TN Bridge Host Integration Pack

Developer's guide

Table of Contents

OHIO Reference 208

1 Introduction

1.1 Overview

TN BRIDGE Host Integration Pack 3.5 is a set of components and productivity tools to extend and take advantage of already developed and tested screen-oriented host applications. It provides full connectivity to IBM S/390 and AS/400 systems through standard telnet protocol.

Offering a set of 100% .NET managed code (ActiveX and Borland's Delphi-native components also available), it allows development under the most popular development environments and programming languages like Microsoft® Visual Basic, VB.NET, C#, ASP, ASP.NET and Borland's Delphi 6 thru CodeGear's Delphi 2007.

Extending legacy applications with Host Integration Pack is much more profitable than starting all over again. It does not requires any kind of extra code in your host, allowing reuse of the already tested and developed host applications.

With Host Integration Pack you won't need to learn about complex communication protocols nor low level API's. Projects development-time are measured in days, not months. Your end-users will obtain new applications in a familiar and more friendly environment, reducing training time and increasing their's productivity.

Host Integration Pack offers a simplified non-event-driven programming model, making most of developments much easier. However, it still allows low-level access and event-driven programming, more suitable for developing complex applications.

Includes a productivity environment that helps during the design and development-cycle of host-integrated applications. This tool drastically reduces the training curve and testing time.

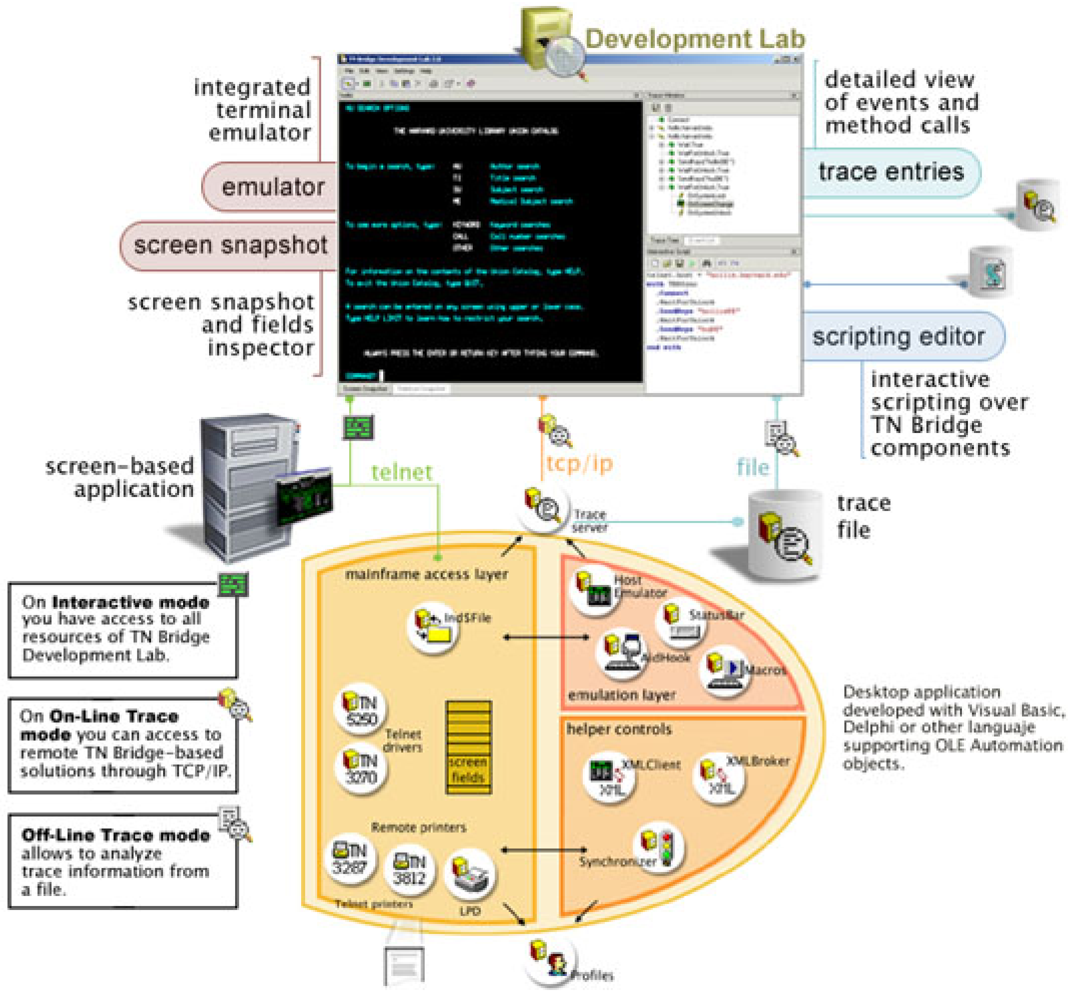
Through OHIO Interface, now an integrated feature of TN BRIDGE Host Integration Pack, you can run an OHIO application without having an emulation session running, you can concentrate on the application functions, without worrying about structure packing details or parameter command codes. OHIO also has all the benefits of the object oriented programming paradigm because it was designed as an object oriented API.

Host Integration Pack offers a powerful XML interface to the host, enabling a fast path to integrate with XML Web Services, applications in Microsoft's .Net architecture, J2EE applications, or interact with robust applications including IBM's WebSphere and Microsoft's BizTalk Servers.

1.2 Architecture

TN BRIDGE Host Integration Pack contains remote printing and file transfer capabilities as well a productivity environment. **Development Lab** is the research tool for TN BRIDGE

Host Integration Pack, tracking and testing environment, which reduces the training curve, as well as development and testing times, while ensures a more reliable application.



On **Interactive mode** you have access to all resources of TN Bridge Development Lab.

On **On-Line Trace mode** you can access to remote TN Bridge-based solutions through TCP/IP.

Off-Line Trace mode allows to analyze trace information from a file.

Desktop application developed with Visual Basic, Delphi or other languaje supporting OLE Automation objects.

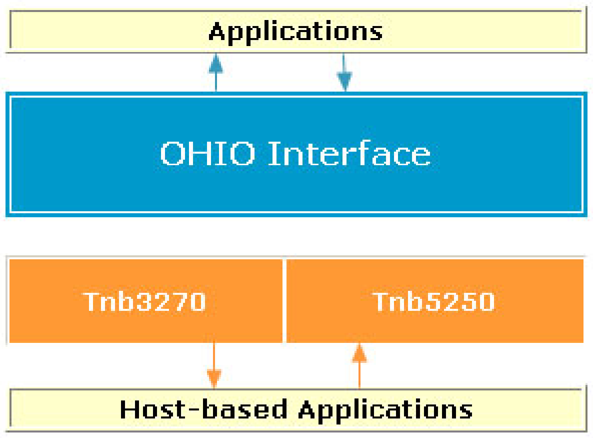
1.3 What's New in 3.5?

Following are the new implementations in TN Bridge Host Integration Pack 3.5:

- 100% .NET manage code.
- Compatibility with Framework 1.1 and 2.0.
- FCL .NET for Visual Studio .NET and Delphi for .NET.
- New XML-To-Host bridging classes.
- Improved Development Lab with XML templates builder.

Following are the new implementations in Host Integration Pack 3.0:

- Extended **OHIO** Interface. By implementing this internationally accepted standard API, Host Integration Pack enables corporations to protect their investment, as well as an easy and smooth migration path to the superior TN BRIDGE technology. TN BRIDGE Extensions to standard OHIO allows taking full advantage of the Host Integration Pack synchronous (non-event-driven) model, as well as access to other Host Integration Pack's exclusive features, like XML streaming interfaces.



- New XML Broker and XML Client components. Host Integration Pack offers a powerful XML interface, enabling a fast path to integrate with XML Web Services, applications in Microsoft's .Net architecture, J2EE applications, or interact with robust applications including IBM's WebSphere and Microsoft's BizTalk Servers.

- New XML Configuration for Profiles component. In this version, each TnbProfile component can be save in configurations in both OLE Compound Documents and XML formats.

- LPD Component. Host Integration Pack 3.0 now includes a Line Printer Deamon Component with SCS (SNA Character String) interface, emulating IBM 3287 and 3812 printers.

1.4 Trial & Full License

All our products have a 30 days free trial period. We offer full pre-sales support. Do not hesitate to contact us if you require assistance or clarification to develop your project.

Evaluation Period

If you downloaded **TN Bridge Host Integration Pack**'s Trial Version from our website or a distribution site, our software will work in Demo mode, allowing time-limited connections to the mainframe.

Please get registered to activate the full demo mode during the 30 days free trial period:

http://www.cybelesoft.com/download/register.aspx?product=HIPA

By filling a short form you will be automatically emailed a trial key file that you must save into the Windows directory and in the folder where your TN Bridge is installed:

For example: C:\Program Files\TN BRIDGE Host Integration Pack 3.5

In case you need to extend your evaluation period, please **contact us**. We will gladly provide you with a trial extension key.

Registering your Full Edition

To register your purchsed TN Bridge license, read the following topic:

- **How to Register your License**.

2 The Development Lab

TN BRIDGE Development Lab is a research, tracking and testing environment which integrates tools for easy development of TN BRIDGE-based applications. Development Lab reduce the learning curve and testing time, thus, development time is much less shorter. This unified environment involves a terminal emulator, trace viewer and interactive scripting, giving the chance to easily understand host events, inspect screens and fields, interactively test TN BRIDGE components, create code snippets, and more.

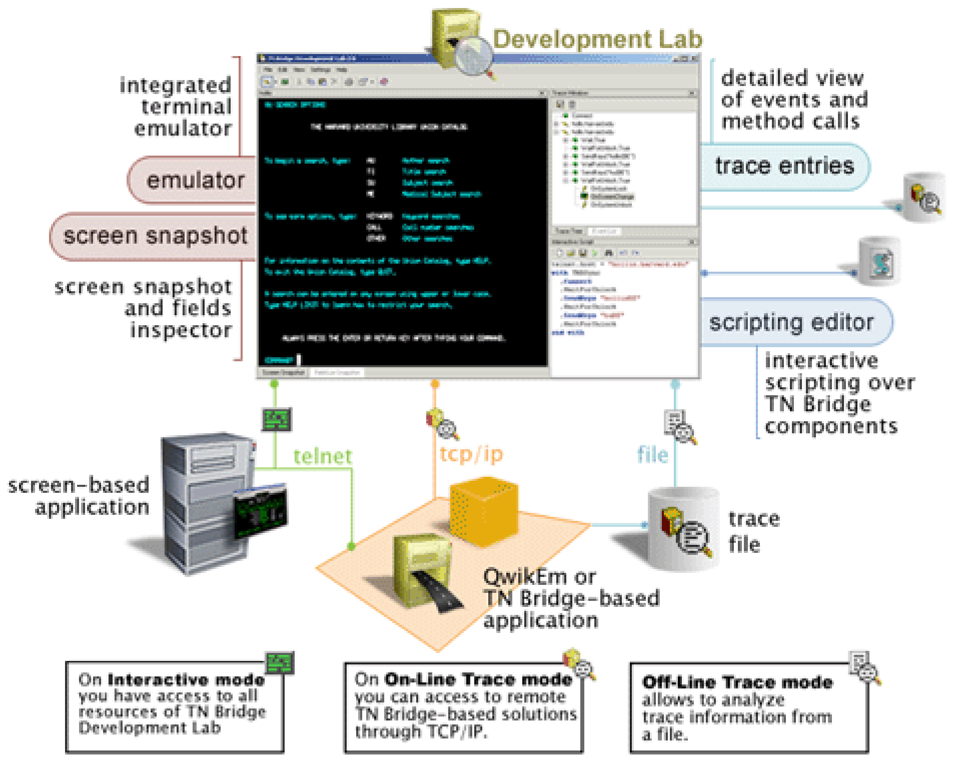
TN BRIDGE Development Lab will help you develop your applications in these scenarios:

- During planning/design time, because you will understand the screen/events sequences produced by your host application. Then you can plan and test your navigation strategy through the host screens. You can do this working in **Interactive mode.**

- During development time, using the remote tracking functionality where you can connect to your TN BRIDGE-based development, even running on another computer. You can do this working in **On-Line Trace Mode**.

To keep your deployed application free of errors. You can work in **On-Line Trace Mode** in order to connect and track a running application even in a remote workstation or locally save the trace information into a file for posterior analysis working in **Off-Line Trace Mode**.

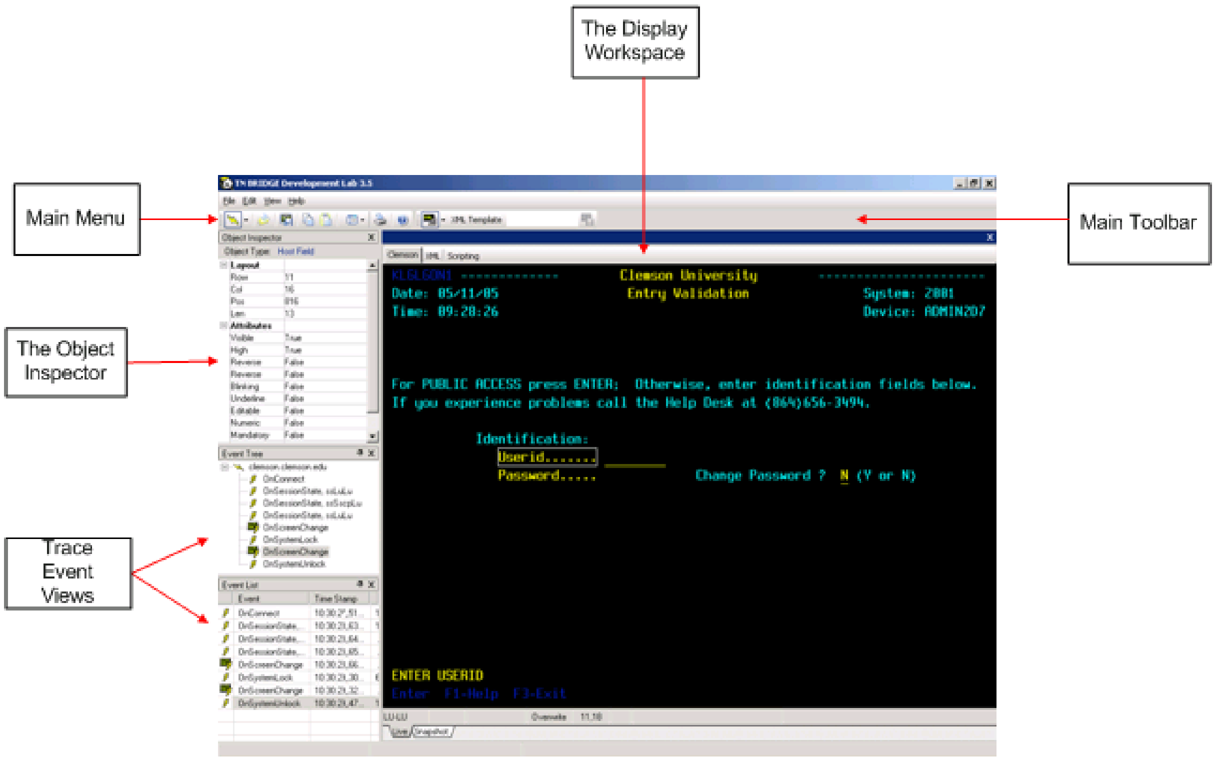
You can also **save screens as XML files** and connect to them to work with mainframe screens being off-line.


Development Lab
integrated terminal emulator
emulator
screen snapshot
screen snapshot and fields inspector
detailed view of events and method calls
trace entries
scripting editor
interactive scripting over TN Bridge components
screen-based application
telnet
tcp/ip
file
trace file
QwikEm or TN Bridge-based application
On Interactive mode you have access to all resources of TN Bridge Development Lab
On On-Line Trace mode you can access to remote TN Bridge-based solutions through TCP/IP.
Off-Line Trace mode allows to analyze trace information from a file.

2.1 Development Lab components

TN BRIDGE Development Lab's main window contains several panels offering access to the different functionalities.

- **Main menu and toolbars**: Application settings and actions can be accessed through menu items and buttons.
- **The Display workspace**: Provides access to the interactive emulation, screen snapshot, XML snapshot and interactive scripting editor.
- **The Trace Event views**: Provides tree and list views to TN BRIDGE's communication events.
- **The Object Inspector**: Displays information about screen-fields and XML objects.

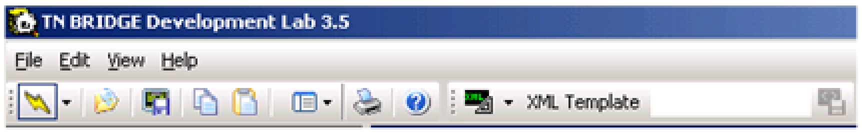


```
KLGLGON1 --------------         Clemson University         ------------------------
Date: 05/11/05                  Entry Validation                      System: 2001
Time: 09:28:26                                                       Device: ADMIN207


For PUBLIC ACCESS press ENTER;  Otherwise, enter identification fields below.
If you experience problems call the Help Desk at (864)656-3494.

          Identification:
            Userid.......  ________
            Password.....               Change Password ?  N (Y or N)




ENTER USERID
Enter  F1-Help  F3-Exit
```

2.1.1 Main Menu and Toolbar

You can access to functions through the main menu items and toolbar buttons.

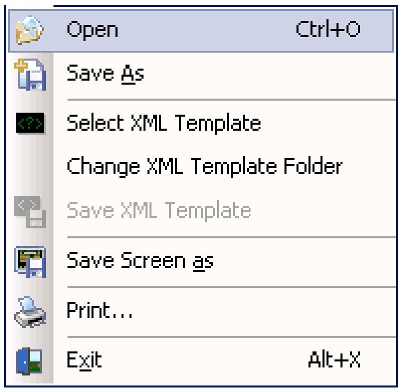
- **Main Menu** offers four submenus for accessing to actions and settings.
- **Main Toolbar** The toolbar buttons offer quick access to the most commonly used functions.



2.1.1.1 Main Menu

At the Main Menu you can find the following options:

File



Open
Opens a TN BRIDGE Trace File.

Select XML Tempate
Allows you to send a hard-copy of the screen to the printer.

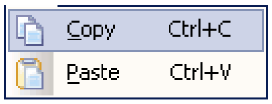
Change XML Template Folder
Use this option to display the standard printer setup dialog.

Save Screen As
Save the current screen in XML format.

Print
Allows you to send a hard-copy of the screen to the printer.

Exit
Terminates z/Scope Classic.

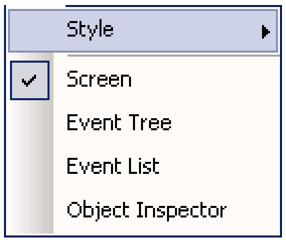
Edit



Copy
Use this option to copy the selected text-area into clipboard buffer.

Paste
Use this option to paste text from the clipboard into the screen at the cursor position.

View



Style
Allows you to choose a different Application Skin.

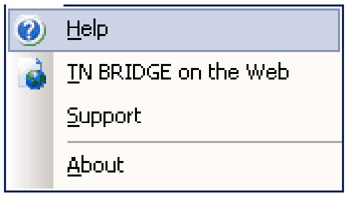
Screen
Shows/Hides the display window

Event Tree
Shows/Hides the Event Tree window

Event List
Shows/Hides the Event List window

Object Inspector
Shows/Hides the Object Inspector window

Help



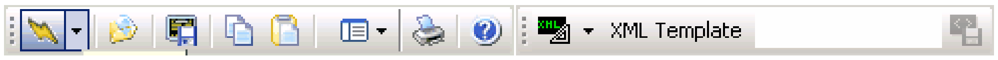
Help
Use this option to get access to this help.

TN BRIDGE on the Web
Visit our Web Site.

Support
Send us your feedback

About
Shows you information about TN BRIDGE Development Lab.

2.1.1.2 Main Toolbar



Connections

Connect/Disconnect

Use this button to connect and disconnect the current connection. This connection's name is displayed at selected session tab.

Open
Opens a TN BRIDGE Trace File

Save Screen As
Save the current screen in XML format.

Edit

Copy text
Use this button to copy a selected text-area into clipboard buffer.

Paste text
Use this button to paste text from the clipboard into the screen at the cursor position.

Preferences

Settings
Use this button to access to Development Lab Preferences window.

Print

Print screen
Use this button to send a hard-copy of the screen to the printer.

Help

Help
Use this button to get access to this help.

XML Template

Enable/Disable Template Mode

Save XML Template file

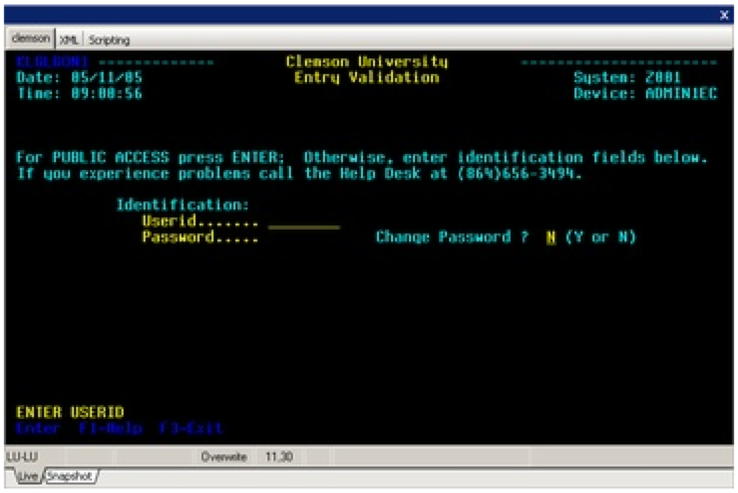
2.1.2 The Display workspace

The display workspace allows access to the following windows:

- **Interactive emulation**
- **Screen snapshot**
- **XML views**
- **Scripting window**

2.1.2.1 Interactive emulation

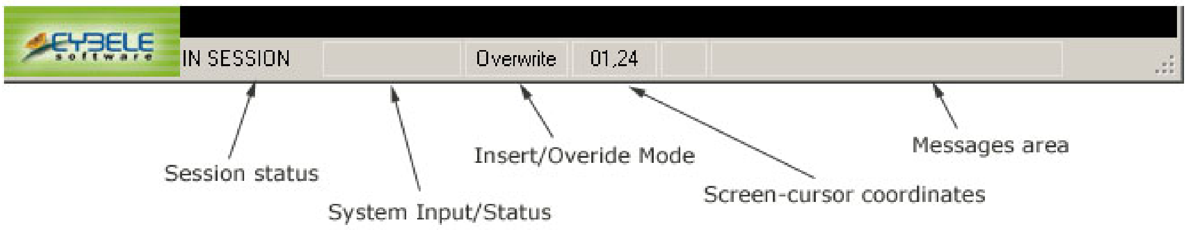
The interactive emulation window allows you to get connected to your mainframe and take the mainframe connection as source of tracing windows, XML transforming and interactive scripting.



```
ELGLGON1 --------------         Clemson University          -----------------------
Date: 05/11/05                   Entry Validation                      System: Z001
Time: 09:00:56                                                   Device: ADMINIEC


For PUBLIC ACCESS press ENTER;  Otherwise, enter identification fields below.
If you experience problems call the Help Desk at (864)656-3494.

             Identification:
                Userid....... ________
                Password.....               Change Password ?  N (Y or N)








ENTER USERID
Enter   F1-Help  F3-Exit
```

The statusbar brings you information about connection status, system status and input status. Additionally, it shows application messages.
The status bar is divided into five panels as described bellow:

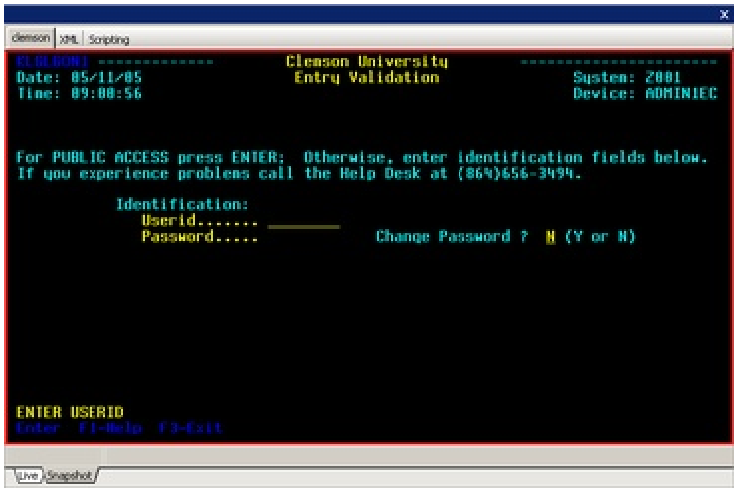
- Connection status.
- System/Input status.
- Insert/Overwrite mode.
- Screen-cursor coordinates.
- Messages area.



See also **Interactive** mode, which explains how you can create a connection setting and get connected to your mainframe.

2.1.2.2 Screen Snapshot

The screen snapshot window shows a snapshot of a mainframe screen. In interactive mode, you can switch to snapshot window by clicking on "Snapshot" tab or selecting a screen in any of the trace view windows.



```
              --------------           Clemson University          ----------------------
Date: 05/11/05                          Entry Validation                     System: 2001
Time: 09:00:56                                                             Device: ADMINIEC


For PUBLIC ACCESS press ENTER:  Otherwise, enter identification fields below.
If you experience problems call the Help Desk at (864)656-3494.

             Identification:
                Userid........ ________
                Password.....               Change Password ?  N (Y or N)







ENTER USERID
Enter   F1-Help   F3-Exit
```

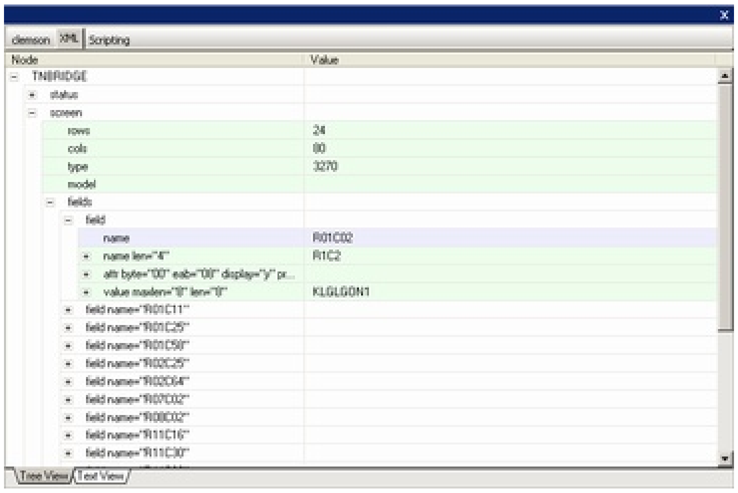
See also **Trace Event views**

2.1.2.3 XML views

The XML view windows allows to see a XML representation of the current screen. The XML representation can be generic or the one resulting by applying an XML template.

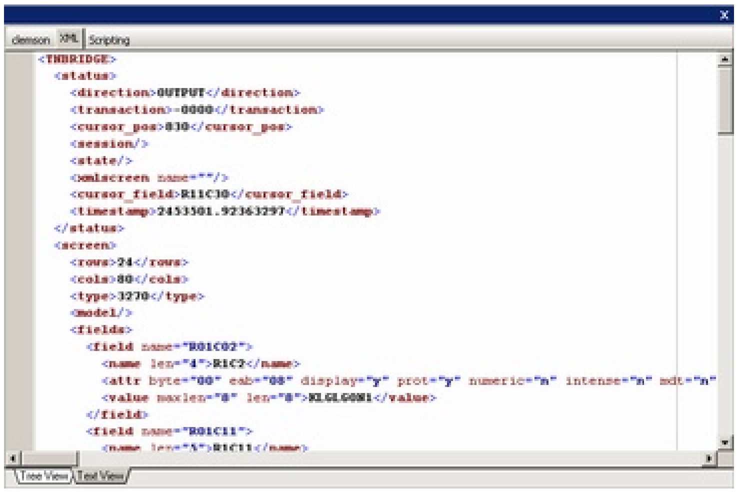
Two views are available:

- XML Tree view:



Node	Value
TNBRIDGE	
status	
screen	
rows	24
cols	80
type	3270
model	
fields	
field	
name	R01C02
name len="4"	R1C2
attr byte="00" eab="00" display="y" pr...	
value maxlen="8" len="8"	KLGLGON1
field name="R01C11"	
field name="R01C25"	
field name="R01C50"	
field name="R02C25"	
field name="R02C64"	
field name="R07C02"	
field name="R08C02"	
field name="R11C16"	
field name="R11C30"	

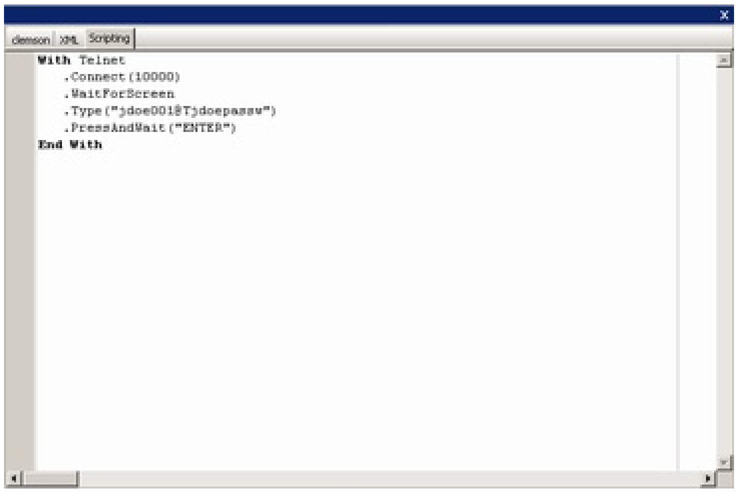
- XML Text view



```
<TNBRIDGE>
  <status>
    <direction>OUTPUT</direction>
    <transaction>-0000</transaction>
    <cursor_pos>830</cursor_pos>
    <session/>
    <state/>
    <xmlscreen name=""/>
    <cursor_field>R11C30</cursor_field>
    <timestamp>2453501.92363297</timestamp>
  </status>
  <screen>
    <rows>24</rows>
    <cols>80</cols>
    <type>3270</type>
    <model/>
    <fields>
      <field name="R01C02">
        <name len="4">R1C2</name>
        <attr byte="00" eab="08" display="y" prot="y" numeric="n" intense="n" mdt="n"
        <value maxlen="8" len="8">KLGLGON1</value>
      </field>
      <field name="R01C11">
        <name len="5">R1C11</name>
```

2.1.2.4 Scripting window

The scripting window shows the scripting editor, where you can create code snippets to test the mainframe connection.



```
With Telnet
    .Connect(10000)
    .WaitForScreen
    .Type("jdoe001@Tjdoepassw")
    .PressAndWait("ENTER")
End With
```

2.1.3 The Trace Event views

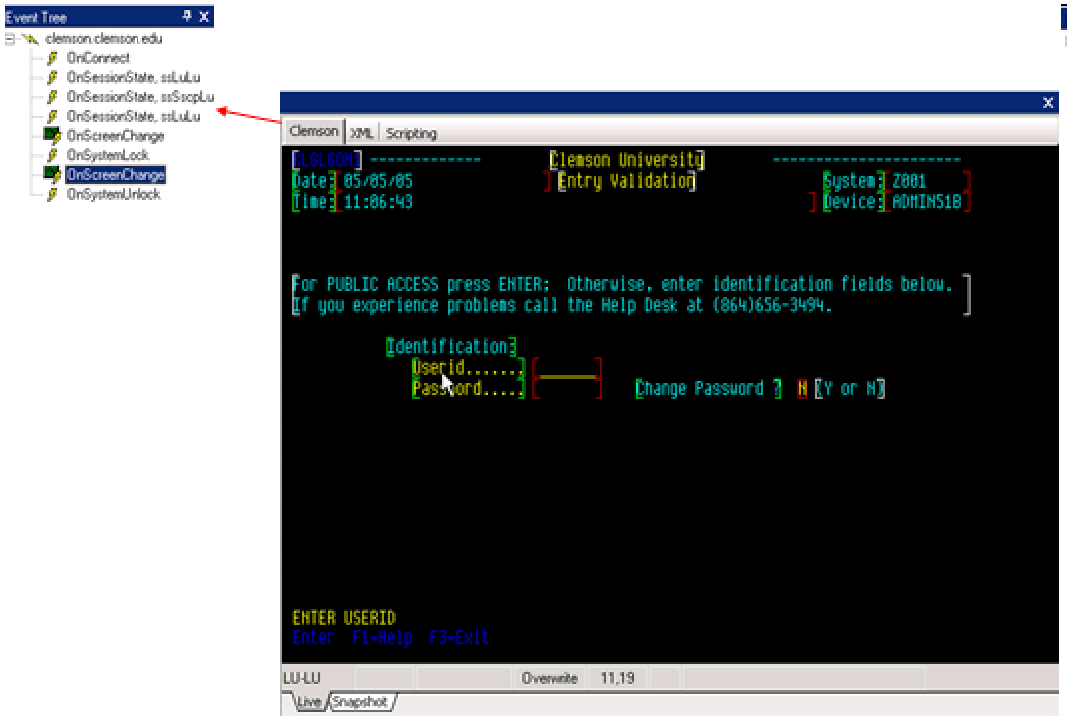
The Trace window shows the events generated by TN BRIDGE in relation to the communication with the mainframe. This information can be gathered from the interactive emulation, from an external application using the TN BRIDGE components or from file.

Trace events are displayed in two views:

- **as Tree view**, where events are displayed in hierarchical way.
- **as List view**, where events are displayed as a list.

2.1.3.1 Tree View

Events are nested by connection and send event.



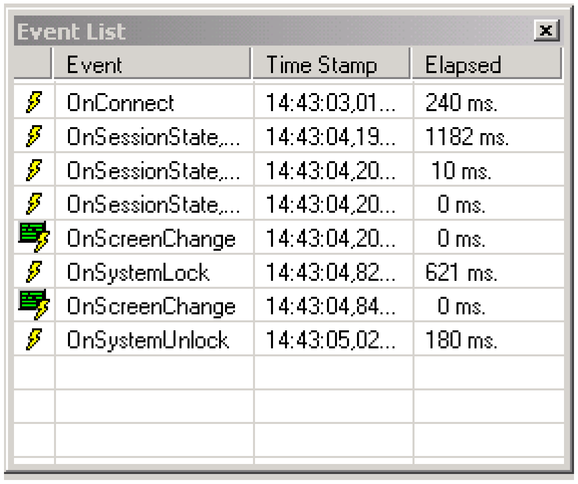
```
KLGLGON1 -------------          Clemson University       -----------------------
Date: 05/05/05                  Entry Validation              System: Z081
Time: 11:06:43                                                Device: ADMIN51B


For PUBLIC ACCESS press ENTER:  Otherwise, enter identification fields below.
If you experience problems call the Help Desk at (864)656-3494.

           Identification:
             Userid.......  [        ]
             Password.....  [        ]     Change Password ?  N [Y or N]










ENTER USERID
Enter  F1=Help  F3=Exit
```

	Connect method	Represent a *Connect* method call.
	General Event	Represent the following events: • *OnConnect* • *OnDisconnect* • *OnSystemLock* • *OnSystemUnlock* • *OnSessionState* • *OnConnectionFail*
	OnScreenChange event	Represent the *OnScreenChange* event and the associated screen data received.
	SendAid method	Represent a *SendAid* method call (or any *SendAid* related methods or properties).
	OnSendAid event	Represent the *OnSendAid* event and the associated screen data sent.
	General Method	Represent a method call.
	Script execution	Represent the execution of a script code block (an ASP page) using TN BRIDGE controls. All TN BRIDGE methods and events between the start and the end of the script will be included here.

The Trace Entries pane shows a tree with TN BRIDGE's method calls, communication events and custom trace data. Each method call entry is shown as a group, nesting the upcoming data (including another method calls) until it ends.

2.1.3.2 List View

The Event List has a record of all the events procesed during the session. It also displays additional information such us the exact time of the event reception and the elapsed time it took to be triggered.



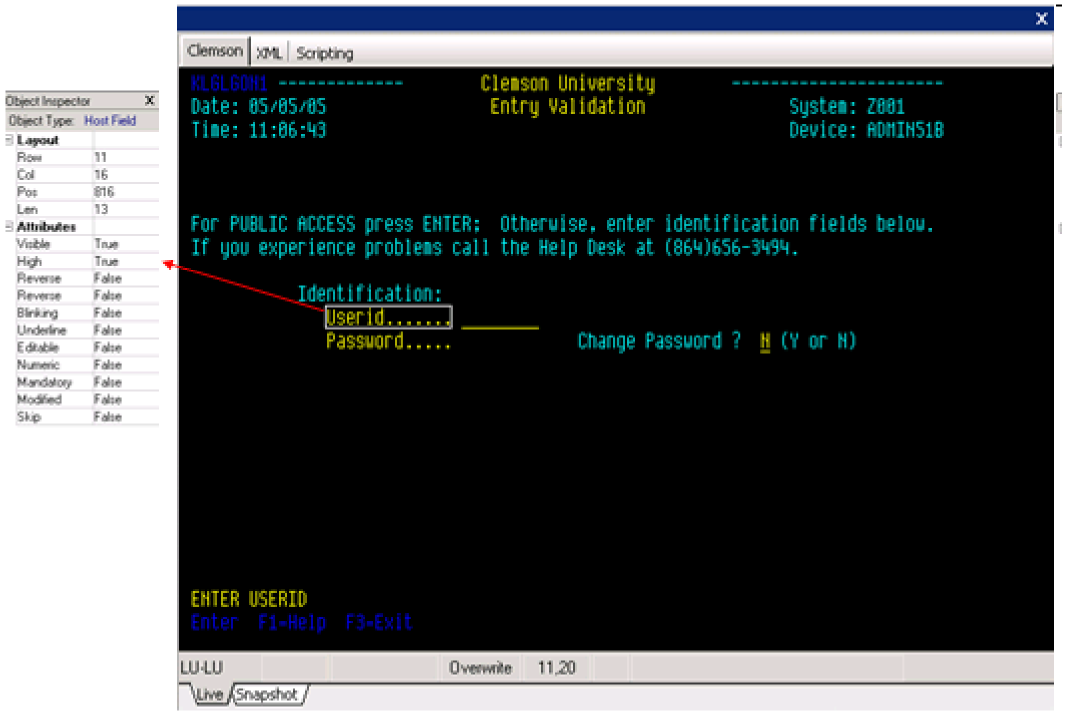
Event List

	Event	Time Stamp	Elapsed
	OnConnect	14:43:03,01...	240 ms.
	OnSessionState,...	14:43:04,19...	1182 ms.
	OnSessionState,...	14:43:04,20...	10 ms.
	OnSessionState,...	14:43:04,20...	0 ms.
	OnScreenChange	14:43:04,20...	0 ms.
	OnSystemLock	14:43:04,82...	621 ms.
	OnScreenChange	14:43:04,84...	0 ms.
	OnSystemUnlock	14:43:05,02...	180 ms.

2.1.4 The Object Inspector

The Object Inspector allows to inspect attirbutes of screen-fields and XML fields.

- To inspect the attributes of an specific field, double-click on it.



Object Type:	Host Field
Layout	
Row	11
Col	16
Pos	816
Len	13
Attributes	
Visible	True
High	True
Reverse	False
Reverse	False
Blinking	False
Underline	False
Editable	False
Numeric	False
Mandatory	False
Modified	False
Skip	False

```
KLGLGON1 -------------       Clemson University      ----------------------
Date: 05/05/05               Entry Validation             System: Z001
Time: 11:06:43                                            Device: ADMIN51B


For PUBLIC ACCESS press ENTER:  Otherwise, enter identification fields below.
If you experience problems call the Help Desk at (864)656-3494.

           Identification:
              Userid.......  ________
              Password.....             Change Password ?  N (Y or N)







ENTER USERID
Enter  F1-Help  F3-Exit
```

```
LU-LU                          Overwrite    11,20
```

See **XMLTemplates** for information about XML fields.

2.2 Working with TN BRIDGE Development Lab

In this chapter we'll see how to use the main functionalities provided by TN BRIDGE Development Lab.

- **Development Lab modes** details the modes of operation of Development Lab.
- **Scripting** explains how to create code snippets to test the host connection and play with TN BRIDGE objects.
- **Working with XML** explains how to use Development Lab to create XML templates.

2.2.1 Develoment Lab modes

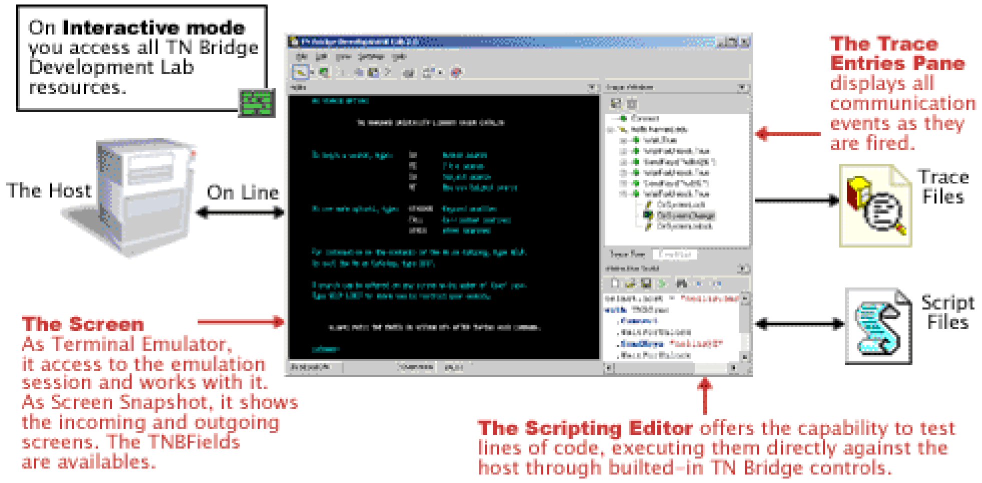
Main use of TN BRIDGE Development Lab is tracing TN BRIDGE-based applications. It provides three modes of operation: interactive, on-line trace and off-line trace.

- **Interactive** explains how to use is mode in which we can access to a mainframe using a built-in terminal emulator and watch generated events and fields.
- **On-Line Tracing** explains how to connect and trace TN BRIDGE applications.
- **Off-Line Tracing** explains how to open and read trace files generated by TN BRIDGE applications.

2.2.1.1 Interactive

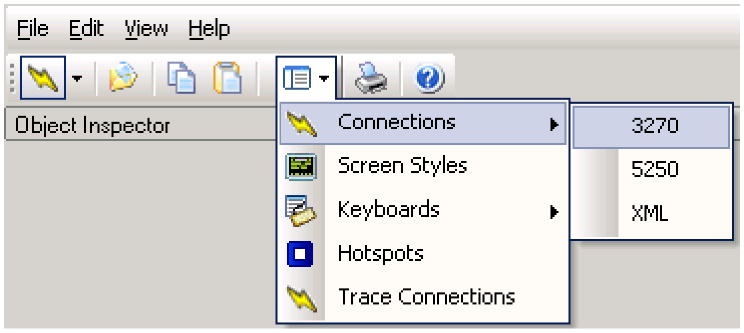
This mode provides a built-in terminal emulator which can be used to get connected to a Tn3270 or Tn5250 host. Addiionally, it allows to use XML files to simulate mainframe screens.

- **Creating a connection** shows how to create and configure a client-to-host connection
- **Connecting to the mainframe** explains how to connect to the host using the previously saved connection.
- **Using XML connections** explains how to create and get connected to host screens simulated by XML files.

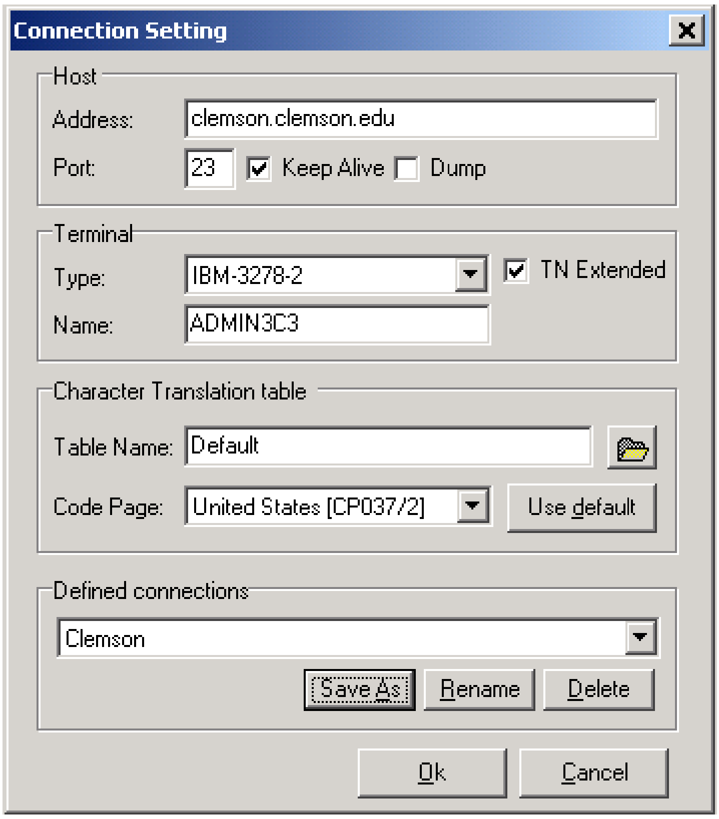

On **Interactive mode** you access all TN Bridge Development Lab resources.
The Host
On Line
The Trace Entries Pane displays all communication events as they are fired.
Trace Files
Script Files
The Screen As Terminal Emulator, it access to the emulation session and works with it. As Screen Snapshot, it shows the incoming and outgoing screens. The TNBFields are availables.
The Scripting Editor offers the capability to test lines of code, executing them directly against the host through builted-in TN Bridge controls.

2.2.1.1.1 Creating a connection

To define a new connection to a mainframe click on Settings/Connection/3270 or 5250 popup item.



The following dialog will appear:


Connection Setting

Host
Address: clemson.clemson.edu
Port: 23 [✓ Keep Alive] Dump

Terminal
Type: IBM-3278-2 [✓ TN Extended]
Name: ADMIN3C3

Character Translation table
Table Name: Default
Code Page: United States [CP037/2]

Defined connections
Clemson

After filling in the appropiate fields you must save the connection by clicking "Save As" button.

Parameters

Host Address
Host name (DNS) or IP address.

Host Port
Host connection port. Default value to standard telnet port (23).

Keep Alive
Enables keep-alive mechanism, needed for some telnet servers.

Dump
Enables generation of telnet communication logs.

Terminal Type
Sets the terminal type.

TN Extended
Enables Extended Telnet protocol (TN3270E or TN5250E).

Terminal Name
Logical unit or device name to be used in this connection.

Table Name
Load the character conversion table from an .ebc file.

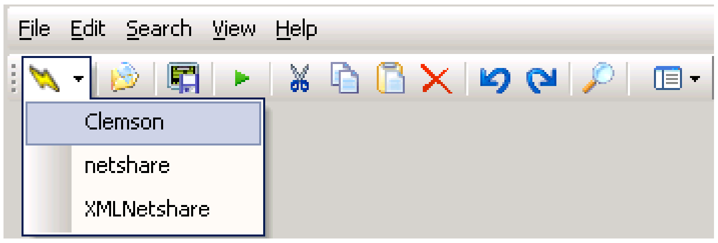
Code Page
Allows the selection of an internal character conversion table.

Use Default
Resets the Code Page parameter to its default setting.

Defined connections
Shows and allows to choose previously defined connections.

2.2.1.1.2 Connecting to the mainframe

Once you have defined one or more host connections you'll be able to get connected to one of these by clicking on the arrow of the connect button and selecting the connection.



