

ESKER *Tun[®] Plus*

Application Access – ActiveX[®]

Tun Plus 2014 - Version 15.0.0 Issued December 2013
Copyright © 1989-2014 Esker S.A. All rights reserved.

Copyright © 1998-2008 The OpenSSL Project. All rights reserved.
Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.
Copyright © 1995-2005 The Cryptix Foundation Limited. All rights reserved.
Copyright © 1995 Tatu Ylonen <ylo@cs.hut.fi>, Espoo, Finland. All rights reserved
Copyright © 1998 CORE SDI S.A., Buenos Aires, Argentina. All rights reserved
Copyright © 1995, 1996 by David Mazieres <dm@lcs.mit.edu>
Copyright © 1983, 1990, 1992, 1993, 1995 The Regents of the University of California. All rights reserved.
Copyright © 1998-2003 by Neil Hodgson neilh@scintilla.org. All Rights Reserved.
For additional information, conditions of use, and disclaimers, see copyright.pdf file.
Use and duplicate only in accordance with the Software License Agreement: Tun Products.

Esker, the Esker logo, Esker Pro, Extending the Reach of Information, Tun, and Tun Emul are trademarks, registered trademarks or service marks of Esker S.A. in the U.S., France and other countries. The following are trademarks of their respective owners in the United States and other countries: Microsoft, Windows, BackOffice, MS-DOS, XENIX are registered trademarks of Microsoft Corp. Netscape and Netscape Navigator are registered trademarks of Netscape Communications Corp. IBM, AS/400, and AIX are registered trademarks of IBM Corp. SCO is a registered trademark of Caldera International, Inc. NetWare is a registered trademark of Novell, Inc. Sun, Sun Microsystems and Java are trademarks of Sun Microsystems, Inc. Oracle is a registered trademark of Oracle Corp. Informix is a registered trademark of Informix Software Inc. Sybase is a registered trademark of Sybase, Inc. Progress is a registered trademark of Progress Software Corp. All other trademarks mentioned are the property of their respective owners.

Information in this document is subject to change without notice.



[See the list of Esker locations in the world.](#)

No part of this document may be reproduced or transmitted in any form or by any means without the prior written consent of Esker S.A.

Table of Contents

| | |
|--|-----------|
| Esker Viewer | 8 |
| Workspace..... | 8 |
| Wizard..... | 9 |
| Open Esker Viewer without a workspace..... | 9 |
| Start a workspace..... | 9 |
| Opening a session or workspace..... | 9 |
| Creating a new session..... | 9 |
| Configuring a session with the wizard..... | 10 |
| Opening an existing session or workspace..... | 10 |
| Saving a session or workspace..... | 10 |
| Modification of session connection parameters..... | 11 |
| Editing tools..... | 11 |
| Resources editor..... | 11 |
| Customizing the interface..... | 12 |
| Adding a command or menu to a toolbar or menu bar..... | 12 |
| Changing the appearance of an item..... | 13 |
| Adding a separator..... | 13 |
| Deleting an item..... | 13 |
| Retrieving the default toolbars and menu bars..... | 14 |
| Toolbar management..... | 14 |
| Content of Tools menu..... | 15 |
| Options..... | 16 |
| Configuring the firewall..... | 17 |
| Configuring SSL..... | 17 |
| Configuring SSH..... | 18 |
| Packager..... | 19 |
| | |
| Asynchronous Emulation | 21 |
| Creating an asynchronous emulation session..... | 21 |
| Dynamic Data Exchange..... | 26 |
| Use..... | 27 |
| Command syntax..... | 28 |
| Examples..... | 29 |
| Excel example..... | 30 |
| | |
| Synchronous Emulation | 33 |
| Synchronous emulation connection protocols..... | 33 |
| Multiple connection attempts..... | 35 |
| Opening an emulation session..... | 35 |
| Options..... | 35 |
| APL Mode(3270 emulation)..... | 37 |

| | |
|--|-----------|
| HLLAPI (3270 emulation only)..... | 38 |
| Using HLLAPI..... | 39 |
| Accessing data on an IBM MainFrame server from a Windows application (HLLAPI)..... | 39 |
| IBM Printers Emulation..... | 44 |
| Logical Unit (LU)..... | 44 |
| Using IBM printer emulation via Esker Viewer..... | 44 |
| Using IBM printer emulation via the print server..... | 45 |
| IBM Print Server..... | 45 |
| Print server administration..... | 45 |
| IBM printer emulation connection..... | 46 |
| Options..... | 46 |
| 3287 or 3812 Print Configuration..... | 48 |
| PC Print Configuration..... | 48 |
| 3287 or 3812 Emulation Status..... | 49 |
| Print Commands..... | 50 |
| Using Emulators..... | 51 |
| Configuring the screen..... | 51 |
| Asynchronous emulator..... | 51 |
| Synchronous emulator..... | 52 |
| Choosing the font..... | 53 |
| Customizing colors..... | 53 |
| Capturing attributes with the mouse..... | 53 |
| Asynchronous emulation..... | 54 |
| Synchronous emulation..... | 54 |
| Macros..... | 55 |
| Screen printing..... | 55 |
| Configuring a printer..... | 55 |
| Printing with template (3270/5250)..... | 56 |
| Printing with a template..... | 57 |
| Transparent printing (asynchronous emulation)..... | 57 |
| Cut & Paste..... | 57 |
| Copy options (asynchronous emulation)..... | 57 |
| File transfer..... | 58 |
| Asynchronous emulation..... | 58 |
| 3270 synchronous emulation..... | 59 |
| Multiple file transfer (synchronous emulation)..... | 60 |
| Working with a personal function-key panel..... | 61 |
| Asynchronous emulation..... | 61 |
| Synchronous emulation..... | 62 |
| Customizing the connection..... | 62 |
| Customizing the terminal..... | 62 |

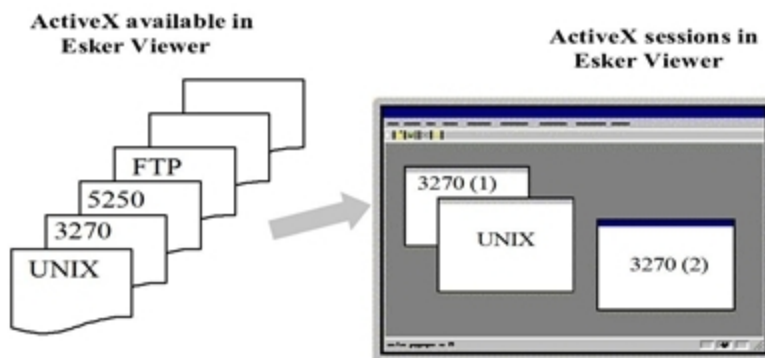
| | |
|---|-----------|
| Asynchronous emulation..... | 62 |
| Synchronous Emulation..... | 63 |
| Modifying the character table (IBM synchronous emulations)..... | 63 |
| Customizing the keyboard..... | 64 |
| Asynchronous emulation..... | 64 |
| Synchronous emulation..... | 70 |
| Choosing a national keyboard (UNIX asynchronous emulation)..... | 71 |
| Configuring the mouse (asynchronous emulation)..... | 71 |
| Script editor..... | 73 |
| Hotspots..... | 75 |
| Types of controls..... | 75 |
| List of controls per hotspot..... | 76 |
| Creating a new hotspot..... | 76 |
| Creating a new control for a hotspot..... | 79 |
| Creating a control from a zone selected on the screen..... | 79 |
| Modifying a hotspot or control..... | 79 |
| Deleting a hotspot or control..... | 79 |
| Control actions..... | 79 |
| Macros..... | 82 |
| Start the macro administrator..... | 82 |
| Creating and debugging a macro using the macro editor..... | 82 |
| Creating a macro using the macro recorder..... | 83 |
| Modifying a macro or a macro library using the macro editor..... | 83 |
| Creating a macro library..... | 84 |
| Deleting a macro library..... | 84 |
| Executing a macro..... | 84 |
| Encrypting macros..... | 84 |
| Encrypting a character string..... | 84 |
| Saving macros..... | 85 |
| Example macro: Recording a connection macro with password encryption..... | 85 |
| UNIX connection macro in VBScript..... | 85 |
| IBM connection macro in JavaScript..... | 85 |
| Function-Key Panel Editor..... | 87 |
| Starting the function-key panel editor..... | 87 |
| Creating a panel..... | 87 |
| Creating a button..... | 87 |
| Linking properties to a button..... | 88 |
| Button parameters dialog..... | 88 |
| Using Lock buttons..... | 89 |
| Define default button settings..... | 89 |

| | |
|---|------------|
| Panel Settings and Positioning..... | 90 |
| Set Tab Order..... | 90 |
| Opening an existing function-key panel..... | 91 |
| Saving a function-key panel..... | 91 |
| Testing a key panel..... | 91 |
| Selecting buttons..... | 91 |
| Moving and resizing buttons..... | 91 |
| Sizing multiple buttons identically..... | 91 |
| Duplicating a button..... | 92 |
| Deleting a button..... | 92 |
| Aligning the buttons..... | 92 |
| Aligning the buttons on a grid..... | 92 |
| Toolbar..... | 93 |
| Advanced Use of Asynchronous Emulator..... | 95 |
| Escape sequences..... | 96 |
| Content of an escape sequences file..... | 97 |
| Terminal initialization..... | 97 |
| Sequence headers..... | 98 |
| Defining escape sequences..... | 98 |
| Example: setting the mouse for ansi emulation..... | 102 |
| Example: reassignment of a keyboard key for ansi emulation..... | 102 |
| Function keys..... | 102 |
| Content of a function keys file..... | 103 |
| Integration of function keys in the emulator..... | 103 |
| Terminal configuration..... | 104 |
| Content of a terminal configuration file..... | 104 |
| Details..... | 105 |
| National keyboards..... | 105 |
| Reading a .nat file..... | 106 |
| Control codes..... | 106 |
| Code conversion..... | 107 |
| Character tables..... | 108 |
| Internal character table management..... | 109 |
| Alternate character font..... | 110 |
| Eastern European character sets..... | 111 |
| Samples of Esker emulator actions..... | 112 |
| Quitting Esker Viewer upon server request..... | 112 |
| File transfer requested by server..... | 112 |
| Windows to UNIX copy..... | 113 |
| UNIX to Windows copy..... | 113 |
| PC programs started by server..... | 113 |

| | |
|--|------------|
| Macro execution requested by server..... | 114 |
| Transparent printing..... | 114 |
| Actions..... | 114 |
| Example 1: transparent printing on the PC's default printer via the Windows print manager..... | 115 |
| Example 2: Direct transparent printing on a printer port..... | 115 |
| Dynamically changing terminal type..... | 116 |
| Changing sessions automatically..... | 116 |
| Mouse support in UNIX applications..... | 116 |
| Provided actions..... | 117 |
| Implementation..... | 120 |
| Miscellaneous solutions..... | 121 |
| Color attributes in emulation..... | 121 |
| 132-column emulation..... | 122 |
| Emulation with 25 lines..... | 122 |
| Scancode emulation..... | 123 |
| Using scancode mode..... | 123 |
| Using COM3 and COM4..... | 124 |

Esker Viewer

Esker Viewer is a Windows application that hosts ActiveX components. Esker Viewer uses a MDI (Multiple Document Interface), allowing simultaneous opening of several windows. Each ActiveX component can have its own toolbars and menu bars, and each of its sessions appears in a different window within the same application assembly.



Esker Viewer can contain:

- Asynchronous terminal emulations (UNIX, DEC, HP...)
- 3270 and 5250 terminal emulations for access to IBM MainFrame and AS/400 servers
- 3287 and 3812 printer emulations
- FTP file transfer sessions

The ActiveX component and Esker Viewer communicate via script; Esker Viewer implements the ActiveX standard Scripting. This standard enables it to understand the Microsoft VBScript and JavaScript languages among others.

Workspace

A workspace corresponds to all the files and parameters required to set up one or more sessions: number and type of sessions to be opened, session opening and closing parameters, display of windows, menus, toolbars, etc. A workspace is saved into a .CWZ file.

Note:

A workspace records the parameters of one or more ActiveX sessions hosted by Esker Viewer in a single file. IBM sessions (.CFS and .INI files), UNIX configurations (.CFG and .CFZ files) and FTP macros (.MAC files), used in the previous versions of Tun are retained; you can load this type of field and then integrate them into a workspace, you can run an FTP macro of the old format, you can

save individual sessions in .CFS or .INI format for IBM and .CFG or .CFZ format for UNIX. Note that it is now only possible to save more than one session in a workspace (.CWZ file).

Wizard

An emulation session wizard is proposed in Esker Viewer and enables you to associate a login macro and startup short cut with a user-defined session.

Open Esker Viewer without a workspace

1. From the **Start** menu, select **Esker Tun > Application Access > Generic Application Access**. Under Windows 8, right-click an empty area of the Start screen and click **All apps**. To start the application, find and click its tile.
2. Open a session or a workspace, using **File > New** or **Open**.

Start a workspace

From the **Start** menu, select **Esker Tun > Application Access**, and a configuration. Under Windows 8, right-click an empty area of the Start screen and click **All apps**. To start the application, find and click its tile.

This can be either a configuration setup by the Esker setup program, or a configuration that you added yourself.

Opening a session or workspace

You can open a session in two ways:

- From Esker Viewer, create a new session by selecting a session type from those proposed.
- Open one or more existing sessions recorded in one of the following formats: workspace (CWZ), UNIX (Asynchronous) (CFG or CFZ), IBM terminal or printer (CFS or INI), or FTP (FTP). You can also run an FTP macro (MAC) written with EScript.

Note:

CFG, CFZ, CFS, INI and MAC files correspond to formats used by previous Tun versions.

Creating a new session

1. Select **File > New**.
 2. Select a session type, and then click **OK**. The connection assistant will help you make the connection (except FTP session).
- > Depending on the type of session chosen, refer to the corresponding chapter in order to find out the connection parameters. For FTP file transfer sessions, refer to the **Network Resource Access** manual.

Configuring a session with the wizard

1. After selecting the connection assistant from the **New Session** box (see above), select the type of connection you want to create and click on **OK**. The connection box of the selected emulation type appears. Make the connection to continue.
2. Once the connection is established, you can record a connection macro. To do this, check **Start the recording of a new connection macro**, then click **OK**. The macro recording starts. You will return to the wizard after recording the macro.
 - If you don't want to record a connection macro, uncheck this, click on **OK** and move onto the short-cut creation step.
3. Check **Create a shortcut**, and then click **OK**.
4. You must then save your workspace in a **.CWZ** extension file before you select the location of the shortcut. Enter the shortcut name in the **Name** field.
 - Click **Modify link** or **Change icon** to modify the shortcut command line and its icon.
5. Select a shortcut destination from the three following options:
 - **Start Menu**: The shortcut will be added to the Application Access program group.
 - **Desktop**.
 - **Quick Launch Bar** (if you're using Internet Explorer 4).

The fourth option is available when performing an installation via the **Deployer**.

In case of a connection problem at this stage, you can:

- Create a shortcut despite the connection failure. Click **Next**. You will get to step 3.
- Redefine connection parameters to try to connect again: Click **Previous**.

Opening an existing session or workspace

Select **File > Open**. A standard Windows box then appears, offering selection of existing files. The file can be a workspace (.CWZ), a UNIX emulation configuration (.CFG or .CFZ), a 3270 or 5250 emulation configuration (.CFS or .INI), an FTP session (.FTP) or an FTP macro written in EScript (.MAC).

Saving a session or workspace

To return to sessions as you configured them, save their configuration parameters in a configuration file.

Select **File > Save**. A standard Windows save box then appears, asking you to choose the file name. You can save:

- All sessions opened in Esker Viewer in a .CWZ file. This file is the workspace, which you can retrieve as-is later.

- Only the active session when selecting the **Save** option, as a .CFG or .CFZ file (UNIX emulation), .CFS or .INI (IBM emulation), or .FTP (FTP session).

Modification of session connection parameters

You can customize sessions at any time. To do this, select **Session > Connection > Configuration**.

Editing tools

Esker Viewer provides a set of editing tools supplementing the basic functions:

- A resources editor for asynchronous emulation: you can display the additional resources used in the workspace (key panel files, image files, macro files, etc.).
- A macros editor (edit, save, run).
- A key panels editor: using an external executable (Panel Editor), you can create a key panel specifically for your emulation.
- A context editor for the asynchronous emulations: within the same tab box, you will find the elements for customizing a UNIX session (font, colors, background, etc.). This editor is totally compatible with contexts used in the previous Tun versions (.CTX).
- A terminal editor for asynchronous emulations: within the same dialog box, you will find all of the terminal customization files. This editor ensures total compatibility with terminals from previous Tun versions (.TER).

Note:

All these editors are accessible from the **Tools** menu.

Resources editor

Use the resources editor to associate additional resources to the current session. The standard resources are those files directly associated with the session when it is configured. They include:

- The terminal file (.TER).
- The customization files (keyboard, function keys, active zones, colors, screen background image, etc.).
- The key panel file where applicable (.PAN).
- The start and end macro files where applicable.



Examples of additional resources include:

- A second key panel file used in place of the standard key panel file (and displayed when executing an action designed to change key panels).
- A macro file called up by an action triggered when a key is pressed.

The resources editor also enables standard resources to be included in the session archive file (.CWZ): the session is established independently, without looking for its configuration files on the disk. In this case, even if they are moved, modified or deleted from their location, the session will be established using the parameters previously archived.

> **Display resources editor**

Select **Tools > Resources editor**.

- To add an extra resource (for example, a key panel file called up by an action), click  and select the file add.
- To delete a resource, select it and click .
- To archive the standard resources and make the session independent, check **Generate a self-sufficient configuration at saving**.

Customizing the interface

Using the Esker Viewer, you can add, delete, or move a toolbar buttons, menus, or menu options. You can also change the appearance of the buttons or the text of the menus and options.

Notes:

Most interface customization commands are accessible from **Tools > Customize**. If you delete the **Tools** menu or the **Customize** option, you can access this option by calling up the context menu from any toolbar or menu bar.

The content of the menu bar and the toolbars displayed depends on active session.

Adding a command or menu to a toolbar or menu bar

1. Select **Tools > Customize**, then select the **Commands** tab:
2. To modify a toolbar or a menu bar, call the bar.
3. Select the environment to modify the menu or toolbar from **Show menus**.
4. For each environment, the content of the **Category** and **Buttons** lists varies, as does the list of toolbars available from the **Toolbars** tab.
5. From the **Category** drop-down list, select the commands category you want:
 - All menus: select **Popup menus**.
 - All commands: select **All commands**.
 - The commands of a given toolbar: select the toolbar on which you want to choose the command.
 - The commands of a given menu: select the menu on which you want to choose the command.
 - A status bar indicator: select **Indicators**.

6. In the **Buttons** list, click on the command or menu you want to add to the toolbar or menu bar and drag it with the mouse to where you want it to be on the toolbar or menu bar. The button properties dialog box appears.

Note:

When adding a menu to a toolbar, clicking on this menu (after quitting **Customizations** mode) will call its options as if you had selected it from the menu bar.

The text of the ToolTip associated with a command appears under **Description** when you select a command from **Buttons**. The ToolTips appear as the mouse passes over each button on the toolbar if you selected the **Show ToolTips** option in the **Toolbar** tab.

7. Click **Browse** to choose images other than those proposed by default.
8. Enter the name of the image file you want to use. Click **Browse** to select the file from a directory. The formats authorized for illustrating a button are:
 - Bitmap images (.BMP)
 - Icons (.ICO)
 - Icons associated with a program or a library; enter an executable file name (.EXE) or a library name (.DLL).
9. Commands or menus may be represented by text (default or chosen text), an image, or both. Some commands have a default text and image, others simply a text.
10. In the **Button properties** dialog, select an appearance type. As applicable, select the type of image and/or enter the text. Click **OK**.

Changing the appearance of an item

In the **Commands** tab on the **Customize** dialog, click **Change selection**. You can then:

- Choose the appearance of the item.
- Access the item properties box: select option **Properties**, and then change the button appearance parameters.

Adding a separator

1. In the **Commands** tab of the **Customize** dialog, click **Change selection**.
2. Select the **Separator** option to add a separator before the item selected. Deselect this option to remove a separator if there is one placed before the item selected.

Note:

It is not always possible to add a separator, for example, in front of a menu option already separated from another by a separator, or above the first option of a menu. In this case, **Separator** is either selected or shaded.

Deleting an item

You can delete an item from a menu bar or toolbar, in three ways:

1. In the **Commands** tab of the **Customize** dialog, click **Change selection** and then select **Delete**.
2. Right-click an item and select **Delete**.
3. Drag the selected element off the menu bar or toolbar. When a small cross appears, release the mouse button.

Retrieving the default toolbars and menu bars

If you have changed a toolbar or menu bar, you can retrieve the initial contents of this bar as long as you have not yet saved your workspace.

1. In the **Toolbar** tab of the **Customize** dialog box, select the toolbar you want to reset.
2. Click **Default selection**. A message warns you that you are about to lose all the changes made to this toolbar or menu bar.
3. Click **Yes** to confirm the operation.

To retrieve the initial contents of all the toolbars, click on the **Default** button.

Toolbar management

Toolbars are managed using the **Toolbars** tab of the **Customize** dialog box.

> To create a new toolbar

1. In the **Toolbars** tab of the **Customize** dialog, click **New**.
2. Enter the name of the toolbar, and then click **OK**. The toolbar name appears in the list of toolbars, and the toolbar appears.
3. Add commands and menus, and reposition as desired.
 - **Show ToolTips**: Select this to have the ToolTip appear when the mouse passes over this command.
 - **Large buttons**: Select this to display the images in a large format. This option only works when using the windows accessibility function.

> Delete a toolbar

1. In the **Toolbar** tab in the **Customize** dialog, select a toolbar.
2. Click **Delete**. A delete confirmation message appears.
3. Click **Yes** to confirm deletion.

> Renaming a toolbar

1. In the **Toolbar** tab of the **Customize** dialog, select a toolbar.
2. Enter its new name in **Toolbar name**. The menu bar and toolbars supplied by default cannot be renamed.

> Display or hide a toolbar or menu bar

Use one of the following methods:

- Right-click any toolbar or menu bar and select or deselect the bar from the context menu.
- If the toolbar is floating, click on the toolbar close button (✕).
- In the **Toolbar** tab of the **Customize** dialog, select or deselect a toolbar or menu bar.

> **Moving or copying a menu or command**


1. If the menu or command to be moved or copied is in a toolbar, or has to be moved or copied into a toolbar, display the toolbar concerned.
2. Activate customization mode by selecting **Tools > Customize**.
3. In the toolbar or source menu, select the menu or command to be moved or copied.
 - To move the menu or menu option, drag it to the target toolbar, the menu bar or the target menu. Release the mouse when at the required location.
 - To copy the menu or menu option, hold down the **Ctrl** key and proceed as for a simple drag operation. Release the mouse and the **Ctrl** key at the required location.

Content of Tools menu

When customizing the **Tools** menu, you can create new commands and then incorporate them into the **Tools** menu. This means that you can execute other applications from Esker Viewer. For example, in the default version of Esker Viewer, you can start the function key-panel editor.

1. Select **Tools > Customize**.
2. Click the **Tools** tab. The content of the menu appears. The following are associated with each menu option:
 - The execute program commands
 - Any parameters (optional)
 - The default directory associated with the program (optional)
3. Check **Request parameters** to have the user enter the parameters of the command when executing it.

> **Adding an option to the Tools menu**


1. From the **Tools** tab in the **Customize** dialog, click .
2. Enter the command name, as it will appear in the **Tools** menu.
 - **Commands:** Enter the file path that corresponds to the command (C:\Program Files\Microsoft Office\Office\Winword.exe adds a Word start command to the menu).
 - **Settings:** (Optional) Enter parameters for the command here. (C:\temp\index.doc would open this document in Word).

- **Directory:** Enter the directory set by default for this program. This field is optional.
- **Request parameters:** Check this to have the user enter the command parameters when executing it.



> **Changing the name of an option on the Tools menu**

From the **Tools** tab in the **Customize** dialog, double-click a menu option and enter a new name.

> **Deleting an option from the Tools menu**

From the **Tools** tab in the **Customize** dialog, select a menu option and click .

> **Changing the order of options in the Tools menu**

From the **Tools** tab in the **Customize** dialog, click  or  to move the selected option up or down.

Options

You can alter the parameters of Esker Viewer. Select **Tools > Options**. Change the options, and then click **OK**.

- **Windows menu contains:** Enter the number of windows that the Window menu can contain (value between 0 and 10).
- **Number of recent files:** Enter the number of recent files that the **File** menu can contain (value between 0 and 10). The recent files are the last files in the session or workspace accessed by Esker Viewer.
- **Reload the last work space at startup:** To reopen this workspace the next time you start Esker Viewer, check this. It is a default setting.
- **Show splash screen at startup:** By default, a splash screen appears while Esker Viewer is starting. Uncheck this to hide this screen.
- **Confirm the session removal:** A confirmation window displays when the user closes the session.
- **Save Esker Viewer application positioning:** Check this to keep the same positioning of Esker Viewer application window at next startup.
- **Save on exit:** By default, when you quit Esker Viewer (or close a work space), any changes made to the session or workspace can be saved after confirmation by the user. Change this option either by asking for systematic quit and save, without confirmation (**Always**) or quit without save (**Never**).
- **Directories:** Enter the name of the default directory (absolute path) in the **Default load and save directory by default**. Click **Browse** to select it from your file tree.

Configuring the firewall

Connections to computers outside the local network can be put through a proxy firewall. Esker Viewer handles configuration of a Proxy firewall.

