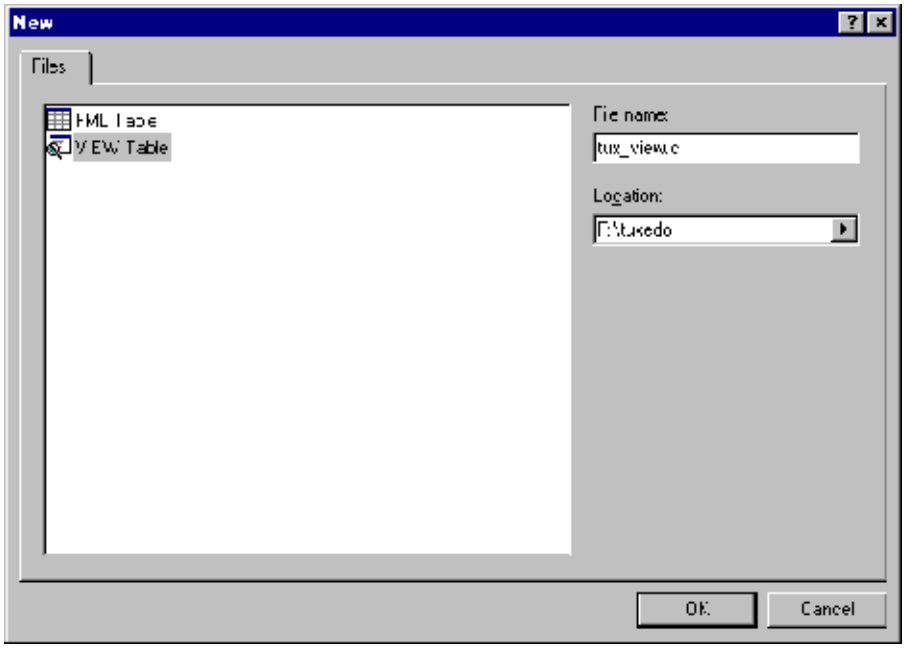
Using the FML Table Editor

To invoke the FML Table Editor, follow these steps:

1. On the Windows command line, type the full pathname of `tuxdev` (`%TUXDIR%\bin`) and press Enter.
2. Type `tuxdev` and press Enter to display the BEA Tuxedo Developer window.
3. Choose one of the actions described in the following table.

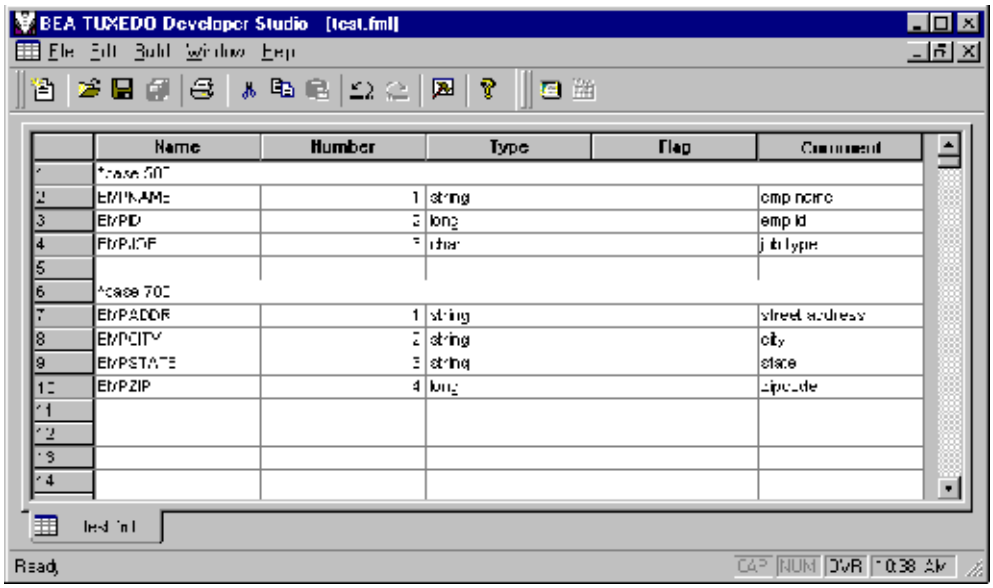
To . . .	Complete These Steps . . .
Create a file	1. In the BEA Tuxedo Developer window, choose File→New. The New window appears, as shown in the figure titled "New Window" on page 3-15. 2. Proceed to step 4.
Modify an existing file	1. In the BEA Tuxedo Developer window, choose File→Open. 2. Select the file (FML filenames include a `.fml` extension) to edit. 3. Skip the remaining steps.