See also Creating a connection

2.2.1.1.3 Using XML Connections

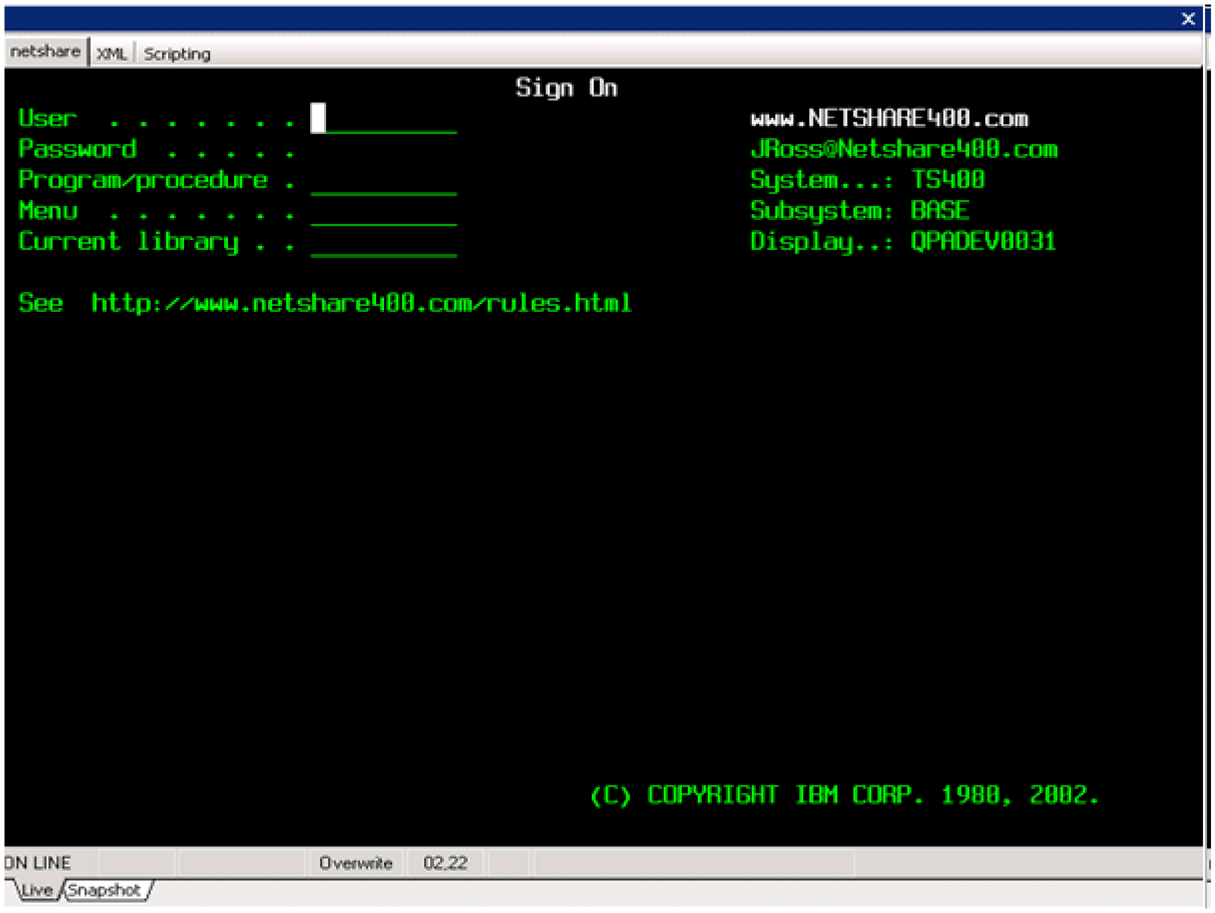
XML connections allows you work with mainframe screens in an off-line fashion. To work in this mode, you need to create the XML files corresponding to the mainframe application you want to work with.

- **Saving a screen as an XML file** shows how to save screens in files with XML format.
- **Connecting to a set of XML screen files** Loads a previously persisted screen from an XML file.

2.2.1.1.3.1 Saving screens as XML files

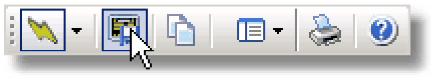
To save a screen in XML format you must do the following:

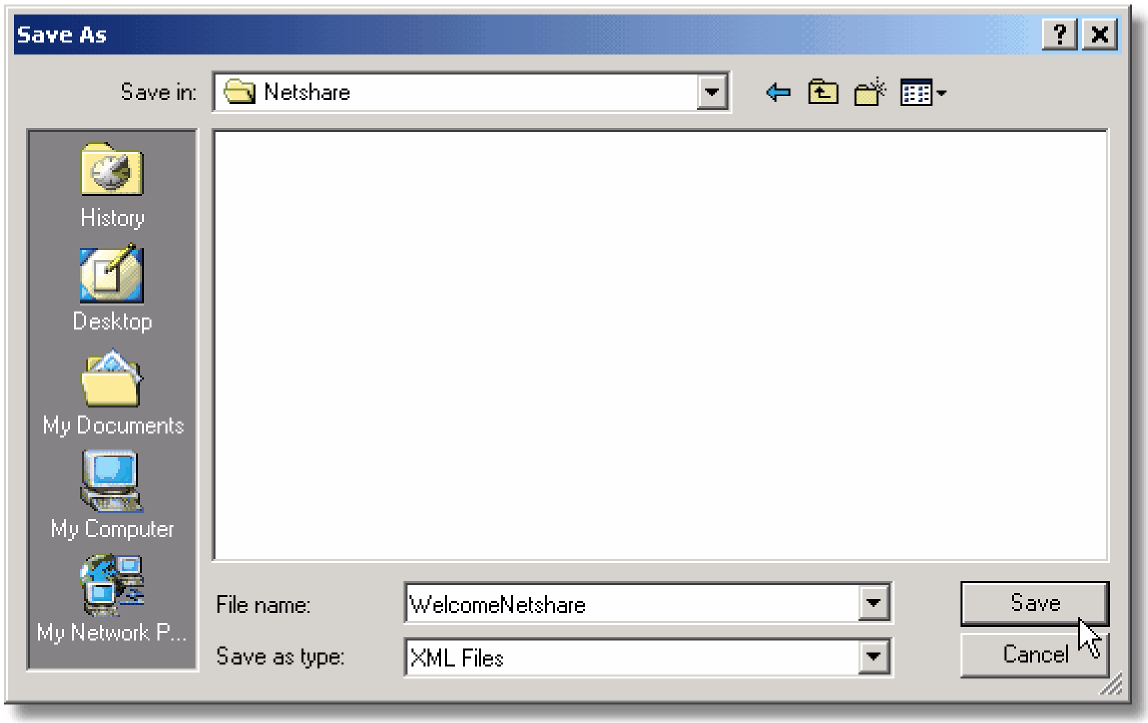
- Go to the mainframe screen you want to save



```
                                  Sign On
User  . . . . . . . . _________                     www.NETSHARE400.com
Password  . . . . . .                               JRoss@Netshare400.com
Program/procedure . . _________                     System...: TS400
Menu  . . . . . . . . _________                     Subsystem: BASE
Current library . . . _________                     Display..: QPADEV0031

See  http://www.netshare400.com/rules.html




                                        (C) COPYRIGHT IBM CORP. 1980, 2002.
```

- Click on Save Screen button





File name: WelcomeNetshare

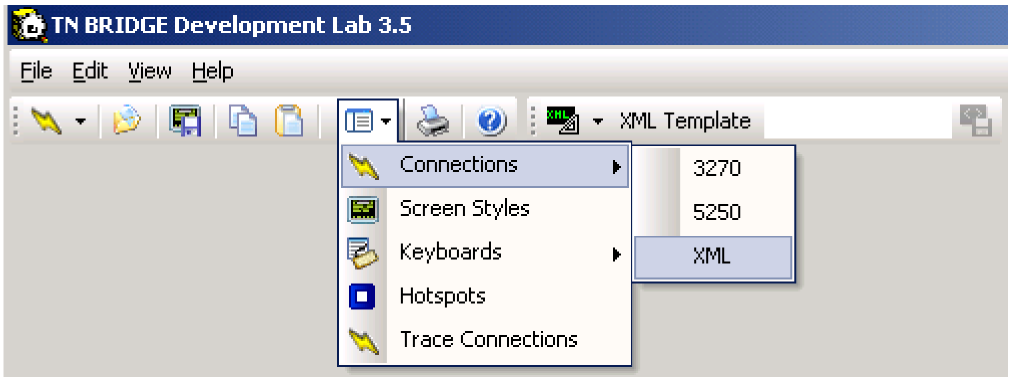
Save as type: XML Files

- Repeat the operation for all the screens you want to see off-line.

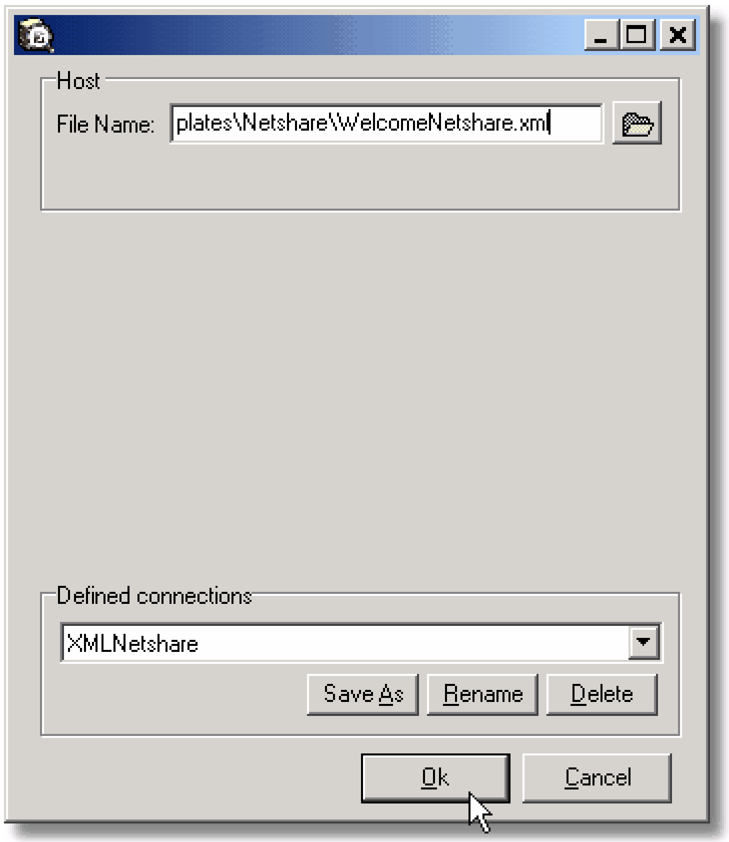
2.2.1.1.3.2 Connecting to a set of XML screen files

To connect to a set of XML-screen files, follow the steps below:

1. Go to XML Connection Preferences.



2. Select the starting XML Screen file name you want to work with and save the connection; in our case we have saved it with the name XMLNetsahre

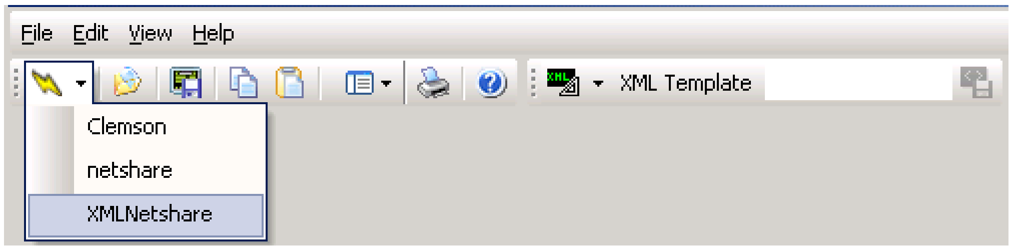

Host

File Name: plates\Netshare\WelcomeNetshare.xml

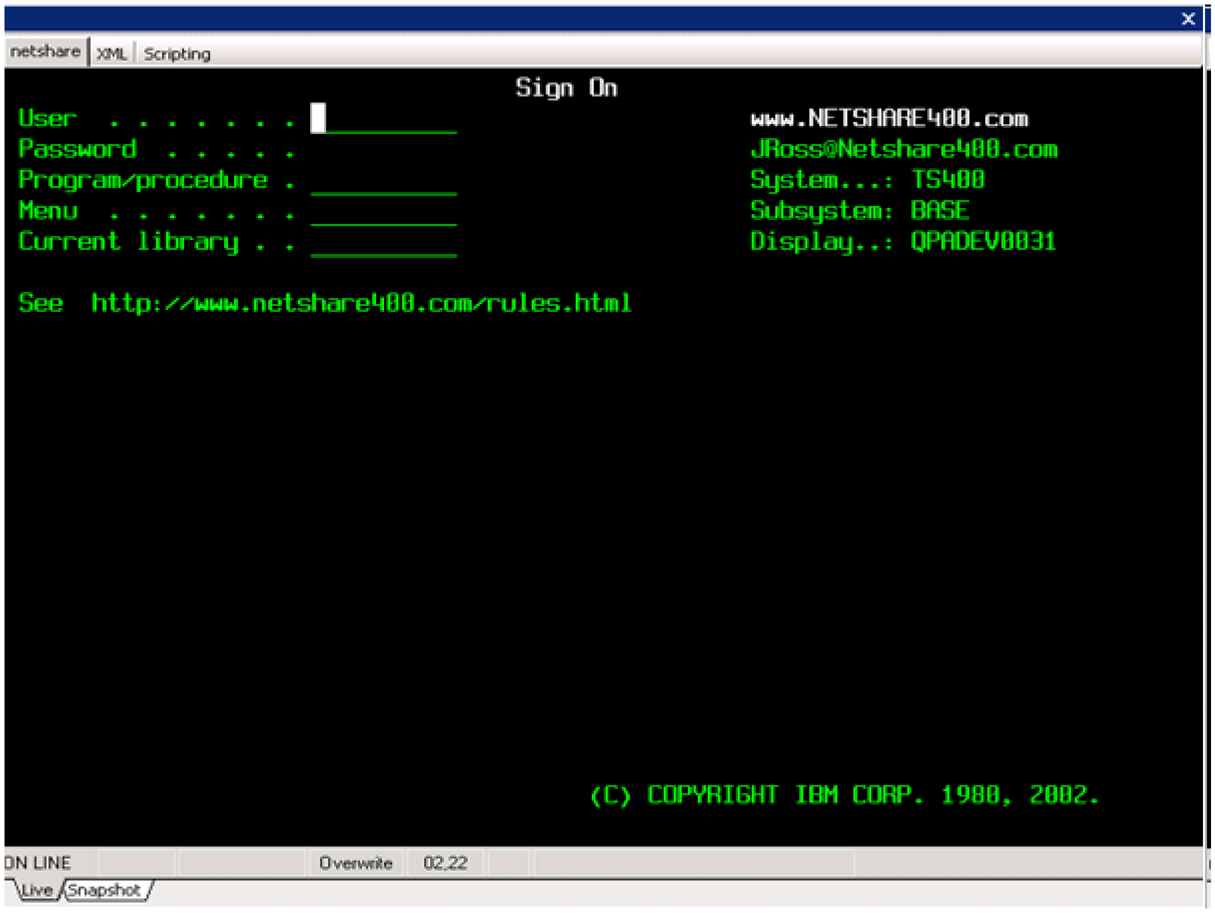
Defined connections

XMLNetshare

3. To connect, click on your defined XML connection.



4. If the screen is displayed, you are having a successful XML offline session.



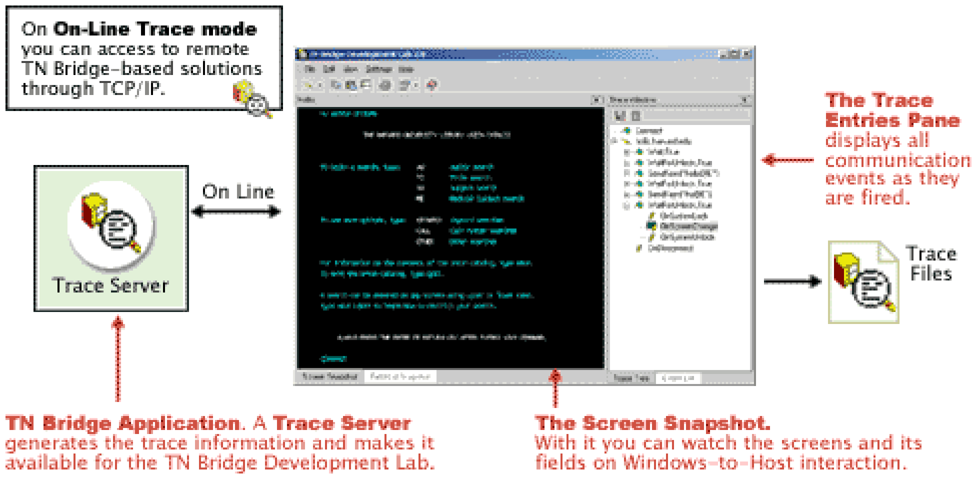
```
                              Sign On
User  . . . . . . . .                     www.NETSHARE400.com
Password  . . . . .                       JRoss@Netshare400.com
Program/procedure .                       System...: TS400
Menu  . . . . . . .                       Subsystem: BASE
Current library . .                       Display..: QPADEV0031

See  http://www.netshare400.com/rules.html

                                        (C) COPYRIGHT IBM CORP. 1980, 2002.
```

5. You can use PgDown/PgUp to move across the XML-screen files saved in the same directory.

2.2.1.2 On-Line tracing

On-Line Trace connection was created to access external applications using TN BRIDGE components.

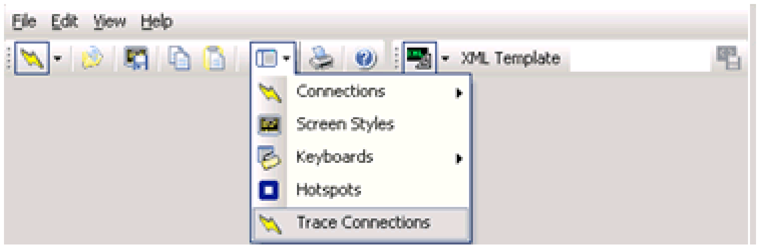
- **Connecting to a TN BRIDGE Application** Shows how to trace a TN BRIDGE application running in your computer or even on a computer within your LAN.


On **On-Line Trace mode** you can access to remote TN Bridge-based solutions through TCP/IP.
Trace Server
On Line
The Trace Entries Pane displays all communication events as they are fired.
Trace Files
TN Bridge Application. A **Trace Server** generates the trace information and makes it available for the TN Bridge Development Lab.
The Screen Snapshot. With it you can watch the screens and its fields on Windows-to-Host interaction.

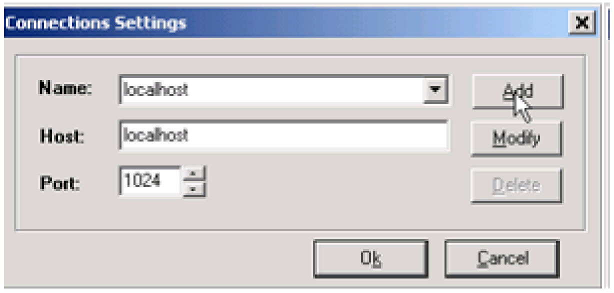
2.2.1.2.1 Connecting to a TN BRIDGE Application

To connect to a remote TN BRIDGE application, start TN BRIDGE Development Lab and follow these steps:

- Select Settings\Trace Connections item.



- Enter the required information and click on add to save the connection.


Connections Settings

Name: localhost

Host: localhost

Port: 1024

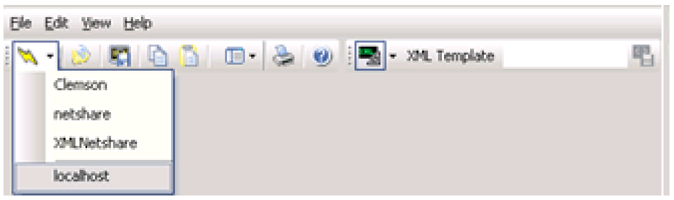
Parameters

Name
Descriptive name for the connection.

Host
IP address or DNS name of the computer where the external TN BRIDGE application is running.

Port
TCP port specified in TN BRIDGE's Trace component.

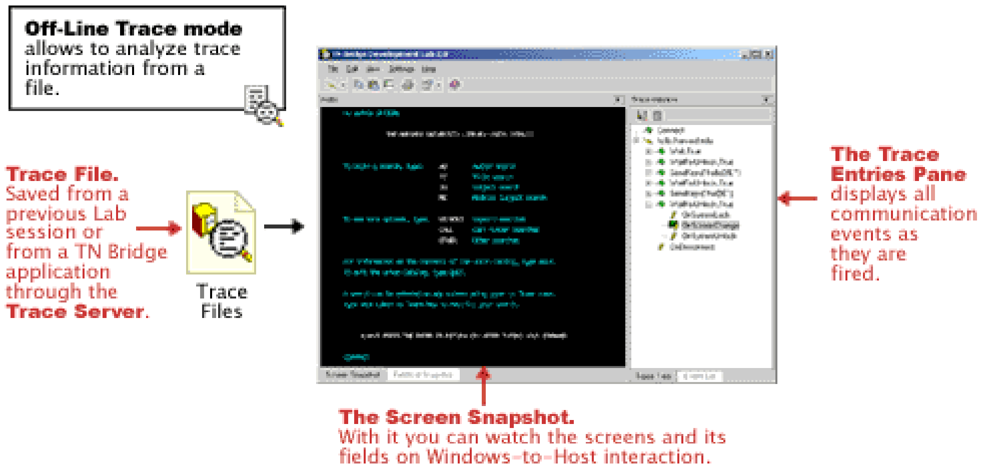
- You are now ready to connect to a remote TN BRIDGE Application



2.2.1.3 Off-Line tracing

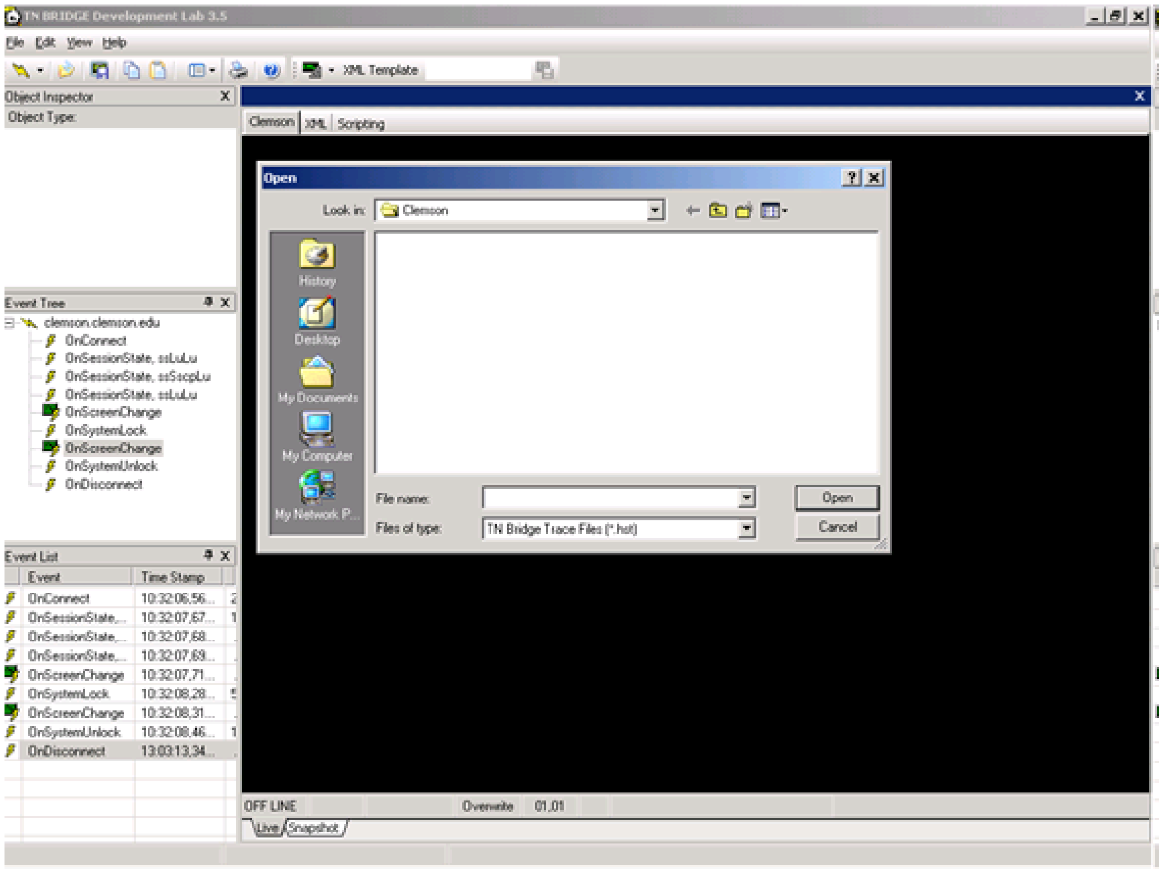
TN BRIDGE Development Lab allows you open trace files.

- **Opening a Trace File** Loads a trace file


Off-Line Trace mode allows to analyze trace information from a file.
Trace File. Saved from a previous Lab session or from a TN Bridge application through the **Trace Server.**
Trace Files
The Trace Entries Pane displays all communication events as they are fired.
The Screen Snapshot. With it you can watch the screens and its fields on Windows-to-Host interaction.

2.2.1.3.1 Opening a Trace File

To open a previously saved trace file, select Open from the File Menu.



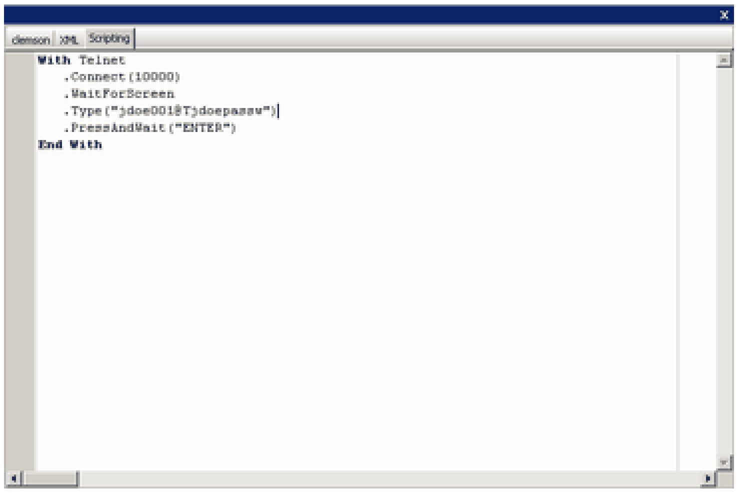
Event	Time Stamp
OnConnect	10:32:06,56...
OnSessionState,...	10:32:07,67...
OnSessionState,...	10:32:07,68...
OnSessionState,...	10:32:07,69...
OnScreenChange	10:32:07,71...
OnSystemLock	10:32:08,28...
OnScreenChange	10:32:08,31...
OnSystemUnlock	10:32:08,46...
OnDisconnect	13:03:13,34...

2.2.2 Scripting

Development Lab's main use is understanding the event flow and mainframe application's logic. Scripting inside Development Lab allows you to write code to test the mainframe logic and event flow from TN BRIDGE programming point of view. Also it can help you create code snippets to be included in your application.

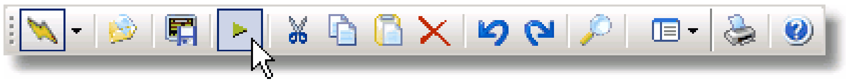
Code can be written in Microsoft VB Script language, accessing methods and properties of Tn3270/Tn5250 components. A global object "Telnet" is available, associated to the Tn3270 or Tn5250 currently in use in the interactive emulation.

As an example, you can write the following code access a mainframe application:

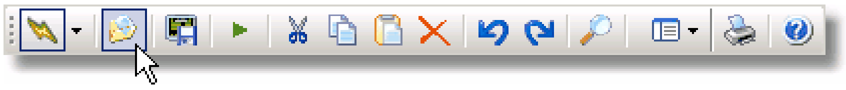


```
With Telnet
    .Connect(10000)
    .WaitForScreen
    .Type("jdoe001@Tjdoepassw")
    .PressAndWait("ENTER")
End With
```

To run the script click on Play button on the common toolbar.



To load a saved click on Open button of the common toolbar.



2.2.3 Working with XML

XML provides the facility to define tags and the structural relationship between them. As a result, developers can create their own customized tags (the extensible part of the puzzle) in order to define, share, and validate information between computing systems and applications.

Host Integration Pack allows you to build an XML bridge between your application and the mainframe.

- **Generic XML** explains the generic XML that TN BRIDGE provides by mapping screen-fields.
- **XML Templates** explains how we can create XML Templates top provide customized XML tags an attributes.

2.2.3.1 Generic XML

By default, screen information is mapped to XML tags and attributes as described:

```
<TNBRIDGE>
    <status>
        <direction... />
        <transaction... />
        <cursor_pos... />
        <session... />
        <state... />
        <xmlscreen... />
        <cursor_field... />
        <timestamp... />
    <status />
    <screen>
        <rows... />
        <cols... />
        <type... />
        <model... />
        <fields>
            <field ...>
                <name... />
                <attr... />
                <value... />
            <field />
            .
            .
            .
        <fields />
    <screen />
<TNBRIDGE />
```

TNBRIDGE is the root tag and includes the status and screen tags as children.

status: indicates the connection status
Contains the following tags:

- **direction:** indicates INPUT or OUTPUT direction. When the XML is generated by the XmlBroker control it is set with OUTPUT value.
- **transaction:** indicates transaction identifier.
- **cursor_pos:** indicates cursor location.
- **session:** LU-LU (in session with VTAM application), noSession (not in session) or SSCPLU (in session with VTAM).
- **state:** indicates LOCKED or UNLOCKED state.
- **xmlscreen:** indicates the screen name. You must set this propery using the ScreenName property.
- **cursor_field:** indicates the field's name where the cursor is located.
- **timestamp:** indicates the date the file was created.

Screen: indicates screen's characteristics and contains a collection of screen fields.
Contains the following tags:

- **rows:** indicates the number of rows of the screen.
- **cols:** indicates the number of columns of the screen.
- **type:** indicates terminal type (3270 or 5250).
- **model:** indicates terminal model.
- **fields:** contains a collection of fields.

Here you can see an example of XML Code:

```
  <TNBRIDGE>
       <status>
              <direction>OUTPUT</direction>
              <transaction>00A8DB7000002</transaction>
              <cursor_pos>830</cursor_pos>
              <session>LU-LU</session>
              <state>UNLOCKED</state>
              <xmlscreen name="Session" />
              <cursor_field>R11C30</cursor_field>
              <timestamp>2453360.14494451</timestamp>
       </status>
       <screen>
              <rows>24</rows>
              <cols>80</cols>
              <type>TN3270</type>
              <model>2E</model>
              <fields>
                     <field name="R01C02">
                            <name len="4">R1C2</name>
                            <attr byte="E4" eab="08" display="y" prot="y" numeric="n"
  intense="n" mdt="n" />
                            <value maxlen="8" len="8">KLGLGON1</value>
                     </field>
                     <field name="R01C11">
                            <name len="5">R1C11</name>
                            <attr byte="E4" eab="28" display="y" prot="y" numeric="n"
  intense="n" mdt="n" />
                            <value maxlen="13" len="13">-------------</value>
                     </field>
                     .
                     .
                     .
              </fields>
       </screen>
</TNBRIDGE>
```

2.2.3.2 XML Templates

What is an XML Template?

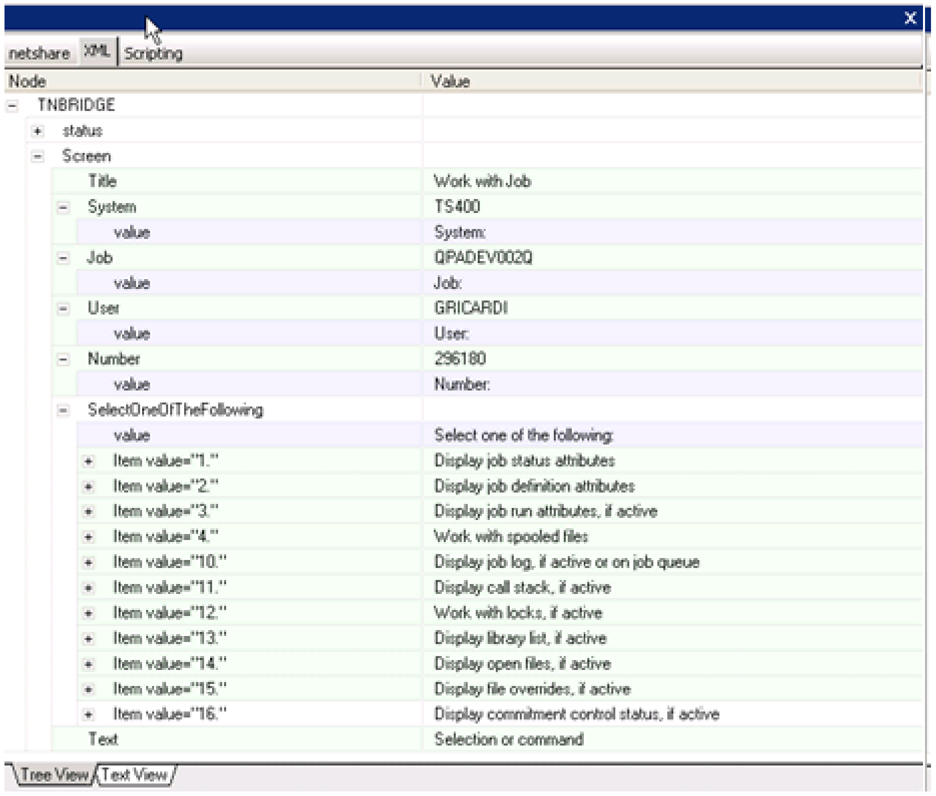
By default, Host Integration Pack provides a **generic XML** representation. Because this generic XML contains complete screen information it is useful to recreate a screens on the comsumer side. However, when it comes to real business information, it doesn't provides a real advantage regarding to accessing screen data directly.

Using Develpement Lab you have the possibility to generate XML Templates containing the necessary information to create tags and attributes representing the screen information as you perceive it, rather than using protocol data.
This result in a much more legible and understandable XML file.

In this chapter we will analyze:

- **Enabling Template Mode** explains how to activate XML Template's functionalities.
- **Building an XML Template** shows how to persist data fields to an XML Template file.

Example XML generated file



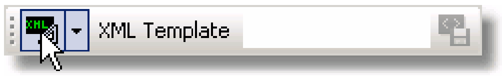
Node	Value
TNBRIDGE	
status	
Screen	
Title	Work with Job
System	TS400
value	System:
Job	QPADEV002Q
value	Job:
User	GRICARDI
value	User:
Number	296180
value	Number:
SelectOneOfTheFollowing	
value	Select one of the following:
Item value="1."	Display job status attributes
Item value="2."	Display job definition attributes
Item value="3."	Display job run attributes, if active
Item value="4."	Work with spooled files
Item value="10."	Display job log, if active or on job queue
Item value="11."	Display call stack, if active
Item value="12."	Work with locks, if active
Item value="13."	Display library list, if active
Item value="14."	Display open files, if active
Item value="15."	Display file overrides, if active
Item value="16."	Display commitment control status, if active
Text	Selection or command

You have now generated a nice readable XML Template file.

2.2.3.2.1 Enabling Template Mode

In order to work with XML Template files, you must first enable the XML Template mode.

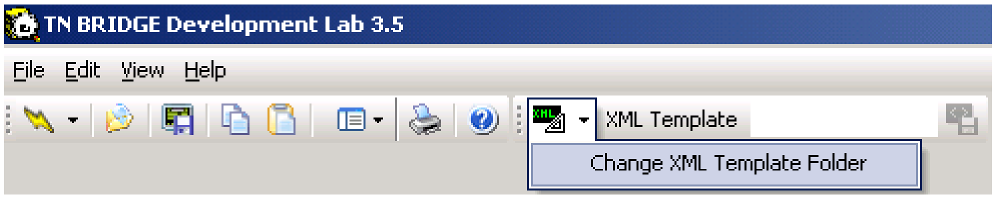
- Click on the icon as shown below.



A window will ask you to specify a search path. The application will look for matching XML Templates for the current screen in the specified folder.

Keep in mind that XML Template files for a specified connection must be saved on the same directory, so they can be automatically recognized as the screen changes.

- To change the folder path do the following:



2.2.3.2.2 Building XML Templates

Building XML Templates involves recognizing and telling Development Lab where the data is located as well as its attributes. In some way, we need to define what data should be considered variable and what fixed text. This can be done both automatically and manually.

Automatic recognizing is the best option to start defining a XML Template, because it will do the larger and tediuos work. However it's far beyond perfect, so you will need to make manual changes to attributes and sometimes manually redefine XML template data.

A XML Template is composed by XML Template items. Each XML Template item has an screen-area assigned as well as attributes that specifies how this item should behave at rutime.
Currently, XML Template items are divided into XML Template Fields, XML Template Tables and custom XML Template Areas.

- **Generating XML Fields** shows you how to automatically create XML template fields.
- **Using the Object Inspector** shows you how use the object inspector to modify

attributes of XML fields.

- **Identifying the screen** explain the importance of some fields in the XML template.
- **Deleting XML Template Fields** shows you how to delete XML template fields.
- **Saving an XML Template** explain how to save the XML template.
- **Adding XML Template Fields** shows you how to add a single XML Template field.
- **Adding XML Template Tables** shows you how to define a table.
- **Add custom XML Template Areas** shows you how create user-defined areas to populate at runtime.

2.2.3.2.2.1 Generating XML Template Fields

Using this option you can automatically generate XML-Template Fields inside of a selected area. An XML Template Field is an area that can have the following attributes:

- Label
- Text
- Label & Text

Text fields are those areas considered as they contain variable information and doesnt carry by themselves information about what type of information they contain. This fields have Label fields associated.
Label fields are considered those that gives information about what a Text field means.
Label & Text are considered those that carries both inherent information of the data if contains and variable data.

For example, in:

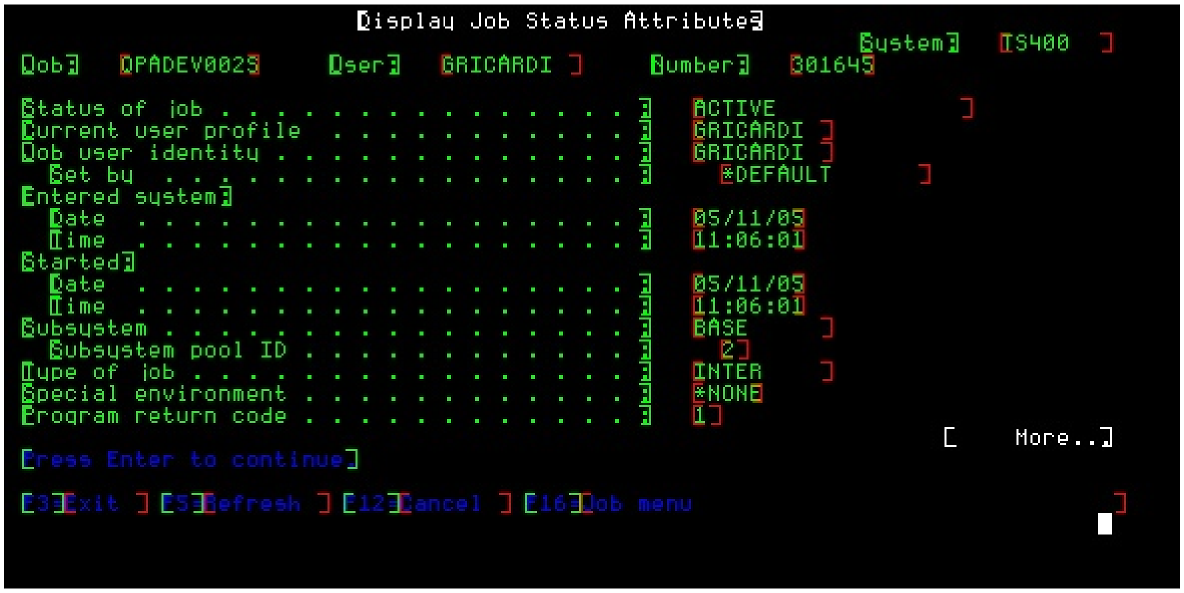
UserId: jdoe01

"UserId" is considered as a Label field an "jdoe01" as Text field, whose Label field is "UserId"
In image below, "User Taks", which is the title of the page, is considered Label & Text field, with label "Title".

In order to generate XML Template Fields select an area which contains the fields you want to add to the XML Template.

The resulting template fields are shown:

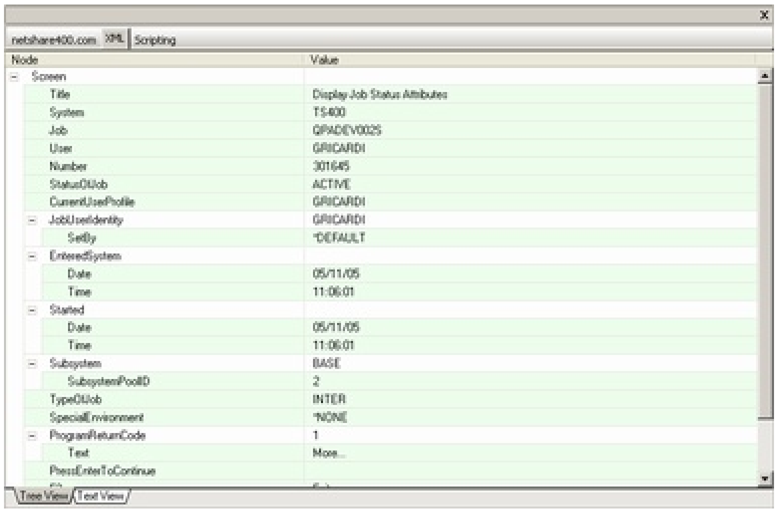
- Label fields enclosed in green brackets
- Text fields enclosed in red brackets
- Label & Text fields enclosed in white brackets



```
                          Display Job Status Attributes
                                                          System:   TS400
Job:    QPADEV0025      User:    GRICARDI        Number:    301645

Status of job . . . . . . . . . . . . . . . :   ACTIVE
Current user profile  . . . . . . . . . . . :   GRICARDI
Job user identity . . . . . . . . . . . . . :   GRICARDI
  Set by  . . . . . . . . . . . . . . . . . :     *DEFAULT
Entered system:
  Date  . . . . . . . . . . . . . . . . . . :   05/11/05
  Time  . . . . . . . . . . . . . . . . . . :   11:06:01
Started:
  Date  . . . . . . . . . . . . . . . . . . :   05/11/05
  Time  . . . . . . . . . . . . . . . . . . :   11:06:01
Subsystem . . . . . . . . . . . . . . . . . :   BASE
  Subsystem pool ID . . . . . . . . . . . . :     2
Type of job . . . . . . . . . . . . . . . . :   INTER
Special environment . . . . . . . . . . . . :   *NONE
Program return code . . . . . . . . . . . . :   1
                                                                     More...
Press Enter to continue.

F3=Exit   F5=Refresh   F12=Cancel   F16=Job menu
```

Note: You can now use the ***Object Inspector*** *to customize the field you need.*

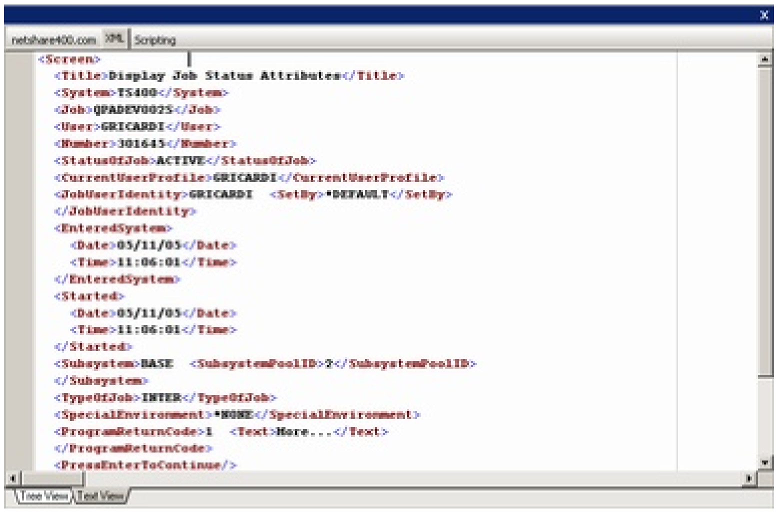
Switching to XML view we get:



netshare400.com | XML | Scripting

Node	Value
Screen	
Title	Display Job Status Attributes
System	TS400
Job	QPADEV0025
User	GRICARDI
Number	301645
StatusOfJob	ACTIVE
CurrentUserProfile	GRICARDI
JobUserIdentity	GRICARDI
SetBy	*DEFAULT
EnteredSystem	
Date	05/11/05
Time	11:06:01
Started	
Date	05/11/05
Time	11:06:01
Subsystem	BASE
SubsystemPoolID	2
TypeOfJob	INTER
SpecialEnvironment	*NONE
ProgramReturnCode	1
Text	More...
PressEnterToContinue	

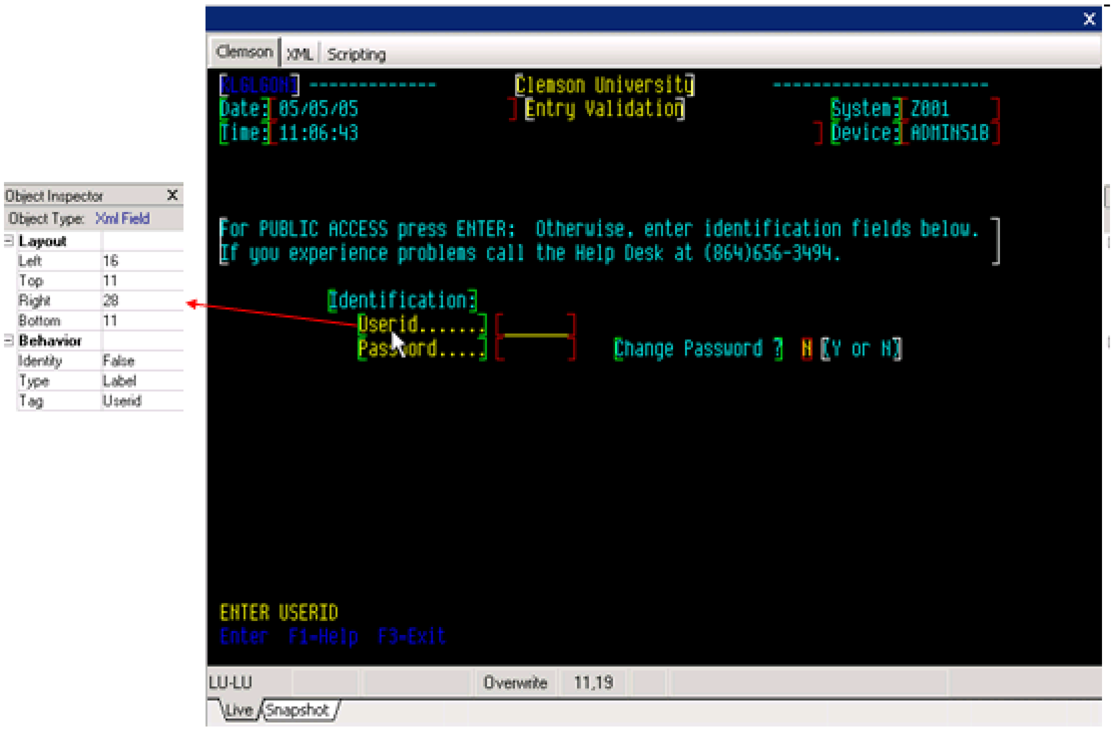
Tree View | Text View



```
<Screen>
  <Title>Display Job Status Attributes</Title>
  <System>TS400</System>
  <Job>QPADEV0025</Job>
  <User>GRICARDI</User>
  <Number>301645</Number>
  <StatusOfJob>ACTIVE</StatusOfJob>
  <CurrentUserProfile>GRICARDI</CurrentUserProfile>
  <JobUserIdentity>GRICARDI  <SetBy>*DEFAULT</SetBy>
  </JobUserIdentity>
  <EnteredSystem>
    <Date>05/11/05</Date>
    <Time>11:06:01</Time>
  </EnteredSystem>
  <Started>
    <Date>05/11/05</Date>
    <Time>11:06:01</Time>
  </Started>
  <Subsystem>BASE  <SubsystemPoolID>2</SubsystemPoolID>
  </Subsystem>
  <TypeOfJob>INTER</TypeOfJob>
  <SpecialEnvironment>*NONE</SpecialEnvironment>
  <ProgramReturnCode>1  <Text>More...</Text>
  </ProgramReturnCode>
  <PressEnterToContinue/>
```

2.2.3.2.2.2 Using Object Inspector

The Object Inspector allows you to customize the fields which are going to be inserted in the XML Template.



