BEA Tuxedo

Using the BEA Tuxedo System on Windows NT

BEA Tuxedo Release 7.1
Document Edition 7.1
May 2000

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Using the BEA Tuxedo System on Windows NT

Document Edition	Date	Software Release
7.1	May 2000	BEA Tuxedo Release 7.1

Contents

CHAPTER

1 Differences Between Using the BEA Tuxedo System on UNIX and on Windows NT

- Windows NT Platform Considerations
- Configuration Issues
- Specifying Machine Type and User ID Numbers
- Using Network Drives
- Updating TxRPC
- Using the bankapp Driver
- Starting BEA Tuxedo System Applications Automatically

See Also

For information on installing the BEA Tuxedo system on a Windows NT system, refer to *Installing the BEA Tuxedo System*. For more information on the BEA Tuxedo system, refer to the following documents:

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

Windows NT Platform Considerations

The following list summarizes some considerations to keep in mind when using the BEA Tuxedo system on a Windows NT platform.

- When you specify path names, 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 the BEA Tuxedo system for Windows NT.

- Filenames follow Windows NT 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

The following list provides some configuration issues of which you should be aware when using the BEA Tuxedo system software on a Windows NT platform.

- Server names are case sensitive.
- Windows NT machine names specified in BEA Tuxedo application files should always be upper case.
- The `OPENINFO` string in the BEA Tuxedo configuration file must be 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

On the Machines page of the configuration file, include the following three entries.

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

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

Whenever you create a configuration file for an environment with both UNIX and NT machines, include these entries in the Machines page for every Windows NT node in your configuration.

Using Network Drives

For reliability purposes, it is recommended that you not use network drives. If, however, you attempt to start up the BEA Tuxedo system on a Windows NT machine that has the `TUXCONFIG` file 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 NT system,you must ensure that the memory buffer is released from the same heap in which it was allocated. Failure to do so results in a segmentation fault.

For example, a memory buffer that is allocated using `Falloc()` must be released using `Ffree()`. On the other hand, a memory buffer that is allocated using `malloc()` and freed using `Ffree()` will produce a segmentation fault. `free()` must be used, in this case, to free the memory buffer.

For more information on `Falloc, Falloc32(3fml)`and `Ffree, Ffree32(3fml)`, refer to the *BEA Tuxedo FML Function Reference*. For more information on `malloc()` and `free()`, refer to the documentation distributed with your operating system.

Updating TxRPC

When the BEA Tuxedo system interoperates with OSF/DCE, you must update the environment file to include the appropriate `PATH` variable for the application, allowing BEA Tuxedo system programs to find the OSF/DCE DLLs whenever needed. Review the `bldc_dce(1)` and `blds_dce(1)` reference pages in *BEA Tuxedo Command Reference* for information pertaining to using these commands on the Windows NT platform:

- The `bldc_dce` command builds a BEA Tuxedo system client that can be called via OSF/DCE RPC.
- The `blds_dce` command builds a BEA Tuxedo system server that calls OSF/DCE.

Using the bankapp Driver

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

Starting BEA Tuxedo System Applications Automatically

When the BEA Tuxedo system is installed on Windows NT as a server, it may be useful to configure the machine to start a BEA Tuxedo system application automatically when booting up your system using the srvany.exe utility program contained in *Microsoft's© Resource Kit for Windows NT*. Refer to `srvany.wri` and `rktools.hlp` for instructions on this procedure.

To ensure proper operation of BEA Tuxedo system 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 NT method or by using the BEA Tuxedo system BEA Tuxedo control panel. For more information, refer to "Modifying Environment Variables" on page 2-5.

CHAPTER

2 Configuring the BEA Tuxedo System for Windows NT

- Invoking the Control Panel to Configure the BEA Tuxedo System
- Accessing Machines on a Network
- Modifying Environment Variables
- Configuring tlisten Processes to Start Automatically
- Maximizing System Performance

Note: The BEA Administration Console offers extensive online help. Through the BEA Administration Console help, you can find instructions for all the administrative tasks that the BEA Administration Console helps you perform, plus reference information for all configuration tool folders.

Invoking the Control Panel to Configure the BEA Tuxedo System

In addition to the BEA Administration Console, the BEA Tuxedo system for Windows NT provides a control panel that you can use to configure the BEA Tuxedo system for Windows NT. You can use the BEA Tuxedo control panel to perform the following tasks:

- Access machines on a network by setting the Machines page
- Modify environment variables on the Environment page
- Direct BEA Tuxedo system messages to the Windows NT Event Log by setting the Logging page
- Configure one or more `tlisten` processes to start automatically by setting the Listener page
- Maximize system performance by tuning the IPC Resources page setting

To open the control panel:

1. Select Start–>Control Panel.

   The Microsoft Windows Control Panel opens, similar to that which is shown in the following figure.