1. Select **Tools > Firewall**.
2. To configure the firewall, check **Use a Proxy server** (Socks protocol).
3. Change the options by following the instructions below and then clicking **OK**.
 - **Proxy Server:** Enter the name or IP address of the server that is used as firewall (only enter a name if your system uses a DNS).

Note:

You can use the proposed list of servers: this corresponds to the servers recorded locally in the servers table (**HOSTTAB**) and on the NIS server if available (NIS resources are shown in yellow).

- **Port number:** By default, the port number corresponding to the Socks protocol is 1080. If your configuration uses another port, enter the new value in this field.
- **Use for local addresses:** By default, access to all the machines on the LAN will be via the firewall configured in this way. To avoid this for connections to a local address, uncheck this.
- **Use these settings for all Tun applications:** By default, the firewall configuration is valid for all Tun applications. To have other Tun applications use other firewall parameters, uncheck this.
- To return to the last settings of the Tun applications firewall, click **Restore general settings**.

Configuring SSL

Esker Viewer enables you to use SSL to cipher your data between Telnet emulation customers and SSL servers. Select **Tools > SSL**.

- **Use SSL:** Check this to cipher the exchange data using the SSL protocol.
- **Certificate:** The certificate contains the server public key that authenticates the server. The certificate is generated on the SSL server in a file (.PEM). Copy and paste the content of the file into the **Certificate** field.

> Configuring the SSL server

To use the SSL protocol, you must have an SSL server. This server can be:

- The host, if the Telnet server supports SSL.
- A machine having an SSL server.
- A machine having an SSL server using the Socks v4 protocol.

Note:

Esker supplies SSL server software.

To configure your SSL server in Esker Viewer, you must either modify the parameters of your SSL properties box or modify the connection parameters of your session, depending on the type of server you are using.

- **Host which supports SSL:** In the connection parameters of your emulation session, replace the Telnet port number by the port number of your SSL server.
- **SSL server without Socks protocol:** The client connects to the SSL server, which deals with connection to the host. This configuration offers the advantage of masking the host from the end user.
In the connection parameters of your emulation session, replace the server name by the IP address name of the SSL server with which you wish to connect (only enter a name if you have a name server).
- **SSL server with Socks v4 protocol:** The client connects to the SSL server using the Socks v4 protocol. The server redirects the connection towards the host transparently as regards the user. This configuration offers the advantage of not needing any modification to the existing emulation sessions.

Click on the SSL server tab to configure the SSL server for the Socks v4 protocol:

- **Use an SSL server (SOCKS protocol):** Check to use the SSL server.
- **SSL server name:** Enter the name or the IP address of the SSL server to which you will connect (only enter a name if you have a name server).
- **Port number:** The default port number is 8197, and corresponds to the port number used by the SSL server supplied by Esker

Configuring SSH

Esker's SSH offers the ability to use the SSH connection type to connect to a host via a network. Select Tools > SSH.

> Configuring SSH

On the General tab, the box displays the current SSH version information.

- **Enable SSH1/SSH2:** Check one or both options to enable the protocol level for SSH. At least one option must be selected. If both are selected, Tun will use the highest level negotiated by the host.
- **Enable Compression:** If the host supports this option, Tun will compress the SSH data. This is most helpful for slower connections.

Host Authentication relies on a public key received from the host. To counter "Man in the Middle" attacks, the key may be copied to the host prior to the initial SSH connection. The key may be copied directly into the text box displayed and will be saved with the workspace. The key may also be added to the other hosts files using the Edit buttons, and will be stored on the local drive. These key files are located in the two directories

<%ALLUSERSPROFILE%>\Application Data\Esker\SSHX and <%USERPROFILE%>\Application Data\Esker\SSHX.

- Check **Use system known hosts file** to allow Tun SSH to use the keys stored in that file to authenticate the host.
- Check **Use user known hosts file** to allow Tun SSH to use the keys stored in that file to authenticate the host.
- Check **Accept unknown hosts** to allow Tun SSH to connect to host who's key has not previously been stored.
- Check **Ask Conformation for unknown hosts** to provide the user with prompts for accepting the connection and saving a new key. New keys are saved in the user known hosts file.

User Authentication may be used if the host requires a user public key for authentication. This tab allows the user to generate a private/public key pair for this purpose. Once the keys are generated, the public key may be copied directly out of the dialog. The public and private keys are also stored in the local directory <%USERPROFILE%>\Application Data\Esker\SSHX

- Click **Generate RSA Keys** to generate an SSH2 RSA public/private key pair.
- Click **Generate DSA Keys** to generate an SSH2 DSS/DSA public/private key pair.
- Click **Generate SSH1 Keys** to generate an SSH1 RSA public/private key pair.

The key length may also be modified by editing Key Length text box. We strongly advise you to use a key length of 1024 bits or higher; a lower number may compromise security.

The Advanced tab contains a list of the supported key algorithms, ciphers, Message Authentication Codes (MAC), and client authentication methods. These items are used during the SSH negotiation between the SSH client and server. To limit the SSH client negotiation methods, simply edit these text boxes directly and remove the unwanted methods.

Packager

You can design your own Esker Viewer in the form of a customized, independent executable file, in other words, one that contains all files needed for execution. Only ActiveX component type files (.OCX) and libraries (.DLL) should be given as executable complements.

This gives users an executable file containing a fully customized workspace. It does not allow them to open any other workspace, nor to modify the workspace. The user only has the parameters and functions chosen by the person who created the executable file. You can redistribute executables designed to perform different tasks.

To create a new executable:

1. Define the working environment (customizing the workspace and its various sessions, customizing the menus and toolbars).
2. Save of the customized workspace in a file with extension .CWZ.

3. Select **Tools > Packager**.
4. Enter the name of the executable to create (.EXE) and that of the workspace (.CWZ).
Make sure that you do not overwrite the default Esker Viewer executable supplied by Esker.
5. Click **Generate**.

You can now distribute it to any user that has Tun installed.

Asynchronous Emulation

Asynchronous emulation involves the bi-directional exchange of characters between the PC and the server. There's no precise mechanism governing the exchange. You can type characters at the same time that the server returns others them. Every session is associated to a terminal. If you have existing context files (.CTX), you can keep using them by loading them when configuring the session.

Creating an asynchronous emulation session

1. See “Esker Viewer” on page 8 to learn more about the various emulation session opening methods. Select an asynchronous session.
2. Fill out the fields required for connection, following the instructions below.
3. Click **OK**.
 - **Connection type:** Select the connection type you require: connection via TCP/IP network (**Telnet**), RS232 link (**Serial**), modem (**TAPI modem**), or SSH. The connection settings displayed in the lower right part of the dialog box vary depending on the type of connection chosen.
 - **Terminal:** Select the type of terminal you want to emulate from the drop-down list. The terminal type you choose depends on the server type or the type of application you want to use.
 - **Display settings:** Select parameters to define the session: font, screen dimensions, font color and style, screen background, function key-panel, use of the mouse, etc. This field is optional.
 - **Close session on confirmation:** Check this to display a confirmation dialog box on exiting the program.

TCP/IP connection settings

- **Host name:** Enter the **name** or **IP address** of the host to connect to (only enter a name if you have a name server).

Note:

The proposed list of servers corresponds to the servers recorded locally in the servers table (HOST-TAB) and on the NIS server, if available (NIS resources are shown in yellow).

- **Telnet port number:** By default, the port number corresponding to the Telnet protocol is 23. If your configuration uses another port, enter the new value in this field.

RS232 connection settings

- **Port:** Select the PC communication port from the drop-down list (COM1 through COM32).
- **Baud rate:** The transmission speed over an asynchronous line is measured in **bauds** (bits per second). The baud rate can be set between 75 and 115200. The value should match the speed set on the server (in the file `/etc/gettydefs`).
- **Data bits:** This field describes the number of significant bits used to make a byte. This number is almost always either 7 or 8. Check the setting on your host machine to be sure.
- **Stop bits:** Either 1 or 2 bits mark the end of a byte.
- **Parity:** The **parity bit** provides protection against transmission errors. The parity bit can be either **even** or **odd**. **None** means that no parity bit is sent after a byte. In 8-bit transmissions, **Space** or **Mark** can be used to set the last bit in a byte to 0 or 1 respectively.

Advanced configuration

Click on the **Advanced** button to complete configuration of your connection:

- **I/O buffer size:** This field defines the size of the Input/Output buffers (in bytes). The default value of 2048 is usually sufficient.
- **Flow control:** Flow control keeps the Input/Output buffers from overflowing and losing data. It's very important that Tun is set the same as the host machine.
- The **Xon/Xoff** and **Xany/Xoff** flow controls are the most commonly used on UNIX servers. With **Xon/Xoff**, when the buffers in the PC in emulation are 75% full, Tun sends a DC3 (^S) character to the host asking it to suspend data transmission. When the buffers are 75% empty, Tun sends a DC1 (^Q) character to request that the host resume data transmission. When using **Xany/Xoff** the PC in emulation still sends a DC3 to suspend transmission, but can send any character to resume.
- Some UNIX servers handle flow control directly through cabling. Instead of using special characters (DC1 & DC3), electronic signals are sent when the PC's buffers are full. This is known as hardware handshaking. In general, two types of hardware handshaking are used: **DTR** and **DSR** signals or **RTS** and **CTS** signals.

Modem connection settings

Tun uses the TAPI standard interface to configure your modem settings. This interface offers a simple modem installation procedure with automatic detection and a single configuration. Once configured, the modem can be used by other communication applications. The **TAPI modem** connection-specific settings are described below:

- **Modem:** This list contains all the modems installed on your PC.
- **Configure the modem:** Click this to display the selected modems settings in the list. The dialog box that appears is the same one that you use to configure your modem from the **Modems** icon in the Control Panel.

- **Phone number:** Enter the telephone number of the host system in this field, unless you prefer to dial manually. Use a comma to insert a two-second pause during dialing. This may help when obtaining external lines in office buildings or for dialing internationally. If you complete this field, the emulator will attempt to dial up when the session opens.
- **Timeout:** If no response is heard from the remote host in the time you specify, the emulator considers the connection as failed. Thirty seconds is usually a reasonable value.
- **Dialing rules:** Check this to automatically modify the phone number dialed according to the geographic area called or the calling area.
- **Dialing rules:** Click this button to configure the dial settings.
- **Country code:** Select the country to dial from the list. The selected country code will precede the server phone number to form the dialed number.
- **Area code:** Specify the area code (optional). This code is used in specific countries only.
- **Call location:** In the drop-down list, select the configuration that corresponds to your calling area. A configuration contains several parameters that simplify dialing according to your phone settings. To change these parameters, click **Advanced**. The dialog box that appears can also be displayed from the **Modems** icon in the Windows Control Panel.

SSH connection settings

SSH General tab:

- **Enable SSH 1:** Select this option to activate SSH protocol version 1 for this session. This option is enabled by default. Protocol version 1 permits the use of Triple DES (3DES) and Blowfish encryption ciphers.
- **Enable SSH 2:** Select this option to activate SSH protocol version 2 (the newer version) for this session. This option is enabled by default. Protocol version 2 permits the use of Arcfour encryption as well as Triple DES (3DES), Blowfish and CAST128 ciphers in CBC mode.

Note: If both SSH versions are checked, the highest SSH version supported by the host will be used.

- **Enable compression:** Allows the client to compress data before encryption. If the host also supports compression, then transmissions between the client and host are compressed. For large files, this decreases transmission times over slower connections.

SSH Host Authentication tab:

There are three locations for storing host authentication keys. All three locations will accept both SSH1 and SSH2 level keys. You can recognize each type of key by its format. SSH1 keys contain the host name or IP address, the RSA key length, the decimal exponent and modulus, followed by optional comments. The SSH2 keys contain the host name or IP address, the type of key ("ssh-rsa" or "ssh-dss"), the base 64 encoded public key, followed by optional comments.

- **Known Hosts:** Copy and paste host keys directly into the box. Hosts listed here are treated as known hosts for this workspace, and are eligible for public key authentication. This box accepts both SSH1 and SSH2 level keys.
- **Use system known hosts file:** Select this option to use the system known hosts file. Click Edit to add, change, or delete keys from this file. The system known hosts file is set up by the administrator, and lists known hosts that can be used by all users on a particular machine.
- **Use user known hosts file:** Select this option to use the user known hosts file. Click Edit to add, change, or delete keys from this file. The user known hosts file lists known hosts that can be used by a specific user on a particular machine.

If you disable this option, you will not be able to save new host keys, and you will see a warning message each time you try to log on to an unknown host, even if you have logged on to it before.

- **Accept unknown hosts:** Allows you to connect to hosts that have not been saved as known hosts. Once you have connected to a host this way, Tun asks if you want to add that host's key to the user known hosts file, if using that file is enabled.
- **Ask for confirmation for unknown hosts:** If this option is enabled, a message appears whenever you try to connect to a host that has not been saved as a known host. Once you have confirmed that you want to connect to an unknown host, your connection to the host is established.
- **Reset defaults:** Click this button to change each setting on this tab back to its default value. The defaults are based on the current Open SSH standard at the time this version of Tun was developed.

SSH System or User Known Hosts File dialogs:

These dialogs are accessed by clicking the Edit buttons in the SSH Host Authentication tab. They allow direct access to the known hosts files.

- **Known Hosts:** Copy and paste host keys directly into the box. Hosts listed here will be treated as known hosts and eligible for public key authentication. The keys here will be saved in the file listed below. This box accepts both SSH1 and SSH2 level keys. This box will be grayed out if you do not have write access to the file.
- **File path:** Shows the location of the file containing the host keys listed in the text box above.

Note: If the user has read only access to the file, the File path and Known Hosts boxes are uneditable.

SSH User Authentication tab:

On this tab, you can generate public/private key pairs. Once you generate the key pair, copy the public key to the host for use in key authentication.

- **Generate keys:** Click these buttons to generate a user authentication key pair (public and private) described below. These may be used to authenticate the user on the server machine.
- **RSA public key:** Lists the public key generated using the RSA algorithm. This key is used with the SSH 2 protocol.
- **DSA public key:** Lists the public key generated using the DSA algorithm. This key is used with the SSH 2 protocol.
- **SSH1 public key:** Lists the public key generated using the RSA (from SSH1) algorithm. This key is used with the SSH 1 protocol.

Note

: To delete a key, you must delete the key file from the appropriate directory:

Key files

RSA keys: rd_isa and rd_isa.pub

DSA keys: id_dsa and id_dsa.pub

SSH1 keys: identity and identity.pub

Key file directories

Win 9x: C:\Windows

Windows NT 4: C:\Winnt\Profiles\

Windows 2000 and XP: C:\Documents and Settings\\Application Data\Esker\SSHX

- **Key length:** Sets the size of generated keys in bits. The value must be between 0 and 9999; the default is 1024. Note that entering a value lower than 1024 could result in compromised security. Entering a very large value can significantly extend the time it takes to generate the key.

Note: After changing the key length, click Apply before generating a new key or the program will use the previous key length.

SSH Advanced tab:

To change the values, simply add, change, or delete the text in each box. The items in each box on this tab are comma-separated.

- **Public key algorithms:** Lists the algorithms supported by the client.
- **Ciphers:** Lists the ciphers supported by the client.
- **MACs:** Lists the MACs supported by the client.
- **Authentication methods:** Lists the authentication methods supported by the client.
- **Reset defaults:** Click this button to change each setting on this tab back to its default value. The defaults are based on the current Open SSH standard at the time this version of Tun was developed.

Dynamic Data Exchange

DDE is a standard inter-application communication protocol defined by Microsoft. It allows Windows applications that support this protocol to exchange data with each other. Two applications that take part in a dynamic data exchange "engage" in a DDE conversation. The application that starts the DDE conversation and that wants to access data is called the DDE client. The application that answers the client and that has access to the data and can transfer it during the DDE conversation is called the DDE server. An application can take part in several conversations at the same time: It can behave as a client in some conversations and a server in others.

To allow the exchange of data between the client and server applications, the data must be identified by an application name, a topic name and an item name. At the start of the DDE conversation, client and server determine the names of the application and topic. Next, a particular item of data to exchange is specified.

- **Application:** This is the name of the server application to which the client sends its data requests. Application names are sometimes known as service names.
- **Topic:** A topic is a group of items that can be used in a DDE conversation. For example, with applications that manage documents as files, a topic is typically a file. The system topic is a special topic that supplies a group of server application data items that can be widely used by other applications. This topic is always available when the server application is run.
- **Item:** An item is a unit of DDE data linked to the topic that is changed between applications during the conversation. For example, this could be an individual cell in a spreadsheet.
- **Conversations:** One of the powerful features of DDE is that a client application can send commands or submit requests to a server application that has been designed to recognize and accept them. The type of command or request that a server can accept depends on the server.

Put simply, a DDE communication between a client application and a server application is like a phone conversation:

1. The client begins by initiating the conversation using a function like DDEInitiate, which defines a DDE link. This function typically contains an application name, a topic and possibly an item that function like a phone number since they let the client identify the server with which it wants to communicate.
2. Once the link is defined, and a channel number attributed to it, the client can use this reference number to:
 - send data items to the server using a function like DDEPoke (the name may change depending of the application)
 - fetch data items from the server using a function like DDERequest (the name may change depending of the application)
 - send commands to the server using a function like DDEExecute (the name may change depending of the application)

3. Finally, when the client has finished communicating with the server, it "hangs up" by ending the DDE link using a function called DDETerminate (or something similar). Note that closing the server or client application always ends the links between the two.

Use

Emulwin is used as the server application name. In Tun, a topic corresponds to an open session. Each topic has its own name. The topic name is the name of the host. If more than one session is opened on the same host machine, the topic name, like the session name, is the name of the host followed by a colon (:) and a number.

For example, the first session opened on the host machine *risc* is called "risc", the second "risc:1" and the third "risc:2". The system topic is called **System**. Application names associated with topic names are used to start and end DDE exchanges ("initiate" and "terminate").

The contents and structure of topic items, as well as the commands, depend on the server application. Server and client can exchange recognized items (using "poke" and "request") and the client can send supported commands to the server (using "execute"). The items recognized by the Tun DDE server for its standard system topic and the other topics are described below.

Note:

Tun supports the "Execute Control 1" protocol, which returns information in reply to a command request from a client application. Tun uses this information to supply an error message when a command fails. For example, you can declare an item "name", recognized by the topic, using the command [Result(name)]. This item contains the results of the commands that are subsequently executed. If a command fails on execution, the error message returned by the server is placed in "name".

Items supported by the system topic

- **SysItems**: Returns a list of system topic items.
- **Topics**: Returns a list of available topics, that is the sessions opened in Tun.
- **Formats**: Returns a list of supported formats.
- **Protocols**: Returns a list of supported protocols.
- **Help**: Returns help on the use of the DDE server.

Commands supported by the system topic

- **Open ("configurationfilename")**: Opens a new session (that is, one or more pre-defined sessions); an error occurs if Tun can't find the configuration file, if the file contains errors, or if a configuration file is already open.
- **Close**: Closes all the emulation sessions.
- **Resize(0)**: Reduces the size of the server application window.
- **Resize(1)**: Returns the window to its original size (before it was reduced or increased).
- **Resize(2)**: Increases the size of the server application window.

- **Result(name)**: Defines the item that contains the message returned by executed commands.

Items supported by the other topics (emulation sessions)

- **TopicItemList**: Returns a list of the topic's items.
- **Formats**: Returns the formats supported by the topic.
- **Screen (page, line, columns, length)**: Returns a selected string of given length.
- **ScreenRect (page, start line, start column, end line, end column)**: Returns a rectangular selection.
- **Host**: Sends data to the host machine.

Commands supported by the other topics (emulation sessions)

- **SendData("string")**: Sends a string over the connection. The data can't be sent if a file transfer is taking place.
- **Macro("name", arg)**: Executes the macro "name", passing it the optional parameter "arg": An error occurs if the macro is already running or if a file transfer is taking place.
- **Terminate**: Closes the emulation session. The session isn't closed if a file transfer is taking place.
- **Result(name)**: Defines the item that holds the messages returned by the executed commands.

Command syntax

Commands must respect the following syntax:

```
[command(arg, ...)]
```

You can pass command arguments directly as long as they only include the following characters: a-z, A-Z, 0-9, _ and \$. To pass a more complex character string as an argument (including, for example, spaces and slashes), put it in quotes and follow these instructions:

- To pass the quotation marks character as an argument, you must double it: So the command [command("")] passes single quotation marks as an argument.
- To pass the backslash character "\", you must double it, otherwise it's ignored: For example, the command [command(\x)] passes only the character "x"; you must use [command(\\)] to pass the character "\".

The command **SendData("string")** also uses special Tun encoding in addition to the DDE encoding:

| Notation | Meaning |
|----------|------------------|
| \e | Escape character |

| | |
|-------------------|---|
| <code>\E</code> | Escape character |
| <code>\n</code> | Line feed |
| <code>\r</code> | Carriage return |
| <code>\t</code> | Tab |
| <code>\b</code> | Backspace |
| <code>\f</code> | Form feed |
| <code>\s</code> | Space |
| <code>\\</code> | Backslash "\" |
| <code>\0xn</code> | Character's hexadecimal value (n: 0 through FF) |
| <code>\0n</code> | Character's octal value (n: 0 through 377) |
| <code>\n</code> | Character's decimal value (n: 0 through 255) |

For example, to pass a single backslash character (\) as an argument of **SendData**, use the Tun notation `\\`, doubling the backslash: You must then double these backslashes for DDE decoding to give [**SendData**("\\")]. To pass a line feed character, you must use the notation `\n`, and double the backslash character: [**SendData**("\\n")].

Examples

You can use the topics, items and commands described above in application programming. The following examples can be used in Word (WordBasic) and Excel (Visual Basic for Excel).

Initializing a DDE conversation

```
Word  DDEInitiate(Application$, Topic$)
      ex: channum=DDEInitiate("Emulwin", "System")

Excel  object.DDEInitiate(app, topic)
      ex: channum=Application.DDEInitiate(app:="Emulwin", topic:="System")
```

Sending data

```
Word  DDEPoke Channel, Item$, Data$
      ex: DDEPoke channum, "Host", "text"

Excel  objet.DDEPoke(channel, item, data)
      ex: Application.DDEPoke channum, "Host", "text"
```

Retrieving data

```
Word  DDERequest$( Channel, Item$)
      ex: data$=DDERequest$( channum, "ScreenRect(0,8,41,12,43)")

Excel  objet.DDERequest(channel, item)
      ex: data=Application.DDERequest(channum, "ScreenRect(0,8,41,12,43)")
```

Sending characters

Word DDEExecute Channel, Command\$

ex: DDEExecute channum, "[SendData("+Chr\$34+"text"+Chr\$34 +")]"

Excel objet.DDEExecute (channel, string)

ex: Application.DDEExecute channum, "[SendData("+Chr\$34 + "text" + Chr\$34+")]"

Closing the conversation

Word DDETerminate Channel

ex: DDETerminate channum

Excel object.DDETerminate(channel)

ex: Application.DDETerminate channum

Excel example

This example illustrates an alternative method to **Paste with link** to regularly update data in an Excel spreadsheet from a UNIX server. It implements DDE between a Tun emulation session (DDE server) and an Excel macro (DDE client). Tun includes a DDE example that uses a DDE link between data on a UNIX host and an Excel spreadsheet running a macro. To view the results, do the following:

1. Copy the file ddedemo2.sh to the /home directory on the UNIX server from C:\...\Tun\Emul\Demo\Dde.
2. In Tun, create a configuration in which the first session is a link to your UNIX server, and save it as "ddeconf.cfg" in the directory C:\...\Tun\Emul\Demo\Dde.
3. Start Excel and open the file ddedemo2.xls in the directory C:\...\Tun\Emul\Demo\Dde.
4. Open the spreadsheet "Chart1" or "Sales" to view the results.
5. To run the macro, choose **Tools > Macro**, select DDEMacro, and then click **Run**.

Note:

The macro refers to C:\Program Files (x86)\Tun\EMUL\emul32.exe in its code, which requires that your application should also be in the C:\Program Files (x86)\Tun\EMUL directory. Otherwise, enter the path corresponding to your application directory in the ddedemo2.xls macro.

The macro uses the following Visual Basic for Excel DDE commands:

object.**DDEInitiate**(app, topic):

- **app**: DDE application server name
- This function returns the number of the channel opened for the DDE link.

object.**DDERequest**(channel, item):

- **channel**: channel number returned by the DDEInitiate method
- **item**: item requested

```
object.DDEExecute(channel, string):
```

- **channel**: channel number returned by the DDEInitiate method
- **string**: command defined in the receiving application

```
object.DDETerminate(channel):
```

- **channel**: channel number returned by the DDEInitiate method

In all the above methods, the "object" is optional and refers to the "Application" object (Microsoft Excel).

This macro is written in Visual Basic for Excel. It performs the following operations:

1. Starts Tun.
2. Initializes a DDE link between an Excel spreadsheet and Tun's system topic using the method DDEInitiate.
3. Opens a specific configuration using the method DDEExecute and the command Open, and reduces Tun's main window with the command Resize.
4. Fetches a list of open sessions using the method DDERequest with the item Topics, and selects the first session from the list (corresponding to the UNIX server).
5. Initializes a DDE link between the Excel spreadsheet and this session using the method DDEInitiate.
6. Starts the emulation session by executing a Tun connection session and sending commands using the method DDEExecute with the command SendData.
7. Updates the data from the emulation screen using the method DDERequest with the item ScreenRect.
8. Closes all the emulation sessions using the method DDEExecute with the Close command from the system topic.
9. Closes the open DDE links using the method DDETerminate.