Object Inspector
Object Type: Xml Field

Layout	
Left	16
Top	11
Right	28
Bottom	11
Behavior	
Identity	False
Type	Label
Tag	Userid

```
XLGLGON  -------------          Clemson University         ----------------------
Date: 05/05/05                  Entry Validation                  System: Z001
Time: 11:06:43                                                    Device: ADMIN51B


For PUBLIC ACCESS press ENTER:  Otherwise, enter identification fields below.
If you experience problems call the Help Desk at (864)656-3494.

              Identification:
                 Userid.......
                 Password.....             Change Password ?  N (Y or N)





ENTER USERID
Enter  F1=Help  F3=Exit
```

Parameters

Identity
Determines if the field will be verified to match the current screen in order to assign the correct XML Template.

Note: The XML Template will be assigned to the screen **only** if **all** the identities with true value match.

Type
Defines the control's type.

Tag
Defines the XML tag the field will create.

2.2.3.2.2.3 Identifying the screen

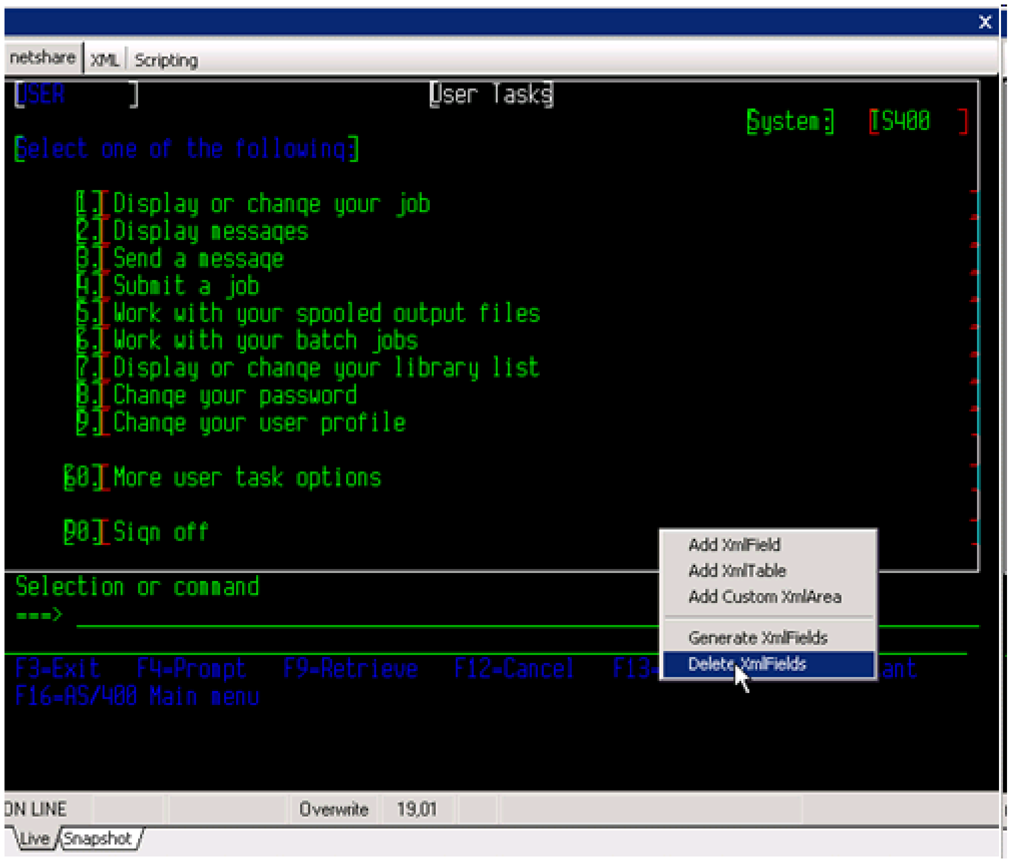
Once you have created and saved a couple of XML Template files you will notice that when you retrieve from host a screen that already has a XML Template that matchs, it is automatically selected and shown in the display. When a screen arrives, the XML Template engine looks in the template's directory for the template that own those XML Template Fields that have the "Identity" attribute set to true and match in the current screen.

When generating XML Template Fields in automatic way, these fields are in general determined by Development Lab. However as we said, this is far beyond perfect, so you might need to modify this attribute on some fields.
Always take into account that "Identity" fields must be those that are unique to the screen and wont suffer any change neither in its content nor its location.

2.2.3.2.2.4 Deleting Template Fields

XML Template Fields can be deleted in two ways:

- right clicking on the field and selecting Delete XmlField
- selecting an screen area, right clicking and selecting Delete XmlFields.

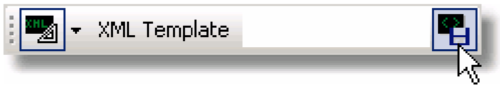


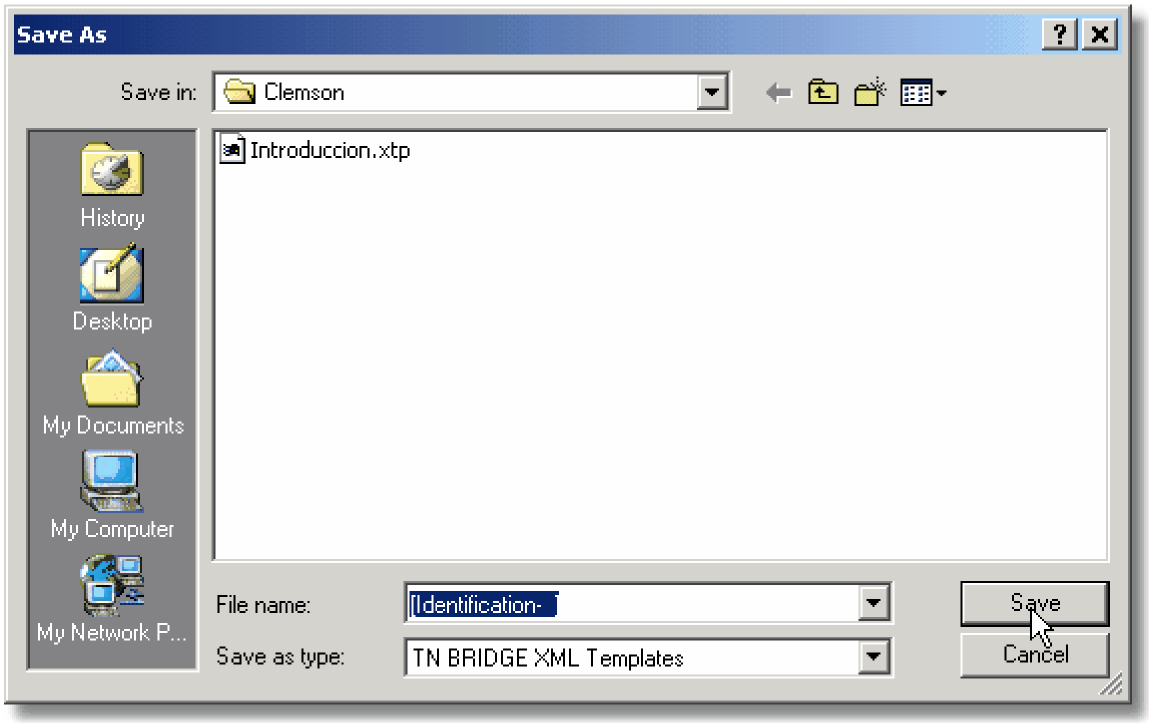
```
[USER      ]                        [User Tasks]
                                                          [System:]  [TS400  ]
[Select one of the following:]

     [1.] Display or change your job
     [2.] Display messages
     [3.] Send a message
     [4.] Submit a job
     [5.] Work with your spooled output files
     [6.] Work with your batch jobs
     [7.] Display or change your library list
     [8.] Change your password
     [9.] Change your user profile

    [60.] More user task options

    [90.] Sign off

Selection or command
===> ________________________________________________________________

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13-                    ant
F16=AS/400 Main menu
```

2.2.3.2.2.5 Saving an XML Template

To save a Template in order to reuse it in future sessions do the following:

- To save your XML Template click on the save button and select the destination folder.

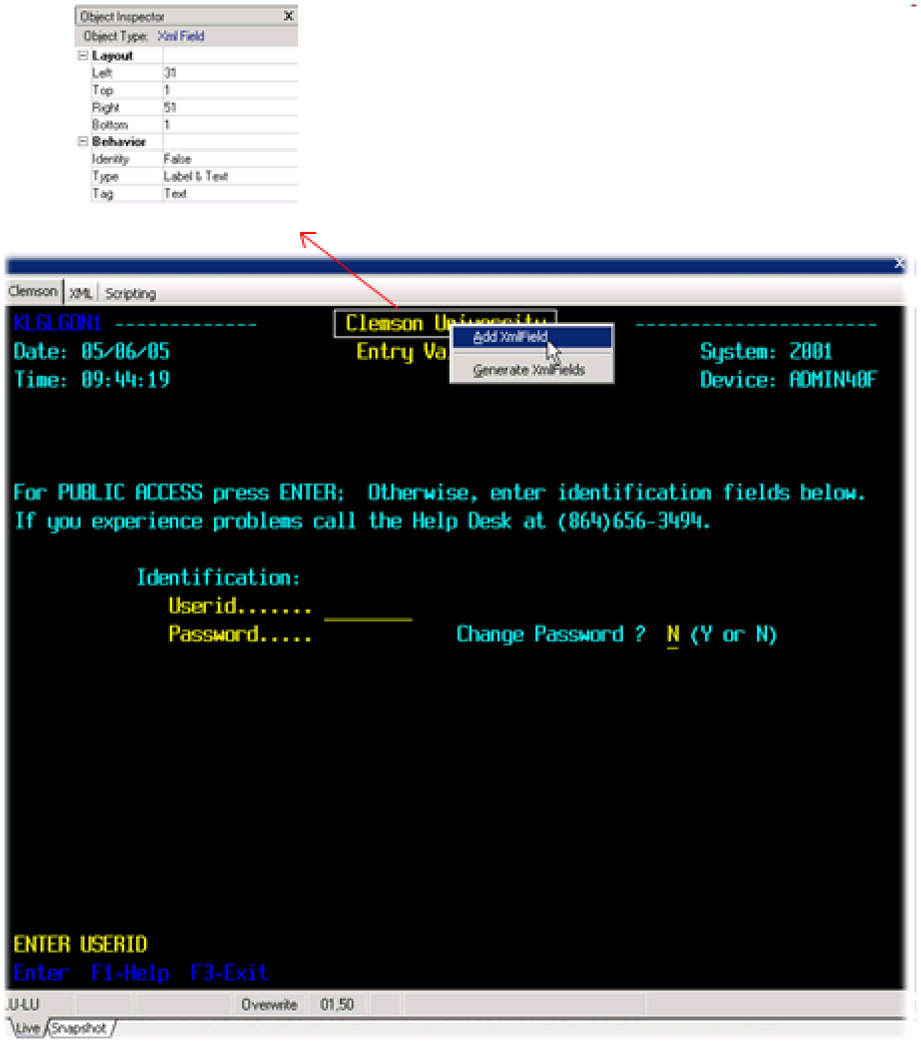




Note: Sometimes the save icon can be disabled, this is because the XML Template is empty. You must add some field before you can save it.

2.2.3.2.2.6 Adding XML Template Field

To add XML Template fields manually by selecting an area, right-clicking and selecting Add XmlField. Then click on the added field and customize it using the **Object Inspector**



Object Type: Xml Field

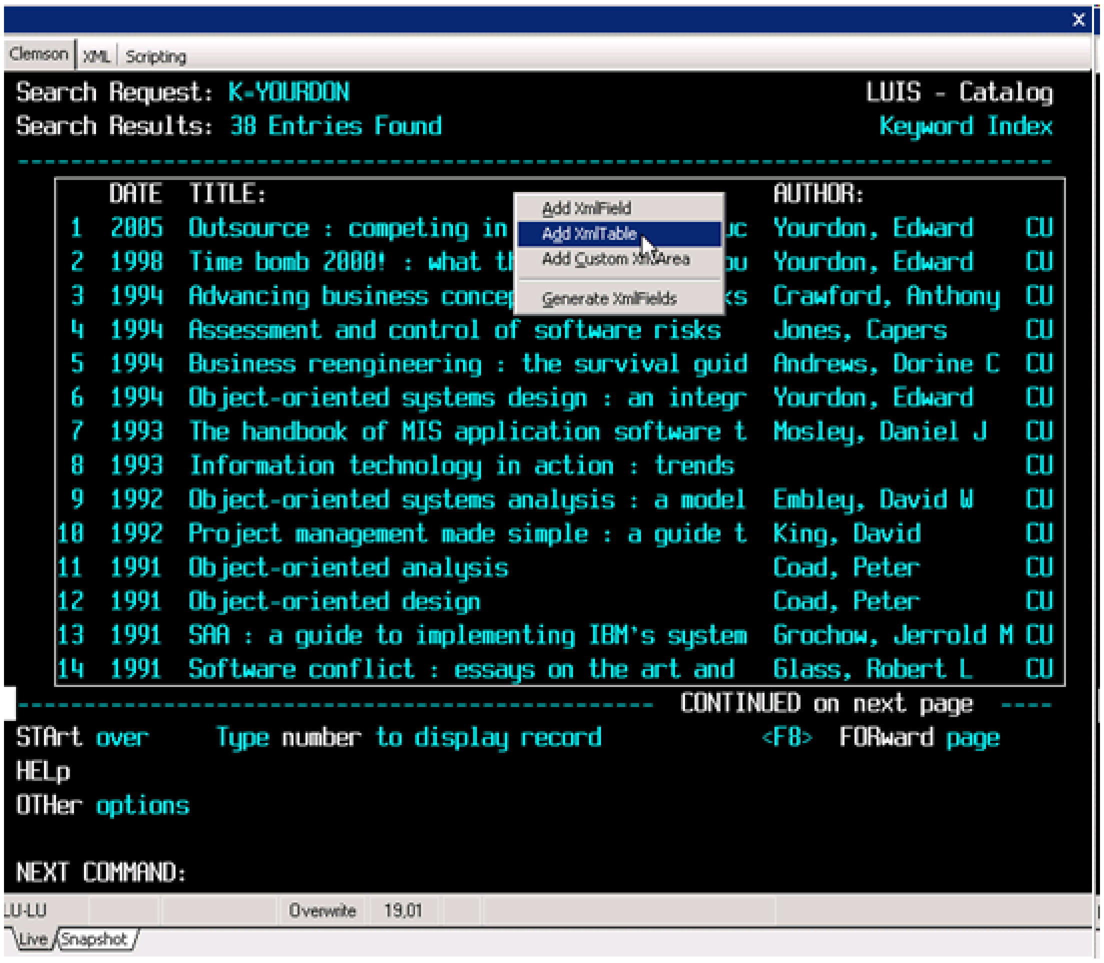
Layout	
Left	31
Top	1
Right	51
Bottom	1
Behavior	
Identity	False
Type	Label & Text
Tag	Text

```
KLGLGON1 -------------     Clemson University     ----------------------
Date: 05/06/05               Entry Va                    System: 2001
Time: 09:44:19                                           Device: ADMIN40F


For PUBLIC ACCESS press ENTER;  Otherwise, enter identification fields below.
If you experience problems call the Help Desk at (864)656-3494.

           Identification:
              Userid....... ________
              Password.....             Change Password ?  N (Y or N)








ENTER USERID
Enter  F1-Help  F3-Exit
```

2.2.3.2.2.7 Adding XML Template Table

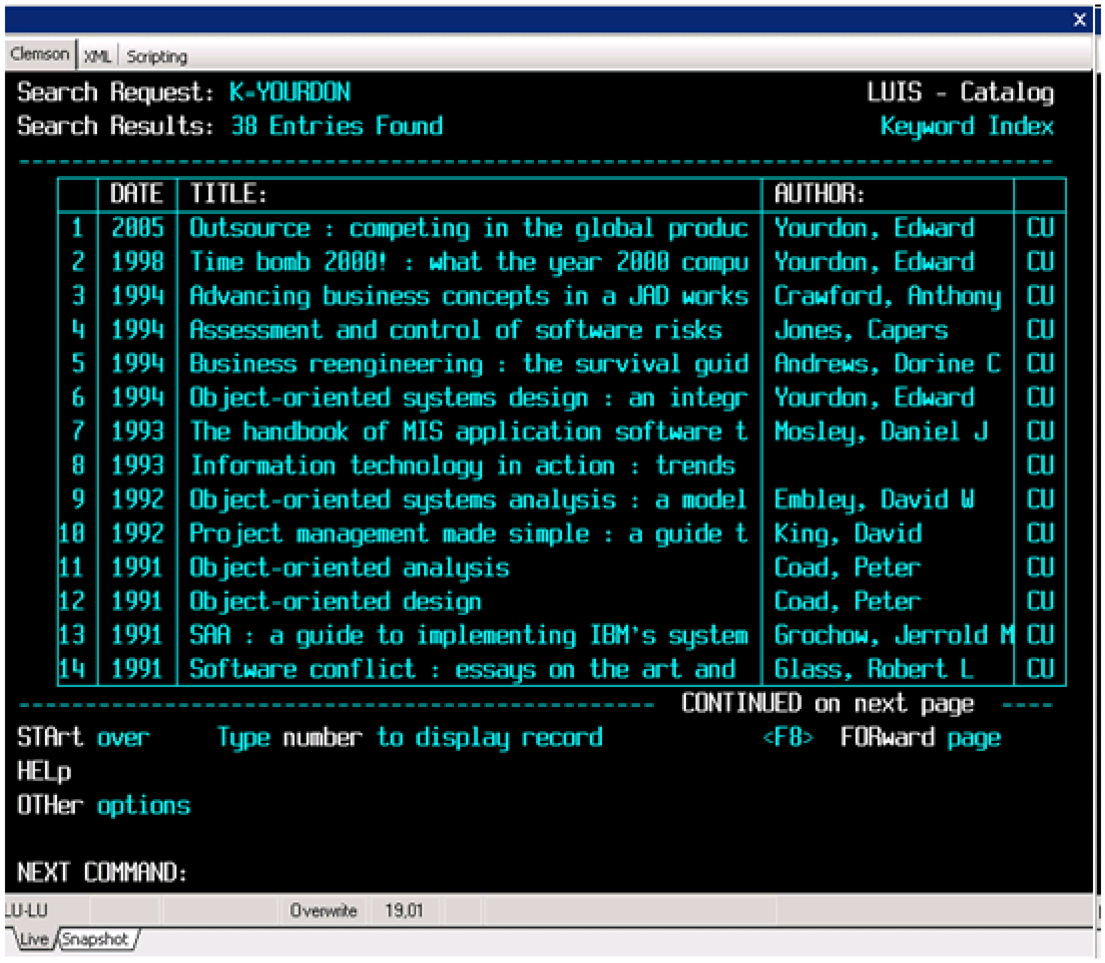
You can also create XML Template Tables.

- Select the area on which you want to create the table and right click on it.



```
Search Request: K-YOURDON                                    LUIS - Catalog
Search Results: 38 Entries Found                            Keyword Index
----------------------------------------------------------------------------
      DATE  TITLE:                                    AUTHOR:
   1  2005  Outsource : competing in              uc  Yourdon, Edward     CU
   2  1998  Time bomb 2000! : what th             ou  Yourdon, Edward     CU
   3  1994  Advancing business concep             ks  Crawford, Anthony   CU
   4  1994  Assessment and control of software risks  Jones, Capers       CU
   5  1994  Business reengineering : the survival guid Andrews, Dorine C  CU
   6  1994  Object-oriented systems design : an integr Yourdon, Edward    CU
   7  1993  The handbook of MIS application software t Mosley, Daniel J   CU
   8  1993  Information technology in action : trends                     CU
   9  1992  Object-oriented systems analysis : a model Embley, David W    CU
  10  1992  Project management made simple : a guide t King, David        CU
  11  1991  Object-oriented analysis                   Coad, Peter        CU
  12  1991  Object-oriented design                     Coad, Peter        CU
  13  1991  SAA : a guide to implementing IBM's system Grochow, Jerrold M CU
  14  1991  Software conflict : essays on the art and  Glass, Robert L    CU
----------------------------------------------  CONTINUED on next page ----
STArt over     Type number to display record          <F8>  FORward page
HELp
OTHer options

NEXT COMMAND:
```

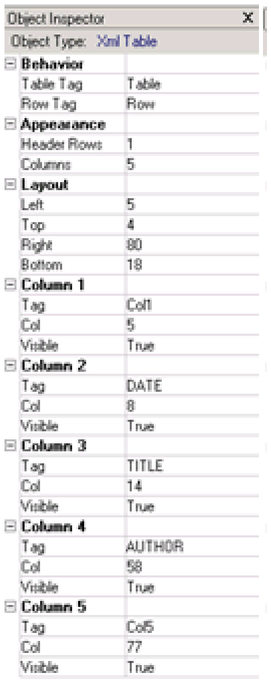
• The generated table will look like this.



```
Search Request: K-YOURDON                                    LUIS - Catalog
Search Results: 38 Entries Found                              Keyword Index
---------------------------------------------------------------------------
   |  | DATE | TITLE:                                   | AUTHOR:          |   |
   | 1| 2005 | Outsource : competing in the global produc| Yourdon, Edward  | CU|
   | 2| 1998 | Time bomb 2000! : what the year 2000 compu| Yourdon, Edward  | CU|
   | 3| 1994 | Advancing business concepts in a JAD works| Crawford, Anthony| CU|
   | 4| 1994 | Assessment and control of software risks  | Jones, Capers    | CU|
   | 5| 1994 | Business reengineering : the survival guid| Andrews, Dorine C| CU|
   | 6| 1994 | Object-oriented systems design : an integr| Yourdon, Edward  | CU|
   | 7| 1993 | The handbook of MIS application software t| Mosley, Daniel J | CU|
   | 8| 1993 | Information technology in action : trends |                  | CU|
   | 9| 1992 | Object-oriented systems analysis : a model| Embley, David W  | CU|
   |10| 1992 | Project management made simple : a guide t| King, David      | CU|
   |11| 1991 | Object-oriented analysis                  | Coad, Peter      | CU|
   |12| 1991 | Object-oriented design                    | Coad, Peter      | CU|
   |13| 1991 | SAA : a guide to implementing IBM's system| Grochow, Jerrold M| CU|
   |14| 1991 | Software conflict : essays on the art and | Glass, Robert L  | CU|
----------------------------------------------- CONTINUED on next page ----
STArt over     Type number to display record          <F8>  FORward page
HELp
OTHer options

NEXT COMMAND:
```

Note: The number of columns are detected automatically. If the result is not what you expected, you can customize them to your wish.

- Left click on the table to customize it with the **Object Inspector**.



Object Inspector

Object Type: Xml Table

Behavior	
Table Tag	Table
Row Tag	Row
Appearance	
Header Rows	1
Columns	5
Layout	
Left	5
Top	4
Right	80
Bottom	18
Column 1	
Tag	Col1
Col	5
Visible	True
Column 2	
Tag	DATE
Col	8
Visible	True
Column 3	
Tag	TITLE
Col	14
Visible	True
Column 4	
Tag	AUTHOR
Col	58
Visible	True
Column 5	
Tag	Col5
Col	77
Visible	True

Deleting the Table

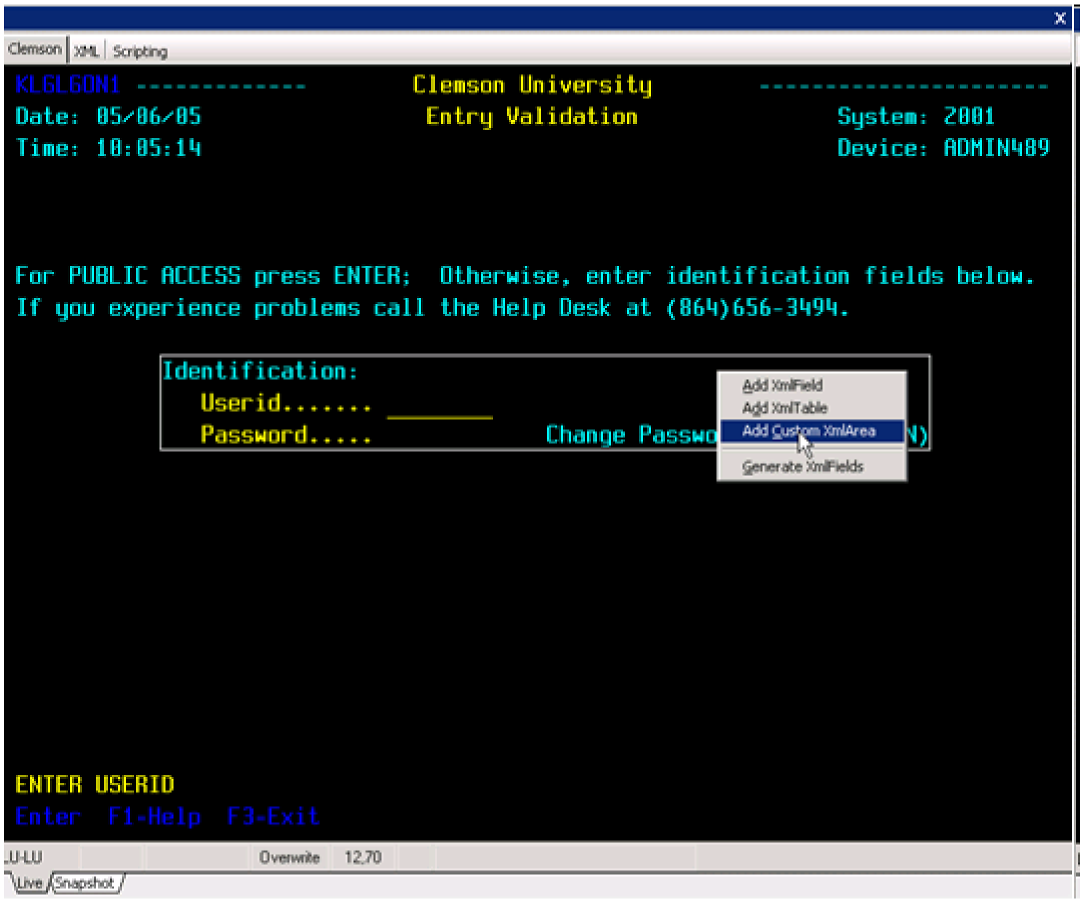
To delete the Table you have to do the inverse; by performing this action the XML Template will automatically be updated.

2.2.3.2.2.8 Adding custom XML Template Areas

The addition of a custom area to your template gives you the possibility to produce by yourself its inner XML at runtime.

To add a custom XML Template Area do the following:

- Select the area.



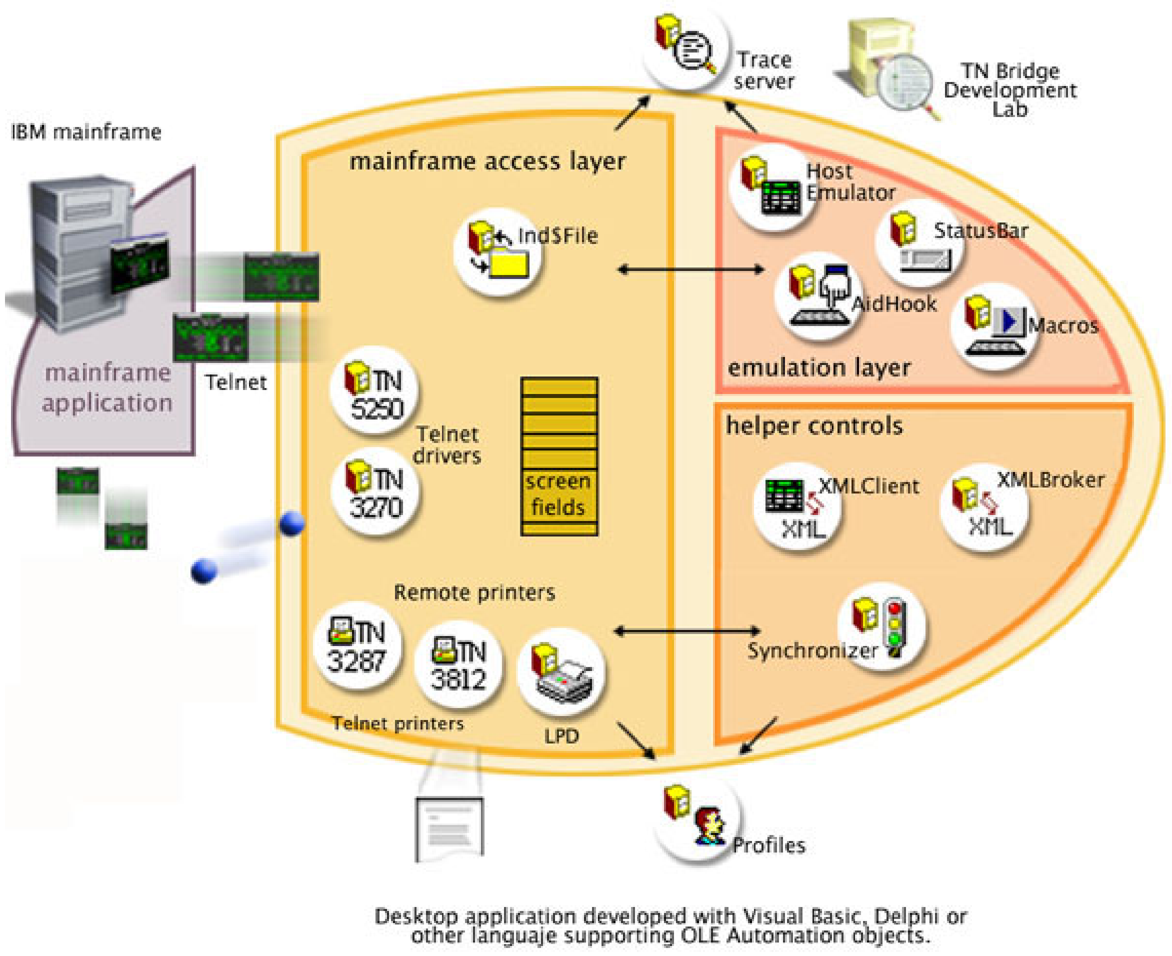
```
KLGLGON1 -------------          Clemson University          ----------------------
Date: 05/06/05                   Entry Validation                    System: 2001
Time: 10:05:14                                                       Device: ADMIN489


For PUBLIC ACCESS press ENTER;  Otherwise, enter identification fields below.
If you experience problems call the Help Desk at (864)656-3494.

            Identification:
              Userid.......  ________
              Password.....                Change Passwo


ENTER USERID
Enter  F1-Help  F3-Exit
```

3 Host Integration Pack Programming

3.1 Host Integration Pack's component layers

TN BRIDGE Integration Pack design and run-time are implemented with .NET native components separated in three layers, according its purpose:

- **Mainframe Access Layer**
- **Emulation Layer**
- **Helper Controls**



3.2 Accessing the Mainframe

Accessing the mainframe with TN BRIDGE Host Integration Pack involves the use of the main communication components: the Tn3270 and Tn5250 components.They expose a set of properties, methods and events to handle the exchange of data.

- **Tn3270 Component**

  Implements the client TN3270E communication protocol. It connects to the telnet 3270 server that is included in the TCP/IP package for IBM S/3XX Mainframes (if it has TCP/IP support) or to an external gateway TN3270-SNA like Microsoft SNA Server. It supports connecting to a LU pool or specific LU Name. Supports extended Data Stream and IBM 3278 mod 2,3 and 4 emulation.

- **Tn5250 Component**

  Implements the client TN5250 communication protocol. It connects to the telnet 5250 server that is included in the TCP/IP package for the IBM AS/400 (if it have TCP/IP support) or to an external gateway TN5250-SNA like Microsoft SNA Server. Supports 24x80 and 27x132 terminal types.

Next, we'll learn about the **main programming tasks** needed to get connected and exchange data with the mainframe.

3.2.1 Main Programming Tasks

Host Integration Pack provides methods, properties and events for connecting to the mainframe, accessing to screen data and fields, sending data and closing the connection. Also, provides means for making these calls in synchronous or asynchronous fashion.

- **Connecting/Disconnecting** explains how we can programatically get connected/ disconnected to the mainframe both in synchronic and asynchronic fashion.
- **Processing Screen Data** shows the methods we can use to get access to screen data.
- **Sending Data** shows the methods we can use to fill input fields and send them along with a function key.
- **Handling Events** gives an explanation of the events that Tn components fire reacting to the different real events carried by communication with the mainframe.
- **Waiting Methods** shows the different alternatives for handling events in synchronous fashion.
- **Using Tn Pools** explain how we can acquire Tn objects from a named Pool.

3.2.1.1 Connecting/Disconnecting

Host Integration Pack provides two methods for connecting up to the mainframe and one for disconnecting.

- **Synchronous Connection** shows how we can get connected in synchronous fashion.
- **Asynchronous Connection** shows how we can get connected in asynchronous fashion.
- **Disconnection** shows how to terminate the connection.

3.2.1.1.1 Synchronous Connection

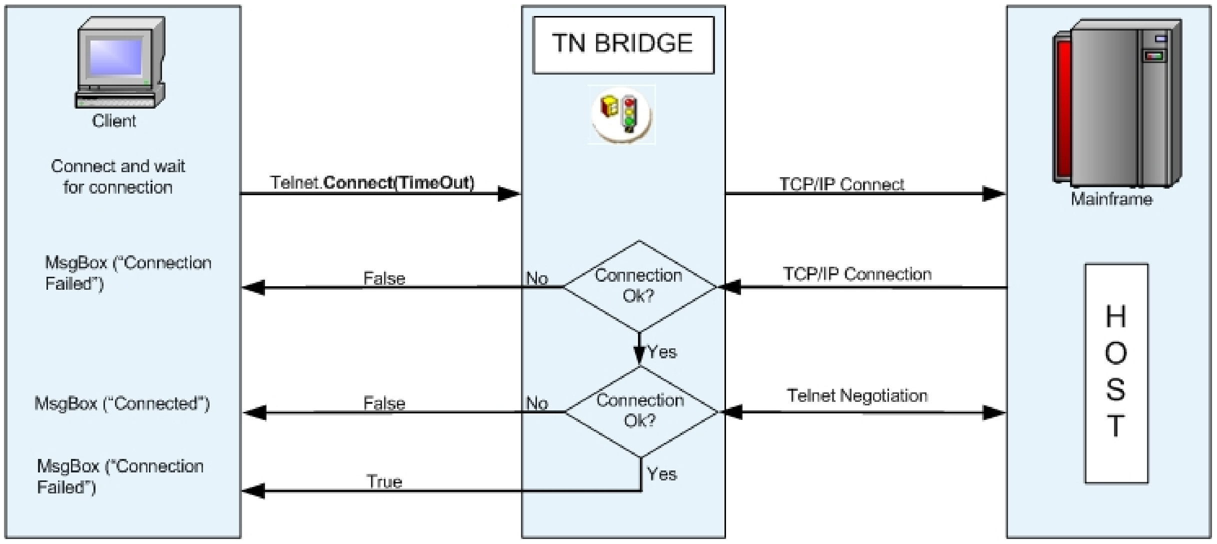
This method try to establish the connection to the specified Host and Port. This call blocks the code execution, while waits until the connection is succesfully established or the specified timeout expires.

The Host and Port properties must be set for the connection to be successfully established.

VB.NET
*[Boolean] = tnxxxx.***Connect(**Timeout As Integer**)**

C#
*[bool] = tnxxxx.***Connect(**int Timeout**);**

The following graphic shows the sequence of actions corresponding to the synchronous Connect method:



Examples:

VB.NET:
```
Private Sub cmdConnect_Click(eventSender As object, eventArgs As
System.EventArgs)
  tn3270.Host = "clemson.clemson.edu"
  If tn3270.Connect(10000) Then
    MessageBox.Show("Connected!")
  End If
End Sub
```

C#:
```
private void cmdConnect_Click(object eventSender, System.EventArgs eventArgs)
{
       tn3270.Host = "clemson.clemson.edu";
       if (tn3270.Connect(10000))
       {
              MessageBox.Show("Connected!");
       }
}
```

3.2.1.1.2 Asynchornous Connection

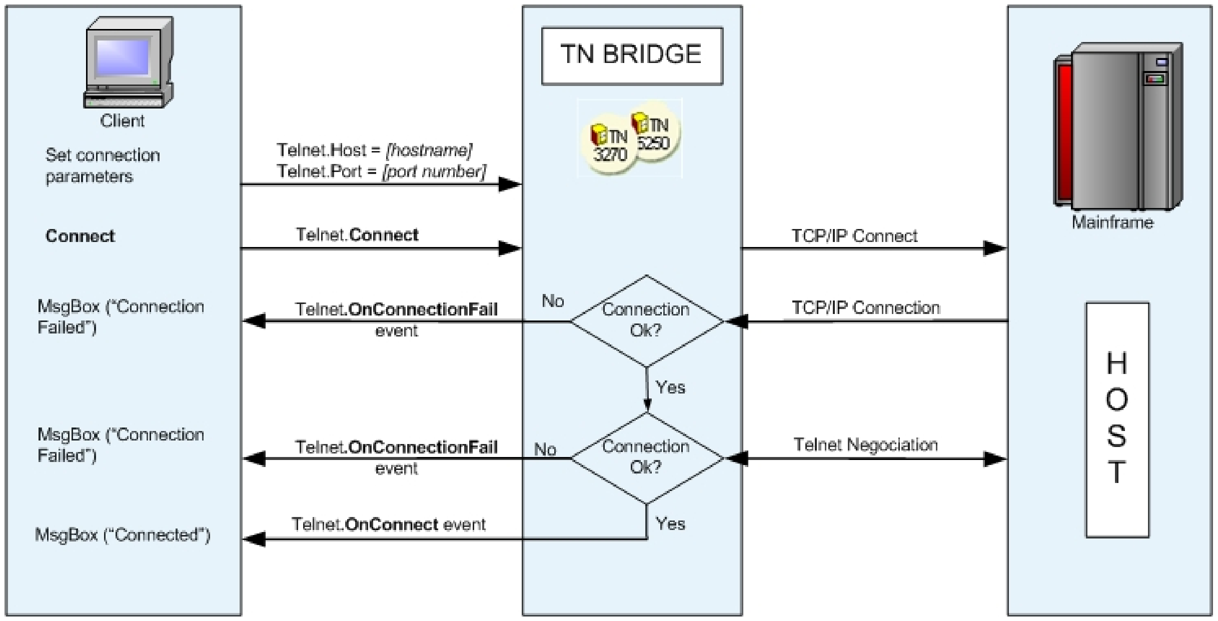
This method try to establish the connection to the specified Host and Port. This call **doesn't** block the code execution, so you should use the OnConnect and OnConnectionFail events to determine whether the connection is succesfully established or not. Alternativally, you can make a call to WaitForConnect synchronous method.

The Host and Port properties must be set for the connection to be successfully established.

VB.NET
*tnxxxx.***Connect()**

C#
*tnxxxx.***Connect();**

The following graphic shows the sequence of actions corresponding to the asynchronous Connect method:



VB.NET:
```
Private Sub cmdConnect_Click(eventSender As object, eventArgs As
System.EventArgs)
   tn3270.Host = Edit1.Text
   tn3270.Port = 23
   tn3270.Connect()
End Sub

Private Sub tn3270_OnConnect(eventSender As object, eventArgs As
System.EventArgs)
   MsgBox("Connected")
End Sub

Private Sub tn3270_OnConnectionFail(eventSender As object, eventArgs As
System.EventArgs)
   MsgBox("Connection failed")
End Sub
```

C#:
```
private void cmdConnect_Click(object eventSender, System.EventArgs eventArgs)
{
       tn3270.Host = Edit1.Text;
       tn3270.Port = 23;
       tn3270.Connect();
```

```
        }

        private void tn3270_OnConnect(object eventSender, System.EventArgs eventArgs)
        {
                MsgBox("Connected");
        }

        private void tn3270_OnConnectionFail(object eventSender, System.EventArgs
eventArgs)
        {
                MsgBox("Connection failed");
        }
```

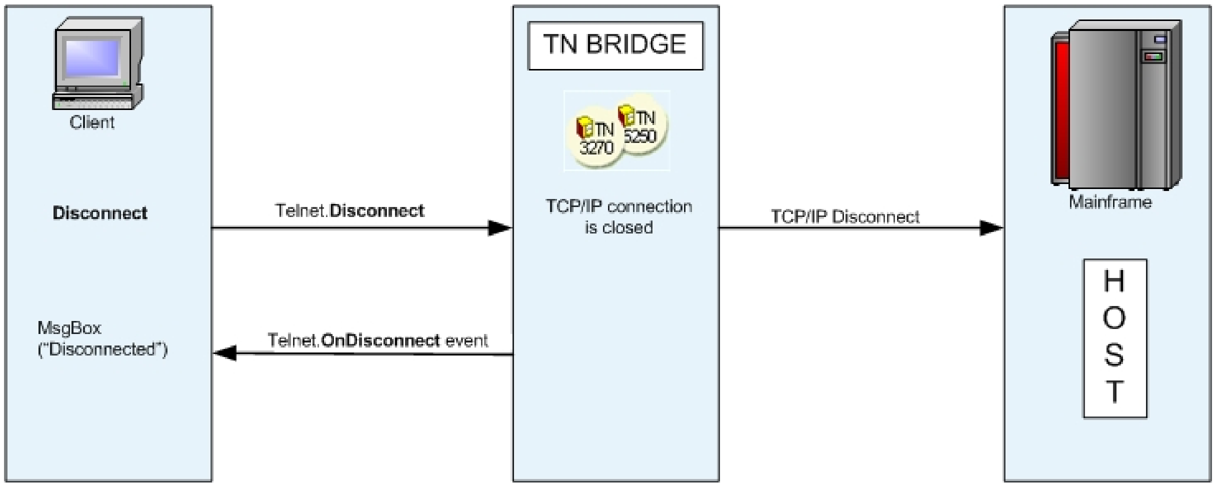
3.2.1.1.3 Disconnection

This method terminates the current connection.

VB.NET
*tnxxxx.***Disconnect()**

C#
*tnxxxx.***Disconnect();**

The following graphic shows the sequence of actions corresponding to the Disconnect method:



```
        VB.NET:
        Private Sub cmdDisconnect_Click(eventSender As object, eventArgs As
System.EventArgs)
            tn3270.Disconnect()
        End Sub

        C#:
        private void cmdDisconnect_Click(object eventSender, System.EventArgs
eventArgs)
        {
            tn3270.Disconnect();
        }
```

3.2.1.2 Processing Screen data

The access to the screen data can be done through GetText method call or using the HostFields collection.

- **Using GetText** shows how we can read screen data using the GetText method.
- **Using HostFields** explain about HostFields collection and its use.

3.2.1.2.1 Using GetText

The GetText method is the simplest way to read data from the screen buffer.

VB.NET
[*String* =] *tnxxxx.***GetText (***StartPos As Integer, [EndPos As Integer], [Attr As Boolean = False], [Eab As Boolean = False]***)**
[*String* =] *tnxxxx.***GetText (***Row As Integer, Col As Integer, Len as Integer, [Attr As Boolean = False], [Eab As Boolean = False]***)**

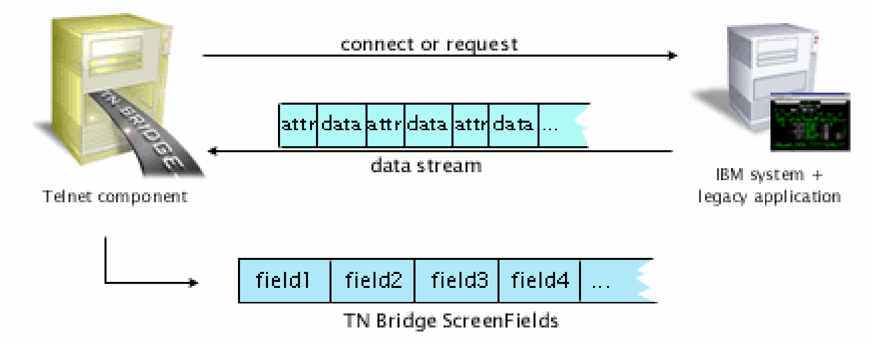
C#
[*string* =] *tnxxxx.***GetText (***int StartPos, [int StartPos], [bool Attr = False], [bool Eab = False]***);**
[*string* =] *tnxxxx.***GetText (***int Row, in Col, int Len, [bool Attr = False], [bool Eab = False]***);**

Start position and the end position of the text buffer to retrieve can be set using StartPos and EndPos parameters or Row,Col and Len parameters.

```
        VB.NET:
        Private Sub tn3270_OnScreenChange(eventSender As object, eventArgs As
System.EventArgs)
            ' this code gets the Text from position 150
            Dim s As String = tn3270.GetText(150)
            ...
            ' this code gets the Text from position 150 to 200
            Dim s As String = tn3270.GetText(150,200)
            ...
            ' this code gets the Text from row 15 column 4 with a lenght of 20
            Dim s As String = tn3270.GetText(15,4,20)
        End Sub

        C#:
        private void tn3270_OnScreenChange(object eventSender, System.EventArgs
eventArgs)
        {
            // this code gets the Text from position 150
            string s = tn3270.GetText(150);
            ...
            // this code gets the Text from position 150 to 200
            string s = tn3270.GetText(150,200);
            ...
            // this code gets the Text from row 15 column 4 with a lenght of 20
            string s = tn3270.GetText(15,4,20);
         }
```

3.2.1.2.2 Using HostFields

The mainframe screen arrives in a buffer with a block of data: the *data-stream*. This data-stream is made up of a succession of orders, attributes and data.



The screen buffer is made up of fields. Each field is delimited by an attribute and the next one.

TN BRIDGE read these fields and collect them into the HostFields collection. Fields in HostFields collection can be accessed by index or name. The name is built up with its screen coordinates as follows: *R[row number]C[column number]* For example, the field at row 4 column 30 in the screen is identified as *R4C30*.

VB.NET:

```
Private Sub tn3270_OnScreenChange(eventSender As object, eventArgs As
System.EventArgs)
    ' this code gets the Data in the Field located at Row 10 Column 2
    Dim s As String = tn3270.HostFields["R10C2"].Data
    ...
    ' this code gets the Data in the Field 10
    Dim s As String = tn3270.HostFields[10].Data
End Sub
```

C#:

```
private void tn3270_OnScreenChange(object eventSender, System.EventArgs
eventArgs)
{
    // this code gets the Data in the Field located at Row 10 Column 2
    string s = tn3270.HostFields["R10C2"].Data;
    ...
    // this code gets the Data in the Field 10
    string s = tn3270.HostFields[10].Data;
    ...
}
```

3.2.1.3 Sending Data

Sendind data to the mainframe can be splitted into two steps: filling the edit data buffer and sending all the modified data to the mainframe along with a function key.

- **Using Type method** explains how to fill input fields by simulating the keyboard entry.
- **Using EditFields collection** explain the use of EditFields collection.
- **The Press and PressAndWait methods** explain how to use unblocking or blocking methods to send data and function keys to the mainframe.

3.2.1.3.1 Using Type method

The Type method can be used to send key sequences to the mainframe starting from the current cursor position.

VB.NET
*tnxxxx.***Type (***Keys As String***)**

C#
*tnxxxx.***Type (***string Keys***)**;

By using special codes you can send several special keys. These codes consist of an escape character ("@") and a mnemonic code that corresponds to the supported function.

Type method can also make entered data to be sent along with an AID key (Attention Identifier key), avoiding the use of Press method.

The following table lists the functions keys and its corresponding codes.

Code Value	Meaning
@A@Q	Attention
@<	Backspace
@B	BackTab (Left Tab)
@C	Clear
@E	Enter
@F	Erase Field
@A@Q	Sys Request
@T	Tab (Right Tab)
@x	PA1
@y	PA2
@z	PA3
@1	PF1
@2	PF2
@3	PF3
@4	PF4
@5	PF5
@6	PF6

@7	PF7
@8	PF8
@9	PF9
@a	PF10
@b	PF11
@c	PF12
@d	PF13
@e	PF14
@f	PF15
@g	PF16
@h	PF17
@i	PF18

VB.NET:
```
Private Sub SendUserAndPassword(userId As String, password As String)
   '@T is translated to TAB key
   tn3270.Type(userId+"@T"+password)
   tn3270.PressAndWait(Enter)
End Sub
```

C#:
```
private void SendUserAndPassword(string userId,string password)
{
   // @T is translated to TAB key
   tn3270.Type(userId+"@T"+password);
   tn3270.PressAndWait(Enter);
}
```

3.2.1.3.2 Using EditFields Collection

The EditFields collection contain a reference to all the fields in HostFields collection with the unprotected attribute set to true. These fields can be accessed in order to modify its data, which later will be sent to the mainframe along with a function key (AID key).