Figure 2-1 Microsoft Windows Control Panel

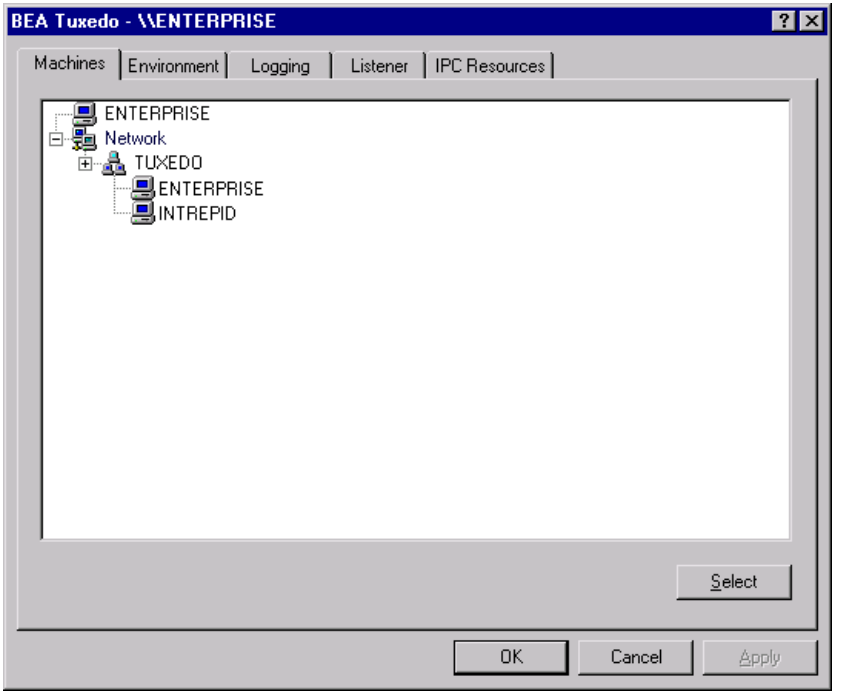
2. Double-click on the BEA Tuxedo icon to open the BEA Tuxedo control panel.

Accessing Machines on a Network

The Machines page of the BEA Tuxedo control panel enables a BEA Tuxedo administrator to access any machine on the Microsoft Windows Network running Windows NT, where the administrator has logon privileges. The administrator can then: set environment variables remotely; determine the location of BEA Tuxedo event logging; add, start, or remove `tlisten` services; and tune IPC resources.

The following figure shows the Machines page.

Figure 2-2 Machines Page



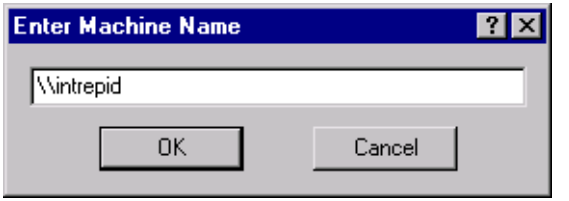
An administrator can access a remote machine by locating the machine in a network tree.

If you know the name of a machine, but not its work group:

1. Click Select.

   The Enter Machine Name dialog box is displayed, as shown in the following figure.

Figure 2-3 Enter Machine Name Dialog Box



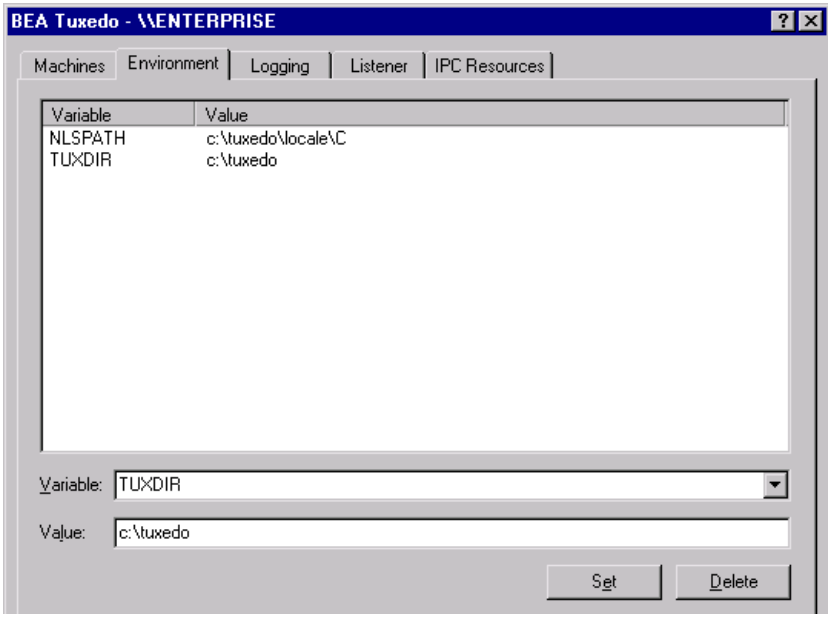
2. Enter the name of the remote machine in the field (for example, `\\intrepid` in the previous figure) and click OK.

All subsequent actions on other folders listed in the BEA Tuxedo control panel take place on the selected machine.

Modifying Environment Variables

The Environment page of the BEA Tuxedo control panel enables a BEA Tuxedo administrator to modify BEA Tuxedo environment variables in a way that is similar to the method used to modify Windows NT environment variables. The following figure shows the Environment page.

Figure 2-4 Environment Page



The Variable field contains a list of the most commonly used BEA Tuxedo environment variables.

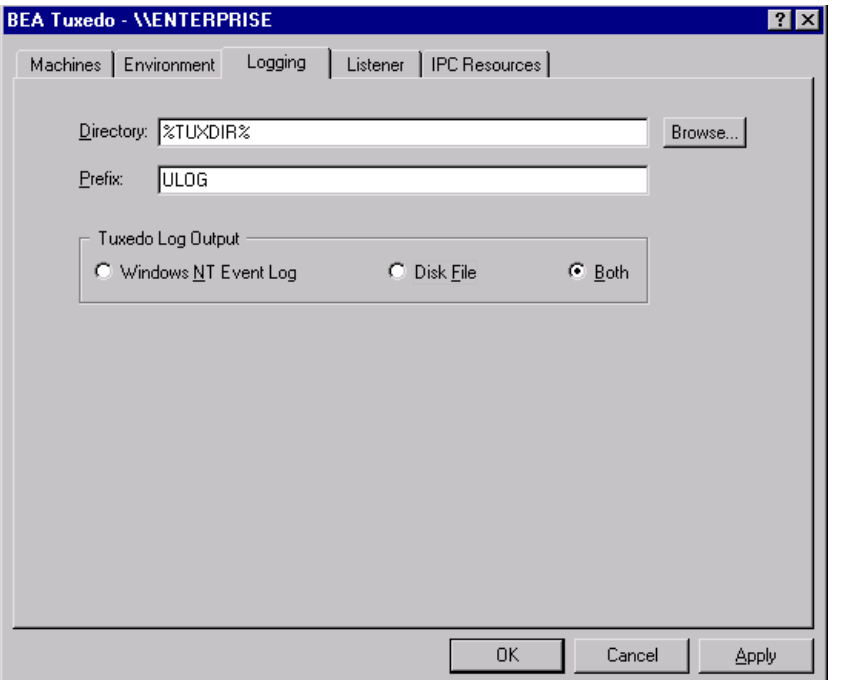
To add or edit a variable, select the variable, enter a value for it in the Value field, and click Set. To delete a variable, click the variable you want to delete and click Delete. Clicking Set saves any changes made to the BEA Tuxedo environment.