The code for the sample macro is as follows:

```
Sub DDEMacro()  
RetVal = Shell("C:\PROGRA~1\TUN\EMUL\EMUL32.EXE", 1) 1  
canal1 = Application.DDEInitiate(app:="EMULWIN", topic:="System") 2  
Application.DDEExecute canal1, "[Open(" + Chr$(34) + "demo\\\\d- 3  
e\\\\ddeconf.cfg")] [Resize(0)]"  
listTopics = Application.DDERequest(canal1, "Topics") 4  
session1$ = listTopics(1)  
waitTime = TimeSerial(Hour(Now()), Minute(Now()), Second(Now()) + 3)  
Application.Wait waitTime  
canal2 = Application.DDEInitiate(app:="EMULWIN", topic:=session1$) 5  
DDEExecute canal2, "[Macro(" + (Chr$(34) + "demo\\\\d- 6  
e\\\\dde\\\\loginde.mac +  
Chr$(34) + ")]"  
waitTime = TimeSerial(Hour(Now()), Minute(Now()), Second(Now()) + 20)
```

```

Application.Wait waitTime
DDEExecute canal2, "[Senddata(" + Chr$(34) + "cd home" + "\\r" + Chr$(34) +
")]"
DDEExecute canal2, "[Senddata(" + Chr$(34) + "./ddedemo2.sh" + "\\r" +
Chr$(34) + ")]"
For k = 1 To 20
    waitTime = TimeSerial(Hour(Now()), Minute(Now()), Second(Now()) + 1)
    Application.Wait waitTime
    sales = DDERequest(canal2, "ScreenRect(0,8,41,12,43)")
    For i = LBound(sales) To UBound(sales)
        s$ = sales(i, 1)
        j = i + 4
        cell$ = "D" + j
        Worksheets("Sales").Range(cell$).Value = Val(s$)
    Next i
Next k
Application.DDEExecute canal1, "[Close]"
Application.DDETerminate canal1
Application.DDETerminate canal2
End Sub

```

Note:

The TimeSerial function and the Wait method are used in the macrocode to synchronize it with Tun. The synchronization timeout value is:

waitTime = TimeSerial(Hour(Now()), Minute(Now()), Second(Now()) + 1)

If necessary, you can change the value by increasing the number of seconds added to Second(Now()).

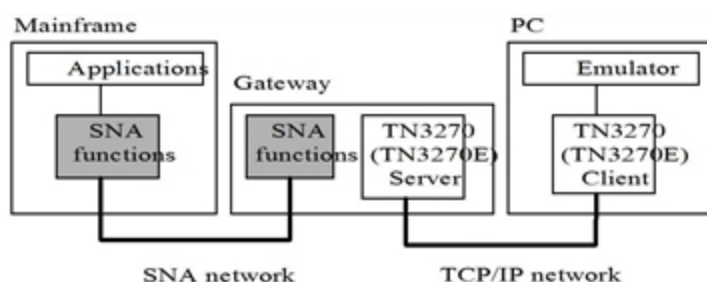
Synchronous Emulation

The synchronous terminal emulators provide access to the IBM MainFrame (3270 emulation) and AS/400 (5250 emulation) servers.

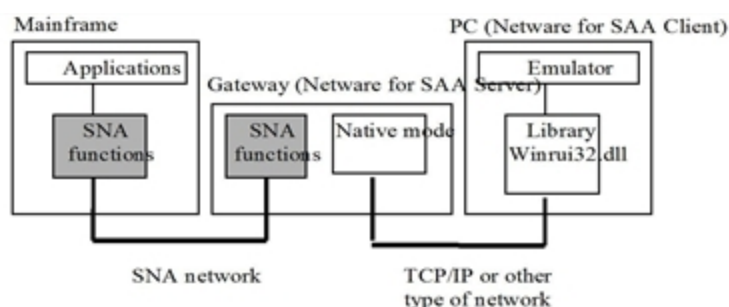
Synchronous emulation connection protocols

To communicate with this type of server, the terminal emulator can establish a connection based on the following protocols:

- Connection using the Telnet 3270/5250 protocol (TN3270/TN5250) over TCP/IP.
- Connection based on the extended version of this protocol: TN3270E/TN5250E.
- Connection via a UNIX SNA-TCP/IP gateway in TN3270/TN5250 or TN3270E/TN5250E mode. The PC emulates a 3270(E)/5250(E) terminal by connecting to the gateway as if it was a server.

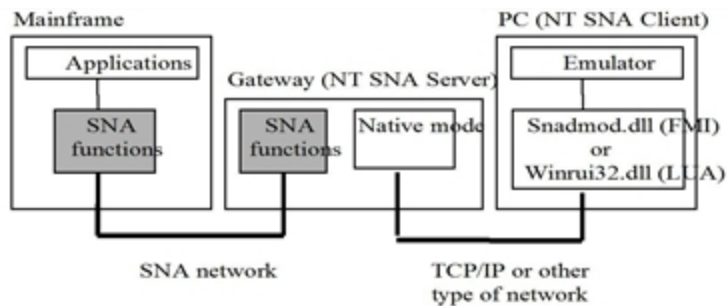


- (3270) Native mode connection via a Netware for SAA gateway. The emulator communicates with the **Winrui32.dll** library located on the PC and supplied by Novell. This DLL then manages the connection with the gateway:



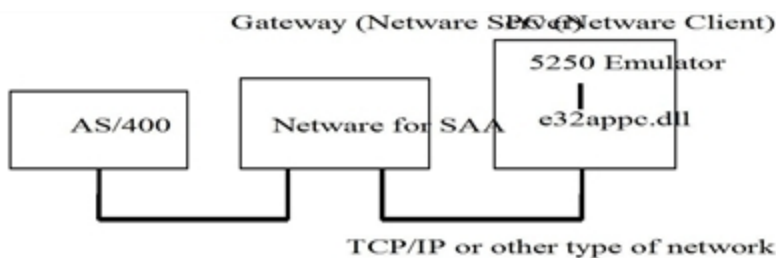
To use Esker's IBM 3270 emulator via a Netware for SAA gateway in native mode, you must first install the **Novell Netware client** on your PC (which includes **Winrui32.dll**) as well as the **IntranetWare for SAA client** from **Novell**.

- (3270) Native mode connection via a Microsoft SNA Server gateway (LUA or FMI). The emulator communicates with a DLL library that's specific to the gateway on the PC. The DLL then manages the connection with the gateway.
FMI is the access mode used for access to Terminal (3270) or Printer (3287) type LUs defined on the gateway. These latter must correspond to a terminal or printer LU on the MainFrame server. In FMI native mode, the emulator communicates with library **Snadmod.dll**, supplied with the SNA client.
LUA is the access mode used to access type 0, 1, 2 and 3 LUs on the MainFrame server, in other words terminals and printers, but also other types of LUs. In LUA native mode, the emulator communicates with library **Winrui32.dll**, supplied with the SNA client and which implements the RUI API allowing use of type 0, 1, 2 and 3 LUs.



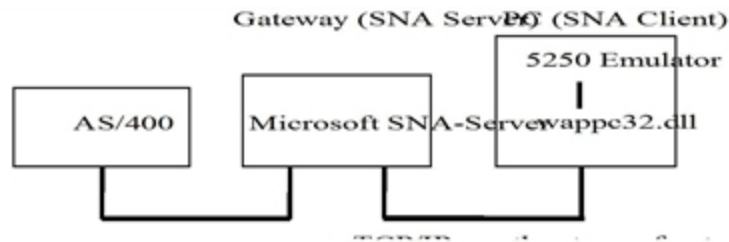
To use Esker's 3270 IBM emulator in native mode with a Microsoft SNA Server gateway, you must first install the client part of SNA Server on the PC (SNA Server Client, version 3.0) supplied with Microsoft® **BackOffice**®. See Microsoft's documentation for installation details.

- (5250) Native mode connection through a Netware for SAA gateway over TCP/IP or IPX/SPX. Netware for SAA is integrated as follows:
 The emulator sends its commands to an APPC function library named **e32appc.dll**, supplied by IBM.
 The API transforms these requests into statements that comply with the LU6.2 protocol.



The PC running the emulator must have **Novell's Netware** client (which comprises API **e32appc.dll**) as well as **Novell's Intranetware for SAA** client.

- (5250) Native mode connection through a Microsoft SNA Server gateway over TCP/IP or IPX/SPX. Microsoft SNA server is integrated as follows:
 The emulator sends its commands to an APPC function library supplied by IBM (**wappc32.dll**).
 The API transforms these requests into statements that comply with the LU6.2 protocol.



The PC running the emulator must have the **SNA Client** (SNA Server Client, version 3.0 supplied with Microsoft® BackOffice®), including the API **wappc32.dll**.

Multiple connection attempts

The Esker IBM 3270 and IBM 5250 emulators also permit multiple connection attempts. Define several configurations for connection to a MainFrame server or AS/400 system. If the first connection configuration used fails, then the next configuration is used. The order in which the configurations are used, can be either the order given by the list of configurations or a random order.

This type of connection is particularly useful if you have a number of redundant gateways for accessing a given server. It enables you to connect to this server without stipulating a specific gateway. Thus, if one gateway is unserviceable, the connection will use another.

Opening an emulation session

When you select a synchronous IBM type session (see **Using Esker Viewer** in the **Esker Viewer** chapter), the connection dialog box appears. This box contains two tabs:



- The **Session** tab enables you to specify the type and the parameters of the connection to be made.
- The **Status** tab gives the state of the connection (the state is **Unconnected** at the moment of connection).

Options

- **Close Session on Confirmation:** Check this to display a confirmation dialog box on exiting the program.
- **Startup Connection:** If checked, the program will automatically reconnect at the host.
- **Reconnection after a shutdown:** If checked, the program will automatically re-establish the connection if it's interrupted by the host.
- **Terminal Type:** This field sets the type of 3270/5250 terminal to use.
- **Detects the model (3270/3270E):** This enables you to change the terminal model specified at connection to meet the needs of the application. Check this to adapt the terminal model automatically to the application used.
- **Host name:** Enter the name of the server or its IP address. You can also select a server from the drop-down list (only enter a name if you have a names server). In the case of a connection via a gateway, the host name corresponds to the gateway name.


- **Port:** The default port number is 23. You can change this number, if necessary.
- **Uses TN3270E/Uses TN5250E:** Selected by default. The TN3270E/TN5250E protocol is used to make the connection and if the server contacted cannot support the TN3270E/TN5250E, a TN3270/TN5250 connection is then negotiated. Uncheck this to use the TN3270 protocol without extension.
- **Logical unit name** (only for TN3270E connection): Enter the name of the logical unit (LU) you want to work on. The LU gives the type of resource to which you are connecting on the MainFrame server.
- **Unit name:** (5250/5250E) Optional field. If you enter nothing in the field, the AS/400 system will attribute a default name to your client PC to uniquely identify it. You can, however, enter a terminal name to identify your machine.
- **System name:** (5250) Enter the name or IP address of the server or select one from the drop-down list (only enter a name if you have a names server).

Click on the **Advanced** button to customize the advanced properties of the 5250 session.

- **Identification:** To enable automatic login, fill the required parameters when login to the AS/400 system: user name and password, library, menu and program (these settings correspond to the fields that appear on the login screen of an AS/400 system).
- **Encrypt:** Check this to encrypt the identification password on login.
- **Terminal parameters:** You can modify the default configuration of the AS/400 terminal on which the connection is established. Specify a new **Code Page** and/or **Charset** to use if necessary. Click **Default** to reset these values to your emulation configuration defaults (given by the character set).
The **UserVar Text** and **UserVar Binary** lists let you configure other advanced parameters of the terminal. To add an additional IBM parameter (text or binary), click . Then, enter the name of the IBM parameter in the upper left part of the field, and its associated value in the right part. To remove a character from the list, select it and click .
- **Mode name:** By default, the mode name is QPCSUPP. This mode is specific to the gateway and APPC library. Typically, you don't need to change this value. Contact your network administrator if you want to change it.
- **Local LU name** (only for SNA Server connections): For connections through an SNA Server gateway, enter the name of the local LU as it's registered on the gateway. This field must be completed.
This field isn't shown for Netware for SAA gateway connections. The name of the local logical unit must be correctly registered on the gateway for the connection to succeed.
- **User name/Password:** Enter the user name and associated password that allow you to connect to the AS/400 system. The user name and password are defined on the gateway.




Multiple connection configurations

The list gives all the connection configurations already defined.

- To define a new connection configuration and add to the list, click . Then, fill out the connection parameters as if defining a new connection.
- To change the configuration settings of an existing connection, click on it.

Note:

For more details on the parameters of a connection configuration, refer to the description of the **Parameters** tab corresponding to the type of connection for the configuration you want to add or modify.

- To delete a connection configuration from the list, select it and click on the  button.
- **Random selection:** To use the configurations in the order given by the list, uncheck this . The configurations will be tried in this order, one after the other, until the connection is successful or until all the configurations have been tried. Then click  or  to define the order in which they are to be used.
To use the configurations in a random order, check this. The configurations will be tested in a random order, until the connection is successful or until all the configurations have been tried. This method is useful for avoiding systematic overload of the servers used in the connection configurations at the beginning of the list.
- **Timeout:** This enables you to specify the maximum wait time (in seconds) after which a connection attempt is considered to have failed: the next connection configuration is then used (if there is one).

Note:

This delay is identical for each connection attempt.

APL Mode(3270 emulation)

APL characters are semi-graphical characters that are used on MainFrame servers. Esker's 3270 emulator allows you to use these characters in your emulation sessions. To use APL characters:

- Use the **Alt Gr+F8** (or **Ctrl+Alt+F8** if your keyboard doesn't have the **Alt Gr** key) default keyboard key combination. The **APL** message appears in the OIA bar.
- When in APL mode, use the **Alt Gr+F8** (or **Ctrl+Alt+F8**) key combination to switch back to normal mode.

Note:

The PC keyboard cannot be edited in the APL mode.

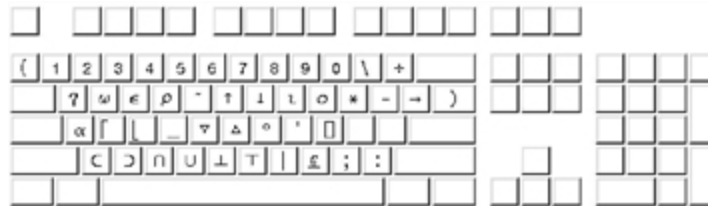
The values returned by the PC keyboard keys in APL mode are displayed below, depending on the combination used (none, combination with the **Shift** key or combination with the **Alt** key). The PC keyboard used for this description is the following type:



APL keyboard (without key combination):



APL keyboard combined with the **Shift** key:



APL keyboard combined with the **Alt** key:



HLLAPI (3270 emulation only)

HLLAPI (High Level Language Application Interface) is a set of functions that makes it possible to program in high level languages such as C, Pascal, Basic and even COBOL. With HLLAPI functions, the programs that you write behave in the same way, as far as the host is concerned, as the user of a 3270 terminal. The HLLAPI interface identifies the functions and the data structures that are used, and carries out the operations defined in the user's program on the remote machine. Use the HLLAPI interface with Esker's 3270 synchronous emulator to transparently access data on a mainframe server using Windows applications you've written in C, C++, or Visual Basic.

Esker's synchronous emulator uses an HLLAPI interface that's compatible with IBM's EHLLAPI and Microsoft's WinHLLAPI: All of the functions defined by these interfaces

are supported. The HLLAPI functions used by the Windows applications you write are contained in the file **whll32.dll** supplied with Tun.

Using HLLAPI

To use the HLLAPI interface, you must use HLLAPI functions when you program your applications and install the function library **whll32.dll** on your PC. A Windows HLLAPI application can call one or more emulation sessions. To identify the different emulation sessions, the program uses the short name attributed to a session. Since the short name is a letter of the alphabet, the maximum number of simultaneous 3270 sessions that can be opened is limited to 26. To define a short name for a 3270 emulation session:

1. Click the TN3270 icon in the **Tun** group to start **Tun 3270**. Under Windows 8, right-click an empty area of the Start screen and click All apps. To start the application, find and click its tile.
2. Choose **File > New Session** from the main menu to open a new emulation session.
3. On the **HLLAPI** tab, click the letter of the alphabet you want to use as a short name.

Note:



The short names currently used by other 3270 emulation sessions running on your PC appear grayed. You can enter a comment describing your session in the **Long Name** field (optional) on this tab. You can change the short and long names after starting the session by right-clicking the emulation screen and choosing **Session Properties**



Accessing data on an IBM MainFrame server from a Windows application (HLLAPI)

The following HLLAPI example shows how to access the data in a 3270 emulation session from a Windows application without displaying the 3270 emulator screen at all. Using this principle, you can completely change an emulation session's interface, making it more user-friendly while maintaining full 3270 functionality. The application in the example uses four main functions:

- **Connect**: Establishes a connection between the emulation session and the application.
- **Disconnect**: Breaks the connection with the emulation session.
- **GetScreen**: Copies the contents of the emulation screen to the application.
- **SendString**: Sends a string of characters to the emulation session screen.

To proceed, **Tun 3270** must be running with an opened session with the short name "A". You can then perform the following operations:

1. Click  to establish the connection between the application and the 3270 emulation session with the short name "A".
2. Click  to send a character string to the emulation session, just as if you entered the string directly in the 3270 emulation screen. Enter the character string in the dialog box that appears.

3. Click  to copy the contents of the emulation session screen to the application window:
You can then use this data.
4. Click  to disconnect the application from the emulation session.

Programming

The five steps described below refer to the code source in the sample application shown at the end of this chapter. The steps show how to declare the **whll32.dll** DLL (step 1) and use its functions (steps two through five).

Note:

The sample source code (in Visual Basic 5.0) was copied to the Tun installation directory when Tun was installed. The code is in **frmMain.frm**, **frmSend.frm**, **frmAbout.frm**, **Module1.bas**, **HLLA-PI.vbp**.

> Declare functions that call the DLL whll32.dll

The DLL routines are stored in files that are external to the Visual Basic application files. Indicate the location of these routines in your application and the arguments they take. Then, supply this information for each routine by declaring the function in the **Declarations** section of the Visual Basic module. Once you've declared the DLL routine, it can be called by any Visual Basic function using a **Call** statement. You must declare the following two DLL routines from whll32.dll:

- **WinHLLAPIStartup()**: Enables the application to specify the required version of Windows HLLAPI and other information on its Windows HLLAPI implementation. This function must be called before using Windows HLLAPI functions. It returns a value indicating if the given version is supported and if the declaration of its HLLAPI implementation was properly made.
- **WinHLLAPI()**: Enables the application to call functions from the DLL, specifying the necessary parameters (they aren't always all used and their type depends on the function).

> HLLAPI syntax:

```
int WinHLLAPIStartup(WORD wVersionRequired, LPWHLLAPIDATA lpData)
```

- *wVersionRequired*: Windows HLLAPI to support
- *lpData*: Structure containing information on the HLLAPI implementation

```
extern VOID FAR PASCAL WinHLLAPI(LPWORD lpwFunction, LPBYTE lpbyString, LPWORD lpwLength, LPWORD lpwReturnCode)
```

- *lpwFunction*: Number of the function to call
- *lpbyString*: String used to pass data (from the application to WinHLLAPI or vice-versa)
- *lpwLength*: Length of the string passed
- *lpwReturnCode*: Return code showing status of the function called

> Defining the connection procedure

This procedure starts the **WinHLLAPIStartup** function, then calls the HLLAPI function number 1: **Connect Presentation Space**. This establishes a connection between a given session on the host machine (presentation space) and the Windows HLLAPI application. The session with which the connection is established is defined by its short name passed as a parameter. In the following example code, the short name passed as a parameter is "A": The connection will be established with the emulation session that is defined by the short name "A". The syntax for this function is:

```
WinHLLAPI (CONNECTPS, lpbyString, lpwLength, lpwReturnnCode)
```

- **CONNECTPS** (*lpwFunction*): 1
- *lpbyString*: Alphabetic character identifying the session (short name) when the function is called
- *lpwLength*: Not used (1 by default)
- *lpwReturnnCode*: Return code indicating connection status

> **Defining the disconnection procedure**

This procedure calls HLLAPI function number 2: **Disconnect Presentation Space**. The function disconnects a session with the host machine. The syntax for this function is as follows:

```
WinHLLAPI (DISCONNECTPS, lpbyString, lpwLength, lpwReturnnCode)
```

- **DISCONNECTPS** (*lpwFunction*): 2
- *lpbyString*: Not used
- *lpwLength*: Not used
- *lpwReturnnCode*: Return code indicating connection status

> **Defining the procedure for retrieving the emulation session screen**

This procedure calls HLLAPI function number 5: **Copy Presentation Space**. This function copies the emulation session's current screen contents to a character string buffer. In the example, this character string is copied to the variable screen, which is then displayed in the application's window. The syntax for this function is:

```
WinHLLAPI (COPYPS, lpbyString, lpwLength, lpwReturnnCode)
```

- **COPYPS** (*lpwFunction*): 5
- *lpbyString*: Character string the emulation session's screen's contents are copied to (when you define this variable, you must assign it at least the number of bytes required by the maximum screen size)
- *lpwLength*: Not used
- *lpwReturnnCode*: Return code indicating the status of the screen copy

> **Defining the procedure for sending data to the emulation session**

This procedure calls HLLAPI function number 3: **Send Key**. Function number 5 sends a key sequence (the maximum number is 255) to the connected emulation session. These key sequences are displayed in a session as if entered by a user. In the following example code, the string StringToSend, entered by the user in an application window, is sent. The syntax for this function is:

```
WinHLLAPI (SENDKEY, lpbyString, lpwLength, lpwReturnnnCode)
```

- *SENDKEY (lpwFunction):* 3
- *lpbyString:* String containing the sequence of keys sent when the function is called
- *lpwLength:* Length of this strength in bytes
- *lpwReturnnnCode:* Return code indicating the status of the string send

Extract from the corresponding Visual Basic code ("Declarations" part of the module):

```
Type WHLLAPIDATA
    wVersion As Integer
    szDescription(128) As Byte
End Type

Public Declare Function WinHLLAPIStartup Lib "WHLL32.DLL" (ByVal Version As Integer, ByRef lpData As WHLLAPIDATA) As Integer
Public Declare Sub WinHLLAPI Lib "WHLL32.DLL" (ByRef lpwFunction As Integer, ByVal lpbyString As String, ByRef lpwLength As Integer, ByRef lpwReturnCode As Integer)
Public fMainForm As frmMain

Sub Main()
    Set fMainForm = New frmMain
    Load fMainForm
End Sub

Sub Connect()
    Dim dat As WHLLAPIDATA
    code = WinHLLAPIStartup(1, dat)
    Call WinHLLAPI(1, "a", 1, code)
End Sub

Sub Disconnect()
    Dim dat As WHLLAPIDATA
    Call WinHLLAPI(2, "", 0, code)
End Sub
```

```
Sub GetScreen()  
    Dim screen As String * 2000  
    Call WinHLLAPI(5, screen, 0, code) 4  
    fMainForm.Text1.Text = screen  
End Sub  
  
Sub SendString(StringToSend As String)  
    Call WinHLLAPI(3, StringToSend, Len(StringToSend), code) 5  
End Sub
```

IBM Printers Emulation

The IBM synchronous emulator enables the printing of IBM terminal data to a printer connected to a Windows PC. With the help of the IBM 3270 or 5250 terminal emulator, you only need to connect to the terminal via an emulation session on your PC. Data coming from the terminal are then displayed on the emulation screen, and can be printed using the emulator's print command.

You can also print data from the terminal, emulating an IBM printer on your PC. Instead of being displayed on an emulation screen of the PC, the data will be printed directly on a printer connected to the PC. You will then be able to print data from your IBM applications to any printer accessible from your PC, just like with a real IBM printer.

Esker provides IBM 3287 and 3812 printer emulation on your PC:

- **3287** emulation prints data flows from IBM MainFrame servers. Opening an emulation session identified by a type 1 or type 3 LU on a MainFrame server carries out 3287 printer emulation. The 3287 emulation can use the following types of connection: TN3270E connection, connection through an NT SNA Server, or Netware for SAA gateway.
- **3812** emulation to print data flows from IBM AS/400 systems. Opening an emulation session identified by a type 1 LU on an AS/400 system carries out 3812 printer emulation. The 3812 emulation can use the following types of connection: TN5250E connection, connection through an NT SNA Server, or Netware for SAA gateway.

Logical Unit (LU)

The IBM SNA architecture is based on a connection oriented, hierarchical and centralized model. In this model, an LU (Logical Unit) designates a terminal or a printer. There are different types of LU:

- LU1 and LU3 designate printers. LU3 is currently the most used LU type for IBM printing. It supports DSC mode data flows. LU1, less frequently used, supports the SCS mode, which allows to send page layout commands such as tabs, margins, page breaks, bold, italic, etc from the server.
- LU2 designates 3270 terminals.

Using IBM printer emulation via Esker Viewer

To print from a terminal emulation session associated to a printer emulation session:

1. Run **Esker Viewer** and connect to your terminal from a 3270 or 5250 emulation session.
2. Select **File > New** and specify the session's connection settings

3. Select **Session > Terminal** from the printer emulation session window. Then, configure the 3287 or 3812 printer emulation session on your server and establish the connection.
4. From the printer emulation session window, select **Session > Print**. Configure the 3287 or 3812 printing options used by your server to send data to be printed, and then configure the printer connected to your PC to print data from the server.