VB.NET
[*Field* =] *tnxxxx.***EditFields (***Index As Integer***)**
[*Field* =] *tnxxxx.***EditFields (***Index As String***)**

C#
[*Field* =] *tnxxxx.***EditFields [***int index***]***;*
[*Field* =] *tnxxxx.***EditFields [***string index***]***;*

VB.NET:
```
Private Sub cmdLogin_Click(eventSender As object, eventArgs As
System.EventArgs)
   ' this code sets userId in Field located at Row 10 Column 2
   tn3270.EditFields["R10C2"].Data = "puser"
   ' this code sets the password in Field located at Row 12 Column 2
   tn3270.EditFields["R12C2"].Data = "k2156"
```

```
          ' this code sends the ENTER Attention Identifier Key along with all modified
EditFields
          tn3270.Press("Enter")
       End Sub
```

C#:

```
       private void cmdLogin_Click(object eventSender, System.EventArgs eventArgs)
       {
          // this code sets userId in Field located at Row 10 Column 2
          tn3270.EditFields["R10C2"].Data = "puser";
          // this code sets the password in Field located at Row 12 Column 2
          tn3270.EditFields["R12C2"].Data = "k2156";
          // this code sends the ENTER Attention Identifier Key along with all
modified EditFields
          tn3270.Press("Enter");
       }
```

3.2.1.3.3 The Press and PressAndWait methods

In a real terminal, the typed data is sent to the mainframe upon pressing one of the keys known as Attention Identifier keys (AID). These keys act as function keys that are sent along with the typed data. In Host Integration Pack, you can use the Press and PressAndWait methods to simulate this action.

- **The Press method** explains how you can send an AID key in unblocking fashion
- **The Press and PressAndWait methods** explains how you can send an AID Key in a blocking fashion.

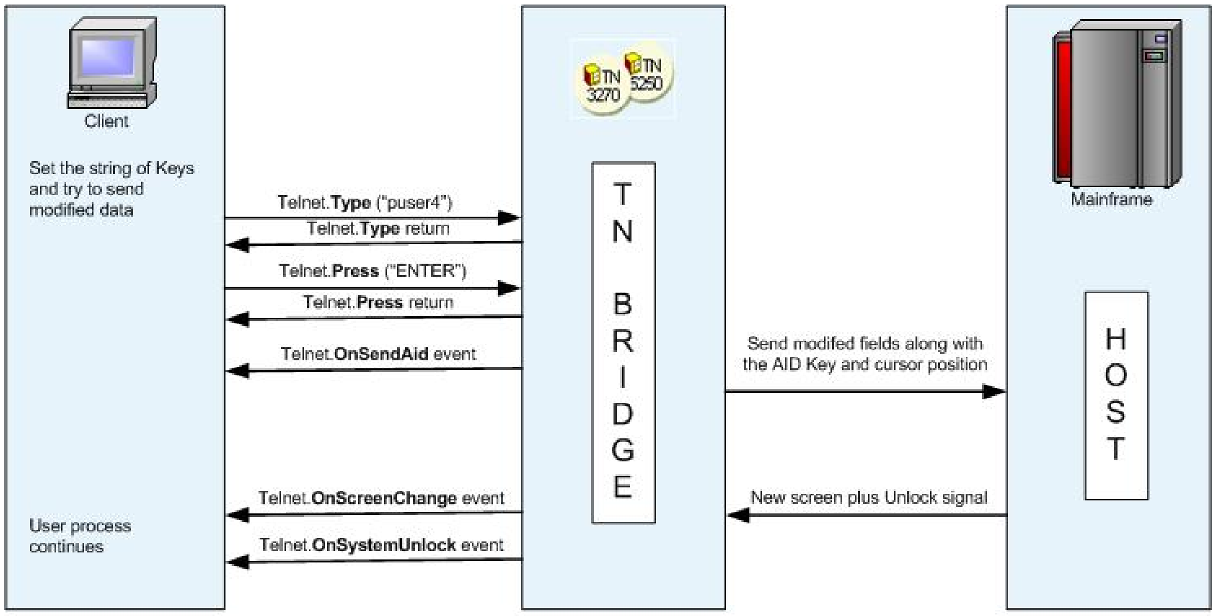
3.2.1.3.3.1 The Press method

The Press method sends an Attention Identifier Key along with the modified fields. Modified fields can be input fields (unprotected) or sometimes protected fields having the property Modified set to True.

VB.NET
*tnxxxx.***Press (***aidKey as AidKey***)**
*tnxxxx.***Press (***aidKey As String***)**

C#
*tnxxxx.***Press (***AidKey aidKey***)***;*
*tnxxxx.***Press (***string aidKey***)***;*

The following graphic shows the sequence of actions corresponding to the asynchronous Press method:


Client
Set the string of Keys and try to send modified data
User process continues
Telnet.Type ("puser4")
Telnet.Type return
Telnet.Press ("ENTER")
Telnet.Press return
Telnet.OnSendAid event
Telnet.OnScreenChange event
Telnet.OnSystemUnlock event
TN 3270
TN 5250
TN BRIDGE
Send modifed fields along with the AID Key and cursor position
New screen plus Unlock signal
Mainframe
HOST

Note: The execution of the Type method is stopped when an AID Key is sent to the Host.

VB.NET*:*

```
        Private Sub cmdLogin_Click(eventSender As System.Object, eventArgs As
System.EventArgs)
            ' this code sets userId in Field located at Row 10 Column 2
            tn3270.EditFields["R10C2"].Data = "puser"
            ' this code sets the password in Field located at Row 12 Column 2
            tn3270.EditFields["R12C2"].Data = "k2156"
            ' this code sends the ENTER Attention Identifier Key along with all modified
EditFields
            tn3270.Press("Enter")
        End Sub
```

C#*:*

```
        private void cmdLogin_Click(object eventSender, System.EventArgs eventArgs)
        {
            // this code sets userId in Field located at Row 10 Column 2
            tn3270.EditFields["R10C2"].Data = "puser";
            // this code sets the password in Field located at Row 12 Column 2
            tn3270.EditFields["R12C2"].Data = "k2156";
            // this code sends the ENTER Attention Identifier Key along with all
modified EditFields
            tn3270.Press("Enter");
        }
```

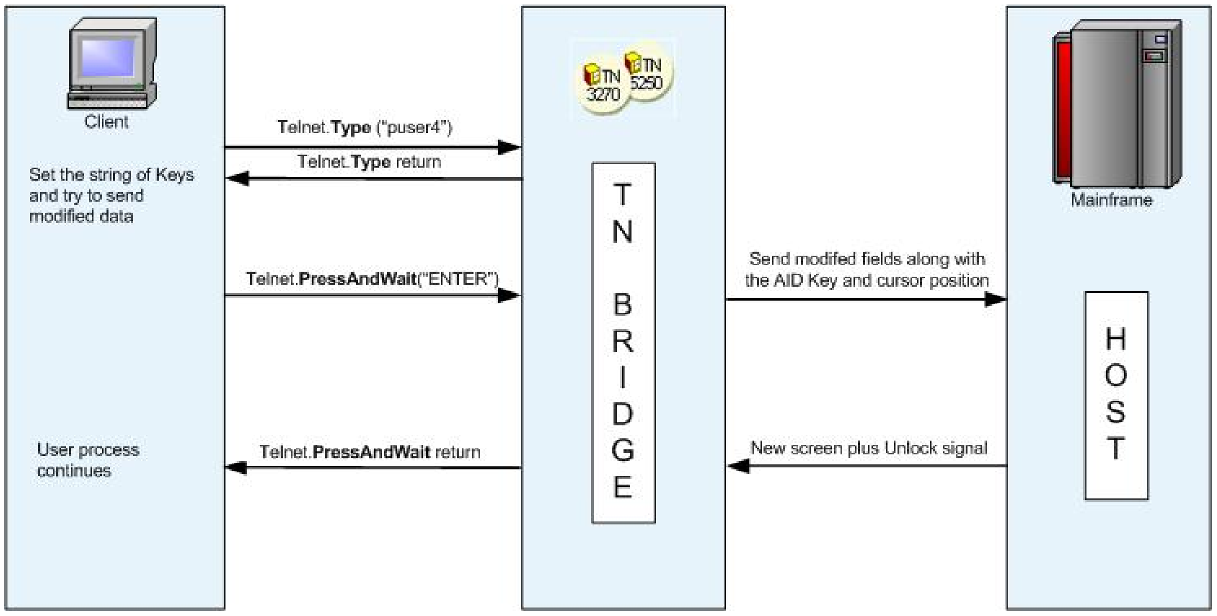
3.2.1.3.3.2 The PressAndWait method

The PressAndWait method acts like Press method, but it blocks the code execution waiting until the system gets unlocked.

VB.NET
*tnxxxx.***PressAndWait (***aidKey as AidKey***)**
*tnxxxx.***PressAndWait (***aidKey As String***)**

C#
*tnxxxx.***PressAndWait (***AidKey aidKey***);**
*tnxxxx.***PressAndWait (***string aidKey***);**

The following graphic shows the sequence of actions corresponding to the synchronous PressAndWait method:


Client
Set the string of Keys and try to send modified data
User process continues
Telnet.Type ("puser4")
Telnet.Type return
Telnet.PressAndWait("ENTER")
Telnet.PressAndWait return
TN 3270
TN 5250
TN BRIDGE
Send modifed fields along with the AID Key and cursor position
New screen plus Unlock signal
Mainframe
HOST

VB.NET:

```
        Private Sub cmdLogin_Click(eventSender As object, eventArgs As
System.EventArgs)
            ' this code sets userId in Field located at Row 10 Column 2
            tn3270.EditFields["R10C2"].Data = "puser"
            ' this code sets the password in Field located at Row 12 Column 2
            tn3270.EditFields["R12C2"].Data = "k2156"
            ' this code sends the ENTER Attention Identifier Key along with all modified
EditFields
            tn3270.Press("Enter")
        End Sub
```

C#:

```
        private void cmdLogin_Click(object eventSender, System.EventArgs eventArgs)
        {
            // this code sets userId in Field located at Row 10 Column 2
            tn3270.EditFields["R10C2"].Data = "puser";
            // this code sets the password in Field located at Row 12 Column 2
            tn3270.EditFields["R12C2"].Data = "k2156";
            // this code sends the ENTER Attention Identifier Key along with all
modified EditFields
            tn3270.PressAndWait("Enter");
        }
```

3.2.1.4 Handling Events

Tn events are specially important in asynchronous programming. A typical case is a custom terminal emulator, where the screens and state info must be reflected in the user interface.

The Tn events are:

OnConnect	: Fires after a successfully connection to the mainframe is established.
OnConnectFail	: Fires after the connection to the mainframe fails.
OnDisconnect	: Fires after a disconnection with the mainframe.
OnSessionState	: Fires when a session-state change takes place.
OnSystemLock	: Fires when a terminal changes to a system locked state.
OnSystemUnLock	: Fires when a terminal changes to a system unlocked state.
OnScreenChange	: Fires after a new data screen has arrived.
OnSendAid	: Fires when an AID key is about to be sent (before is sent).

Some code examples:

- You can monitor the connection status with the following events:

VB.NET:
```
Private Sub tn3270_OnConnect(eventSender As System.Object, eventArgs As
System.EventArgs)
   MessageBox.Show("Connected to the Host!");
End Sub

Private Sub tn3270_OnConnectionFail(eventSender As System.Object, eventArgs As
System.EventArgs)
   MessageBox.Show("Connection Failed!");
End Sub

Private Sub tn3270_OnDisconnect(eventSender As System.Object, eventArgs As
System.EventArgs)
   MessageBox.Show("Disconnected from the Host!");
End Sub
```

C#:
```
private void tn3270_OnConnect(object eventSender, System.EventArgs eventArgs)
{
   MessageBox.Show("Connected to the Host!");
}

private void tn3270_OnConnectionFail(object eventSender, System.EventArgs
eventArgs)
{
   MessageBox.Show("Connection Failed!");
}

private void tn3270_OnDisconnect(object eventSender, System.EventArgs
eventArgs)
{
   MessageBox.Show("Disconnected from the Host!");
```

```
        }
```

- In these events you can display or use the host data

VB.NET:
```
        Private Sub tn3270_OnScreenChange(eventSender As System.Object, eventArgs As
System.EventArgs)
           MessageBox.Show("A new screen has been received");
        End Sub

        Private Sub tn3270_OnSystemUnLock(eventSender As System.Object, eventArgs As
System.EventArgs)
           MessageBox.Show("The latest screen has arrived");
        End Sub
```

C#:
```
        private void tn3270_OnScreenChange(object eventSender, System.EventArgs
eventArgs)
        {
           MessageBox.Show("A new screen has been received");
        }

        private void tn3270_OnSystemUnLock(object eventSender, System.EventArgs
eventArgs)
        {
           MessageBox.Show("The latest screen has arrived");
        }
```

- These events controls when you are able to send data to the mainframe

VB.NET:
```
        Private Sub tn3270_OnSystemUnLock(eventSender As System.Object, eventArgs As
System.EventArgs)
           MessageBox.Show("We can send data to the host now");
        End Sub

        Private Sub tn3270_OnSystemLock(eventSender As System.Object, eventArgs As
System.EventArgs)
           MessageBox.Show("We can''t send data at this moment!");
        End Sub
```

C#:
```
        private void tn3270_OnSystemUnLock(object eventSender, System.EventArgs
eventArgs)
        {
           MessageBox.Show("We can send data to the host now");
        }

        private void tn3270_OnSystemLock(object eventSender, System.EventArgs
eventArgs)
        {
           MessageBox.Show("We can''t send data at this moment!");
        }
```

- Now you know how to connect to the host and handle the communication events. It's time to process the host data:

VB.NET:

```
Private Sub tn3270_OnSystemUnLock(eventSender As System.Object, eventArgs As System.EventArgs)
    'At this time we can send data to the host
    'First we check if we are in the log-on screen

   If (tn3270.GetText(15,1,tn3270.Cols).Substring(10,8) = "UserId:") Then
    'Now we fill in the input fields
    tn3270.EditFields[0].Data = "MyUserId"
    tn3270.EditFields[1].Data = "MyPasword"
    'and send them to the host with the Enter Key!
    tn3270.Press(Enter)
  Else If
    ...
  End If
End Sub
```

C#:

```
private void tnb3270_OnSystemUnLock(object sender, System.EventArgs e)
{
   // At this time we can send data to the host
   // First we check if we are in the log-on screen

  if (tn3270.GetText(15,1,tn3270.Cols).Substring(10,8) = "UserId:")
  {
    // Now we fill in the input fields
    tn3270.EditFields[0].Data = "MyUserId";
    tn3270.EditFields[1].Data = "MyPasword";
    // and send them to the host with the Enter Key!
    tn3270.Press(Enter);
  }
  else
  {
       ...
  }
}
```

3.2.1.5 Waiting Methods

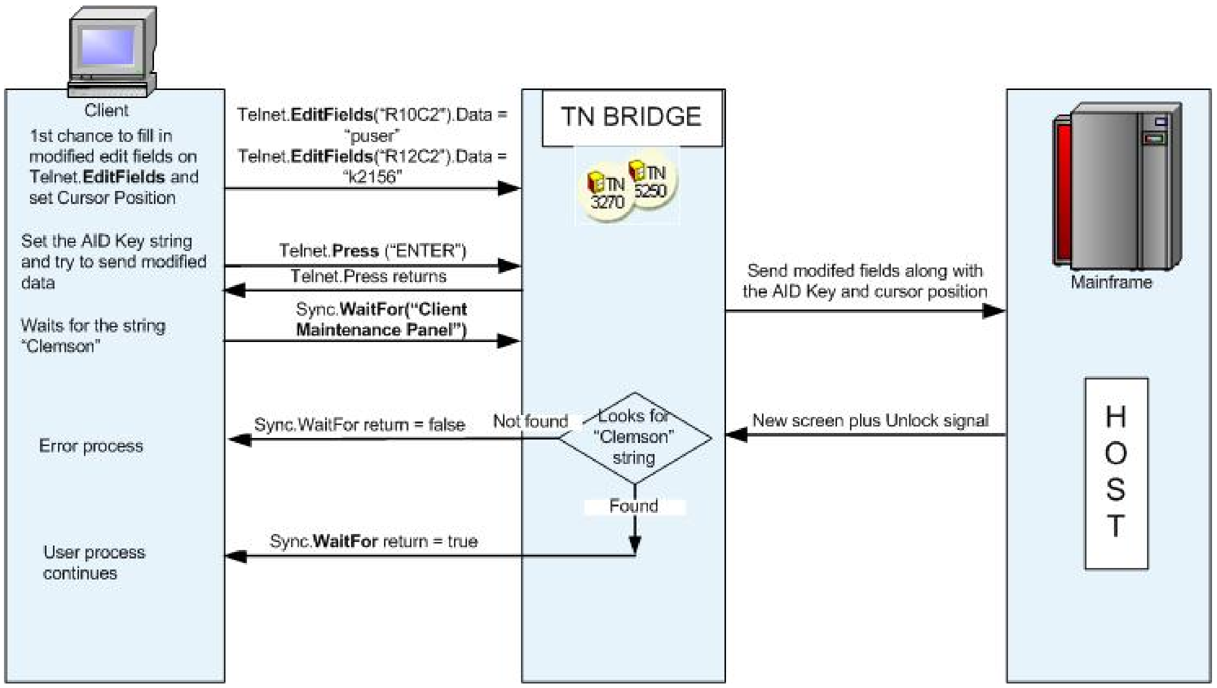
There are several waiting methods that allow to wait for specifics events or event sequence, blocking the code execution until the event is raised or the operation times out. Main methods are:

VB.NET
[Boolean] = *tnxxxx.***WaitForConnect (***[TimeOut As Integer]***)**
[Boolean] = *tnxxxx.***WaitForScreen (***[TimeOut As Integer]***)**
[Boolean] = *tnxxxx.***WaitForUnlock (***[TimeOut As Integer]***)**
[Boolean] = *tnxxxx.***WaitFor (***StrValue As String, [TimeOut As Integer]***)**
[Integer] = *tnxxxx.***WaitFor (***StrValues As Array of String, [TimeOut As Integer]***)**

C#
[bool] = *tnxxxx.***WaitForConnect (***[int TimeOut]***);**
[bool] = *tnxxxx.***WaitForScreen (***[int TimeOut]***);**
[bool] = *tnxxxx.***WaitForUnlock (***[int TimeOut]***);**
[bool] = *tnxxxx.***WaitFor (***string StrValue, [int TimeOut]***);**
[int] = *tnxxxx.***WaitFor (***string[] StrValues, [int TimeOut]***);**

WaitForConnect waits until the telnet connection is successfully established.
WaitForScreen waits until a new screen arrives after the connection is established or an AID key was sent.
WaitForUnlock waits until the host system turns to an unlocked state.
WaitFor waits for a screen containing the specified string or strings.

The following graphic shows the sequence of actions corresponding to the WaitFor method:


Client
1st chance to fill in modified edit fields on Telnet.EditFields and set Cursor Position
Set the AID Key string and try to send modified data
Waits for the string "Clemson"
Error process
User process continues
Telnet.EditFields("R10C2").Data = "puser"
Telnet.EditFields("R12C2").Data = "k2156"
Telnet.Press ("ENTER")
Telnet.Press returns
Sync.WaitFor("Client Maintenance Panel")
Sync.WaitFor return = false
Not found
Sync.WaitFor return = true
TN BRIDGE
TN 3270
TN 5250
Looks for "Clemson" string
Found
Send modifed fields along with the AID Key and cursor position
New screen plus Unlock signal
Mainframe
HOST

Examples

```
        VB.NET:
        Private Sub cmdGetInfo_Click(eventSender As object, eventArgs As
System.EventArgs)
          tn3270.Host = Edit1.Text
          If (tn3270.Connect(10000) Then
            tn3270.Type("Anonymous")
            tn3270.PressAndWait("ENTER")
            tn3270.Type("srch "+Edit2.Text+"@E")
            tn3270.WaitFor("Library")
            Edit3.Text = tn3270.GetText(20,5,30)
            tn3270.Disconnect()
          End If
End Sub
```

C#:

```
private void cmdGetInfo_Click(object eventSender, System.EventArgs eventArgs)
{
  tn3270.Host = Edit1.Text;
  if (tn3270.Connect(10000))
  {
    tn3270.Type("Anonymous");
    tn3270.PressAndWait("ENTER");
    tn3270.Type("srch "+Edit2.Text+"@E");
    tn3270.WaitFor("Library");
    Edit3.Text = tn3270.GetText(20,5,30);
    tn3270.Disconnect();
  }
}
```

3.2.1.6 Using Tn Pools

TN BRIDGE Host Integration Pack allows persisting Tn3270/5250 objects in a Pools that are available in the whole application domain.

The method used to acquire a Tn object from the pool is:

VB.NET
*Shared tnxxxx.***FromPool(**poolId as String,[sessionId as String],[InactivityTimeout as Integer]**)**

C#
*static tnxxxx.***FromPool(**string poolId,[string sessionId],[int InactivityTimeout]**);**

This method is static (shared) and it returns a Tn from the Pool with the name poolId and optionally identified by SessionId parameter. In case of no Tn object is available, it returns a new Tn object which is also added to the pool.

The Tn object must be released by using the method:

VB.NET
*tnxxxx.***Release()**

C#
*tnxxxx.***Release();**

Persisting telnet objects in pools allows to:

- **Reusing the connection** in a disconnected environment like ASP.NET
- **Keeping Tn objects ready** for their reuse.

3.2.1.6.1 Reusing the connection in ASP.NET

In a disconnected environment like ASP.NET, the Tn Pool allow to persist the Tn object on a session basis.
Upon creation, Tn object generates an unique identifier that can be accessed through the SessionId property. When running under ASP.NET, this SessionId is set to the value of

SessionId property belonging to the Session object. This allows to easily ask to the pool for a session´s Tn object.

VB.NET:
```
Private Sub Page_Load()
  Dim telnet As Tn3270 = Tn3270.FromPool("Clemson",Session.SessionId)
  Try
    If Not telnet.IsConnected() Then
      telnet.Host = "clemson.clemson.edu"
      telnet.Port = 23
      telnet.Connect(10000)
    End If
    ...
  Finally
    telnet.Release()
  End Try
End Sub
```

C#:
```
public void Page_Load()
{
  Tn3270 telnet = Tn3270.FromPool("Clemson",Session.SessionId);
  try
  {
    if (!telnet.IsConnected())
    {
      telnet.Host = "clemson.clemson.edu";
      telnet.Port = 23;
      telnet.Connect(10000);
    }
    ...
  }
  finally
  {
    telnet.Release();
  }
}
```

3.2.1.6.2 Keeping Tn objects ready

Many times, having these objects connected and logged in into a mainframe application is the common start and ending point for making a mainframe transacion (CICS transactions are a clear example). So maintaining Tn objects ready for their reuse is a good technique for avoiding unnecessary delays.

In these cases you can acquire a Tn object just by passing the pool name without any object identifier. This way you will get a free Tn object to reuse.

VB.NET:
```
<WebMethod>
Private Sub Page_Load()
  Dim telnet As Tn3270 = Tn3270.FromPool("Clemson")
  Try
    If Not telnet.IsConnected() Then
      telnet.Host = "clemson.clemson.edu"
      telnet.Port = 23
```

```
        telnet.Connect(10000)
      End If
      ...
    Finally
      telnet.Release()
    End Try
End Sub
```

C#:

```
[WebMethod]
public void Connect()
{
  Tn3270 telnet = Tn3270.FromPool("Clemson");
  try
  {
    if (!telnet.IsConnected())
    {
      telnet.Host = "clemson.clemson.edu";
      telnet.Port = 23;
      telnet.Connect(10000);
    }
    ...
  }
  finally
  {
    telnet.Release();
  }
}
```

3.3 Working with the Display

3.3.1 Terminal Emulator Controls

The TN BRIDGE Emulation Layer is made up of 5 Terminal Emulator components. These components implement the main features for a terminal emulator program like: Display Screen, Status Bar, Keyboard Handling, Macro Sequences and Profiles.

- **Emulation Screen: Display component**

  Implements an emulation panel. This panel shows host data and accepts input data to be sent to the mainframe. The displayed data is formatted according to the data stream format (3270 or 5250).

- **Emulation Status Bar: StatusBar component**

  Implements an emulation status bar. It works joined to the TNBEmulator component, complementing it.

- **Keyboard Handling: Keyboard component**

  Process the keyboard and maps to Emulation Function Keys and Attention Identifier Keys. The Emulation Function Keys are those which are processed locally to manage the emulator behavior for input data (For example: Tab key, Cursor movement keys, Home key, etc.). The Attention Identifier Keys are those which provoke sending data and the properly Attention Identifier Code (For example: Enter key, PF01 key, Clear key, etc.).

- **Macro Sequences: Macro component**

  Implements the recording and playing mainframe navigation sequences to automate tasks.

- **Profiles: Profiles component**

  Implements a common interface for saving and loading profiles like screen emulation styles, host connection profiles and macro sequences.

3.3.2 Main Programming Tasks

For using the Terminal Emulator components the main tasks are:

- Drop a Tn3270 or Tn5250 Component on your form.
- Drop to your form the following components: Display, StatusBar and Keyboard.
- Set the *Telnet property* for Emulator and StatusBar components to the telnet component (Tn3270 or Tn5250). You can set it at run-time using the following code in the Load or Create event of the form:

  VB.NET:
  ```
  Private Sub Form_Load(sender As System.Object, eventArgs As System.EventArgs)
    tnDisplay.Telnet = tn3270
    tnStatusBar.Telnet = tn3270
    ...
  End Sub
  ```

  C#:
  ```
  private void Form_Load(object sender, System.EventArgs e)
  {
    tnDisplay.Telnet = tn3270;
    tnStatusBar.Telnet = tn3270;
    ...
  }
  ```

- Set the *Emulatorcomponent property* from StatusBar component to the Display dropped in the form.

  VB.NET:
  ```
    tnStatusBar.Display = tnDisplay
  ```

  C#:
  ```
    tnStatusBar.Display = tnDisplay;
  ```

- Set *Keyboardcomponent property* from Emulator component to the Keyboard dropped in the form.

  VB.NET:
  ```
    tnDisplay.Keyboard = tnKeyboard
  ```

C#:
```
  tnDisplay.Keyboard = tnKeyboard;
```

- Now, all what you have to do is fill in the *Host property* for the Telnet component chosen, with your Mainframe DNS name or TCP/IP address and execute the *Connect method*. For example:

VB.NET:
```
        Private Sub Button1_Click(sender As System.Object, eventArgs As
System.EventArgs)
          'In the edit box you have to type the DNS name or
          'TCP/IP address of the mainframe you want to connect to.
          tn3270.Host = Edit1.Text
          tn3270.Connect()
          ...
        End Sub
```

C#:
```
        private void Button1_Click(object sender, System.EventArgs e)
        {
          // In the edit box you have to type the DNS name or
          // TCP/IP address of the mainframe you want to connect to.
          tn3270.Host = Edit1.Text;
          tn3270.Connect();
          ...
        }
```

Now you have a complete terminal emulator.

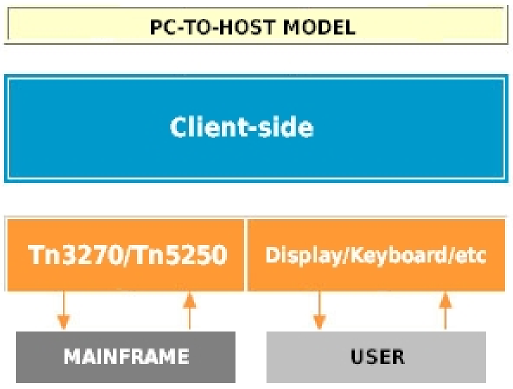
3.4 Programming with XML

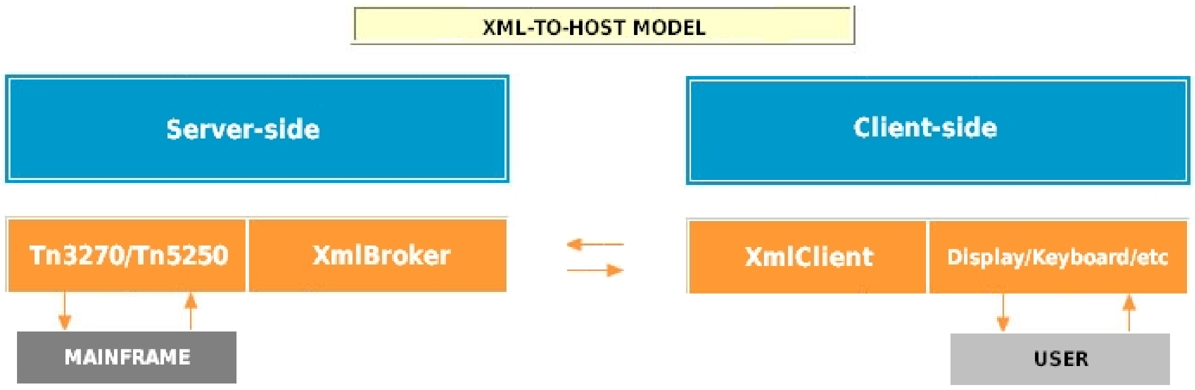
TN Bridge Host Integration Pack's Helper Controls allow you to **build an XML bridge** between your application and the mainframe.
The XmlBroker component is the one that allows to produce XML code from mainframe screens and convert XML input data and send it to the mainframe.

- **Working with XmlBroker** explains how we can provide or consume XML based on HostFields.
- **Working with XmlClient** explains how to use the provided XML by the XmlBroker with emulation purposes.
- **Working with XmlVirtual** explains how to create a simulated host application.

3.4.1 Xml-to-Host with Helper Controls

Helper Controls let you easily create your own Xml-to-Host connection. As shown in the next scheme, you can switch from a *Pc-to-Host connection model* (using Tn3270/Tn5250 controls from the client directly to the mainframe) to a *Xml-to-Host connection model* by using the **XmlClient** on the client-side and the **XmlBroker** on the server-side.





With Helper Controls you can develop a *thin-client* application (using **Display**, **StatusBar**, **Keyboard** and **XmlClient** controls) for communicating with a remote server that contains **XmlBroker** and **Telnet** controls. This remote server could be an ASPX application, a WebService, an specific TCP Server for attending **XmlClient**, and so.
XmlClient and **XmlBroker** exchange *XML* data between themselves. So, you should pass the *XML* data to the other endpoint when some event succeed.
Basically, **XmlClient** should be linked with other components in client-side and telnet object is linked to a **XmlBroker** in server-side. When an event succeed in **XmlClient** or **Telnet** object, it should be notified to the other side.

For more information, see **XmlClient** and **XmlBroker** components.

3.4.2 Working with XmlBroker

The XmlBroker is the key component to get a Xml representation of the screen.

Its usage can be seen in the following code snippet:

VB.NET:
```
Private Sub GetXml() As String
  Dim broker As XmlBroker = new XmlBroker(telnet)
  GetXml = broker.Xml
End Sub
```

```
Private Sub SetXml(ByVal Xml As String)
  Dim broker As XmlBroker = new XmlBroker(telnet)
  broker.Xml = Xml
End Sub
```

C#:

```
public string GetXml()
{
  XmlBroker broker = new XmlBroker(telnet);
  return broker.Xml;
}

public void SetXml(string Xml)
{
  XmlBroker broker = new XmlBroker(telnet);
  broker.Xml = Xml;
}
```

- See also **Provided Generic XML**

3.4.2.1 Provided Generic XML

By default, screen information is mapped to XML tags and attributes as described:

```
<TNBRIDGE>
    <status>
        <direction... />
        <transaction... />
        <cursor_pos... />
        <session... />
        <state... />
        <xmlscreen... />
        <cursor_field... />
        <timestamp... />
    <status />
    <screen>
        <rows... />
        <cols... />
        <type... />
        <model... />
        <fields>
            <field ...>
                <name... />
                <attr... />
                <value... />
            <field />
            .
            .
            .
        <fields />
    <screen />
<TNBRIDGE />
```

TNBRIDGE is the root tag and includes the status and screen tags as children.

status: indicates the connection status
Contains the following tags:

- **direction:** indicates INPUT or OUTPUT direction. When the XML is generated by the XmlBroker control it is set with OUTPUT value.
- **transaction:** indicates transaction identifier.
- **cursor_pos:** indicates cursor location.
- **session:** LU-LU (in session with VTAM application), noSession (not in session) or SSCPLU (in session with VTAM).
- **state:** indicates LOCKED or UNLOCKED state.
- **xmlscreen:** indicates the screen name. You must set this propery using the **ScreenName** property.
- **cursor_field:** indicates the field's name where the cursor is located.
- **timestamp:** indicates the date the file was created.

Screen: indicates screen's characteristics and contains a collection of screen fields. Contains the following tags:

- **rows:** indicates the number of rows of the screen.
- **cols:** indicates the number of columns of the screen.
- **type:** indicates terminal type (3270 or 5250).
- **model:** indicates terminal model.
- **fields:** contains a collection of fields.

Here you can see an example of XML Code:

```
<TNBRIDGE>
     <status>
            <direction>OUTPUT</direction>
            <transaction>00A8DB7000002</transaction>
            <cursor_pos>830</cursor_pos>
            <session>LU-LU</session>
            <state>UNLOCKED</state>
            <xmlscreen name="Session" />
            <cursor_field>R11C30</cursor_field>
            <timestamp>2453360.14494451</timestamp>
     </status>
     <screen>
            <rows>24</rows>
            <cols>80</cols>
            <type>TN3270</type>
            <model>2E</model>
            <fields>
                   <field name="R01C02">
                          <name len="4">R1C2</name>
                          <attr byte="E4" eab="08" display="y" prot="y" numeric="n"
intense="n" mdt="n" />
                          <value maxlen="8" len="8">KLGLGON1</value>
                   </field>
                   <field name="R01C11">
```

```
                                        <name len="5">R1C11</name>
                                        <attr byte="E4" eab="28" display="y" prot="y" numeric="n"
  intense="n" mdt="n" />
                                        <value maxlen="13" len="13">-------------</value>
                                 </field>
                                 .
                                 .
                                 .
                          </fields>
                   </screen>
</TNBRIDGE>
```

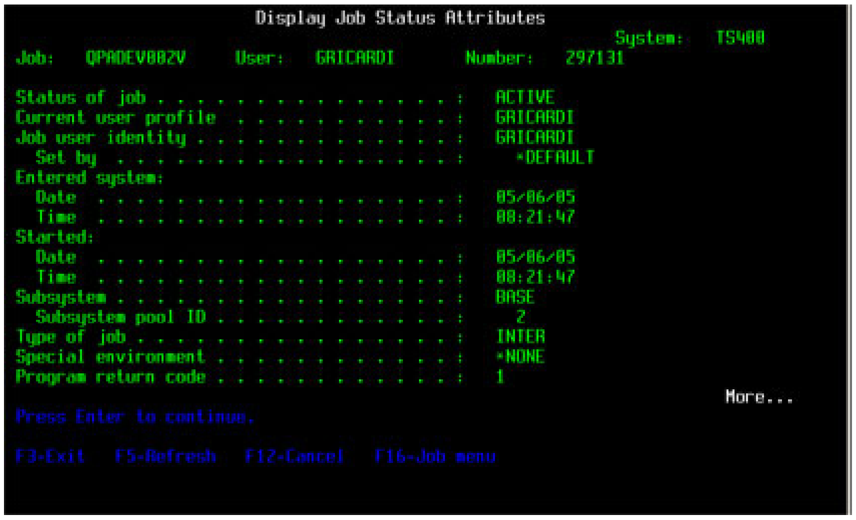
3.4.2.2 Providing a more friendly XML

Generating a more friendly XML involves the use of XMLTemplates. Next we'll see:

- **What is an XML Template?**
- **Using XML Templates**

3.4.2.2.1 What is an XML Template

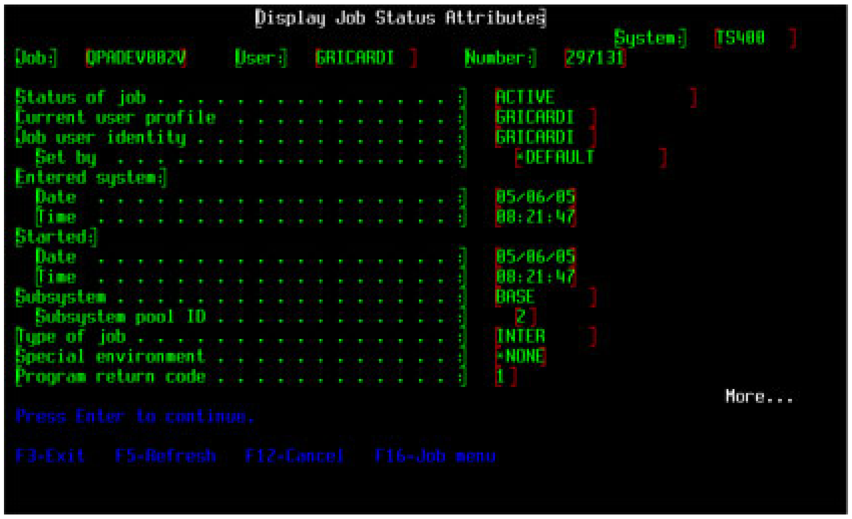
An XML Template is a template file that provides information about how to build an XML file having a particular mainframe screen as data source. XML Templates are built with **Development Lab**. Let's see an example:



```
                         Display Job Status Attributes
                                                          System:   TS400
 Job:   QPADEV002V     User:   GRICARDI       Number:   297131

 Status of job . . . . . . . . . . . . . . . :   ACTIVE
 Current user profile  . . . . . . . . . . . :   GRICARDI
 Job user identity . . . . . . . . . . . . . :   GRICARDI
   Set by  . . . . . . . . . . . . . . . . . :     *DEFAULT
 Entered system:
   Date  . . . . . . . . . . . . . . . . . . :   05/06/05
   Time  . . . . . . . . . . . . . . . . . . :   08:21:47
 Started:
   Date  . . . . . . . . . . . . . . . . . . :   05/06/05
   Time  . . . . . . . . . . . . . . . . . . :   08:21:47
 Subsystem . . . . . . . . . . . . . . . . . :   BASE
   Subsystem pool ID . . . . . . . . . . . . :     2
 Type of job . . . . . . . . . . . . . . . . :   INTER
 Special environment . . . . . . . . . . . . :   *NONE
 Program return code . . . . . . . . . . . . :   1
                                                                More...
 Press Enter to continue.

 F3=Exit   F5=Refresh   F12=Cancel   F16=Job menu
```

When transforming this screen to XML we would like to use tags according what we know are labels and values according what we know that actually are values. As an example, we know that "Status of job" is a label and "ACTIVE" is its value. So, what we would like to see in translated to XML would be something like:
<StatusOfJob>ACTIVE</StatusOfJob>

With the help of Development Lab and our own help, we can define a template to instruct TN BRIDGE to translate the mainframe screen to a readable XML.



```
                         Display Job Status Attributes
                                                      System:   TS400
Job:   QPADEV002V     User:   GRICARDI        Number:   297131

Status of job . . . . . . . . . . . . . . . :   ACTIVE
Current user profile  . . . . . . . . . . . :   GRICARDI
Job user identity . . . . . . . . . . . . . :   GRICARDI
  Set by  . . . . . . . . . . . . . . . . . :     *DEFAULT
Entered system:
  Date  . . . . . . . . . . . . . . . . . . :   05/06/05
  Time  . . . . . . . . . . . . . . . . . . :   08:21:47
Started:
  Date  . . . . . . . . . . . . . . . . . . :   05/06/05
  Time  . . . . . . . . . . . . . . . . . . :   08:21:47
Subsystem . . . . . . . . . . . . . . . . . :   BASE
  Subsystem pool ID . . . . . . . . . . . . :     2
Type of job . . . . . . . . . . . . . . . . :   INTER
Special environment . . . . . . . . . . . . :   *NONE
Program return code . . . . . . . . . . . . :   1
                                                                    More...
Press Enter to continue.

F3=Exit   F5=Refresh   F12=Cancel   F16=Job menu
```

Here we can see how Labels and values were separated by Development Lab and this is the XML we get as result:

```
<Screen>
  <Title>Display Job Status Attributes</Title>
  <System>TS400</System>
  <Job>QPADEV002V</Job>
  <User>GRICARDI</User>
  <Number>297131</Number>
  <StatusOfJob>ACTIVE</StatusOfJob>
  <CurrentUserProfile>GRICARDI</CurrentUserProfile>
  <JobUserIdentity>GRICARDI
       <SetBy>*DEFAULT</SetBy>
  </JobUserIdentity>
  <EnteredSystem>
    <Date>05/06/05</Date>
    <Time>08:21:47</Time>
  </EnteredSystem>
  <Started>
    <Date>05/06/05</Date>
    <Time>08:21:47</Time>
  </Started>
  <Subsystem>BASE
       <SubsystemPoolID>2</SubsystemPoolID>
  </Subsystem>
  <TypeOfJob>INTER</TypeOfJob>
  <SpecialEnvironment>*NONE</SpecialEnvironment>
  <ProgramReturnCode>1</ProgramReturnCode>
</Screen>
```

It mightn't be perfect, but with an additional customization it might!

3.4.2.2.2 Using XML Templates

How do we use XML Templates? First we need to create XML templates for our mainframe screens. These templates are saved into a common directroy. Once we have these templates ready we can use the XmlBroker component and XmlTemplate class as follows:

VB.NET:
```
Private Sub ProduceXml() As String
  'first we instanciate the XmlBroker
  Dim broker As XmlBroker = new XmlBroker(tn3270)

  'then we change its XML producer by assigning an XmlTemplate object
  broker.Producer = new XmlTemplate("c:\My Templates")

  'now, the returned Xml will be produced by the XmlTemplate object and
  'the corresponding template file.
  ProduceXml = broker.Xml
End Sub
```

C#:
```
public string ProduceXml()
{
  // first we instanciate the XmlBroker
  XmlBroker broker = new XmlBroker(tn3270);

  // then we change its XML producer by assigning an XmlTemplate object
  broker.Producer = new XmlTemplate(@"c:\My Templates");

  // now, the returned Xml will be produced by the XmlTemplate object and
  // the corresponding template file.
  return broker.Xml;
}
```

We need to tell to XmlTemplate object the directory where the XML template files are located. Then, when the XmlBroker needs to produce a new XML, it ask to the producer (in this case XmlTemplate) if it can produce XML content from the current screen. Then, XmlTemplate object looks for a template that matchs with the current screen and returns true or false according to it. Next, in case it returned true, it is asked to produce the XML content which finally will be the one delivered by XmlBroker's Xml property; in case it returned false, the XmlBroker uses the default producer to generate XML content, returning a generic XML based on HostFields.

3.4.3 Working with XmlClient

To use the XML data provided by the XmlBroker component, use the XmlClient component.

This component acts like a telnet component and interacts with the other **TN Bridge Host Integration Pack** emulation components in that way.

Its usage can be seen in the following code snippet, which supposes a Form with a Display and a XmlClient components:

VB.NET:
```
Private Sub Form_Load(sender As System.Object, eventArgs As System.EventArgs)
  ' associate the XmlClient component instance to the Telnet property of the
Display component
  tnDisplay1.Telnet = xmlClient1
```

```
        End Sub

        Private Sub ShowProducedXml(ByVal Xml As String)
          ' assigning the Xml property will refresh the Display
          xmlClient1.Xml = Xml
        End Sub
```

C#:

```
        private void Form_Load(object sender, System.EventArgs e)
        {
          // associate the XmlClient component instance to the Telnet property of the
Display component
          tnDisplay1.Telnet = xmlClient1;
        }

        public void ShowProducedXml(string Xml)
        {
          // assigning the Xml property will refresh the Display
          xmlClient1.Xml = Xml;
        }
```

3.4.4 Working with XmlVirtual

The XmlVirtual component allows you to create a simulated host application (no real connection needed) by combining XML-screen files and code to drive the screen navigation.

This component acts like a telnet component and interacts with the other **TN Bridge Host Integration Pack** emulation components in that way.

You can generate the XML-screen files using **Development Lab**.

XmlVirtual component usage can be seen in the following code snippet, which supposes a Form with a Display and a XmlVirtual components and two xml files in the same directory:

VB.NET:

```
        Private Sub Form_Load(sender As System.Object, eventArgs As System.EventArgs)
          ' associate the XmlVirtual component instance to the Telnet property of the
Display component
          tnDisplay1.Telnet = xmlVirtual1
          xmlVirtual1.StartFileName = "start.xml"
          ' when calling Connect method the StartFilename property will be used
          ' to provide the content for the first screen
          xmlVirtual1.Connect()
        End Sub

        Private Sub xmlVirtual1_OnNewScreen(eventSender As System.Object, eventArgs As
System.EventArgs)
          ' when this event is fired, you can analize EditFields and AidKey to
determine
          ' the next screen you should load using LoadScreen method
          xmlVirtual1.LoadScreen("next.xml")
        End Sub
```

C#:

```
        private void Form_Load(object sender, System.EventArgs e)
        {
          // associate the XmlClient component instance to the Telnet property of the
Display component
          tnDisplay1.Telnet = xmlVirtual1;
          xmlVirtual1.StartFileName = "start.xml";
          // when calling Connect method the StartFilename property will be used
          // to provide the content for the first screen
          xmlVirtual1.Connect();
        }

        private void xmlVirtual1_OnNewScreen(object source, System.EventArgs act)
        {
          // when this event is fired, you can analize EditFields and AidKey to
determine
          // the next screen you should load using LoadScreen method
          xmlVirtual1.LoadScreen("next.xml");
        }
```

4 .NET Components

Find in the topics within this section a complete reference for the TNBridge Host Integration Pack .NET components.

- Syntax Conventions
- License Activation
- Mainframe Access Controls
- Terminal Emulator Controls
- Helper Controls
- Trace Services
- Exceptions

4.1 Syntax conventions

The following text formats are used throughout the Components Reference:

For each property and method:

Italic text	Denotes an object, in this case an Integration Pack control.
Bold text	Denotes a property or a method names.
[= *value*]	Denotes values to be assigned.
[*value* =]	Denotes return values.
(*TimeOut*)	Denotes a parameter to be passed to a method.
(*[TimeOut]*)	Denotes an optional parameter to be passed to a method.

4.2 License Activation

By default, **TN Bridge Host Integration Pack for .NET** searchs for the license keys in the same folder where **Cybele.TNBridge.dll** is located, and in the Windows directory.

If there is no license present in these folders, **TNBHIP.NET** starts in DEMO Mode. That means that telnet connections won't last more than 2 minutes.

To activate your copy for trial testing, register for free in our website to obtain a 30-day trial key.

You will receive a file called **Cybele.TNBridge.Trial.xml**, which is used to activate **TNBHIP.NET** for design and runtime for a limited period of time. Place it in any of the two folders mentioned before so you can evaluate all the functionallity of our component suite.

After evaluation, when you purchase a registered copy of **TN Bridge Host Integration Pack for .NET**, you will receive two separated activation keys:

Cybele.TNBridge.Design.xml
- This is used to activate **TNBHIP.NET** for *design-time* and *debug-time*.
- It needs to be present in the developer/s machine/s.
- As any other activation key, it should be placed in any of the two folders mentioned before.

Cybele.TNBridge.Runtime.xml
- This is used to activate **TNBHIP.NET** for *run-time*.
- It needs to be present in the machine/s which run the application or web service using **TNBHIP.NET**.
- As any other activation key, it should be placed in any of the two folders mentioned before.

There is an optional way to set the runtime key: by code. When runtime activating your copy by code, remember to *call any of the two following static methods before* interacting with our components.

Cybele.TNBridge.License.SetRuntimeKey(*string xml*)
- This method receives the **Cybele.TNBridge.Runtime.xml** content as a string parameter.

C# Example:
```
  Cybele.TNBridge.License.SetRuntimeKeyFromFile(@"C:\dir\subdir
\Cybele.TNBridge.Runtime.xml");
```

Cybele.TNBridge.License.SetRuntimeKeyFromFile(*string xmlfile*)
- This method receives the **Cybele.TNBridge.Runtime.xml** specific location (instead of those by default) as a string parameter.