Directing BEA Tuxedo Messages to the Windows NT Event Log

The Logging page of the BEA Tuxedo control panel enables you to direct BEA Tuxedo system messages to the Windows NT Event Log.

The following figure shows the Logging page.

Figure 2-5 Logging Page

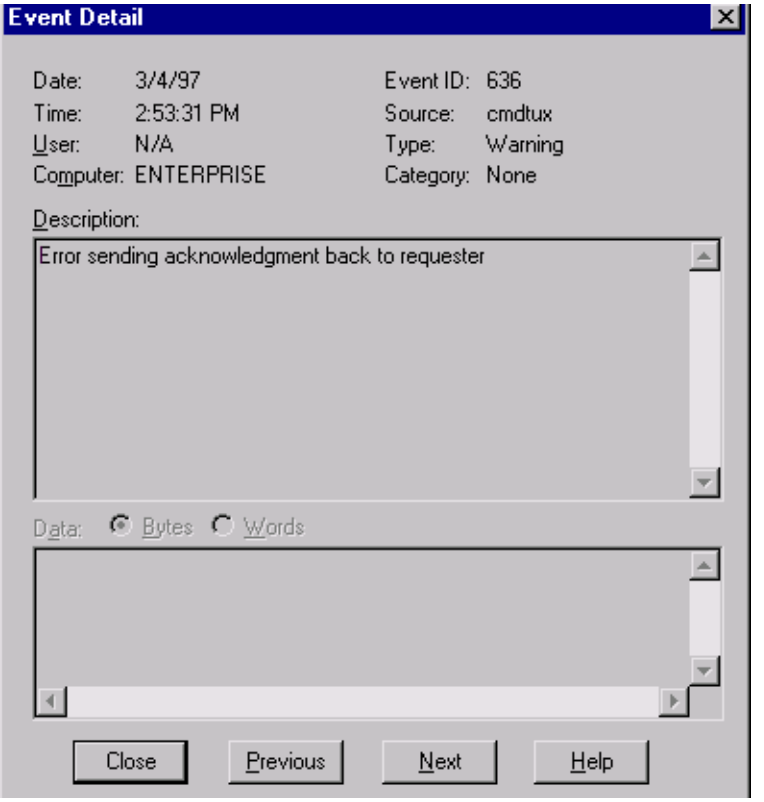


You can select the Logging option (Windows NT Event Log) or the traditional `ULOG` (Disk File), or both. If you want traditional `ULOG` messages, select the directory into which `ULOG` messages will be written, as well as a prefix for the name of the log file. The default prefix is `ULOG`, and the default filename is `ULOG.`*`mmddyy`*.

To view Event Log entries, from the Windows NT desktop select Start–>Programs–>Administrative Tools–>Event Viewer. In the Event Viewer Log window, verify that Application is checked.

The following figure shows the Windows NT Event Detail window with some sample entries.

Figure 2-6 Event Detail Window



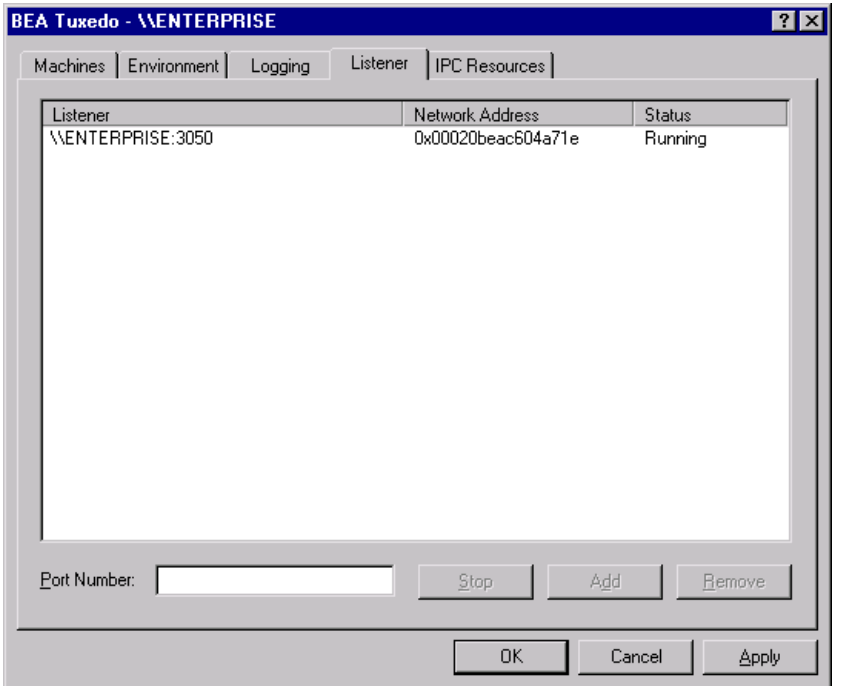
The entry for each event identifies the message number and catalog that you can use to look up the description and recommended action for that event in the *BEA Tuxedo System Messages*. Also provided are the type (that is, Information, Warning, or Error) and a brief description.