Using IBM printer emulation via the print server

To allow data printing from your IBM system without connecting to it from an Esker Viewer emulation session:

1. From the **Start** menu, select **Esker Tun > Application Access > 3287 Printer / 3812 Printer**. Under Windows 8, right-click an empty area of the Start screen and click All apps. To start the application, find and click its tile.
2. When you start the print server, it appears as an icon in the task bar. Click on this icon to open the print server window.
3. Select **File > New** and specify the session connection settings.
4. From the printer emulation session window, select **Session > Terminal**.
5. From the printer emulation session window, select **Session > Print**.

IBM Print Server

Esker's print server allows you to configure, and thus centralize, several 3287 and/or 3812 emulation sessions from the PC print server. At the server startup, these printer emulation sessions are connected, which allows you to print data from your IBM servers to the PC printers. The print server also provides a printing follow-up. Starting the print server

1. From the **Start** menu, select **Esker Tun > Application Access > 3287 Printer / 3812 Printer**. Under Windows 8, right-click an empty area of the Start screen and click All apps. To start the application, find and click its tile.
2. Click on the print server icon in the task bar.

> Stopping the print server

Select **File > Exit** from the print server window.

Print server administration

To use the print server on your PC, configure a 3287 or 3812 printer emulation session for each IBM terminal from which print jobs will be launched to the PC. Use the print server window to administrate these sessions.

1. Click the print server icon in the task bar (🖨️). The print server interface is the same as Esker Viewer's, except:
 - The only two possible connection types are the 3287 and 3812 emulations.

- A window on the left displays all of the configured printer emulation sessions in a tree list.
2. Select **File > New** and specify the session connection settings.
 3. For each session, configure print settings: print options, and the PC printer used.
 4. Select **File > Save Workspace**.

The specified workspace will be started at each print server startup, which will enable the using of printers emulation sessions to print data from IBM terminals to the PC's printers. The workspace now contains several windows. Each window corresponds to a printer emulation session and provides you with information on this session's status. A specific window displays all of the sessions in a tree list, with:

- Both type and status of the connection set between the PC and the IBM server.
- The printer used on the PC to print data flows from the IBM server.

> **Removing a session**

Close a session's window and save the workspace.

> **Connecting/Disconnecting a session**

Select a session's window, and then select **Session > Connection > Connection** or **Disconnection**.

> **Editing an emulation session's settings**

Select a session's window, and then select **Session > Connection > Configuration**.

IBM printer emulation connection

Esker's IBM 3287/3812 emulator enables you to make the following types of connection:

- Direct connection to a MainFrame/AS/400 server or via a SNA-TCP/IP UNIX gateway using the TN3270E/TN5250E protocol.
- Connection to a MainFrame/AS/400 server via a Microsoft NT SNA Server or Novell Netware for SAA gateway.

When you select an IBM print type session (see **Using Esker Viewer** in the **Esker Viewer** chapter), the connection dialog box appears. This box contains two tabs:



- The **Session** tab enables you to specify the type and the parameters of the connection to be made.
- The **State** tab gives the state of the connection (the state is **Unconnected** at the moment of connection).

Options

- **Close Session on Confirmation:** Check this to display a confirmation dialog box on exiting the program.

- **Startup Connection:** If checked, the program will automatically reconnect to the host at startup.
- **Reconnection after a shutdown:** If checked, the program will automatically re-establish the connection if it's interrupted by the host.
- **Terminal Type:** This field sets the type of terminal display to use.
- **Host name/System name:** Enter the name of the server or its IP address. You can also select a server from the drop-down list (only enter a name if you have a names server).
- **Port:** The default port number is 23. You can change this number, if necessary.
- **Associated printer (3287):** When you define a 3287 emulation session, you must state the printer logical unit (LU) that you want to use on the host machine. Some terminal LUs are associated with printer LUs (LU1, LU3) on the host machine. Check to associate a terminal LU (LU2) with a printer LU rather than simply enter a printer LU.
- **Printer logical unit (3287):** You can enter the name of the printer LU (LU1, LU3) that'll be used by the host machine for 3287 print emulation. The name of the printer LU must correspond to the type of terminal defined on the host machine, namely Model 1.
- **Terminal logical unit (3287):** This field only appears if **Associated printer** is checked. It lets you enter an LU terminal name (LU2) that's associated with the name of the printer LU on the host machine that will be used to emulate 3287 printing. The name of the terminal LU associated with a printer LU must correspond to the type of terminal defined on the host machine, namely a Model 1 printer.
- **Unit name (3812):** Optional field. If you enter nothing in the field, the AS/400 system will attribute a default name to your client PC to uniquely identify it. You can, however, enter a terminal name to identify your machine.

Click **Advanced** to customize the advanced properties of the 3812 session.

- **Identification:** To enable automatic login, fill the required parameters when login to the AS/400 system: user name and password.
- **Encrypt:** Check to encrypt the password during login.
- **Queue Name:** Specify the messages queue on the AS/400 (QSYSOPR by default).
- **Library Name:** Specify the library on the AS/400 (*LIBL by default).
- **Font:** Indicate the IBM code of the font to be used. Refer to your IBM documentation for more details.
- **Terminal parameters:** You can modify the default configuration of the AS/400 terminal on which the connection is established. Specify a new **Code Page** and/or **Charset** to use if necessary. Click **Default** to reset these values to your emulation configuration defaults (given by the character set).
The **UserVar Text** and **UserVar Binary** lists let you configure other advanced parameters of the terminal. To add an additional IBM parameter (text or binary), click . Then, enter the name of the IBM parameter in the upper left part of the field, and its associated value in the right part. To remove a character from the list, select it and click .

- **Mode name:** By default, the mode name is QPCSUPP. This mode is specific to the gateway and APPC library. Typically, you don't need to change this value. Contact your network administrator if you want to change it.
- **Local LU name (NT SNA Server):** For connections through an NT SNA Server gateway, enter the name of the local LU as it's registered on the gateway. This field must be completed.
This field isn't shown for Netware for SAA gateway connections. The name of the local logical unit must be correctly registered on the gateway for the connection to succeed.
- **User name/Password:** Enter the user name and associated password that allow you to connect to the AS/400 system. The user name and password are defined on the gateway.

3287 or 3812 Print Configuration

These print options allow you to specify, for each printer emulation session, both mode and print format used by your MainFrame or AS/400 system server to send data to be printed to the PC.

> 3287/3812 print options

- Select **Terminal** in the printer emulation session window.
- In **Automatic print options**, select the option for the automatic print mode that will be used by the server:
 - If the server sends an instruction indicating the end of the print job (EOJ), select **Print on reception of "End Of Job"**. When this instruction is received, the data is printed automatically.
 - If the server doesn't send this instruction, select **Print after timeout**. Once the print job is finished, the data will be automatically printed after the time in seconds specified in the **Timeout** field. If you don't specify a time in this field, the data will not be printed automatically. You must then start the print job manually.
- The **Rows** section lets you define the width of the pages sent to print. Select the **Automatic** option to use the page format returned by the server: Alternatively, select **User** to specify a given page width or the number of characters printed per line.
- The **Columns** section lets you define the height of the pages sent to print. Select the **Automatic** option to use the page format returned by the server: Alternatively, select **User** to specify a given page height or the number of characters printed per column.

PC Print Configuration

For each printer emulation session, it is necessary to configure the PC print mode and the printer used (if any) to print data coming from the IBM terminal. Select > **Print** in the printer emulation session window. The PC print configuration dialog box appears.

- **Use this specific printer:** Select to print the received data on a printer connected to your PC. Then, select the printer of your choice from the drop-down list. As in any other Windows application, the locally defined printers are available on it.
- **Setup printer:** Choose to configure the printer on which you want to print.

- **Print in a text file:** Instead of directly sending print data received from the server, you can record the corresponding documents in one or more text file(s).
- To record the received documents one after the other in a single text file, select **Print to a text file** and uncheck **Multiple Files**. Specify the path of the text file to be used or click **Browse** to select it. To record documents in distinct text files, select **Print in a text file** and **Multiple Files**. Specify the path where the text files will be saved or click **Browse** to select it (the various file names will be created automatically from the directory name).
- **Orientation:** Specify the print orientation for print jobs from the printer emulation session.
- **Non-graphic print:** You can define the font used to print in text mode. You can set the font you choose as the default print font for all printings (including that requested from **File > Print**) by checking **Use selected font**. Click **Font Setup** to specify the font to use and its style and size. Select **Use size of selected font** to set the font size as the default print size (if you don't select this check box, the next nearest size is used for printing).

Note:

The **Script** drop-down list box in the font definition dialog box lets you choose the character set encoding used by the font.

3287 or 3812 Emulation Status

You can query the status of the print jobs sent from the IBM mainframe server or the AS/400 at any time from the printer emulation session window. This window gives you the connection and print status, and lists the following events:

- Connection and disconnection of the session.
- Print jobs.
- Communication with the server or printing errors.

The upper-end of the window will provide you with information on the 3287 emulation status:



If no 3287 or 3812 emulation session is active on the PC, the 3287 or 3812 emulator status is **Idle**.



If a 3287 emulation session is running on the PC and a connection is established with a host machine (IBM mainframe server), the 3287 emulator is **Active**.



If the printer is active (in the case of 3287) and an application on the IBM mainframe or the AS/400 system establishes a connection with the printer (via its LU), the 3287 or 3812 emulator switches to **Session**. It's then ready to receive and print data from the IBM machine application.



If a print job is being processed, the 3287 or 3812 emulator status is **Printing**.



If the print job is temporarily stopped with the **Suspend** command, the 3287 or 3812 emulator status is **Suspended**.

Print Commands

You can suspend and resume data sending; cancel printing or force the printing of received data.

- **Suspend/Resume printing:** Activate the print session window, and select **Session > Commands > Suspend printing** or **Resume printing**. The 3287 or 3812 print session status is then changed to **<Inactive>**.
- **Cancel printing:** When the print session status is **<Inactive>**, select **Session > Commands > Cancel printing**. None of the received data is printed and the print job is considered as finished.
- **Flush printing:** When the print session status is **<Inactive>**, select **Session > Commands > Flush printing**. The emulator will print the data already received. You must restart the print job for data to be received.
- **Log file:** You can record the information contained in the print emulation session window in a log file to consult later.
Select **Session > Log**. The Log configuration dialog box appears.
Select **Use a log file** to save the contents of the emulation session window in a log file (this option is not selected by default).
Enter both path and name of the file to be used (.LOG), or click **Browse** to select a file.

Using Emulators

You can configure the number of lines and columns in your application, add scroll bars, choose the type of line scrolling and cursor coupling, and center the terminal in the emulator window or not. You can also have a bitmap image of your emulation session as the screen background.

Configuring the screen

Select **Session > Screen**.

Asynchronous emulator

The dialog contains two tabs. The first tab configures of the emulation screen (dimensions, scrolling, etc.), and the second configures the display of the bitmap image on the screen background.

- **Screen size:** The default setting for Esker's asynchronous emulator emulates 80 x 25 screens. You can change this setting in the **Lines Used** and **Columns Used** fields. This is useful for emulating terminals that use different dimensions (for example, 132 columns or 43 lines).
Typically, the asynchronous emulator simultaneously stores only 25 lines. You can change this value in the **Lines Stored** field. The maximum value for this field is 1024. Use the scroll bars on the right of the emulation screen to view the lines stored by the asynchronous emulator that haven't been displayed yet.
- **Scroll type:** Choose a type of scrolling when displaying the scrollbars:
Jump: enter the number of lines scrolled per jump.
Smooth: adjust the scroll speed as you wish.
- **Scroll bars:** The **Vertical** and **Horizontal** check boxes toggle the display of horizontal and vertical scroll bars along the side and bottom of the emulation screen. The scroll bars are useful if you've opted for more than 25-line storage and you're not using **Dynamic Sizing**.
- **Cursor coupling:** The **Horizontal** and **Vertical** check boxes let you enable or disable cursor coupling in either direction. Cursor coupling scrolls the screen so that the cursor is always visible. If either is unchecked, you can move the cursor outside the range of the terminal window.
- **Center terminal:** Check to center the terminal window into the emulation window.
- **Frame terminal:** While the terminal screen is centered in the emulation window, you can put a frame round it. To do that, check this.
- **File:** This field should hold the name of the bitmap file (.bmp) with the required image.

- **Alignment:** The vertical and horizontal alignment options give the position of the image in the emulation window. These options are only of use when the image is not adjusted to fit the display zone.
- **Tile:** If the image is smaller than the emulation screen, it can be duplicated until it fits all the available space by selecting this check box.
- **Scroll with text:** If the bitmap is tiled and the display zone is not the window, you can scroll the bitmap with the text if you check this.
- **Fit to display zone** If the bitmap doesn't coincide exactly with the size of the emulation screen, you can resize it until it occupies all the available space by selecting this check box.
- **Display zone:** The image display zone can be:
 - **Total memory:** the image location is relative to the zone bounded by all the lines memorized.
 - **Terminal:** the image is displayed in the zone occupied by the emulated terminal lines.
 - **Window:** the image location is relative to the window, independently of the number of lines displayed or memorized.

Synchronous emulator

- **Image:** Use the Browse button to look for a bitmap image to display as the screen background, or enter the file's absolute path.
- **None:** no image displayed. The background stays as-is.
- **Tile:** the image is tiled over the background.
- **Center:** the image is centered in the middle of the screen.
- **Fit to Window:** the image is adjusted to fill the entire screen.
- **Size:** Adjust the terminal size using the following options:
 - **Adjust to font size:** the terminal size is adjusted to fit the screen displayed using the font defined in Session > Font.
 - **Adjust to window size:** the terminal size is adjusted to fit the entire emulation window. If you reduce the emulation window, the font will be reduced.
 - **Adjust to window width:** the terminal width is adjusted to fit the emulation window width. The terminal height depends on the font used.
 - **Adjust to Window height:** the terminal height is adjusted to fit the emulation window height. The terminal width depends on the font used.
- **Display a frame around terminal:** You can add a frame around the terminal by selecting the Display a frame around terminal check box.

Choosing the font

You can choose a font different from that displayed on the screen by default. You can also alter the size. Any non-proportional character font available under Windows can be used in emulation. To use all the semi-graphic characters correctly, the font needs to be OEM, and not ANSI.

Note:

The **SystemPC** font is supplied by Esker for the best possible use of emulation with Windows. It's a fixed OEM font ranging in size from 4 to 30 that was developed especially for the asynchronous emulator. Use **Sys132PC** for emulation with 132 columns.

Select **Session > Font**.

- **Font used:** You can define the type and size of font that's used by the emulator. (Asynchronous) Use **80 columns** and **132 columns** to select the character font, depending on the width of the emulated terminal.
- **Dynamic Sizing:** This function causes the emulator to change the size of the font in keeping with the size of the emulation window so that a full terminal screen is always displayed (80 x 25 or 132 x 25, depending on the terminal). Dynamic sizing is best used with character fonts that are available in multiple sizes.

Use ANSI to OEM conversion (asynchrone only): Apply a conversion of the text entered using the keyboard from the local code page of the client machine to the code page used in Tun.


•

Customizing colors

By default, the characters appear on the PC screen as they are displayed on the terminal. You can change the displayed attributes (**Normal**, **Reverse**, **Underline**, etc.) by giving them colors and styles of your choice (bold, italic, underline). Select **Session > Colors**.

Capturing attributes with the mouse

You can select an attribute directly by capturing it on the screen. To do this:

1. Click  in the **Attributes section**. The color configuration box disappears.
2. Use the mouse cross-hair cursor to click on the character for which you want to redefine the attribute. The color configuration box then reappears. The attribute(s) corresponding to the captured character are selected.

Note:

If you opened the configuration box from the context menu of a character displayed on the screen (**Edit attributes**), the attribute of the character concerned is selected when the configuration box is opened.

Asynchronous emulation

- **Selecting attributes:** To modify the attributes, select Attributes. You can combine several attributes. By default, Normal is selected (no box is checked). To define another attribute or combination of attributes, check them.
- **Selecting colors:** Some applications return characters in a particular color or on a particular background. To define these characters, select **Colors**, and then choose the character or background color for the application from the drop-down list.

Changing colors and styles

After selecting an attribute, a combination of attributes, or a character or background color, you can then apply the colors and styles you wish to this element.

In the **Colors** tab, select foreground and background colors. Depending on the type of attribute, a check box appears:

- **Pre-defined colors:** Check to use a pre-defined colors for the attribute selected (option for all the attributes except Reverse video and Underline).
- **Basic color effect:** Reverse video and Underline attributes can be based on **Normal**. Check this to do it, uncheck it to customize these attributes independently.

Attribute combinations that aren't associated with particular colors adopt the color of a sub-combination (for example, if the combination Blink + Dim + Reverse isn't associated with a color, it uses the color associated with Blink + Dim).


In the **Styles** sub-tab, select the text style (**Italic**, **Bold**, **Underline**) and the background character style.

Synchronous emulation

- **Selecting attributes:** Select a standard attribute or color from the drop-down list(s).
- **Altering the color:** In the **Colors** tab, select foreground and background colors. For 3270 emulation, click **Uniform Background Color** to assign the selected color to all the attributes. For 5250 emulation, click **Apply Normal Attribute Background Color to All Attributes** so that all the attributes have the same background color as the normal attribute.
- **Displaying Attribute characters:** In a string of characters there are special characters, represented on the screen by a blank, that contain information on the characters that follow (for example, on the format of a character string). Normally, these characters are the same color as the background and therefore don't appear on the screen. In both 3270 and 5250 emulation, making them a different color can make these characters visible. To do this, assign a color to **Attribute Characters**, and select **Display Attribute Characters** to apply that color to these characters.
- **Changing styles:** In the **Styles** tab, select the font style (**Italic**, **Bold**, **Underlined**, **Blink**) and the style of the characters and borders.

Macros

You can associate a macro to the beginning or to the end of the emulation session.

1. Select **Session > Macro**.
2. Select a macro file:
 - .MAC extension files contain macros written in EScript. Enter the macro file name in the right field. Click  to browse for the file.
 - .JS (JavaScript) or .VBS (VBScript) extension file containing a set of functions listed on the right. Select a function to use as the session startup or end macro in the dropdown list.
3. Add any other required parameters into **Parameters**, separating each occurrence with a space. If a parameter contains a space, you must specify this between double quotes (" ").

Screen printing

Select **File > Print**, and then the print mode:

- Text mode. If you print in text mode from the asynchronous emulator, you can choose the font used for printing. To do that, check **Use the selected font** from the dialog and click **Font** to choose the font.
- Graphic mode. You can print your emulation screens using a white background to save ink. Check **Force black printing on white background**.
- With template (3270 or 5250).

Configuring a printer

Select **Session > Printing**.

- Select a printer. Click **Specific printer** to configure it.
- (Asynchronous) In certain emulations that include transparent printing, you may set a wait time before closing the print spool. This prevents the connection from being closed after each data flow. Enter a value (in ms) for the wait time in the **Print spool timeout (ms)** field.
- (Asynchronous) Check **Manage transparent printing** to configure transparent printing:
(Asynchronous) If necessary, check **Specify the number of lines** and enter the number of lines you want to print on a page. Do the same with **Specify the number of columns**.
- Set the default font for all printing by checking **Use selected font**. Click **Font Setup** to specify the font, style, and size. Check **Use selected font size** to set the font size as the default print size (if you don't check this, the next nearest size is used for printing).
- (Synchronous) Select the printing mode: **Print Screen in Text Mode** (faster) or **Print Screen in Graphic Mode**.


- Use **Script** drop-down list in the font definition dialog to choose the character set encoding used by the font.

Printing with template (3270/5250)

You will frequently want to print several or all of the pages, during a synchronous emulation session. When there are many pages, this job quickly becomes tedious and time-consuming. To avoid this, the IBM 3270 and 5250 emulators from Esker offer printing with a template, the principle of which is to create a reusable print module. A print template contains:

- The screen area to be printed
- Initial and final texts markers for printing
- Page scroll keys

Select **Session > Print template**. The template configuration box appears.

- **Load**: Click to open an existing print template. Print template files are text files with the PTP extension.
- **Comment**: Enter an optional comment.
- **Print area**: Enter the coordinates for the screen area to be printed in the order "top left" and "bottom right". Click the screen at the top left-hand corner of the print area and note the cursor coordinates on the right-hand side of the status bar. Click the bottom right-hand corner of the print area and note the coordinates. Enter them in the dialog.
- The coordinates can also be entered by clicking , dragging the cross hairs to a position on the emulation screen, and releasing the mouse button. The dialog disappears while you position the cursor. It reappears once you've chosen the position.
- **Full Screen**: Click this to apply the coordinates of the full screen.
- **Previous page key / Next page key**: Enter the Previous Page Key and the Next Page Key in the appropriate fields. These are system-dependent and are often shown at the bottom of the emulation screen where the commands are displayed. If they're not shown at the bottom of the emulation screen page, you should consult the documentation for the type of host you're connecting to. The codes for these keys are sent to the host to scroll from one page to another. If these aren't entered, only the current page is printed. Note that the system might require that an alphabetic key be pressed with the **Enter** key. In this case, use the extra field provided for this purpose (the first field).
- **Return to initial page after printing**: If selected, the program returns to the initial page when printing is completed.
- **Initial text / Final text**: Enter the text to delineate as the first and last pages to be printed. In the respective **Search Area** boxes, enter the coordinates of the screen zone where the emulator is to search for the initial and final character strings.

Printing with a template

Select **File > Print > With template**. The **Print with Template** dialog has the look and behavior of a normal Windows print dialog box with one or two differences:

- Instead of a print selection option, there is a **Current Page to End** option, which prints from the current page to the **Finishing Text** indicated by the template. This can be the end of the file or an intermediary point.
- You can choose a template by using **Browse** and choosing a template file (.PTP).

The printout consists of the emulation screen pages without any unnecessary emulation screen information from outside the print area.

Transparent printing (asynchronous emulation)

In the character flow sent to the terminal emulator, the server can insert escape sequences to inform the emulator that the following characters are to be:

- Displayed on the screen (conventional terminal emulation)
- Sent directly to the printer connected to the PC emulating the terminal

This lets the centralized application access the printer without using a buffer or print server. Esker supplies a series of actions used to reroute the characters received from the server either to the PC screen, or to the printer, or to both simultaneously. To work in transparent print mode, the emulator must have escape sequences associated with these actions in the escape sequences configuration file.

Cut & Paste

The Esker emulators allow the use of the clipboard for copying, cutting and pasting characters. This function can be of use both in the emulator (for example to send the copied text to the communication channel) and outside the emulator (for example to copy a part of the screen into a word processor).

- **Copying text to the clipboard**: Select a text area, then select **Edit > Copy**.
- **Pasting the clipboard contents**: Position the cursor and select **Edit > Paste**. In asynchronous emulation (UNIX), to send the clipboard contents directly to the communication channel, without having to hit **Enter**, select **Paste and send**.
- **Pasting the selection** (asynchronous emulation): Select a text area and then select **Edit > Send selection**.
- **Clear selection** (IBM synchronous emulation): Select a text area and then select **Edit > Clear**.

Copy options (asynchronous emulation)

You can set the copy options in asynchronous emulation (UNIX). To do this, select **Edit > Copy options**. The following copy options are available:

- **Rectangular Selection:** if checked, the selected area is rectangular; otherwise the lines between the starting and end points are completely selected.
- **LF at End of line/CR at End of Line:** these options stipulate whether or not LF or CR characters are to be inserted at the end of the lines.
- **Wait time:** to prevent overloading the communication channel when cutting/pasting a large amount of text, the Wait time field enables the operation to be delayed by defining the time between copying of two blocks of 128 characters from the clipboard to the emulation window.

File transfer

File transfer in terminal emulation enables files to be exchanged between the PC and the server. This can be particularly useful, for example in order to use the content of a file in a word processor or share the files with other users. Files can be transferred between a PC and a server via FTP (in this case, you must have an FTP client module on the PC), or from an emulation session (available with the Esker asynchronous and IBM 3270 emulators).

- **Configuring file transfer:** Select **Transfer > Configuration**.
- **Sending a file:** Select **Transfer > Send**.
- **Receiving a file:** Select **Transfer > Receive**.
- **Canceling a transfer:** Select **Transfer > Cancel**.
- **Transferring several files (IBM 3270):** Select **Transfer > Batch**.

Asynchronous emulation

The file transfer protocols available in asynchronous emulation are:

- **ASCII:** This is the most basic type of file transfer. Receiving consists of capturing the characters sent across a connection in a file. Sending consists of emptying the contents of a PC file onto the network connection, but doesn't provide any means of controlling reception. The host is responsible for capturing the data into a file. The command most frequently used on hosts for this is:

```
stty -echo ; cat >/tmp/fichier; stty echo
```

- **Rtunplus:** Esker's proprietary file transfer protocol, RTUNPLUS, has the advantage of being simple to use, and freely installable on any UNIX host. RTUNPLUS is provided in UNIX executable format for SCO UNIX and XENIX, and the source code (rtunplus.c) can be compiled on other hosts.
- **X, Y and Z-Modem:** These protocols are those normally installed on most BBSs (Bulletin Board Systems). They are general-purpose file transfer protocols for use with telephone links. The server programs for these protocols are usually not delivered as standard on most UNIX systems, but are generally available in the public domain as executables or source files for compilation.

Select a protocol from the **Protocol** drop-down list.