C# Example:
```
  Cybele.TNBridge.License.SetRuntimeKey("<License Type=\"RUNTIME\" Serial=\"TNBN-RT35-
XXXX-XXXXXX-XXXXXX\">"+
  "<User Name=\"Your Name\" Company=\"Your Company, Inc.\" Email=\"user@server.com\" /
>"+
  "<Period First=\"\" Last=\"\" />"+
  "<Version Low=\"3.5\" High=\"3.5\" />"+
  "<Limitations Users=\"1\" Connections=\"100\" />"+
  "<Signature xmlns=\"http://www.w3.org/2000/09/xmldsig#\">"+
  "  <SignedInfo>"+
  "    <CanonicalizationMethod Algorithm=\"http://www.w3.org/TR/2001/REC-xml-c14n-
20010315\" />"+
  "    <SignatureMethod Algorithm=\"http://www.w3.org/2000/09/xmldsig#rsa-sha1\" />"+
  "    <Reference URI=\"\">" +
  "      <Transforms>"+
  "        <Transform Algorithm=\"http://www.w3.org/2000/09/xmldsig#enveloped-
signature\" />"+
  "      </Transforms>"+
  "      <DigestMethod Algorithm=\"http://www.w3.org/2000/09/xmldsig#sha1\" />"+
  "      <DigestValue>xXxXxXxXxXxXxXxXxXxXxXxXxXxX</DigestValue>"+
  "    </Reference>"+
  "  </SignedInfo>"+
  "
<SignatureValue>xXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxX
xXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxX
xXxXxXxXxXxXxXxX</SignatureValue>"+
  "</Signature>"+
  "</License>");
```

4.3 Mainframe Access Controls

Find in the topics within this section a complete reference for the TNBridge Host Integration Pack .NET Mainframe Access Controls.

4.3.1 Tn5250/Tn3270/XmlClient/XmlVirtual Reference

4.3.1.1 Tn5250/Tn3270/XmlClient/XmlVirtual Components

Both controls share a common programming interface, except for a few properties that are specific to one control.

Properties

- **AidKey**
- **CodePage**
- **Cols**
- **ConversionTable**
- **ConnectionName**
- **Debug**
- **DebugDir**
- **DeviceName**
- **EmptyScreenDelay**
- **Extended**
- **History**
- **HistoryCount**
- **HistoryIndex**
- **Host**
- **KeepAlive**
- **MaxHistoryCount**
- **Password**
- **Port**
- **Profiles**

- Rows
- SyncronizeEvents
- TerminalType
- UserId
- WaitTimeOut

Methods

- Connect
- Disconnect
- FromPool
- GetText
- IsConnected
- IsLocked
- IsScreenEmpty
- Press
- PressAndWait
- Release
- SetText
- Type
- Wait
- WaitFor
- WaitForConnect
- WaitForNewScreen
- WaitForScreen
- WaitForNewUnlock
- WaitForUnlock

Events

- OnConnect
- OnConnectionFail
- OnDisconnect
- OnScreenChange
- OnSendAid
- OnSessionState
- OnSystemLock
- OnSystemUnlock

4.3.1.2 Properties

4.3.1.2.1 AidKey

Sets the Attention Identifier Key and sends the modified fields to the Mainframe. Modified fields can be input fields (unprotected) or, sometimes protected fields, having the property Modified set to True.

VB.NET Syntax

*tnxxxx.***AidKey** [= ***AidKey***]

C# Syntax

tnxxxx.**AIDKey** [= ***AidKey***];

It can take any of AidKey constant values:

Constant Value	Meaning
NoOp	No Action
Enter	ENTER key
Clear	CLEAR key
ATTN	Attention key
SysReq	System Request key
PF1	PF01 key
PF2	PF02 key
PF3	PF03 key
PF4	PF04 key
PF5	PF05 key
PF6	PF06 key
PF7	PF07 key
PF8	PF08 key
PF9	PF09 key
PF10	PF10 key
PF11	PF11 key
PF12	PF12 key
PF13	PF13 key
PF14	PF14 key
PF15	PF15 key
PF16	PF16 key
PF17	PF17 key
PF18	PF18 key
PF19	PF19 key
PF20	PF20 key
PF21	PF21 key
PF22	PF22 key
PF23	PF23 key
PF24	PF24 key
PA1	PA1 key
PA2	PA2 key
PA3	PA3 key
Reset	Reset key
PgUp	Page Up key
PgDown	Page Down key
PgRight	Page Right key
PgLeft	Page Left key
Print	Print key
Help	Help key
RecordBackSpace	RecordBackSpace key

See also **AidKey** constants.

4.3.1.2.2 Codepage

Specifies the internal ASCII-EBCDIC conversion table.

VB.NET Syntax

*tnxxxx.***CodePage** [= *String*]

C# Syntax

*tnxxxx.***CodePage** [= *string*];

Remarks

It can take any of the Code Pages constant values:

Constant Value	Meaning
Default = "0"	United States
Australia = "037"	Australian
Austria = "273"	Austria
Austria2 = "1141"	Austria
Belgium = "500"	Belgium
Belgium2 = "1148"	Belgium
Canada = "037"	Canada
Canada2 = "1047"	Canada
Canada3 = "1140"	Canada
CzechRepublic = "2975"	Czech Republic
Denmark = "277"	Denmark
Denmark2 = "1142"	Denmark
Finland = "278"	Finland
Finland2 = "1143"	Finland
France = "297"	France
France2 = "1147"	France
Germany = "273"	Germany
Germany2 = "1141"	Germany
Greece = "423"	Greece
Greece2 = "875"	Greece
Iceland = "871"	Iceland
Iceland2 = "1149"	Iceland
Italy = "280"	Italy
Italy2 = "1144"	Italy
LatinAmerica = "284"	Latin America
LatinAmerica2 = "1145"	Latin America
Netherlands = "037"	Netherlands

Norway = "277"	Norway
Norway2 = "1142"	Norway
Poland = "2978"	Poland
Portugal = "037"	Portugal
Russian = "2979"	Russia
Spain = "284"	Spain
Spain2 = "1145"	Spain
Sweden = "278"	Sweden
Sweden2 = "1143"	Sweden
Switzerland = "500"	Switzerland
Switzerland2 = "1148"	Switzerland
UnitedKingdom = "285"	United Kingdom
UnitedKingdom = "1146"	United Kingdom
UnitedStates = "037"	United States
UnitedStates2 = "037/2"	United States
UnitedStates3 = "1047"	United States
UnitedStates4 = "1140"	United States
International = "850"	International

Default value is UnitedStates.

See also **ConversionTable** property and CodePage constants.

4.3.1.2.3 Cols

Return the total number of columns of the current terminal type.

VB.NET Syntax

[*Integer* =] *tnxxxx*.Cols

C# Syntax

[*int* =] *tnxxxx*.Cols;

Remarks

The Cols property may return 80 or 132 values depending on the terminal type chosen.

See also **Rows** and **TerminalType** properties.

4.3.1.2.4 ConnectionName

Sets/gets the connection name under which will be stored the connection.

VB.NET Syntax

tnxxxx.**ConnectionName** [= *String*]

C# Syntax

*tnxxxx.***ConnectionName** [= *String*];

The default values are "Tn3270E" for Tn3270 Control and "Tn5250" for Tn5250 Control.

Remarks

This property is valid only if a **Profiles** property has been assigned to a **Profiles** Control.

See also **Profiles** property and **Profiles** Component.

4.3.1.2.5 ConversionTable

Specifies an external ASCII-EBCDIC conversion table to be loaded from a file. If no one is specified, an internal conversion table is used, according to **CodePage** property setting.

VB.NET Syntax

*tnxxxx.***ConversionTable** [= *String*]

C# Syntax

*tnxxxx.***ConversionTable** [= *string*];

Remarks

The file specified must exist and it might contain a valid conversion table format.

The format of the conversion table is:

```
[Ascii-To-Ebcdic]
Ascii hexadecimal code 1   = Ebcdic hexadecimal code 1
Ascii hexadecimal code 2   = Ebcdic hexadecimal code 2
            ..                          ..
Ascii hexadecimal code n   = Ebcdic hexadecimal code n

[Ebcdic-To-Ascii]
Ebcdic hexadecimal code 1 = Ascii hexadecimal code 1
Ebcdic hexadecimal code 2 = Ascii hexadecimal code 2
            ..                          ..
Ebcdic hexadecimal code n = Ascii hexadecimal code n
```

See also **CodePage** property.

4.3.1.2.6 Debug

Allows the generation of trace information for debugging purposes.
The debugging can help you to track down both, errors and logic information.

VB.NET Syntax

*tnxxxx.***Debug** [= *Boolean*]

C# Syntax

*tnxxxx.***Debug** [= *bool*];

The following are possible values for Debug property:

Constant Value	**Meaning**
True	Enables debug information.
False	Disables debug information.

Remarks

By default, the debug file is saved in the application directory. You can change the directory by setting the **DebugDir** property.

Default value is False.

See also **DebugDir** property.

4.3.1.2.7 DebugDir

Specifies the directory for debug files.

VB.NET Syntax

*tnxxxx.***DebugDir** [= *String*]

C# Syntax

*tnxxxx.***DebugDir** [= *string*];

Remarks

By default, the debug file is saved in the application's directory. You can change it by setting this property to another directory. This directory must exists, otherwise, no debug information will be generated and an error will not occur.

The default value is the application directory.

See also **Debug** property.

4.3.1.2.8 DeviceName

Sets an specific telnet device name defined in the telnet server.

VB.NET Syntax

*tnxxxx.***DeviceName** [= *String*]

C# Syntax

*tnxxxx.***DeviceName** [= *string*];

Remarks

If no name is given, the telnet server will assign an available device from a public pool in case of it exists.

See also **UserId** and **Password** properties.

4.3.1.2.9 EditFields

Returns an array containing the editable screen fields.

VB.NET Syntax

[***Field*** =] *tnxxxx.***EditFields (***Index As Integer***)**

C# Syntax

[***Field*** =] *tnxxxx.***EditFields [***int index***]***;*

Remarks

This list includes only the unprotected **Fields** of the mainframe's screen reformated to fit in standard gui elements; when a field goes beyond the number display size in columns, the field is splitted in as many fields as necessary with a maximun length of the number of display columns.

If you specified a wrong index value, a null value is returned.

See also **HostFields** property, **Fields** and **Field** classes.

4.3.1.2.10 EmptyScreenDelay

Sets/gets the additional wait time to bypass empty host screens. This time is measured in milliseconds.

VB.NET Syntax

*tnxxxx.***EmptyScreenDelay** [= *Integer*]

C# Syntax

*tnxxxx.***EmptyScreenDelay** [= *int*];

Remarks

When you make a call to the **Wait** method, if an empty screen arrives from the host and the system is in an unlocked state, an internal **WaitForNewScreen** is made with a timeout taken from this property value. It allows to bypass intermediate empty screens without additional programming effort.
Applies only to the **Wait** method and other methods that call a Wait internally like **Connect** and **Type** methods.

Default value is 0.

See also **Connect**, **Type** and **Wait** methods.

4.3.1.2.11 ExcludeEmptyFields

Controls whether the Tn3270/Tn5250 component will exclude or not protected fields that has no content.

VB.NET Syntax

*tnxxxx.***ExcludeEmptyFields** [= *Boolean*]

C# Syntax

*tnxxxx.***ExcludeEmptyFields** [= *bool*];

Remarks

Default value is False.

See also **HostFields** property.

4.3.1.2.12 Extended

Specifies if the telnet 5250 extended protocol is allowed by this client.

VB.NET Syntax

*tnxxxx.***Extended** [= *Boolean*]

C# Syntax

*tnxxxx.***Extended** [= *bool*];

Remarks

If the host or telnet server supports the Tn5250E protocol, it will try to negotiate with this client to work in extended mode. If this property is set to False, this client will deny the request from the server and both will continue in Tn5250 mode. Default value is True.

See also **Host**, **Port** and **TerminalType** property.

4.3.1.2.13 History

The History keeps track of all visited screens. This method returns one of the history screens fields collection.

VB.NET Syntax

[***Fields*** =] *tnxxxx.*History **(***Index As Integer***)**

C# Syntax

[***Fields*** =] *tnxxxx.*History **[***int index***]***;*

Remarks

The HistoryIndex 0 (zero) corresponds to the latest screen, the HistoryIndex 1 (one) is the previous one and so on. In order to get the first screen in the history you should access the HistoryIndex which corresponds to the **HistoryCount** minus 1 (HistoryCount -1) .

See also HistoryCount, HistoryIndex and MaxHistoryCount properties.

4.3.1.2.14 HistoryCount

This property indicates how many screens are currently saved in the history

VB.NET Syntax

*tnxxxx.*HistoryCount [= *Integer*]

C# Syntax

*tnxxxx.*HistoryCount [= *int*];

Remarks

See Also History, HistoryIndex and MaxHistoryCount properties.

4.3.1.2.15 HistoryIndex

This property indicates the index of the currently active screen. Through this property you will be able to change the active screen.

VB.NET Syntax

*tnxxxx.*HistoryIndex [= *Integer*]

C# Syntax

*tnxxxx.*HistoryIndex [= *int*];

Remarks

See Also History, HistoryCount and MaxHistoryCount properties.

4.3.1.2.16 Host

TCP/IP address or DNS name of the Host to connect to. It can be a name like 'myhost.mydomain.com' or directly an IP address such as '206.155.164.20'.

VB.NET Syntax

*tnxxxx.***Host** [= *String*]

C# Syntax

*tnxxxx.***Host** [= *string*];

Remarks

The host name stored in the Host property must be alpha numeric, and can't contain spaces nor special characters (i.e. : " / ? \)

See also **Port** property, **Connect** method and **OnConnect** event.

4.3.1.2.17 HostFields

Gets an array containing the reformated screen fields.

VB.NET Syntax

[***Field*** =] *tnxxxx.***HostFields (***Index As Integer***)**

C# Syntax

[***Field*** =] *tnxxxx.***HostFields [***int index***];**

Remarks

This list includes all **Fields** (protected and unprotected) of the mainframe's screen reformated to fit in standard gui elements; when a field goes beyond the number display size in columns, the field is splitted in as many fields as necessary with a maximun length of the number of display columns.

If you specify a wrong index value, a null value is returned.

See also **EditFields** property, **Fields** and **Field** classes.

4.3.1.2.18 FieldSplitting

Determines the format of **Fields** in **HostFields** and **EditFields** colections.

VB.NET Syntax

*tnxxxx.***FieldSplitting** [= *bool*]

C# Syntax

*tnxxxx.***FieldSplitting** [= *bool*];

Remarks

FieldSplitting property controls how fields in HostFields and EditFields collections are constructed. When active, fields that go beyond the maximun of display column are splitted in two or more fields.

Default is true.

See also **Field** class, **HostFields** and **EditFields** properties.

4.3.1.2.19 KeepAlive

Allows to enable a keep-alive mechanism used by some telnet servers.

VB.NET Syntax

*tnxxxx.***KeepAlive** [= *Boolean*]

C# Syntax

*tnxxxx.***KeepAlive** [= *bool*];

Remarks

Default value for telnet 3270 server (IBM S/3XX mainframes) is True.
Default value for telnet 5250 server (IBM AS/400 mainframes) is **False**.

4.3.1.2.20 MaxHistoryCount

This property indicates the maximum number of screens stored by the history.

VB.NET Syntax

*tnxxxx.*MaxHistoryCount [= *Integer*]

C# Syntax

*tnxxxx.*MaxHistoryCount [= *int*];

Remarks

See Also History, HistoryIndex and HistoryCount properties.

4.3.1.2.21 Password

Sets/gets a Password for the UserId specified.

VB.NET Syntax

*tnxxxx.***Password** [= *String*]

C# Syntax

*tnxxxx.***Password** [= *string*];

See also **DeviceName** and **UserId** properties.

4.3.1.2.22 Port

TCP Port of the Tn3270/Tn5250 host to connect to.

VB.NET Syntax

*tnxxxx.*Port [= *Integer*]

C# Syntax

*tnxxxx.*Port [= *int*];

Remarks

By default it takes the standard telnet port. You can change it if the non standard port is to be used.
Default value is 23.

In XmlClient and XmlVirtual components this property is ignored and it's value is 0.

See also **Host** property, **Connect** method and **OnConnect** event.

4.3.1.2.23 Profiles

Sets the **Profiles** Control as the profile manager for this control.

VB.NET Syntax

*tnxxxx.***Profiles** [= *Profiles*]

C# Syntax

*tnxxxx.***Profiles** [= *Profiles*];

Remarks

After setting this property, you can access to a collection of connections to be generated and customized through the **ShowEditor** method.

See Also Profiles control and **ShowEditor** method.

4.3.1.2.24 Rows

Return the total number of rows of the current terminal type.

VB.NET Syntax

[*Integer* =] *tnxxxx*.Rows

C# Syntax

[*int* =] *tnxxxx*.Rows;

Remarks

Depending on the terminal type chosen, this property returns the values 24, 32 or 43.

See Also **Cols** and **TerminalType** property.

4.3.1.2.25 SessionState

Returns the current session state.

VB.NET Syntax

[*SessionState* =] *tnxxxx*.SessionState

C# Syntax

[*SessionState* =] *tnxxxx*.SessionState;

Remarks

While the telnet connection is closed, this property returns always NoSession value.

When the telnet connection is opened it can return:

- NoSession, indicating that the connection was established at telnet level but not at 5250 Data Stream level.
- SSCPLU, indicating that a connection with the VTAM was established, but there's not a session with an final application (like CICS, TSO, etc.). (This state is only valid for TN3270 Hosts).
- LULU, indicating that the connection with the VTAM application (like CICS, TSO, etc.) has been established.

See also SessionState constants.

4.3.1.2.26 SynchronizeEvents

Controls whether synchronization of events with a main window thread is required or not.

VB.NET Syntax

*tnxxxx.***SynchronizeEvents** [= *Boolean*]

C# Syntax

*tnxxxx.***SynchronizeEvents** [= *boolean*];

Remarks

In windowed applications, events fired in another thread different from the main-window thread need to be synchronized to prevent corruption in the gui framework. To avoid loss of events in non-windowed applications, this property should be set to false.

Default value is True.

4.3.1.2.27 TerminalType

Sets/gets the terminal type.

VB.NET Syntax

*tnxxxx.***TerminalType** [= ***TerminalModel***]

C# Syntax

*tnxxxx.***TerminalType** [= ***TerminalModel***];

It can take any of the following constant values:

For 5250:

Constant Value	Meaning
0	IBM5211m2
1	IBM3179m2
2	IBM3477mFC
3	IBM3180m2
4	IBM3477mFG
5	IBM3196mA1
6	IBM5292m2
7	IBM5291m1
8	IBM5251m11

For 3270:

Constant Value	Meaning
0	IBM3278m2
1	IBM3278m2E
2	IBM3278m3
3	IBM3278m3E
4	IBM3278m4
5	IBM3278m4E
6	IBM3278m5
7	IBM3278m5E

See also **TerminalModel** for Tn5250 and **TerminalModel** for Tn3270 constants.

4.3.1.2.28 Text

Returns the entire screen.

VB.NET Syntax

[*String* =] *tnxxxx*.**Text**

C# Syntax

[*string* =] *tnxxxx*.**Text**;

See also **GetText** method.

4.3.1.2.29 UserId

Sets/gets a UserId for the device name specified.

VB.NET Syntax

*tnxxxx.***UserId** [= *String*]

C# Syntax

*tnxxxx.***UserId** [= *string*];

See also **DeviceName** and **Password** properties.

4.3.1.2.30 WaitTimeOut

Sets/gets the default timeout value in milliseconds for all the wait methods.

VB.NET Syntax

*tnxxxx.***WaitTimeout** [= *Integer*]

C# Syntax

*tnxxxx.***WaitTimeout** [= *int*];

Remarks

You can assign any value, there isn't a limit of maximum value.

Default value is 60000 milliseconds (equivalent to 1 minute).

4.3.1.3 Methods

4.3.1.3.1 Connect

The Connect method establishes the connection to the Telnet server. The **Host** and **Port** properties must be assigned for the connection to be successfully established.

VB.NET Syntax

*tnxxxx.*Connect

[Boolean] = *tnxxxx.*Connect(Timeout as Integer)

C# Syntax

*tnxxxx.*Connect();

[bool] = *tnxxxx.*Connect(int Timeout);

Remarks

Try to establish the connection to the specified **Host** and **Port**. In first form, this is done in asynchronous fashion, returning the control inmediatelly. In second form, it blocks the code execution waiting until the connection is succesfully established or the specified timeout expires.

If the specified host doesn't exist or if the connection fails, the OnConnectionFail event is fired. If the connection is successfully established, the **OnConnect** event is fired.

See also **Host** and **Port** properties, **Disconnect** method, **OnConnect** and **OnConnectionFail** events.

4.3.1.3.2 Disconnect

The Disconnect method ends the connection with the Telnet server.

VB.NET Syntax

*tnxxxx.*Disconnect

C# Syntax

*tnxxxx.*Disconnect();

Remarks

After the client disconnects, the OnDisconnect event is fired. If the client is not connected to any server, the disconnect method has no effect.

See also **Connect** method and **OnDisconnect** event.

4.3.1.3.3 FromPool

Gets a free Tn3270/Tn5250 object from a pool of objects.

VB.NET Syntax

*tnxxxx.***FromPool(**poolId as String,[sessionId as String],[InactivityTimeout as Integer]**)**

C# Syntax

*tnxxxx.***FromPool(**string poolId,[string sessionId],[int InactivityTimeout]**);**

Remarks

Gets a Tn3270/Tn5250 object from a pool named "poolId". If there's no a free object in the named pool, it creates a new one, adds it to the pool and returns the newly created object.

If the sessionId parameter is specified, the specific associated object is returned. Under ASP.NET environment, each Telnet object has associated the Session's SessionId string. Under ASP.NET, calling this method passing the Sessions's SessionId as second parameter it is a safe way to get access to the same telnet object from one page to another under the same ASP.NET session.

InactivityTimeout parameter indicates, in minutes, how much time the telnet object will remain in the pool without activity. Default timeout is 5 minutes.

To return an object to the pool you must call Release method.

See also **Release** method.

4.3.1.3.4 GetText

GetText returns the text buffer of a portion of the current screen. You can receive the attribute or extended attribute of each field together to the character data.

VB.NET Syntax

[*String* =] *tnxxxx.*GetText (*StartPos As Integer, [EndPos As Integer], [Attr As Boolean = False], [Eab As Boolean = False]*)

[*String* =] *tnxxxx.*GetText (*Row As Integer, Col As Integer, Len as Integer,[Attr As Boolean = False], [Eab As Boolean = False]*)

C# Syntax

[*string* =] *tnxxxx.*GetText (*int StartPos, [int StartPos], [bool Attr = False], [bool Eab = False]*);

[*string* =] *tnxxxx.*GetText (*int Row, in Col, int Len, [bool Attr = False], [bool Eab = False]*);

Remarks

Start position and the end position of the text buffer to retrieve can be set using StartPos and EndPos parameters or Row,Col and Len parameters.

The Attr parameter will indicate to GetText that includes the field attribute information. The Attr describes the field properties and occupies the first character position of each field in the character buffer and on the screen. This character is in HLLAPI-Compliance format.

The Eab parameter will indicate to GetText that includes the extended attribute information. The Eab is associated with individual characters. Each character in the buffer (except for fields attributes) has a character attribute. In the returned String, each character will be preceded by his Eab value. This character is in HLLAPI-Compliance format.

4.3.1.3.5 IsConnected

Returns a boolean value indicating if a connection is currently established with the Host.

VB.NET Syntax

[*Boolean* =] *tnxxxx.***IsConnected**

C# Syntax

[*bool* =] *tnxxxx.***IsConnected()**;

See also **Connect** and **Disconnect** methods.

4.3.1.3.6 IsLocked

Returns a boolean value indicating if the Host is in a locked state.

VB.NET Syntax

[*Boolean* =] *tnxxxx.***IsLocked**

C# Syntax

[*bool* =] *tnxxxx.***IsLocked()**;

4.3.1.3.7 IsScreenEmpty

Returns a boolean value indicating in the screen has all the fields with no content or in filled with space characters.

VB.NET Syntax

[*Boolean* =] *tnxxxx.***IsScreenEmpty**

C# Syntax

[*bool* =] *tnxxxx.***IsScreenEmpty()**;

4.3.1.3.8 Press

Sets an Attention Identifier Key and sends the modified fields to the Mainframe. Modified fields can be input fields (unprotected) or sometimes protected fields, having the property Modified set to True.

VB.NET Syntax

[*Boolean* =] *tnxxxx.*Press (*Value As string*)

C# Syntax

[*bool* =] *tnxxxx.*Press (*string Value*);

This method works like the AidKey property, but it takes a string with the name of the AID key as a parameter. You can specify only one key for each call to the method.

The following table lists the possible string values:

String Value	Meaning
Enter	ENTER key
Clear	CLEAR key
ATTN	Attention key
SysReq	System Request key
PF1	PF01 key
PF2	PF02 key
PF3	PF03 key
PF4	PF04 key
PF5	PF05 key
PF6	PF06 key
PF7	PF07 key
PF8	PF08 key
PF9	PF09 key
PF10	PF10 key
PF11	PF11 key
PF12	PF12 key
PF13	PF13 key
PF14	PF14 key
PF15	PF15 key
PF16	PF16 key
PF17	PF17 key
PF18	PF18 key
PF19	PF19 key
PF20	PF20 key
PF21	PF21 key
PF22	PF22 key
PF23	PF23 key
PF24	PF24 key
PA1	PA1 key

See also **AIDKey** property.

4.3.1.3.9 PressAndWait

Sends an Attention Identifier Key along with all modified fields to the Mainframe and waits until the system become unlocked.

VB.NET Syntax

[*Boolean* =] *tnxxxx.*PressAndWait(*Value As string,[TimeOut As Integer]*)

C# Syntax

[*bool* =] *tnxxxx.*PressAndWait (*string Value,[int TimeOut]*);

This method works as a combination of Press and WaitForUnlock methods. Timeout is an optional parameter and specify how much time in milliseconds the call will stop waiting for an unlock signal. If not is specified, the timeout will take the value of WaitTimeout property.

Returns true if the

See also Press and WaitForUnlock methods and WaitTimeout property.

4.3.1.3.10 Release

Returns a Tn3270/Tn5250 object to the pool of telnet objects.

VB.NET Syntax

*tnxxxx.***Release**

C# Syntax

*tnxxxx.***Release();**

Remarks

Returns a Tn3270/Tn5250 object to the pool of telnet objects. Used in combination with **FromPool** method.

See also **FromPool** method.

4.3.1.3.11 SetText

SetText allows to put text information into edit buffer.

VB.NET Syntax

*tnxxxx.*SetText (*StartPos As Integer, String text*)

*tnxxxx.*SetText (*Row As Integer, Col As Integer, String text*)

C# Syntax

*tnxxxx.*SetText (int *StartPos, string text*)

*tnxxxx.*SetText (int *Row, int Col, string text*)

Remarks

SetText allows to put text information into edit buffer, starting according to position StartPos or Row and Col parameters.

4.3.1.3.12 Type

Type can be used to send key sequences to the mainframe (including AID keys). By using special codes you can send several kinds of keys. These codes consist of an Escape character (default is "@") and a mnemonic code that corresponds to the supported host functions.

VB.NET Syntax

*tnxxxx.*Type (*DataString As String*)

C# Syntax

*tnxxxx.*Type (*string DataString*);

The following table lists the functions keys and its corresponding codes.

Code Value	Meaning
@A@Q	Attention
@<	Backspace
@B	BackTab (Left Tab)
@C	Clear
@E	Enter
@F	Erase Field
@A@Q	Sys Request
@T	Tab (Right Tab)
@x	PA1
@y	PA2
@z	PA3
@1	PF1
@2	PF2
@3	PF3
@4	PF4
@5	PF5
@6	PF6
@7	PF7
@8	PF8
@9	PF9
@a	PF10
@b	PF11
@c	PF12
@d	PF13
@e	PF14

@f	PF15
@g	PF16
@h	PF17
@i	PF18

Here is an example of the *Keys* parameter:

```
"logon applid(cics)@E"
```

Remarks

The execution of the SendKeys is stopped when an AID Key is sent to the Host.

See also **AIDKey** property.

4.3.1.3.13 Wait

General wait mechanism. This method basically waits until the mainframe application turns in an unlocked state and is able to receive input data.

VB.NET Syntax

[Boolean] = *tnxxxx.*Wait (*[TimeOut] As Integer*)

C# Syntax

[bool] = *tnxxxx.*Wait (*int [TimeOut]*);

The *TimeOut* parameter is measured in milliseconds.

Remarks

The Timeout parameter is optional, assuming the **WaitTimeout** property value by default. This method returns True if the wait was successful and False if the timeout period has expired.
This method uses alternatively the **WaitForConnect**, **WaitForUnlock** and **WaitForNewScreen** methods, according to the actual mainframe state (disconnected, locked or unlocked).
If you pass a timeout value in the Timeout parameter of this method, the value of the property WaitTimeout will not have any effect.

See also **WaitTimeout** property, **WaitForConnect**, **WaitForUnlock** and **WaitForNewScreen** methods.

4.3.1.3.14 WaitFor

Wait for an screen containing the specified string.

VB.NET Syntax

*[Boolean] = tnxxxx.*WaitFor (*StrValue As String, [TimeOut As Integer]*)

*[Integer] = tnxxxx.*WaitFor (*StrValues As Array of String, [TimeOut As Integer]*)

C# Syntax

*[bool] = tnxxxx.*WaitFor (*string StrValue, [int TimeOut]*);
*[int] = tnxxxx.*WaitFor (*string[] StrValues, [int TimeOut]*);

The *TimeOut* parameter is measured in milliseconds.

Remarks

The Timeout parameter is optional, assuming the **WaitTimeout** property value by default. In first form, this method returns True if the wait was successful and False if the timeout period has expired. In second one, returns the index of the string in StrValues that was found at the current screen and -1 if the timeout period has expired.

If you pass a timeout value in the Timeout parameter of this method, the value of the property **WaitTimeout** will not have any effect.

See also **WaitTimeout** property, **WaitForScreen**, **WaitForUnlock** and **WaitForNewScreen** methods.

4.3.1.3.15 WaitForConnect

This method waits until the telnet connection is successfully opened, or the timeout period expires.

VB.NET Syntax

*[Boolean] = tnxxxx.*WaitForConnect (*[TimeOut As Integer]*)

C# Syntax

*[bool] = tnxxxx.*WaitForConnect (*[int TimeOut]*);

The *TimeOut* parameter is measured in milliseconds.

Remarks

The Timeout parameter is optional, assuming the **WaitTimeout** property value by default. This method returns True if the wait was successful and False if the timeout period has expired.

If you pass a timeout value in the Timeout parameter of this method, the value of the property **WaitTimeout** will not have any effect.

See also **WaitTimeout** property and **Wait** method.

4.3.1.3.16 WaitForNewScreen

This method waits until a new screen arrives within the specified period of time.

VB.NET Syntax

[Boolean] = *tnxxxx.*WaitForNewScreen (*[TimeOut As Integer]*)

C# Syntax

[bool] = *tnxxxx.*WaitForNewScreen (*[int TimeOut]*);

The *TimeOut* parameter is measured in milliseconds.

Remarks

This method is similar to **WaitForScreen** method except that it always waits for a new screen arrival.
The Timeout parameter is optional, assuming the **WaitTimeout** property value by default. This method returns True if the wait was successful and False if the timeout period has expired.
If you pass a timeout value in the Timeout parameter of this method, the value of the property **WaitTimeout** will not have any effect.

See also **WaitTimeout** property and **WaitForScreen** methods.

4.3.1.3.17 WaitForScreen

This method waits for the first screen after a system lock.

VB.NET Syntax

[Boolean] = *tnxxxx.*WaitForScreen (*[TimeOut As Integer]*)

C# Syntax

[bool] = *tnxxxx.*WaitForScreen (*[int TimeOut]*);

The *TimeOut* parameter is measured in milliseconds.

Remarks

If the host system changes from an unlocked to a locked state (normally when we send an Aid key) the method waits until the first screen arrives. If WaitForScreen method is called after that event, it returns immediately.
The Timeout parameter is optional, assuming the **WaitTimeout** property value by default. The method returns True if the wait was successful and False if the timeout

period has expired.
If you pass a timeout value in the Timeout parameter of this method, the value of the property **WaitTimeout** will not have any effect.

See also **WaitTimeout** property and **WaitForNewScreen** method.

4.3.1.3.18 WaitForNewUnlock

This method waits until a new system unlock arrives within the specified period of time.

VB.NET Syntax

[Boolean] = *tnxxxx.*WaitForNewUnlock (*[TimeOut As Integer]*)

C# Syntax

[bool] = *tnxxxx.*WaitForNewUnlock (*[int TimeOut]*);

The *TimeOut* parameter is measured in milliseconds.

Remarks

This method is similar to **WaitForUnlock** method except that it always waits for a new system unlock arrival.
The Timeout parameter is optional, assuming the **WaitTimeout** property value by default. This method returns True if the wait was successful and False if the timeout period has expired.
If you pass a timeout value in the Timeout parameter of this method, the value of the property **WaitTimeout** will not have any effect.

See also **WaitTimeout** property and **WaitForUnlock** method.

4.3.1.3.19 WaitForUnlock

This method waits until the host system turns to an unlocked state.

VB.NET Syntax

[Boolean] = *tnxxxx.*WaitForUnlock (*[TimeOut As Integer]*)

C# Syntax

[bool] = *tnxxxx.*WaitForUnlock (*[int TimeOut]*);

The *TimeOut* parameter is measured in milliseconds.

Remarks

If the host system is in an unlock state this method returns immediately. If the host system is in a lock state, the method waits until a system unlock event arrives (**OnSystemUnlock**).
The Timeout parameter is optional, assuming the **WaitTimeout** property value by default. The method returns True if the wait was successful and False if the Timeout period has expired.
If you pass a timeout value in the Timeout parameter of this method, the value of the property **WaitTimeout** will not have any effect.

See also **WaitTimeout** property and **WaitForNewUnlock** method.

4.3.1.4 Events

4.3.1.4.1 OnConnect

Occurs after a successfully connection to the server is established.

See also **Connect** method and, **OnConnectionFail** and **OnDisconnect** events.

4.3.1.4.2 OnConnectionFail

Occurs after the connection to the server fails.

See also **Connect** method, **OnConnect** and **OnDisconnect** events.

4.3.1.4.3 OnDisconnect

Occurs after a disconnection of the server.

See also **Disconnect** method.

4.3.1.4.4 OnScreenChange

Occurs after a new data screen has arrived.

See also **OnSystemLock** and **OnSystemUnlock** events.

4.3.1.4.5 OnSendAid

Occurs before an Aid key is to be sent.

Remarks

This event is fired when a **Press** or **Type** methods or the **AidKey** property are used to send an Aid Key and data to the mainframe. Occurs before the data is sent, allowing to take actions like filling edit fields or saving data typed in the emulator control.

See also **Press**, **Type** methods and **AidKey** property.

4.3.1.4.6 OnSessionState

Occurs when a session-state change takes place.

Remarks

A session state can be NoSession, SsCplu and LuLu. Any transition between the states fires this event.

See also **SessionState** property.

4.3.1.4.7 OnSystemLock

Occurs when a terminal changes to a system locked state.

Remarks

During this state, the terminal is waiting for any response from the mainframe. Sending data isn't possible until the **OnSystemUnlock** event is fired.

See also **OnSystemUnlock** event.

4.3.1.4.8 OnSystemUnlock

Occurs when a terminal changes to a system unlocked state.

Remarks

Only during this state, the component can send data to the host system.

See also **OnSystemLock** and **OnScreenChange** events.

4.3.2 Tn5250/Tn3270

Specific properties, methods and events for Tn5250/Tn3270 controls.

4.3.2.1 Properties

4.3.2.1.1 EditFields

Array containing the editable screen fields.

VB.NET Syntax

[*Field* =] *tnxxxx.***EditFields (***Index As Variant***)**

C# Syntax

[*Field* =] *tnxxxx.***EditFields (***Index:Variant***);**

Remarks

This list includes only the unprotected **Fields** of the mainframe's screen. If you specified a wrong index value, a null value is returned.

See also **HostFields** property, **Fields** and **Field** Classes.

4.3.2.1.2 HostFields

Array containing the screen fields.

VB.NET Syntax

[*Field* =] *tnxxxx.***HostFields (***Index As Variant***)**

C# Syntax

[*Field* :=] *tnxxxx.***HostFields [***Index:Variant***]***;*

Remarks

This list includes all **Fields** (protected and unprotected) of the mainframe's screen. If you specified a wrong index value, a null value is returned.

See also **EditFields** property, **Fields** and **Field** classes.

4.3.2.1.3 IsExtended

Returns a boolean value indicating if the current connection uses Tn3270E or Tn5250E capabilities.

VB.NET Syntax

[*Boolean* =] *tnxxxx.***IsExtended**

C# Syntax

[*Boolean* :=] *tnxxxx.***IsExtended**;

See also **Extended** property.

4.3.2.1.4 LastErrMsg

Returns a string that specifies the last communication error reported by the host in the telnet current connection.

VB.NET Syntax

[*String* =] *tnxxxx.***LastErrMsg**

C# Syntax

[*String* :=] *tnxxxx.***LastErrMsg**;

4.3.2.1.5 SSL

Allows to enable the SSL protocol (Secure Sockets Layer).

VB.NET Syntax

*tnxxxx.***SSL** [= *Boolean*]

C# Syntax

*tnxxxx.***SSL**; [:= *Boolean*]

See also SSLMethod, SSLDisplay, SSLCertFile properties.

4.3.2.1.6 SSLMethod

Sets/gets the supported SSL method.

VB.NET Syntax

*tnxxxx.***SSLMethod** [= ***SSLMethodType***]

C# Syntax

*tnxxxx.***SSLMethod**; [:= ***SSLMethodType***]

See also SSL, SSLCertPassword, SSLCertFile properties and **SSLMethodType** constants.

4.3.2.1.7 SSLEnableDialogs

Allows to enable SSL dialogs mechanism.

VB.NET Syntax

*tnxxxx.***SSLEnableDialogs** [= *Boolean*]

C# Syntax

*tnxxxx.***SSLEnableDialogs**; [:= *Boolean*]

4.3.2.1.8 SSLRootCertFile

Sets/gets the SSL root's certification file.

VB.NET Syntax

*tnxxxx.***SSLRootCertFile** [= *String*]

C# Syntax

*tnxxxx.***SSLRootCertFile**; [:= *String*]

See also SSL, SSLCertPassword, SSLMethod and SSLCertFile.

4.3.2.1.9 SSLCertFile

Sets/gets the SSL Certificate file.

VB.NET Syntax

*tnxxxx.***SSLCertFile** [= *String*]

C# Syntax

*tnxxxx.***SSLCertFile**; [:= *String*]

See also SSL, SSLMethod, SSLCertPassword properties.

4.3.2.1.10 SSLKeyFile

Sets/gets the SSL key file.

VB.NET Syntax

*tnxxxx.***SSLKeyFile** [= *String*]

C# Syntax

*tnxxxx.***SSLKeyFile**; [:= *String*]

See also SSL, SSLMethod, SSLCertPassword, SSLCertFile and **SSLKeyFile** properties.

4.3.2.1.11 SSLCertPassword

Sets/gets the SSL Certificate password.

VB.NET Syntax

*tnxxxx.***SSLCertPassword** [= *String*]

C# Syntax

*tnxxxx.***SSLCertPassword**; [:= *String*]

See also SSL, SSLMethod, SSLCertFile properties.

4.3.2.1.12 XLockDelay

Controls the delay in milliseconds between the last OnScreenChange and OnSystemUnlock events.

VB.NET Syntax

tnxxxx.XLockDelay [= *Integer*]

C# Syntax

tnxxxx.XLockDelay [= *int*];

4.3.2.2 Methods

Enter topic text here.

4.3.2.2.1 SetCertificateOptions

Sets the SSL options used to validate the certificates.

VB.NET Syntax

*tnxxxx.*SetCertificateOptions (AcceptSelfSigned As Boolean, AcceptExpired As Boolean, AcceptInvalidCA As Boolean, AcceptAnyInvalid As Boolean, AcceptNotYetValid As Boolean)

C# Syntax

*tnxxxx.*SetCertificateOptions (bool AcceptSelfSigned, bool AcceptExpired, bool AcceptInvalidCA, bool AcceptAnyInvalid, bool AcceptNotYetValid);

Parameters	Meaning
AcceptSelfSigned	Allows to accept or not self signed certificates.
AcceptExpired	Allows to accept or not expired certificates.
AcceptInvalidCA	Allows to accept or not any invalid CA certificate.
AcceptAnyInvalid	Allows to accept or not any invalid certificate.
AcceptNotYetValid	Allows to accept or not certificates not yet valid.

Display	Enable/disable the display certificate mode.

4.3.2.3 Events

Enter topic text here.

4.3.2.3.1 OnMessageWaitingOff

Occurs when the Message Waiting status of the status bar turns to off.

See also OnMessageWaitingOn event.

4.3.2.3.2 OnMessageWaitingOn

Occurs when the Message Waiting status of the status bar turns to on.

See also OnMessageWaitingOff event.

4.3.2.4 Constants

Enter topic text here.

4.3.2.4.1 SSLMethodType

These values are used for the SSLMethodType property of Tn5250 control.

Constant Value	Meaning
Default	Supports SSL 2.0 and 3.0 version.
SSL2	Supports SSL 2.0 version.
SSL3	Supports SSL 3.0 version.
TLS	Supports TLS 1.0 version.

4.3.3 Field Class Reference

4.3.3.1 Field Class

Implements a Screen field object.

Properties

- **Attr**
- **AutoEnter**
- **Blinking**
- **ByPass**
- **Col**
- **Color**
- **Data**
- **Eab**
- **Editable**

- **High**
- **Len**
- **Mandatory**
- **Modified**
- **MonoCase**
- **Name**
- **Numeric**
- **Pos**
- **Reverse**
- **Row**
- **Skip**
- **Underline**
- **Visible**
- **HllApiAttr**
- **HllApiEab**

4.3.3.2 Properties

4.3.3.2.1 Attr

Sets/gets the attribute of a Field.

VB.NET Syntax

*Field.***Attr** [= ***FieldAttribute***]

C# Syntax

*Field.***Attr** [= ***FieldAttribute***];

Remarks

This property is for reserved use.

See Also **FieldAttribute** constants.

4.3.3.2.2 AutoEnter

Returns true, if an Enter AID Key should be sent when the focus of the field gets lost.

VB.NET Syntax

[*Boolean* =] *Field.***AutoEnter**

C# Syntax

[*bool* =] *Field.***AutoEnter**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program.

4.3.3.2.3 Blinking

Returns true if the field should be shown blinking.

VB.NET Syntax

[*Boolean* =] *Field.***Blinking**

C# Syntax

[*bool* =] *Field.***Blinking**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program.

See Also **High**, **Reverse** and **Underline** properties.

4.3.3.2.4 ByPass

Returns true for unprotected fields that shouldn't gain the keyboard focus.

VB.NET Syntax

[*Boolean* =] *Field.***ByPass**

C# Syntax

[*bool* =] *Field.***ByPass**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program. This property is valid only for unprotected fields.

4.3.3.2.5 Col

Sets/gets the screen column coordinate where the field begins.

VB.NET Syntax

*Field.***Col** [= *Short*]

C# Syntax

*Field.***Col** [= *short*];

See Also **Row**, **Pos** and **Len** properties.

4.3.3.2.6 Color

Returns an integer that represents the default color corresponding to the standard or extended field attribute.

VB.NET Syntax

[*System.Drawing.Color* =] *Field.***Color**

C# Syntax

[*System.Drawing.Color* =] *Field.***Color**;

Remarks

This property returns an integer value and it's a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program.

See Also **Blinking**, **High**, **Reverse** and **Underline** properties.

4.3.3.2.7 Data

Sets/gets and gets the data of the field.

VB.NET Syntax

*Field.***Data** [= *String*]

C# Syntax

*Field.***Data** [= *string*];

Remarks

Changing the data of an unprotected field sets the **Modified** property to true.

See Also **Modified** property.

4.3.3.2.8 Eab

Sets/gets the extended attribute of a Field.

VB.NET Syntax

[*Char* =] *Field.***Eab**

C# Syntax

[*char* =] *Field.***Eab**;

Remarks

This property is for reserved use.

4.3.3.2.9 Editable

Returns true if the field is editable (unprotected), otherwise returns false.

VB.NET Syntax

[*Boolean* =] *Field.***Editable**

C# Syntax

[*bool* =] *Field.***Editable**;

See Also **Mandatory** property.

4.3.3.2.10 High

Returns true if the field should be shown with a high intensity color or bold, otherwise returns false.

VB.NET Syntax

[*Boolean* =] *Field.***High**

C# Syntax

[*bool* =] *Field.***High**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program.

See Also **Blinking**, **Color**, **Reverse** and **Underline** properties.

4.3.3.2.11 Len

Sets/gets the field's length.

VB.NET Syntax

*Field.***Len** [= *Short*]

C# Syntax

*Field.***Len** [= *short*];

See Also **Row**, **Col** and **Pos** properties.

4.3.3.2.12 Mandatory

Returns true if this field is editable and must be filled in.

VB.NET Syntax

[*Boolean* =] *Field.***Mandatory**

C# Syntax

[*bool* =] *Field.***Mandatory**;

Remarks

This property is valid only for unprotected fields.

See Also **Editable** property.

4.3.3.2.13 Modified

This property indicates if the field has been modified.

VB.NET Syntax

*Field.***Modified** [= *Boolean*]

C# Syntax

*Field.***Modified** [= *bool*];

See Also **Data** property.

4.3.3.2.14 MonoCase

This property indicates if the field accepts uppercase characters only.

VB.NET Syntax

[*Boolean* =] *Field.***MonoCase**

C# Syntax

[*bool* =] *Field.***MonoCase**;

Remarks

This property is valid only for unprotected fields.

See Also **Numeric** property.

4.3.3.2.15 Name

Returns a string with the field name. This name is made by the telnet controls and it can be modified before first use.

VB.NET Syntax

*Field.***Name** [= *String*]

C# Syntax

*Field.***Name** [= *string*];

Remarks

The string returned is the result of a concatenation between "R", the value of Row property, "C" and the value of Col property. For example, the name of a field located at row 4 column 30 will be *R4C30*.

4.3.3.2.16 Numeric

Indicates if the field accepts numeric digits only.

VB.NET Syntax

[*Boolean* =] *Field.***Numeric**

C# Syntax

[*bool* =] *Field.***Numeric**;

Remarks

This property is valid only for unprotected fields.

See Also **Monocase** property.

4.3.3.2.17 Pos

Returns the field position in the screen.

VB.NET Syntax

*Field.***Pos** [= *Short*]

C# Syntax

*Field.***Pos** [= *short*];

Remarks

This property returns an integer ranging from 1 to Fiel*d.***Cols** property multiplied by *Field.***Rows** property. The order of the numeration is from the left to the right and from top to bottom.

See Also **Col**, **Row** and **Len** properties.

4.3.3.2.18 Reverse

Returns true if the field should be shown in reverse video and false if the field should be shown in normal video.

VB.NET Syntax

[*Boolean* =] *Field.***Reverse**

C# Syntax

[*bool* =] *Field.***Reverse**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program.

See Also **Blinking**, **Color**, **High** and **Underline** property.

4.3.3.2.19 Row

Sets/gets the screen row coordinate where the field begins.

VB.NET Syntax

*Field.***Row** [= *Short*]

C# Syntax

*Field.***Row** [= *short*];

See Also **Col**, **Len** and **Pos** properties.

4.3.3.2.20 Skip

Returns true for fields that shouldn't gain the keyboard focus.

VB.NET Syntax

[*Boolean* =] *Field.***Skip**

C# Syntax

[*bool* =] *Field.***Skip**;

Remarks

When the property returns True, the cursor will skip this field preventing it to be written.
Normally, you should consider this property if you are going to implement a terminal emulation program.

4.3.3.2.21 Underline

Returns true if the field should be shown underlined and false if the field should not be shown underlined.

VB.NET Syntax

[*Boolean* =] *Field.***Underline**

C# Syntax

[*bool* =] *Field.***Underline**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program.

See Also **Blinking**, **Color**, **High** and **Reverse** properties.

4.3.3.2.22 Visible

Returns true if the field is visible, otherwise it returns false.

VB.NET Syntax

[*Boolean* =] *Field.***Visible**

C# Syntax

[*bool* =] *Field.***Visible**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program. If the field is a label (protected) and visible is false, it shouldn't be shown. If the field is editable and visible is false, it means that this is a password field.

4.3.3.2.23 HllApiAttr

Returns the attribute of Field in HLLAPI-compliance format.

VB.NET Syntax

[*Byte* =] *Field.***HllApiAttr**

C# Syntax

[*byte* =] *Field.***HllApiAttr**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program.

See Also **Editable**, **Blinking**, **Color**, **High** and **Reverse** properties.

4.3.3.2.24 HllApiEab

Returns the extended attribute of Field in HLLAPI-compliance format.