Configuring tlisten Processes to Start Automatically

The Listener page of the BEA Tuxedo control panel enables you to configure one or more `tlisten(1)` processes to start automatically when you boot up your system.

The following figure shows the Listener page.

Figure 2-7 Listener Page



To add a listener, enter a port number in the Port Number field and click Add. After you click OK or Apply, and reopen the BEA Tuxedo control panel, you can start or stop `tlisten(1)` services right on the Listener page. You can also use the Windows NT Services control panel to start or stop a `tlisten` service or to configure a `tlisten` service to start automatically.

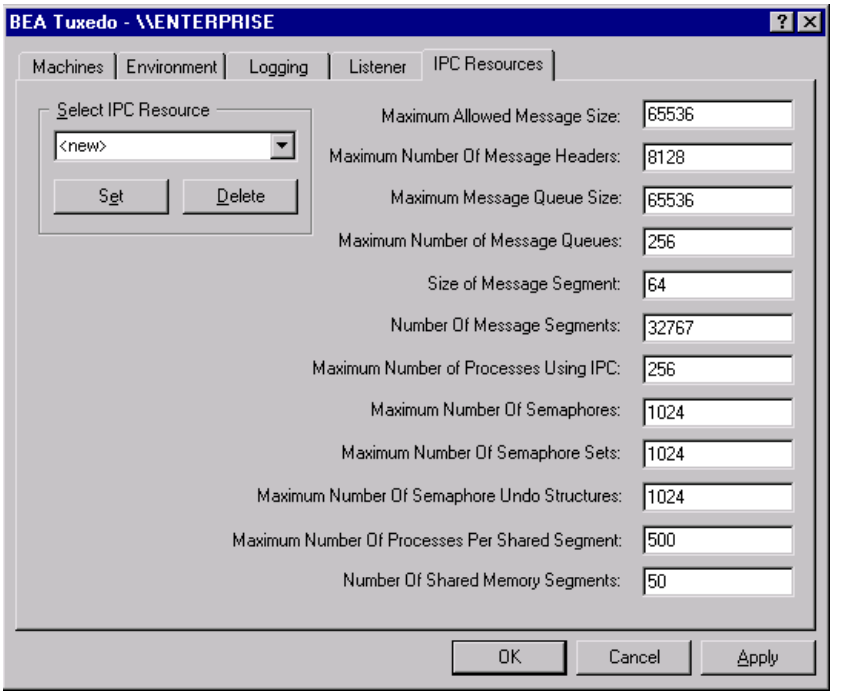
You can use the `tlisten(1)` program to perform administrative actions in an application distributed across multiple computers. You must start the `tlisten` program on each computer before running the application. Generally, you need one `tlisten` for each BEA Tuxedo application running on the computer.

Maximizing System Performance

The BEA Tuxedo system for Windows NT provides you with the BEA Tuxedo IPC Helper (`TUXIPC`), an interprocess communication subsystem that is installed with the product. On most systems IPC Helper runs as installed; however, you can use the IPC Resources page of the BEA Tuxedo control panel to tune the `TUXIPC` subsystem and maximize performance.

The following figure shows the IPC Resources page.

Figure 2-8 IPC Resources Page



With the IPC Resources page of the BEA Tuxedo control panel, you can set a variety of IPC resources to the right of each box. In the Select IPC Resources field, you can name a set of resources. Selecting the default Medium disables any changes to the parameters. If you backspace over Medium (a single backspace removes the whole string), the parameters are made available for changes.

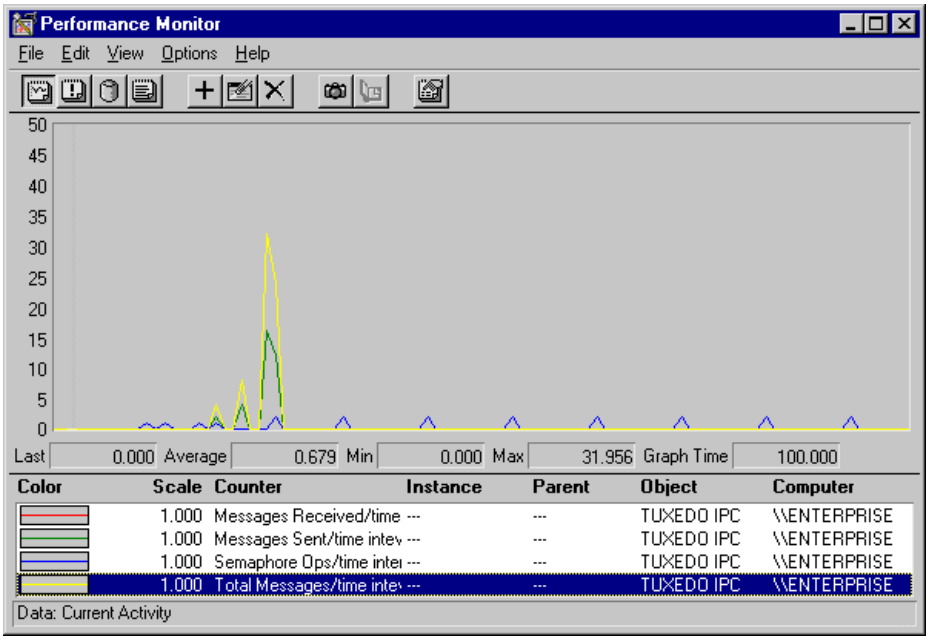
To add a new configuration, select <new>, edit the BEA Tuxedo IPC parameters as needed, and click OK to associate the name with the revised set of resources. Then click Apply to save the changes in the Registry table. You must stop and then restart the `tuxipc.exe` service to make the changes take effect.