The file transmission box is as follows. The file reception box is similar to the transmission box, apart from a few differences concerning the transfer direction: for example, the **Local source file** field becomes **Host source file**

- **Local:** Enter the source file name.
- **Host Target file:** Enter the target file name.
- **Text file: convert:** This option converts the contents of the file from the local character set to the character set of the remote machine. The **Conversion** button displays a dialog box that lets you select the two character sets.
- **CR/LF -> LF:** This option applies or cancels CR/LF to LF (or LF to CR/LF) conversions. It's useful when transferring text files from a PC to a UNIX server or vice-versa.
- **Use Protocol:** Specify one of the available transfer protocols.
- **Options:** Click to select a protocol and set related parameters.
- **If ... File Exists:** This option determines what action to take if you transfer a file that already exists on the destination machine.
- **Send:** Click this button to begin the transfer procedure. If you're using RTUNPLUS, then the UNIX host must be in the "shell" (with the #, %, or \$ prompt on the command line) to run the server portion of the protocol. To use X, Y and Z-modem, you may need to manually start the server on the UNIX host (that is, with a command such as **xmodem -r /tmp/tmp.file**).

3270 synchronous emulation

File transfer in 3270 synchronous emulation uses the IBM IND\$FILE protocol. The protocol configuration box is as follows:

- **Host System:** Defines the type of host operating system you're connecting to.
- **Timeout:** Indicates the length of time the program must wait for a host response to the file transfer request. The default is 30 seconds.
- **Packet Size:** The default packet size is 2040 bytes. The speed of the file transfer is directly proportional to the size of the packet.
- **Host Command:** Indicate the command that is to handle the transfer. The default is IND\$FILE.
- **Initial Action:** Defines the initial action to be performed before the transfer starts. For example, you can clear the terminal screen, if the **Clear** key is set, or send the **Enter** or **Home** key code first.
- **ASCII/EBCDIC Conversion:** Choose the appropriate option for the file transfer you're preparing.
Host Performed Conversion means characters are converted to the host code page by the host.
Use Current Character Set converts received characters to the character set loaded.

The PC performs this action.

You can select a different character set by browsing the .SET files.

- **Server command:** Enter the command sent to the server to retrieve the file list when performing a transfer. By default, the **FILELIST** command is used for a CMS system and **LISTCAT** for a TSO system.
- **Initial action:** If necessary, select the initial actions that precede the command.


The file transmission box is as follows. The file reception box is similar to the transmission box, apart from the configuration of the save format. You can switch from one to the other by selecting the **Send** or **Receive** options in these boxes. This is particularly useful when transferring a number of files.


- **Remote file:** Enter the name of the file on the server.
- **Local file:** Enter the name of the local file (absolute path). You can use **Browse** to search for the files.
- **System:** Select the server system (**CMS**, **TSO**, **CICS**).
- **Conversion:** Set the conversion type: Check **ASCII/EBCDIC** to perform the conversion to EBCDIC, **CR/LF** to replace CR/LF codes with end of record codes, and **JIS-CII** to perform the conversion from Shift JIS to EBCDIC (check **NOSO** (SO/SI code) then if the file only contains DBCS characters).
- **Mode:** Select the copy mode (replace existing file or add to this file).
- **Record format:** For a file transfer to a TSO or CMS server, select the options as needed.
The default record format means that the record length complies with the server's default values. To avoid this, check **Variable** or **Fixed** for CMS and TSO environments. You can also use **Undefined** for the TSO environment.
For TSO environments, select the unit: **Default** to use the server's default unit, otherwise **Cylinders**, **Tracks** and **Blocks**. Enter the block size value, and the track and cylinder values if you select these units as space media.
- **Additional options:** This field lets you add parameters for the IND\$FILE program that haven't been entered elsewhere.



Multiple file transfer (synchronous emulation)

Multiple file transfer enables you to specify the list of files to be exchanged between the PC and the server once you start the transfer operation. When transferring multiple files, a dialog notes the status of each transfer. You can save the contents of this dialog box in a log file, and then view it later.

Select **Transfer > Batch**.

- **Transfer List:** Each file on this list corresponds to a file to be sent or received, with its transfer parameters.
To add a file to the list for transfer, click . Then, complete the file transfer parameters.
To change the transfer parameters of a file, double-click on it.

To delete a file from the list, select the file and click .

To define the order of transfer for files, select the file and click  or .

- **Time to wait:** Enables you to define a waiting period between transfer of two files on the list.
- **Transfer:** Click to start the multiple transfer operation.
- **Save:** Click to save the list of files to be transferred. Then, name the file and path (.XFR), and click **OK**.
- **Load:** Reload this list of files later by clicking this and selecting the transfer file (.XFR).

Working with a personal function-key panel

In the emulation window, you can display a key panel enabling you to access the main functions of your terminal with a mouse click.

> Opening a key panel

Select **Session > Function key panel**.

> Show or hide function-key panel associated with the session

Select or deselect **Display > Key panel**.

> Start the function-key panel editor to create or modify a key panel

Select **Tools > Key panels editor**.

Asynchronous emulation

- **File:** Select the function-key panel containing the function-key panel definition. Click **Browse** if the function-key panel does not appear in the default list proposed and look for the function-key panel of your choice. The function-key panel files are text files with extension .PAN.
- **Docking:** The function-key panel can be transformed into a toolbar. To do this, select the docking option you want to apply to the function-key panel.

Note:

If the width or the height of the function-key panel is greater than 1/8th of the width or the height respectively of the screen, it can't be docked.

- **Default:** The type of docking used is that defined when the function-key panel was created, in the panel parameters.
- **Disabled:** The docking of the function-key panel is not possible.
- **As toolbar:** The function-key panel becomes a standard toolbar that you can dock on the window border.
- **Wrap Toolbar:** If there are too many keys on the panel, they're arranged in more than one line or column.

- **Dock as Panel:** The function-key panel becomes a toolbar, but the layout of the keys remains unchanged.
- **Show function-key panel:** By default, the function-key panel is not displayed in the emulation window. You can select the display option from the panel configuration box.

Synchronous emulation

Select the function-key panel that contains the function-key panel definition. Click **Browse** if the function-key panel does not appear in the proposed default list and look for the function-key panel of your choice. The function-key panel files are text files with extension .PAN.

- **Docking:** The function-key panel can be transformed into a toolbar. To do this, select the docking option you want to apply to the function-key panel: docking as a toolbar or as a function-key panel.
- **Show panel:** By default, the function-key panel is not displayed in the emulation window. You can select the display option from the panel configuration box.

Customizing the connection

You can view or modify the connection parameters of a session, connect according to these parameters, or disconnect and, in the case of asynchronous emulation, send a command on the communication channel.

- **Viewing or modifying the connection parameters:** Select **Session > Connection > Configuration**. The dialog that appears is identical to the connection box when you start an emulation session using **File > New**.
- **Connecting:** Select **Session > Connection** to connect. **Disconnecting:** Select **Session > Connection > Disconnection** to disconnect the current session.
- **Sending a particular command (asynchronous emulation):** Select **Session > Connection > Commands**. Then, select a command.

Customizing the terminal

A terminal has its own configuration parameters (for example, cursor size, special character sets, etc.). You can change these parameters. Select **Session > Terminal**.

Asynchronous emulation

When you choose a type of terminal at connection, the existing terminals have a corresponding configuration file associated with the session. This file contains the terminal configuration, the main functions of which you can then modify.

Note:

The terminal configuration files have .SES extensions. The parameters that can be modified here are those declared in the **[Intro]** section of these files.

Synchronous Emulation

- **Print Key:** In 3270 emulation, the Print key option can be ignored or used to print according to the local print mode. In 5250 emulation, the Print key option can be used as it normally is on a 5250 terminal, or activated according to the local copy option.
- **Ctrl key:** Select either or both of these options to allow the redefinition of this key. Redefine the **Ctrl key** by choosing **Session > Keyboard**.
- **Caps Lock Key Definition** determines whether the Caps Lock key is allowed to affect the whole keyboard or simply the main alphanumeric keyboard, excluding the numerical keypad. In this way, the symbolic keys on the keypad, **Up**, **Down**, **Home**, etc., are freed for other purposes.
- **Type ahead:** You can also choose to use the keyboard in advance, by defining the size of the buffer used for the keyboard.
- **Beep for Operator Error:** Check to have the PC beep when a user makes an error.
- **Cursor style:** Define Cursor style: Underscore, Block or Vertical Bar, Solid or Blink.
- **Show Rule Cursor:** If selected, the line at which the cursor is positioned is underlined.
- **Numeric Fields Support** (3270 emulation): Select to restrict user input to numeric characters. If unchecked, any alphabetical character can be entered in numeric fields.
- **Moving the cursor at first mouse click:** When the Esker Viewer window is inactive, right-clicking the Esker Viewer window can generate two different situations:
The Esker Viewer window becomes active and the cursor moves to the mouse pointer's location. Check **Move the cursor at first mouse click**.
The Esker Viewer window becomes active, but the cursor remains at its previous position.

Modifying the character table (IBM synchronous emulations)

Synchronous emulation uses the EBCDIC extended character set, which can vary from country to country. Select **Session > Character table**.

The current EBCDIC table is shown on the right. It presents the characters displayed on the screen. The character set available on the PC is on the left. Choose between the ANSI and OEM character sets. You can only modify the EBCDIC table. You can:

- Load the table corresponding to your language by clicking **Load** and selecting the .set file of your choice.
- Select from the left-hand table (PC characters) a special character to display in place of another. Then, drag it into the right-hand table over the character it is to replace.
- Click **Save** to save the modified character table.

Customizing the keyboard

Esker supplies its emulators with a default keyboard configuration. You can change this configuration by customizing each key so that a different result is obtained when struck. Depending on the type of emulation, you can associate the following with a PC keyboard key:

- The value of a terminal keyboard key.
- A character or character string.
- A function key.
- A function related to the type of terminal.
- A macro or a script.

Select **Session > Keyboard**. The configuration box shows two keyboards:

- The PC keyboard at the top.
- The emulated terminal keyboard at the bottom.

Asynchronous emulation

- **PC Keyboard/Terminal Keyboard** : Each type of emulated terminal has a corresponding default keyboard configuration. This default configuration is in fact an association between a PC keyboard and a terminal keyboard.
You can select another type of keyboard for the PC and for the emulated terminal. Select a keyboard type from either of the drop-down lists in the configuration box.
The PC keyboard configuration files have the .KBM extension. The terminal keyboard configuration files have the .KBT extension. Only .KBM and .KBT files present in the emulator's installation directory will be available in the drop-down lists.
- **PC keyboard**: Each of the keys on the PC's keyboard has an associated scan code. When you press a key you send the code over the communication channel. For example, when you press the key "e", you send the code 18. The keys configuration file, associated with the type of terminal emulated, interprets the code. For a given scan code, eight values are possible: base key, and the base key combined with **Shift**, **Alt**, **Ctrl**, **Ctrl+Shift**, **Alt+Shift**, **Alt+Ctrl**, and **Ctrl+Alt+Shift** keys.
The key configuration files are text files with the .KEY extension. They contain one section per keyboard key, with each section comprising the values for the various possible combinations of the key.
- **Lock state**: Specifies the key behavior in relation to the **Caps lock** and **Num lock** keys. If **Caps lock** mode is enabled, the key pressed in **Caps lock** mode will be considered as "shifted". This is often the case with characters. If **Num lock** mode is enabled, the key pressed in **Num lock** mode will be considered as "shifted". It is often the case with Numeric Key Pad keys. If **No** is selected, both **Caps lock** and **Num lock** won't have any effect.
Sometimes the keyboard characters are language-specific. The scan code mnemonic is **nat**, which means that a national file must interpret the code sent. The national files are

text files with the .NAT extension. They can replace the keyboard configuration files when a national keyboard is specifically loaded into the emulation.

- **Terminal keyboard:** The dialog displays the values and actions generated for the different states of the keys on the selected terminal keyboard. You can also see the values returned by the keys of the current terminal keyboard and use these values for your PC's keyboard. These values cannot be modified.
- **Canceling changes:** To cancel the changes you make to a scan code and apply the previous value, click the **Scan code** section **Cancel** button. The cancellation only applies to the current changes: If you click another key without canceling, the changes are accepted.
To cancel changes made to the keyboard since the last save, click **Cancel** on the right of the keyboard.
- **Saving the modified keyboard:** Click **Save** or **Save As** to save the modified keyboard file. Keyboard files are text files with the .KEY extension.

Assigning a value to a PC keyboard key

1. Display the emulated terminal keyboard by selecting the **Terminal Keyboard** option.
 2. Select the key to be assigned on the terminal keyboard.
 3. Drag the terminal keyboard key to that on the PC keyboard.
- Otherwise, to assign a particular value to a PC keyboard key that has been pre-selected, display the actions editor by selecting the **Actions list** option.
 - The left-hand drop-down list can be used to select the type of value assigned to the key: character string, scripts, .MAC type macros, function keys, and mnemonics. On the right, you can type in or select from a list the value assigned to the key.

> Sending a character string

From the actions editor, select **String** from the left-hand drop-down list. For a simple character, in the right-hand editing field, enter the character to send when the key is struck:

- For a printable character: type in the character directly with or without quotation marks (a or "a" or 'a' will send letter a).
- For a decimal character: type in the decimal value of the character, between 0 and 255 (97 for a, 63 for ?).
- For an octal character: type in the octal value of the character, between 0 and 0377, preceded by 0 (0141 for a, 077 for ?).
- For a hexadecimal character: type in the hexadecimal value of the character, between 0 and FF preceded by 0x (0x61 for a, 0x3f for ?).

Note:

The ' character is coded by \' and the \ character by '\\'

To have a key send a character string, enter the characters (between quotes and in the correct order) on the desired key in the right-hand editing field. If a character is not printable, use decimal, hexadecimal, or octal notation, preceded by the \ character. Mnemonics aren't recognized in character strings.

Examples of character strings:

| | |
|------------------|---|
| Value: 'aef' | Result: string "aef" displayed on the screen |
| Value: 'a\033be' | Result: character "a", escape and character "b" not displayed, then character "e" (final result: string "ae" displayed on the screen) |
| Value: 'a\0x08b' | Result: character "a", backspace (bs) on the "a", then character "b" (final result: character "b" displayed on the screen) |
| Value: "\32i\10' | Result: space (sp), character "i" and linefeed (lf) |

> **Running a local script**

To write a small script for the key, select in the actions editor the script language that you want to use in the left drop down list:

- VBScript
- JScript (JavaScript)
- EScript, Esker's proprietary language


Click  in the right edit field to launch the script editor.

> **Running an existing script function**

To use a function that is already registered in a functions library, select in the actions editor the .VBS (VBScript) or .JS (JavaScript) file in the left drop down list. Select the function to assign to the keyboard key in the right drop down list.

> **Running a macro**

This feature enables you to use the macro files created with the macro language of previous Tun versions (EScript). The macro files are text files with extension .MAC.

1. From the actions editor, select **Macro** from the left-hand drop-down list.
2. In the right-hand editing field, enter the macro file path or select it by clicking .

> *Associating a function key*

Function key files are text files with the .FUN extension.

1. From the actions editor, select **Function** from the left-hand drop-down list.
2. Select the function key of your choice from the right-hand drop-down list:
 - The labels proposed are those listed in emul.fky, copied into the emulator's installation directory.

- The values assigned to the various function keys are listed in the function keys file in each type of emulated terminal.

Emul.fky comprises one section per type of terminal, containing the labels of from one to one hundred function keys. You can modify this file to call labels of your choice, symbolizing the function keys of the emulated terminal.

Associating a mnemonic to a key

Mnemonics are short words interpreted by the emulator to perform a particular action. In the actions editor, select **Mnemonics** in the left drop down list. The mnemonics of the Esker asynchronous emulator can symbolize:

- A simple character (**nul** to send 0).
- An action (**nlock** to activate the NumLock key to lock the numerical keypad).
- A dead key, or a key that only has an effect after another key is pressed (**tilde** to add a ~ to a letter).

Simple characters

| Mnemonic | Decimal | Octal | Hexadecimal |
|----------|---------|-------|-------------|
| nul | 0 | 00 | 0x00 |
| soh | 1 | 01 | 0x01 |
| stx | 2 | 02 | 0x02 |
| etx | 3 | 03 | 0x03 |
| eot | 4 | 04 | 0x04 |
| enq | 5 | 05 | 0x05 |
| ack | 6 | 06 | 0x06 |
| bel | 7 | 07 | 0x07 |
| bs | 8 | 010 | 0x08 |
| ht | 9 | 011 | 0x09 |
| lf | 10 | 012 | 0x0a |
| nl | 10 | 012 | 0x0a |
| vt | 11 | 013 | 0x0b |
| ff | 12 | 014 | 0x0c |
| np | 12 | 014 | 0x0c |
| cr | 13 | 015 | 0x0d |
| so | 14 | 016 | 0x0e |
| si | 15 | 017 | 0x0f |
| dle | 16 | 020 | 0x10 |

| | | | |
|-----|-----|------|------|
| dc1 | 17 | 021 | 0x11 |
| dc2 | 18 | 022 | 0x12 |
| dc3 | 19 | 023 | 0x13 |
| dc4 | 20 | 024 | 0x14 |
| nak | 21 | 025 | 0x15 |
| syn | 22 | 026 | 0x16 |
| etb | 23 | 027 | 0x17 |
| can | 24 | 030 | 0x18 |
| em | 25 | 031 | 0x19 |
| sub | 26 | 032 | 0x1a |
| esc | 27 | 033 | 0x1b |
| fs | 28 | 034 | 0x1c |
| gs | 29 | 035 | 0x1d |
| rs | 30 | 036 | 0x1e |
| us | 31 | 037 | 0x1f |
| sp | 32 | 040 | 0x20 |
| del | 127 | 0177 | 0x7f |
| ind | 132 | 0204 | 0x84 |
| nel | 133 | 0205 | 0x85 |
| ssa | 134 | 0206 | 0x86 |
| esa | 135 | 0207 | 0x87 |
| hts | 136 | 0210 | 0x88 |
| htj | 137 | 0211 | 0x89 |
| vts | 138 | 0212 | 0x8a |
| pld | 139 | 0213 | 0x8b |
| plu | 140 | 0214 | 0x8c |
| ri | 141 | 0215 | 0x8d |
| ss2 | 142 | 0216 | 0x8e |
| ss3 | 143 | 0217 | 0x8f |
| pu1 | 145 | 0221 | 0x91 |
| pu2 | 146 | 0222 | 0x92 |
| sts | 147 | 0223 | 0x93 |
| cch | 148 | 0224 | 0x94 |

| | | | |
|-----|-----|------|------|
| mw | 149 | 0225 | 0x95 |
| spa | 150 | 0226 | 0x96 |
| epa | 151 | 0227 | 0x97 |
| csi | 155 | 0233 | 0x9b |
| st | 156 | 0234 | 0x9c |
| osc | 157 | 0235 | 0x9d |
| pm | 158 | 0236 | 0x9e |
| apc | 159 | 0237 | 0x9f |

Actions

| Mnemonic | Action |
|-----------------|--|
| nop | No action |
| lshift | Activates left shift key |
| rshift | Activates right shift key |
| ctrl | Activates control (Ctrl) key |
| alt | Activates Alternative (Alt) key |
| clock | Activates Caps Lock key |
| nlock | Activates Num Lock key |
| slock | Activates Scroll Lock key |
| cal0.....cal9 | Successive pressing of numerical keys to obtain the corresponding decimal code (such as <Alt> 1-2-3 in MS-DOS) |
| hdcopy | Hard Copy of the screen |
| scr1 | Switches to session 1 |
| scr2 | Switches to session 2 |
| scr3 | Switches to session 3 |
| scr4 | Switches to session 4 |
| altpg1...8 | Switches to the specified page (on multi-page terminals) |
| nscr | Switches to the next session |
| send | Sends Windows file (Alt-F7) |
| receive | Begins reception of a Windows file (Alt-F8) |
| freceive | Ends reception of a Windows file (Alt-F9) |
| brk | Sends a break signal to the host |
| femul | End of emulation (Alt-F10) |
| win | Returns keyboard control to Windows |

Dead Keys

| Mnemonic | Example |
|-----------------|----------------|
| acute | È |
| grave | É |
| cflex | Â |
| tilde | Ñ |
| trema | Ï |
| ring | Å |
| cedil | Ç |
| bar | € |

Synchronous emulation

Click on a PC keyboard key to identify the value sent when this key is struck, while reading the content of the active edit field. You can also find out the value of a key combined with the **Shift**, **Ctrl**, and **Alt** keys: activate the base key and the combination key, then read the content of the active edit field.

A key whose scan code has changed appears in dark gray on the PC keyboard.

Note:

It is only possible to combine with the **Ctrl** key on the left or right of the PC keyboard if the **Left Ctrl key** and/or **Right Ctrl key** check boxes have been selected in the **Keyboard** tab of the **Options** dialog.

To reassign its default value to a key, right click the key, select a combination, and click **Default**. In the same way, to delete the key value, choose **Clear**.


> *Assigning a terminal keyboard's key value*

To assign a real terminal keyboard's key value to the PC keyboard key:



1. Select a key to change on the PC's keyboard as well as the combination key.
2. Select a key on the terminal's keyboard to assign as well as the combination key.
3. Drag and drop the terminal keyboard key to the PC key position.

> *Assigning actions*

To assign one or more actions to a key, double-click the key and select **Shift**, **Alt** or **Ctrl** to define the key combination. You can also double-click the active edit field. The actions editor appears.




To add an action, click . Select the action to assign to the key from the list box.

- **Function**: Select the function you want from the list box.
- **String**: Enter the character string in the field on the right.

- **Macro:** Enter the file path for the macro in the field on the right or select a macro file using .
- In the case of a local script, select the script language to use and click  in the right edit field to launch the script editor and write your script.
- To use an existing script, select the .VBS (**VBScript**) or .JS (**JavaScript**) file in the left drop down list. Select the function to assign to the keyboard key from the right drop down list.

Note:

The order in which you enter the actions is the order in which they're executed when the key or combination of keys is pressed.

To delete an action from the list, select it and click . To move an action up or down in the list, select it and click  or .

Click **Save** to save the modified keyboard.

Click **Load** to load an existing keyboard.

Note:

Keyboard configuration files in synchronous emulation are text files with the .KBD extension.

Choosing a national keyboard (UNIX asynchronous emulation)

When customizing the keyboard, you can associate the **Nat** mnemonic with a keyboard key, so that the emulator displays a character specific to a language on the screen. In this case, the national keyboard file associated with the session must correspond to the desired language.

Note:

The national keyboard files are text files with the .NAT extension.

Select **Session > National keyboard**. Then, select the language of your choice. If you select **Auto**, the default language used by your system will be selected.

Configuring the mouse (asynchronous emulation)

Associating actions with one or more mouse events enhances the more traditional use of the mouse. Events are the simple click and double-click on one or two of the three mouse buttons (left button, right button and middle button on certain mice). Esker asynchronous emulation enables a series of actions to be assigned to the two or three buttons, actions that will be executed when these buttons are simple or double-clicked.

Select **Session > Mouse**.



- The left window proposes the various mouse events: simple or double-click on the left button, the middle button (if there is one), and the right button.
- Each of these events can be associated with one or more instructions, which will run as a sequence unless one of them is blocking (see list below). In this case, the following ones will not be executed. The list of instructions appears in the right window for each mouse event selected in the left window.

Note:





Actions associated with double-clicking a mouse button are executed following those associated with a single click, if any.

> Adding/deleting an event

By default, six possible events with a three-button mouse are proposed.

- You can delete an event from this list by selecting it and then clicking  in the left window.
- To add one when the list is incomplete, click  in the right window.

> Adding/deleting/moving an instruction

- Select an event in the left window. To add an instruction, click  in the right window. From the drop-down list, select the type of instruction to add and, if necessary, complete the right-hand edit field of the drop-down list.
- To delete an instruction, click  in the right-hand window.
- To move an instruction, use  and  to move it down or up.

> Sending a string

This will send a string as if it entered on the keyboard. Enter a string in the editing field to the right of the list of instructions.

> Running a local script

To write a small script to assign to the mouse event, select the script language to use from the left drop down list:


- VBScript
- Jscript (JavaScript)
- EScript, Esker's proprietary language

Click  in the right edit field to launch the script editor.

> Running an existing script function

To use an existing function, select a .VBS (VBScript) or .JS (JavaScript) file in the left drop down list. Select the function to assign to the mouse event from the right drop down list.

> **Running a macro**

The selected event triggers a macro. In the right editing field, enter the macro file path or select it by clicking .

> **Function keys**

The selected event sends the value of a function key. Select the function key of your choice from the right drop-down list:

- The labels are listed in emul.fky and copied into the emulator's installation directory.
- The values assigned to the various function keys are listed in the function keys file available for each type of emulated terminal.

Note:

Function key files are text files with the .FUN extension.

> *Other functions*

The other available functions are as follows:

- **Send mouse event to host:** mouse activation will have a meaning specific to the application. This action is blocking if the mouse support is activated in the application.
- **Send word under cursor / Send character under cursor:** the word/character on which the mouse event takes place is sent as if it were entered directly on the keyboard.
- **Context menu:** where applicable, the context menu concerning the screen area clicked by the mouse. This action is blocking.
- **«No menu» mode menu:** when the menu bar is hidden, the list of menus is displayed during the mouse event. The menu actions can then be accessed. This action is blocking.
- **Normal selection:** the mouse event enables a screen area to be selected while holding the button pressed down during selection. This action is blocking.

Script editor

Three languages are available:

- VBScript
- JScript (JavaScript)
- EScript, Esker's proprietary language

Once you selected one of these three languages, you can use buttons, when available, to write your script:

: To cut and paste selection into the Clipboard.

: To paste selection into the Clipboard.

: To paste the Clipboard contents.

: To add special characters.