VB.NET Syntax

[*Byte* =] *Field.***HllApiEab**

C# Syntax

[*byte* =] *Field.***HllApiEab**;

Remarks

This is a terminal-emulation behavior attribute. Normally, you should consider this property if you are going to implement a terminal emulation program.

See Also **Blinking**, **Color**, **High** and **Reverse** properties.

4.3.4 Fields Class Reference

4.3.4.1 Fields Class

Implements the Screen Fields list.

Properties

- **Count**
- **CursorField**
- **CursorPos**
- **ErrorField**
- **Items**

4.3.4.2 Properties

4.3.4.2.1 Count

Returns the total number of **Fields** in the collection.

VB.NET Syntax

[*Integer* =] *Fields*.**Count**

C# Syntax

[*int* =] *Fields*.**Count**;

See also **Items** property.

4.3.4.2.2 CursorField

Returns the field where the cursor is located.

VB.NET Syntax

[*Field* =] *Fields*.**CursorField**

C# Syntax

[*Field* =] *Fields*.**CursorField**;

See also **CursorPos** property.

4.3.4.2.3 CursorPos

Gets/sets the host-screen cursor position.

VB.NET Syntax

Fields.**CursorPos** [= *Short*]

C# Syntax

Fields.**CursorPos** [= *short*];

Remarks

Valid cursor position goes from 1 to the total number of rows multiplied by the total number of columns available for the terminal type specified.

4.3.4.2.4 ErrorField

Returns the error field which contains the error messages sent by the host (Write Error Message Order or field displayed out of terminal size).

VB.NET Syntax

[*Field* =] *Fields*.**ErrorField**

C# Syntax

[*Field* =] *Fields*.**ErrorField**;

4.3.4.2.5 Items

Contains a collection of **Fields** objects.

VB.NET Syntax

[*Fields* =] *Fields*.**Items**

C# Syntax

[*Fields* =] *Fields*.**Items**;

Remarks

Use Items to get an specific field from the array. The Index parameter indicates the object's index in the collection, where 0 is the index of the first element and (**count** - 1) is the index of the last element.
Also, you can access an specific **Field** by its field name.
Use Items with the **Count** property to iterate through all the objects in the list.

See also **Count** and **Name** properties from **Field** class.

4.3.5 Pool Class Reference

4.3.5.1 Pool Class

This object is used for administering pools of telnet objects.

Properties

- **AcquiredCount**
- **Count**
- **Id**
- **ReleasedCount**
- **Size**
- **Timeout**

Methods

- **ReleaseAll**

4.3.5.2 Properties

4.3.5.2.1 AcquiredCount

Returns the ammount of allocated obejcts that are in use.

VB.NET Syntax

[*Integer* =] *Pool.AcquiredCount*

C# Syntax

[*int* =] *Pool*.**AcquiredCount**;

See also **Count** and **ReleasedCount** properties.

4.3.5.2.2 Count

Returns the ammount of obejcts allocated in the pool.

VB.NET Syntax

[*Integer* =] *Pool.Count*

C# Syntax

[*int* =] *Pool*.**Count**;

See also **AcquiredCount** and **ReleasedCount** properties.

4.3.5.2.3 Id

Specifies the pool group identification.

VB.NET Syntax

Pool.Id [= *String*]

C# Syntax

Pool.**Id** [= *string*];

Remarks

This property must be alphanumeric.

4.3.5.2.4 ReleasedCount

Returns the ammount of allocated obejcts that are available.

VB.NET Syntax

[*Integer* =] *Pool.Count*

C# Syntax

[*int* =] *Pool*.**ReleasedCount**;

See also **Count** and **AcquiredCount** properties.

4.3.5.2.5 Size

Specifies the maximun ammount of objects for the pool.

VB.NET Syntax

Pool.Size [= *Integer*]

C# Syntax

Pool.**Size** [= *int*];

Remarks

When the pool size is exhausted, no more objects are allocated and calls to Acquire methods return null objects.

4.3.5.2.6 Timeout

Specifies the inactivity timeout for the named pool.

VB.NET Syntax

Pool.Timeout [= *Integer*]

C# Syntax

Pool.**Timeout** [= *int*];

Remarks

Specifies the inactivity timeout for the pool. Once elapsed the specified time the timedout telnet object is disconnected and destroyed.

4.3.5.3 Methods

4.3.5.3.1 ReleaseAll

Returns all the acquired objects to the pool.

VB.NET Syntax

*Pool.***ReleaseAll**

C# Syntax

Pool.**ReleaseAll()**;

4.3.6 IndFile Reference

4.3.6.1 IndFile Control

Properties

- **AbortString**
- **Append**
- **Ascii**
- **BlkSize**
- **BufSize**
- **Bytes**
- **CrLf**

- Direction
- FtCommand
- FtMode
- FtName
- HostFileName
- HostKind
- LocalFileName
- LRecl
- PrimSpace
- Profiles
- RecFmt
- SecSpace
- ShowDialog
- ShowEditor
- SubKey
- Telnet
- Units

Methods

- AboutBox
- Abort
- Receive
- Run
- Send

Events

- OnAborting
- OnComplete
- OnRunning
- OnUpdateLength

4.3.6.2 Properties

4.3.6.2.1 AbortString

This property contains a strings that describes why the file transfer was aborted.

Visual Basic Syntax

[String =] *IndFile*.**AbortString**

C# Syntax

[string =] *IndFile*.**AbortString**;

See also Abort method.

4.3.6.2.2 Append

If set to True, the file to be transferred is appended to the existing one. If the destination file doesn't exist (HostFileName in case of Send method or LocalFileName in case of Receive method), it is treated as new.
If set to False, no append is taken place. If the destination file exists, it is replaced.

Visual Basic Syntax

IndFile.**Append** [= Boolean]

C# Syntax

IndFile.**Append**; [= bool]

4.3.6.2.3 Ascii

It is used when transferring files from host to PC to convert EBCDIC characters to Ascii characters.

Visual Basic Syntax

IndFile.**AscII** [= Boolean]

C# Syntax

IndFile.**AscII**; [= bool]

See also CrLf property.

4.3.6.2.4 BlkSize

Is the block size to be used for the host destination file (**HostFileName** property) when a new file is created in **Send** method or **Run** method with **Direction**=dSend indicator set. Must be a multiple of record length (**Lrecl** property), if fixed blocked (**RecFm** property) file format is specified.

Visual Basic Syntax

IndFile.**BlkSize** [= Integer]

C# Syntax

IndFile.**BlkSize**; [= int]

See also **HostFileName** property, **Lrecl** property, **RecFm** property, **Run** method, **Send** method.

4.3.6.2.5 BufSize

This property has the size of the buffer used to do the file transfer. It's only used when the connection mode is DFT.

Visual Basic Syntax

IndFile.**BufSize** [= Integer]

C# Syntax

IndFile.**BufSize**; [= int]

See also Stream property, **Bytes** property.

4.3.6.2.6 Bytes

Is the current bytes transferred to the destination file, **HostFileName** in **Send** or **Run** (**Direction**=0) methods, or **LocalFileName** in **Receive** or **Run** (**Direction**=1) methods.

Visual Basic Syntax

IndFile.**Bytes** [Integer =]

C# Syntax

IndFile.**Bytes**; [int =]

See also **OnUpdateLength** event.

4.3.6.2.7 CrLf

It is used when transferring file from a host to a PC.
If set to True, CRLF characters, chr(13)+chr(10), are added to the end of each line. If set to False no characters are added.

Visual Basic Syntax

IndFile.**CrLf** [= Boolean]

C# Syntax

IndFile.**CrLf**; [= bool]

See also **Ascii** property.

4.3.6.2.8 Direction

This property is used in conjunction with the **Run** method to indicate the transfer direction of the File Transfer.

Visual Basic Syntax

IndFile.**Direction** [= TxDirection]

C# Syntax

IndFile.**Direction**; [= TxDirection]

Remarks

Possible values are:

Constant Value	Meaning
dSend	send direction (from **LocalFileName** file name to

	HostFileName file name).
dReceive	receive direction (from **HostFileName** file name to **LocalFileName** file name).

See also TxDirection constants, Run method.

4.3.6.2.9 FtCommand

Sets/gets the name of the file transfer command in the host system, usually "IND$FILE". This property doesn't have to be changed from its default value.

Visual Basic Syntax

IndFile.**FtCommand** [= "IND$FILE"]

C# Syntax

IndFile.**FtCommand**; [= "IND$FILE"]

4.3.6.2.10 FtMode

Indicates the File Transfer Mode. By default the value is ftDFT, but if the host can't handle DFT mode it automatically changes to ftCut.

Visual Basic Syntax

[TxFtMode =] *IndFile*.**FtMode**

C# Syntax

[TxFtMode =] *IndFile*.**FtMode***;*

Remarks

Possible values are:

Constant Value	Meaning
ftDFT	DFT mode
ftCut	Cut mode

See also TxFtMode constants.

4.3.6.2.11 FtName

It's the name of the file transfer session.

Visual Basic Syntax

IndFile.**FtName** [= String]

C# Syntax

IndFile.**FtName**; [= string]

See also **Profiles** property, **SubKey** property.

4.3.6.2.12 HostFileName

Sets/gets the name in the host system of the file to be transferred. It is used either in the **Send** / **Receive** methods mode or the **Run** method mode. The file name syntax has to be valid for the host system.

Example:

"File Name a1"	(file name, file type, file mode) valid name for VM/CMS.
"sys1.vtamlst(atcstr00)"	valid name for PDS member name under Tso.
"sys2.user.dat"	valid name for catalogued file.

Visual Basic Syntax

IndFile.**HostFileName** [= "FileName.dat"]

C# Syntax

IndFile.**HostFileName**; [= "FileName.dat"]

See also **LocalFileName** property.

4.3.6.2.13 HostKind

Sets/gets the environment that the host supports.

Visual Basic Syntax

IndFile.**HostKind** = *TxHostKind*

C# Syntax

IndFile.**HostKind** = *TxHostKind;*

Remarks

Possible values are:

Constant Value	Meaning
hVM	VM
hTSO	TSO
hCics	Cics

See also TxHostKind constants.

4.3.6.2.14 LocalFileName

Sets/gets the name in the local system of the file to be transferred. It is used either in the **Send** / **Receive** methods mode or the **Run** method mode. The file name syntax has to be valid in the local system.

Example:

Visual Basic Syntax

IndFile.**LocalFileName** [= "FileName.dat"]

C# Syntax

IndFile.**LocalFileName**; [= "FileName.dat"]

See also HostFileName property.

4.3.6.2.15 LRecl

Sets/gets the record length to be used for the host destination file (**HostFileName** property) when a new file is created during the execution of a **Send** method or **Run** method (with **Direction**=0 indicator set).
It must be a submultiple of block size (**BlkSize** property), if fixed blocked (**RecFm** property) file format is specified.

Visual Basic Syntax

IndFile.**LRecl** [= Integer]

C# Syntax

IndFile.**LRecl**; [= int]

See also **HostFileName** property, **BlkSize** property, **RecFm** property, **Run** method, **Send** method.

4.3.6.2.16 PrimSpace

Sets/gets the primary space quantity used to allocate the host **HostFileName** file in the **Units** selected, when a new file is created during the execution of a **Send** method or **Run** method (with **Direction**=0 indicator set).

Visual Basic Syntax

IndFile.**PrimSpace** [= Integer]

C# Syntax

IndFile.**PrimSpace**; [= int]

See also **HostFileName** property, **SecSpace** property, **BlkSize** property, **RecFm** property, **Units** property, **Run** method, **Send** method.

4.3.6.2.17 Profiles

Sets/gets the Profiles Component as the profile manager for this Component.

Visual Basic Syntax

*IndFile.***Profiles** [= Profiles]

C# Syntax

*IndFile.***Profiles**; [= Profiles]

After setting this property, you can access to a collection of connections to be generated and customized through the **ShowEditor** method.

See also **SubKey** property, **ShowEditor** property.

4.3.6.2.18 RecFmt

Sets/gets the record format of the host **HostFileName** file.

Visual Basic Syntax

IndFile.**RecFmt** [= Integer]

C# Syntax

IndFile.**RecFmt**; [= int]

Possible values for this property are:

Value	Meaning
rDefault	Default file format
rFixed	Fixed file format
rVariable	Variable file format
rUndefined	Undefined file format

See also **TxRecordFmt** constant, **BlkSize** property, **HostFileName** property, **PrimSpace** property, **Run** method, **SecSpace** property, **Send** method, **Units** property.

4.3.6.2.19 SecSpace

Sets/gets the secondary space quantity used to allocate the host **HostFileName** file in the **Units** selected, when a new file is created during the execution of a **Send** method or **Run** method (with **Direction**=0 Send indicator set).

Visual Basic Syntax

IndFile.**SecSpace** [= Integer]

C# Syntax

IndFile.**SecSpace**; [= int]

See also **HostFileName** property, **PrimSpace** property, **BlkSize** property, **RecFm** property, **Units** property, **Run** method, **Send** method.

4.3.6.2.20 ShowDialog

Specifies if the file transfer "Bytes Transferred" dialog appears in each buffer sent to the destination. This dialog could be used to Cancel the file transfer in progress.

Visual Basic Syntax

IndFile.**ShowDialog** [= Boolean]

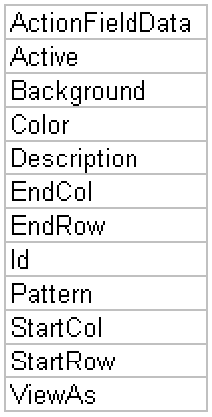
C# Syntax

IndFile.**ShowDialog**; [= bool]

See also **OnAborting** event.

4.3.6.2.21 ShowEditor

Specifies if the *File Transfer Settings* dialog appears previously to each file transfer request to let the user modify the file transmission parameters.
All the fields in both *Transfers* and *Advanced* tabs have the corresponding property that can be set programmatically instead than doing it through this dialog.



ActionFieldData
Active
Background
Color
Description
EndCol
EndRow
Id
Pattern
StartCol
StartRow
ViewAs

Visual Basic Syntax

IndFile.**ShowEditor** [= Boolean]

C# Syntax

IndFile.**ShowEditor**; [= bool]

4.3.6.2.22 SubKey

Sets/gets the subkey under which the connection profiles will be stored.

Visual Basic Syntax

IndFile.**SubKey** [= *String*]

C# Syntax

IndFile.**SubKey**; [= *string*]

Remarks

Default value is "TNB3270E" for TN3270 Component.

See also Profiles property, FtName property.

4.3.6.2.23 Telnet

Sets/gets the Tn3270 Component as the telnet client Component.

Visual Basic Syntax

*IndFile.***Telnet** [= *TN3270*]

C# Syntax

*IndFile.***Telnet**; [= *TN3270*]

4.3.6.2.24 TimeOut

Sets/gets the time slice to wait for the host connection to be established.

Visual Basic Syntax

IndFile.**TimeOut** [= Integer]

C# Syntax

IndFile.**TimeOut**; [= int]

See also **Receive** method, **Run** method, **Send** method.

4.3.6.2.25 Units

Sets/gets the type of allocation unit of the host **HostFileName** file used when a new file is created during the execution of a **Send** method or **Run** method (with **Direction**=dSend Send indicator set).

Value	**Meaning**
uDefault	Default unit of allocation
uTracks	Tracks unit of allocation
uCylinder	Cylinder unit of allocation
uAvBlocks	Average Blocks unit of allocation

Visual Basic Syntax

IndFile.**Units** [= TxUnits]

C# Syntax

IndFile.**Units**; [= TxUnits]

See also **TxUnits** constant, **BlkSize** property, **PrimarySpace** property, **RecFm** property, **SecondarySpace** property.

4.3.6.3 Methods

4.3.6.3.1 AboutBox

Shows Integration Pack about dialog box.

Visual Basic Syntax

*IndFile.***AboutBox**

Delphi Syntax

*IndFile.***AboutBox**;

4.3.6.3.2 Abort

The transfer is interrupted after changing the current transfer's status to abort.

Visual Basic Syntax

IndFile.**Abort()**

Delphi Syntax

IndFile.**Abort()**;

See also **AbortString** property, **OnAborting** event.

4.3.6.3.3 Receive

This method is used to receive into **LocalFileName** file the **HostFileName** file.

Visual Basic Syntax

IndFile.**Receive()**

Delphi Syntax

IndFile.**Receive()**;

See also **Direction** property, **Run** method, **Send** method.

4.3.6.3.4 Run

Is used to send **LocalFileName** file from the PC to the **HostFileName** in the host if **Direction** property is set to 0=Send, or vice versa if **Direction** property is set to 1=Receive.

Visual Basic Syntax

IndFile.**Run()**

Delphi Syntax

IndFile.**Run()**;

See also **Direction** property, **Receive** method, **Send** method.

4.3.6.3.5 Send

This method is used to send **LocalFileName** file from the PC to the **HostFileName** in the host using the properties directly modified or using the Editor dialog to fill them interactively.

Visual Basic Syntax

IndFile.**Send()**

Delphi Syntax

IndFile.**Send()**;

See also **Direction** property, **Run** method, **Receive** method.

4.3.6.4 Events

4.3.6.4.1 OnAborting

Occurs after the Cancel button in the **'Bytes Transferred'** dialog is pressed indicating a user defined cancel operation.

Declaration

procedure OnAborting (Sender: TObject)

Parameters

Sender: The *IndFile* object that raises the event.

See also **OnComplete** event.

4.3.6.4.2 OnComplete

Occurs after the file transfer operation is finished.

Declaration

procedure OnComplete (Sender: TObject)

Parameters

Sender: The *IndFile* object that raises the event.

See also **OnAborting** event.

4.3.6.4.3 OnRunning

Occurs when the file transfer operation has been started.

Declaration

procedure OnRunning (Sender: TObject)

Parameters

Sender: The *IndFile* object that raises the event.

See also **OnComplete** event.

4.3.6.4.4 OnUpdateLength

Occurs every time a buffer of data of **BufSize** bytes are sent from the source file to the destination file. Previously this event is fired the **Bytes** property is updated accordingly to reflect the total amount of data sent.

Declaration

procedure OnUpdateLength (Sender: TObject)

Parameters

Sender: The *IndFile* object that raises the event.

See also **OnRunning** event.

4.3.6.5 Constants

4.3.6.5.1 TxFtMode

These values are used in the **FTMode** property of **IndFile** control.

Constant Value	Meaning
ftDFT	DFT mode
ftCut	Cut mode

4.3.6.5.2 TxHostKind

These values are used in the **HostKind** property of **IndFile** control.

Constant Value	**Meaning**
hVM	VM
hTSO	TSO
hCics	Cics

4.3.6.5.3 TxRecordFmt

These values are used in the **RecFm** property of **IndFile** control.

Constant Value	**Meaning**
rDefault	Default file format
rFixed	Fixed file format
rVariable	Variable file format
rUndefined	Undefined file format

4.3.6.5.4 TxUnits

These values are used in the **Units** property of **IndFile** control.

Constant Value	**Meaning**
uDefault	Default unit of allocation
uTracks	Tracks unit of allocation
uCylinder	Cylinder unit of allocation
uAvBlocks	Average Blocks unit of allocation

4.3.6.5.5 TxDirection

These values are used in the **Direction** property of **IndFile** control.

Constant Value	**Meaning**

dSend	send direction (from **LocalFileName** file name to **HostFileName** file name).
dReceive	receive direction (from **HostFileName** file name to **LocalFileName** file name).

4.3.7 Constants

4.3.7.1 AidKey

These values are used in the **AidKey** property of telnet controls.

Constant Value	Meaning
NoOp	No Action
Enter	ENTER key
Clear	CLEAR key
ATTN	Attention key
SysReq	System Request key
PF1	PF01 key
PF2	PF02 key
PF3	PF03 key
PF4	PF04 key
PF5	PF05 key
PF6	PF06 key
PF7	PF07 key
PF8	PF08 key
PF9	PF09 key
PF10	PF10 key
PF11	PF11 key
PF12	PF12 key
PF13	PF13 key
PF14	PF14 key
PF15	PF15 key
PF16	PF16 key
PF17	PF17 key
PF18	PF18 key
PF19	PF19 key
PF20	PF20 key
PF21	PF21 key
PF22	PF22 key
PF23	PF23 key
PF24	PF24 key
PA1	PA1 key
PA2	PA2 key

PA3	PA3 key
Reset	Reset key
PgUp	Page Up key
PgDown	Page Down key
PgRight	Page Right key
PgLeft	Page Left key
Print	Print key
Help	Help key
RecordBackSpace	RecordBackSpace key

4.3.7.2 CodePage

These values are used in the **CodePage** property of telnet controls.

Constant Value	Meaning
Default = "0"	United States
Australia = "037"	Australian
Austria = "273"	Austria
Austria2 = "1141"	Austria
Belgium = "500"	Belgium
Belgium2 = "1148"	Belgium
Canada = "037"	Canada
Canada2 = "1047"	Canada
Canada3 = "1140"	Canada
CzechRepublic = "2975"	Czech Republic
Denmark = "277"	Denmark
Denmark2 = "1142"	Denmark
Finland = "278"	Finland
Finland2 = "1143"	Finland
France = "297"	France
France2 = "1147"	France
Germany = "273"	Germany
Germany2 = "1141"	Germany
Greece = "423"	Greece
Greece2 = "875"	Greece
Iceland = "871"	Iceland
Iceland2 = "1149"	Iceland
Italy = "280"	Italy
Italy2 = "1144"	Italy
LatinAmerica = "284"	Latin America
LatinAmerica2 = "1145"	Latin America
Netherlands = "037"	Netherlands
Norway = "277"	Norway
Norway2 = "1142"	Norway

Poland = "2978"	Poland
Portugal = "037"	Portugal
Russian = "2979"	Russia
Spain = "284"	Spain
Spain2 = "1145"	Spain
Sweden = "278"	Sweden
Sweden2 = "1143"	Sweden
Switzerland = "500"	Switzerland
Switzerland2 = "1148"	Switzerland
UnitedKingdom = "285"	United Kingdom
UnitedKingdom = "1146"	United Kingdom
UnitedStates = "037"	United States
UnitedStates2 = "037/2"	United States
UnitedStates3 = "1047"	United States
UnitedStates4 = "1140"	United States
International = "850"	International

4.3.7.3 FieldAttribute

These values are used in the **Atrr** property of the **Field** Class.

Constant Value	Meaning
0	Protected
1	Unprotected
2	Alphanumeric
3	Numeric
4	AlphabeticOnly
5	NumericOnly
6	SignedNumeric
7	Digits
8	MagneticStripe
9	High
10	Normal
11	Invisible
12	Display
13	NonDisplay
14	Modified
15	NonModified
16	AlphanumericShift
17	NumericShift
18	DupEnable
19	ByPass
20	AutoEnter
21	FieldExitRequired

22	MonoCase
23	Mandatory
24	NoAdjust
25	RightZero
26	RightBlank
27	MandatoryFill
28	KataShift
29	Graphic

4.3.7.4 SessionState

These values are used in the **SessionState** property of telnet controls.

Constant Value	Meaning
0 = NoSession	Not in session.
1 = SscpLu	In session with VTAM.
2 = LuLu	In session with VTAM Application.

4.3.7.5 TerminalType for Tn5250

These values are used in the **TerminalType** property of Tn5250 control.

Constant Value	Meaning
0	IBM5211m2
1	IBM3179m2
2	IBM3477mFC
3	IBM3180m2
4	IBM3477mFG
5	IBM3196mA1
6	IBM5292m2
7	IBM5291m1
8	IBM5251m11

4.3.7.6 TerminalType for Tn3270

These values are used in the **TerminalType** property of Tn3270 control.

Constant Value	Meaning
0	IBM3278m2
1	IBM3278m2E
2	IBM3278m3
3	IBM3278m3E
4	IBM3278m4
5	IBM3278m4E
6	IBM3278m5
7	IBM3278m5E

4.4 Terminal Emulator Controls

Find in the topics within this section a complete reference for the TNBridge Host Integration Pack .NET Terminal Emulator Controls.

4.4.1 Display Control

4.4.1.1 Display Control

This is an implementation of the screen-emulation panel as an Control without the keyboard interface.

Properties

- **CursorPos**
- **EnableSelection**
- **ErrorLine**
- **FontAutoSize**
- **InhibitInputOnError**
- **InputInhibitEnabled**
- **InsertMode**
- **IsInputInhibited**
- **Keyboard**
- **LeftMargin**
- **Profiles**
- **Ruler**
- **StyleName**
- **Telnet**
- **Text**
- **TopMargin**

Methods

- **Copy**
- **GetText**
- **Paste**
- **Press**
- **PressAndWait**
- **PrintScreen**
- **PutFields**
- **Refresh**
- **SetText**
- **ShowEditor**
- **Type**

Events

- **OnCursorPos**
- **OnInputInhibitChange**
- **OnInsertModeChange**

Constants

- **RulerType**

4.4.1.2 Properties

4.4.1.2.1 CursorPos

Gets/sets the host-screen cursor position.

VB.NET Syntax

Display.**CursorPos** [= *Integer*]

C# Syntax

Display.**CursorPos** [= *int*];

Remarks

Valid cursor position goes from 1 to the total number of rows multiplied by the total number of columns available for the terminal type specified. The current cursor position is available in both **HostFields** and **EditFields** properties from **Tn5250/Tn3270/XmlClient/XmlVirtual** class. To set a new cursor position you must do it on **EditFields** property.

See also **HostFields** and **EditFields** property from **Tn5250/Tn3270/XmlClient/XmlVirtual** class.

4.4.1.2.2 EnableSelection

Enables/disables the selection of an emulation-screen area for copying it to the clipboard.

VB.NET Syntax

Display.**EnableSelection** [= *Boolean*]

C# Syntax

Display.**EnableSelection** [= *bool*];

Remarks

Set the property to true if you plan to use **Copy** method, to allow the selection of a copy area**.**

Default value is **True**.

See Also **Copy** and **Paste** methods.

4.4.1.2.3 ErrorLine

Set/gets the line number where the error message will be displayed.

VB.NET Syntax

Display.**ErrorLine** [= *Integer*]

C# Syntax

Display.**ErrorLine** [= *int*];

Remarks

Default value is **0**.

See Also **InhibitInputOnError** property.

4.4.1.2.4 FontAutoSize

Set/gets the autosize state of the emulator-screen font.

VB.NET Syntax

Display.**FontAutoSize** [= *Boolean*]

C# Syntax

Display.**FontAutoSize** [= *bool*];

Remarks

Default value is **True**.

4.4.1.2.5 InhibitInputOnError

Set/gets a boolean value to lock/unlock the keyboard when the host returns an error message.

VB.NET Syntax

Display.**InhibitInputOnError** [= *Boolean*]

C# Syntax

Display.**InhibitInputOnError** [= *bool*];

Remarks

Default value is **False**.

See Also **ErrorLine** property.

4.4.1.2.6 InputInhibitEnabled

Enables/disables the input inhibit condition when any key is pressed on a protected field.

VB.NET Syntax

Display.**InputInhibitEnabled** [= *Boolean*]

C# Syntax

Display.**InputInhibitEnabled** [= *bool*];

Remarks

Default value is **True**.

See Also **IsInputInhibited** property and **OnInputInhibitChange** event.

4.4.1.2.7 InsertMode

Sets/gets the insert/overwrite keyboard mode for typing on unprotected fields.

VB.NET Syntax

Display.**InsertMode** [= *Boolean*]

C# Syntax

Display.**InsertMode** [= *bool*];

Remarks

Default value is **False**.

4.4.1.2.8 IsInputInhibited

Returns true if the input of the emulator-screen is inhibited.

VB.NET Syntax

[*Boolean* =] *Display*.**IsInputInhibited**

C# Syntax

[*bool* =] *Display*.**IsInputInhibited**;

See Also **InputInhibitEnabled** property and **OnInputInhibitChange** event.

4.4.1.2.9 Keyboard

Sets/gets the **Keyboard** Control as the keyboard handler for this control.

VB.NET Syntax

*Display.***Keyboard** [= *Keyboard*]

C# Syntax

*Display.***Keyboard** [= *Keyboard*];

Remarks

Normally, to send information to the host through the Display you must call the **Type** method and then send the properly AID key to the telnet control (see **AidKey** property or **Press** method of Tn3270/Tn5250 Controls). Assigning this property to a **Keyboard** Control, you don't need to do these calls. When the Display receives a function key from Keyboard, process it internally, sending the modified fields and the AID key pressed to the host.

See Also **Keyboard** control.

4.4.1.2.10 LeftMargin

Sets/gets emulation screen left margin.

VB.NET Syntax

*Display.***LeftMargin** [= *Integer*]

C# Syntax

*Display.***LeftMargin** [= *int*];

Remarks

Left margin emulation value is measured in pixels.

4.4.1.2.11 Profiles

Sets the **Profiles** Control as the profile manager for this control.

VB.NET Syntax

Display.**Profiles** [= *Profiles*]

C# Syntax

Display.**Profiles** [= *Profiles*];

Remarks

After setting this property, you can access a collection of styles to be generated and customized through the **ShowEditor** method.

See Also **Profiles** control, **StyleName** property and **ShowEditor** method.

4.4.1.2.12 Ruler

Sets/gets the format style of the ruler. This ruler shows the cursor position.

VB.NET Syntax

Display.**Ruler** [= **RulerType**]

C# Syntax

Display.**Ruler** [= **RulerType**];

Remarks

For example, *ruCrosshair* constant value will draw one vertical and one horizontal lines, and the intersection will be the cursor position.

See Also **CursorPos** property and **RulerType** constants.

4.4.1.2.13 StyleName

Sets/gets the Style from the collection of styles saved through the **Profiles** Control.

VB.NET Syntax

Display.**StyleName** [= *String*]

C# Syntax

Display.**StyleName** [= *string*];

Remarks

This property is valid if a **Profiles** property has been previously assigned.

See Also **Profiles** control and **ShowEditor** method.

4.4.1.2.14 Telnet

Sets/gets the Tn3270 or Tn5250 Control as the telnet client control.

VB.NET Syntax

Display.**Telnet** [= *IComApi*]

C# Syntax

Display.**Telnet** [= *IComApi*];

Remarks

You must set this property for Display control to allow it to work properly.

4.4.1.2.15 Text

Returns the entire screen.

VB.NET Syntax

[*String* =] *Display*.**Text**

C# Syntax

[*string* =] *Display*.**Text**;

See also **GetText** method.

4.4.1.2.16 TopMargin

Sets/gets emulation screen top margin.

VB.NET Syntax

Display.**TopMargin** [= *Integer*]

C# Syntax

Display.**TopMargin** [= *int*];

Remarks

Top margin emulation value is measured in pixels.

4.4.1.3 Methods

4.4.1.3.1 Copy

The Copy method copies the selected area to the clipboard.

VB.NET Syntax

Display.Copy

C# Syntax

Display.Copy;

Remarks

The **EnableSelection** property must be set to true to allows you select an area of the emulator-screen.

See also **EnableSelection** property and **Paste** method.

4.4.1.3.2 GetText

GetText returns the text buffer of a portion of the current screen. You can receive the attribute or extended attribute of each field together to the character data.

VB.NET Syntax

[*String* =] *Display.*GetText (*StartPos As Integer, [EndPos As Integer], [Attr As Boolean = False], [Eab As Boolean = False]*)

[*String* =] *Display.*GetText (*Row As Integer, Col As Integer, Len as Integer,[Attr As Boolean = False], [Eab As Boolean = False]*)

C# Syntax

[*string* =] *Display.*GetText (*int StartPos, [int StartPos], [bool Attr = False], [bool Eab = False]*);

[*string* =] *Display.*GetText (*int Row, in Col, int Len, [bool Attr = False], [bool Eab = False]*);

Remarks

Start position and the end position of the text buffer to retrieve can be set using StartPos and EndPos parameters or Row,Col and Len parameters.

The Attr parameter will indicate to GetText that includes the field attribute information. The Attr describes the field properties and occupies the first character position of each field in the character buffer and on the screen. This character is in HLLAPI-Compliance format.

The Eab parameter will indicate to GetText that includes the extended attribute information. The Eab is associated with individual characters. Each character in the buffer (except for fields attributes) has a character attribute. In the returned String, each character will be preceded by his Eab value. This character is in HLLAPI-Compliance format.

4.4.1.3.3 Paste

The Paste method pastes text from the clipboard to the current cursor position.

VB.NET Syntax

*Display.*Paste

C# Syntax

*Display.*Paste;

See also **Copy** method.

4.4.1.3.4 Press

Sets an Attention Identifier Key and sends the modified fields to the Mainframe. Modified fields can be input fields (unprotected) or sometimes protected fields, having the property Modified set to True.

VB.NET Syntax

[*Boolean* =] *Display.*Press (*Value As string*)

C# Syntax

[*bool* =] *Display.*Press (*string Value*);

This method works like the AidKey property, but it takes a string with the name of the AID key as a parameter. You can specify only one key for each call to the method.

The following table lists the possible string values:

String Value	Meaning
Enter	ENTER key
Clear	CLEAR key
ATTN	Attention key
SysReq	System Request key
PF1	PF01 key
PF2	PF02 key
PF3	PF03 key
PF4	PF04 key
PF5	PF05 key
PF6	PF06 key
PF7	PF07 key

PF8	PF08 key
PF9	PF09 key
PF10	PF10 key
PF11	PF11 key
PF12	PF12 key
PF13	PF13 key
PF14	PF14 key
PF15	PF15 key
PF16	PF16 key
PF17	PF17 key
PF18	PF18 key
PF19	PF19 key
PF20	PF20 key
PF21	PF21 key
PF22	PF22 key
PF23	PF23 key
PF24	PF24 key
PA1	PA1 key

See also **AIDKey** property.

4.4.1.3.5 PressAndWait

Sends an Attention Identifier Key along with all modified fields to the Mainframe and waits until the system become unlocked.

VB.NET Syntax

[*Boolean* =] *Display.*PressAndWait(*Value As string,[TimeOut As Integer]*)

C# Syntax

[*bool* =] *Display.*PressAndWait (*string Value,[int TimeOut]*);

This method works as a combination of Press and WaitForUnlock methods. Timeout is an optional parameter and specify how much time in milliseconds the call will stop waiting for an unlock signal. If not is specified, the timeout will take the value of WaitTimeout property.

Returns true if the

See also Press and WaitForUnlock methods and WaitTimeout property.

4.4.1.3.6 PrintScreen

Prints all the emulator-screen area.

VB.NET Syntax

*Display.*PrintScreen

C# Syntax

*Display.*PrintScreen;

4.4.1.3.7 PutFields

Sends the modified fields to the Tn3270/Tn5250/XmlClient/XmlVirtual component in use by the Display control.

VB.NET Syntax

Display.**PutFields**

C# Syntax

Display.**PutFields()**;

Remarks

To complete the data transmission to the host, you must set the AidKey property or use the Press or PressAndWait methods from the Tn3270/Tn5250/XmlClient/XmlVirtual component in use by the Display control.

4.4.1.3.8 Refresh

Refresh the fields for the current screen.

VB.NET Syntax

*Display.*Refresh

C# Syntax

*Display.*Refresh();

4.4.1.3.9 SetText

SetText allows to put text information into edit buffer.

VB.NET Syntax

*Display.*SetText (*StartPos As Integer, String text*)

*Display.*SetText (*Row As Integer, Col As Integer, String text*)

C# Syntax

*Display.*SetText (int *StartPos, string text*)

*Display.*SetText (int *Row, int Col, string text*)

Remarks

SetText allows to put text information into edit buffer, starting according to position StartPos or Row and Col parameters.

4.4.1.3.10 ShowEditor

Shows a modal dialog with the screen properties.

VB.NET Syntax

*Display.*ShowEditor

C# Syntax

*Display.*ShowEditor;

Remarks

If you haven't assigned the **Profiles** property, the style list will be shown disabled. Setting the **Profiles** property to a valid **Profiles** control, allows you to define screenstyles at run-time and save them into a file for future reuse.

See also **Profiles** property.

4.4.1.3.11 Type

Type can be used to send key sequences to the mainframe (including AID keys). By using special codes you can send several kinds of keys. These codes consist of an Escape character (default is "@") and a mnemonic code that corresponds to the supported host functions.

VB.NET Syntax

*Display.*Type (*DataString As String*)

C# Syntax

*Display.*Type (*string DataString*);

The following table lists the functions keys and its corresponding codes.

Code Value	Meaning
@A@Q	Attention
@<	Backspace
@B	BackTab (Left Tab)
@C	Clear
@E	Enter
@F	Erase Field
@A@Q	Sys Request
@T	Tab (Right Tab)
@x	PA1
@y	PA2
@z	PA3
@1	PF1
@2	PF2
@3	PF3
@4	PF4
@5	PF5
@6	PF6
@7	PF7
@8	PF8
@9	PF9
@a	PF10
@b	PF11
@c	PF12

@d	PF13
@e	PF14
@f	PF15
@g	PF16
@h	PF17
@i	PF18

Here is an example of the *Keys* parameter:

```
"logon applid(cics)@E"
```

Remarks

The execution of the SendKeys is stopped when an AID Key is sent to the Host.

See also **AIDKey** property.

4.4.1.4 Events

4.4.1.4.1 OnCursorPos

Occurs when the emulator cursor changes its position.

4.4.1.4.2 OnInputInhibitChange

Occurs when the input from keyboard changes to inhibit or vice versa.

See also **OnInsertModeChange** event.

4.4.1.4.3 OnInsertModeChange

Occurs when the input from keyboard changes to insert mode or overwrite mode.

See also **OnInputInhibitChange** event.

4.4.1.5 Constants

4.4.1.5.1 RulerType

These values are used in the property Ruler of Display control.

Constant Value	Meaning
None = 0	No lines
Crosshair = 1	Horizontal and vertical lines
Vertical = 2	Only vertical line
Horizontal = 3	Only horizontal line

4.4.2 Keyboard Control

4.4.2.1 Keyboard Control

This is an implementation of a keyboard interface, with mapping to the standard function keys of the 5250 and 3270 emulation.

Properties

- Active
- KeyboardFile
- KeyboardType
- Profile
- Profiles

Methods

- ShowEditor

Constants

- Type

4.4.2.2 Properties

4.4.2.2.1 Active

Enables/disables the keyboard hook.

VB.NET Syntax

*Keyboard.***Active** [= *Boolean*]

C# Syntax

*Keyboard.***Active** [= *bool*];

Remarks

Default value is **False**.

4.4.2.2.2 KeyboardFile

Allows you to specify an external file with a new keyboard mapping.

VB.NET Syntax

*Keyboard.***KeyboardFile** [= *String*]

C# Syntax

*Keyboard.***KeyboardFile** [= *string*];

Remarks

The file to be used can be exported from its property page or dialog box shown using the **ShowEditor** method**.**

See Also **ShowEditor** method.

4.4.2.2.3 KeyboardType

Specify the keyboard mapping type.

VB.NET Syntax

*Keyboard.***KeyboardType** [= **Type**]

C# Syntax

*Keyboard.***KeyboardType** [= **Type**];

Remarks

This control maintain separates keyboard mapping tables for each connection type. You can modify it through the property page associated to the control.

See Also **Type** constants.

4.4.2.2.4 Profile

Sets or gets the name of the current keyboard map set.

VB.NET Syntax

*Keyboard.***Profile** [= *String*]

C# Syntax

*Keyboard.***Profile** [= *string*];

Remarks

The default keyboard map is called "Default". You can access a collection of keyboard maps to be generated and customized through the **ShowEditor** method. Then you can assign a different profile using this property.
Different profiles generated will be stored through **Profiles** control and the physical file will be identified through **KeyboardFile** property.

See Also **Profiles** control, **KeyboardFile** property and **ShowEditor** method.

4.4.2.2.5 Profiles

Sets the **Profiles** Control as the profile manager for this control.

VB.NET Syntax

*Keyboard.***Profiles** [= *Profiles*]

C# Syntax

*Keyboard.***Profiles** [= *Profiles*];

Remarks

After setting this property, you can access a collection of keyboard maps to be generated and customized through the **ShowEditor** method.

See Also **Profiles** control and **ShowEditor** method.

4.4.2.3 Methods

4.4.2.3.1 ShowEditor

Shows a modal dialog with the screen properties.

VB.NET Syntax

*Keyboard.*ShowEditor

C# Syntax

*Keyboard.*ShowEditor;

4.4.2.4 Constants

4.4.2.4.1 Type

These values are used in the **KeyboardKind** property of telnet controls.

Constant Value	Meaning
KBD5250	5250 Keyboard
KBD3270	3270 Keyboard

4.4.3 StatusBar Control

4.4.3.1 StatusBar Control

This control implements a status bar that shows the connection and session status.

Properties

- **Display**
- **Telnet**

4.4.3.2 Properties

4.4.3.2.1 Display

Sets/gets the Display Control for emulator events handling.

VB.NET Syntax

*StatusBar.*__Display__ [= ***Display***]

C# Syntax

*StatusBar.*__Display__ [= ***Display***];

Remarks

You must set this property to allow **StatusBar** control to work properly.

4.4.3.2.2 Telnet

Sets/gets the Tn3270/Tn5250/XmlClient/XmlVirtual Control as the telnet client control.

VB.NET Syntax

*StatusBar.*__Telnet__ [= *IComApi*]

C# Syntax

*StatusBar.*__Telnet__ [= *IComApi*];

Remarks

You must set this property to allow **StatusBar** control to work properly.

4.4.4 Profiles Control

4.4.4.1 Profiles Control

This component implements a way to store profiles in an XML File.

Properties

- **FileName**
- **ItemsCount**
- **ReadOnly**

Methods

- **Items**
- **Delete**
- **Enumerate**
- **GetCurrent**
- **Rename**
- **SetCurrent**

Constants

- Key

4.4.4.2 Properties

4.4.4.2.1 FileName

Sets or gets the file name for a profile. This file will be used to save a profile or to load an existing one.

VB.NET Syntax

*Profiles.***FileName** [= *String*]

C# Syntax

*Profiles.***FileName** [= *string*];

Remarks

Internally a profile file is divided in sections (keys). You can use only one file for all kind of profiles you want to save, for example: connections, keyboard maps, macros, screen styles and hotspots, or one file for each kind of profile.

Sections can be accessed through **Key** property.

See Also **Key** property.

4.4.4.2.2 ItemsCount

Returns the count of Profiles items.

VB.NET Syntax

*Profiles.**ItemsCount*** [= *Integer*]

C# Syntax

*Profiles.**ItemsCount*** [= *int*];

4.4.4.2.3 ReadOnly

Indicates if the profile file has read only attribute set.

VB.NET Syntax

[*Boolean* =] *Profiles.***ReadOnly**

C# Syntax

[*bool* =] *Profiles.***ReadOnly**;

Remarks

If the attribute means that the profile file is read-only (ReadOnly property returning True value), profile information will not be saved preventing and error message to occur.

See Also **FileName** property.

4.4.4.3 Methods

4.4.4.3.1 Items

Returns the profile name at *index* position.

VB.NET Syntax

*Profiles.**Items*** **(***Index As Integer***)**

C# Syntax

*Profiles.**Items*** **(**int index**)**;

4.4.4.3.2 Delete

Deletes the profile specified.

VB.NET Syntax

*Profiles.***Delete (***SubKey As String, Profile As String***)**

C# Syntax

*Profiles.***Delete (***string SubKey, string Profile***)**;

Remarks

SubKey parameter is the subkey of the caller control and Profile parameter is the name of the configuration that is going to be deleted.

See also Key and **FileName** properties, and **Rename** method.

4.4.4.3.3 Enumerate

Enumerates the profiles names for that *key*.

VB.NET Syntax

*Profiles.****Enumerate*** **(***Value As Key***)**

C# Syntax

*Profiles.****Enumerate*** **(**Key value**)**;

4.4.4.3.4 GetCurrent

Returns a string with the current profile.

VB.NET Syntax

[*String* =] *Profiles.***GetCurrent (***SubKey As String***)**

C# Syntax

[*string* =] *Profiles.***GetCurrent (***string SubKey***)**;

Remarks

This method is generally used to get the default profile name to be loaded.

4.4.4.3.5 Rename

Renames the profile specified.

VB.NET Syntax

*Profiles.***Rename (***SubKey As String, OldProfile As String, NewProfile As String***)***;*

C# Syntax

*Profiles.***Rename (***string SubKey, string OldProfile, string NewProfile***)***;*

Remarks

OldProfile parameter is the name of the profile that is going to be renamed and NewProfile parameter is the new name of the profile.

See also Key and **FileName** properties, and **Delete** method.

4.4.4.3.6 SetCurrent

Sets the current profile.

VB.NET Syntax

*Profiles.***SetCurrent (***SubKey As String, Profile As String***)**

C# Syntax

*Profiles.***SetCurrent (***string SubKey, string Profile***)**;

Remarks

This method is generally used to set the default profile name to be loaded the next time the application starts.

4.4.4.4 Constants

4.4.4.4.1 Key

These values are used in the method Enumerate of the Profiles control.

Value
Tn3270
Tn5250
Keyboard3270
Keyboard5250
Display
Macro

4.5 Helper Controls

Find in the topics within this section a complete reference for the TNBridge Host Integration Pack .NET Helper Controls

4.5.1 XmlTemplate Class

4.5.1.1 XmlTemplate Class

The XmlTemplate class implements the runtime side for working with XML Templates created with Development Lab. It implements the IXmlProducer interface and can be assigned to Producer property of XmlBroker to produce custom XML.

Properties

- **Id**
- **Telnet**

Methods

- HostToXml
- XmlToHost
- CanProduce

Events

- **OnCustomXml**

4.5.1.2 Properties

4.5.1.2.1 Telnet

Sets/gets the telnet client component for working its fields.

VB.NET Syntax

*xmlTemplate.***Telnet** [= *tnxxxxx*]

C# Syntax

*xmlTemplate.***Telnet** [= *tnxxxx*];

4.5.1.2.2 Id

Gets an Id for the active XML Template.

VB.NET Syntax

[*String* =] *XmlTemplate.***Id**

C# Syntax

[*String* =] *XmlTemplate.***Id**;

Remarks

Returns a string with an identification corresponding to the active XML Template. This Id is equivalent to the filename of the XML template excluding the extension.

4.5.1.3 Methods

4.5.1.3.1 CanProduce

Returns true if the XmlTemplate can produce Xml.

VB.NET Syntax

[*Boolean* =] *XmlTemplate.***CanProduce**

C# Syntax

[*Boolean* =] *XmlTemplate.***CanProduce**;

Remarks

When you call CanProduce method it tries to find an XML template that matches with the current Screen.
In case it finds it, returns true telling that it can produce an XML content. In other case returns false.

4.5.1.3.2 HostToXml

Returns a XML representation of the current mainframe screen.

VB.NET Syntax

[*String* =] *XmlTemplate.***HostToXml**

C# Syntax

[*String* =] *XmlTemplate.***HostToXml**;

Remarks

Returns an XML content based on the XML template that matches with the current screen.

4.5.1.3.3 XmlToHost

Interpretes the input XML to fill the unprotected fields.

VB.NET Syntax

*XmlTemplate.***HostToXml**(Xml As String)

C# Syntax

*XmlTemplate.***HostToXml**(Xml:String);

Remarks

Interpretes the XML passed as parameter and fill unprotected fields, based on the active XML template for the current screen.

4.5.1.4 Events

4.5.1.4.1 OnCustomXml

Occurs during the processing of a custom XML area.

Remarks

When this event fires you have the chance to produce your custom XML content for that area.

4.5.2 XmlBroker Component

4.5.2.1 XmlBroker Component

This component acts as broker to convert screen data to XML code and back.

Properties

- **FieldsType**
- **IncludeStatus**
- **ScreenName**
- **SessionId**
- **Telnet**
- **Xml**
- **XmlProducer**

Methods

- **LoadFromFile**

- **SaveToFile**

See also **XML Reference** for XmlBroker.

4.5.2.2 Properties

4.5.2.2.1 FieldsType

Determines whether the Fields type are HostFields or Fields.

VB.NET Syntax

*xmlBroker.***FieldsType** [= *FieldsMode*]

C# Syntax

*xmlBroker.***FieldsType** [= *FieldsMode*];

4.5.2.2.2 IncludeStatus

Determines whether the Status tag and inner elements are included or not. Default value is true.

VB.NET Syntax

*xmlBroker.***IncludeStatus** [= *Boolean*]

C# Syntax

*xmlBroker.***IncludeStatus** [= *boolean*];

4.5.2.2.3 ScreenName

Sets/gets the corresponding screen name.

VB.NET Syntax

*xmlBroker.***ScreenName** [= *String*]

C# Syntax

*xmlBroker.***ScreenName** [= *string*];

See also **XML Reference** for XmlBroker.

4.5.2.2.4 SessionId

Retrieves the Id of the active Session.