When interpreting the `Maximum Number of Processes Using IPC` parameter on the `IPC Resources` page, please remember:

- You must count any multicontexted BEA Tuxedo 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 server multiple times. Your total should match the number of contexts calculated by adding 1 to the value of `MAXDISPATCHTHREADS`, where 1 represents the main dispatcher thread.

You can view the performance of a running BEA Tuxedo application from the Performance Monitor, as shown in the following figure.

Figure 2-9 Performance Monitor



CHAPTER

3 Developing a BEA Tuxedo Application on Windows NT

- Using Development Tools
- Using the buildserver and buildclient Commands
- Adding BuildTuxedo to the MSDEV Tools Menu
- Creating BEA Tuxedo Project Files
- Setting Up Your Environment
- Debugging a BEA Tuxedo Server Application
- Developing a BEA Tuxedo Application Without Visual C++
- Using the Tuxdev Application
- Using the BEA Tuxedo Editors

See Also

For information on the BEA Tuxedo ATMI, refer to the following documents:

- *Programming a BEA Tuxedo Application Using C*
- *Programming a BEA Tuxedo Application Using COBOL*
- *Programming a BEA Tuxedo Application Using FML*
- *Programming a BEA Tuxedo Application Using TxRPC*
- *BEA Tuxedo Command Reference*
- *BEA Tuxedo C Function Reference*
- *BEA Tuxedo COBOL Function Reference*
- *BEA Tuxedo File Formats and Data Descriptions Reference*

Using Development Tools

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

The coding required to create BEA Tuxedo service requests, send requests, set up conversational connections, and get replies is fundamentally the same in both UNIX and NT 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 Windows 95 and Windows 98, and all currently supported Windows NT environments (Intel and Alpha).

- 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

Although the buildserver(1) and buildclient(1) commands are available on the Windows NT platform, there are differences between how the options to these commands work on NT platforms and non-NT platforms. The following table lists these differences.

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.

To. . .	In a Non-Integrated Development Environment, Use This Option. . .	In an Integrated Development Environment. . .
Specify the last file to be linked	`-l`	1. Select Settings from the `msdev` Build 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 page. 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 page. 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: (a) select Settings from the `msdev` Build menu, and then (b) select either 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, select Options from the MSDEV Tools menu. From the Options dialog box, select the Directories tab.

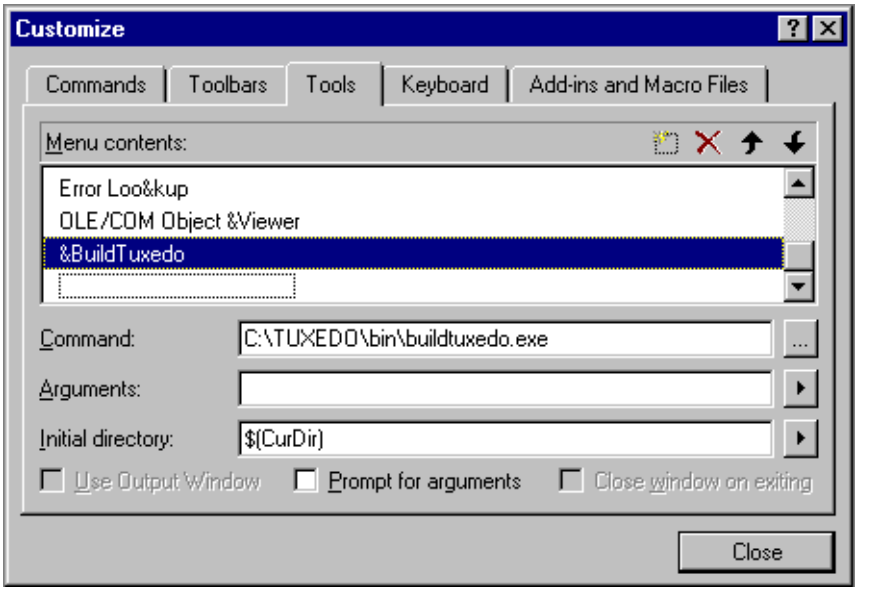
Adding BuildTuxedo to the MSDEV Tools Menu

To add `BuildTuxedo` to the MSDEV Tools menu:

1. From within the Microsoft Visual C++ (`msdev`) graphical user interface (GUI), select Tools–>Customize–>Tools.

   The Customize window is displayed, as shown in the following figure.

Figure 3-1 Customize Menu



2. Click New (broken box icon with twinkle) and type `&BuildTuxedo` in the blank box at the end of the Menu Contents box.

   Note: Placing an ampersand (&) before any letter enables that letter as a hot key.

3. In the Command field, type the entire path name of `BuildTuxedo` (`%TUXDIR%\bin\BuildTuxedo`), or select `...` to search for the name, and click Open to display the name in the Command field.

4. In the Initial Directory field, click the right arrow and select Current Directory.

   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.

5. Click Close.

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

Creating BEA Tuxedo Project Files

`BuildTuxedo` maintains a separate project file in the current directory for each BEA Tuxedo 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` make file (*`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, select Cancel or Esc.