Hotspots

A hotspot is a screen zone identified by a character string. When the emulator recognizes this string in a screen, it displays one or more controls on the screen. When the user activates a control, it executes one or more actions.

Note:

Actions may be executed automatically upon string recognition, without any control appearing and without any user action. This is an “auto-active” control.

If several zones are recognized, the hotspot is identified with the largest one. When several controls are liable to overlap, the largest is only one displayed. If a control is to display on the recognized string, the control will match the size of the string.

Types of controls

The different types of hotspot controls available are:

- **Button:** one or more actions are performed when the user clicks on the button. The button text can be static or dynamic (updated by the emulator).
- **Menu:** one or more actions are executed when selecting a menu option. The list of menu options appears when the button representing the menu is clicked.
- **Text:** with neither button nor menu, you can trigger actions on a particular zone of the screen. Several cases are possible:

A character string appears on the chosen screen zone, and one or more actions execute when the user clicks on the text. This is a Text control. The displayed text can be static or dynamic (updated by the emulator).

If the control text is empty (with no additional screen display), the control is Transparent text.

As soon as the string is identified on the screen, one or more actions automatically execute without user intervention. This is an Auto-active control.

- **Combined list:** the character string selected from the list is sent to the connection. The content of the list can be static or dynamic (updated by the emulator).

Note on combined lists:

In the case of asynchronous emulation, a combined list control can only be active if the cursor is on its zone. The character string selected from the list is then transmitted after striking the **Enter** key on the keyboard.

In the case of synchronous emulation, several controls can be carried out simultaneously. When a

control becomes active, the cursor is positioned in the zone it controls. The character string selected from the list is then executed when the cursor leaves the control.

- **Scrollbar:** one or more actions execute when the user clicks a particular part of the scrollbar. This enables list navigation with the mouse.

List of controls per hotspot

All the following operations are performed using **Session > Hotspots**.

The list of hotspots takes the form of an objects tree. In this tree, you will configure the hotspot(s) of your choice, whose controls will be displayed on the screen when the associated character string is recognized.








- For each recognized string, there is a hotspot object.
- For each hotspot, there are one or more controls.


By default, the tree contains a single object, which is the root of all the others.

Creating a new hotspot

1. Go to the hotspots root and click **New**.

- **Description:** Enter a name for the hotspot. This name identifies the hotspot in the tree.
- **Type:** Select the first control type to associated with the hotspot by clicking the corresponding button:

| | | | |
|---|------------------------|---|-----------------|
|  | A button |  | A text |
|  | A menu |  | A scrollbar |
|  | A transparent text |  | A combined list |
|  | An auto-active control | | |

- **Search string:** Enter the character string to be recognized.
- **Match case:** The case is sensitive (upper case recognized or not).
- **Whole word:** The string represents a complete word. Uncheck if the character string may not be a complete word.
- **Regular expression:** The string is a regular expression containing generic characters (wild-cards), such as *, ?. If checked, choose wild-card characters from the list, using . Allowed wild-cards are:
 - "." to search for any character
 - "[]" to enter a range in which the character looked for must be found.
 - "[^]" to enter a range in which the character looked for must not be found.
 - "*" to look for 0 to n occurrences of the character situated immediately to the left.

"+" to look for at least one occurrence of the character situated immediately to the left.
"?" to look for 0 to 1 occurrences of the character situated immediately to the left.
Precede all wildcards with "\".

- **Full screen/Zone:** Select **Full screen** to search for the string in the entire screen, or define a particular screen zone by selecting **Zone**. In this case, you can:
Enter the coordinates of the zone directly in the boxes (in the order: row and column of top-left corner, row and column of bottom-right corner).
 - Manually capture the screen zone concerned by the search, after clicking **Capture**.
2. You can also specify the display attribute of the characters in the string to be recognized. To do this, click **Advanced**.
- **Use attributes:** Check, then select or deselect attribute types to include or exclude from the search.
If attribute A is deselected, the character string search will include all attributes except attribute A.
If attribute A is selected, the character string search will only include attribute A.
 - **Split characters:** If you are looking for a string containing a number of complete words, specify the characters that separate these words. This field is available if **Whole word** in the previous screen is selected.
3. Click **OK** to return to the previous screen.
4. Click **Next**.
- **Description:** Enter a name to identify the control.
 - **Positioning:** Select the option corresponding to the control position on the screen:
 - **On the hotspot:** the control will position itself on the recognized string.
 - **Fixed on screen:** the control will position itself in the coordinates in the following order: row and column of top-left corner, row and column of bottom-right corner. You can use **Capture**, and then capture the coordinates of the hotspot directly on the screen using the mouse.

Note:

In the case of an Auto-active control, you do not define the control's position. These trigger the execution of one or more actions as soon as the hotspot is identified, without any particular display.

5. Click **Next**.
6. **Appearance:** This screen appears for Button, Menu, and Text controls. You can state which text and/or image is to appear at the location of the chosen control.
- **Use the identified zone text/Display text:** To display the same text as the recognized zone, select **Use the identified zone text**. Otherwise, select **Display text** and enter the display and tooltip text. If, in order to modify the displayed text dynamically, you associate an action with the initialization phase of a Text button or control, the dynamic text will take priority over the text entered here.

- **Display image:** To display an image on the button, enter the absolute path of the image file chosen.
7. **Scrollbar:** This screen appears for Scrollbar controls.
- **Direction:** Select the orientation of the scrollbar (horizontal or vertical).
 - **Scale:** Select the type of scrollbar movement:
 - **Undefined number of steps:** Use with an indefinite number of steps (a step is made at each click on the ends of the scrollbar).
 - **Fixed values:** Set scroll movement values. Then, enter the number of steps, the number of elements situated in a page for scrolling in page up/down mode (click inside the scrollbar) and the initial position of the scrollbar (in steps).
8. **Actions:** This screen is available for Button, Menu, Text, Transparent text, Auto active, and Scrollbar type controls. It can be used to customize:
- **Button:** actions to execute when the control appears or when clicked.
 - **Menu:** actions to execute when the control appears or when selecting an option.
 - **Text/Transparent text:** actions to execute when the control appears and mouse events.
 - **Auto active:** actions to execute when the string is recognized.
 - **Scrollbar:** actions to execute for each scrollbar movement.

Note:

For any control, for each event you can define several actions that will run as a sequence. Selecting the event in the left-hand window on the **Actions** screen calls up the list of corresponding actions in the right-hand window.

- **Disable after execution:** Check to allow execution of these actions only once.
9. **Options:** This screen is available for the Button, Menu, Text and Combined list type controls. In the case of a **Combined list** type control, you will only define the font format and the effects.
- **Font:** Select the name of the font you want to use for the text displayed:
The default font proposed is **System**, and corresponds to the font of your client station. You cannot alter its size from the emulator.
You can also select a particular font (for example Arial, Times New Roman, etc.) for which you can choose the size.
 - **Style:** Select the style of your text: **bold**, **italic**, or **underlined**.
 - **Text:** Select the option you require: **centered**, justified **left**, or justified **right**.
 - **Image:** Select the option you require: **centered**, justified **left**, or justified **right**.
 - **Colors:** Select the control **Background color**, the **Text** color and the color that will be replaced by transparency in the image (**Image transparent color**).

Creating a new control for a hotspot

1. In the list of hotspots (**Session > Hotspots**), go to the mother hotspot and click **New**. The type of control selection box appears.
2. Select the type of control to create, and then click on **OK**.
3. For each type of control, a particular wizard is started, helping you to customize the control.
4. Follow the instructions given in each screen proposed by the wizard.

Creating a control from a zone selected on the screen

To create a control for a string selected on the screen:

1. Display the context menu by right clicking after selecting string to be recognized.
2. Select **Create a hotspot**. The hotspot wizard opens.
3. For each screen, proceed as if creating a hotspot as above.

Note:

After creating a control from a zone selected on the screen, the hotspot and its control created in this way appear in the list of hotspots defined in the session. This list is accessible from **Session > Hotspots**. You can then add other controls to this hotspot.

Modifying a hotspot or control

After creation you can modify the definition of any hotspot: identification, string to recognize and search parameters.

1. Display the list of hotspots by selecting **Session > Hotspots**.
2. Select a hotspot or control from the list.
3. Click **Modify**.
4. Make the changes.

Deleting a hotspot or control

1. Display the list of hotspots by selecting **Session > Hotspots**.
2. To delete a hotspot or a control from the list of hotspots, select it and click **Delete**.
3. To delete all elements from the list of hotspots, click **Delete all**. The hotspots root will remain available for creating new hotspots and their controls.

Control actions

Events on a button: There are two default events:

- Initialization (optional), which corresponds to the moment at which a string is recognized and the control is created. You may use this phase is useful to retrieve a character string and assign it as a button text.
- A click on the button.

Menu options: A menu is in fact a button comprising several options. To associate actions with the menu, you must first of all define the options it is to contain, and then associate them with one or more actions.

Events on a text: A text control is a screen zone that reacts to mouse events. By default, a single event is proposed: initialization as in the case of a button (for dynamic text definition). You can then add the mouse events for which one or more actions are to be executed.

Events on a transparent text: This is similar to text, but the text initialization option is not available.

Events on an auto-active control: Auto-active controls automatically trigger one or more actions at string recognition. The only event linked to this control is **Matching** (string recognition).


Events on a scrollbar: By default, the five events linked to a scrollbar are proposed:

- Initialization, executed before each other event. This phase in particular consists in repositioning the cursor in the zone controlled by the scrollbar, to enable execution of the actions associated with the events.
- Left/right and up/down movements.
- Page jumps (page left, page right, or page up, page down).


Combined list options: The list is created when the string is recognized. Therefore it contains only the initialization event linked to the **Combined list** type control. A list contains character strings. The string chosen by the user will be sent on the communication channel. The list can be filled out in two ways: manually or by a script.

- **Manual method:** You can enter all the elements of the list in the editor proposed in this screen. To do this, select **List**, and, with the editor, add the elements as they will appear.
- **Script method:** You can create a script that will be able to retrieve the information necessary for producing the list. To do this, select the type of script language you want to use (**VBScript**, **JScript** (JavaScript), or **EScript**), then write your script in the script editor.

> Adding an event (Text or Transparent text)

1. In the events list, click .
2. Select the type of event from the right-hand drop-down list and the combination key that can be used in the right-hand drop-down menu (**Normal** for no key).

> Adding an option (Menu and Combined list)


1. In the list of options, click .

2. Enter the name of the option, as it will appear in the menu or combined list.



> **Modifying the name of an option (Menu and Combined list)**

1. In the list of options, double-click on an option.
2. Enter its new name.


> **Deleting an event or option**



1. In the list of events or options, select the event or option.
2. Click .

> **Modifying the order of the options (Menu and Combined list)**

1. In the list of events or options, select the event or option.
2. Use  and  to move it upwards or downwards.

> **Associating an action with an event or option**

1. Select the event or option from the left-hand list.
2. In the list of actions, click .
3. Select the type of action to be executed.

- **String:** In the field to the right of the type of action, enter the character string you want to send on the communication channel.
 - **Function:** In the drop-down list to the right of the type of action, select the function key you want to send on the communication channel. In the case of an asynchronous emulation, the labels are listed emul.fky, copied into the emulator's installation directory. The values assigned to the various function keys are identified in the function keys file available to each type of terminal emulated.
 - **Mnemonic (asynchronous):** In the drop-down list to the right of the type of action, select the mnemonic you want to send. Mnemonics are short words interpreted by the emulator in order to perform a particular action.
- > See “Customizing the keyboard” on page 64 for a sample list of mnemonics.
- **Macro:** This feature enables you to use the macro files created with the macro language used in the previous Tun versions. In the edit field to the right of the type of action, enter the path for the macro file (.MAC) or select it by clicking .
 - **Running a local script:** If you wish to write a small script, select the script language that you want to use in the left drop down list. Click  in the right edit field to launch the script editor.
 - **Running an existing script function:** To use a function that is already registered in a functions library, select in the actions editor the .VBS (VBScript) or .JS (JavaScript) file name in the left drop down list. From the right drop down list, select the function to assign to the control.

Macros

Use macros to automate actions that you carry out regularly in Esker emulators. A macro is a small program consisting of a series of commands or instructions to automatically execute a complete series of actions that you would normally have done manually. You can use macros to:

- Connect/disconnect to the server automatically.
- Describe an action or series of actions to execute when the user indicates with the mouse or a keyboard key.
- Combine several commands to automate a series of tasks.

You can create macros in macro libraries in a workspace, either by recording them from an emulation session or by creating them from scratch. You can then associate these macros with a keyboard key, a hotspot, or to a session's connection and disconnection. These macros may be written in **JavaScript** or **VBScript**, with which you can use emulation OCX functions supplied by Esker and are accessible from emulators via the macro administrator. These macros are stored in files representing macro libraries (.JS for JavaScript macros and .VBS for VBScript macros) that may contain one or more macros.

You can also reuse your macros written in the old Esker macro language or in **EScript**. These macros are recorded in files with a .MAC extension. They may be modified using a text editor and executed from emulators via the macro administrator.

- > Descriptions of emulation OCXs and the EScript language are given in the on-line help files **Emu-IApi.hlp** and **Escript.hlp** respectively, located in the **Pc2host/win_32/docs/misc** directory on the CD-ROM.

Start the macro administrator



Macro libraries are saved in the workspace and are organized using the macro administrator. Select **Tools > Macro > Macros**. The macro administrator dialog appears.

Creating and debugging a macro using the macro editor

1. In the macro administrator dialog, select a destination macro library file into which you will add a macro. The language used for this macro will be the same as the language used for the library file in which it is created.
2. Click **New macro**. The macro library file opens in the macro editor.
3. Once you have written and debugged the code for your macro, save the file and close the editor. You can also debug the macro.


Creating a macro using the macro recorder

You can use the Esker macros recorder to record a series of actions carried out in an emulation session, in the language of your choice. The series of actions will be replayed when a recorded macro is run.

1. In the macro administrator dialog, select a destination macro library file into which you will add a macro. The language used for this macro will be the same as the language used for the library file in which it is created.
2. Click **Record** to start the macro recorder. The record macro dialog appears.
3. In the **Macro name** field, enter the name of the new macro.
4. Click **OK**. The macro recorder is then active.
5. Perform the series of actions to record in the macro.
 - You can choose to automatically encrypt character strings sent to a host in the recorded macros script. Click  on the record tool bar. The encryption mode depends on the select option.
 - You instruct the emulator to wait until it receives a character string before carrying out the next operation. Use the mouse to select the character line(s) to wait for, and then click  on the record tool bar.


Note:

In synchronous emulation, the default waiting time for a character string is 60 seconds. In asynchronous emulation, the default waiting time for a character string is equal to the time between the last recorded action and when the **Wait for selection** option is selected. You can modify this waiting time by editing the macro.

6. You can pause and resume recording a macro at any time. Click  on the record tool bar.

Note:

When recording a macro in an asynchronous emulation, the recorder will interpret any pause that you make as a "Sleep(time)" type instruction, where time is the time during which the keyboard is inactive.

7. Click  on the record tool bar. The macro will then be recorded in the specified macro file.

Modifying a macro or a macro library using the macro editor

1. In the macro administrator dialog, select the macro or the macro library file that you want to modify.
2. Click **Edit**. The corresponding macro library file is then opened in the macro editor.
3. Make the modifications, then save the file and close the editor.

Creating a macro library

1. In the macro administrator dialog, click **New file**.
2. Input the **File name** of the macro library.
3. Enter a comment in the **file description** field.
4. Select the language to be used in this macro library file (VBScript or JavaScript).

Deleting a macro library

1. In the macro administrator dialog box, select a macro library file.
2. Click **Delete**.

Executing a macro

- To execute a macro written in **JavaScript** or **VBscript** in the macro libraries, select the macro in the macro administrator dialog. Then, click **Run**.
- To execute a macro written in **EScript** or in the old Esker macro language, click **Run** in the macro administrator dialog. Then, enter the path of the file corresponding to the macro to execute.

Encrypting macros

You can use macros to exchange character strings with a server. These character strings appear in the macro script. It is important to make these strings invisible in some cases (for example a password). Do this by encrypting these character strings. There are two ways of encrypting a character string:

- Using an external authentication library (**passlib.dll**) . Select **Tools > Macro > External library**.
- Using the encryption algorithms proposed by Esker. Deselect **Tools > Macro > External library**.

Encrypting a character string






1. Select **Tools > Macro > Encrypt a string**.
2. Enter the character string to encrypt, and click **Encrypt**.
3. Select the encrypted string.
4. Copy it into the clipboard by pressing **<Ctrl><C>**.
5. Click **Close**.
6. Copy this character string into your macro script using **<Ctrl><V>**.

Saving macros

Macros are saved in the macro library file in the current emulator workspace. Save your workspace to save the modifications made to macro files.

Example macro: Recording a connection macro with password encryption

The following example shows how to record a connection macro with encryption of the password sent to the server:

1. Start recording a macro within an emulation session.
2. Deactivate encryption by deselecting .
3. Press **Enter** to see the "login" prompt.
4. Select "login" on the emulation screen, and then click  in the record tool bar.
5. Enter the user name and press **Enter**.
6. Select "password" on the emulation screen, and then click  in the record tool bar.
7. Activate encryption by selecting  in the record tool bar.
8. Enter the password and press **Enter**.
9. Deactivate encryption by selecting **Macro > Encrypt sends**.
10. Indicate that the macro must recognize the shell prompt, by selecting "\$" and then clicking  in the record tool bar.
11. Stop recording.

UNIX connection macro in VBScript

```
' DESCRIPTION:Default Macro File for VBScript scripting
Sub HPCConnection()
Sleep = Application.ActiveDocument.Object.session.Sleep(6000)
Send = Application.ActiveDocument.Object.session.Send("root\r")
Sleep = Application.ActiveDocument.Object.session.Sleep(6000)
Send = Application.ActiveDocument.Object.session.Send 8 ("@0bLaZcE.2eH84hS5viA")
Sleep = Application.ActiveDocument.Object.session.Sleep(12000)
Send = Application.ActiveDocument.Object.session.Send("ls\r")
End Sub
```

IBM connection macro in JavaScript

```
// DESCRIPTION:Default Macro File for JavaScript scripting
function IBMConnection()
```

```
{  
Application.ActiveDocument.Object.session.Send("admin")  
Application.ActiveDocument.Object.session.PressKey("Field Exit")  
Application.ActiveDocument.Object.session.Wait(10000)  
Application.ActiveDocument.Object.session.Send 8 ("xbGazdE.BfC8DhA6H")  
Application.ActiveDocument.Object.session.PressKey("Enter")  
Application.ActiveDocument.Object.session.Wait( 10000)  
}
```

Function-Key Panel Editor

A function-key panel is a window containing buttons that may be summoned during an emulation session. From a panel, you can:

- Transmit a pre-defined character string on the communication channel. This simulates a keyboard entry.
- Execute a macro.
- Execute a specific action, like launching an application, printing, and so on.

Key panel buttons may contain:

- Text (centered, left, or right justified, on several lines).
- An image (centered or tiled).
- An image and text.

Just as the keyboard changes with the use of the Shift or Alt keys, the buttons on a key panel can have three levels. Each button level can display a different value and carry out a specific operation. The level is defined by the use of a **Lock** button.

Key panels allow users to replace the keyboard with mouse actions. Key panels help traditional operations offer the same ease of use as standard Windows applications. Each key panel may contain up to 128 buttons.

Starting the function-key panel editor

You can start the function-key panel editor from the **Start** menu in the same way as any other program, or from an emulation window using **Tools > Function-key panel Editor**.

Creating a panel



Select **Options > New**.

Creating a button

There are two types of button:

- **Simple** buttons (required) offer three levels (or options) depending on the state of the lock button (if any) on the panel.
- **Lock** buttons (optional) change the state (function) of all the simple buttons on the same key panel. A panel may have no more than one lock button. Lock buttons have three lev-

els (settings), allowing each of the simple buttons on the same panel to have three different functions.

1. Click  to create a simple button, or click  to create a lock button.
2. Click and drag the pointer inside the key panel window to define the button's size and shape.
3. Release the mouse button.

Linking properties to a button

Double-click a button to display the **Button Parameters** dialog. The dialog box for the definition of simple buttons has three levels indicated by the Level 1, Level 2 and Level 3 tabs. Each level represents one of the three possible states the button can have. The fields are the same for each level.

Note:

Lock buttons have no **Action** associated with them. Lock buttons change the state of the other buttons on the function-key panel.

Button parameters dialog

The dialog for the definition of buttons has three levels indicated by the **Level 1**, **Level 2**, and **Level 3** tabs. Each level represents one of the three possible states the button can have. Buttons may have levels if there is a **Lock** button on the panel. The information for a button may vary with each level, so that a button can have a different name, label, and function with each click of the lock button on the panel.

Text

Enter the label for the button.

Tooltip

Enter the tooltip text. This text appears when the mouse pointer hovers over the button for a short time.

Image


The button can use a bitmap (.bmp) image as a label. Enter the full path name here, or use **Browse** button to find an image in the LDAP database.

Settings


Click this to specify options for the text and/or image. The **Select Button Options** dialog appears.



Action

The **Action** field defines the character string sent to the screen or the action taken clicking a button in an emulation session. Actions and strings are given line by line according to the order of execution. To add an action or string to a panel button:

1. Click  to add a new action. Two fields appear.
2. Choose the type of action to associate with the button in the list box on the left. The default type is Text. This type sends a string of characters. You may choose a different type from the list:
 - **Function-key** assigns a function-key to the button.
 - **Macro** executes a macro.
 - **Action** executes an action defined in the **Escript.hlp** file.
 - **Other** allows the use of panels created with earlier versions.
3. Give details of the action in the right field. This depends on the selected action:
 - **Text**: Enter a character string to send.
 - **Function-key**: Enter a function key number, or select one from list.
 - **Macro**: Enter a path to the macro (.MAC) or click the button to select one.
 - **Action**: Enter the action number or name. See **Escript.hlp** for a list of available actions.

To change an action you've already defined, double-click and modify it.

To delete an action, select it and click .

When a button is associated with multiple actions, you can choose the execution order. To do that, select an action and click  to move it up the list, or  to move it down the list.

Using Lock buttons

When you place text or an image on the lock button at level 1, 2, or 3, the button has one, two, or three different states (respectively). All of the simple buttons on the panel will then have one, two, or three settings (one for each of the defined states of the lock button).

When the user clicks a lock button, the button will change the state of the simple buttons on the same panel according to the levels defined in its **Button Parameters** dialog.

Define default button settings

You can define default button settings that are used automatically for any new button.

1. Choose **Options > Default Button Settings**.
 - If the option isn't available (**Selected Button Options** appears instead), a button in the model key panel is selected. Make sure no buttons are selected before choosing the option.

2. A dialog box similar to the **Selected Button Settings** dialog opens. See **Creating a simple button**.

Panel Settings and Positioning



You can define the settings assigned to the current function-key panel. Choose **Options > Panel Settings**

- **Panel type:** Select the type of emulation you're going to use (3270, 5250 or asynchronous emulation).
- **Title bar:** Select the type of panel title bar you want (normal, half-height or none).
- **Panel name:** Enter the name of the key panel for each button level. Click the arrow beside the field and choose the level (level 1, 2 or 3).
- **Docking:** Select the type of docking you want to give the function-key panel. If you choose **Normal**, **Wrapped** or **Don't change**, select the possible positions for the key panel (top or bottom, right or left). The type of docking selected here is the **Default** option proposed by the emulator when the function-key panel is selected.
- **None:** The function-key panel can't be transformed into a toolbar.
- **Normal:** The function-key panel becomes a normal toolbar.
- **Wrapped:** If there are too many keys to fit in the window, they're re-organized on more than one line or column.
- **Don't change:** The function-key panel changes into a toolbar, but the layout of the keys remains the same, even if it doesn't fit on the screen.
- **Panel paste:** The function-key panel is positioned in the emulation window as defined by the **Possible positions** fields (centered, top, bottom, left or right). The panel doesn't change into a toolbar.

To view your changes, choose **Options > Test Mode**, or click .

Set Tab Order

The tab order controls the order in which the buttons are selected or highlighted as the user presses the **Tab** key.

1. Click  to display the button order for modification.
2. Right-click the panel button that is to be highlighted by default with the right mouse button. This sets that button's number to zero (0).
3. Click each of the remaining buttons in the desired tab order.
4. Repeat steps 2 and 3 to change the order again (if an error occurs).
5. When the order is correct, click .

Note:

Lock buttons are always last in tab order.


Opening an existing function-key panel

You can adapt an existing function-key panel. Examples supplied by Esker can be found in the emulators' installation directory. A panel file has the extension .PAN. To open an existing function-key panel:


1. Click  or choose **File > Open**.
2. Choose the key panel file you want to open.


Saving a function-key panel

You can save your function-key panel in a file with the .PAN extension. To do that:


1. Click  or choose **File > Save** (or **Save As** to save the function-key panel under a different name).
2. Enter the name of the function-key panel.

Testing a key panel

To test the key panel without loading it into the emulator: click  or choose **Options > Test Mode**. This function simulates the key panel as if it were being used in the terminal emulator.

To stop the test, click the same button again .

Selecting buttons

To select one button, click it. To select multiple buttons, Click , and do one of the following:


- Draw a rectangle around the buttons you want to select.
- Hold the **Shift** key, and click each button to include.


Moving and resizing buttons


To move a button or change its size, select the button with the mouse. This displays dimensioning handles at each corner. Click and drag these handles to resize the button. To move the button, click inside the handles, and then drag the button to a new position. You may move or resize multiple (selected) buttons at once.