VB.NET Syntax

[*String* =] *XmlBroker.***SessionId**

C# Syntax

[*String* =] *XmlBroker.***SessionId**;

4.5.2.2.5 Telnet

Sets/gets the telnet client component for working its fields.

VB.NET Syntax

*xmlBroker.***Telnet** [= *tnxxxxx*]

C# Syntax

*xmlBroker.***Telnet** [= *tnxxxx*];

4.5.2.2.6 Xml

When assigning an XML code, it generates a Fields collection. Reading this property, generates and returns an XML code from unprotected Fields, cursor location and AID Key.

VB.NET Syntax

*XmlBroker.***Xml** [= *String*]

C# Syntax

*XmlBroker.***Xml** [= *String*];

See also **XML Reference** for XmlBroker.

4.5.2.2.7 XmlProducer

Assigns an XML object capable to produce XML content.

VB.NET Syntax

*xmlBroker.***XmlProducer** [= *IXmlProducer*]

C# Syntax

*xmlBroker.***XmlProducer** [= *IXmlProducer*];

See also **XmlTemplate** class

4.5.2.3 Methods

4.5.2.3.1 LoadFromFile

Imports an XML file and then sets the **XML** property with it.

VB.NET Syntax

*XmlBroker.*LoadFromFile *(XMLFile As String)*

C# Syntax

*XmlBroker.*LoadFromFile *(string XMLFile);*

4.5.2.3.2 SaveToFile

Exports the XML code corresponding to the *XmlBroker object* with all the unprotected fields and the screen configuration data.

VB.NET Syntax

*XmlBroker.*SaveToFile *(XMLFile As String)*

C# Syntax

*XmlBroker.*SaveToFile *(string XMLFile);*

4.5.2.4 XML Reference

This XML code is generated automatically by the XmlBroker control. The following is the tags' hierarchy it uses:

```
<TNBRIDGE>
    <status>
        <direction... />
        <transaction... />
        <cursor_pos... />
        <session... />
        <state... />
        <xmlscreen... />
        <cursor_field... />
        <timestamp... />
    <status />
    <screen>
        <rows... />
        <cols... />
        <type... />
        <model... />
        <fields>
            <field ...>
                <name... />
                <attr... />
                <value... />
            <field />
            .
            .
            .
        <fields />
    <screen />
<TNBRIDGE />
```

4.5.2.4.1 XML Tags and Attributes

4.5.2.4.1.1 TNBRIDGE

This is the root tag and includes all the other tags. These are the following:

- Status: indicates connection characteristics.
- Screen: indicates screen's characteristics and contains a collection of screen fields.

See also **XML Example**.

4.5.2.4.1.2 Status

Contains the following tags:

- direction: indicates INPUT or OUTPUT direction. When the XML is generated by the XmlBroker control it is set with OUTPUT value.
- transaction: indicates transaction identifier.
- cursor_pos: indicates cursor location.
- session: LU-LU (in session with VTAM application), noSession (not in session) or SSCPLU (in session with VTAM).
- state: indicates LOCKED or UNLOCKED state.
- xmlscreen: indicates the screen name. You must set this propery using the **ScreenName** property.
- cursor_field: indicates the field's name where the cursor is located.
- timestamp: indicates the date the file was created.

See also **XML Example** and **ScreenName** property.

4.5.2.4.1.3 Screen

Contains the following tags:

- rows: indicates the number of rows of the screen.
- cols: indicates the number of columns of the screen.
- type: indicates terminal type (3270 or 5250).
- model: indicates terminal model.
- fields: contains a collection of fields.

See also **XML Example**.

4.5.2.4.2 XML Example

Here you can see an example of XML Code:

```
<RIDGE>
     <status>
            <direction>OUTPUT</direction>
            <transaction>00A8DB7000002</transaction>
            <cursor_pos>830</cursor_pos>
            <session>LU-LU</session>
            <state>UNLOCKED</state>
            <xmlscreen name="Session" />
            <cursor_field>R11C30</cursor_field>
            <timestamp>2453360.14494451</timestamp>
     </status>
     <screen>
            <rows>24</rows>
            <cols>80</cols>
            <type>TN3270</type>
            <model>2E</model>
            <fields>
                   <field name="R01C02">
                          <name len="4">R1C2</name>
                          <attr byte="E4" eab="08" display="y" prot="y" numeric="n"
intense="n" mdt="n" />
                          <value maxlen="8" len="8">KLGLGON1</value>
                   </field>
                   <field name="R01C11">
                          <name len="5">R1C11</name>
                          <attr byte="E4" eab="28" display="y" prot="y" numeric="n"
intense="n" mdt="n" />
                          <value maxlen="13" len="13">-------------</value>
                   </field>
                   .
                   .
                   .
            </fields>
     </screen>
</RIDGE>
```

4.5.3 XmlClient Component

4.5.3.1 XmlClient Component

This control acts like a virtual Tn3270/Tn5250 component. It generates the Fields necessary as input for the emulation control or any other user interface from XML code as well generates the XML code from the modified Fields, cursor location and 'pressed' AID key. When used in conjunction with the XmlBroker component allows build a bridge between the 3270/5250 Telnet components and the emulation interface using XML streaming. This bridge could be implemented as a client/server comunication using TCP/IP protocols, Web Services, etc.

Remarks

Since this component shares a common programming interface with the other Telnet controls, the properties, methods and events described in Tn5250/Tn3270/XmlClient/XmlVirtual Components may also apply.

The *following members* are specific to this component or override the common behavior:

Properties

- **XML**

Methods

- Connect
- Disconnect
- **LoadFromXMLFile**
- **SaveToXMLFile**

Events

- **OnConnect**
- **OnConnectionFail**
- **OnDisconnect**
- **OnScreenChange**
- **OnSendAid**
- **OnSystemLock**
- **OnSystemUnlock**

4.5.3.2 Properties

4.5.3.2.1 XML

When assigning an XML code, it generates a Fields collection. Reading this property, generates and returns an XML code from unprotected Fields, cursor location and AID Key.

VB.NET Syntax

XmlClient.**XML** [= *String*]

C# Syntax

XmlClient.**XML** [= *string*];

4.5.3.3 Methods

4.5.3.3.1 Connect

The Connect method initializes the connection sequence process.

VB.NET Syntax

*XmlClient.***Connect**

C# Syntax

*XmlClient.***Connect()**;

See also **Disconnect** method, **OnConnect** and **OnConnectionFail** events.

4.5.3.3.2 Disconnect

The Disconnect method ends the connection sequence process.

VB.NET Syntax

*XmlClient.***Disconnect**

C# Syntax

*XmlClient.***Disconnect()**;

See also **Connect** method and **OnDisconnect** event.

4.5.3.3.3 LoadFromFile

Imports an XML file and then sets the **XML** property with it.

VB.NET Syntax

*XmlClient.*LoadFromFile*(XMLFile As String)*

C# Syntax

*XmlClient.*LoadFromFile*(string XMLFile);*

4.5.3.3.4 SaveToFile

Exports the XML code corresponding to the *XmlClient object* with all the unprotected fields and the screen configuration data.

VB.NET Syntax

*XmlClient.*SaveToFile*(XMLFile As String)*

C# Syntax

*XmlClient.*SaveToFile*(string XMLFile);*

4.5.3.4 Events

4.5.3.4.1 OnConnect

Occurs after the program called the **Connect** method, only if the connection has been successfully established.

See also **Connect** method, **OnConnectionFail** and **OnDisconnect** events.

4.5.3.4.2 OnConnectionFail

Occurs after the program called the **Connect** method, only if the connection couldn't be established.

See also **Connect** method, **OnConnect** and **OnDisconnect** events.

4.5.3.4.3 OnDisconnect

Occurs when the **Disconnect** method is called.

See also **Disconnect** method, **OnConnect** and OnDisconnect events.

4.5.3.4.4 OnScreenChange

Occurs when an valid XML is assigned, either thru XML is property or LoadFromXmlFile method.

See also XML property, **OnSystemLock** and **OnSystemUnlock** events.

4.5.3.4.5 OnSendAid

Occurs before an AID key is about to be sent.

Remarks

This event is fired when a **Press**, **PressAndWait**, **Type** methods or the **AidKey** property are used. Occurs before the data is sent, allowing to take actions like filling edit fields or saving data typed in the emulator component.

See also **Press**, **PressAndWait**, **Type** methods and **AidKey** property.

4.5.3.4.6 OnSystemLock

Occurs when the XML stream changes to a locked state.

Remarks

During this state, sending data isn't possible until the **OnSystemUnlock** event is fired.

See also **OnSystemUnlock** event.

4.5.3.4.7 OnSystemUnlock

Occurs when the XML stream changes to an unlocked state.

Remarks

During this state, sending data is possible until the **OnSystemLock** event occurs.

See also **OnSystemLock** event.

4.5.4 XmlVirtual Component

4.5.4.1 XmlVirtual Component

This component allows you to create a simulated host application by combining XML-screen files and code to drive the screen navigation. XML-screen files can be taken using Development Lab.

Remarks

Since this component shares a common programming interface with the other Telnet controls, the properties, methods and events described in Tn5250/Tn3270/XmlClient/XmlVirtual Components may also apply.

The *following members* are specific to this component or override the common behavior:

Properties

- **BaseDirectory**
- **CurrentScreen**
- **StartFilename**
- **XML**

Methods

- Connect
- Disconnect
- **LoadScreen**

Events

- **OnConnect**
- **OnConnectionFail**
- **OnDisconnect**
- **OnScreenChange**
- **OnSendAid**
- **OnSystemLock**
- **OnSystemUnlock**
- **OnNewScreen**

4.5.4.2 Properties

4.5.4.2.1 BaseDirectory

Gets/sets the base directory from where Screen XML files will be loaded.

VB.NET Syntax

[*String* =] *XmlVirtual.***BaseDirectory**

C# Syntax

[*String* =] *XmlVirtual.***BaseDirectory**;

Remarks

BaseDirectory is used to automatically load screen files when you press PgDown/PgUp keys.

See also **OnNewScreen** event and **StartFilename** property.

4.5.4.2.2 CurrentScreen

Gets the name of current XML filename used to render the current screen.

VB.NET Syntax

[*String* =] *XmlVirtual.***CurrentScreen**

C# Syntax

[*String* =] *XmlVirtual.***CurrentScreen**;

4.5.4.2.3 StartFilename

Gets/Sets the start XML File.

VB.NET Syntax

[*String* =] *XmlVirtual.***StartFileName**

C# Syntax

[*String* =] *XmlVirtual.***StartFileName**;

Remarks

When you call the connect method the StartFilename will be the one that will be used to provide the content for the first screen. BaseDirectory will be set to the directory where StartFilename belongs.

See also **OnNewScreen** event and **BaseDirectory** property.

4.5.4.2.4 XML

When assigning an XML code, it generates a Fields collection. Reading this property, generates and returns an XML code from unprotected Fields, cursor location and AID Key.

VB.NET Syntax

*XmlVirtual.***XML** [= *String*]

C# Syntax

*XmlVirtual.***XML** [= *String*];

4.5.4.3 Methods

4.5.4.3.1 Connect

The Connect method initializes the connection sequence process. It loads the **StartFilename** XML File.

VB.NET Syntax

*XmlVirtual.***Connect**

C# Syntax

*XmlVirtual.***Connect**;

See also **Disconnect** method, **OnConnect** and **OnConnectionFail** events.

4.5.4.3.2 Disconnect

The Disconnect method ends the connection sequence process.

VB.NET Syntax

*XmlVirtual.***Disconnect**

C# Syntax

*XmlVirtual.***Disconnect**;

See also **Connect** method and **OnDisconnect** event.

4.5.4.3.3 LoadScreen

Loads an XML file to provide with new screen content.

VB.NET Syntax

[*Boolean* =] *XmlVirtual.*LoadScreen (*xmlFileName As string*)

C# Syntax

[*bool* =] *XmlVirtual.*LoadScreen (*string xmlFileName*);

4.5.4.4 Events

4.5.4.4.1 OnConnect

Occurs when the **Connect** method is called and the **StartFilename** is loaded.

See also **Connect** method and **OnConnectionFail** and **OnDisconnect** events.

4.5.4.4.2 OnConnectionFail

Occurs when the program fails to load the **StartFilename** XML File.

See also **Connect** method, **OnConnect** and **OnDisconnect** events.

4.5.4.4.3 OnDisconnect

Occurs when the **Disconnect** method is called.

See also **Disconnect** method and **OnConnect** and OnDisconnect event.

4.5.4.4.4 OnNewScreen

Occurs when a new XML file should be loaded.

Remarks

When this event is fired, you can analize **EditFields** and **AidKey** to determine the next screen you should load using **LoadScreen** method.

See also **LoadScreen** method, **HostFields** and **EditFields** properties.

4.5.4.4.5 OnScreenChange

Occurs when an valid XML is assigned, either thru XML is property or LoadFromXmlFile method.

See also XML property, **OnSystemLock** and **OnSystemUnlock** events.

4.5.4.4.6 OnSendAid

Occurs before an AID key is about to be sent.

Remarks

This event is fired when a **Press**, **PressAndWait**, **Type** methods or the **AidKey** property are used. Occurs before the data is sent, allowing to take actions like filling edit fields or saving data typed in the emulator component.

See also **Press**, **PressAndWait**, **Type** methods and **AidKey** property.

4.5.4.4.7 OnSystemLock

Occurs when the XML stream changes to a locked state.

Remarks

During this state, sending data isn't possible until the **OnSystemUnlock** event is fired.

See also **OnSystemUnlock** event.

4.5.4.4.8 OnSystemUnlock

Occurs when the XML stream changes to an unlocked state.

Remarks

During this state, sending data is possible until the **OnSystemLock** event occurs.

See also **OnSystemLock** event.

4.6 Trace Services

Find in the topics within this section a complete reference for the TNBridge Host Integration Pack Trace Services

4.6.1 Setting up Trace Services

To trace a TN Bridge application you need to setup the Trace Agent built into the **Trace** Control.
The following steps shows you how to setup the Trace Services to work with your application:

- Drop a **Trace** control into a form.
- Set the **Telnet** property of the trace control to the Telnet control:

```
        Trace1.Telnet = Telnet
```

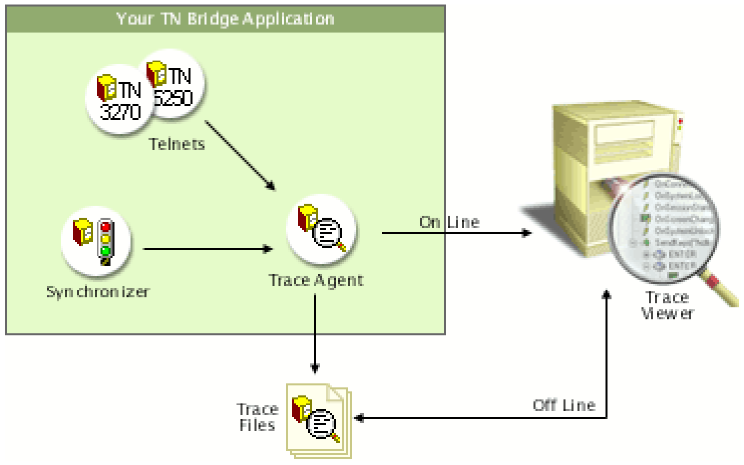
- Set the TCP/IP listening port to the **Port** property (at design time or at run time):

```
        Trace1.Port = 1024
```

- Set the **Active** property to true. When doing this at design-time, the activation takes place at run-time:

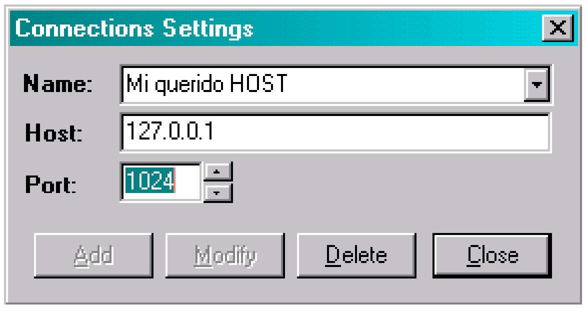
```
        Trace1.Active = True
```

- Run your application.
- Each time the telnet control connects to a mainframe, a trace file is generated with a connection identifier as its name and extension the ".hst". This file will be stored into the folder specified in Trace Directory property or in windows temporary files folder. By default, this files will be removed when the trace becomes inactive, but you can change this behavior setting the **PurgeTraceFiles** property to **False**.



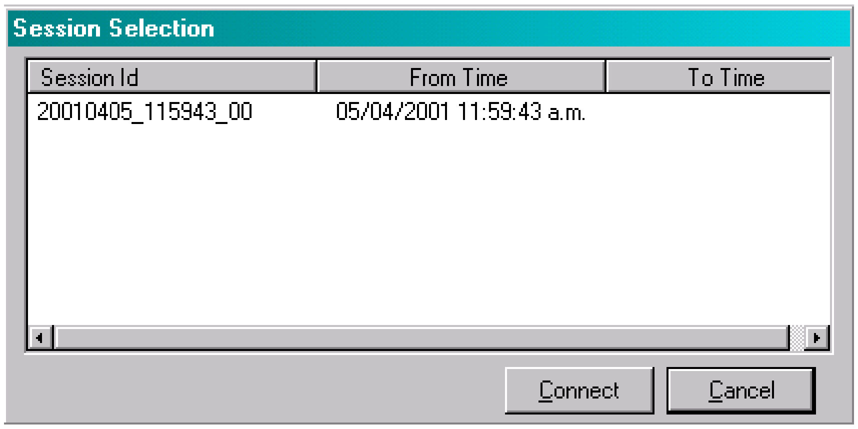
For on-line monitoring:

- Open the TN Bridge Trace Viewer and setup a connection pointing to the machine IP address where your TN Bridge application is running. Also set the TCP **Port** to the listening port specified in your **Trace** control. This is the Connections Settings dialog box where you can complete the information:



Name: Mi querido HOST
Host: 127.0.0.1
Port: 1024

- Click the connect button. A dialog box with the available connections will be shown. Choose one from the list and click the Connect button.



Session Id	From Time	To Time
20010405_115943_00	05/04/2001 11:59:43 a.m.	

For off-line monitoring:

Open the TN Bridge Trace Viewer and choose the file menu and open menu item. Select the trace file previously generated by your TN Bridge application. (Remember that this file will be available only if you set the Trace's **PurgeTraceFiles** property to False. Also, is strongly recommended to set the **Directory** property to an existing a known folder).

4.6.2 TN Bridge Trace Server

4.6.2.1 Trace Control

This control implements a Trace Agent. It allows real-time monitoring of Telnet events, analyses mainframe's screen information and synchronized methods calls.

Properties

- **Active**
- **CurrentFileName**
- **Directory**
- **HidePasswordFields**

- Port
- PurgeTraceFiles
- Telnet
- Visible

Methods

- HideTrace
- ShowTrace
- TraceText

4.6.2.2 Properties

4.6.2.2.1 Active

Activates or desactivates Trace Agent functionality.

Vb.NET Syntax

*Trace.***Active** [= *Boolean*]

C# Syntax

*Trace.***Active** [= *bool*];

Remarks

If you assign the value of this property at design time, it acts only as the initial value for run-time.

Default value is **False**.

4.6.2.2.2 CurrentFileName

Retrieve the current trace filename.

Vb.NET Syntax

[*Boolean* =] *Trace.***CurrentFileName**

C# Syntax

[*bool* =] *Trace.***CurrentFileName**;

4.6.2.2.3 Directory

Sets/gets the folder directory name where the trace files will be saved.

Vb.NET Syntax

*Trace.***Directory** [= *String*]

C# Syntax

*Trace.***Directory** [= *string*];

Remarks

If you don't specify a directory name, the trace files will be saved into the windows temporary file directory.

4.6.2.2.4 HidePasswordFields

Enables or disables the password masking mode for hiding data of password fields.

Vb.NET Syntax

*Trace.***HidePasswordFields** [= *Boolean*]

C# Syntax

*Trace.***HidePasswordFields** [= *bool*];

Remarks

When this property is set to **True**, password fields are masked with asterisks.

Default value is **True**.

4.6.2.2.5 Port

Sets/gets the listening TCP port number for the current application process.
The Trace control works at process level, generating independent traces on a connection basis.

Vb.NET Syntax

*Trace.***Port** [= *Integer*]

C# Syntax

*Trace.***Port** [= *int*];

Remarks

The default port value is 1024.

4.6.2.2.6 PurgeTraceFiles

Allows to purge trace files when the server becomes inactive.

Vb.NET Syntax

*Trace.***PurgeTraceFiles** [= *Boolean*]

C# Syntax

*Trace.***PurgeTraceFiles** [= *bool*];

Remarks

By default, trace information is kept into temporary files. When the **Active** property is set to **False** or the process is finished, these files are deleted.

Default value is **True**.

4.6.2.2.7 Telnet

Sets/gets the Tn3270 or Tn5250 Control as the telnet client control.

Vb.NET Syntax

*Trace.***Telnet** [= *IComApi*]

C# Syntax

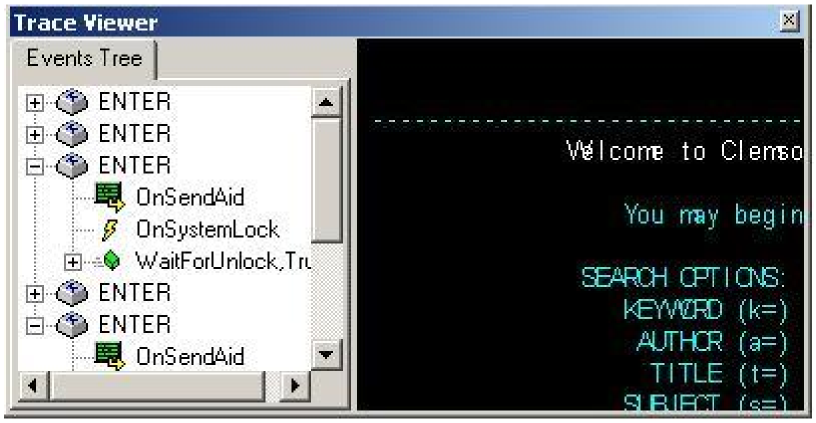
*Trace.***Telnet** [= *IComApi*];

Remarks

You must set this property to allow Trace component to work properly.

4.6.2.2.8 Visible

Activates or deactivates Trace Viewer window. It can be used at the same time you are developing an Integration Pack application, helping about host events that happen when application code is running. Trace Viewer window helps to follow those host events as you modify application's code.



```
-------------------------------
Welcome to Clemso
You may begin
SEARCH OPTIONS:
KEYWORD (k=)
AUTHOR (a=)
TITLE (t=)
SUBJECT (s=)
```

Vb.NET Syntax

*Trace.***Visible** [= *Boolean*]

C# Syntax

*Trace.***Visible** [= *bool*];

Remarks

If you set the value of this property at design time to True, Trace Viewer window is shown before you run Integration Pack application, and will be prepared to show hosts events at application run-time.

Default value is **False**.

4.6.2.3 Methods

4.6.2.3.1 HideTrace

Hides Trace Viewer window. It can be used at the same time you are developing an Integration Pack application, helping you with host events that happen when application code is running.

Vb.NET Syntax

*Trace.***HideTrace**

C# Syntax

*Trace.***HideTrace**;

Remarks

This method is used in developing Integration Pack's application process.

See also ShowTrace method.

4.6.2.3.2 ShowTrace

Shows Trace Viewer window. It can be used at the same time you are developing an Integration Pack application, helping about host events that happen when application code is running.

Vb.NET Syntax

*Trace.***ShowTrace**

C# Syntax

*Trace.***ShowTrace**;

Remarks

This method is used in developing Integration Pack's application process.

See also **HideTrace** method.

4.6.2.3.3 TraceText

Allows to insert custom string data into the trace file.

Vb.NET Syntax

*Trace.***TraceText (***Msg As String***)**

C# Syntax

*Trace.***TraceText (***string Msg***)**;

4.7 Exceptions

The following exceptions are thrown by the TN Bridge .NET Components:

Cybele.TNBridge.InvalidLicenseException
This exception is thrown when you are trying to use a component without a valid license.

Cybele.TNBridge.LicenseLimitReachedException
This exception is thrown when you are trying to use a component without a enough licenses.

Cybele.TNBridge.ProfileException
This exception is thrown when the Profile control fails to delete or rename a profile.

Cybele.TNBridge.EntryErrorException
This exception is thrown when the Telnet.SendKeys method founds an Invalid Cursor Position.

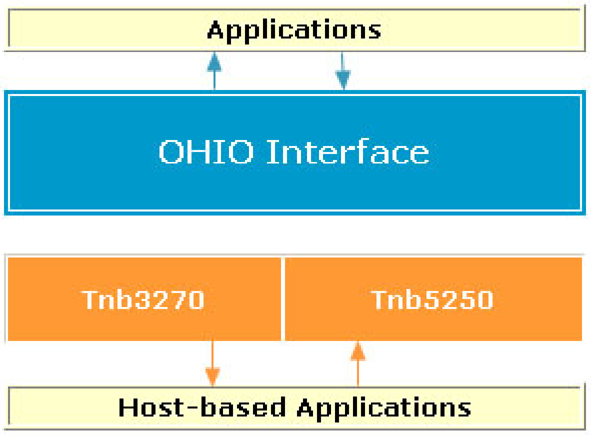
Cybele.TNBridge.InvalidParameterException &
Cybele.TNBridge.ParameterErrorException
These exceptions are thrown when you are trying to assign or send an invalid parameter to a Telnet control.

5 OHIO Reference

- **Introduction to OHIO**
- **Programming Reference**

5.1 Introduction

Open Host Interface Objects (OHIO) is an IBM and Attachmate inspired object-oriented host access API for software implementing tn3270 and tn5250 protocols. Vendor neutral and submitted to the IETF in 1998 as an Internet standard.
OHIO address the need for a standardized advanced programming interface to the host data. OHIO does not modify the TN3270/TN5250 protocol or datastream but instead provides a common access method to that data once it arrives at the client. It uses an Object Oriented approach to divide the data into logical objects, and provides methods on those objects to allow standard access to the data.



Some OHIO's advantages are:

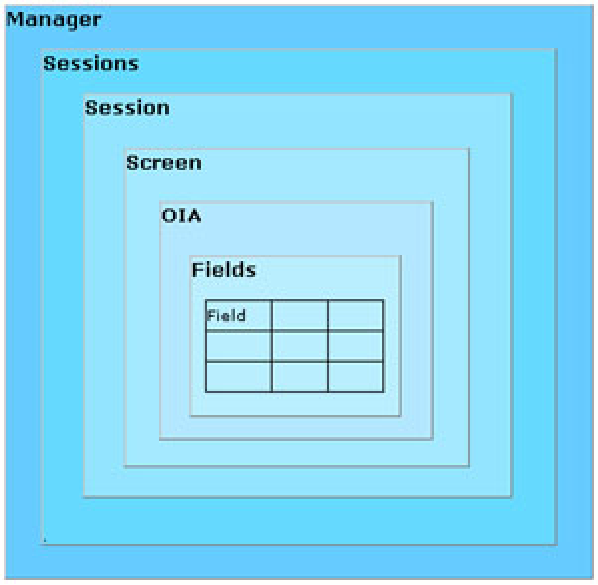
- You can run an OHIO application without having an emulation Session running.
- OHIO has all the benefits of the object oriented programming paradigm because it was designed as an object oriented API.
- You can concentrate on the application functions, without worrying about structure packing details or parameter command codes.

In this chapter we detail:

- **Ohio containment hierarchy**
- **Ohio inheritance hierarchy**

5.1.1 Ohio containment hierarchy

The following is the Ohio containment hierarchy:



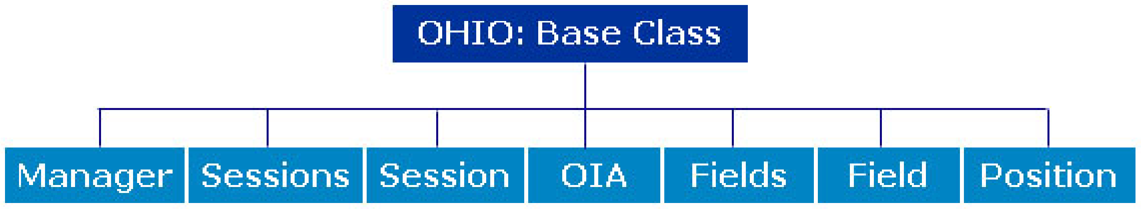
Additional utility classes:

- **Position**

Note: "1" based counting is used throughout this document for both positions (the first position on the Screen is position 1, not position 0) and sizes (the first item in a collection is item 1, not item 0).

5.1.2 Ohio inheritance hierarchy

The Ohio inheritance hierarchy is:



5.2 Programming Reference

In TN Bridge Host Integration Pack for .NET, OHIO has its own namespace (Cybele.TNBridge.Ohio), and all its classes are contained in it.
In this chapter we describe the following topics:

- **Ohio Base Class**
- **Class Definition**
- **Constants**

5.2.1 Ohio Base Class

This is the base class for all Ohio classes. Contains all common Ohio methods and properties.

See also Ohio Base Class **Properties** and **Methods**.

5.2.1.1 Syntax Conventions

The following text formats are used throughout the Components Reference:

For each property and method:

Italic text Denotes an object, in this case a TNBridge control.
Bold text Denotes a property or a method names.
[= value] or [= value] Denotes values to be assigned.
[value =] or [value =] Denotes return values.
(TimeOut**)** Denotes a parameter to be passed to a method.
([TimeOut]**)** Denotes an optional parameter to be passed to a method.

Note: When we refer to *OhioObject* we are indicating that this object can be any of the objects that belong to the Ohio Base Class.

5.2.1.2 Methods

- **CreateOhioPosition**

5.2.1.2.1 CreateOhioPosition

Creates an Position object.

VB.NET

*[Position =] OhioObject.***CreateOhioPosition(**row As Integer, col As Integer**)**

C#

*[Position =] OhioObject.***CreateOhioPosition(**int row, int col**)**;

Parameters

Parameter	Description

row	The row coordinate
col	The column coordinate

5.2.1.3 Properties

- **OhioVersion**
- **VendorName**
- **VendorObject**
- **VendorProductVersion**

5.2.1.3.1 OhioVersion

Returns the Ohio Version level for this implementation (OHIO 1.00).

VB.NET

[String =] OhioObject.**OhioVersion**

C#

*[String =] OhioObject.***OhioVersion**;

5.2.1.3.2 VendorName

Returns the name of the vendor providing this OHIO implementation. Format is vendor defined.

VB.NET

[String =] OhioObject.**VendorName**

C#

*[String =] OhioObject.***VendorName**;

5.2.1.3.3 VendorObject

Returns the VendorObject for the current Session. This property returns different objects according to the object who calls it.

VB.NET

[Telnet =] Screen.**VendorObject**

[Telnet =] Session.**VendorObject**

[OIA =] OIA.**VendorObject**

[Sessions =] Sessions.**VendorObject**

[Field =] Field.**VendorObject**

C#

*[Telnet =] Screen.***VendorObject**;

[Telnet =] Session.**VendorObject**;

[OIA =] OIA.**VendorObject**;

[Sessions =] Sessions.**VendorObject**;

*[Field =] Field.***VendorObject**;

5.2.1.3.4 VendorProductVersion

Indicates the vendor product version that is providing the OHIO implementation. Format is vendor specific.

VB.NET

[String =] OhioObject.**VendorProductVersion**

C#

[String =] OhioObject.**VendorProductVersion**;

5.2.2 Class Definition

- **OhioManager**
- **Sessions**
- **Session**
- **Screen**
- **Fields**
- **Field**
- **OIA**
- **Position**

5.2.2.1 Syntax Conventions

The following text formats are used throughout the Components Reference:

For each property and method:

Italic text Denotes an object, in this case a TNBridge control.
Bold text Denotes a property or a method names.
[= value] or [= value] Denotes values to be assigned.
[value =] or [value =] Denotes return values.
(TimeOut**)** Denotes a parameter to be passed to a method.
([TimeOut]**)** Denotes an optional parameter to be passed to a method.

5.2.2.2 OhioManager

The OhioManager is the central repository for access to all OHIO Sessions. It contains a list of all Session objects available on this system.

See also OhioManager **Properties** and **Methods**.

5.2.2.2.1 Methods

- **OpenSession**
- **CloseSession**

5.2.2.2.1.1 OpenSession

Returns an Session object based on the search parameters provided.

VB.NET

*[Session =] OhioManager.***OpenSession(**ConfigurationResource As Integer, SessionName As String**)**

*[Session =] OhioManager.***OpenSession(**ConfigurationResource As IComApi, SessionName As String**)**

*[Session =] OhioManager.***OpenSession(**ConfigurationResource As String, SessionName As String**)**

C#

*[Session =] OhioManager.***OpenSession(**int configurationResource, string sessionName**)**;

*[Session =] OhioManager.***OpenSession(**IComApi configurationResource, string sessionName**)**;

*[Session =] OhioManager.***OpenSession(**string configurationResource, string sessionName**)**;

Parameters

Parameter	**Description**
ConfigurationResource	A vendor specific string used to provide configuration information

SessionName	The unique name associated with an Session

The parameters are used as follows:

```
Is ConfigurationResource provided?
  Yes - Is SessionName provided?
    Yes - Is Session object with matching
        SessionName available on the system?
      Yes - Error, attempting to create an
          Session object with a non-unique
          SessionName.
      No - Create an Session object using
          SessionName and ConfigurationResource.
    No - Start a new Session using
        ConfigurationResource and generating a new
        SessionName.
  No - Is SessionName provided?
    Yes - Is Session object with matching
        SessionName available on the system?
      Yes - Return identified Session object.
      No - Return null.
    No - Return null.
```

5.2.2.2.1.2 CloseSession

Closes an Session object. The Session is considered invalid and is removed from the list of Session objects. You can use this methods with the following syntaxes:

VB.NET

OhioManager.**CloseSession(**SessionObject As Session**)**

OhioManager.**CloseSession(**SessionName As String**)**

C#

*OhioManager.***CloseSession(**Session sessionObject**)**;

*OhioManager.***CloseSession(**string sessionName**)**;

Parameters

Parameters	**Description**
SessionObject	The Session to close.
SessionName	The SessionName of the Session to close.

5.2.2.2.2 Properties

- **Sessions**

5.2.2.2.2.1 Sessions

Returns an Sessions object containing the Session objects available on this system. This list of objects is a static snapshot at the time the Sessions object is created.

VB.NET

*[Sessions =] OhioManager.***Sessions**

C#

*[Sessions =] OhioManager.***Sessions**;

5.2.2.3 Sessions

Contains a collection of Session objects. This list is a static snapshot of the list of Session objects available at the time of the snapshot.

See also Sessions **Properties** and **Methods**.

5.2.2.3.1 Methods

- **AddSession**
- **CloseSession**

5.2.2.3.1.1 AddSession

Adds a Session to the Sessions collection based on the specified configuration file.

VB.NET

[Session =] Sessions.**AddSession(**ConfigurationResource As String, SessionName As String**)**

[Session =] Sessions.**AddSession(**ConfigurationResource As Integer, SessionName As String**)**

[Session =] Sessions.**AddSession(**ConfigurationResource As IComApi, SessionName As String**)**

C#

*[Session =] Sessions.***AddSession(**string configurationResource, string sessionName**)***;*

*[Session =] Sessions.***AddSession(**int configurationResource, string sessionName**)***;*

*[Session =] Sessions.***AddSession(**IComApi configurationResource, string sessionName**);**

Parameters

Parameters	**Description**
ConfigurationResource	Full path of the Configuration File (.XML) to use for host connection information. This parameter can be a string, an integer or a IComApi type.
SessionName	Session's name.

5.2.2.3.1.2 CloseSession

Closes an Session object. The Session is considered invalid and is removed from the list of Session objects. You can use this methods with the following syntaxes:

VB.NET

Sessions.**CloseSession(**SessionObject As Session**)**

Sessions.**CloseSession(**SessionName As String**)**

C#

Sessions.**CloseSession(**Session sessionObject**)**;

Sessions.**CloseSession(**string sessionName**)**;

Parameters

Parameters	**Description**
SessionObject	The Session to close.
SessionName	The SessionName of the Session to close.

5.2.2.3.1.3 Refresh

Updates the collection of Session objects. All Session objects that are available on the system at the time of the refresh will be added to the collection.
Indexing of Session objects will not be preserved across refreshes.

VB.NET

*Sessions.***Refresh**

C#

*Sessions.***Refresh();**

Parameters

None.

5.2.2.3.2 Properties

- **Count**
- **Item**
- **Manager**

5.2.2.3.2.1 Count

Returns the number of Sessions objects contained in this collection (the number of Sessions currently opened).

VB.NET

*[Integer =] Sessions.***Count**

C#

*[int =] Sessions.***Count***;*

5.2.2.3.2.2 Item

Returns the Sessions object at the given index. "One based" indexing is used in all Ohio collections. For example, the first Session in this collection is at index 1.

VB.NET

[Session] = *Sessions.***Item(**index As Integer**)**
[Session] = *Sessions.***Item(**name As String**)**

C#

[Session] = *Sessions.***Item(**int index**)**;
[Session] = *Sessions.***Item(**string name**)**;

Parameters

Parameters	Description
index / name	The index / name of the target Session.

5.2.2.3.2.3 Manager

Returns the OhioManager object for the current Session.

VB.NET

*Sessions.***Manager** *[= OhioManager]*

C#

*Sessions.***Manager**; *[= OhioManager]*

5.2.2.4 Session

The Session represents a host Session.

See also Session **Properties**, **Methods** and **Events**.

5.2.2.4.1 Methods

- **Connect**
- **Disconnect**
- **WaitForConnect**
- **WaitForDisconnect**

5.2.2.4.1.1 Connect

Starts the communications link to the host.

VB.NET

*Session.***Connect**

C#

*Session.***Connect**;

Parameters

None.

5.2.2.4.1.2 Disconnect

Stops the communications link to the host.

VB.NET

*Session.***Disconnect**

C#

*Session.***Disconnect**;

Parameters

None.

5.2.2.4.1.3 WaitForConnect

This method waits until the telnet connection is successfully opened, or the timeout period expires.

VB.NET

*[Boolean =] Session.***WaitForConnect(**[TimeOut] As Integer**)**

C#

*[bool =] Session.***WaitForConnect(**int [TimeOut]**)***;*

Remarks

The Timeout parameter is optional. This method returns True if the wait was successful and False if the Timeout period has expired. The TimeOut parameter is measured in milliseconds.
If you pass a timeout value in the Timeout parameter of this method, the value of the property WaitTimeout will not have any effect.

5.2.2.4.1.4 WaitForDisconnect

This method waits until the telnet connection is successfully closed, or the timeout period expires.

VB.NET

*[Boolean =] Session.***WaitForDisconnect(**[TimeOut] As Integer**)**

C#

*[bool =] Session.***WaitForDisconnect(**int [TimeOut]**)**;

Remarks

The Timeout parameter is optional. This method returns True if the wait was successful and False if the Timeout period has expired. The TimeOut parameter is measured in milliseconds.
If you pass a timeout value in the Timeout parameter of this method, the value of the property WaitTimeout will not have any effect.

5.2.2.4.2 Properties

- **ConfigurationResource**
- **Connected**
- **Manager**
- **Screen**
- **SessionName**
- **SessionState**
- **SessionType**
- **Trace**

5.2.2.4.2.1 ConfigurationResource

Indicates the ConfigurationResource for this Session object.

VB.NET

*[String =] Session.***ConfigurationResource**

C#

*[String =] Session.***ConfigurationResource***;*

Remarks

This property returns the full path of the configuration resource file for the current Session.

5.2.2.4.2.2 Connected

Indicates whether this Session object is connected to a host.

VB.NET

*[Boolean =] Session.***Connected**

C#

*[bool =] Session.***Connected***;*

Remarks

True means connected, False means not connected.

5.2.2.4.2.3 Manager

Returns the OhioManager object for the current Session.

VB.NET

*Session.***Manager** *[= OhioManager]*

C#

*Session.***Manager**; *[= OhioManager]*

5.2.2.4.2.4 Screen

Returns the Screen object for the current Session.

VB.NET

*[Screen =] Session.***Screen**

C#

*[Screen =] Session.***Screen**;

5.2.2.4.2.5 SessionName

Returns the SessionName for this Session object.

VB.NET

*[String =] Session.***SessionName**

C#

[String =] Session.**SessionName**;

Remarks

The SessionName is unique among all instances of Session.

5.2.2.4.2.6 SessionState

Returns the SessionState for the Session object.

VB.NET

*[String =] Sessions.***SessionState**

C#

*[String =] Session.***SessionState***;*

5.2.2.4.2.7 SessionType

Returns the SessionType for this Session object.

VB.NET

*[Integer =] Session.***SessionType**

C#

*[int =] Session.***SessionType**;

5.2.2.4.2.8 Trace

Sets the TNBXTrace Control as its trace agent.

VB.NET

*Session.***Trace** [= *Trace*]

C#

*Session.***Trace**; [= *Trace*]

Remarks

You must set this property to get trace information from TNBXSync control.

5.2.2.4.3 Events

- **OnConnect**
- **OnConnectionFail**
- **OnDisconnect**
- **OnScreenChange**
- **OnSendAid**
- **OnSessionState**
- **OnSystemLock**
- **OnSystemUnlock**

5.2.2.4.3.1 OnConnect

Occurs after a successfully connection to the server is established.

5.2.2.4.3.2 OnConnectionFail

Occurs after the connection to the server fails.

5.2.2.4.3.3 OnDisconnect

Occurs after a disconnection of the server.

5.2.2.4.3.4 OnScreenChange

This event is generated whenever the virtual Screen is modified.

5.2.2.4.3.5 OnSendAid

Occurs before an Aid key is to be sent.

Remarks

This event is fired when a **SendAid** or **SendKeys** methods are used to send an Aid Key and data to the mainframe. Occurs before the data is sent, allowing to take actions like filling edit fields or saving data typed in the emulator control.

5.2.2.4.3.6 OnSessionState

Occurs when a Session-state change takes place.

Remarks

A Session state can be ssNoSession, ssSSCPLU and ssLULU. Any transition between the states fires this event.

5.2.2.4.3.7 OnSystemLock

Occurs when a terminal changes to a system locked state.

Remarks

During this state, the terminal is waiting for any response from the mainframe. Sending data isn't possible until the **OnSystemUnlock** event is fired.

See also **OnSystemUnlock** event.

5.2.2.4.3.8 OnSystemUnlock

Occurs when a terminal changes to a system unlocked state.

Remarks

Only during this state, the component can send data to the host system.

See also **OnSystemLock** and **OnScreenChange** events.

5.2.2.5 Screen

Screen encapsulates the host presentation space. The presentation space is a virtual Screen which contains all the characters and attributes that would be seen on a traditional emulator Screen. This virtual Screen is the primary object for text-based interactions with the host. The Screen provides methods that manipulate text, search the Screen, send keystrokes to the host, and work with the cursor.
An Screen object can be obtained from the Screen property of an instance of Session.
The raw presentation space data is maintained in a series of planes which can be accessed by various methods within this class.
The text plane contains the actual characters in the presentation space. Most of the methods in Screen class work exclusively with the text plane.
The remaining planes contain the corresponding attributes for each character in the text plane. The color plane contains color characteristics. The field plane contains the field attributes.
The extended plane contains the extended field attributes. The color, field, and extended planes are not interpreted by any of the methods in this class.

See Also Screen **Properties** and **Methods**.

5.2.2.5.1 Methods

- **GetData**
- **FindString**
- **Lock**
- **Refresh**
- **SendAid**
- **SendKeys**
- **SetString**
- **Unlock**
- **Wait**
- **WaitFor**
- **WaitForNewScreen**
- **WaitForNewUnlock**
- **WaitForScreen**
- **WaitForUnlock**

5.2.2.5.1.1 GetData

Returns a character array containing the data from the Text, Color, Field or Extended plane of the virtual Screen.

VB.NET

*[Char Array =] Screen.***GetData(**start As Position, end As Position, plane As Integer**)**

C#

*[char[] =] Screen.***GetData(**startpos, endpos:Position; plane:integer**)***;*

Parameters

Parameter	Description
start	The row and column where to start. The position is inclusive (for example, row 1, col 1 means that position 1,1 will be used as the starting location and 1,1 will be included in the data). "start" must be positionally less than "end".
end	The row and column where to end. The position is inclusive (for example, row 1, col 1 means that position 1,1 will be used as the ending location and 1,1 will be included in the data). "end" must be positionally greater than "start".
plane	A valid OHIO_PLANE value.

5.2.2.5.1.2 FindString

Searches the text plane for the target string. If found, returns an Position object containing the target location. If not found, returns a null. The TargetString must be completely contained by the target area for the search to be successful. Null characters in the text plane are treated as blank spaces during search processing.

VB.NET

*[Position =] Screen.***FindString(**TargetString As String, startPos As Position, length As Integer, dir As OHIO_DIRECTION, IgnoreCase As Boolean**)**

C#

*[Position =] Screen.***FindString(**string targetString, Position startPos, int length, int dir, bool ignoreCase**)**;

Parameters

Parameter	Description
TargetString	The target string.
startPos	The row and column where to start. The position is inclusive (for example, row 1, col 1 means that position 1,1 will be used as the starting location and 1,1 will be included in the search).
length	The length from startPos to include in the search.
dir	An OHIO_DIRECTION value.
IgnoreCase	Indicates whether the search is case sensitive. True means that case will be ignored. False means the search will be case sensitive.

5.2.2.5.1.3 Lock

Locks the current Screen content. Specially useful when checking the fields contents, call Lock() to prevent the fields contents to be modified when a new screen arrives from Host.

VB.NET

*Screen.***Lock**

C#

*Screen.***Lock()**;

Parameters

None.

5.2.2.5.1.4 SendAid

The SendAid method sends an "aid" keystroke to the virtual Screen. These aid keys can be though of as special
keystrokes, like the Enter key, the Tab key, or the Page Up key. All the valid special key values are contained in the OHIO_AID enumeration.

VB.NET

*Screen.***SendAid(**aidkey As Integer**)**

C#

*Screen.***SendAid(**int aidKey**)**;

Parameters

Parameters	**Description**
aidKey	The aid key to send to the virtual Screen.

5.2.2.5.1.5 SendKeys

The SendKeys method sends a string of keys to the virtual Screen. This method acts as if keystrokes were being typed from the keyboard.
The keystrokes will be sent to the location given. If no location is provided, the keystrokes will be sent to the
current cursor location.

VB.NET

*Screen.***SendKeys(**text As String, location As Position**)**

C#

*Screen.***SendKeys(**string text, Position location**)**;

Parameters

Parameters	Description
text	The string of characters to be sent.
location	Position where the string should be written.

5.2.2.5.1.6 SetString

The SetString method sends a string to the virtual Screen at the specified location. The string will overlay only unprotected fields, and any parts of the string which fall over protected fields will be discarded.

VB.NET

*Screen.***SetString(**text As String, location As Position**)**

C#

*Screen.***SetString(**string text, Position location**)**

Parameters

Parameters	Description
text	String to place in the virtual Screen
location	Position where the string should be written

5.2.2.5.1.7 Unlock

Unlocks the current Screen content. It will let the Screen's contents to update if a new screen arrives from Host.

VB.NET

*Screen.***Unlock**

C#

*Screen.***Unlock()**;

Parameters

None.

5.2.2.5.1.8 Wait

General wait mechanism. This method basically waits until the mainframe application turns in an unlocked state and is able to receive input data.

VB.NET

*[Boolean] = Screen.***Wait(**[TimeOut] As Integer**)**

C#

*[bool] = Screen.***Wait(**int [TimeOut]**)**

Remarks

The Timeout parameter is optional and it is measured in milliseconds. This method returns True if the wait was successful and False if the timeout period has expired.

See also **WaitForConnect**, **WaitForUnlock** and **WaitForNewScreen** methods.

5.2.2.5.1.9 WaitFor

This method waits for a Screen containing the specified string. In the first case shown below, it returns True if the wait was successful and False if the Timeout period has expired. In the second case it receives an array of
strings as parameter, and returns an integer indicating the number of the string (into the array of strings) that was found.

VB.NET

*[Boolean =] Screen.***WaitFor(**Value As String, [TimeOut] As Integer, [ResetWait] As Boolean**)**

*[Integer =] Screen.***WaitFor(**Values As String, [TimeOut] As Integer, [ResetWait] As Boolean**)**

C#

*[bool =] Screen.***WaitFor(**string Value, int [TimeOut], bool [ResetWait]**)**;

*[int =] Screen.***WaitFor(**string[] Values, int [TimeOut], bool [ResetWait]**)**;

Parameters

Parameters	Description
Value	The string of characters expected to be found.
Values	The array of strings where one of them is expected to be found. In Visual Basic this is a string with several values separated by ";"
TimeOut	Optional parameter measured in milliseconds.
ResetWait	While this optional parameter is false, continue lopping until the expected string is found.