Setting Up Your Environment

Before you can build your BEA Tuxedo 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

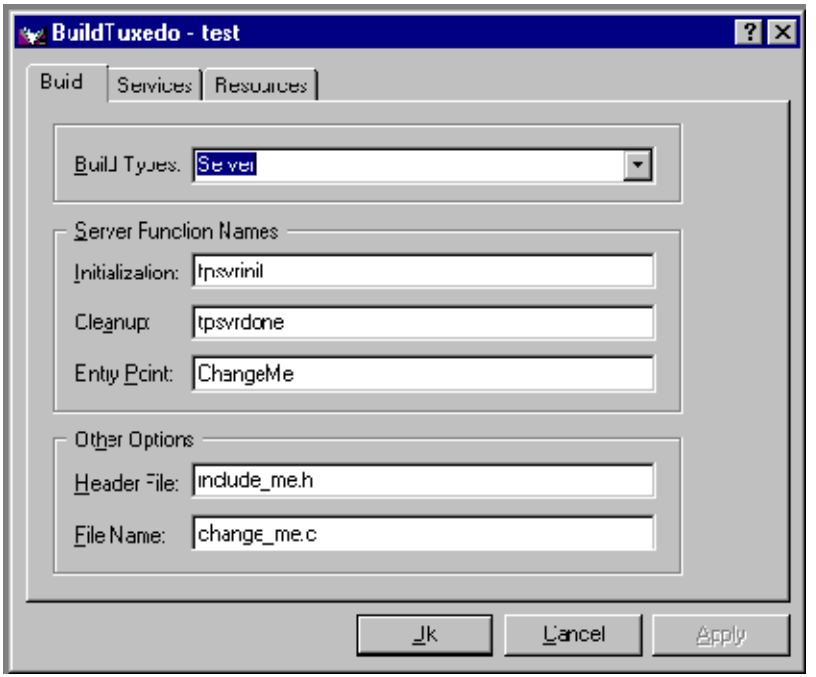
To provide this information, access the `BuildTuxedo` Project dialog box in the `msdev` GUI.

Specifying the Build Type, Header File, and Filename

To specify the build type, header file, and filename:

1. From the `BuildTuxedo` dialog box, select the Build tab.

Figure 3-2 Build Page



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

Note: Windows 95 and Windows 98 users can select Workstation Client type only.

If you Select. . .	Then, After 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 that `BuildTuxedo` generates. You can then call this function from anywhere in the application.

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

How BuildTuxedo Uses the Header File

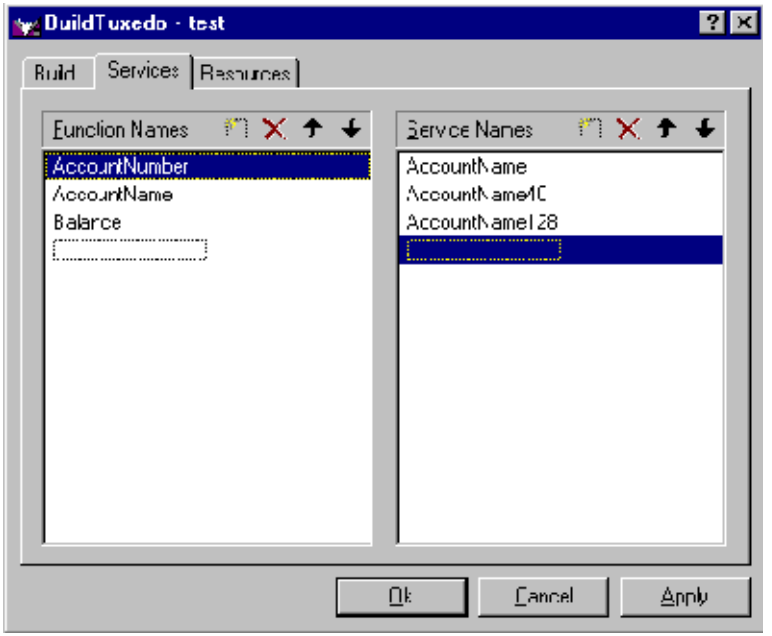
In the header file, `BuildTuxedo` adds the necessary pragma statements to build the current BEA Tuxedo project correctly. `BuildTuxedo` opens the file specified in the Header File field of the Build page, if the file is present; otherwise, it creates a new one. A section starting with `//Begin Tuxedo Section*****DO NOT EDIT*****` and ending with `//End Tuxedo Section` designates the area in the header file that `BuildTuxedo` maintains and into which pragma statements are written. 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 specify function and service names. The following figure illustrates the Services page.

Figure 3-3 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.

(If you are using the `buildserver(1)` command, you can provide this information with the `-s` option. Refer to *Programming a BEA Tuxedo Application Using C* or *Programming a BEA Tuxedo Application Using COBOL* for more information on the `-s` option.)

To add or edit names on the Function Names list:

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

To	Action
Add an item	1. Click New (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.

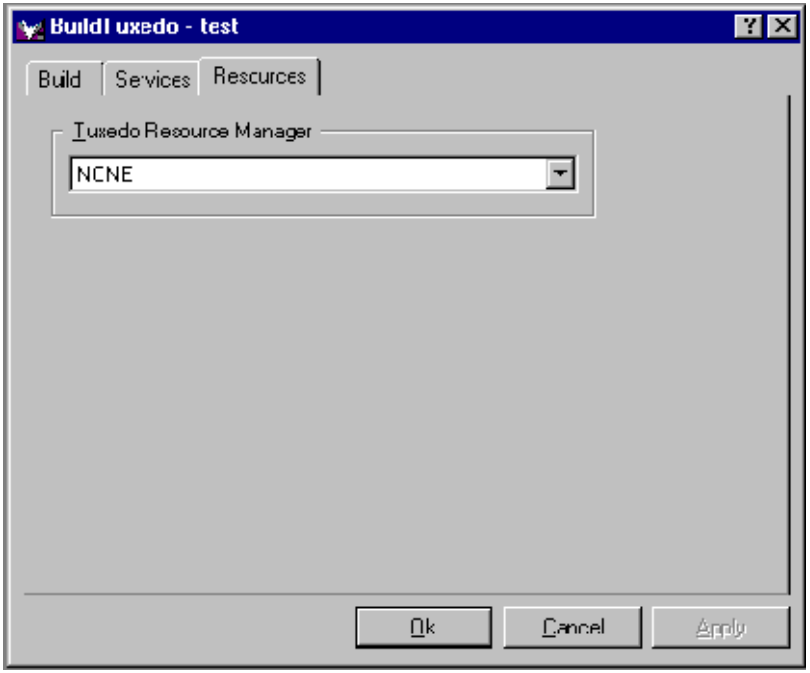
4. When complete, click Enter to save your changes and exit. (To exit without saving your changes, click Esc or Tab.)