Sizing multiple buttons identically

To size several buttons identically, select them, and then click:



 Gives buttons the same width.

 Gives buttons the same height.

 Give buttons both the same height and width.

Duplicating a button


To duplicate a button, select it and do one of the following:

- Hold the **Ctrl** key on the keyboard down and move the mouse. When you release the mouse button, an identical button to the selected button appears.
- Click  and then .
- Choose **Edit > Copy** and then **Edit > Paste**.

All the data on the button, text or bitmap, is copied.

Deleting a button

To delete a button, select it and do one of the following:

- Use the **Del** key on the keyboard.
- Click .
- Choose **Edit > Cut**.


In the last two cases, the button is copied to the clipboard. You can recover the button by using the **Paste** function.

Aligning the buttons

You can align buttons as follows:

- Align the buttons on a grid.
- Align the buttons in relation to each other.
- Center the buttons in relation to the panel (vertically, horizontally).
- Apply the same dimensions to several buttons (same width, same height or same dimensions).

Aligning the buttons on a grid

By default, the buttons are positioned in the panel as you draw them. To activate the grid, click . The grid settings (display, number of pixels) are defined in the **Alignment** dialog. To change the alignment grid settings, choose **Options > Align**.

- To activate the grid, select **Use grid**.
- Select **Display grid** to display the grid.

- You can also enter the spacing between the horizontal and vertical lines of the grid in pixels. The default value is 10.

Relative button alignment

To align several buttons (at least 2) in relation to each other, select them, and then click the appropriate button in the toolbar:



Aligns buttons on the left side.



Aligns buttons on the right side.





Aligns buttons along the top.



Aligns buttons along the bottom.

Centering buttons

To center a button in the key panel, select it and click  (to center it vertically) or  (to center it horizontally).

Toolbar



Opens a new panel file and closes the current panel file, if one is open.



Opens an existing panel file.



Saves the panel file. The default extension is .PAN.



Cuts the selected button(s) and copies it/them to the clipboard.



Copies the selected button(s) to the clipboard.



Copies the button in the clipboard to the current panel.



Normal selector arrow. Can be used to select one or more buttons in the model panel.



Click to draw a new button.



Click to draw a lock button. Only one lock button is allowed per panel.



Toggles test mode on and off.



Sets tab order. Click the first button with the right mouse button and number the remaining buttons by clicking them with the left button.



Aligns the selected buttons on the left side.



Aligns the selected buttons on the right side.



Aligns the selected buttons along the top.



Aligns the selected buttons along the bottom.



Centers the selected buttons vertically.



Centers the selected buttons horizontally.



Attributes the same width to the selected buttons.



Attributes the same height to the selected buttons.



Attributes the same width and height to the selected buttons.



Activates alignment.

A

Advanced Use of Asynchronous Emulator

Tun's asynchronous emulator lets you define and customize every aspect of an emulation session, including keyboards, escape sequences, and character tables.

The emulation parameters are grouped together in an entity called the terminal. Each type of terminal (file `.ter`) is associated with various files containing the information necessary for communications between the PC and the server: keyboard file `.key`, function keys file `.fun`, escape sequences file `.seq`, and so on.

Data from the PC to the server goes through the following filters. The filters perform different operations on the data depending on the type of data and the filter settings.

- Keyboard filter (for `.key` files): each code sent from the keyboard (each key is identified by a code known as the scan code) refers to a piece of information. This information can be:

A character (or a string of characters) to send.

A script or a macro of type `.mac` to be executed.

A function key (link with the function keys file `.fun`).

A mnemonic for which the various lists can be found in the Using the emulators chapter, Keyboard customization, Asynchronous Emulation section. If the mnemonic is "nat", this means that you should refer to a specific national feature (national file `.nat`).

- Function-key filter (`.fun` file): Each function-key is assigned a value.
- Code conversion filter (`.snd` file): Converts ASCII files in some types of emulation.
- National filter (file `.nat`) : the scan code sent by the keyboard can be re-routed by the `.key` file to a value of the `.nat` file specific to the language used.

In the next stage, the data reaches the host. The host processes the data and sends a reply to the PC. The reply is also filtered:

- Escape sequence filter (for `.seq` files): The emulator processes the escape sequences sent by the Unix host and links them to one or more actions (clear the screen, move the cursor, start an application, etc.).
- Control code filter (for `.cod` files): When the host sends special characters known as control codes (the decimal values 0 through 31 and 128 through 159), Tun Plus looks up the actions mapped to the characters in a table).
- Character table filter (for `.tab` files): Displays the characters correctly on the screen. It ensures Unix host codes are correctly interpreted by the emulator.).

You can modify the .key, .fun and .seq files for particular sessions. You seldom need to modify the .nat, .snd, .cod and .tab files.

All the configuration files are text files which can be opened in a Notepad-type utility and then modified.

You can however access these files and configure them using the configuration window of the terminal settings. This window is accessible from the Administrator tool, when you choose Properties in the Terminal context menu in an asynch session. The following window opens:



The corresponding filename of the terminal type for the current session appears in the dialog's title bar. All displayed files correspond to the parameters associated with this type of terminal (for example: keyboard file ansi.key, function keys file ansi.fun).

To load another terminal type, click Load and select a .ter extension file.

Files in any field can be edited from this dialog by selecting them and clicking Modify. For a .key file (Keyboard field), a model of the keyboard appears. For all other file types, the file opens in Notepad.

Escape sequences

The asynchronous emulator uses ".seq" files to interpret the flow of data from the host machine. The ".seq" files associate one or more actions (cursor movement, clear screen...) with the reception of character strings (typically called escape sequences).

> Opening an existing .seq file from the Administrator reference directory

A .seq file is a text file that can be read by any text editor. The escape sequence files supplied by Tun can be found in the Tools\Applications Access\Unix Emulations\Specific Data\Escape Sequences directory of the resources tree. To edit a .seq file, select Properties from its context menu.

> Opening an existing .seq file from a session in the Administrator

Select the Properties option in the context menu of the Terminal element of a session.

Click on the .seq file chosen from the drop-down menu of Escape sequences then click on the Modify button. The .seq file opens in the default test editor.

Content of an escape sequences file

A .ses escape sequences file comprises three separate parts:

- Terminal initialization, which enables the terminal to be set to the initial status needed for establishing communication between the server and the terminal. This part can be described in the first line(s) of the file.
- The escape sequences header, when common to all the sequences (optional part).
- Definition of the escape sequences.

Here, for example, is an extract from file vt52.seq:

```
195 (2)
\033
H s 92
A s 93
B s 94
C s 95
D s 96
Y%p0%{32}%-%c%p1%{32}%-%c p 91
I s 112
J s 49
K s 52
F s 211
```

Escape sequence files may need editing in the following cases:

- If the current initialization strings are not appropriate.
- If the action associated with an escape sequence is not appropriate.
- If you want to add escape sequences for particular actions.

Syntax

The escape sequences and the initialization strings are described by actions specific to the Tun asynchronous emulator. Each action is identified by its number or label and its parameters, if any. The list of asynchronous emulator actions is given in the Escript.hlp file, copied into the installation directory.

Note:

If an emulation session has been customized or newly defined, the File Reception feature (with <Alt><F8> and <Alt><F9>) will help you to capture and analyze escape sequences and display characters sent by the host (using a Debug utility).

Terminal initialization

The first line of an escape sequence file contains the list of actions necessary for the terminal to function properly. You can add or replace actions according to your needs (see

following paragraphs for information on obtaining on-line help with escape sequences).

The Initialization line contains different actions separated by spaces. Parametered actions must be written with the appropriate parameters enclosed in parentheses and separated by commas. If there are a lot of actions, you can divide the initialization sequence into several lines, terminating each line, except the last one, with the backslash character \ (for example, the 2nd line in the file wyse60.seq).

Here, for example, is an initialization line:

```
195(0) 1195(0) 196(2) 197(2) 216
```

or with the actions label:

```
TabAsG1(0) TabAsG2(2) TabAsG3(2) G2IntoGR
```

These actions are defined as follows:

| Action | Description |
|--------|---------------------------------------|
| 195(0) | Assignment of character table 0 as G1 |
| 196(2) | Assignment of character table 2 as G2 |
| 197(2) | Assignment of character table 2 as G3 |
| 216 | Lock G2 in GR |

Sequence headers

If all the sequences in an emulation begin with the same characters, it's best to enter them on the second line of the ".seq" file. This line serves as a header for every line that follows, and allows the emulator to treat sequences sent by the server more rapidly. The escape character (\033) is very often used a header.

If you don't use a Sequence Header, you must leave the second line blank.

Defining escape sequences

The remaining fields define the actions that are related to a particular sequence. There are two types of sequences:

- Simple sequences that don't change.
- Parametered sequences that may vary.

Simple sequences

A simple sequence is a character string that doesn't contain a variable zone, and may be directly associated with one or more actions.

For example, here's a string of three characters that move the cursor one position to the left:

```
\E[D s 96
```

or with the actions label:

```
\E[D s MoveCursorLeft
```

Parametered sequences

A parametered sequence is composed of a succession of strings beginning with the character %, which serves to identify the presence of a variable. A sequence can contain several parameters, defined in three parts:

- Definition of the parameters themselves.
- Calculations and controls to be carried out on the parameter.
- Parameter format.

Parametered actions

In the case of a parametered action, there are two possibilities:

- The escape sequence is simple: the action parameters are constants.

Example:

```
c s 270("vt100")  
(or c s ChangeTerminal("vt100"))
```

Escape sequence \033c executes the 270 ChangeTerminal action (dynamic terminal change) whose parameter value is vt100.

- The escape sequence is parametered: the parameters are in the order expected by the action, which then recovers the values taken from the sequence.

Example:

```
Y%p0%{32}%-c%p1%{32}%-c p 91  
(or Y%p0%{32}%-c%p1%{32}%-c p MoveCursor)
```

Escape sequence \033Y executes the 91 MoveCursor action by recovering the values of two parameters (p0 for columns and p1 for rows).

In addition, after the action retrieves the parameter value, operations can be performed on this parameter before it is used by the action.

Example:

```
31(-30) [30,37]
```

- Check whether parameter value is between 30 and 37. If not, the action will not be carried out.
- Subtract 30 from the parameter value before it is used by the action.

Defining parameters

Note:

In the following notations, [] indicates an optional range.

Parameter definition uses the following syntax:

- %[?value by default]p[0-9] Allocation of a parameter
- *Example* %?1p2 Third parameter with a default value = 1
- %[?value by default]pi Allocation of several parameters
- *Example* %?1pi
- %g[a-z] Allocation of a variable
- *Example* %gh Allocation of the variable h

Calculations and controls

In conventional mathematic notation, the operands are separated by the operators and a particular significance forced by using parentheses. In RPN, the operands and operators are simply stacked and ‘popped’ and ‘pushed’ accordingly.

| Operator | Function | Example |
|------------|---|---|
| %[min,max] | Checks the contents in the range | %[0x40,0x7f] The variable must be between 0x40 and 0x7f |
| %c' | Stacks a constant | %'b' |
| %"string" | Push the string <i>string</i> on the string stack | %"green" |
| %{nn} | Stacks a decimal constant | %{64} |
| %g[a-z] | Pops a variable off the stack | %gh |
| %P[a-z] | Stacks a variable | %Ph |
| %V | Push the vertical position of the cursor on to the integer stack | |
| %H | Push the horizontal position of the cursor onto the integer stack | |
| %+ | Add | |
| %- | Subtract | |
| %* | Multiply | |
| %/ | Divide | |
| %m | Modulo (remainder) | |
| %& | Bitwise AND | |
| % | Bitwise OR | |
| %^ | Bitwise Exclusive OR | |
| %= | Equality | |
| %> | Greater than | |
| %< | Less than | |

| Operator | Function | Example |
|----------|-----------------|--------------------------------|
| %A | Logical AND | |
| %O | Logical OR | |
| %! | Logical NOT | |
| %~ | Bitwise NOT | |
| %I | Bitwise Inverse | (01100010 becomes 01000110) |

Parameter format

This is shown as follows:

| | |
|------------|--|
| %c | Single character |
| %s | Character string delimited by " or " |
| %S(string) | Character string ending by string. string isn't stacked, and must be less than 10 characters. The decimal, hexadecimal and octal notations must begin with the character \. The character) mustn't be used within string, and must be coded \0x29. Note: %S() represents a character string ended by the first received character. |

| % [[:]flag] [dim[.precision]][type] | |
|-------------------------------------|---|
| flag | Can have the values - + or # The result is centered to the left The result always includes the sign + or - If the first character of a conversion with a sign does not have a sign, a space precedes the result. This implies that if the flags blank and + are listed, the blank flag isn't taken into account. This flag means that the value has to be converted to a format depending upon the type of the corresponding argument. This flag has no effect on the type d. In the case of a conversion of type o, it raises the precision in such a way as to force the first digit of the result to 0. In the case of a conversion of type x or X, a result other than zero is prefixed as 0x or 0X. |
| dim | Gives the minimum number of characters occurring in the parameter. If this dimension begins with '0', the number is padded on the left by zeros and not blanks. |
| precision | Indicates the required number of digits (and not characters) corresponding to the parameter. |
| type | Can have the following values: d, o, x or X. |

A signed decimal is converted into a integer

An octal notation is converted into an integer

A non-signed hexadecimal is converted into an integer (use the lowercase letters a, b, c, d, e and f).

A non-signed hexadecimal is converted into an integer (use the uppercase letters A, B, C, D, E and F).

Example: setting the mouse for ansi emulation

```
\033Mm%p0%d;%p1%dX
```

There are two parameters in this sequence:

- %p0%d : first parameter
- %p1%dX : second parameter

These parameters are a succession of digits indicating an integer (d).

Example: reassignment of a keyboard key for ansi emulation

```
\033Q%p0%[0,9]%{59}%+%d%p1%S()
```

There are two parameters in this sequence:

- %p0%[0,9]%{59}%+%d : first parameter
- %p1%S() : second parameter

The first parameter is an integer expressed in decimal format, while the second is a character string bounded by the first character received.

For the first parameter, the following processing is required:

| | |
|--------|---|
| %[0,9] | Controls that the character is in the range of decimal values 0 through 9 |
| %{64} | Stacks the value 59 |
| %+ | Addition in Reverse Polish Notation: (car59+) is equivalent to (car+59) |

Function keys

The emulator uses the .fun files to define each of the function keys used by an emulation.

> Opening an existing .fun file from the Administrator reference directory

A .fun file is a text file that can be read by a Notepad-type text editor. You can therefore open a .fun file from this type of editor. The function key files supplied by Tun can be

found in the Tools\Applications Access\Unix Emulations\Specific Data\Function keys directory of the resources tree. To edit a .fun file, select Properties in its context menu.

> Opening an existing .fun from an session in the Administrator

Select the Properties option in the context menu of the Terminal element of an session.

Click on the .fun file chosen from the Function keys drop-down menu and then click on the Modify button. The .fun file opens in the default text editor.

Content of a function keys file

A .fun function keys file associates each function key on the terminal with the character string to be sent when it is struck.

Here, for example, is an extract from file vt100.fun:

```
[fKeyActions]
fKey1=\033OP
fKey2=\033OQ
fKey3=\033OR
fKey4=\033OS
fKey5=brk
fKey6=\033[17~
fKey7=\033[18~
fKey8=\033[19~
fKey9=\033[20~
fKey10=\033[21~
...
```

If necessary, you can change the value associated with each of the function keys. Character string encoding follows the same rules as those defined for the character strings of the .key keyboard files.

Integration of function keys in the emulator

The function keys can be invoked when striking a key on the keyboard or at a mouse event. Reference to the function keys file can thus be made in the keyboard and mouse definitions.

The emul.fky file is used to display simple labels for the various terminal function keys in the keyboard and mouse configuration boxes.

Example:

Function key fKey22 has label Shift F10: when a keyboard key or mouse event is associated with function key Shift F10, the keystroke or mouse activation corresponds to function key fKey22 whose value is defined in the function keys file (for example fKey22=\033[34~ in vt100 emulation).

Terminal configuration

You can associate a terminal configuration file (extension .ses), whose contents interact with the ".seq" and ".cod" files, with each type of terminal. A number of configuration files are supplied with Tun's emulator so that standard configuration parameters for the chosen terminal can be associated with each session.

> Opening an existing .ses file from the Administrator reference directory

A .ses file is a text file that can be read in a Notepad-type text editor. You can therefore open a .ses file from this type of editor. The terminal configuration files supplied by Tun can be found in the Tools\Applications Access\Unix Emulations\Specific Data\Terminals of the resources tree. To edit a .ses file, select Properties in its context menu.

> Opening an existing .ses file from an session in the Administrator

Select the Properties option in the context menu of the Terminal element of an session.

Click on the .ses file chosen from the Parameters drop-down menu, then click on the Modify button. The .ses file opens in the default test editor.

Content of a terminal configuration file

Here, for example, is an extract from the vt220.ses file:

```
[Intro]
ID=19971009
ParamNb=5
Param1=Cursor
Param2=Wrap
Param3=KeyMode
Param4=Keyboard
Param5=AbortEsc

[Cursor]
Label=SetupCursorStyle
ItemNb=2
Item1=SetupCursUnderline
Item2=SetupCursBlock
Action1=127 (12,14)
Action2=127 (0,14)
InitDefault=1
InitAction=%gS136%{16}%/{5}%>%{2}%{1}%@

[Wrap]
Label=SetupAutowrap
ItemNb=2
Item1=SetupON
Item2=SetupOFF
Action1=62
Action2=63
InitDefault=1
InitAction=%gS4%{2}%{1}%@

...
```


A .ses file is used to define the different terminal configuration parameters (for example, cursor style, keyboard type, sequence abort or not, etc.). Each section of the .ses file describes the various possible options for a parameter which can be selected by the user in the terminal configuration box.

Details

Identifiers

The names of combo-boxes and the listed items are stored in the emulator's language files (".lg"). The file ".ses" contains the related name file identifiers (SetupCursorStyle, SetupAutoWrap, etc. in the example) as parameters.

Order of the combo-boxes

The combo-boxes appear in the order of their definition in the section [Intro]. The items listed in a combo-box appear in the order of their definition.

Actions

The field InitAction must contain a number from 1 through N. This number is the initial choice in the combo-box when it's opened in an active session. In other cases, the field InitDefault is used. InitDefault is set in accordance with the initialization parameters of the ".seq" file. That way, the ".ses" and ".seq" files aren't contradictory.

If the dialog box relates to the active session, the actions linked to the options in the combo-box (Action1...ActionN) are executed when the dialog box is validated (OK is pressed).

Loading order of the ".ses" file

The .ses file is loaded after the ".seq" file but before the configuration file containing the users' choices.

National keyboards

In highly-specific multi-lingual environment situations, the emulator enables national filters to be employed, for example so that particular accents can be used. Using the nat mnemonic, it is then possible to assign a particular value to a keyboard key, as referenced in the .nat file. The values referenced in a .nat file are simple characters or mnemonics.

Example:

You are working in a French environment (azerty keyboard). However, a user temporarily wants to use a US keyboard configuration (qwerty keyboard). After selecting the US keyboard from the National keyboard option, the user can then redefine the keyboard by associating the nat mnemonic with key «a» on the PC keyboard. This indicates that the emulator has to refer to the .nat file to find out the value of this key in a qwerty environment (in fact, striking the «a» key will display a «q» on the screen, as if a qwerty keyboard were being used).

Reading a .nat file

When a PC keyboard key refers to the nat mnemonic, the character or mnemonic associated with it is located on the row corresponding to the key's scan code. The first column of the row gives the scan code, while the eight other columns give the value assigned to the key, in the following order:

- Key without combination (basic)
- Shift
- Ctrl
- Ctrl/Shift
- Alt
- Alt Shift
- Alt Ctrl
- Alt Ctrl Shift

Example:

The keyboard key with scan code 16 refers to row 16 of the .nat file. The value of this key, when combined with the Shift key, is defined in the third column of row 16.

> Opening an existing .nat file

A .nat file is a text file that can be read in a Notepad-type text editor. You can therefore open a .nat file from this type of editor. The national files supplied by Tun can be found in the Tools\Applications Access\Unix Emulations\Specific Data\National Keyboards directory of the Administrator resources tree. To edit a .nat file, select Properties in its context menu.

Control codes

Characters whose decimal values range from 0 through 31, and from 128 through 159 are called control codes. Control codes often directly trigger particular actions. In Tun's emulator, control codes are configured in files with a ".cod" extension. Here, for example, are the contents of the file ansi.cod:

```
nul 0
soh 0
stx 0
etx 0
eot 0
enq 0
ack 261
bel 250
bs 96
ht 99
lf 113
vt 0
```

```
ff 51
cr 97
so 0
si 0
dle 0
dc1 0
dc2 0
dc3 0
dc4 0
nak 0
syn 0
etb 0
can 0
em 0
sub 0
esc 0
fs 0
gs 0
rs 0
us 0
```

The first column contains the control code mnemonic and the second column contains the number of the action to be carried out on reception of the corresponding code. Three options can be used in the second column:

- Leave the value blank (the character is displayed on the terminal).
- Respond with 0 (no action is carried out, and the character isn't listed).
- Select an action from the list in the Escript.hlp file.

A control code can only execute a simple action.

> Opening an existing .cod file from the Administrator reference directory

A .cod file is a text file that can be read in a Notepad-type text editor. You can therefore open a .cod file from this type of editor. The control code files supplied by Tun can be found in the Tools\Applications Access\Unix Emulations\Specific Data\Control codes directory of the resources tree. To edit a .cod file, select Properties in its context menu.

> Opening an existing .cod file during an emulation session

Select the Properties option in the context menu of the Terminal element of an session.

Click on the .cod file chosen from the Control codes drop-down menu then click on the Modify button. The .cod file opens in the default text editor.

Code conversion

In some emulations, ASCII characters need to be converted before they can be sent correctly to the server. This conversion is managed by the .snd sent code files.

Here, for example, is an extract from file vt220.snd:

```
- \0xa1
> \0xa2
```

```

œ \0xa3
_ \0xa5
! \0xaa
® \0xab
ø \0xb0
ñ \0xb1
ý \0xb2
þ \0xb3
æ \0xb5
ã \0xb6

```

The left-hand column contains ASCII characters and the right-hand column contains the codes to be sent to the host machine.

> Opening an existing .snd file from the Administrator reference directory

A .snd file is a text file that can be read in a Notepad-type text editor. You can therefore open a .snd file from this type of editor. The sent codes files provided by Tun can be found in the Tools\Applications Access\Unix Emulations\Specific Data\PC to host conversions directory of the resources tree. To edit a .snd file, select Properties in its context menu.

> Opening an existing .snd file from an session in the Administrator

Select the Properties option in the context menu of the Terminal element of an session.

Click on the .snd file chosen from the Sent codes drop-down menu and then click on the Modify button. The .snd file opens in the default text editor.

Character tables

The character tables act as filters for displaying characters on the screen: An 8-bit character has 256 possible values. IBM-compatible micro-computers have their own encoding of these 256 characters. Certain characters are standard: 65 is represented as 'A', 66 as 'B', 48 as 'O', etc. Other characters, such as control characters, have a particular meaning for IBM. Different terminals have different character sets.

The objective is to define the tables for character representation. A representation is always encoded with 7 bits, that is, with the values 0 through 127.

The tables `ascii.tab` and `asciie.tab` (ASCII and the extended ASCII character set) represent the micro-computer codes 0 through 127 (`ascii.tab`) and 128 through 255 (`asciie.tab`).

Other tables are:

| | |
|-----------|----------------------|
| UK.TAB | Britain |
| DECSU.TAB | DEC supplementary |
| DECSP.TAB | DEC special graphics |

A character table file looks like this:

```

80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af

```

```

b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf
c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df
e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef
f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff

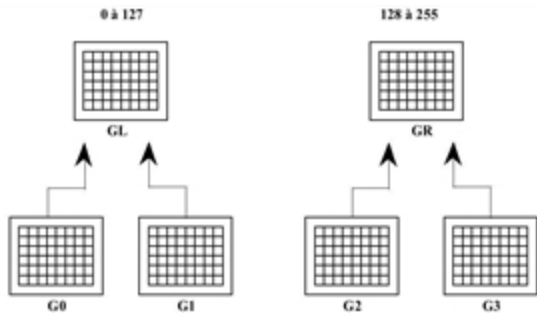
```

There are 128 fields corresponding to the 128 possible arrangements of 7 bits. The horizontal axis shows the first four bits and the vertical axis shows the last three bits. This table only contains hexadecimal codes.

To change a particular character, its hexadecimal code must first be looked up in the ASCII table and then entered in the ".tab" file.

Internal character table management

To manage these tables during an emulation session, Tun's emulator uses a model based on VT100 emulation. There are several tables present in the memory of a VT100, but only 4 tables are available at any given time:



By default, one of the tables G0, G1, G2, or G3 is loaded into GL and GR. GL represents all the characters from 0 through 127, and GR, the characters from 128 through 255.

In Tun's emulator, there are four actions related to this procedure:

| Actions | Description |
|---------|---------------------------------|
| 194 | Assigns a character table to G0 |
| 195 | Assigns a character table to G1 |
| 196 | Assigns a character table to G2 |
| 197 | Assigns a character table to G3 |

These actions are defined by a table number corresponding to those indicated in the terminal configuration box (Terminal option of the Session menu).