Figure 3-5 New Window



4. In the New window, select FML Table and type the name of the new file in the File Name field.
5. Click OK to invoke the FML Table Editor.

Figure 3-6 FML Table Editor



As shown in the preceding figure, the FML Table Editor contains five columns—Name, Number, Type, Flag, and Comment—and an unlimited number of rows.

The following table describes the purpose of each column in the FML Table Editor.

Table 3-2 FML Table Editor Column Description

This Column . . .	Enables You to . . .
Name	Enter comment and/or base numbers. Each comment or number must be preceded by a pound sign (#) or an asterisk (*); otherwise, the line is assumed to be an active table entry. Data entered in an empty cell is assumed to be a new entry. (Blank lines are allowed.)
Number	Specify the relative number of the field.
Type	Select from a list of all valid values for this field.
Flag	Select flag settings (when the column is active).
Comment	Expand or clarify any entry designated in the Name column.

You can open new (unnamed) FML tables. A blank grid is created for the table named `FML Table`*x*, where *x* is a value tracked by the MDI and incremented by one each time a new table is created. You can specify a name if and when the table is saved. You can also open an existing text file for editing. Unless otherwise specified, files are saved in the directory in which the file was opened in tabbed delimited format, with `.fml` appended to the end of the filename when applicable. You can compile this file to produce either a 16-bit or 32-bit FML header file.

Using the VIEW Table Editor

To invoke the VIEW Table Editor, follow these steps:

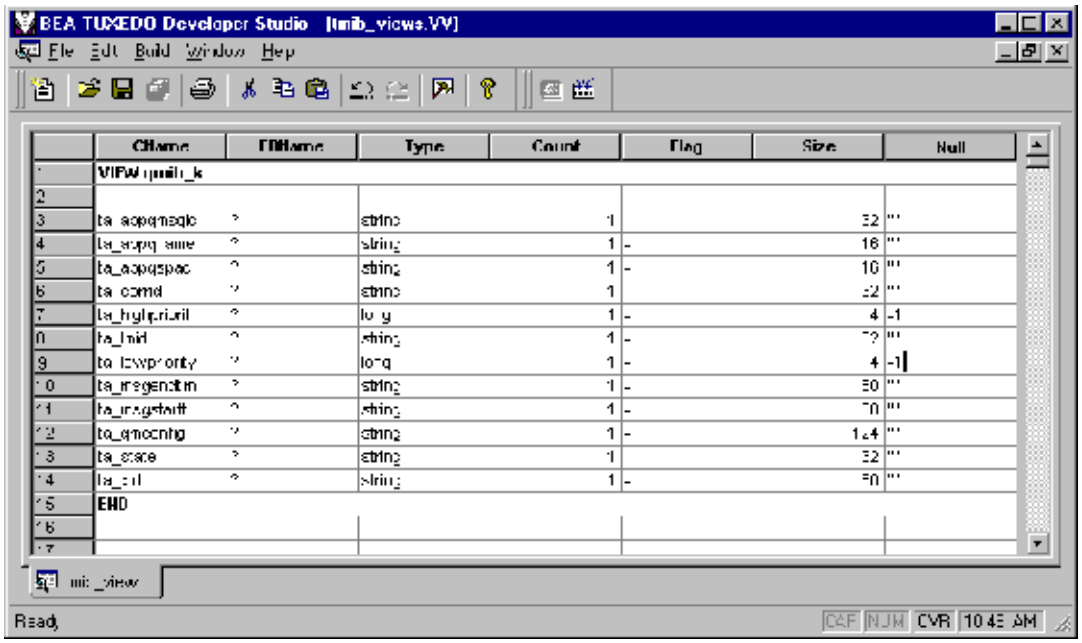
1. On the Windows command line, type the full pathname of `tuxdev` (`%TUXDIR%\bin`) and press Enter.
2. Type `tuxdev` and press Enter to display the BEA Tuxedo Developer window.
3. Choose one of the actions described in the following table.