Specifying a Resource Manager

To specify a resource manager:

1. Select the Resources tab from the `BuildTuxedo` menu to display the Resources page, as shown in the following figure.

Figure 3-4 Resources Page



2. In the Tuxedo Resource Manager field, type the entire path name 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 Server Application

To debug a server that has not been booted, complete the following procedure.

Note: You should use this procedure only if you have built the server application using the Debug configuration for your project.

1. At any Windows NT command prompt, type `tmboot -n -d 1 -s` *`servername`* to display the command-line options that `tmboot(1)` uses to start *`servername`*.
2. Execute the `tmboot -M` command to boot the BBL. (If necessary, boot additional application servers or machines.)
3. Select Project–>Settings in `msdev`.
4. In the Program Arguments field, select Debug and type the command-line options used in Step 1.
5. Start the debugger and proceed as needed to debug 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 NT prompt.

   Warning: Do not select Debug–>Stop to stop the server as the BEA Tuxedo system may subsequently attempt to restart it.

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

Developing a BEA Tuxedo Application Without Visual C++

If you need to develop a BEA Tuxedo application without Visual C++, use the `buildserver(1)` and `buildclient(1)` commands. To do this, specify the compiler and link options necessary to build a BEA Tuxedo application. Refer to *BEA Tuxedo File Formats and Data Descriptions Reference*, *Tutorials for Developing a BEA Tuxedo Application*, and *Programming a BEA Tuxedo Application Using C* or *Programming a BEA Tuxedo Application Using COBOL* for information on using these tools.

To build a debug version of your 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:

1. In the Microsoft Visual C++ (`msdev`) environment, select Tools–>Customize–>Tools–>Add.
2. In the Add Tool dialog box, type the entire path name in the Command field (`TuxDev.exe` is located in `%TUXDIR%\bin`), or select Browse to search for the name and click OK to display the path name in the Customize window.
3. To change the item displayed on the Tools menu, change the entry in the Menu Text field.

Note: To create a hot key for a tool, include an ampersand (&) before any letter in the tool name. This enables you to invoke the tool at any time simply by typing that letter.

4. In the Initial Directory field, click on the right arrow and select Current Directory to display its name (`$CurDir`).

Using the BEA Tuxedo 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 documents/views simultaneously. You can also edit multiple files of various types at the same time. Both editors closely resemble a Microsoft Excel spreadsheet with all of the associated functionality:

- The 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.
- The 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:

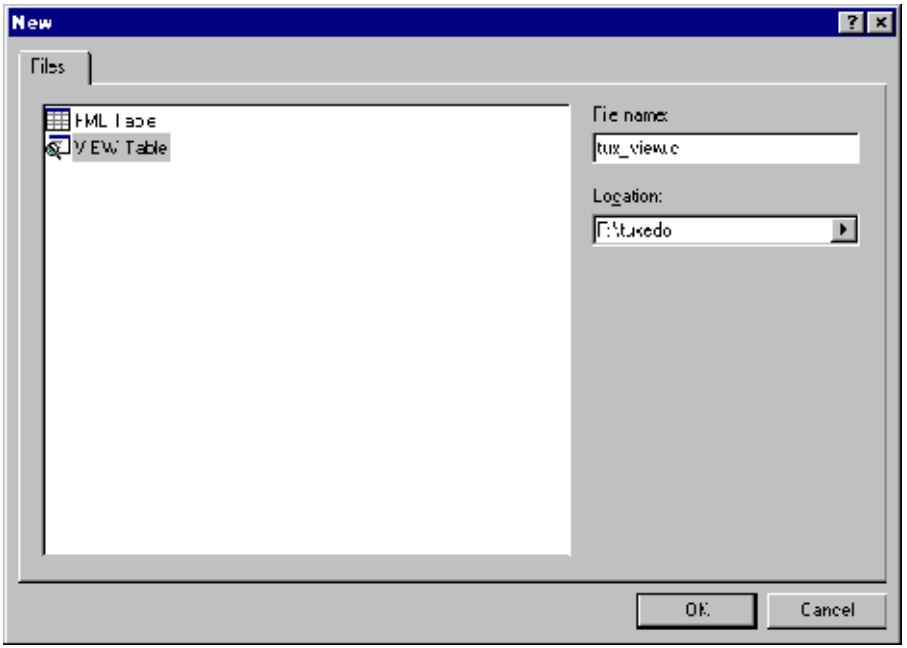
1. On the command line, enter the entire path name of `Tuxdev` (`%TUXDIR%\bin`) and select Enter.
2. Enter `tuxdev` and select Enter.

   The BEA Tuxedo Developer window is displayed.
3. Choose one of the following actions.

To. . .	Complete These Steps. . .
Create a file	In the BEA Tuxedo Developer window, select File–>New to display the New window shown in the following figure. Proceed to Step 4.
Modify an existing file	In the BEA Tuxedo Developer window, select File–>Open. Select the file (FML filenames have a `.fml` extension) to edit. Skip the remaining steps.

The following figure illustrates the New window.