In Tun's emulator, eight actions allow you to fill GL and GR:

| Actions | Description |
|---------|---------------|
| 210 | lock G0 in GL |

| Actions | Description |
|----------------|--------------------|
| 211 | lock G1 in GL |
| 212 | lock G2 in GL |
| 213 | lock G3 in GL |
| 214 | lock G0 in GR |
| 215 | lock G1 in GR |
| 216 | lock G2 in GR |
| 217 | lock G3 in GR |

Finally, four other simple actions let you access the next character in the tables G0, G1, G2, or G3 without using GL or GR:

| Actions | Description |
|----------------|---------------------|
| 218 | Selective use of G0 |
| 219 | Selective use of G1 |
| 220 | Selective use of G2 |
| 221 | Selective use of G3 |

This organization of 4 active tables (two of which are available by default) is complex. Most emulations have two permanent tables (GL and GR). The configuration file lets you use 10 alternative tables. You load one of these tables into GR or GL as follows:

Example:

```
194(4) 214
```

or with the action label:

```
TabAsG0(4) G0IntoGR
```

loads the 5th table into G0, then locks G0 into GR.

This organization lets you set the parameters to match virtually all existing terminal emulations.

Alternate character font

By default, PCs are only able to display 256 characters simultaneously. This limit poses some problems when trying to create emulations for more complex terminals that offer four or five different fonts.

In MS-DOS with a VGA or SVGA card, or in Windows, Tun's emulator can display 2 x 256 characters simultaneously by using an alternate font.

For a ".tab" file to be able to use an alternate font, simply replace the desired hexadecimal value by the number 1.

For example, the value 182 refers to the 130th position (82nd in hexadecimal) of the alternate font.

Eastern European character sets

TunPlus supports Eastern European character sets for Czech, Croatian, Estonian, Latvian, and Lithuanian in the IBM3151, VT320, and VT220 emulations. To enable these character sets:

1. Click File > New.
2. Select Asynchronous Emulation.
3. Set the terminal and display type.

| Terminal | Display Type |
|----------|--|
| IBM3151 | <ul style="list-style-type: none">• For Czech or Croatian, set the Terminal and Display to IBM3151_2e• For Estonian, Latvian, and Lithuanian, set the Terminal and Display to IBM3151_13e |
| VT220 | <ul style="list-style-type: none">• For Czech or Croatian, set the Terminal to VT220_2e. Set the Display to VT320_2e. (VT220 and VT320 use the same display file.)• For Estonian, Latvian, and Lithuanian, set the Terminal to VT220_13e. Set the Display to VT320_13e. (VT220 and VT320 use the same display file)• For Czech or Croatian, set the Terminal to VT320_2e. Set the Display to VT320_2e.• For Estonian, Latvian, and Lithuanian, set the Terminal to VT320_13e. Set the Display to VT320_13e. |

4. Connect to the host.
 - For IBM3151 or VT320, confirm that the terminal settings are correct by clicking on Session>Terminal.
 - For IBM3151, change the Extended Page to 8859.
 - For VT320, ensure that the VT default character set is Dec Multinational.
5. Click on Tools > Display Editor > Fonts tab. The "Use Ansi to Oem Conversion" must be disabled.
6. Save the workspace.

Samples of Esker emulator actions

For its asynchronous emulator, Esker offers a set of over 350 actions, which can be combined within the text files for advanced terminal emulation configuration. This chapter presents a few examples of how to use these actions. The **Escript.hlp** file identifies the Esker actions by specifying their name, number and any parameters. To enable the terminal emulator to interpret these escape sequences, simply add them (if they are not already there) to the escape sequences file corresponding to the type of terminal emulated.

Note:

See the chapter "Advanced use of asynchronous emulator" in this manual for a detailed explanation of the configuration files described in the following examples.

Quitting Esker Viewer upon server request

Action **299 (Exit)** enables you to completely quit Esker Viewer hosting the ActiveX asynchronous emulation component. In ANSI emulation, this action is associated by default with the following escape sequence:

```
\033Q s 299(0)
```

The parameter associated with this action is the return code to be sent back by Esker Viewer to the program that started it. 299(1) is equivalent to `exit(1)` in a C program.

Example:

The following shell script enables you to close **Esker Viewer** as if you used **File > Quit** :

```
echo "\033\033Q"
```

File transfer requested by server

Actions **271** and **272 (RcopyPut and RcopyGet)** initiate file transfers between the PC and the server without user intervention. In ANSI emulation, these actions are associated by default with the following escape sequences:

```
\033\033put%p0%s p 271
```

```
\033\033get%p0%s p 272
```

- Parameter `p0` associated with these actions is a character string designating the source file(s) and the destination file(s).

You can also perform file transfer using macros (same **RcopyPut** and **RcopyGet** commands)

Windows to UNIX copy

The following escape sequence enables you transfer a PC Windows file to the UNIX server:

```
\033put%p0%s p 271
```

Example:

The following shell script shows how to copy the file **C:\autoexec.bat** to the current directory of a UNIX session.

```
echo "Windows to Unix copy"
echo "\033\033put\"c:\\\autoexec.bat\"\"
echo "Windows to UNIX copy finished "
```

Note:

The large number of '\' characters can be explained by the fact that the succession of command interpreters removes most of these characters.

UNIX to Windows copy

The following escape sequence enables you to transfer a file from the server to the PC running Windows:

```
\033get%p0%s p 272
```

Example:

The following shell script lets you transfer the file **/etc/passwd** to the root directory of the PC:

```
echo "UNIX to Windows copy"
echo "\033\033get\"/etc/passwd\"\"
echo " UNIX to Windows copy finished"
```

PC programs started by server

Action296 (**ExecDOSProg**) may be used to start Windows applications without user intervention. The program to execute is a parameter passed to these actions. By default, this action is not associated with an emulation.

```
\033X%p0%s p 296
```

Example:

The following shell script allows execution of the Write program:

```
echo "\033\033X\"C:\\\WINDOWS\\WRITE\"\"
```

Note:

The use of several '\' characters is necessary because of the succession of command interpreters that "strip" most of these characters.

Macro execution requested by server

Action **264 (ExecMacro)** starts a .MAC type macro at the server's request, without user intervention. In ANSI emulation, this action is associated by default with the following escape sequence:

```
\033M%p0%s p 264
```

- The parameter p0 for this action is a character string indicating the command line you want to execute.

Example:

The following shell script allows execution of macro **sysadmsh.mac**:

```
echo "\033\033M\"sysadmsh.mac\""
```

Transparent printing

In the character flow sent to the terminal emulator, the server can insert escape sequences to inform the emulator that the following characters are to be:

- Either displayed on the screen (conventional terminal emulation).
- Or sent directly to the printer connected to the PC emulating the terminal

This functionality lets the centralized application access the printer without using a buffer or print server.

Esker supplies a series of actions used to reroute the characters received from the server either to the PC screen, or to the printer, or to both simultaneously. To work in transparent print mode, the emulator must have escape sequences associated with these actions in the escape sequences configuration file.

Actions

- **260 (SetDisplayOff)** inhibits screen display of the received characters.
- **267 (BeginPrint)** enables the received characters to be re-routed to the printer using the Windows print manager.
- **261 (EndPrint)** ends re-routing of the received characters to the printer.
- **262 (BeginRecvFile)** enables a file to be received. Its parameter is the destination file name. This action can be used to send a file directly to an LPT or COM port: simply enter the name of the LPT or COM port as the action parameter.
- **263 (EndRecvFile)** terminates reception of a file.
- **259 (SetDisplay)** displays the received characters on the screen. This action is used when screen display of the characters has been inhibited (with the **260 SetDisplayOff** action).

Example 1: transparent printing on the PC's default printer via the Windows print manager

The following two escape sequences allow printing on the PC's default printer:

```
[5i s 260 267
```

```
[4i s 261 259
```

- The first sequence involves switching to transparent print mode and starting printing on the PC's default printer of the characters sent by the server.
- The second sequence involves terminating printing and inhibiting the transparent print mode so that the characters sent by the server can once again be displayed on the screen.

Example:

The following is an example of using the UNIX shell to transparently print the file `/etc/passwd` using the default printer:

```
echo "Start transparent printing test(xxxx.seq)"
echo -n "\033[5i"
echo "File PASSWORD"
echo "-----"
cat /etc/passwd
echo "-----"
echo "End of file"
echo "\033[4i"
echo "End of test"
```

Example 2: Direct transparent printing on a printer port

The following two escape sequences allow direct printing on a printer port:

```
[5i s 260 262("LPT1")
```

```
[4i s 263 259
```

- The first sequence involves switching to transparent print mode and starting printing by sending a file to port LPT1.
- The second sequence involves ending transmission of the file and inhibiting transparent printing mode so that the characters sent by the server can once again be displayed on the screen.

Example:

In the following example, the UNIX shell transparently prints a copy of the file `/etc/passwd` on the LPT1 port of the PC:

```
echo "Start transparent printing test"
echo -n "\033[5i"
echo "Transparent copy test"
```

```
echo "-----"
cat /etc/passwd
echo "End of test"
echo "\033[4i"
```

Note:

If a Windows application prints at the same time as the emulator prints transparently, the print jobs will be mixed up because of the direct printing on the port.

Dynamically changing terminal type

Applications on UNIX hosts don't always use the same terminal type, even on the same server. Using one application after another in an emulation session can therefore present problems. Action **270 (ChangeTerminal)** was designed to correct this situation. It allows the terminal type to be changed without closing the current session. In ANSI emulation, this action is associated by default with the following escape sequence:

```
\033T%p0s p 270
```

- Parameter p0 associated with this action is a character string designating the type of terminal to be emulated (ANSI, VT320, IBM3151...).

Example:

The following shell script enables the terminal to be changed to VT220:

```
echo "\033\033T\"VT220\""
```

Changing sessions automatically

Action **294 (SetDisplaySession)** allows the UNIX host to change the active terminal session without user intervention. In ANSI emulation, this action is associated by default with the following escape sequence:

```
\033S%p0%1d p 294
```

- Parameter p0 associated with this action is an integer between 0 and 31, designating the number of the session to be activated.

Example:

Use the following shell script to switch to session 2 and then session 1:

```
#To activate session No2
echo "\033\033S1"

#To activate session No1
echo "\033\033S0"
```

Mouse support in UNIX applications

Esker's asynchronous emulator can send definable sequences each time a mouse event occurs, just as if a function key were pressed on the keyboard:

- Mouse movement
- Single or double click on right button.
- Release right button
- Single or double click on left button.
- Release left button
- Single or double click on middle button.
- Release middle button

The sequences that are sent always include the current mouse position in **screen coordinates** or **virtual coordinates**. To limit the data exchange across connections, UNIX applications can request that the terminal emulator send only certain events.

In addition, the application can control the mouse in the following ways:

- Display mouse
- Hide mouse
- Move mouse
- Return mouse status and position in a specified format
- Define the time interval in a "double-click"
- Define the time interval for sending mouse movement
- Return mouse status and position

Provided actions

The following actions are provided for mouse management:

- Initialize mouse.
- Deactivate mouse.
- Display mouse cursor.
- Hide mouse cursor.
- Move mouse cursor.
- Query mouse status.
- Activate mouse and select events.
- Define format of events expected by the application.

Mouse initialization

Action **277 (InitMouse)** is used to initialize the mouse. In ANSI emulation, this action is associated by default with the following escape sequence:

\033Mi%p0%2d;%p1%2d;%p2%2dX p 277

This action requires three parameters:

- **p0** (in integer format) is the event mask expected by the program:
- Move mouse 0x01
- Click left button 0x02
- Release left button 0x04
- Click right button 0x08
- Release right button 0x10
- Click middle button 0x20
- Release middle button 0x40
- Double click 0x80
- **p1** (integer format) is the time interval between mouse "reporting", expressed by the number of clock ticks (1 second = 18.2 ticks).
- **p2** (integer format) is the length of time taken by a double-click, expressed as the number of clock ticks (5 tends to be a good value).

Note:

If p2=0, the double-click is ignored.

Those actions associated with a double-click on a mouse button are run after any actions associated with a single click. This is standard Windows procedure.

Deactivate mouse

Action **278 (ReleaseMouse)** is used to deactivate the mouse. In ANSI emulation, this action is associated by default with the following escape sequence:

\033Mc s 278

This action requires no parameters.

Display mouse cursor

Action **279 (ShowMouse)** is used to display the mouse cursor in the application. In ANSI emulation, this action is associated by default with the following escape sequence:

\033Md s 279

This action requires no parameters.

Hide mouse cursor

Action **280 (HideMouse)** is used to hide the mouse cursor in the application. In ANSI emulation, this action is associated by default with the following escape sequence:

\033Mh s 280

This action requires no parameters.

Move the mouse cursor

Action **281 (MoveMouse)** is used to move the mouse cursor to a given position. In ANSI emulation, this action is associated by default with the following escape sequence:

```
\033Mm%p0%d;%p1%dX p 281
```

This action requires 2 parameters:

- **p0**: integer representing the new X position (columns).
- **p1**: integer representing the new Y position (rows).

Query mouse status

Action **282 (QueryMouse)** is used to obtain the status of the mouse (status of buttons plus position). In ANSI emulation, this action is associated by default with the following escape sequence:

```
\033Mq s 282
```

This action requires no parameters.

Activate mouse and select events

Action **283 (ActivateMouse)** is used to select certain mouse events. In ANSI emulation, this action is associated by default with the following escape sequence:

```
\033Me%p0%dX p 283
```

- **p0** (integer format) is what the application expects to receive:

| | | |
|--------------|------|---|
| EVENT_ALL | 0x02 | Activates all the events defined by initialization |
| EVENT_MOVE | 0x01 | If (!EVENT_ALL) all events except mouse movement are returned if one of the buttons isn't pressed |
| XY_PHYSICAL | 0x04 | The current cursor position is returned in screen coordinates (virtual coordinates are the default) |
| XY_RELATIVE | 0x08 | Returns the position of the mouse cursor in relative coordinates to its previous position |
| LEFT_PANEL | 0x10 | Indicates that the left mouse button is reserved for the function-key panel |
| RIGHT_PANEL | 0x20 | Indicates that the right mouse button is reserved for the function-key panel |
| CENTER_PANEL | 0x40 | Indicates that the center mouse button is reserved for the function-key panel |

Format definition of events expected by the application

Action **284 (DefineEventsSeq)** is used to define the format of mouse events expected by the application. In ANSI emulation, this action is associated by default with the following escape sequence:

```
\033Mf%p0%s p 284
```

This action follows the mouse initialization action. It takes one parameter:

- **p0** is a character string in C format that shows how to encode transmitted events. For example:

```
\033[Mf%d;%d;%d
```

The first parameter in this string is always the mouse status (see initialization constants). The next two parameters show the x and y positions of the cursor.

By default, the string format is:

```
%02x%03x%03x
```

Implementation

Mouse support is already configured for ANSI emulation with the following lines added to **ansi.seq**:

```
\033Mi%p0%2d;%p1%2d;%p2%2dX p 277
\033Mc s 278
\033Md s 279
\033Mh s 280
\033Mm%p0%d;%p1%dX p 281
\033Mq s 282
\033Me%p0%dX p 283
\033Mf%p0%s p 284
```

Using a mouse is too complicated to simulate with a simple UNIX command or shell script. For this reason, C source code (**mouse.c**) is provided by Esker in the installation directory. This file is all you need to implement a mouse interface for UNIX. Try compiling it and using it under emulation.

Miscellaneous solutions

Color attributes in emulation

To make emulation more attractive, Esker asynchronous emulator lets you change the colors of classic screen attributes such as highlighting, underlining, etc. If you want to use different colors, you must change the **Initialization** line (the first line) of the appropriate .SEQ file.

Note:

A simpler solution uses **Session > Colors** from the emulation session.

The colors you can use and their associated codes are:

| Decimal Code | Hex Code | Color |
|--------------|----------|---------------|
| 0 | 0 | black |
| 1 | 1 | blue |
| 2 | 2 | green |
| 3 | 3 | cyan |
| 4 | 4 | red |
| 5 | 5 | magenta |
| 6 | 6 | brown |
| 7 | 7 | light gray |
| 8 | 8 | dark gray |
| 9 | 9 | light blue |
| 10 | A | light green |
| 11 | B | light cyan |
| 12 | C | light red |
| 13 | D | light magenta |
| 14 | E | yellow |
| 15 | F | white |

Six different actions are used to control color selection:

- **action 30** color for normal video
- **action 31** color for reverse video

- **action 66** blink color
- **action 67** underline color (since VGA displays are unable to provide underlining in text mode)
- **action 68** highlight color
- **action 69** dim color

Action 30 is used with two parameters: The first determines character color; the second determines background color. For example, you can display white characters on a blue background with the parameters (15,1). For color changes to take effect, enter the action number followed by the parameters between parentheses on the **Initialization** line of the .SEQ file, for example:

```
30 (7,1)
```

The same logic applies to action 31 (selection of reverse video color). Action 66 is defined with only one hexadecimal parameter. If you want a white blinking character with a blue background, you can include the sequence: 66(0x71) on the **Initialization** line. The same applies to actions 67, 68, and 69.

Note:

These parameters (66, 67, 68, 69) are coded in one byte and are defined in hexadecimal notation. For example, if you want a light green blinking character on a light magenta background, use the corresponding hexadecimal codes.

Sample sequence using this notation:

```
66 (0xAD)
```

132-column emulation

It's very easy to configure emulation with 132 columns in Windows. However, to see the entire screen, you must assign the Sys132PC font to the display settings file (.CTX).

You can use emulation session settings to set the number of columns in an emulation session. Choose **Session > Font**. For an emulation to always run with 132 columns, you can change the current .SEQ file. Modify the value of the parameter passed to action 1 as follows:

- **1(4)** 132 columns on a monochrome VGA screen
- **1(5)** 132 columns on a VGA color screen.

Action "1" determines the video display at startup. Usually the parameter is:

- **1(3)** 80 columns on a VGA color screen.

Emulation with 25 lines

Most emulations are defined by default to display 24 lines on the screen. You can include the following action in the initialization string in the .SEQ file to specify the number of lines used:

```
5 (0,23)
```

- Action 5 defines the screen margins: The first parameter (0) designates the upper margin, and the second parameter (23) designates the lower margin. This action can be used in the initialization string as well as elsewhere in the .SEQ file.

To emulate with 25 lines, add 1 to the second parameter every time action 5 is defined. If it isn't present in the .SEQ file, insert it only in the initialization string. The following parameters cause emulation to use 25 lines:

```
5(0,24)
```

For example, the file **wyse60.seq** contains the following lines:

```
5(0,23) 62 72 (First line, initialization string)
\033
...
...
e( s 5(0,23)
e) s 5(0,24)
...
...
```

Change these lines to:

```
5(0,24) 62 72 (First line, initialization string)
\033
...
...
e( s 5(0,24)
e) s 5(0,25)
...
...
```

Scancode emulation

Some word processing programs in UNIX (Word, WordPerfect) need to use more keys than are usually provided by ordinary terminals. These programs also need the <Alt> keys to send values. To handle this problem, these programs recommend the use of scancode emulations in which all the keys on a keyboard simply send their scancode, and not several different values. Esker asynchronous emulator supports scancode emulation with actions 152 and 153. In standard ANSI emulation, these actions are associated with the following escape sequences:

```
\033~5
\033~4
```

Using scancode mode

Follow these steps to use the emulator in scancode mode:

1. Switch the emulator to scancode mode by sending the sequence **\033~5**
2. Change the tty on the UNIX host using this command:

```
stty isscancode xscancode
```

To switch back to the emulator's native mode:

1. Send the sequence `\033~4`
2. Change the tty on the UNIX machine back to "normal" with the command:

```
stty -isccancode -xscancode
```

Using COM3 and COM4

Only COM1 and COM2 are completely standard on PCs. You can add two additional COM ports (COM3 and COM4) with proper definition of the IRQ and I/O addresses. In Windows, COM3 and COM4 are defined using the Control Panel. Usually, ports COM3 and COM4 use the same IRQ as COM1 and COM 2, but have different I/O addresses (COM3=3E8 and COM4=2E8).

Index

.

.COD 106

.TAB 108

A

Actions

- Dynamically changing terminal type 116
- File transfer 112
- Quitting emulator 112

ActiveX 8

ActiveX Scripting standard 8

Additional resources (asynchronous emulation) 11

Alternate character font 110

APL 37

- Characters 37
- Keyboard 37
- Mode 37

Archive (asynchronous emulation) 12

Asynchronous emulation

- .cod files 95
- .fun files 95
- .key files 95
- .nat files 95
- .seq files 95
- .snd files 95
- .tab files 95
- .ter files 95
- Additional resources 11
- Archive 12
- Character tables 95
- Code conversion 95
- Context 21
- Control codes 95
- DDE 26
- Escape sequences 95
- Function-keys 95
- Keyboard 95
- Keyboard selection 64
- Mapping an instruction to a mouse event 72
- Mouse configuration 71
- National keyboard 71, 95
- Other functions using the mouse 73
- Resources editor 11
- Running a macro using the mouse 73
- Running a script using the mouse 72-73
- Selecting a mouse event 72
- Sending a function-key using the mouse 73

- Sending a string using the mouse 72
- Standard resources 11
- Terminal 95

Auto-active (hotspots) 75

B

Baud rate 22

Button (hotspots) 75

C

Category Customization 12

Character tables (asynchronous emulation) 95

Close session on confirmation 21

Code conversion (asynchronous emulation) 95

Color selection 121

Colors

Customization 53

Combined list (hotspots) 75

Commands category 12

Configuration files

.seq 96

Connection

- Commands 62
- Customization 62
- Disconnection 62

Connection (3287 emulation) 46

Connection (3812 Emulation) 46

Connection parameters 11

Context (asynchronous emulation) 21

Control codes (asynchronous emulation) 95

Controls (hotspots) 75

- Auto-active 75
- Button 75
- Combined list 75
- Menu 75
- Scrollbar 76
- Text 75
- Transparent text 75

Copy/Paste 57

CREATING Macros 82

Customization 12

- Separator 13
- Tools menu 15

D

DDE
Topic 26

DDE (asynchronous emulation) 26

Debug 97

Distribution 19

E

EBCDIC 63

Editor
Script 73

Emulation 3287
Connection 46

Emulation 3812
Connection 46

Emulation screen
Customization 51

Encrypting macros 84

Escape sequence headers 98

Escape sequences 96

Escape sequences (asynchronous emulation) 95

Esker Viewer

- Creating a session 9
- Editing tools 11
- Number of recent files 16
- Opening a session 9-10
- Opening a workspace 9-10
- Options 16
- Recent files 16
- Saving a session 10
- Saving a workspace 10
- Splash screen 16
- Wizard 9-10
- Workspace 8

F

File reception 97

File transfer 58, 112

Files

- .cod files 95
- .CTX 21
- .fun files 95
- .key files 95
- .nat files 95
- .seq files 95
- .snd files 95
- .tab files 95
- .ter files 95
- CFG 9

- CFS 9
- CFZ 9
- File .CWZ 8
- File .INI 9
- File .MAC 9
- Firewall 17
- Function-key panels
 - Show or hide 61
- Function-keys (asynchronous emulation) 95
- H**
- Highlight 122
- HLLAPI 38
- Hotspots 75
 - auto-active 75
 - Controls 75
- I**
- IBM printers emulation 44
- Interface 12
- J**
- JScript 8
- JScript script languages 8
- K**
- Keyboard
 - Customization 64
- Keyboard (asynchronous emulation) 95
- L**
- Large buttons on a toolbar 14
- Logical unit 44
- LU 44
- M**
- Macro administrator 82
- Macros 82
 - Administrator 82
 - Creating a macro library 84
 - Deleting a macro library 84
 - Edit 82
 - Encrypt 84
 - Execution 84
 - Modification 83

- Recording 83
- Save 85
- Menu (hotspots) 75
- Menu bar
 - Separator 13
- Mouse
 - Configuration 71
 - Mapping an instruction 72
 - Other functions (asynchronous emulation) 73
 - Running a macro (asynchronous emulation) 73
 - Running a script (asynchronous emulation) 72-73
 - Selecting an event (asynchronous emulation) 72
 - Sending a function-key (asynchronous emulation) 73
 - Sending a string (asynchronous emulation) 72
- Multiple file transfer (synchronous emulation) 60
- N**
- National keyboard (asynchronous emulation) 71, 95
- O**
- Options
 - Esker Viewer's options 16
- P**
- Packager 19
- passlib.dll 84
- Port
 - Firewall (Proxy) 17
- Printing
 - Printing with templates 56-57
- Proxy 17
- R**
- Resources editor (asynchronous emulation) 11
- S**
- Script edition
 - Running a script using the mouse 72-73
- Script editor 73
- Scrollbar (hotspots) 76

- Separator in a bar 13
- Servers
 - Proxy server 17
- Session in Esker Viewer
 - Connection parameters 11
 - Opening 10
 - Saving 10
- Socks 17
- SSH 18
- SSL 17
- Standard resources (asynchronous emulation) 11
- Synchronous emulation
 - Character tables 63
 - Multiple file transfer 60
 - Printers emulation 44
- T**
- Template
 - Printing with templates 56
- Terminal
 - Customization 62
 - Initialization 98
- Terminal (asynchronous emulation) 95
- Terminal definition (asynchronous emulation) 21
- Terminal initialization 122
- Terminal type 116
- Text (hotspots) 75
- Toolbar 14
 - Button image 13
 - Button text 13
 - Large buttons 14
 - Separator 13
 - ToolTips 13
- Tools 15
 - Editors 11
- Tools menu 15
 - Commands 15
 - Parameters 15
- ToolTip in a toolbar 13
- Transparent printing 114
- Transparent text (hotspots) 75
- U**
- Underline color 122

V

VBScript 8

VBScript Script languages 8

W

Wizard 9-10

Workspace 8
 Opening 9-10
 Saving 10

X

Xon/Xoff 22