Table 3-3 Using the View Table Editor Actions

To . . .	Complete These Steps . . .
Create a file	1. In the BEA Tuxedo Developer window select File→New to display the New window, as shown in the figure titled "New Window" on page 3-15. 2. Proceed to step 4.
Modify an existing file	1. In the BEA Tuxedo Developer window, choose File→Open. 2. Select the file (FML filenames include a `.fml` extension) to edit. 3. Skip the remaining steps.

4. In the New window, select VIEW Table and type the name of the new file in the File Name field.
5. Click OK to invoke the VIEW Table Editor.

Figure 3-7 VIEW Table Editor



As shown in the preceding figure, the VIEW Table Editor contains seven columns— CName, FBName, Type, Count, Flag, Size, and Null—and an unlimited number of rows.

You can enter comments in the CName column, as long as the required pound sign (#) is shown at the beginning of each comment. Blank lines are also allowed. When a CName entry is not preceded by a pound sign and is not NULL, the line is assumed to be an active table entry.

The following table describes the purpose of each column in the VIEW Table Editor.

Table 3-4 VIEW Table Editor Column Description

This Column . . .	Enables You to . . .
CName	Enter comment and/or base numbers. Each comment or number must be preceded by a pound sign (#) or an asterisk (*); otherwise, the line is assumed to be an active table entry. (Blank lines are allowed.)
FBName	Specify the **[field in the]** fielded buffer; this name is displayed in the field table file.
Type	Select from a list of all valid values for this field.
Count	Specify the number of elements to allocate (that is, the maximum number of occurrences to be stored for this member).
Flag	Select flag settings (when the column is active).
Size	Indicate the size of the member if the type is `string`, `carray`, or `dec_t`. Otherwise, you can specify '-', and the view compiler computes the size.
Null	Specify a null value or '-' (default null value) for that field.

Note: You must specify the following information to denote the start and end of the view information:

```
VIEW table_name
.
.
.
END
```

The information must appear in the CName column in a row by itself. You can enter multiple views within the same file, provided that each table entry is preceded by `VIEW` *`table_name`* and followed by `END`.

You can open new (unnamed) view files in which a blank grid is created for the view, `Viewx`, where *x* is a value tracked by the MDI and incremented by one each time you create a new view. You can specify a name if and when you save the view file. You can also edit an existing view file (either text or binary). Unless otherwise specified, files are saved in the directory in which the file was opened in tabbed delimited format, with `.v` appended to the filename when applicable. You can compile this file to produce either 16- or 32-bit binary `VIEW` and header files.

Working in Multiple Documents Simultaneously

The MDI, as part of the basic framework, provides a multiple-document architecture in which you can open documents and views, regardless of type, at the same time. Examples of this design are `msdev`, `Excel`, and `Word`. In BEA Tuxedo terms, you can open *x* number of FML tables and *y* number of VIEW Table files at any time, and then use any one of them. Each document looks and feels like a workbook that contains tabs for each open document.

How the Editors Validate Entries

The following table describes the information that is validated in each column of the FML Table Editor.

Table 3-5 Information Validated in the FML Table Editor

In This Column . . .	This Information Is Validated . . .
Name	Comments, base numbers, and valid text strings
Number	Numbers only. (The range is determined by a 16/32-bit user mode.)
Type	Valid FML types.
Flag	Valid FML flags. Extra checking is done for mutually exclusive flags.
Comment	Entries are not validated.

The following table describes the information that is validated in each column of the VIEW Table Editor.

Table 3-6 Information Validated in the VIEW Table Editor

In This Column . . .	This Information Is Validated . . .
CName	Entries are *not* validated.
FBName	Entries are *not* validated.
Type	Valid BEA Tuxedo types.

Table 3-6 Information Validated in the VIEW Table Editor

In This Column . . .	This Information Is Validated . . .
Count	Numbers only. (The range is determined by a 16/32-bit user mode.)
Flag	Valid FML flags. Extra checking is done for mutually exclusive flags.
Size	Numbers only. (The range is determined by a 16/32-bit user mode.)
Null	Entries are *not* validated.

See Also

For information about the BEA Tuxedo ATMI, see the following documents:

- *Programming a BEA Tuxedo ATMI Application Using C*
- *Programming a BEA Tuxedo ATMI Application Using COBOL*
- *Programming a BEA Tuxedo ATMI Application Using FML*
- *BEA Tuxedo Command Reference*
- *BEA Tuxedo ATMI C Function Reference*
- *BEA Tuxedo ATMI COBOL Function Reference*
- *BEA Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*