5.2.2.5.1.10 WaitForNewScreen

This method waits until a new Screen arrives within the specified period of time. It is similar to WaitForScreen method except that it always waits for a new Screen arrival.

This method returns True if the wait was successful and False if the Timeout period has expired.

VB.NET

*[Boolean =] Screen.***WaitForNewScreen(**[TimeOut] As Integer**)**

C#

*[bool =] Screen.***WaitForNewScreen(**int [TimeOut]**)**;

Parameters

Parameters	Description
TimeOut	Optional parameter measured in milliseconds.

5.2.2.5.1.11 WaitForNewUnlock

This method waits until a new system unlock arrives within the specified period of time. It is similar to WaitForUnlock method except that it always waits for a new system unlock arrival. It returns True if the wait was successful and False if the Timeout period has expired.

VB.NET

*[Boolean =] Screen.***WaitForNewUnlock(**[TimeOut] As Integer**)**

C#

*[bool =] Screen.***WaitForNewUnlock(**int [TimeOut]**)**;

Parameters

Parameters	Description
TimeOut	Optional parameter measured in milliseconds.

5.2.2.5.1.12 WaitForScreen

This method waits for the first Screen after a system lock. If the host system changes from an unlocked to a locked state (normally when we send an Aid key) the method waits until the first Screen arrives. If WaitForScreen method is called after that event, it returns immediately. The method returns True if the wait was successful and False if the Timeout period has expired.

VB.NET

*[Boolean =] Screen.***WaitForScreen(**[TimeOut] As Integer**)**

C#

*[bool =] Screen.***WaitForScreen(**int [TimeOut]**)**;

Parameters

Parameters	Description
TimeOut	Optional parameter measured in milliseconds.

5.2.2.5.1.13 WaitForUnlock

This method waits until a new system unlock arrives within the specified period of time. It is similar to WaitForUnlock method except that it always waits for a new system unlock arrival. This method returns True if the wait was successful and False if the Timeout period has expired.

VB.NET

*[Boolean =] Screen.***WaitForUnlock(**[TimeOut] As Integer**)**

C#

*[bool =] Screen.***WaitForUnlock(**int [TimeOut]**)**;

Parameters

Parameters	Description
TimeOut	Optional parameter measured in milliseconds.

5.2.2.5.2 Properties

- **Columns**
- **Cursor**
- **Fields**
- **OIA**
- **Rows**
- **String/Text**

5.2.2.5.2.1 Columns

Indicates the number of columns in the presentation space.

VB.NET

*[Integer =] Screen.***Columns**

C#

*[int =] Screen.***Columns**;

Remarks

This property returns an integer representing the number of columns on the current host Screen.

5.2.2.5.2.2 Cursor

Specifies the location of the cursor in the presentation space.

VB.NET

*Screen.***Cursor** *[= Integer]*

C#

*Screen.***Cursor***; [= int]*

Remarks

This property returns an integer which represents the current cursor position, or sets the cursor position according to the *Position* integer value passed as parameter.

5.2.2.5.2.3 CursorObj

Specifies the location of the cursor in the presentation space. The row and column of the cursor is contained within the Position object.

VB.NET

*Screen.***Cursor** *[= Position]*

C#

*Screen.***Cursor***; [= Position]*

5.2.2.5.2.4 Fields

Returns the Fields object associated with this presentation space. This provides another way to access the data in the virtual Screen. The Fields object contains a snapshot of all the fields in the current virtual Screen. Fields provide methods for interpreting the data in the non-text planes. Zero length fields (due to adjacent field attributes) are not returned in the Fields collection.
For unformatted Screens, the returned collection contains only one Field that contains the whole virtual Screen.

VB.NET

*[Fields =] Screen.***Fields**

C#

*[Fields =] Screen.***Fields**;

Remarks

Returns the Fields object for the current host Screen.

5.2.2.5.2.5 Manager

Returns the OhioManager object for the current Screen.

VB.NET

*Screen.***Manager** *[= OhioManager]*

C#

*Screen.***Manager**; *[= OhioManager]*

5.2.2.5.2.6 OIA

Returns the OIA object associated with this presentation space.

VB.NET

*[OIA =] Screen.***OIA**

C#

*[OIA =] Screen.***OIA**;

Remarks

This object can be used to query the status of the operator information area.

5.2.2.5.2.7 Rows

Indicates the number of rows in the presentation space.

VB.NET

*[Integer =] Screen.***Rows**

C#

*[int =] Screen.***Rows**;

Remarks

This property returns an integer representing the number of rows on the current host Screen.

5.2.2.5.2.8 String/Text

Returns the entire text plane of the virtual Screen as a string.

VB.NET

*[String =] Screen.***String**

C#

*[String =] Screen.***String**;

Remarks

All null characters and Field Attribute characters are returned as blank space

characters.
In Delphi, this property is called *Text* because *String* is a reserved word and can't be used as a function name.

5.2.2.6 Fields

Fields contains a collection of the fields in the virtual Screen. It provides methods to iterate through the fields, find fields based on location, and find fields containing a given string. Each element of the collection is an instance of Field.
Fields can only be accessed through Screen using the Fields property. Fields is a static view of the virtual Screen and does not reflect changes made to the virtual Screen after its construction. The field list can be updated with a new view of the virtual Screen using the Refresh() method.

Note: All Fields objects returned by methods in this class are invalidated when Refresh() is called.

See also Fields **Properties** and **Methods**.

5.2.2.6.1 Methods

- **Item**
- **FindByString**
- **FindByPosition**
- **Refresh**

5.2.2.6.1.1 Item

Indicates the Session object at the given index. "One based" indexing is used in all Ohio collections. For example, the first Session in this collection is at index 1.

VB.NET

*[Field =] Fields.***Item(**index As Integer**)**

C#

*[Field =] Fields.***Item(**int index**)**;

Parameters

Parameters	**Description**
index	The index of the target Session.

Returns null if no object with that name exists in this collection.

5.2.2.6.1.2 FindByPosition

Searches the collection for the target position and returns the Field object containing that position. If not found, returns a null.

VB.NET

[Field =] Fields.**FindByPosition(**targetPosition As Position**)**

C#

*[Field =] Fields.***FindByPosition(**Position targetString**)**;

Parameters

Parameters	**Description**
TargetPosition	The target row and column.

5.2.2.6.1.3 FindByString

Searches the collection for the target string and returns the Field object containing that string. The string
must be totally contained within the field to be considered a match. If the target string is not found, a null will be returned.

VB.NET

[Field =] Fields.**FindByString(**targetString As String, startPos As Position, length As Integer, dir As Integer, ignoreCase As Boolean**)**

C#

*[Field =] Fields.***FindByString(**string targetString, Position startPos, int length, int dir, bool ignoreCase**)**;

Parameters

Parameter	**Description**
targetString	The target string.
startPos	The row and column where to start. The position is inclusive (for example, row 1, col 1 means that position 1,1 will be used as the starting location and 1,1 will be included in the search).
length	The length from startPos to include in the search.
dir	An OHIO_DIRECTION value.
ignoreCase	Indicates whether the search is case sensitive. True means that case will be ignored. False means the search will be case sensitive.

5.2.2.6.1.4 Refresh

Updates the collection of Session objects. All Session objects that are available on the system at the
time of the refresh will be added to the collection.
Indexing of Session objects will not be preserved across refreshes.

VB.NET

*Fields.***Refresh**

C#

*Fields.***Refresh**;

Parameters

None.

5.2.2.6.2 Properties

- **Count**

5.2.2.6.2.1 Count

Returns the number of Session objects contained in this collection.

VB.NET

*[Integer =] Fields.***Count**

C#

*[int =] Fields.***Count**;

5.2.2.7 Field

A field is the fundamental element of a virtual Screen. A field includes both data and attributes describing the field. The Field class encapsulates a virtual Screen field and provides methods for accessing and manipulating field attributes and data.
Field objects can be accessed only through the Fields object.

See also Properties and **Methods**.

5.2.2.7.1 Methods

- **GetData**

5.2.2.7.1.1 GetData

Returns a character array containing the data from the Text, Color, Field or Extended plane of the virtual Screen.

VB.NET

*[Array =] Field.***GetData(**plane As Integer**)**

C#

*[char[] =] Field.***GetData(**int plane**)**;

Parameters

Parameter	**Description**
plane	A valid OHIO_PLANE value.

5.2.2.7.2 Properties

- **Attribute**
- **EndPos**
- **Hidden**
- **HighIntensity**
- **Length**
- **Modified**
- **Numeric**
- **PenSelectable**
- **ProtectedField**
- **StartPos**
- **String/Text**

5.2.2.7.2.1 Attribute

Indicates the attribute byte for the field.

VB.NET

*[Integer =] Field.***Attribute**

C#

*[int =] Field.***Attribute**;

5.2.2.7.2.2 EndPos

Returns the ending position of the field. The position can range from 1 to the size of the virtual Screen. The ending position of a field is the position of the last character in the field.

VB.NET

*[Position =] Field.***EndPos**

C#

*[Position =] Field.***EndPos**;

Remarks

Returns an integer indicating the position of the last character in the field.

5.2.2.7.2.3 Hidden

Indicates whether or not the field is hidden.

VB.NET

*[Boolean =] Field.***Hidden**

C#

*[bool =] Field.***Hidden**;

Remarks

This property returns True if the field is hidden, otherwise false.

5.2.2.7.2.4 HighIntensity

Indicates whether or not the field is high-intensity.

VB.NET

*[Boolean =] Field.***HighIntensity**

C#

*[bool =] Field.***HighIntensity**;

Remarks

Returns True if the field is high intensity, otherwise False.

5.2.2.7.2.5 Length

Returns the length of the field.

VB.NET

*[Integer =] Field.***Length**

C#

*[int =] Field.***Length**;

Remarks

A field's length can range from 1 to the size of the virtual Screen.

5.2.2.7.2.6 Modified

Indicates whether or not the field has been modified.

VB.NET

*[Boolean =] Field.***Modified**

C#

*[bool =] Field.***Modified**;

Remarks

This property returns True if the field has been modified, otherwise False.

5.2.2.7.2.7 Numeric

Indicates whether the field which contains the cursor is a numeric-only field.

VB.NET

*[Boolean =] Field.***Numeric**

C#

*[bool =] Field.***Numeric**;

Remarks

This property returns True if the cursor is in a numeric-only field, false otherwise.

5.2.2.7.2.8 PenSelectable

Indicates whether or not the field is pen-selectable.

VB.NET

*[Boolean =] Field.***PenSelectable**

C#

*[bool =] Field.***PenSelectable**;

Remarks

This property returns True if the field is pen-selectable, otherwise False.

5.2.2.7.2.9 ProtectedField

Indicates whether or not the field is protected.

VB.NET

*[Boolean =] Field.***Protected**

C#

*[bool =] Field.***Protected**;

Remarks

This property returns True if the field is protected, otherwise False.

5.2.2.7.2.10 StartPos

Indicates the starting position of the field.

VB.NET

*[Position =] Field.***StartPos**

C#

*[Position =] Field.***StartPos**;

Remarks

The position can range from 1 to the size of the virtual Screen. The starting position of a field is the position of the first character in the field.

5.2.2.7.2.11 String/Text

Returns the entire text plane of the virtual Screen as a string.

VB.NET

*Field.***String** *[= String]*

C#

*Field.***String**; *[= String]*

Remarks

All null characters and Field Attribute characters are returned as blank space characters.
In Delphi, this property is called *Text* because *String* is a reserved word and can't be used as a function name.

5.2.2.8 OIA

The operator information area of a host Session. This area is used to provide status information regarding the state of the host Session and location of the cursor.
An OIA object can be obtained using the OIA() method on an instance of Screen.

5.2.2.8.1 Properties

- **Alphanumeric**
- **CommCheckCode**
- **InputInhibited**
- **MachineCheckCode**
- **Numeric**
- **Owner**
- **ProgCheckCode**

5.2.2.8.1.1 AlphaNumeric

Indicates whether the field which contains the cursor is an alphanumeric field. True if the cursor is in an alphanumeric field, false otherwise.

VB.NET

*[Boolean=] OIA.***Alphanumeric**

C#

*[bool =] OIA.***Alphanumeric**;

Remarks

This property returns True if the cursor is in an alphanumeric field, False in the other case.

5.2.2.8.1.2 CommCheckCode

Returns the communication check code.

VB.NET

*[Integer =] OIA.***CommCheckCode**

C#

*[int =] OIA.***CommCheckCode**;

Remarks

If InputInhibited returns OHIO_INPUTINHIBITED_COMMCHECK, this property will return the communication check code.

5.2.2.8.1.3 InputInhibited

Indicates whether or not input is inhibited.

VB.NET

*[Integer =] OIA.***InputInhibited**

C#

*[int =] OIA.***InputInhibited**;

Remarks

If input is inhibited, SendKeys or SendAID calls to the Screen are not allowed. Why input is inhibited can be determined from the value returned. If input is inhibited for more than one reason, the highest value is returned.

5.2.2.8.1.4 MachineCheckCode

Specifies the machine check code.

VB.NET

*[Integer =] OIA.***MachineCheckCode**

C#

*[int =] OIA.***MachineCheckCode**;

Remarks

If InputInhibited returns OHIO_INPUTINHIBITED_MACHINECHECK, this property will return the machine check code.

5.2.2.8.1.5 Numeric

Indicates whether the field which contains the cursor is a numeric-only field.

VB.NET

*[Boolean =] OIA.***Numeric**

C#

*[bool =] OIA.***Numeric**;

Remarks

This property returns True if the cursor is in a numeric-only field, false otherwise.

5.2.2.8.1.6 Owner

Indicates the owner of the host connection.

VB.NET

*[Integer =] OIA.***Owner**

C#

*[int =] OIA.***Owner**;

Remarks

The Integer returned by this property specifies one of the **OHIO_OWNER** values.

5.2.2.8.1.7 ProgCheckCode

Returns the program check code.

VB.NET

*[Integer =] OIA.***ProgCheckCode**

C#

*[int =] OIA.***ProgCheckCode**;

Remarks

If InputInhibited returns OHIO_INPUTINHIBITED_PROGCHECK, this property will return the program check code.

5.2.2.9 Position

Holds row and column coordinates. An Position can be constructed by using **CreateOhioPosition**() on any **Ohio** class.

See also the Position **Properties**.

5.2.2.9.1 Properties

- **Col**
- **Pos**
- **Row**

5.2.2.9.1.1 Col

Indicates the column coordinate.

VB.NET

Position.**Col** *[= Integer]*

C#

Position.**Col**; *[= int]*

5.2.2.9.1.2 Pos

Gets or sets the current position.

VB.NET

Position.**Pos** *[= Integer]*

C#

Position.**Pos**; *[= int]*

Remarks

Value indicates the position to set to the **Position** object.

5.2.2.9.1.3 Row

Indicates the row coordinate.

VB.NET

Position.**Row** *[= Integer]*

C#

Position.**Row**; *[= int]*

5.2.3 Constants

- **Ohio Constants**
- **3270 Key Values**
- **5250 Key Values**

5.2.3.1 Ohio Constants

Constants for all Ohio classes.

OHIO_INFO

Constant	**Value**	**Description**
OHIO_VERSION_LEVEL	OHIO 1.00	The OHIO version level of this implementation. The form is "OHIO nn.nn"
OHIO_VENDOR_NAME	Cybele Software Inc.	The name of the vendor providing this OHIO implementation. Format is vendor defined.

OHIO_VENDOR_PRODUCT_VERSION	TN Bridge Integration Pack 3.0	The vendor product version that is providing the OHIO implementation. Format is vendor specific.

uTnbOHIO_Session cfg keys

Constant	**Value**	**Description**
OHIO_Session_TYPE	OHIO_Session_TYPE=	The OHIO Session Type: unknown host, 3270 or 5250 host.
OHIO_Session_HOST	OHIO_Session_HOST=	The OHIO Session Host.
OHIO_Session_PORT	OHIO_Session_PORT=	The OHIO Session Port.
OHIO_Session_NAME	OHIO_Session_NAME=	The OHIO Session Name.
OHIO_Session_CONFIG_RESOURCE	OHIO_Session_CONFIG_RESOURCE=	The OHIO Session Configuration Resource.
OHIO_Session_TN_ENHANCED	OHIO_Session_TN_ENHANCED=	
OHIO_Session_Screen_SIZE	OHIO_Session_Screen_SIZE=	The Screen Size for the OHIO Session.
OHIO_Session_CODE_PAGE	OHIO_Session_CODE_PAGE=	The Code Page for the OHIO Session.

OHIO_DIRECTION

Constant	**Value**	**Description**
OHIO_DIRECTION_FORWARD	0	Forward (beginning towards end)
OHIO_DIRECTION_BACKWARD	1	Backward (end towards beginning)

OHIO_TYPE

Constant	**Value**	**Description**
OHIO_TYPE_UNKNOWN	0	Unknown Host
OHIO_TYPE_3270	1	3270 Host
OHIO_TYPE_5250	2	5250 Host

OHIO_STATE

Constant	**Value**	**Description**
OHIO_STATE_DISCONNECTED	0	The communication link to the host is disconnected
OHIO_STATE_CONNECTED	1	The communication link to the host is connected

OHIO_PLANE

Constant	**Value**	**Description**
OHIO_PLANE_TEXT	1	Indicates Plane Text (character data)

OHIO_PLANE_COLOR	2	Indicates Plane Color (standard HLLAPI CGA color values)
OHIO_PLANE_FIELD	4	Indicates Field Attribute Plane (field attribute bytes)
OHIO_PLANE_EXTENDED	8	Indicates Extended Plane (extended attribute bytes)

OHIO_COLOR

Constant	Value	Description
OHIO_COLOR_BLACK	0	Black
OHIO_COLOR_BLUE	1	Blue
OHIO_COLOR_GREEN	2	Green
OHIO_COLOR_CYAN	3	Cyan
OHIO_COLOR_RED	4	Red
OHIO_COLOR_MAGENTA	5	Magenta
OHIO_COLOR_YELLOW	6	Yellow
OHIO_COLOR_WHITE	7	White

OHIO_EXTENDED

Constant	Value	Description
OHIO_EXTENDED_HILITE	$C0	Bitmask for Highlighting Bits
OHIO_EXTENDED_COLOR	$38	Bitmask for Color Bits
OHIO_EXTENDED_RESERVED	$07	Bitmask for Reserved Bits
OHIO_EXTENDED_HILITE_NORMAL	$00	Normal highlighting
OHIO_EXTENDED_HILITE_BLINK	$40	Blinking highlighting
OHIO_EXTENDED_HILITE_REVERSEVIDEO	$80	Reverse video highlighting
OHIO_EXTENDED_HILITE_UNDERSCORE	$C0	Under score highlighting
OHIO_EXTENDED_COLOR_DEFAULT	$00	Default color
OHIO_EXTENDED_COLOR_BLUE	$08	Blue
OHIO_EXTENDED_COLOR_RED	$10	Red
OHIO_EXTENDED_COLOR_PINK	$18	Pink
OHIO_EXTENDED_COLOR_GREEN	$20	Green
OHIO_EXTENDED_COLOR_TURQUOISE	$28	Turquoise
OHIO_EXTENDED_COLOR_YELLOW	$30	Yellow
OHIO_EXTENDED_COLOR_WHITE	$38	White
OHIO_EXTENDED_COLUMN_SEPARATOR	$02	Separator

OHIO_FIELD

Constant	Value	Description
OHIO_FIELD_ATTRIBUTE	$C0	Bitmask for field attribute

OHIO_FIELD_PROTECTED	$20	Protected field
OHIO_FIELD_NUMERIC	$10	Numeric field
OHIO_FIELD_HIDDEN	$0C	Hidden field
OHIO_FIELD_PEN_SELECTABLE	$08	Pen selectable field
OHIO_FIELD_HIGH_INTENSITY	$04	High Intensity field
OHIO_FIELD_RESERVED	$02	Reserved field
OHIO_FIELD_MODIFIED	$01	Modified field

OHIO_UPDATE

Constant	Value	Description
OHIO_UPDATE_CLIENT	0	Update initiated by client
OHIO_UPDATE_HOST	1	Update initiated by host

OHIO_OWNER

Constant	Value	Description
OHIO_OWNER_UNKNOWN	0	Unitialized
OHIO_OWNER_APP	1	Application or 5250 host
OHIO_OWNER_MYJOB	2	3270 - Myjob
OHIO_OWNER_NVT	3	NVT (Network Virtual Terminal) mode
OHIO_OWNER_UNOWNED	4	Unowned (3270 only)
OHIO_OWNER_SSCP	5	SSCP (3270 only)

OHIO_INPUTINHIBITED

Constant	Value	Description
OHIO_INPUTINHIBITED_NOTINHIBITED	0	Input not inhibited
OHIO_INPUTINHIBITED_SYSTEM_WAIT	1	Input inhibited by a System Wait state ("X SYSTEM" or "X []")
OHIO_INPUTINHIBITED_COMMCHECK	2	
OHIO_INPUTINHIBITED_PROGCHECK	3	
OHIO_INPUTINHIBITED_MACHINECHECK	4	
OHIO_INPUTINHIBITED_OTHER	5	Input inhibited by something other than above states

uTnbOHIO_Session_STATE

Constant	Value	Description
OHIO_Session_STATE_NOSession	0	No Session established
OHIO_Session_STATE_SSCPLU	1	SSCPLU Session state
OHIO_Session_STATE_LULU	2	LULU Session state

uTnbOHIO_EXCEPTION

Constant	Value
OHIO_EXCEPTION_SESSIONALREADYEXISTS	Already exists a session with that name
OHIO_EXCEPTION_BADCFGFILE	Error in configuration file
OHIO_EXCEPTION_BADHOSTNAME	There is no data provided in '+OHIO_SESSION_HOST+' configuration field
OHIO_EXCEPTION_BADPORTNUMBER	Data provided in '+OHIO_SESSION_PORT+' configuration field is not a number
OHIO_EXCEPTION_INVALIDPORTNUMBER	Data provided in '+OHIO_SESSION_PORT+' configuration field is out of range
OHIO_EXCEPTION_TOOMANYSESSIONS	Too many sessions

5.2.3.2 3270 Key Values

Aid Key Constants	Value
OHIO_AID_KEY_ATTN	$0182
OHIO_AID_KEY_BACKSPACE	$0183
OHIO_AID_KEY_BACKTAB	$0184
OHIO_AID_KEY_CLEAR	$0186
OHIO_AID_KEY_CURSORSEL	$0188
OHIO_AID_KEY_DELETE	$0189
OHIO_AID_KEY_DOWN	$018B
OHIO_AID_KEY_DUP	$018C
OHIO_AID_KEY_ENTER	$018D
OHIO_AID_KEY_ERASEEOF	$018E
OHIO_AID_KEY_ERASEINPUT	$018F
OHIO_AID_KEY_HOME	$0190
OHIO_AID_KEY_LEFT	$0193
OHIO_AID_KEY_MARK	$0194
OHIO_AID_KEY_NEWLINE	$0195
OHIO_AID_KEY_PA1	$0196
OHIO_AID_KEY_PA2	$0197
OHIO_AID_KEY_PA3	$0198
OHIO_AID_KEY_PF1	$01A0
OHIO_AID_KEY_PF2	$01A1
OHIO_AID_KEY_PF3	$01A2
OHIO_AID_KEY_PF4	$01A3
OHIO_AID_KEY_PF5	$01A4
OHIO_AID_KEY_PF6	$01A5
OHIO_AID_KEY_PF7	$01A6
OHIO_AID_KEY_PF8	$01A7
OHIO_AID_KEY_PF9	$01A8
OHIO_AID_KEY_PF10	$01A9
OHIO_AID_KEY_PF11	$01AA
OHIO_AID_KEY_PF12	$01AB
OHIO_AID_KEY_PF13	$01AC
OHIO_AID_KEY_PF14	$01AD
OHIO_AID_KEY_PF15	$01AE

OHIO_AID_KEY_PF16	$01AF
OHIO_AID_KEY_PF17	$01B0
OHIO_AID_KEY_PF18	$01B1
OHIO_AID_KEY_PF19	$01B2
OHIO_AID_KEY_PF20	$01B3
OHIO_AID_KEY_PF21	$01B4
OHIO_AID_KEY_PF22	$01B5
OHIO_AID_KEY_PF23	$01B6
OHIO_AID_KEY_PF24	$01B7
OHIO_AID_KEY_RESET	$019A
OHIO_AID_KEY_RIGHT	$019B
OHIO_AID_KEY_SYSREQ	$019C
OHIO_AID_KEY_TABFORWARD	$019D
OHIO_AID_KEY_UP	$019F
OHIO_AID_KEY_PRINT	$01C0
OHIO_AID_KEY_PGUP	$01C1
OHIO_AID_KEY_PGDOWN	$01C2

5.2.3.3 5250 Key Values

Aid Key Constants	Value
OHIO_AID_KEY_ATTN	$0182
OHIO_AID_KEY_BACKSPACE	$0183
OHIO_AID_KEY_BACKTAB	$0184
OHIO_AID_KEY_CLEAR	$0186
OHIO_AID_KEY_CURSORSEL	$0188
OHIO_AID_KEY_DELETE	$0189
OHIO_AID_KEY_DOWN	$018B
OHIO_AID_KEY_DUP	$018C
OHIO_AID_KEY_ENTER	$018D
OHIO_AID_KEY_ERASEEOF	$018E
OHIO_AID_KEY_ERASEINPUT	$018F
OHIO_AID_KEY_HOME	$0190
OHIO_AID_KEY_LEFT	$0193
OHIO_AID_KEY_MARK	$0194
OHIO_AID_KEY_NEWLINE	$0195
OHIO_AID_KEY_PA1	$0196
OHIO_AID_KEY_PA2	$0197
OHIO_AID_KEY_PA3	$0198
OHIO_AID_KEY_PF1	$01A0
OHIO_AID_KEY_PF2	$01A1
OHIO_AID_KEY_PF3	$01A2
OHIO_AID_KEY_PF4	$01A3
OHIO_AID_KEY_PF5	$01A4
OHIO_AID_KEY_PF6	$01A5
OHIO_AID_KEY_PF7	$01A6
OHIO_AID_KEY_PF8	$01A7
OHIO_AID_KEY_PF9	$01A8
OHIO_AID_KEY_PF10	$01A9
OHIO_AID_KEY_PF11	$01AA

6 XML Reference

6.1 Introduction

XML (e**X**tensible **M**arkup **L**anguage) is a framework for defining markup languages:

- There is no fixed collection of markup tags - you may define your own tags, tailored for your kind of information.
- Each XML language is targeted at its own application domain, but the languages will share many features.
- There is a common set of generic tools for processing documents.

XML is designed to:

- Separate syntax from semantics to provide a common framework for structuring information (browser rendering semantics is completely defined by stylesheets).
- Allow tailor-made markup for any imaginable application domain.
- Support internationalization (Unicode) and platform independence.
- Be the future of structured information, including databases.

6.2 Programming Reference

Just like in previous versions of TN Bridge, when working with profiles, you can operate with a single or several files. Main difference in this version is that subkeys are fixed. When you set the StreamingType property of the TNBProfiles component to *XML* value, subkeys will be predetermined tags, and the Key will be **TNBRIDGE** tag, that is the root.

The following is the tags' hierarchy we will use:

```
<TNBRIDGE>
    <Connections>
        <Connection>
            <Host ... />
            <Display ... />
        <Connection>
        .
        .
        .
    <Connections />
    <ScreenStyles>
        <ScreenStyle>
            <General>
                <Font ... />
            <General />
            <FieldMapping>
                <FieldMap>
                    <Input ... />
                    <Output ... />
                <FieldMap />
                .
                .
                .
            <FieldMapping />
        <ScreenStyle />
        .
        .
        .
    <ScreenStyles />
    <KeyboardMaps>
        <KeyboardMap>
            <KeyMap>
                <FunctionKey ... />
                <MappedKey ... />
            <KeyMap />
            .
            .
            .
        <KeyboardMap />
        .
        .
        .
    <KeyboardMaps>
      <HotSpots>
        <HotSpotsGroup>
            <HotSpot ... />
        <HotSpotsGroup />
        .
        .
        .
      <HotSpots />
      <TnbMacros>
        <Macro>
            <Action ... />
          <Macro />
          .
          .
          .
      <TnbMacros />
<TNBRIDGE />
```

6.2.1 XML Tags and Attributes

6.2.1.1 TNBRIDGE

This is the root tag and includes all the other tags. These are the following:

- **Connections**: contains a collection of Connection tags.
- **ScreenStyles**: contains a collection of ScreenStyle tags.
- **KeyboardMaps**: contains a collection of KeyboardMap tags.
- **HotSpots**: contains a collection of HotSpot tags.
- **TnbMacros**: contains a collection of Macro tags.

6.2.1.2 Connections

Contains a collection of **Connection** tags. Each **Connection** represents a particular connection and has the following tags:

- **Host:** specifies Host attributes.
- **Display:** specifies display options for the current connection.

Attributes

Connections MRUName

Indicates last connection name used.

Syntax Example:

<Connections **MRUName**="Clemson"/>

Connection Name

Indicates connection name.

Syntax Example:

<Connection **Name**="Clemson"/>

Connection Type

Indicates the type of the target system. Possible selection values are:

- IBM Mainframe (TN3270)
- IBM AS/400 (TN5250)

Syntax Example

<Connection **Type**="TN3270"/>

6.2.1.3 ScreenStyles

Contains a collection of **ScreenStyle** tags. Each **ScreenStyle** represents a particular screen style and has the following tags:

- **General**: contains general screen settings.
- **FieldMapping**: contains a collection of Fieldmaps.

Attributes

ScreenStyles MRUName

Indicates last screen style name used.

Syntax Example:

<ScreenStyles **MRUName**="Green"/>

ScreenStyle Name

Indicates the screen style name.

Syntax Example:

<ScreenStyle **Name**="Monocromatic">

6.2.1.4 KeyboardMaps

Contains a collection of **KeyboardMap** tags. Each **KeyboardMap** represents a particular keyboardmap and contains a collection of KeyMaps.

Attributes

KeyboardMaps MRUName

Indicates last keyboardmaps name used.

Syntax Example:

<KeyboardMaps **MRUName**="Default++"/>

KeyboardMap Name

Specifies the keyboard map's name.

Syntax Example:

<KeyboardMap **Name**="Default++"/>

KeyboardMap Type

Indicates the keyboard map type.

Syntax Example:

<KeyboardMap **Type**="3270"/>

6.2.1.5 HotSpots

Contains a collection of **HotSpotGroup** tags. Each **HotSpotGroup** contains a group of HotSpots.

Attributes

HotSpots MRUName

Indicates last hotspots name used.

Syntax Example:

<HotSpots **MRUName**="Clemson"/>

HotSpotGroup Name

Specifies the HotSpotGroup's name.

Syntax Example:

<HotspotGroup **Name**="Clemson"/>

HotSpotGroup StartCol

Specifies the HotSpotGroup's start column number coordinate.

Syntax Example:

<HotspotGroup **StartCol**="0"/>

HotSpotGroup StartRow

Specifies the HotSpotGroup's start row number coordinate.

Syntax Example:

<HotspotGroup **StartRow**="0"/>

HotSpotGroup EndCol

Specifies the HotSpotGroup's end column number coordinate.

Syntax Example:

<HotspotGroup **EndCol**="0"/>

HotSpotGroup EndRow

Specifies the HotSpotGroup's end row number coordinate.

Syntax Example:

<HotspotGroup **EndRow**="0"/>

HotSpotGroup ViewAs

Indicates the HotSpotGroup's shape. This shape can be: None, Plain, Button or Link.

Syntax Example:

<HotspotGroup **ViewAs**=""/>

6.2.1.6 TnbMacros

Contains a collection of **Macro** tags. Each **Macro** contains a set of **Actions** that indicates what the macro does.

Attributes

Macro Version

Indicates macro's version.

Syntax Example:

<macro **version**="1"/>

Macro Name

Specifies the Macro's name.

Syntax Example:

<macro **Name**="Save"/>

Macro DateTime

Specifies the date and time the macro was created.

Syntax Example:

<macro **datetime**="17/12/2004 15:19:23"/>

6.2.2 XML Example

Here you can see an example of a profile generated with XML Code:

```
<?xml version="1.0" ?>
<TNBRIDGE>
     <Connections MRUName="Clemson">
            <Connection Name="Clemson" Type="TN3270">
                   <Host Address="clemson.clemson.edu" KeepAlive="False"
Extended="False" />
                   <Display Terminal="IBM-3278-2-E" />
            </Connection>
            .
            .
            .
     </Connections>
     <ScreenStyles MRUName="Green">
            <ScreenStyle Name="Monocromatic">
                   <General>
                          <Font Auto="0" Size="9" ChSpacing="0" Name="Fixedsys" />
                   </General>
                   <FieldMapping>
                          <FieldMap>
                                 <Input High="True" Unprotected="True" />
                                 <Output Color="White" />
                          </FieldMap>
                          .
                          .
                          .
                   </FieldMapping>
            </ScreenStyle>
            .
            .
            .
     </ScreenStyles>
     <KeyboardMaps MRUName="Default++">
            <KeyboardMap Name="Default++" Type="3270">
                   <KeyMap>
                          <FunctionKey Name="Clear" />
                          <MappedKey LShift="0" RShift="0" LControl="0" RControl="0"
LMenu="0" RMenu="0" NumLock="0" ScanCode="69" />
                   </KeyMap>
                   .
                   .
                   .
            </KeyboardMap>
            .
            .
            .
     </KeyboardMaps>
     <Hotspots MRUName="Clemson">
            <HotspotGroup Name="Clemson" StartRow="0" StartCol="0" EndRow="0"
EndCol="0" ViewAs="">
                   <Hotspot Name="Clemson University" Id="{2000374A-DAF7-498A-9E6D-
FCB10E98E671}" StartRow="1" StartCol="32" EndRow="2" EndCol="50" ViewAs="Button"
ActionFieldData="@E" Pattern="Clemson University" FgColor="Navy" BgColor="Black" />
                   .
                   .
                   .
            </HotspotGroup>
            .
            .
            .
     </Hotspots>
</TNBRIDGE>
```

7 Purchasing TN Bridge Host Integration Pack

By purchasing any edition of TN Bridge Host Integration Pack you will gain access to technical support, free upgrades and updates and the activation of advanced features in your edition.

In this section you will find information regarding the different licensing options you have that will help you choose the type of order you need to place. Also this section explains how to place your order and finally activate your product so that you can enjoy all of the z/Scope benefits.

- Licensing Information
- How to Place an Order
- TN Bridge Registration
- Technical Support

7.1 Licensing Information

When it comes to purchasing TN Bridge Host Integration Pack, there are different editions available. Our wide range of possibilities assures you that you will make the best deal.

- TN Bridge Host Integration Pack for DotNET
- TN Bridge Host Integration Pack for ActiveX *
- TN Bridge Host Integration Pack for Delphi *

**Find the help file for TN Bridge Host Integration Pack for Delphi and ActiveX here: http://www.cybelesoft.com/helps/tnb/index.html*

Cybele Software offers perpetual licensing for all our products. The pricing will vary according to the TN Bridge Host Integration Pack package you choose (Developer/Team) and the amount of licenses.

We offer Technical Support by e-mail and/or phone, which also includes free updates and upgrades during the covered period and our full commitment to timely fix bugs and problems. The cost is 20% of the product's value in advance, only mandatory for the first year. Thereafter, we encourage users to renew the annual maintenance contract in order to be eligible for technical support and product upgrades. The maintenance fee after the first year will still be 20% of the updated price of the purchased product.

Cybele Software offers volume pricing according to the amount of the purchase.

If you have any other question, contact us at sales@cybelesoft.com. Our sales representatives will get in touch with you to assist you with your licensing situation.

7.2 How to Place an Order

There are many ways to order your Tn Bridge licenses:

- Place an Online Order through our Web Site:

  http://www.cybelesoft.com/buy/

- Let us know about your licensing needs and we will send you an official quotation. Fill out the form in the following link:

  http://www.cybelesoft.com/contact/

- Contact us at sales@cybelesoft.com. Our sales representatives will get in touch with you to assist you with the purchase.

- You can also call us anytime to any of these phone numbers and place the order immediately:

  Toll Free: 1-866-462-9768
  Local line: 1-302-892-9625
  Fax: 1-302-295-9995

- You can also contact us through Live Chat by pressing this icon in our website:

and immediately have a conversation with a representative without even having to pick up the phone.

There are several payment options, and we also accept Purchase Orders.

When you buy TN Bridge, you will receive a Key to register the Trial version. For instructions on how to register TN Bridge when you purchase a license, see Registering TN Bridge Trial Version

7.3 How to register your License

Choose your licensing model and continue with one of the following topics:

- Register Developer and Server License
- Register Developer and Runtime License

7.3.1 Register Developer and Server License

If you downloaded TN Bridge Host Integration Pack's Trial Version from our web site or a distribution site and you have already purchased a license, you must follow these steps in order to register the product:

You received two registered license files:
Cybele.TnBridge.Design.xml
Cybele.TnBridge.Server.xml

Important: If you used a trial license, erase all the old files before entering the new one, to prevent license corruption.

How to register your TNBridge Host Integration Pack license:

1) The developer license file (Cybele.TnBridge.Design.xml) must be saved into the "windows" directory and where the
Cybele.TnBridge.Vcl.dll assembly is located. Do not change the file name!

2) Additionally, the server license (Cybele.TnBridge.Server.xml) must be set by code passing the whole xml string as
parameter of the Cybele.TNBridge.License.SetRuntimeKey method.

*** C# example on how to embed the runtime code:**

- Example 1, point to the runtime file:
  Cybele.TNBridge.License.SetRuntimeKeyFromFile(@"C:\Windows
\Cybele.TNBridge.Server.xml");

- Example 2, embed the code like this (replace with your own code, this is just an example):
  Cybele.TNBridge.License.SetRuntimeKey("<License Type=\"SERVER\" Serial=\"TNBN-TR35-0505-070315-797313\">"+
  "<User Name=\"Your User\" Company=\"Your Company\" Email=\"your@email.com\" />"+
  "<Period First=\"03/15/2007\" Last=\"04/14/2007\" />"+
  "<Version Low=\"3.5\" High=\"3.5\" />"+
  "<Limitations Developers="0" Servers="1" Users="0" Connections="20" />"+
  "<Signature xmlns=\"http://www.w3.org/2000/09/xmldsig#\">"+
  " <SignedInfo>"+
  " <CanonicalizationMethod Algorithm=\"http://www.w3.org/TR/2001/REC-xml-c14n-20010315\" />"+
  " <SignatureMethod Algorithm=\"http://www.w3.org/2000/09/xmldsig#rsa-sha1\" />"+
  " <Reference URI=\"\">" +
  " <Transforms>"+
  " <Transform Algorithm=\"http://www.w3.org/2000/09/xmldsig#enveloped-signature\" />"+

```
" </Transforms>"+
" <DigestMethod Algorithm=\"http://www.w3.org/2000/09/xmldsig#sha1\" />"+
" <DigestValue>gBffCpUytqZL7qEfZYBsPOQHx+w=</DigestValue>"+
" </Reference>"+
" </SignedInfo>"+
"
<SignatureValue>zoXPEIxrtzbY
+aRlaJx375iGrOwdUNFrAC4IVsSAAq00dxqi23U4EGKV048atowk39PbElHaHB/
EeeXivdFdbcCsNV1YjAze
xAWGHJivAw0uEXRJg5wjyX7QR0i7ZsA51SaSRp26IofgGl+OLOJ/DCoHE
+n28e9SAAAA2LDugI=</SignatureValue>"+
"</Signature>"+
"</License>");
```

*** VB.NET example on how to embed the runtime code:**

- Example 1, point to the runtime file:

```
Cybele.TNBridge.License.SetRuntimeKeyFromFile("C:\Windows
\Cybele.TNBridge.Server.xml")
```

- Example 2, embed the code like this (replace with your own code, this is just an example):

```
Cybele.TNBridge.License.SetRuntimeKey("<License Type=""SERVER"" Serial=""TNBN-
TR35-0505-070315-797313"">" & vbNewLine
& _
"<User Name=""Your User"" Company=""Your Company"" Email=""your@email.com"" /
>" & vbNewLine & _
"<Period First=""03/15/2007"" Las t=""04/14/2007"" />" & vbNewLine & _
"<Version Low=""3.5"" High=""3.5"" />" & vbNewLine & _
"<Limitations Developers="0" Servers="1" Users="0" Connections="20" />" &
vbNewLine & _
"<Signature xmlns=""http://www.w3.org/2000/09/xmldsig#"">" & vbNewLine & _
" <SignedInfo>" & vbNewLine & _
" <CanonicalizationMethod Algorithm=""http://www.w3.org/TR/2001/REC -xml-
c14n-20010315"" />" & vbNewLine & _
" <SignatureMethod Algorithm=""http://www.w3.org/2000/09/xmldsig#rsa -sha1"" /
>" & vbNewLine & _
" <Reference URI="""">" & vbNewLine & _
" <Transforms>" & vbNewLine & _
" <Transform Algorithm=""http://www.w3.org/2000/09/xmldsig#enveloped-
signature"" />" & vbNewLine & _
" </Transforms>" & vbNewLine & _
" <DigestMethod Algorithm=""http://www.w3.org/2000/09/xmldsig#sha1"" />" &
vbNewLine & _
" <DigestValue>gBffCpUytqZL7qEfZYBsPOQHx+w=</DigestValue>" & vbNewLine & _
" </Reference>" & vbNewLine & _
" </SignedInfo>" & vbNewLine & _
"
<SignatureValue>zoXPEIxrtzbY
+aRlaJx375iGrOwdUNFrAC4IVAgASASAqUeexqi23U4EGKV048atowk39PbElHaHB/
EeeXivdFdbcCsNV1Yj
AzexAWGHJivAw0uEXRJg5wjyX7QR0i7ZsA51SaSRp26IofgGl+OLOJ/DCoHaAAE
+n28e9A2LDugAAAI=</SignatureValue>" & vbNewLine
& _
"</Signature>" & vbNewLine & _
"</License>")
```

7.3.2 Register Developer and Runtime License

If you downloaded TN Bridge Host Integration Pack's Trial Version from our web site or a distribution site and you have already purchased a license, you must follow these steps in order to register the product:

You received two registered license files:
Cybele.TnBridge.Design.xml
Cybele.TnBridge.Runtime.xml

Important: If you used a trial license, erase all the old files before entering the new one, to prevent license corruption.

How to register your TNBridge Host Integration Pack license:

1) The developer license file (Cybele.TnBridge.Design.xml) must be saved into the "windows" directory and
where the Cybele.TnBridge.Vcl.dll assembly is located.

2) Additionally, the runtime license must be set by code passing the whole xml string as parameter of the
Cybele.TNBridge.License.SetRuntimeKey method.

*** C# example on how to embed the runtime code:**

- Example 1, point to the runtime file:

```
Cybele.TNBridge.License.SetRuntimeKeyFromFile(@"C:\Windows
\Cybele.TNBridge.Runtime.xml");
```

- Example 2, embed the code like this (replace with your own code, this is just an example):

```
Cybele.TNBridge.License.SetRuntimeKey("<License Type=\"RUNTIME\" Serial=
\"TNBN-TR35-0505-070315-797313\">"+
"<User Name=\"Your User\" Company=\"Your Company\" Email=\"your@email.com\" /
>"+
"<Period First=\"03/15/2007\" Last=\"04/14/2007\" />"+
"<Version Low=\"3.5\" High=\"3.5\" />"+
"<Limitations Users=\"5\" Connections=\"5\" />"+
"<Signature xmlns=\"http://www.w3.org/2000/09/xmldsig#\">"+
" <SignedInfo>"+
" <CanonicalizationMethod Algorithm=\"http://www.w3.org/TR/2001/REC-xml-c14n-
20010315\" />"+
" <SignatureMethod Algorithm=\"http://www.w3.org/2000/09/xmldsig#rsa-sha1\" /
>"+
" <Reference URI=\"\">" +
" <Transforms>"+
" <Transform Algorithm=\"http://www.w3.org/2000/09/xmldsig#enveloped-
signature\" />"+
" </Transforms>"+
" <DigestMethod Algorithm=\"http://www.w3.org/2000/09/xmldsig#sha1\" />"+
" <DigestValue>gBffCpUytqZL7qEfZYBsPOQHx+w=</DigestValue>"+
" </Reference>"+
" </SignedInfo>"+
"
<SignatureValue>zoXPEIxrtzbY
+aRlaJx375iGrOwdUNFrAC4IVsSAAq00dxqi23U4EGKV048atowk39PbElHaHB/
EeeXivdFdbcCsNV1YjAze
xAWGHJivAw0uEXRJg5wjyX7QR0i7ZsA51SaSRp26IofgGl+OLOJ/DCoHE
+n28e9SAAAA2LDugI=</SignatureValue>"+
"</Signature>"+
"</License>");
```

*** VB.NET example on how to embed the runtime code:**

- Example 1, point to the runtime file:

```
Cybele.TNBridge.License.SetRuntimeKeyFromFile("C:\Windows
\Cybele.TNBridge.Runtime.xml")
```

- Example 2, embed the code like this (replace with your own code, this is just an example):

```
Cybele.TNBridge.License.SetRuntimeKey("<License Type=""RUNTIME""
Serial=""TNBN-TR35-0505-070315-797313"">" &
vbNewLine & _
"<User Name=""Your User"" Company=""Your Company"" Email=""your@email.com"" /
>" & vbNewLine & _
"<Period First=""03/15/2007"" Last=""04/14/2007"" />" & vbNewLine & _
"<Version Low=""3.5"" High=""3.5"" />" & vbNewLine & _
"<Limitations Users=""5"" Connections=""5"" />" & vbNewLine & _
"<Signature xmlns=""http://www.w3.org/2000/09/xmldsig#"">" & vbNewLine & _
" <SignedInfo>" & vbNewLine & _
" <CanonicalizationMethod Algorithm=""http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"" />" & vbNewLine & _
" <SignatureMethod Algorithm=""http://www.w3.org/2000/09/xmldsig#rsa-sha1"" /
>" & vbNewLine & _
" <Reference URI="""">" & vbNewLine & _
" <Transforms>" & vbNewLine & _
" <Transform Algorithm=""http://www.w3.org/2000/09/xmldsig#enveloped-
signature"" />" & vbNewLine & _
" </Transforms>" & vbNewLine & _
" <DigestMethod Algorithm=""http://www.w3.org/2000/09/xmldsig#sha1"" />" &
vbNewLine & _
" <DigestValue>gBffCpUytqZL7qEfZYBsPOQHx+w=</DigestValue>" & vbNewLine & _
" </Reference>" & vbNewLine & _
" </SignedInfo>" & vbNewLine & _
"
<SignatureValue>zoXPEIxrtzbY
+aRlaJx375iGrOwdUNFrAC4IVAgASASAqUeexqi23U4EGKV048atowk39PbElHaHB/
EeeXivdFdbcCsNV1Yj
AzexAWGHJivAw0uEXRJg5wjyX7QR0i7ZsA51SaSRp26IofgGl+OLOJ/DCoHaAAE
+n28e9A2LDugAAAI=</SignatureValue>" & vbNewLine
& _
"</Signature>" & vbNewLine & _
"</License>")
```

7.4 Obtaining Technical Support

Cybele's goal is to offer high quality products and services to increase the efficiency and ease-of-use of legacy systems. The whole Company focuses on this goal, and the results of our unique expertise are our reliable solutions. We believe passionately that modern, solid and feature-rich host access solutions can actually increase its users' productivity.

Technical support is a very important benefit to consider, especially when it comes to mission critical software solutions.

Using registered Cybele Software's applications not only allows you to receive free product upgrades and updates but also the certainty that you will have our team of experienced developers and technical support representatives working hard to assist you with any issue, thus making the product much more accessible in any situation.

We are here to help you out from monday to friday 9 a.m. to 5 p.m. eastern time on the phone numbers:

Toll Free: 1-866-462-9768
Local line: 1-302-892-9625
Fax: 1-302-295-9995

If you make your call outside this hour range, you can leave a message and we will get back to you.

You can send us an email to support@cybelesoft.com and we will write you back timely. You can also contact us through Live Chat by pressing this icon in our website:

and immediately have a conversation with a representative without even having to pick up the phone.

Cybele Software Inc.
3422 Old Capitol Trail, suite 1125
Wilmington, DE - 19808
Phone: (302) 892-9625
Fax: (302) 295-9995
e-mail: support@cybelesoft.com
http://www.cybelesoft.com