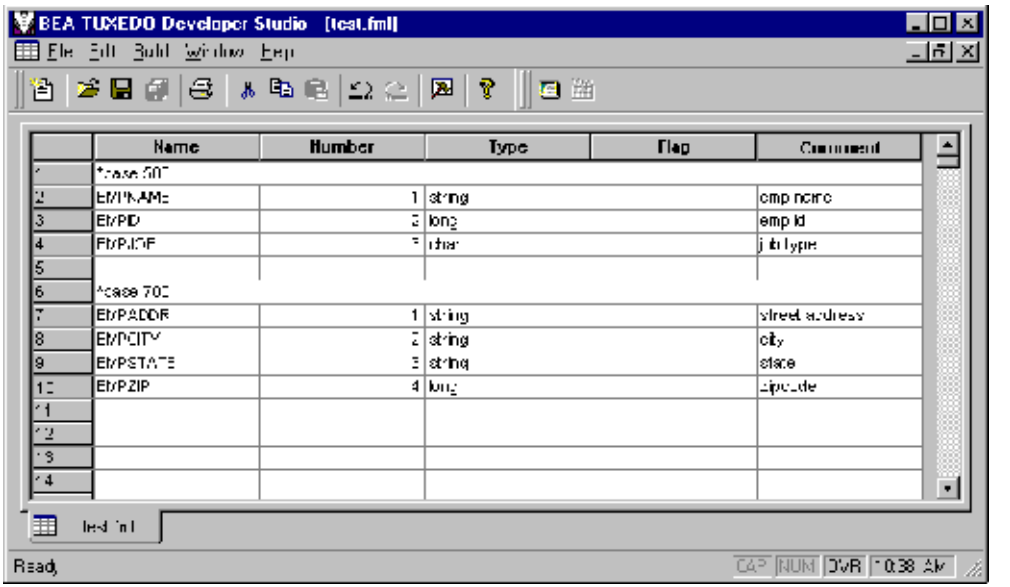
Figure 3-5 New Window



4. In the New window, highlight FML Table and type in the name of the new file in the File Name field.
5. Select OK.

   The FML Table Editor is invoked, as shown in the following figure.

Figure 3-6 FML Table Editor



As shown in the figure, the FML Table Editor contains five columns: Name, Number, Type, Flag, and Comment, with an unlimited number of rows. The following table lists the purpose of each column in the 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:

1. On the command line, enter the entire path name of `Tuxdev` (`%TUXDIR%\bin`) and select Enter.
2. Enter `tuxdev` and select Enter.

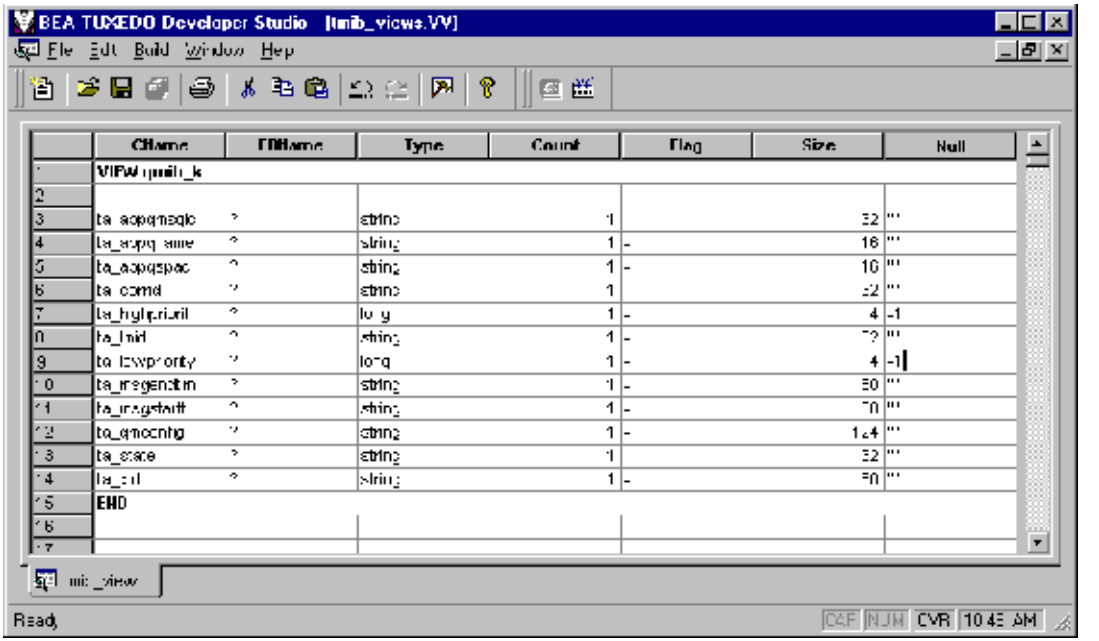
   The BEA Tuxedo Developer window is displayed.
3. Choose one of the following actions.

To. . .	Complete These Steps. . .
Create a file	1. In the BEA Tuxedo Developer window select File–>New to display the New window shown in the following figure. 2. Proceed to Step 4.
Modify an existing file	1. In the BEA Tuxedo Developer window, select File–>Open. 2. Select the file (FML filenames have a `.fml` extension) to edit. 3. Skip the remaining steps.

   The following figure illustrates the New window.
4. In the New window, highlight VIEW Table and type the name of the new file in the File Name field.
5. Select OK.

   The VIEW Table Editor is invoked, as shown in the following figure.

Figure 3-7 VIEW Table Editor



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

You can enter comments in the CName column, provided that each comment begins with the required pound sign (#). 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.

Table 3-3 VIEW Table Editor Column Description

This Column. . .	Enables you to . . .
CName	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.)

This Column. . .	Enables you to . . .
FBName	Name 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 where 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 3-4 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 3-5 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
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.